A Fair Neural-Based Loss Reduction Technique for Queuing Systems with Self-Similar Characteristics

Homayoun Yousefi'zadeh Edmond A. Jonckheere John A. Silvester

***Abstract*— Reducing packet loss and increasing overall efficiency in queuing systems is one of the most important issues in the design of traffic control algorithms. On the other hand, the major issue in multiple source queuing systems is to provide individual sources with the ability to take advantage of a fair portion of shared available resources such as buffer space or server bandwidth. In this paper a novel technique for reducing packet loss in a class of queuing systems accommodating self-similar traffic patterns is introduced. The technique takes advantage of the modeling power of neural networks to offer a dynamic buffer management scheme capable of efficiently solving the trade off between loss rate and fairness issues.**

***Index Terms*— Perceptron Neural Networks, Packet Network, Bursty Traffic, Self-Similarity, Traffic Modeling, Buffer Management, Server Scheduling, Static Partial Sharing, Dynamic Neural Sharing.**

I. Introduction

TELETRAFFIC analysis of computer communication networks is one of the most important applications of mathematical modeling and queuing theory. This is mostly because of the widespread deployment of packet switching, specifically, services from Ethernet LANs, Variable Bit Rate (VBR) video, ATM, and ISDN. Modeling of bursty traffic patterns is among the most challenging problems in teletraffic analysis. Although, numerous models of packet arrival processes were proposed by Ramaswami et al. [23], Hellstern et al. [13], Sriram et al. [24], Heffes et al. [12], it seems that there is still a number of packet traffic features not being understood perfectly. This is partly due to uncertainties in the traffic characteristics and to the difficulties in characterizing traffic arrival models. Adas [1] provided a survey of different teletraffic models in his paper.

Analysis of traffic data from networks and services such as Ethernet LANs [17], Variable Bit Rate (VBR) video [3], ISDN traffic [13], and Common Channel Signaling Network (CCNS) [5] have all convincingly demonstrated the presence of features such as long range dependence, slowly decaying variances, and heavy-tailed distributions. These features are best described within the context of second-order self-similarity and fractal theory approach. Leland and Wilson [17] presented a statistical analysis of Ethernet traffic, in presence of "burstiness" across a wide range of time scales [17] in which traffic spikes ride on longer term ripples, that in turn ride on longer term swells, so on and so forth. This phenomenon is explained in terms of self-similarity, i.e., self-similar phenomena show structural similarities across all or a wide range of time scales. This burst within burst structure not only captures fractal properties observed in actual traffic data but also explains why measurements show no actual burst length for packet traffic patterns despite prediction of conventional models.

Neural networks are a class of nonlinear systems capable of learning and performing tasks accomplished by other systems. Their broad range of applications includes speech and signal processing, pattern recognition, system modeling, and servo mechanism control. Various kinds of neural networks, generally, have energy functions. The learning procedure of neural networks is, indeed, nothing more than decreasing these energy functions until reaching local minimum levels. Neural networks acquire required information from the examples supplied to them in their learning procedure. Systems with neural network building blocks are robust in the sense that the occurrence of small errors in the systems does not interfere with the proper operation of the system. This characteristic of the neural networks makes them quite suitable for traffic modeling.

Reducing packet loss in queuing systems is one of the most important issues in the design of traffic control algorithms. Reducing packet loss in the queuing systems is equivalent to improving efficiency and is usually considered as a performance evaluation tool. For the systems consisting of more than one source, there is another major issue worth considering known as fairness. Fairness provides each individual source with the ability to take advantage of a fair portion of the shared available resources such as buffer space or server bandwidth. The combination of buffer management and scheduling algorithms specifies the fairness and the efficiency of a multiple source queuing system. It is important to note that there is a trade off between the fairness and efficiency issues as quality upgrade in one leads to quality degradation in another.

In this study, two different scheduling algorithms are considered. These are namely Fixed Time Division Multiplexing (FTDM) and Statistical Time Division Multiplexing (STDM). While in FTDM each source takes advantage of a fair portion of the server bandwidth also known as the service rate and there is no bandwidth sharing, in STDM the unused portion of the bandwidth assigned to each source might be used to service other sources. Although FTDM is typically used for ATM switching systems with a number of Virtual Paths, STDM is typically used in ATM queuing systems with a number of Virtual Channels.

There are a number of different buffer management algorithms studied in literature as described in [18], [10], [14], [15], and [9]. The first and probably the simplest method is Complete Sharing (CS) in which the buffer space is shared among all of the existing sources without enforcing any capacity allocation

Homayoun Yousefi'zadeh (hyousefi@uci.edu) is with the Center for Pervasive Communications of the Electrical and Computer Engineering Department at the University of California, Irvine. Edmond A. Jonckheere and John A. Silvester ([jonkhee,silvester]@usc.edu) are with the Department of Electrical Engineering - Systems at the University of Southern California.

mechanism. This method introduces an unfair consumption of the buffer space by greedy sources while providing the lowest loss rates. The second method is called Complete Partitioning (CP) in which the available capacity of the buffer is shared equally among the existing sources. This method has the best fairness characteristic while it greatly suffers from efficiency degradation by introducing the highest loss rate. In presence of a specified scheduling algorithm, a threshold buffer management algorithm is introduced as the third alternative solution. It might be considered as a solution between the above explained two extremes. In a threshold method, each source has its own fixed portion of the buffer space which can only be used for buffering packets generated by that specific source. There is also an additional shared portion of the buffer which is completely shared among the existing sources. This method is also called Partial Sharing (PS).

A dynamic buffer management algorithm is classified under the threshold methods with the ability to adjust the buffer size of each source dynamically. A dynamic buffer management algorithm is able to offer an acceptable solution considering both fairness and efficiency issues, i.e., the algorithm should assign a fair portion of the buffer space to each source with the ability to dynamically adjust the buffer space partitions.

