Dynamic Neural-Based Buffer Management for Queuing Systems with Self-Similar Characteristics

Homayoun Yousefi'zadeh   Edmond A. Jonckheere

Abstract—**Buffer management in queuing systems plays an important role in addressing the tradeoff between efficiency measured in terms of overall packet loss and fairness measured in terms of individual source packet loss. Complete partitioning (CP) of a buffer with the best fairness characteristic and complete sharing (CS) of a buffer with the best efficiency characteristic are at the opposite ends of the spectrum of buffer management techniques. Dynamic partitioning buffer management techniques aim at addressing the tradeoff between efficiency and fairness. Ease of implementation is the key issue when determining the practicality of a dynamic buffer management technique. In this paper two novel dynamic buffer management techniques for queuing systems accommodating self-similar traffic patterns are introduced. The techniques take advantage of the adaptive learning power of perceptron neural networks when applied to arriving traffic patterns of queuing systems. Relying on the water-filling approach, our proposed techniques are capable of coping with the tradeoff between packet loss and fairness issues. Computer simulations reveal that both of the proposed techniques enjoy great efficiency and fairness characteristics as well as ease of implementation.**

Index Terms—**Neural Network Teletraffic Forecasting, Water-Filling, Buffer Management, Packet Loss, Fairness.**

I. Introduction

Neural networks are a class of nonlinear systems capable of adaptively learning and performing tasks accomplished by other systems. Their broad range of applications includes speech and signal processing, pattern recognition, and system modeling. The adaptive learning power of neural networks has also proven useful in various contexts of the literature on computer communication networks. For example, neural networks have been successfully utilized in dynamic allocation of bandwidth for Variable Bit Rate (VBR) video over Asynchronous Transfer Mode (ATM) [9]. 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 neural networks makes them quite suitable for forecasting traffic patterns.

In [35] and [36], the absence of natural length of a burst for the high quality, high time-resolution Ethernet LAN traffic data was reported. The data was collected between August 1989 and February 1992 on several Ethernet LANs. This behavior is very different from conventional telephone traffic and from formal models of packet traffic. The available data sets verified the persistent self-similar feature of Ethernet traffic across the network and across the time. In general, the degree of self-similarity depends on the utilization level of the medium. For the Ethernet, it increases as the utilization increases. Analysis of traffic data from other networks and services such as VBR video [7], ISDN traffic [25], Common Channel Signaling Network (CCSN) [11], ATM traffic [41], and broadband networks [3] 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. Forecasting self-similar traffic patterns is more challenging than forecasting traditional traffic patterns considering their rich dynamics.

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 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 also known as service rate. The combination of buffer management and server bandwidth scheduling specifies the efficiency and the fairness of a multiple source queuing system. Accommodating self-similar traffic patterns further complicates the buffer management of multiple source queuing systems manifesting in higher drop rates and longer queuing delays.

In this study, two different server scheduling schemes are considered. These are namely Fixed Time Division Multiplexing (FTDM) and Statistical Time Division Multiplexing (STDM). In FTDM, each source takes advantage of a preassigned portion of the server bandwidth. Server bandwidth is allocated to each source regardless of whether it has data to transmit. In STDM, each source only utilizes an aggregate portion of the server bandwidth when it has actual data to be transmitted. Simply put, the difference between the two schemes is that in FTDM there is no bandwidth sharing while in STDM the unused portion of the server bandwidth assigned to each source might be used to service packets generated by other sources. FTDM is typically used for ATM switching systems with a number of virtual paths where as STDM is typically used in ATM queuing systems with a number of virtual channels.

There are a number of different buffer management schemes studied in the literature as described in [37], [21], [31], [32], and [19]. These are namely Complete Sharing (CS) with no enforced capacity allocation mechanism, Complete Partitioning (CP) with equal partitioning of the available buffer capacity, and Partial Sharing (PS) with dedicated portions of the buffer space as well as a common shared portion. A survey

Homayoun Yousefi'zadeh is with the Department of EECS at the University of California, Irvine; E-Mail: `hyousefi@uci.edu`. Edmond A. Jonckheere is with the Department of EE-Systems at the University of Southern California; E-Mail: `jonkhee@usc.edu`.

of the literature shows that CS achieves optimal throughput-delay performance. However, the work of [28] suggests that the CS scheme may not perform well when accommodating ill-behaved greedy sources or heavily loaded systems. There is also another variation of CS known as the Static Threshold (ST) scheme in which an arriving packet is accommodated if the queue length is smaller than a given threshold. The works of [28], [18], and [37] all propose simple implementations of Static PS (SPS) methods with the objective of balancing the tradeoff between efficiency and fairness. While implementation of these schemes is relatively simple, their performance suffers as the result of relying on static partitioning. The latter is due to the fact that the schemes can allow packet loss in a partitioned buffer while another partitioned buffer is not full. The work of [27] provides a discussion of performance analysis for a class of SPS methods.

A dynamic buffer management scheme is classified under PS methods with the ability to adjust the buffer size of each source according to the overall buffer occupancy. The schemes of [48], [49], [50], [20], and [38] are all classified under Dynamic Push Out (DPO) which is a variant of dynamic buffer management schemes. The schemes investigate different issues of the main DPO idea. In DPO, a packet that arrives to find a partitioned buffer full pushes out the packet at the head of the longest partitioned buffer. Although offering excellent efficiency and fairness characteristics, DPO has proven to have a very high overhead of implementation. Relying on the max-min proposal of [29], the dynamic buffer management scheme of [10] proposes simpler versions of DPO in which the individual partitioned buffer length threshold, at any instant of time, is proportional to the current amount of unused buffering in the main buffer. Packet arrivals for a partitioned buffer are blocked whenever the partitioned buffer length equals or exceeds the current threshold value. In [23], the authors extend their earlier work of [10] to regulate buffer sharing among traffic classes with different loss priorities.

Other buffer management and scheduling schemes that have been extensively discussed in the literature and can be categorized under the above classifications include Earliest Deadline First (EDF), Complete Sharing with Virtual partitioning (CSVP), and Generalized Process Sharing (GPS). Among the set of articles in the literature, the works of [15], [52], and [12] provide an appropriate overview of the latter techniques, respectively. In [53] and [27], performance analysis studies of a number of buffer management schemes are provided. The tradeoff between the available bandwidth and buffer space is studied in [40]. The work of [33] and [5] are among recent literature articles providing a theoretical and an intelligent treatment of the buffer management problem, respectively.

The schemes introduced in this paper are also classified under dynamic buffer management schemes. They are capable of improving the loss performance of SPS scheme of [37] while considering fairness versus loss trade off. Unlike the family of DPO buffer management schemes, our proposed schemes do not need to monitor and access any information about the status of all of the partitioned buffers. Hence, they can be implemented at each buffer independently. The schemes rely on the power of neural networks to forecast the arriving traffic patterns of multiple source queuing systems and dynamically adjust the buffer space partitioning according to the corresponding traffic arrival pattern. The schemes utilize water-filling technique satisfying the so-called max-min fairness property when dynamically adjusting the buffer space. We note that while our proposed schemes are described for fixed-length packets typically utilized in ATM systems, they can also be applied to to variable-length packets.

