A Neuro-Forecast Water-Filling Scheme of Server Scheduling

Homayoun Yousefi'zadeh
Department of EECS
University of California, Irvine
hyousefi@uci.edu

***Abstract*— Dynamic server scheduling schemes in queuing systems accommodating delay-sensitive traffic need to address the tradeoff between efficiency and fairness. For delay-sensitive traffic, threshold-exceeding delay or equivalently loss is used as a measure of efficiency. In this paper, a pair of dynamic server scheduling schemes for queuing systems accommodating delay-sensitive traffic are compared. Each scheme consists of two components. The first component attempts at forecasting the arriving traffic patterns of the sources sharing the server bandwidth and the second component makes the assignment of server bandwidth among the sources. The schemes utilize BFGS and resilient back-propagation learning in perceptron neural networks to forecast the arriving traffic patterns, respectively. Once the traffic patterns are forecast, the schemes rely on water-filling to make the server bandwidth assignments max-min fair. Our simulations reveal the efficiency and fairness characteristics of the schemes.**

***Index Terms*— Neuro-Forecasting, Water-Filling, Server Scheduling, Delay, Loss, Fairness.**

I. Introduction

The adaptive learning power of neural networks has proven useful in various contexts of the literature of 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) [3]. Systems with neural network building blocks are robust in the sense that occurrence of small errors does not interfere with their proper operation. This characteristic of neural networks makes them quite suitable for forecasting traffic patterns.

Reducing queuing delay and packet loss is an important design issue of traffic control algorithms. For delay-sensitive traffic, queuing delay has a one-to-one relationship with packet loss and is usually viewed as a measure of efficiency. For shared queuing systems, fairness needs to be taken into consideration as a trade off factor. We refer to fairness as a mean of providing each individual source with the ability to take advantage of an appropriate portion of the available resources. Examples of such resource include server bandwidth and/or buffer space.

Fixed Time Division Multiplexing (FTDM) and Statistical Time Division Multiplexing (STDM) are arguably two of the most important server scheduling schemes of the literature of communication networks [15], [22]. While in FTDM each source takes advantage of a fair portion of the server bandwidth and there is no bandwidth sharing, in STDM the unused portion of the bandwidth assigned to each source might be used to service packets generated by other sources.

As described in [19], [11], [16], [17], and [9], buffer management schemes are classified under three main categories. 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 a common shared and dedicated portions of the buffer space. A survey of the literature shows that CS achieves optimal throughput-delay performance. However, it does not perform well when accommodating greedy sources. The work of [13], [8], and [19] 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. 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 [23], [24], [10], and [4] are all classified under Dynamic Push Out (DPO) which is a variant of dynamic buffer management schemes.

Other server scheduling and buffer allocation 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, [7], [26], and [5] provide an appropriate overview of the latter techniques, respectively. In [27] and [12], performance analysis studies of a number of buffer management schemes are provided. The tradeoff between the available bandwidth and buffer space is studied in [20]. The work of [18] and [1] are among recent literature articles providing a theoretical and an intelligent treatment of the buffer management problem, respectively. Our works of [29] and [31] are also classified under dynamic buffer management schemes. They can outperform the SPS scheme of [19] in terms of loss-fairness tradeoff. Our work of [28] applies gradient-based back-propagation learning to forecast the overall queuing delay of a shared buffer.

In this paper, we focus on the issue of server scheduling rather than buffer management for delay-sensitive traffic. We introduce a family of server scheduling schemes classified under dynamic STDM schemes. The schemes rely on neuro-forecasting methods applied to individual arriving patterns of a number of delay-sensitive traffic sources sharing a single server. Once the traffic patterns are forecast, the schemes utilize water-filling technique to dynamically adjust the assignment of server bandwidth. In this paper, we use the terms server bandwidth and service rate interchangeably. Our schemes can be applied to both fixed- and variable-length packets.

An outline of the paper follows. In Section II, we describe our proposed neural network forecasting schemes of teletraffic patterns. In Section III, we discuss the details of our server scheduling scheme. In Section IV, we compare the performance of our proposed schemes together and with FTDM server scheduling scheme. We conclude this paper in Section V.

II. Neuro-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 with two states. Such pattern is delivering traffic at a peak rate in its active state and is silent in its passive state. VBR video sources [2] are among the examples of ON-OFF traffic sources generating delay-senstivie traffic patterns. In this section, we propose neural-forecasting techniques of ON-OFF traffic patterns. Our techniques utilize second order BFGS [25] and resilient back-propagation learning [21].