The algorithm introduced in this paper is, in fact, a dynamic buffer management algorithm. It is capable of improving the loss performance of Static Partial Sharing (SPS) method introduced in [18] while considering fairness versus loss trade off. The algorithm relies on the power of neural networks to model self-similar traffic patterns of the individual sources in a multiple source queuing system and dynamically adjust the portion of the buffer space assigned to each source according to the corresponding traffic generation pattern. Relying on the prediction power of neural networks and as shown in simulations, the technique is able to outperform other threshold algorithms studied in the literature.

An outline of the paper follows. In Section II, we briefly review the characteristics of aggregated traffic patterns with self-similar nature. In Section III, we give an overview of the neural network modeling of bursty traffic patterns. Section IV describes a typical multiple source system used for the application task. In Section V, we discuss the packet loss reduction application and compare the performance of our Dynamic Neural Sharing (DNS) scheme with other buffer management schemes. Finally, Section VI concludes the paper.

II. Aggregated Bursty Traffic

The main objective of the current section is to provide an analytical framework for self-similarity as a statistical property of time series. Intuitively, self-similar phenomena display structural similarities across a significant number of time scales. The degree of self-similarity is sometimes specified by measuring a single parameter called Hurst parameter. In the following section, we provide a brief discussion about mathematical and statistical properties of the self-similar processes.

A. Second-Order Self-Similarity

Suppose $X = (X_t : t = 0, 1, 2, ...)$ is a covariance stationary stochastic process with mean μ, variance σ^2, and autocorrelation function $R(n), \quad n \geq 0$. Particularly, assume the autocorrelation function of X has the form

$$R(n) \sim k_1 n^{-\beta}, \quad \text{as} \qquad n \to \infty \tag{1}$$

where $0 < \beta < 1$ and constants $k_1, k_2, ...$ are finite positive integers. For each $m = 1, 2, 3, ...$ let $X^{(m)} = (X_n^{(m)} : n = 1, 2, 3, ...)$ be the covariance stationary time series with corresponding autocorrelation function $R^{(m)}$ obtained from averaging the original series X over the non-overlapping time periods of size m, i.e., for each $m = 1, 2, 3, \cdots$, the covariance $X^{(m)}$ is given by

$$X_n^{(m)} = \frac{1}{m}(X_{nm-m+1} + \cdots + X_{nm}), \quad n \geq 1 \tag{2}$$

The process X is called exactly second-order self-similar with the self-similarity parameter $H = 1 - \beta/2$ if the corresponding $X^{(m)}$ has the same correlation function as X, i.e., $R^{(m)}(n) = R(n)$ for all $m = 1, 2, 3, \cdots$ and $n = 1, 2, 3, \cdots$. The process X is called asymptotically second-order self-similar with self-similarity parameter $H = 1 - \beta/2$ if $R^{(m)}(n)$ asymptotically approaches to $R(n)$ given by (1), for large m and n. Hence, if the correlation functions of the aggregated processes $X^{(m)}$ are the same as the correlation functions of X or approach asymptotically to the correlation functions of X, then X is called exactly or asymptotically second-order self-similar.

Fractal Gaussian Noise (FGN) is a good example of an exactly self-similar process with self-similarity parameter $H, \quad 1/2 < H < 1$. Fractional Arima processes with the parameters (p, d, q) such that $0 < d < 1/2$ are examples of asymptotically second-order self-similar processes with self-similarity parameter $d + 1/2$.

Mathematically, self-similarity manifests itself in a number of ways as follows.

- The variance of sample mean decreases more slowly than the reciprocal of the sample size. This is called slowly decaying variance property which means $var(X^{(m)}) \sim k_2 m^{(-\beta)}$ as $m \to \infty$ with $0 < \beta < 1$.
- The autocorrelations decay hyperbolically rather than exponentially fast, implying a non-summable autocorrelation function $\sum_n R(n) = \infty$. This is called long range dependence property.
- The spectral density $f(.)$ obeys a power-law near the origin. This is the concept of $1/f$ noise with the meaning $f(\lambda) = k_3 \lambda^{-\gamma}$ as $\lambda \to \infty$ with $0 < \gamma < 1$ and $\gamma = 1 - \beta$.

The most important feature of the self-similar processes is seemingly the fact that their aggregated processes $X^{(m)}$ possess a non-degenerate correlation function as $m \to \infty$. This is completely different from typical packet traffic models previously considered in literature, all of which have the property that their aggregated processes $X^{(m)}$ tend to second order pure noise, i.e., $R^{(m)} \to 0$ as $m \to \infty$.

The concept of self-similar processes provides a very elegant explanation for the Hurst effect phenomenon. In order to describe the Hurst effect, we should first describe the rescaled adjusted range. For a given set of observations $(X_n : n = 1, 2, ..., N)$ with sample mean $\overline{X}(N)$ and sample variance

$S^2(N)$, the rescaled adjusted range denoted by the R/S statistic is given by

$$\frac{R(N)}{S(N)} = \frac{1}{S(N)}[max(W_i) - min(W_i)] \quad (3)$$

where $i = 0, ..., N, \quad W_0 = 0$ and

$$W_n = (X_1 + ... + X_n) - n\overline{X}(N), \quad n \geq 1 \quad (4)$$

While many time series appear to be well represented by the relation $E[R(N)/S(N)] \sim k_4 N^H$, as $N \to \infty$, with Hurst parameter H typically about 0.73, observations X_n from a short-range dependent models are known to satisfy $E[R(N)/S(N)] \sim k_5 N^{0.5}$, as $N \to \infty$. This is usually referred to as the Hurst effect.

B. An Evidence of Self-Similarity: Ethernet and VBR Traffic Measurements

In [3], Beran et al. observed the absence of natural length of a burst for the traffic patterns of Variable Bit Rate (VBR) video traffic. In [16], Leland et al. observed the same feature for the high quality, high time-resolution LAN traffic data collected between August 1989 and February 1992 on several Ethernet LANs. This behavior is very different from conventional telephone traffic and from previously considered formal models of packet traffic. With the available data sets, Leland et al. investigated the persistent feature of Ethernet traffic across the network and across the time irrespective of the medium utilization level. They graphically estimated a Hurst parameter H of about 0.80. In general, the degree of self-similarity depends on the utilization level of the medium. For the Ethernet it increases as the utilization increases.