An outline of the paper follows. In Section II, we briefly review the characteristics of self-similar packet traffic. In Section III, we describe our proposed neural network forecasting schemes of teletraffic patterns. In Section IV, we introduce and analyze a typical multiple source queuing system utilized in the context of our work. Section V provides the details of our buffer management scheme. Section VI includes our simulation results pertaining to packet loss reduction in multiple source queuing systems. In this section, we also compare the performance of our proposed Dynamic Neural Sharing (DNS) schemes with other buffer management schemes. Finally, we conclude the paper in Section VII.

II. Self-Similar Packet Traffic

In this section, we provide a discussion of second-order 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 or a set of parameter(s) called Hurst parameter(s).

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

$$R(k) \sim \eta_1 k^{-\beta}, \quad \text{as} \qquad k \to \infty \tag{1}$$

where $0 < \beta < 1$ and constant η_1 is a finite positive integer. For each $m = 1, 2, 3, ...$ let $X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, ...)$ 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 $m = 1, 2, 3, ...$, $X^{(m)}$ is given by

$$X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + ... + X_{km}), \quad k \geq 1 \tag{2}$$

The process X is called exactly second-order self-similar with the self-similarity parameter $H = 1 - \beta/2$ if the corresponding $X^{(m)}$ processes have the same correlation functions as X, i.e., $R^{(m)}(k) = R(k)$ for all $m = 1, 2, 3, ...$ and $k = 1, 2, 3, ...$ The process X is called asymptotically second-order self-similar with self-similarity parameter $H = 1 - \beta/2$ if $R^{(m)}(k)$ asymptotically approaches to $R(k)$ given by Relationship (1), for large m and k. 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 function 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 $H = d + 1/2$.

Mathematically, self-similarity manifests itself in a number of ways. In our discussion below, we assume that the constants η_2, η_3, η_4, and η_5 are finite positive integers.

- The variance of sample mean decreases more slowly than the reciprocal of the sample size. This is called slowly decaying variance property indicating $var(X^{(m)}) \sim \eta_2 m^{(-\beta)}$ as $m \to \infty$ with $0 < \beta < 1$.
- The autocorrelations decay hyperbolically rather than exponentially fast, implying a non-summable autocorrelation function $\sum_k R(k) = \infty$. 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 $f(\lambda) = \eta_3 \lambda^{-\gamma}$ as $\lambda \to \infty$ with $0 < \gamma < 1$ and $\gamma = 1 - \beta$.

It appears that the most important feature of self-similar processes is that their aggregated process $X^{(m)}$ possesses a non-degenerate correlation function as $m \to \infty$. This is different from traditional packet traffic models with 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 an 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 $(X_k : k = 1, 2, ..., K)$ with sample mean $\overline{X}(K)$ and sample variance $S^2(K)$, the rescaled adjusted range denoted by the R/S statistic is given by

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

where $i = 0, ..., K$ and

$$W_k = (X_1 + ... + X_k) - k\overline{X}(K), \quad k \geq 1 \tag{4}$$

with $W_0 = 0$. While many time series appear to be well represented by the relation $E[R(K)/S(K)] \sim \eta_4 K^H$, as $K \to \infty$, with an average Hurst parameter H typically measured at 0.73 [45], observations X_k from short-range dependent models are known to satisfy $E[R(K)/S(K)] \sim \eta_5 K^{0.5}$, as $K \to \infty$. This is usually referred to as the Hurst effect.

III. Neural Network Forecasting of Packet Traffic

As pointed out in various research articles, many packet traffic sources and patterns exhibit an ON-OFF behavior. An ON-OFF traffic pattern is characterized by two states. Such a pattern is delivering traffic at a peak rate in its active state and is silent in its passive state. Aggregate Ethernet traffic patterns [35] and VBR video sources [7] are among the examples of ON-OFF traffic patterns. In this section, we propose two neural-based techniques of forecasting ON-OFF traffic patterns. While our first technique takes advantage of a first order gradient-based back propagation learning, our second technique utilizes a second order Quasi-Newton back propagation learning.

The main idea of forecasting self-similar traffic patterns with fixed structure neural networks is related to the original proposals of [43] and [14] in which fixed structure feedforward perceptron neural networks with back propagation learning are proposed as potential modeling tools of nonlinear systems. The details of such a modeling task can be found in [24]. Treating self-similar traffic patterns as a class of nonlinear dynamical systems, we use perceptron networks with back propagation learning to model aggregated self-similar traffic patterns as an alternative to stochastic and chaotic systems approaches proposed in [35], [13], [4], and [2]. Our modeling approach provides an attractive solution for traffic modeling and has the advantage of simplicity compared to the previously proposed modeling approaches namely stochastic and deterministic chaotic map modeling. The promise of neural network modeling approach is to replace the analytical difficulties encountered in the other modeling approaches with a straightforward computational algorithm. As opposed to the other modeling approaches, neural network modeling does not investigate identification of appropriate maps neither does it introduce a parameter or a set of parameters describing the fractal nature of traffic. It, hence, need not cope with the complexity of estimating multifractal Hurst parameters [35], [16] and/or fractal dimensions [13]. The proposed neural network forecasting schemes of this work simply take advantage of using a fixed structure nonlinear system with a well defined analytical model that is able to forecast a traffic pattern after learning the dynamics of the pattern through the use of information available in a number of traffic samples. Interestingly and as proposed by Gomes et al. [22], neural networks can also be utilized as appropriate estimators of the Hurst parameter.

The fixed structure, fully connected, feedforward perceptron neural network utilized for the task of forecasting in our study consists of an input layer with eight neurons, three hidden layers with twenty neurons in each layer, and an output layer with one neuron. Fig. 1 illustrates the structure of the neural network. In our perceptron network, a neuron transfers its output

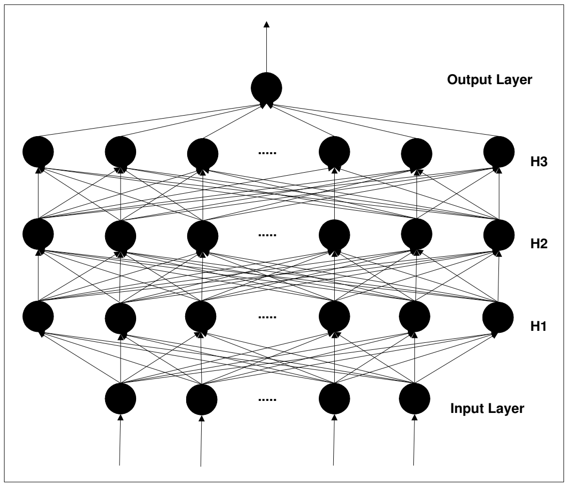

Fig. 1. The fixed structure neural network used for the task of traffic forecasting.

as

$$x^{(j)}[s] = f\{\sum_i (w^{(ji)}[s].x^{(i)}[s-1])\} = f\{\Psi^{(j)}[s]\} \quad (5)$$

where $x^{(j)}[s]$ is the present output state of the j-th neuron in layer s and $w^{(ji)}[s]$ is the weighting function between the i-th neuron in layer $s-1$ and the j-th neuron in layer s. Further, $\Psi^{(j)}[s]$ is the combined input of the j-th neuron in layer s and f is the sigmoid function defined as

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

To consider the threshold effects, each neuron in layer s is assumed to have an extra input with a fixed value of -1 in addition to its inputs from the neurons in layer $s-1$. The learning process of the neural network is nothing more than minimizing its energy function E. The energy function of the network at iteration k of the learning process is defined as

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

where ϑ_k indicates the present output of the network to a given input u_k and y_k corresponds to the actual output.