We propose the use of a fixed structure, fully connected, feedforward perceptron neural network for the task of forecasting. The perceptron network of our study consists of an input layer with six neurons, two hidden layers with twenty neurons in each layer, and an output layer with one neuron. The number of neurons in each layer reflects our best practical findings leading to a balance between complexity and accuracy. In our perceptron network, a neuron transfers its output as

$$x_j^{(s)} = f(\sum_i w_{ij}^{(s)}.x_i^{(s-1)}) = f(\Psi_j^{(s)}) \tag{1}$$

where $x_j^{(s)}$ is the present output state of the j-th neuron in layer s and $w_{ij}^{(s)}$ is the weighting function between the j-th neuron in layer $s-1$ and the i-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

$$f(z) = \frac{1}{1+e^{-z}} \tag{2}$$

The learning process of the neural network is nothing more than minimizing its error function. The error function of the network at iteration k of the learning process is defined as

$$E[k] = \frac{1}{2}\left(y[k] - \vartheta[k]\right)^2 \tag{3}$$

where $\vartheta[k]$ indicates the present output of the network to a given input $u[k]$ and $y[k]$ corresponds to the actual output.

In our previous works of [29] and [31], we proposed the use of first and second order BFGS gradient-based back-propagation learning scheme. We noted that at the cost of higher complexity, a second order BFGS gradient-based method could outperform a first order gradient-based method in terms of learning speed. In our current work, we propose the use of resilient back-propagation [21] in addition to second order Quasi-Newton BFGS learning scheme [25].

The details of BFGS back-propagation learning scheme are described in [31]. Similar to the BFGS back-propagation learning scheme, the resilient 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. However, it attempts at filtering out the blurry effects of the gradient behavior.

In iteration $k+1$ of resilient back-propagation learning, the individual update value of each weighting function is introduced adaptively.

The adaptive update value evolves during the learning process based on observing the error function $E[k]$ of iteration k and $E[k-1]$ of iteration $k-1$ as indicated by the following learning rule:

$$\Delta_{ij}[k] = \begin{cases} \eta^+.\Delta_{ij}[k-1], & \text{if} \quad \frac{\partial E[k-1]}{\partial w_{ij}}.\frac{\partial E[k]}{\partial w_{ij}} > 0 \\ \eta^-.\Delta_{ij}[k-1], & \text{if} \quad \frac{\partial E[k-1]}{\partial w_{ij}}.\frac{\partial E[k]}{\partial w_{ij}} < 0 \\ \Delta_{ij}[k-1], & \text{Otherwise} \end{cases} \tag{4}$$

where $0 < \eta^- < 1 < \eta^+$. In other words, the adaptive learning rule monitors the changes in the sign of $\frac{\partial E[k-1]}{\partial w_{ij}}.\frac{\partial E[k]}{\partial w_{ij}}$. Every time there is a sign change indicating the last update value was too large, the update value is decreased by a factor η^-. if there is no sign change, the update value is slightly increased in order to accelerate the convergence speed. Once the update value for each weighting function is adjusted, the weight-update follows these simple rules: (1) if the error is increasing as indicated by a positive derivative, the weight is decreased by its update value, and (2) if the error is decreasing as indicated by a negative derivative, the weight is increased by its update value. The latter is summarized as:

$$\Delta w_{ij}[k] = \begin{cases} -\Delta_{ij}[k], & \text{if} \quad \frac{\partial E[k]}{\partial w_{ij}} > 0 \\ \Delta_{ij}[k], & \text{if} \quad \frac{\partial E[k]}{\partial w_{ij}} < 0 \\ 0, & \text{Otherwise} \end{cases} \tag{5}$$

However, there is one exception to the rule, i.e., the previous weight-update is reverted under the following condition.

$$\Delta w_{ij}[k] = -\Delta w_{ij}[k-1], \quad \text{if} \quad \frac{\partial E[k-1]}{\partial w_{ij}}\frac{\partial E[k]}{\partial w_{ij}} < 0 \tag{6}$$