III. Neural Network Modeling of Bursty Traffic

Earlier in this paper we mentioned that analysis of traffic data from networks and services have demonstrated the presence of statistical self-similar characteristics. We also mentioned that only few of the formal models of the packet traffic considered in the literature are able to capture self-similar nature of the measured traffic. The following lists a number of implications of the existence of self-similar packet traffic on high-speed networks.

- The Hurst parameter and fractal dimensions such as correlation dimension provide a more satisfactory measure of burstiness for self-similar traffic than the previously used measures such as the index of dispersion of counts.
- The nature of congestion produced by self-similar network traffic models differs from that predicted by standard models. More specifically, the efficiency of the proposed congestion control schemes for high-speed networks greatly depends on how well those schemes perform under the influence of self-similar traffic scenarios.

In [27], we introduce an elegant modeling technique of self-similar packet traffic that is capable of coping with the fractal properties of the aggregated traffic. Motivated by a desire for having a relatively simple model of the complex packet traffic generation process, our technique proposes the use of a fixed structure perceptron neural network with back propagation learning algorithm as a simpler alternative to stochastic and chaotic systems approaches proposed in [16], [7], [2], and [1].

The promise of neural network modeling approach is to replace the analytical difficulties encountered in the other modeling approaches with a straight forward computational algorithm. As oppose to the other modeling approaches, neural network modeling does not introduce a parameter describing the fractal nature of traffic neither does it investigate identification of appropriate maps. It, hence, need not cope with the complexity of estimating Hurst parameter and/or fractal dimensions. The approach simply takes advantage of using a fixed structure nonlinear system with a well defined analytical model that is able to predict a traffic pattern after learning the dynamics of the pattern through the use of information available in a number of traffic samples. The fixed structure feed forward perceptron neural network used for the task of modeling consists of an input layer with eight neurons, three hidden layers with twenty neurons in each layer, and an output layer with one neuron. Figure (1) shows the structure of the network. Although it might be possible to reduce either the number of the neurons in the input layer or the number of hidden layers, we use the above-mentioned structure as the structure practically fits the dynamics of the traffic pattern. The neural network relies on

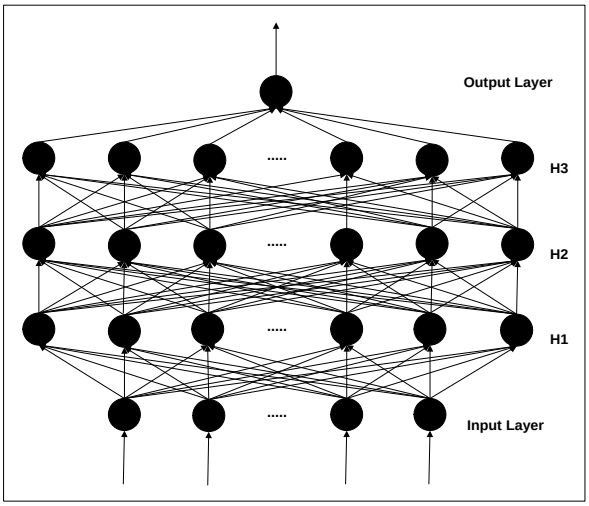

Fig. 1. Fixed structure neural network with 8 neurons in the input layer and 20 neurons in each of the three hidden layers used for traffic modeling task.

back propagation learning algorithm as described in [27], [19], [8], [21], [11], [26], [20], and [4] in which the learning phase is directly followed by the recalling phase when the network output is able to follow the real traffic within an acceptable error bound, ϵ. In what follows we briefly describe back propagation learning algorithm. Using the notation

- $x_j[s]$: The present output state of the j-th neuron from the layer s
- $w_{ji}[s]$: Weighting function of the connection between i-th neuron from layer $(s-1)$ and the j-th neuron from layer s
- $I_j[s]$: The combined input of the j-th neuron of layer s
- $u_j[s]$: The external input of the j-th neuron of the first layer

We note that in our network, a neuron transfers its output as

$$x_j[s] = f\{\sum_i (w_{ji}[s].x_i[s-1])\} = f\{I_j[s]\} \quad (5)$$

where f is the sigmoid function defined as

$$f(z) = \frac{1}{1+e^{-z}} \quad (6)$$

For the absolute error function E defined as

$$E = \frac{1}{2}\sum_k (y_k - \vartheta_k)^2 \quad (7)$$

with ϑ indicating the present output of the network to the input u, y corresponding to the real output, and index k denoting various elements of y and ϑ, the critical parameter that is back propagated into the network is

$$e_j[s] = -\frac{\partial E}{\partial I_j[s]} \quad (8)$$

The relationship between the relative error of a specified neuron in layer s and the local errors in layer $s+1$ is expressed as

$$e_j[s] = x_j[s].(1-x_j[s]) \,.\, \sum_k \{e_k[s+1].w_{kj}[s+1]\} \quad (9)$$

In order to decrease the absolute error function, the weighting functions are changed in the opposite direction of the gradient vector as

$$\Delta w_{ji}[s] = -lc\frac{\partial E}{\partial w_{ji}[s]} = lc.e_j[s].x_i[s-1] \quad (10)$$

where lc denotes the learning coefficient. Our implementation of the back propagation algorithm is, hence, described below.