In our current work, we propose the use of two iterative learning schemes. They are namely the first order gradient-based back propagation and BFGS Quasi-Newton back propagation learning schemes.

A. The First Order Gradient-Based Back Propagation Learning

Our proposed gradient-based back propagation learning scheme overcomes the mismatch between the actual outputs and the generated outputs of the neural network by adjusting the weightings of interconnections in the opposite direction of the gradient vector and its momentums. It is categorized under first degree unconstrained optimization methods and therefore has the advantage of simplicity as well as low space complexity. The latter makes the scheme attractive from the standpoint of implementation in intermediate buffers with limited memory resources. Iteration k of our proposed gradient-based learning scheme is described as

- Form the gradient $s_k = \partial E / \partial w_k$.
- Utilize a quadratic interpolation line search method to find α_k minimizing $E(w_k - \alpha_k\, s_k)$.
- Find the vector of weighting functions' changes as

$$\Delta w_{k+1} = -\alpha_k \,\tfrac{\partial E}{\partial w_k} \,+\, M(\Delta w_k) \quad (8)$$

  with M denoting the momentum term.
- Set $w_{k+1} = w_k + \Delta w_k$.

We refer the reader to Appendix C of [8] for further details of the line search methods.

B. BFGS Quasi-Newton Back Propagation Learning

The BFGS Quasi-Newton back propagation learning scheme also overcomes the mismatch between the actual outputs and the generated outputs of the neural network by adjusting the weightings of interconnections. However, it is categorized under second degree gradient-based unconstrained optimization methods and therefore has much better convergence characteristics compared to any variation of the standard back propagation learning including our first learning scheme. Although quicker convergence of second degree gradient-based methods comes at the cost of requiring to calculate the inverse Hessian matrix in every iteration, BFGS learning avoids such a per iteration calculation as explained below. In iteration $k+1$ of learning, BFGS approximates the inverse Hessian matrix $\frac{\partial^2 E}{\partial w_{k+1}^2}$ by a positive definite matrix D_{k+1} in the form of

$$D_{k+1} \;=\; D_k \,+\, (1 \,+\, \tfrac{q_k^T D_k q_k}{p_k^T p_k})\,\tfrac{p_k p_k^T}{p_k^T p_k} \,-\, \tfrac{D_k q_k p_k^T + p_k q_k^T D_k}{p_k^T p_k} \quad (9)$$

where $p_k = w_{k+1} - w_k$ and $q_k = \frac{\partial E}{\partial w_{k+1}} - \frac{\partial E}{\partial w_k}$. We note that the space complexity of the scheme is higher than that of first degree back propagation learning schemes due to requiring to save the elements of D_k. Iteration k of our proposed BFGS learning scheme is then described as

- Form the gradient $s_k = \partial E / \partial w_k$.
- Calculate matrix D_k as described in Equation (9).
- Utilize a quadratic interpolation line search method to find α_k minimizing $E(w_k - \alpha_k D_k\, s_k)$.
- Set $w_{k+1} = w_k - \alpha_k D_k\, s_k$.

We refer the reader to Section 1.7 of [8] for further details of BFGS algorithm.

The following discussion is applied to both learning schemes described above. In iteration k of learning, both schemes propagate the input vector u_k in the forward direction through the network until reaching to the output ϑ_k. During the propagation process, all of the combined inputs $\Psi^{(j)}$ and output states $x^{(j)}$ for each neuron are set. In iteration k, the neural network input vector u_k consists of samples ϑ_{k-8} through ϑ_{k-1} of the actual traffic pattern. The difference between sample ϑ_k of the actual traffic pattern and the neural network output y_k is used to adjust the weighting functions of the network accordingly. In the next iteration, sample ϑ_{k-8} of the actual traffic pattern is discarded, samples ϑ_{k-7} through ϑ_k of the actual traffic pattern are used as the new input vector, and sample ϑ_{k+1} is used as the new actual output. The neural network continues processing more information in consecutive iterations of the learning phase until the absolute error E is less than a specified error bound ε. Once the absolute error is within the specified error bound ε, the self-generated output of the neural network can be used to forecast a given traffic pattern. The network can independently self-generate samples by discarding the oldest input sample, shifting the input samples by one, and using its output as the most recent input sample. Since the neural network is utilizing sigmoid function, we assume the traffic pattern is active if the generated output of the neural network is above the threshold of 0.5 and passive otherwise. A continuous sequence of learning is carried even after the network is trained considering the fact that the network can only predict a small number of iterations at any time independently before the output error exceeds the acceptable error bound ε.

The number of samples required for the first time training of the neural network depends on the complexity of the traffic pattern dynamics. The time complexity and the space complexity of the first order gradient-based back propagation scheme are respectively $\mathcal{O}(\iota N)$ and $\mathcal{O}(N)$ where N is the number of weighting functions in the network and ι is the number of iterations. Similar complexity terms for the BFGS back propagation scheme are respectively identified as $\mathcal{O}(\iota N)$ and $\mathcal{O}(N^2)$. However, the number of iterations ι for BFGS learning is usually an order of magnitude smaller than the similar quantity for gradient-based learning. Hence, the tradeoff between the two approaches is the better time complexity of BFGS learning versus better space complexity of the first order gradient-based learning.

Further, the complexity characteristics of both neural learning schemes are typically better than those of statistical modeling schemes such as fractional ARIMA processes or the complexity of calculating fractal dimensions such as correlation dimension. However, wide variations of ι prevent us from making a strong claim about the complexity advantage of our neural learning schemes compared to other modeling schemes. Nonetheless combining the straight forward way of implementation with the analysis of complexity, we claim that our proposed neural network schemes provide elegant approaches for the task of traffic forecasting. Further, it is important to note that the schemes of this section can be utilized to model any traffic pattern aside from the fact that the emphasis of our work is on self-similar traffic traffic forecasting.

IV. Multiple Source Queuing System

A. Queuing System Analysis

Our application testbed relies on a multiple source queuing system as illustrated by Fig. 2. The multiple source queuing system consists of a number of ON-OFF sources sharing an available buffer space. The sources can also be thought of as the arriving traffic patterns of the buffer. Depending on the choice of buffer management scheme, the queuing system can include dedicated partitions assigned to individual traffic patterns.

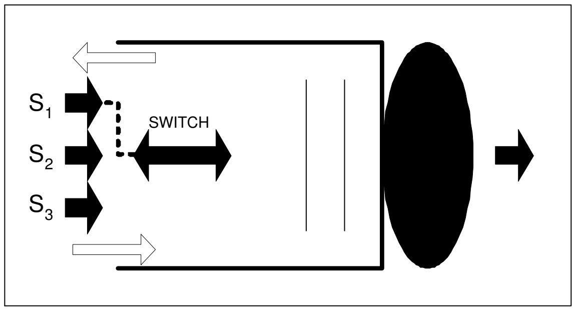

Fig. 2. The structure of a multiple source queuing system.

In our discussion, we view each individual source and its corresponding buffer as a separate First In First Out (FIFO) queuing system. Next, we provide a queuing analysis for each source within the multiple source queuing system. In our FIFO model, there is a finite capacity buffer storing arrived packets before they get transmitted. The occupancy of the buffer is determined by the flow of the arriving packets and the rate at which the packets are serviced. In this model, a queue is identified by its buffer capacity B, and its server capacity O_{max}. In each queue, the arrival 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 packets.