Iteration k of our proposed resilient back-propagation learning scheme is then described as

```
For All Weights {
    if ($\frac{\partial E[k-1]}{\partial w_{ij}}.\frac{\partial E[k]}{\partial w_{ij}} > 0$) then {
        $\Delta_{ij}[k] = \min(\eta^+.\Delta_{ij}[k-1], \Delta_{max})$
        $\Delta w_{ij}[k] = -\text{sign}(\frac{\partial E[k]}{\partial w_{ij}}.\Delta_{ij}[k])$
        $w_{ij}[k+1] = w_{ij}[k] + \Delta w_{ij}[k]$
    }
    else if ($\frac{\partial E[k-1]}{\partial w_{ij}}.\frac{\partial E[k]}{\partial w_{ij}} < 0$) then {
        $\Delta_{ij}[k] = \max(\eta^-.\Delta_{ij}[k-1], \Delta_{min})$
        $w_{ij}[k+1] = w_{ij}[k] - \Delta w_{ij}[k-1]$
        $\frac{\partial E[k]}{\partial w_{ij}} = 0$
    }
    else if ($\frac{\partial E[k-1]}{\partial w_{ij}}.\frac{\partial E[k]}{\partial w_{ij}} = 0$) then {
        $\Delta w_{ij}[k] = -\text{sign}(\frac{\partial E[k]}{\partial w_{ij}}.\Delta_{ij}[k])$
        $w_{ij}[k+1] = w_{ij}[k] + \Delta w_{ij}[k]$
    }
}
```

The flow of information for both learning methods is as follows. 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 $\vartheta[k]$. During the propagation process, all of the combined inputs Ψ_j and output states x_j for each neuron are set. In the adjustment process of the weighting functions, the learning schemes propagate the output layer error to the preceding layer via the existing connections and repeat the operation until reaching the input layer. In other words, output error moves from the output layer -just in the opposite direction of the movement of the original information- one layer at a time until reaching the input layer. In iteration k of the learning, the neural network input vector $u[k]$ consists of samples $\vartheta[k-8]$ through $\vartheta[k-1]$ of the actual traffic pattern. The difference between sample $\vartheta[k]$ of the actual traffic pattern and the neural network output $y[k]$ is then used to adjust the weighting functions of the network accordingly. In the next iteration, sample $\vartheta[k-8]$ of the actual traffic pattern is discarded, samples $\vartheta[k-7]$ through $\vartheta[k]$ of the actual traffic pattern are used as the new input vector, and sample $\vartheta[k+1]$ is used as the new actual output. The neural network continues processing more information in consecutive iterations of the learning phase until the error $E[k]$ is less than a specified bound ε. Once the error is within the specified 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 dynamics of the traffic pattern. The time complexity and the space complexity of BFGS back-propagation scheme of [29] are respectively $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ where N is the number of weighting functions in the network and ι is the number of iterations. Similar complexity terms for resilient back-propagation scheme are respectively identified as $\mathcal{O}(\iota N)$ and $\mathcal{O}(N)$. However, the number of iterations ι for resilient back-propogation learning is usually smaller than the similar quantity for BFGS learning.

III. Server Scheduling Utilizing Water-Filling

The main idea of our server scheduling scheme revolves around partitioning the available server bandwidth of a shared buffer among a number of traffic sources according to their traffic generation patterns. Prediction of the traffic generation pattern can be done utilizing the neuro-forecasting schemes of Section II.

Consider the general assignment of a single server bandwidth with capacity C_B in bits per time unit for the queuing system illustrated in Fig. 1. Packet sizes are assumed to be fixed and L bits long. Therefore, the server capacity in packets per time unit is expressed as $C_P = \lfloor \frac{C_B}{L} \rfloor$. The buffer space is shared following CP scheme. Further, the buffer space is assumed to be large enough warranting that there would be no loss due to buffer overflow. However, each source can accrue loss as the result of delay, i.e., packets arrived within a time epoch have to be serviced or else they are lost.

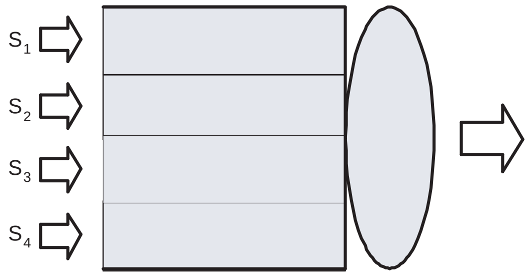

Fig. 1. Quad source queuing system used for assignment of the server bandwidth.

We propose the use of water-filling approach to assign the available server bandwidth among the sources. We utilize the following notation during an epoch with length K to describe the water-filling approach.

- $I^{(i)}[k]$: The input rate of the i-th source at time k.
- $O^{(i)}[k]$: The output rate of the i-th source at time k.
- $Q^{(i)}[k]$: The queuing rate of the i-th source at time k.
- $B^{(i)}[k]$: The number of queued packets of the i-th source at time k.
- $L^{(i)}$: The loss rate of the i-th source at the end of epoch.
- $B^{(i)}$: The dedicated server bandwidth of the i-th source.