- Propagate input u in the forward direction through the network until reaching output ϑ. During propagation of this information through the network, set all of the combined inputs I_j and output states x_j for each neuron.
- For each neuron in the output layer calculate the scaled local error and the variations of weighting functions from relations

$$\begin{aligned} e_o &= -\frac{\partial E}{\partial I_o} = -\frac{\partial E}{\partial \vartheta}\frac{\partial \vartheta}{\partial I_o} = (y-\vartheta).f\prime(I_o) \\ &= (y-\vartheta).\vartheta.(1-\vartheta) \end{aligned} \quad (11)$$

$$\begin{aligned} \underbrace{\Delta w_{ji}[s]}_{(k+1)-\text{th step}} &= lc.e_j[s].\{x_i[s-1] + k.e_i[s-1]\} \\ &+ M(\underbrace{\Delta w_{ji}[s]}_{k-\text{th step}}) \end{aligned} \quad (12)$$

  where M stands for the momentum.
- For each neuron in layer s located below the output layer and above the input layer obtain the scaled relative error and the variations of the weighting functions from relations

$$e_j[s] = x_j[s].(1-x_j[s]) \,.\, \sum_k \{e_k[s+1].w_{kj}[s+1]\} \quad (13)$$

$$\begin{aligned} \underbrace{\Delta w_{ji}[s]}_{(k+1)-\text{th step}} &= lc.e_j[s].\{x_i[s-1] + k.e_i[s-1]\} \\ &+ M(\underbrace{\Delta w_{ji}[s]}_{k-\text{th step}}) \end{aligned} \quad (14)$$

- Update all of the weighting functions by adding the variations to the old values.

In a typical iteration of the learning phase, the neural network is provided with samples $x[k-8]$ through $x[k-1]$ of the real traffic pattern and the difference between sample $x[k]$ of the real traffic pattern and the neural network output is used to adjust the weighting functions of the network accordingly. In the next iteration, sample $x[k-8]$ of the real traffic pattern is discarded, samples $x[k-7]$ through $x[k]$ of the real traffic pattern are used as the new input sample set, and sample $x[k+1]$ is used as the new real traffic sample. The neural network continues processing more information in consecutive iterations of the learning phase until the absolute error is less than a specified error bound. The learning phase of the perceptron neural network is directly followed by the recalling phase when the network output is able to follow the real traffic within the acceptable error bound, ϵ. In each iteration of the recalling phase, the neural network independently generates the samples by discarding the oldest input sample, shifting the input samples by one, and using its output as the most recent input sample. The same sequence of following a learning phase by a recalling phase is repeated when and if the neural network output difference exceeds the acceptable error bound, ϵ. The number of samples required for the training of the neural network depends on the complexity of the traffic pattern dynamics. The method may be used to model source level as well as aggregated level bursty traffic. The time complexity and the space complexity of the back propagation algorithm are respectively $\mathcal{O}(IN)$ and $\mathcal{O}(N)$ where N is the number of weighting functions in the network and I is the number of iterations. Although the complexity is typically better than the complexity of implementing statistical approaches such as fractional ARIMA processes or the complexity of calculating fractal dimensions such as correlation dimension, wide variations of I prevent us from making a strong claim about complexity advantage of the algorithm compare to other algorithms. Nonetheless combining the straight forward way of implementation with the analysis of complexity, we claim that the neural network modeling approach provides an elegant approach for the task of traffic modeling. In the following section, we apply the proposed neural network modeling technique to reduce the packet loss rate of a shared buffer in a typical multiple source system.

IV. Multiple Source System

This section describes the underlying multiple source system used in our study. The multiple source system consists of a number of ON-OFF source models. Traffic pattern of each source includes the packets generated by a number of ON-OFF maps. In this section, we also provides a brief discussion viewing each source and the corresponding buffer as a separate queuing system. This is an appearance of complete partitioning buffer management scheme employing FTDM service scheduling at the output of the queue. The discussion may also be directly applied to the case of complete sharing buffer management scheme employing FTDM service scheduling at the output of the queue or may be slightly changed to apply to the case of complete/partial partitioning buffer management scheme employing STDM service scheduling at the output of the queue. The model may be considered as the so-called burst scale queuing component of an ATM queuing system with a number of Virtual Channels (VCs) and each VC belonging to a traffic source as described by [22].

In our model, there is a finite capacity buffer corresponding to each source storing generated packets before they get transmitted. The occupancy of each buffer is determined by the flow of the cells from the corresponding source and the rate at which the cells are serviced. In this model, a queue is identified by its buffer capacity C_{max}, and its server capacity O_{max}. In each queue, the generation rate is compared with the service rate to determine whether the size of the queue is increasing or decreasing as well as whether the queue is losing cells.

Here the challenge is the dynamic assignment of the buffer space such that the probability of loss is minimized. Using the following notation,

- $I(i,k)$: The input rate of the i-th buffer at time k.
- $O(i,k)$: The output rate of the i-th buffer at time k.
- $Q(i,k)$: The queuing rate of the i-th buffer at time k.
- $L(i,k)$: The loss rate of the i-th buffer at time k.
- $C(i,k)$: The queue size of the i-th buffer at time k.

the state of the queue for each buffer is specified by

$$I(i,k) = O(i,k) + Q(i,k) + L(i,k) \tag{15}$$

at any instant of time as shown in Figure (2). Note that besides $Q(i,k)$ values which could be positive or negative, all of the other values are always positive. Originally, all of the queues

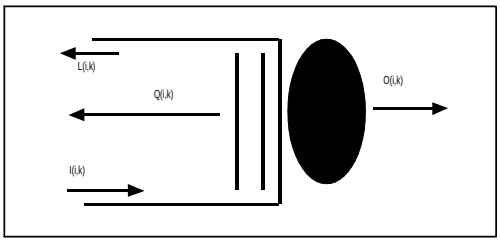

Fig. 2. Queuing diagram of the i-th source at time k.

are empty. A queue begins to form when the buffer input rate exceeds the service rate. Hence, the queue rate $Q(i,k)$ and the loss rate $L(i,k)$ remain zero as long as the input rate is less than or equal the service rate, i.e.,

$$O(i,k) = \begin{cases} I(i,k) & : \quad I(i,k) < O_{max} \\ O_{max} & : \quad I(i,k) = O_{max} \end{cases} \tag{16}$$

The queue size $C(i,k)$ begins to increase as soon as the input rate exceeds the service rate O_{max}. While the queue is not empty, the output rate is always equal to the queue server capacity and the total queuing rate is the difference between the input rate and queue server capacity. The loss rate is zero at this stage.