Using the following notation

- $I_k^{(i)}$: The input rate of the i-th buffer at time k.
- $O_k^{(i)}$: The output rate of the i-th buffer at time k.
- $Q_k^{(i)}$: The queuing rate of the i-th buffer at time k.
- $L_k^{(i)}$: The loss rate of the i-th buffer at time k.
- $B_k^{(i)}$: The queue size of the i-th buffer at time k.
- $B^{(i)}$: The buffer capacity of the i-th buffer.

the state of the queue for each buffer is specified by

$$I_k^{(i)} = O_k^{(i)} + Q_k^{(i)} + L_k^{(i)} \tag{10}$$

at any instant of time as shown in Fig. 3. Note that besides the values of $Q_k^{(i)}$ that could be positive or negative, all of the other values are always positive. Originally, the queue is empty. It

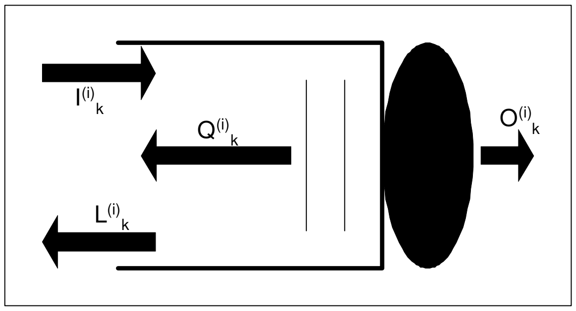

Fig. 3. The queuing diagram of the i-th source at time k.

begins to form when the buffer input rate exceeds the service rate. Hence, the queue rate $Q_k^{(i)}$ and the loss rate $L_k^{(i)}$ remain zero as long as the input rate is less than or equal the service rate, i.e.,

$$O_k^{(i)} = \begin{cases} I_k^{(i)} & : \; I_k^{(i)} < O_{max} \\ O_{max} & : \; I_k^{(i)} = O_{max} \end{cases} \tag{11}$$

The queue size $B_k^{(i)}$ 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. In this scenario, the total queuing rate is the difference between the input rate and the queue server capacity. The loss rate is zero at this stage. The queue keeps growing in size 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 packet loss rate as

$$L_k^{(i)} = I_k^{(i)} - O_k^{(i)} \tag{12}$$

with $O_k^{(i)} = O_{max}$. The effect of a change in the input rate is not immediately appeared if there are packets in the queue waiting to transmit. It is the queuing rate that changes according to

$$Q_{k+1}^{(i)} = Q_k^{(i)} + I_{k+1}^{(i)} - I_k^{(i)} \tag{13}$$

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

$$O_k^{(i)} = \begin{cases} O_{max} & : \quad B_k^{(i)} \geq 0 \\ I_k^{(i)} & : \quad B_k^{(i)} = 0 \end{cases} \tag{14}$$

The behavior of the queuing system described above is ruled by the $G/G/1$ queuing discipline. Analyzing the packet arrival rate of such a queuing system is a rather complicated task. Previously proposed techniques of analysis such as Lindley's Integral Equation described in [39] and Section 8.2 of [34] rely on stochastic theory approaches. The solution to Lindley's Integral Equation may be obtained under certain conditions by relying on spectrum factorization and transform theory techniques. As an alternative, we propose the use of our neural network schemes of Section III to learn the dynamics and forecast the packet arrival of the queuing system.

B. ON-OFF Source Analysis

Having described the multiple source queuing system utilized in our study, we now focus on the ON-OFF traffic pattern of individual sources. In our system, we represent the traffic pattern of a typical ON-OFF source by an artificially generated pattern. The traffic pattern can be generated by a single chaotic map or an aggregate of such maps. Generally speaking, an ON-OFF source is generating traffic at a peak rate when it is active and becomes active as soon as the state variable of the describing chaotic map goes beyond a threshold value. The source becomes passive as soon as the state variable goes below the threshold value. We utilize double intermittency map in our packet generation process as it generates a self-similar traffic pattern according to what is reported in [13]. The describing equation of double intermittency map is

$$x_{k+1} = \begin{cases} \epsilon_1 + x_k + c_1 x_k^m & : \quad 0 \leq x_k \leq d \\ -\epsilon_2 + x_k + c_2(1-x_k)^m & : \quad d \leq x_k \leq 1 \end{cases} \tag{15}$$

where x_k represents the discrete state variable of the map and the rest of the symbols represent various parameters with the property $c_1 = \frac{1-\epsilon_1-d}{d^m}$. Fig. 4 illustrates a sample drawing of double intermittency map. As observed in the figure, the iterative map requires multiple samples to move from one segment to another. We select initial conditions in the range of $x_0 \in [0.1, 0.3]$ along with a fixed threshold value of $d = 0.7$ and parameters $\epsilon_1 = 0.01$, $\epsilon_2 = 0.05$, $m = 5$, $c_1 = 1.73$, $c_2 = 267.49$ to obtain different traffic patterns for different sources. As an alternative, one may use different threshold values with fixed initial conditions to achieve varying traffic patterns.

It is proven in [6] that the binary trace of the autocorrelation of the double intermittency map with $\epsilon_1 = \epsilon_2 = 0$ decays slowly. However, it is unclear whether this property is preserved with nonzero albeit small values of ϵ_1 and ϵ_2 and whether the Hurst parameter is continuous. Nonetheless, numerical simulations seem to indicate that the Hurst parameter is continuous and that for small values of ϵ_1 and ϵ_2 the Hurst parameter can be approximated as $H = 1 = \frac{1}{m-1} = 0.75$. Identifying an approximation of the value of the Hurst parameter for traffic aggregates obtained from a set of double intermittency maps is not straightforward. However, our numerical experiments have identified the range $[0.72, 0.77]$ for aggregates of 30 to 50 sources.

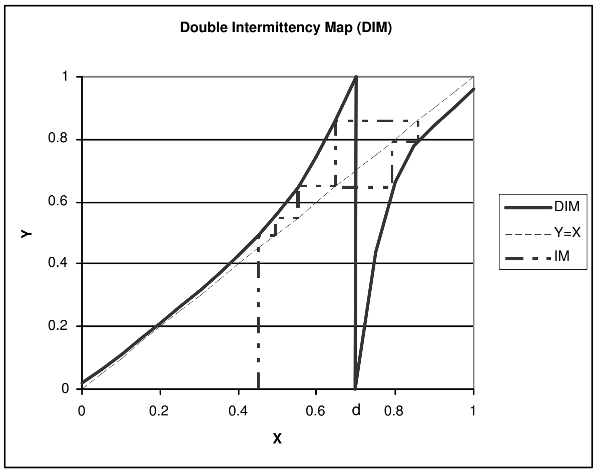

Fig. 4. A sample drawing of the double intermittency map. The legend IM indicates the path traversed by the map through consecutive iterations.

V. Buffer Management Utilizing Water-Filling

The main idea of our neural-based buffer management scheme revolves around partitioning the available space of a shared buffer among a number of traffic sources according to their traffic generation patterns. Forecasting of the traffic generation pattern can be done utilizing one of the neural network learning schemes of Section III.