For a given multiple source queuing system accommodating n sources, let us assume that source i is generating packets with the rate $I^{(i)}[k]$ during a time epoch with length K, i.e., $k \in \{1, \cdots, K\}$. Then, the queue size of source i at time $k+1$ is described as

$$B^{(i)}[k+1] \;=\; B^{(i)}[k] + Q^{(i)}[k] \tag{7}$$

The loss rate of source i at the end of epoch is identified as

$$L^{(i)} \;=\; \sum_{k=1}^{K} I^{(i)}[k] \;-\; \sum_{k=1}^{K} O^{(i)}[k] \tag{8}$$

Source i is guaranteed not to experience any packet loss due to delay by the end of epoch if

$$\sum_{k=1}^{K} I^{(i)}[k] \;=\; \sum_{k=1}^{K} O^{(i)}[k] \tag{9}$$

Hence, source i requests a dedicated server bandwidth in per packet per time unit in a given time epoch as

$$B^{(i)} \;=\; \frac{\sum_{k=1}^{K} I^{(i)}[k]}{K} \tag{10}$$

For an ordered set of $B^{(1)} \leq \cdots \leq B^{(n)}$, our proposed water-filling approach assigns a dedicated portion of the server bandwidth $b^{(i)}$ to source i during the given epoch as

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

$$b^{(i)} = B^{(i)} \quad , \quad 1 \leq i \leq n \tag{11}$$

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

$$b^{(i)} = \begin{cases} B^{(i)} & , \quad 1 \leq i \leq h \\ \frac{C_P - \sum_{j=1}^{h} B^{(j)}}{n-h} & , \quad h+1 \leq i \leq n \end{cases} \tag{12}$$

where h is an integer satisfying the following condition

$$\begin{array}{ccc} B^{(h)} \leq & \frac{C_P - \sum_{j=0}^{h} B^{(j)}}{n-h} & \leq B^{(h+1)} \\ 0 \leq & h & \leq n-1 \end{array} \tag{13}$$

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

We observe that the water-filling approach of Equation (12) starts by dividing the server bandwidth equally among all of the n sources until the first source reaches its requested server bandwidth $B^{(1)}$, then it fixes the assigned server bandwidth for the first source to $B^{(1)}$ and divides the remaining unallocated server bandwidth among the remaining sources equally, and so on. Consequently, the sources with lower requested server bandwidth are more likely to receive their requested server bandwidth in full while the other sources receive equal shares of the remaining server bandwidth guaranteed not to be less than the assigned shares of the sources fully receiving their requested server bandwidth. We note that our proposed water-filling solution is max-min fair according to definition of [14]. The solution has a linear complexity and is hence quite practical from an implementation stand point. In [30], 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 bandwidth allocation problem, we conclude that our proposed water-filling approach is optimal.

IV. Simulation Results

In this section, we evaluate the performance of our server scheduling method by applying it to a quad source system where the traffic of the sources consist of artificial traffic patterns generated by 10, 20, 30, and 60 individual double intermittency map packet generators. In order to generate self-similar traffic patterns, we utilize double intermittency map [6]. According to what is reported in [6], the map generates a self-similar traffic pattern. An individual double intermittency map generates 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. The describing equation of double intermittency map is

$$x[k+1] = \begin{cases} \epsilon_1 + x[k] + c_1 x[k]^m & : \quad 0 \leq x[k] \leq d \\ -\epsilon_2 + x[k] + c_2(1 - x[k])^m & : \quad d \leq x[k] \leq 1 \end{cases} \tag{14}$$

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. 2 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. The self-similar traffic pattern generated by a number of such maps can be thought of as a delay-sensitive VBR video pattern.

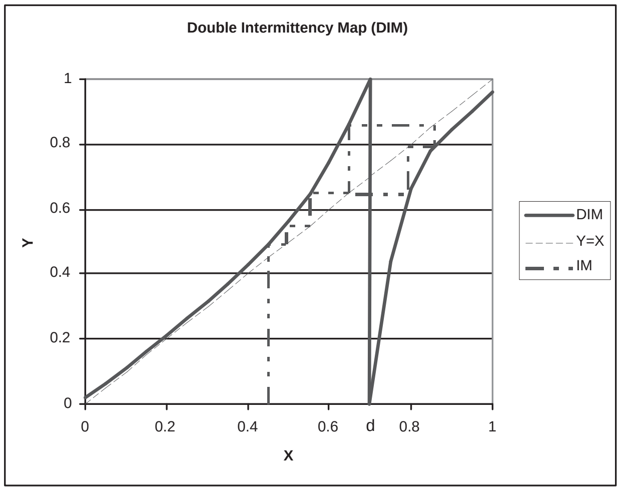

Fig. 2. A sample drawing of the double intermittency map. The legend IM indicates the path traversed by the map through consecutive iterations.