The queue size keeps increasing and finally becomes full if the input rate remains higher than the queue server capacity. In that situation, the queuing rate is zero and the excess input rate is the cell loss rate as,

$$L(i,k) = I(i,k) - O(i,k) \tag{17}$$

with $O(i,k) = O_{max}$. The effect of a change in the input rate is not immediately appeared if there are packets in the queue waiting to be transmitted. It is the queuing rate which changes according to

$$Q(i,k+1) = Q(i,k) + I(i,k+1) - I(i,k) \tag{18}$$

The queue size begins to decrease in size when the input rate becomes less than the server capacity, i.e., $I(i,k) < O_{max}$, and the queuing rate goes below zero as the result, i.e., $Q(i,k) < 0$. The queue becomes empty if this situation lasts. The output rate is obtained from the following equation,

$$O(i,k) = \begin{cases} O_{max} & : \quad C(i,k) \geq 0 \\ I(i,k) & : \quad C(i,k) = 0 \end{cases} \tag{19}$$

After providing a brief queuing analysis for individual queues, it is now time to take a look at the system from a high level point of view. In the following section, it is assumed that a number of sources are sharing the total available buffer space. The traffic pattern of each source includes the packets generated by a number of artificial ON-OFF source models. An ON-OFF source model is generating traffic at a peak rate when it is active and becomes active as soon as the state variable of the describing nonlinear map goes beyond the threshold value. The source becomes passive as soon as the state variable goes below the threshold. We choose double intermittency chaotic map as the artificial ON-OFF source model which is known to generate a bursty traffic pattern according to [7] and [27]. Figure (3) shows a sample state transition of double intermittency map in the phase plane starting from an initial condition. The two map segments of Figure (3) associate with the source in passive and active states. By using different initial conditions and/or different threshold values, different traffic patterns are obtained for different sources.

V. Loss Reduction

Consider a system consisting of a number of sources sharing the space available in a central buffer and generating packets following an ON-OFF source model. Figure (4) shows the

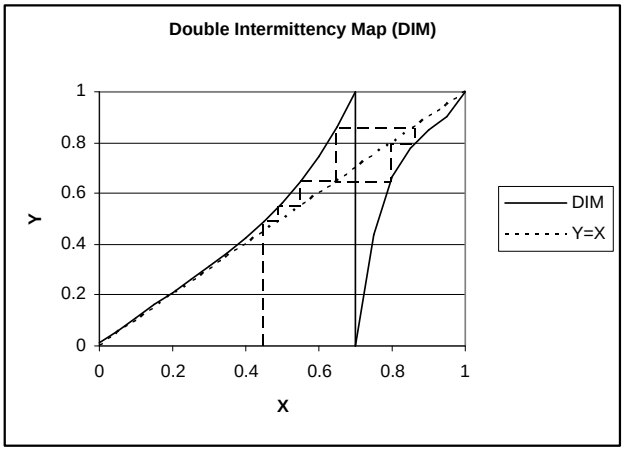

Fig. 3. Sample state transition of double intermittency map.

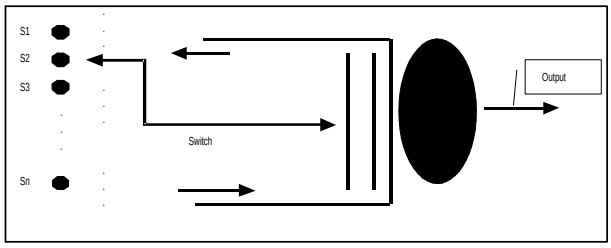

Fig. 4. The structure of a multiple source queuing system.

structure of a multiple source queuing system. In order to show the performance of the modeling approach, four different buffer management scenarios are compared together in presence of FTDM and STDM scheduling algorithms.

- In the first method complete sharing mechanism is deployed, i.e., there is only one queue for all of the sources. This is a simple queuing mechanism in which all of the generated packets are directly sent to the central buffer and wait there until getting transmitted. This method has the advantage that the buffer is shared among all the sources, hence the maximum efficiency for the buffer space is obtained. The drawback of the method is that the space may not be used fairly as a source with a high output rate is able to consume a big portion of the buffer space and cause the queue to overflow. It is worth mentioning that the results of FTDM and STDM are the same in this scenario as there is only one buffer in the system.
- The second method is a simple implementation of complete partitioning scheme in presence of FTDM and STDM in which the capacity of the central buffer is distributed equally among the sources. The most important characteristic of the method is that the buffer space is distributed fairly. FTDM is suffering from a possible low efficiency rate compare to STDM, i.e., sources with lower generation rates may not use the whole portion of the buffer space assigned to them while sources with higher generation rate are losing packets. This is not happening when STDM is employed as the unused portion of the buffer space is used for the packets generated by other sources.
- The third method is a simple implementation of partial sharing scheme that has three equal portions for the three sources with an additional shared portion that can be shared among all the sources. This is also called partial sharing.
- The fourth method is the dynamic assignment of the buffer space relying on the results obtained from the neural network prediction algorithm, i.e., adjusting the buffer space according to the packet generation pattern of each source. This is a generalization of the third method keeping the shared portion size fixed and adjusting the buffer space size of each source dynamically.

It is important to mention that in case of the last three methods, there is a separate queue for each source storing the packets generated by that source. The difference between the third and the fourth scenario is that in the third scenario the buffer space assigned to each source is fixed and each source is able to send its generated packets either to its own buffer or the shared buffer if space is available while in the fourth scenario the portion of the buffer space assigned to the source with a higher packet generation rate is increased in case other sources are not generating enough packets to use their allocated share of the buffer space. This is a dynamic buffer management algorithm with a potential to perform better than the existing buffer management algorithms as it relies on predicted future information.