Consider the general assignment of a given buffer space with capacity B_T among a number of sources as illustrated in Fig. 5. The buffer space is partitioned into a common portion with

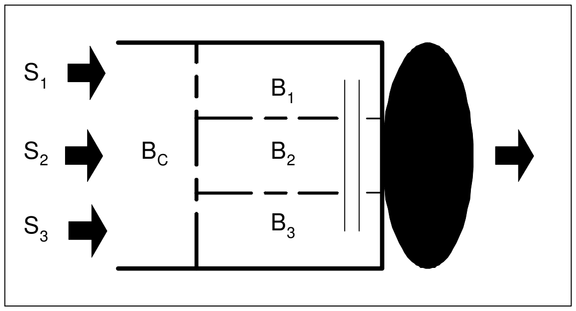

Fig. 5. General assignment of the buffer space (B_T) for a three source queuing system.

a fixed size B_C and a number of dedicated per source portions. We propose the use of water-filling approach to set the size of the dedicated portions of the buffer. We follow the notation of Section IV.A to describe the water-filling approach. For a given multiple source queuing system accommodating n sources, let us assume that source i is generating packets with the rate $I_k^{(i)}$

in a time epoch with length T, i.e., $k \in \{1, \cdots, T\}$. Further, assume that the queue size of the dedicated portion of the buffer to source i at the beginning of the epoch is given by $B_0^{(i)}$. Then the queue size of source i at time $k+1$ is described as

$$B_{k+1}^{(i)} = B_k^{(i)} + Q_k^{(i)} \tag{16}$$

where all of the quantities in Equation (16) are defined in Section IV.A. Source i is guaranteed not to experience any packet loss during the epoch if

$$B_k^{(i)} \leq B^{(i)} \quad \forall k \in \{1, \cdots, T\} \tag{17}$$

Hence, the requested dedicated buffer space of source i in the time epoch can be identified as

$$B^{(i)} = \max_{k \in \{1, \cdots, T\}} B_k^{(i)} \tag{18}$$

Defining $B_S \triangleq B_T - B_C$ as the remaining buffer space after assigning the common portion of the buffer and for an ordered set of $B^{(1)} \leq \cdots \leq B^{(n)}$, our proposed water-filling approach assigns a dedicated portion of the buffer space $b^{(i)}$ to source i as

Case 1: If $B_S \geq \sum_{j=1}^{n} B^{(j)}$

$$b^{(i)} = B^{(i)} \quad , \quad 1 \leq i \leq n \tag{19}$$

Case 2: If $B_S < \sum_{j=1}^{n} B^{(j)}$

$$b^{(i)} = \begin{cases} B^{(i)} & , \quad 1 \leq i \leq h \\ \frac{B_S - \sum_{j=1}^{h} B^{(j)}}{n-h} & , \quad h+1 \leq i \leq n \end{cases} \tag{20}$$

where h is an integer satisfying the following condition

$$\begin{array}{ccc} B^{(h)} \leq & \frac{B_S - \sum_{j=0}^{h} B^{(j)}}{n-h} & \leq B^{(h+1)} \\ 0 \leq & h & \leq n-1 \end{array} \tag{21}$$

for $B^{(0)} \triangleq 0$.

We observe that the water-filling approach of Equation (20) starts by dividing the remaining buffer space equally among all of the n sources until the first source reaches its requested buffer space $B^{(1)}$, then it fixes the assigned buffer space for the first source to $B^{(1)}$ and divides the new remaining buffer space among the remaining sources equally, and so on. Consequently, the sources with lower requested buffer space are more likely to receive their requested buffer space in full while the other sources receive equal shares of the remaining buffer space guaranteed not to be less than the assigned shares of the sources fully receiving their requested buffer space. We note that our proposed water-filling solution is max-min fair according to definition of [29]. The solution has a linear complexity and is hence quite practical from the implementation stand point. In [54], we also prove that the water-filling solution provided above is the solution to an optimal resource allocation problem for a class of piecewise linear utility functions. Because the same resource allocation problem can be applied to the current buffer management problem, we conclude that our proposed water-filling approach is optimal.

VI. Simulation Results

In order to investigate the performance of our proposed buffer management scheme, we utilize a triple source system. The traffic patterns of the first, second, and third source consist of an aggregate pattern generated by 30, 40, and 50 individual double intermittency map packet generators, respectively. We observe that the aggregate traffic patterns exhibit self-similar characteristics with Hurst parameters in the range of $[0.72, 0.77]$. We apply the proposed neural network forecasting schemes of Section III to predict the input rate of each buffer, i.e., $I_k^{(i)}$ with $i \in \{1, 2, 3\}$ over the discrete time k. We note that the volumes of traffic generated by different sources are not the same because of using a different number of per source packet generators.

First, we show that our proposed neural learning schemes are able to forecast self-similar traffic patterns. Fig. 6 illustrates and compares the number of generated packets by a single source double intermittency map and the trained neural network. We have utilized the first order gradient-based training scheme of Section III.A. Fig. 7 shows similar results for an aggregate source double intermittency map. Applying thousands of learning iterations, the single and aggregate traffic pattern have been followed within the specified error range for an average of 41 and 56 samples ahead, respectively. An interesting observation is that applying the neural network modeling scheme to an aggregate source traffic pattern consistently produces better results compared to a single source traffic pattern in terms of the number of accurate post training samples. The observation is expected as increasing the number of sources reduces the degree of self-similarity in an aggregate traffic pattern.

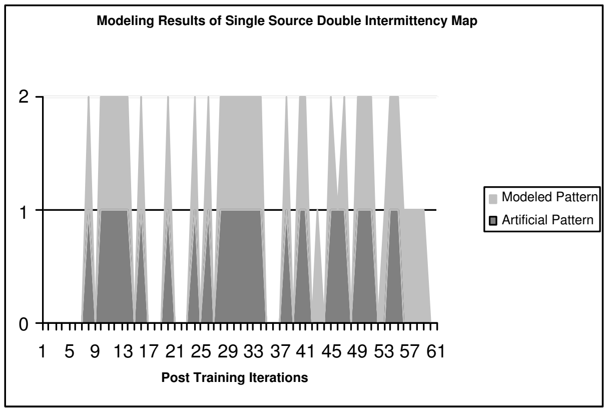

Fig. 6. First order neural network modeling results of a single source double intermittency map.