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. The traffic generated by each source is collected and sent to a shared buffer. Based on utilizing the CP buffer allocation scheme, the shared buffer is partitioned into four per source dedicated portions. Arriving packets are sent to the dedicated portion allocated to the source. While delay-sensitive packets are assumed not to be lost due to buffer overflow, they are lost if not serviced by the end of time epoch. At the output of the buffer, our proposed server scheduling scheme is applied.

We compare the performance of three different server scheduling schemes, namely, FTDM and two dynamic STDM schemes. We note that in FTDM, a round-robin service scheduling scheme is applied to four dedicated buffers. Server bandwidth will be wasted if there is no packet ready to be transmitted. In the two STDM schemes, the server bandwidth is assigned relying on BFGS or resilient back-propagation neuro-forecasting schemes in conjunction with the water-filling result of Section III. Each of the last two methods has a potential to outperform FTDM scheme by relying on forecasting the arriving traffic patterns. The process of utilizing our proposed dynamic server scheduling scheme works as follows. We utilize an independent neural network per an aggregated traffic pattern. Originally, we allow the neural networks to learn the dynamics of the underlying traffic patterns. During the original learning period, FTDM server scheduling is utilized. Once the neural networks have learned the dynamics of the traffic patterns, we proceed with applying consecutive epochs of server scheduling. At the beginning of each epoch, individual portions of the server bandwidth are assigned proportional to the arrival pattern of the sources and utilizing the water-filling approach of Section III. The assignments remain in effect for as long as none of the conditions below is 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 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 FTDM scheme for the rest of the epoch. Server bandwidth is assigned 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. Once more, we only consider delay-related loss events, i.e., packets that are not transmitted at the end of epoch. Our experiments span over different choices of 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. 3 and Fig. 4 respectively show plots of total packet loss and the most passive source packet loss rate versus normalized service rate for the quad source queuing system. The abbreviations BF-STDM and RB-STDM in the figures are used to denote our proposed water-filling STDM schemes utilizing BFGS learning and resilient back-propagation learning, respectively.

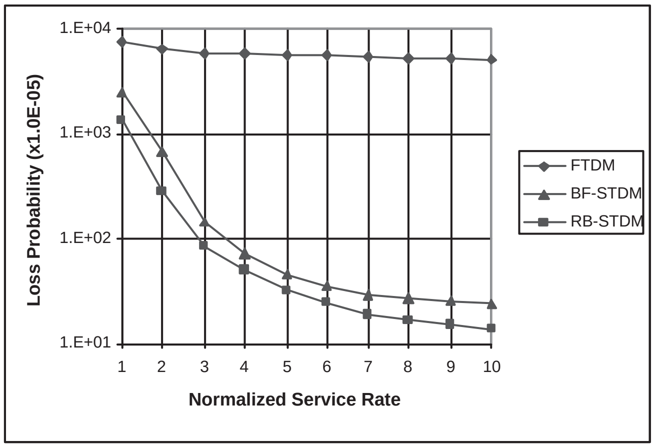

Fig. 3. Plots of total packet delay-caused loss versus normalized service rate for the quad source queuing system using FTDM, BF-STDM, and RB-STDM server scheduling.

The packetized simulation results have been obtained from an iterative algorithm with a total number of one million iterations per choice of server bandwidth. In each case, the fit is taken over the discrete intervals with acceptable accuracy of prediction. To consider practical overhead of managing the buffer, we select an epoch length of 5000 samples.

It is clearly observed from the figures that under BF-STDM and RB-STDM schemes, the total and per source loss rate compared to FTDM scheme are reduced. The results shown under our dynamic STDM scenarios are interpreted as the evidence that the tradeoff between fairness and efficiency has been addressed. Comparing the results of FTDM and dynamic STDM schemes show the higher efficiency of the latter two methods. We also observe that both fairness and efficiency characteristics of the results of RB-STDM scenario outperform the results of BF-STDM scenario.