In order to investigate the performance of the method, a triple source system is used. The traffic patterns of the first, second, and third source consist of an aggregated artificial traffic pattern generated by 30, 40, and 50 individual double intermittency map packet generators respectively. The traffic generated by each source is collected and sent to the corresponding buffer in a round robin manner. It is specially important to note that there is a difference among the number of packets generated by each source as the result of having a different number of ON-OFF packet generators per source. In order to evaluate the performance of different methods, the over all as well as per source loss rate of the system for different choices of buffer size with a fixed service rate are compared together. The buffer space can be shared among all of the sources or may be divided into equal portions for individual source usage. The server bandwidth may also be used according to FTDM or STDM scheduling mechanisms. Figures (5) through (8) show the total and single source packet loss rate versus buffer size diagram for the triple source queuing system in presence of FTDM and STDM scheduling algorithms. In the single source case, we choose the source with the lowest packet generation rate in order to be able to compare the fairness of different schemes. The simulation results have been obtained from an iterative algorithm with a total number of ten million iterations per choice of buffer size. Applying a continuous sequence of learning and recalling phases, the fixed structure neural network has been able to follow the traffic pattern within the specified error range between 20 and 30 times covering an average of fifty samples per time. We note that the performance of different methods can be drastically different as the result of applying different methods for traffic management of a heavily utilized system.

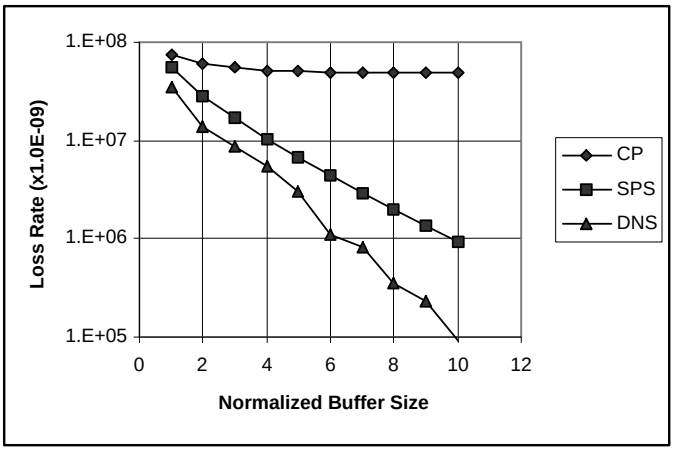

Fig. 5. Total loss rate versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) in presence of FTDM scheduling algorithm.

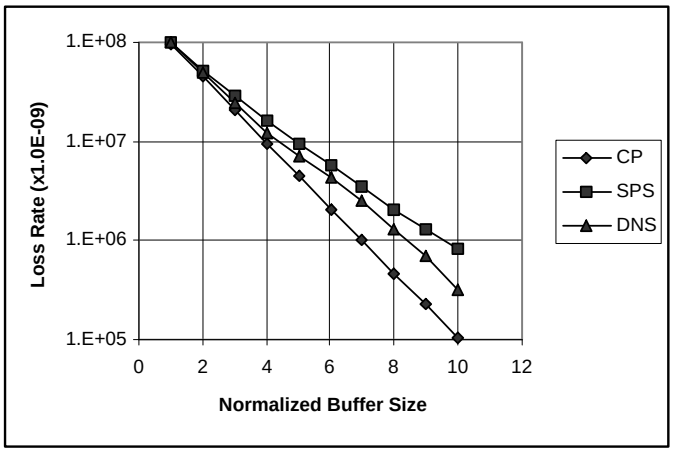

Fig. 6. Single source loss rate versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) in presence of FTDM scheduling algorithm.

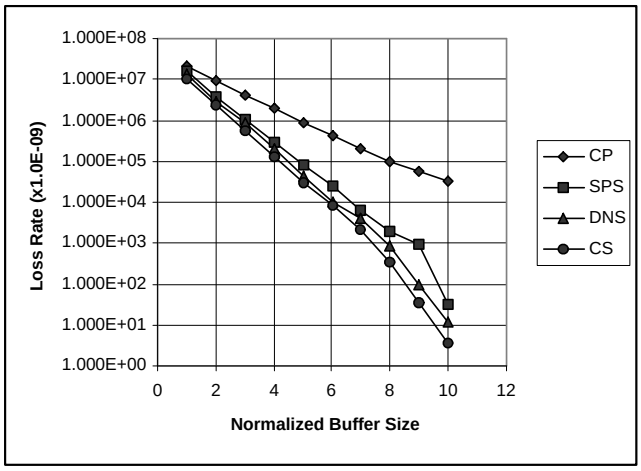

Fig. 7. Total loss rate versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) in presence of STDM scheduling algorithm.

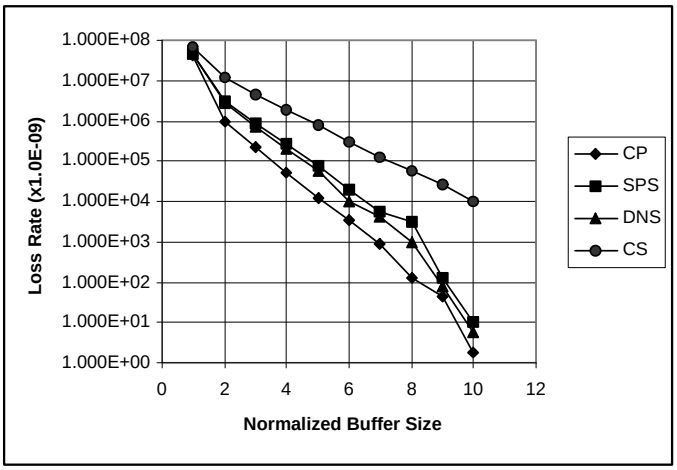

Fig. 8. Single source loss rate versus buffer size diagram for the triple source queuing system using complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) in presence of STDM scheduling algorithm.