We note that the convergence of the learning scheme is time consuming because of the rich dynamics of the traffic pattern. In addition, all of the convergence results are strongly affected by the choice of initial conditions of the weighting functions and the minimum acceptable error bound ε. This is specially important as we have observed situations in which the optimal values of the weighting functions only reflect a local optimum rather than a global optimum. While this is expected due to the nature of the utilized learning algorithms, the effects typically

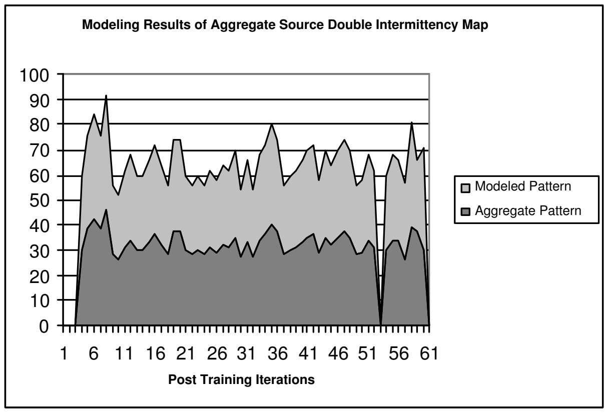

Fig. 7. First order neural network modeling results of a 50 aggregate source double intermittency map.

show in quicker divergence from the acceptable error bounds. Setting $\varepsilon = 0.1$ and the initial values of the weighting functions randomly between 0.01 and 0.09 yields best practical results while avoiding biasing and saturation. We have also observed the convergence of BFGS back propagation learning is eight to ten times faster than the first order gradient-based back propagation learning for the same choices of parameters. Further, its accuracy in terms of the number of samples followed in the post training phase is about 10% better than that of the first order back propagation learning. Another important consideration is that improving the accuracy of the forecasting process in terms of the number of samples could result in a better efficiency, it should not affect the fairness characteristics of the utilized scheme. In addition, we have utilized the hybrid linear/nonlinear learning scheme of [42]. However, we have not found much better convergence results compared to BFGS back propagation learning scheme.

Next, the traffic generated by each source is collected and sent to a shared buffer in a round robin manner. Depending on the buffer management scheme, the shared buffer may be partitioned into a common portion and three dedicated portions. In the latter case, the arriving packets of each source are first destined to the portion of the buffer dedicated to the source. Packets are only sent to the common portion of the buffer if the dedicated portion is full. Packets are lost if there is no space available either in the dedicated or the common portion. At the output of the buffer, the server utilizes either FTDM or STDM scheduling schemes to service the three dedicated portions of the buffer and the common portion.

We compare the performance of five different buffer management scenarios. In the first scenario, CS scheme is deployed, i.e., there is only one queue for all of the sources. It is associated with $B_C = B_T$ and $B_1 = B_2 = B_3 = 0$ in Fig. 5. The second scenario is a simple implementation of CP scheme in which the capacity of the buffer is distributed equally among the sources. It is associated with $B_C = 0$ and $B_1 = B_2 = B_3 = \frac{1}{3}B_T$ in Fig. 5. The third scenario is a simple implementation of SPS scheme that has three equal portions for the three sources with an additional common portion available to all of the sources. It is associated with $B_C \neq 0$ and $B_1 = B_2 = B_3 \neq 0$ in Fig. 5. The last two scenarios are the dynamic assignment of the buffer space relying on the water-filling result of Section V in conjunction with the neural forecasting schemes of Section III. It is important to note that the neural forecasting schemes are applied continuously, i.e., the training continues even after reaching the acceptable error bound. We refer to the neural forecasting schemes as DNS schemes. They are associated with the general case of Fig. 5. The last two scenarios are, in fact, generalizations of the third scenario keeping the common portion size fixed and adjusting the buffer space size of each source dynamically according to their traffic arrival patterns. Each of the last two methods have a potential to outperform the other buffer management schemes as they rely on forecasting the arriving traffic patterns. The process of utilizing our proposed dynamic buffer management scheme works as follows. We utilize an independent neural network per traffic pattern. Originally, we allow the neural networks to learn the dynamics of the underlying traffic patterns. During the original learning period, the dedicated portions of the buffer space are set according to the default values of the third scenario. Once the neural networks have learned the dynamics of the traffic patterns, we proceed with applying consecutive epochs of buffer space allocation. At the beginning of each epoch, individual portions of the buffer space are assigned proportional to the arrival pattern of the sources and utilizing the water-filling approach of Section V. The assignments remain in effect for as long as none of the conditions below are violated: (1) the forecasting errors remain within the acceptable threshold bound ε, (2) the number of samples predicted ahead is not passed the moving average of the accurately predicted samples in all of the previous epochs, and (3) the current epoch has not ended. If conditions (1) or (2) above are violated in the middle of the epoch, the dedicated portions of the buffer space are reset to the default values of SPS scenario for the rest of the epoch. To consider practical overhead of managing the buffer, we have selected an epoch length of 1000 samples. The dedicated portions are set according to the packet arrival pattern of the sources at the beginning of the next epoch and so on.

In order to evaluate the efficiency and fairness of different scenarios, we compare their overall and their most passive source loss rates together. Our experiments span over different choices of the buffer size with a fixed service rate and a moderately loaded queuing system. In our experiments, we rely on the same discrete time scales for both the neural network and the traffic generating intermittency maps.

Fig. 8 and Fig. 9 respectively show plots of total packet loss and the most passive source packet loss rate versus normalized buffer size diagram for the triple source queuing system in presence of FTDM scheduling algorithm. Fig. 10 and Fig. 11 show plots of the same quantities in presence of STDM scheduling algorithm. In the figures, we use the abbreviations DNS1 and DNS2 to denote first degree gradient-based dynamic neural sharing and BFGS dynamic neural sharing, respectively. The simulation results have been collected over ten million iterations per choice of buffer size.

It is clearly observed from the figures that for both FTDM and STDM scheduling algorithms under DNS1 and DNS2 scenarios, the total loss rate compared to CP scenario as well as per source loss rate compared to CS and/or SPS scenarios are re-

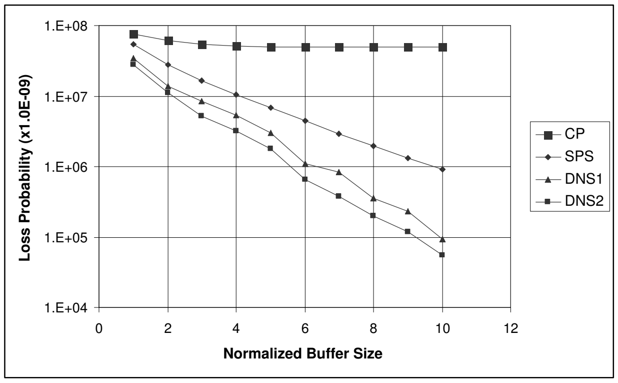

Fig. 8. Total packet loss rate versus buffer size diagram for the triple source queuing system using CP, SPS, DNS1, and DNS2 in presence of FTDM scheduling algorithm.

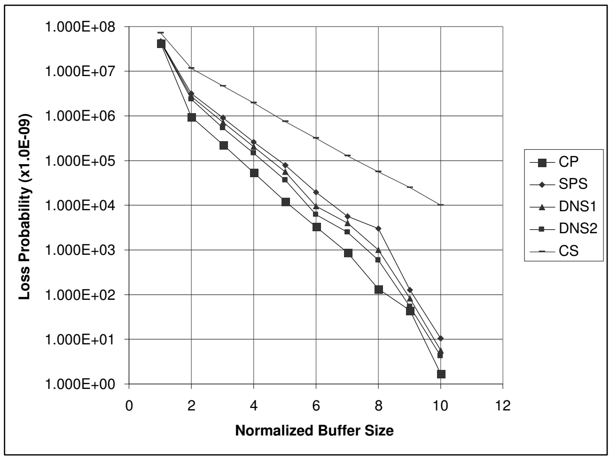

Fig. 11. Single source packet loss rate versus buffer size diagram for the triple source queuing system using CP, SPS, DNS1, DNS2, and CS in presence of STDM scheduling algorithm.

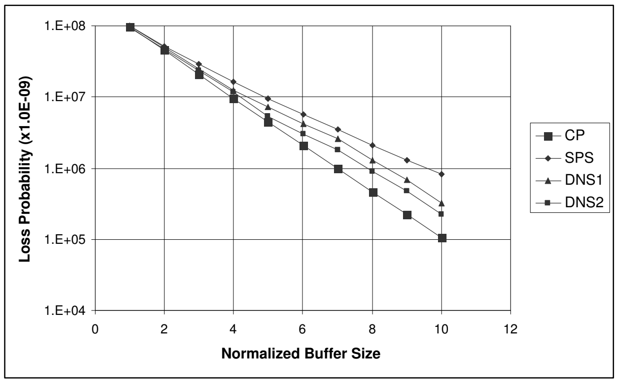

Fig. 9. Single source packet loss rate versus buffer size diagram for the triple source queuing system using CP, SPS, DNS1, and DNS2 in presence of FTDM scheduling algorithm.

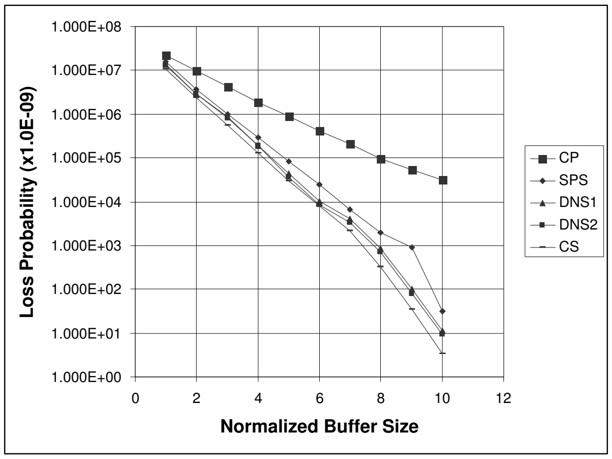

Fig. 10. Total packet loss rate versus buffer size diagram for the triple source queuing system using CP, SPS, DNS1, DNS2, and CS in presence of STDM scheduling algorithm.

duced. The results provided under DNS1 and DNS2 scenarios are interpreted as the evidence that the tradeoff between fairness and efficiency has been addressed. Comparing the results of SPS, DNS1, and DNS2 scenarios show the higher efficiency of the latter two methods. We also observe that both fairness and efficiency characteristics of the results of DNS2 scenario outperform the results of DNS1 scenario. As mentioned before, the price is the higher space complexity of DNS2 compared to that of DNS1.

An important observation is that reducing the epoch length decreases the overall and single source loss rates of both DNS methods at a higher cost of buffer management. For an epoch length of 50 DNS1 and DNS2 loss rates are almost matching the overall and single source loss rates of CS and CP, respectively. Utilizing the standard variant of DPO buffer management scheme also leads to efficiency and fairness characteristics similar to the case of utilizing CS and CP, respectively. Utilization of the other variants of DPO typically leads to trading a lower overhead of implementation with a lower loss performance. Further, we note that the performance of different methods are very different as the result of applying different methods for traffic management of a heavily utilized system.

In addition, it is worth mentioning that the results of the plots may resemble short-range dependent plots in which logarithmic loss curves drop linearly with the increase of the buffer size. As indicated by [17] and other research articles, the plots of loss versus buffer size in the case of self-similar traffic patterns are expected to become flat for an increase in the buffer size beyond a certain threshold. As indicated previously, the range of measured Hurst parameters indicate that the traffic patterns are self-similar. The observations are mostly justified by the moderate levels of server utilization. For heavily utilized servers operating close to full capacity, the observations are consistent with the previously reported observations.

In what follows, we discuss some of the important aspects of the implementation of our experiments. We start by commenting on the choice of our neural network type and structure. We considered perceptron, Hopfield, and Kohonen networks and selected fixed-structure perceptron neural networks due to simplicity of implementation. The number of neurons in each layer reflects our best overall practical findings leading to a balance between complexity and accuracy. We observed that there is a significant improvement in the learning performance when going from a single hidden layer structure to a double hidden layer structure. However, increasing the number of hidden layers beyond two does not have the same effect and is thus not justified considering the overhead of calculations. Further, small variations in the number of neurons of the input and each of the two hidden layers do not have the same effect on the learning performance. We anticipate that the best choice of the structure should be related to the degree of self-similarity captured through the measure of the Hurst parameter or another quantity.

We close this section by mentioning that the results presented in our current work point to some predictability of traffic that can be viewed as a source of contradiction with the results presented in [46] and [47]. We point out that besides the difference in the nature of traffic traces obtained from artificial traffic generators and TCP traffic simulators, the key difference is that the utilized neural network of our study represents a time-varying system considering continuous readjustment of the weighting functions where as the results of those articles are obtained from a set of stationary models.

VII. Conclusion and Future Work

In this paper, we provided dynamic buffer management schemes as an application of adaptive neural learning systems. We utilized two different learning schemes for a fixed structure perceptron neural network to forecast teletraffic patterns. Our proposed schemes were the first order gradient-based learning and BFGS Quasi-Newton learning. Based on our forecasting results, we provided dynamic buffer management schemes to improve the loss performance of static partial sharing buffer management scheme while considering the fairness issue. Our dynamic buffer management schemes relied on the water-filling approach. Our experimentation utilized a multiple source queuing system accommodating artificially generated self-similar traffic patterns. We 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 statistical time division multiplexing. We concluded that our dynamic neural sharing schemes were able to offer the best solutions considering the trade off between fairness and loss issues as well as practicality of implementation.

We note that our dynamic neural sharing schemes are best suited for the class of sources with a life span exceeding the duration of the original learning period. In order to apply our schemes to short-lived sources, we are investigating potential ways of improving the speed of standard traffic learning schemes. We are also studying the applicability of statistical offline learning methods to the dynamics of short-lived sources. Our future work further aims at applying our schemes to a set of real traffic traces.