Figure (5) shows total loss rates of complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) buffer management schemes versus different values of normalized buffer size in presence of FTDM scheduling algorithm. Figure (6) shows individual loss rates of the most passive source for complete partitioning (CP), static partial sharing (SPS), and dynamic neural sharing (DNS) buffer management schemes versus different values of normalized buffer size in presence of FTDM scheduling algorithm. Figure (7) shows total loss rates of complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) buffer management schemes versus different values of normalized buffer size in presence of STDM scheduling algorithm. Figure (8) shows individual loss rates of the most passive source for complete partitioning (CP), static partial sharing (SPS), dynamic neural sharing (DNS), and complete sharing (CS) buffer management schemes versus different values of normalized buffer size in presence of STDM scheduling algorithm. All of the numbers in the figures have been scaled as multiples of 10^{-9}.

It is clearly observed from the figures that for both FTDM and STDM using dynamic neural sharing scheme, the total loss rate compared to complete partitioning scheme as well as per source loss rate compared to complete sharing scheme are reduced. The results may be interpreted as the evidence that dynamic neural sharing scheme has come up with a solution in between the two extreme cases. Comparing the results for static partial sharing and neural dynamic sharing show the higher efficiency of the latter method. This is a significant improvement compare to the other three schemes.

We close this section by some of the practical findings in the implementation of the algorithm. First, we note that the learning algorithm of the perceptron neural network used for the task of modeling is time consuming because of the rich dynamic of the traffic pattern that the neural network is trying to learn. In deed, the neural network needs to access thousands of samples in each training period resulting in a typical sequence of learn-

ing and recalling phases with few hundred thousand samples and hundreds of samples respectively. In addition, all of the convergence results are strongly affected by the choice of initial conditions of the weighting functions of the neural network. As a practical finding, setting the initial values of the weighting functions of the neural network at $w_{ji}(0) = 0.01 \quad \forall i, j$ typically yields good results. Our justification for both of the above-mentioned phenomena is the fact that the proposed neural network is trying to learn complicated dynamics of chaotic maps exhibiting extreme sensitivity to variations of initial conditions.

VI. Conclusion

In this paper, we studied packet loss reduction in a class of multiple source queuing systems as an application of neural network modeling of self-similar packet traffic. We modeled self-similar traffic patterns using a fixed structure perceptron neural network with three hidden layers. The neural network had eight neurons in its input layer, and twenty neurons in each of its three hidden layers. The number of neurons in its output layer was one. In the learning phase of the modeling process, we applied eight consecutive samples of the packets as the input of the neural network and used the ninth sample as the desired output in each iteration. The number of required samples for the network to be trained, depended on the steady state behavior of the traffic pattern and the network load, i.e, the number of required samples increased in case of heavily loaded network indicating more complicated steady state behavior.

We used a neural-based dynamic buffer management scheme called dynamic neural sharing to improve the loss performance of static partial sharing buffer management algorithm while considering the fairness issue. Relying on the prediction power of neural networks, our neural-based algorithm was able to dynamically adjust the buffer allocation of individual sources in a multiple source system with a central shared or partitioned buffer.

We also compared the performance of different buffer management schemes, namely complete sharing, complete partitioning, static partial sharing, and dynamic neural sharing in presence of different server scheduling algorithms, fixed time division multiplexing and statistic time division multiplexing, and concluded that our dynamic neural sharing scheme was able to offer the best solution considering the trade off between fairness and loss issues.

References

[1] A.Adas, "Traffic Models in Broadband Networks", IEEE Communications Magazine, PP 82-89, July 1997.

[2] A. Alkhatib, M. Krunz, "Application of Chaos Theory to the Modeling of Compressed Video", Proceedings of the IEEE ICC 2000 Conference, Vol. 2, New Orleans, June 2000.

[3] J. Beran, R. Sherman, M. S. Taqqu, W. Willinger, "Variable Bit Rate Video Traffic and Long Range Dependence", IEEE/ACM Trans. on Networking, Vol. 2, NO. 3, Apr. 1994.

[4] N. J. Dimpoulos, "A Study of Asymptotic Behavior of Neural Networks", IEEE Trans. on Circuit & Syst., Vol. 36, No.5, 1989.

[5] D. E. Duffy, W. Willinger, "Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks", IEEE JSAC, 1994.

[6] A. Erramilli, J. Gordon, W. Willinger, "Applications of Fractals in Engineering for Realistic Traffic Processes", ITC Vol. 14, PP. 35-44, 1994.

[7] A. Erramilli, R. P. Singh, P. Pruthi, "Chaotic Maps as Models of Packet Traffic", ITC Vol. 14, PP. 329-338, 1994.

[8] S. E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks", Technical Report CMU-CS-88-162, Carnegie Mellon University, June 1988.

[9] G. Gallasi, C. Rigolio, "ATM Bandwidth Assignment and Bandwidth Enforcement Policies", Proc. of IEEE GLOBECOM'89, Dec. 1987.

[10] M. Gerla, L. Kleinrock, "Flow Control: A Comparative Survey", IEEE Trans. on Commun., Vol. COM-28, No. 4, Apr. 1980.

[11] H. Guo, S. Gelfland, "Analysis of Gradient Descent Learning Algorithms for Multilayer Feed Forward Neural Networks", IEEE Trans. on Circuit & Syst., Vol. 38, No.8, Aug. 1991.

[12] H. Heffes, D. M. Lucantoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance", IEEE JSAC, Vol. 9, NO. 7, Sep. 1991.

[13] K. M. Hellstern, P. Wirth, "Traffic Models for ISDN Data Users: Office Automation Application", Proc. ITC-13, Denmark, 1991.

[14] F. Kamoun, L. Kleinrock, "Analysis of Shared Storage in a Computer Network Node Environment under General Traffic Conditions", IEEE Trans. on Commun., Vol. COM-28, No. 7, Jul. 1980.

[15] P. Kermani, L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique", Computer Networks, Vol.3, 1979.

[16] W. E. Leland, W. Willinger, M. S. Taqqu, D. V. Willson, "Statistical Analysis and Stochastic Modeling of Self-Similar Datatraffic", ITC Vol. 14, PP. 319-328, 1994.

[17] W. E. Leland, W. Willinger, M. S. Taqqu, D. V. Willson, "On the Self-Similar Nature of Ethernet Traffic", IEEE/ACM Trans. on Networking, Vol. 2, NO. 1, PP. 1-15, Feb. 1994.

[18] A. Lin, J. A. Silvester, "Priority Queuing Strategies and Buffer Allocation Protocols in Traffic Control at an ATM Integrated Broadband Switching System", IEEE JSAC, Vol. 9, No. 9, Dec. 1991.

[19] M. Minsky, S. A. Papert, "Perceptrons: An Introduction to Computational Geometry", MIT Press, Cambridge, MA, expanded edition, 1988/1969.

[20] G. Mirchandani, W. Cao, "On Hidden Nodes for Neural Nets", IEEE Trans. on Circuit & Syst., Vol.36, No.5, 1989.

[21] A. Van Ooyen, B. Neihuis, "Improving the Convergence of Back Propagation Algorithm", Neural Networks, Vol.5, No.3, 1992

[22] J. M. Pitts, L. G. Cuthbert, M. Bocci, E. M. Scharf, "An Accelerated Simulation Technique for Modeling Burst Scale Queuing Behavior in ATM", ITC Vol. 14, PP. 777-786, 1994.

[23] V. Ramaswami, "Traffic Performance Modeling for Packet Communication: Whence, Where, and Whither", Proc. of Australian Teletraffic Seminar, Nov. 1988.

[24] K. Sriram, W. Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data", IEEE JSAC, Vol. SAC-4, NO. 6, Sep. 1986.

[25] W. Willinger, M. Taqqu, "Self-Similarity through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", IEEE/ACM Trans. on Networking, Vol. 5, No. 1, Feb. 1997.

[26] H. Yang, T. Dillon, "Convergence of Self-Organizing Neural Algorithms", Neural Networks, Vol.5, No.3, 1992.

[27] H. Yousefi'zadeh, "Neural Network Modeling of a Class of ON-OFF Source Models with Self-Similar Characteristics", Submitted to IEEE/ACM Trans. on Networking, Feb. 2002. Also available at http://www.ece.uci.edu/~ hyousefi/pub.html.

[28] H. Yousefi'zadeh, E. A. Jonckheere, "Neural Networks Modeling of Discrete Time Chaotic Maps", Technical Draft, Jan. 2002. Available at http://www.ece.uci.edu/~ hyousefi/pub.html.

[29] H. Yousefi'zadeh, M. Shafiee, A. Ziloochian, "Chaotic Arrays Modeling with Neural Networks", Proc. of the Iranian Conference of Electrical Eng., Vol.4, PP. 667- 679, May 1993.

PLACE PHOTO HERE

Homayoun Yousefi'zadeh was born in Tehran, Iran. He received B.S., M.S., and Ph.D. degrees all in Electrical Engineering from Sharif University of Technology, Tehran Polytechnic Institute, and University of Southern California in 1989, 1993, and 1997 respectively. He is currently with the Center for Pervasive Communications in Electrical and Computer Engineering Department of University of California, Irvine. Dr. Yousefi'zadeh has been involved with a number of academic and industry initiatives in different capacities including chairperson of systems' management workgroup of Storage Networking Industry Association (SNIA), member of Scientific Advisory Board (SAB) of Integrated Media Services Center (IMSC) at the University of Southern of California, referee for IEEE Communication Letters, referee for International Journal of Computer and Electrical Engineering, Smember of American Management Association (AMA), and

member of American Society for Engineering Education (ASEE). His research work and interest include intelligent modeling of nonlinear dynamics, teletraffic analysis and control, real-time media systems, network traffic control, high-speed broadband networks, distributed algorithms, and storage networking.

PLACE PHOTO HERE

Edmond A. Jonckheere was born in Belgium in 1950. He received the M.S. degree in electrical engineering from the University of Louvain, Belgium, the Dr. Eng. degree in aerospace engineering from the University of Toulouse, France, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1973, 1975, and 1978, respectively. From 1973 to 1975, he held a Research Fellowship of the European Space Agency at the Laboratory for Systems Architecture and Analysis (LAAS), Toulouse, France. From 1975 to 1978, he was a Teaching/Research Assistant and subsequently a Research Associate in the Department of Electrical Engineering at the University of Southern California, Los Angeles. In 1979, he was with the Philips Research Laboratory, Brussels, Belgium. In 1980, he returned to the University of Southern California, where he is currently a Professor of Electrical Engineering and a member of the Center for Applied Mathematical Sciences (CAMS). He has also had short-term academic appointments with the Australian National University and the University of Namur, Belgium. His consulting and other activities include the Max Planck Institute, Germany, Memorial Medical Center of Long Beach, CA, Honeywell, Minneapolis, MN, The Aerospace Corporation, El Segundo, CA, Lockheed-Martin, Palmdale, CA, and American GNC Corporation, Simi Valley, CA. His current research interests include modeling complicated nonlinear dynamics, network flow control, and network hyperbolic geometry.

PLACE PHOTO HERE

John A. Silvester (M'79?SM'85) was born in Kent, England, in 1950. He received the B.A. (M.A.) degree in mathematics and operations research from the University of Cambridge in 1971 (1975), the M.S. degree in statistics and computer science from West Virginia University in 1973, and the Ph.D. degree in computer science from UCLA in 1980. Since 1979, he has been with the University of Southern California where he is Vice Provost of Scholarly Technology and Professor in the Department of Electrical Engineering. He was Vice Provost of Academic Computing from 1994 to 1997 and Director of the Computer Engineering Division from 1984 to 1990. As Vice Provost of Scholarly Technology, he serves as the faculty/administration interface for the development of strategic plans for the effective use of information technology to support the academic mission of the university, and is responsible for working with the CIO and campus technology providers to implement these plans. Areas of responsibility include research and instructional computing, campus data networking, and distance learning. He administers several campus-wide grants programs. As a Professor of Electrical Engineering, he teaches courses in broadband networks, introductory computer networks, and computer network design and analysis. His current research interests are traffic modeling, highspeed networks, and wireless packet networks. He is the author of more than 100 technical papers, and has lectured both in the United States and abroad.