References

[1] —, "Dynamic Queue Length Thresholds for Multipriority Traffic," In Proc. of ITC-15, V. Ramaswami and P. Wirth, Eds., 1997.
[2] A. Adas, "Traffic Models in Broadband Networks," IEEE Communications Magazine, July 1997.
[3] R.G. Addie, M. Zukerman, T. Neame, "Fractal Traffic: Measurements, Modelling and Performance Evaluation," In Proc. of IEEE INFOCOM, 1995.
[4] A. Alkhatib, M. Krunz, "Application of Chaos Theory to the Modeling of Compressed Video," In Proc. of IEEE ICC, 2000.
[5] G. Ascia , V. Catania , D. Panno, "An Adaptive Fuzzy Threshold Scheme for High Performance Shared-Memory Switches," In Proc. of ACM symposium on Applied computing, 2001.
[6] M. Barenco, D.K. Arrowsmith, "The Autocorrelation of Double Intermittency Maps and the Simulation of Computer Packet Traffic," DYNAMICAL SYSTEMS, 19(1) 2004, pp. 61-74.
[7] J. Beran, R. Sherman, M.S. Taqqu, W. Willinger, "Variable Bit Rate Video Traffic and Long Range Dependence," IEEE/ACM Trans. on Networking, April 1994.
[8] D.P. Bertsekas, "Nonlinear Programming, 2nd Edition," Athena Scientific Publishing, ISBN 1886529000, 1999.
[9] S. Chong, S.Q. Li, J. Ghosh, "Predictive Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM," IEEE JSAC, January 1995.
[10] A.K. Choudhury, E.L. Hahne, "Dynamic Queue Length Thresholds for Shared-Memory Packet Switches," IEEE/ACM Trans. on Networking, April 1998.
[11] D.E. Duffy, W. Willinger, "Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks," IEEE JSAC, 1994.
[12] A. Elwalid, D. Mitra, "Design of Generalized Processor Sharing Schedulers which Statistically Multiplex Heterogeneous QoS Classes," In Proc. of IEEE INFOCOM, 1999.
[13] A. Erramilli, R.P. Singh, P. Pruthi, "Chaotic Maps as Models of Packet Traffic," In Proc. of ITC-14, 1994.
[14] S.E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks," Technical Report CMU-CS-88-162, Carnegie Mellon University, June 1988.
[15] V. Firoiu, J. Kurose, D. Towsley, "Efficient Admission Control for EDF Schedulers," In Proc. of IEEE INFOCOM, 1997.
[16] A. Feldmann, A.C. Gilbert, W. Willinger, "Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic," In Proc. of ACM SIGCOMM, September 1998.
[17] N.L.S. Fonseca, G.S. Mayor, C.A.V. Neto, "On the Equivalent Bandwidth of Selfsimilar Sources," ACM Trans. on Modeling and Computer Simulation, April 2000.
[18] G.J. Foschini, B. Gopinath, "Sharing Memory Optimally," IEEE Trans. on Communications, March 1983.
[19] G. Gallasi, C. Rigolio, "ATM Bandwidth Assignment and Bandwidth Enforcement Policies," In Proc. of IEEE GLOBECOM, 1987.
[20] L. Georgiadis, I. Cidon, R. Guerin, A. Khamisy, "Optimal Buffer Sharing," IEEE JSAC, September 1995.
[21] M. Gerla, L. Kleinrock, "Flow Control: A Comparative Survey," IEEE Trans. on Communications, April 1980.
[22] G. Gomes, N.L.S. Fonseca, N. Agoulmine, J.N. Souza, "Neurocomputation of the Hurst Parameter," In Proc. of IEEE ITS, 2002.
[23] E.L. Hahne , A.K. Choudhury, "Dynamic Queue Length Thresholds for Multiple Loss Priorities," IEEE/ACM Trans. on Networking, June 2002.
[24] S. Haykin, "Neural Networks: A Comprehensive Foundation, 2/E" ISBN 0132733501, 1998.
[25] K.M. Hellstern, P. Wirth, "Traffic Models for ISDN Data Users: Office Automation Application," In Proc. of ITC-13, 1991.
[26] M.G. Hluchyj, M.J. Karol, "Queueing in High-Performance Packet Switching," IEEE JSAC, December 1988.
[27] G.U. Hwang , B.D. Choi, "Performance Analysis of the DAR(1)/D/C Priority Queue under Partial Buffer Sharing Policy," Computers and Operations Research, November 2004.
[28] M.I. Irland, "Buffer Management in a Packet Switch," IEEE Trans. on Communications, March 1978.
[29] J.M. Jaffe, "Bottleneck Flow Control," IEEE Trans. on Communications, July 1981.
[30] G. Latouche, "Exponential Servers Sharing a Finite Storage: Comparison of Space Allocation Policies," IEEE Trans. on Communications, June 1980.
[31] F. Kamoun, L. Kleinrock, "Analysis of Shared Storage in a Computer Network Node Environment under General Traffic Conditions," IEEE Trans. on Communications, July 1980.
[32] P. Kermani, L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," Computer Networks, Vol.3, 1979.
[33] A. Kesselman , Y. Mansour, "Harmonic Buffer Management Policy for Shared Memory Switches," Theoretical Computer Science, September 2004.

[34] L. Kleinrock, "Queuing Systems: Volume I: Theory," John Wiley & Sons, ISBN 0471491101, 1974.

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

[36] W.E. Leland, W. Willinger, M.S. Taqqu, D.V. Willson, "On the Self-Similar Nature of Ethernet Traffic," IEEE/ACM Trans. on Networking, February 1994.

[37] A. Lin, J.A. Silvester, "Priority Queuing Strategies and Buffer Allocation Protocols in Traffic Control at an ATM Integrated Broadband Switching System," IEEE JSAC, December 1991.

[38] Y.S. Lin, C.B. Shung, "Quasi-Pushout Cell Discarding," IEEE Communications Letters, September 1997.

[39] D.V. Lindley, "The Theory of Queues with A Single Server," In Proc. of Cambridge Phil. Society, Vol. 48, 1952.

[40] S.H. Low, "Equilibrium Bandwidth and Buffer Allocations for Elastic Traffics," IEEE/ACM Trans. on Networking, June 2000.

[41] G.S. Mayor, J.A. Silvester, "Time Scale Analysis of An ATM Queueing System with Long-Range Dependent Traffic," In Proc. of IEEE INFOCOM, 1997.

[42] S. McLoone, M.D. Brown, G. Irwin, A. Lightbody, "A Hybrid Linear/Nonlinear Training Algorithm for Feedforward Neural Networks," IEEE Trans. on Neural Networks, July 1998.

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

[44] J.M. Pitts, L.G. Cuthbert, M. Bocci, E.M. Scharf, "An Accelerated Simulation Technique for Modeling Burst Scale Queuing Behavior in ATM," In Proc. of ITC-14, 1994.

[45] G. Sakalauskiene, "The Hurst Phenomenon in Hydrology," Available at http://www.apini.lt/lt/Zurnalas/LT/25%20zurnalas/03-Gaudentos_red.pdf

[46] K. Shah, S. Bohacek, E.A. Jonckheere, "On Predictability of Data Network Traffic," In Proc. of IEEE ACC, 2003.

[47] K. Shah, S. Bohacek, E.A. Jonckheere, "On the Performance Limitation of Active Queue Management (AQM)," In Proc. of IEEE Conference on Decision and Control, 2004.

[48] A.K. Thareja, A.K. Agarwala, "On the Design of Optimal Policy for Sharing Finite Buffers," IEEE Trans. on Communications, June 1984.

[49] A.K. Thareja, S.K. Tripathi, "Buffer Sharing in Dynamic Load Environment," In Proc. of IEEE INFOCOM, 1984.

[50] D. Tipper, M.K. Sundareshan, "Adaptive Policies for Optimal Buffer Management in Dynamic Load Environments," In Proc. of IEEE INFOCOM, 1988.

[51] S.X. Wei, E.J. Coyle, M.T. Hsiao, "An Optimal Buffer Management Policy for High-Performance Packet Switching," In Proc. of IEEE GLOBECOM, 1991.

[52] G.-L. Wu, J.W. Mark, "A Buffer Allocation Scheme for ATM Networks: Complete Sharing Based on Virtual Partition," IEEE/ACM Trans. on Networking, December 1995.

[53] J.P. Yang, "Performance Analysis of Threshold-Based Selective Drop Mechanism for High Performance Packet Switches," Performance Evaluation, May 2004.

[54] H. Yousefi'zadeh, F. Fazel, H. Jafarkhani, "Hybrid Unicast and Multicast Flow Control: A Linear Optimization Approach," In Proc. of IEEE/IEE High Speed Networks and Multimedia Communications (HSNMC), 2004. Extended Version Available at http://www.ece.uci.edu/˜ hyousefi/publ/fcHSNMC.pdf.