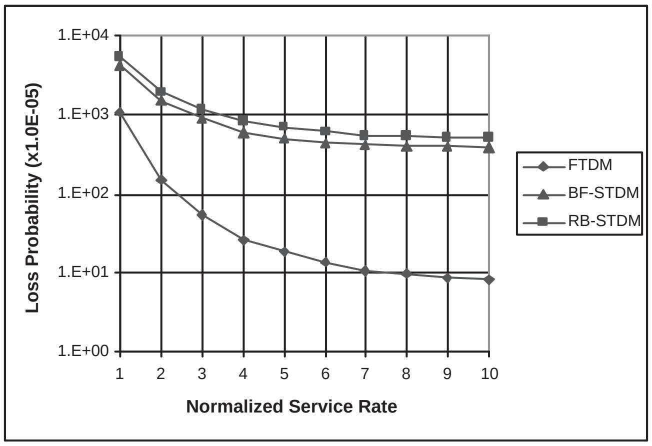

Fig. 4. Plots of single source delay-caused loss versus normalized service rate for the quad source queuing system using FTDM, BF-STDM, and RB-STDM server scheduling.

It is worth mentioning that reducing the epoch length decreases the overall and single source loss rates of both DNS methods at a higher cost of server scheduling. Further, we note that the performance of different methods are very different as the result of having to work with a heavily utilized system.

In what follows, we mention some of the practical findings in the implementation of our experiments. We note that the original convergence of the learning scheme is time consuming because of the rich dynamics of the traffic pattern. In addition, all of the convergence results of BF-STDM are strongly affected by the choice of initial conditions of the weighting functions and the minimum acceptable error bound ε. Our observation is somewhat different in the case of RB-STDM. While the latter method is not very sensitive to the choice of initial conditions of the weighting functions, the effects of the acceptable error bound ε remain significant. We have observed that 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. The remaining parameters of resilient back-propagation learning are set according to what is proposed in [21]. We have also observed that thousands of samples in the original learning period are required. Additionally, we have observed that the speed of convergence of resilient back-propagation is usually better than that of BFGS back-propagation learning for the same choices of traffic patterns and initial conditions.

V. Conclusion

In this paper, we provided a dynamic server scheduling scheme as an application of adaptive neuro-forecasting. We utilized BFGS and resilient back-propagation learning schemes in a fixed structure neural network to forecast traffic patterns. Based on our forecasting results, we provided dynamic server scheduling schemes to improve the delay performance of Statistical Time Division Multiplexing while considering the fairness issue. Our dynamic STDM schemes relied on the water-filling approach. Our experimentation utilized a multiple source queuing system accommodating artificially generated self-similar traffic patterns resembling VBR video traffic. We compared the performance of different server scheduling schemes and concluded that our dynamic STDM schemes were able to address the trade off between fairness and delay.

References

[1] 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.
[2] 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.
[3] 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.
[4] A.K. Choudhury, E.L. Hahne, “Dynamic Queue Length Thresholds for Shared-Memory Packet Switches,” IEEE/ACM Trans. on Networking, April 1998.
[5] A. Elwalid, D. Mitra, “Design of Generalized Processor Sharing Schedulers which Statistically Multiplex Heterogeneous QoS Classes,” In Proc. of IEEE INFOCOM, 1999.
[6] A. Erramilli, R.P. Singh, P. Pruthi, “Chaotic Maps as Models of Packet Traffic,” In Proc. of ITC-14, 1994.
[7] V. Firoiu, J. Kurose, D. Towsley, “Efficient Admission Control for EDF Schedulers,” In Proc. of IEEE INFOCOM, 1997.
[8] G.J. Foschini, B. Gopinath, “Sharing Memory Optimally,” IEEE Trans. on Communications, March 1983.
[9] G. Gallasi, C. Rigolio, “ATM Bandwidth Assignment and Bandwidth Enforcement Policies,” In Proc. of IEEE GLOBECOM, 1987.
[10] L. Georgiadis, I. Cidon, R. Guerin, A. Khamisy, “Optimal Buffer Sharing,” IEEE JSAC, September 1995.
[11] M. Gerla, L. Kleinrock, “Flow Control: A Comparative Survey,” IEEE Trans. on Communications, April 1980.
[12] 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.
[13] M.I. Irland, “Buffer Management in a Packet Switch,” IEEE Trans. on Communications, March 1978.
[14] J.M. Jaffe, “Bottleneck Flow Control,” IEEE Trans. on Communications, July 1981.
[15] G. Latouche, “Exponential Servers Sharing a Finite Storage: Comparison of Space Allocation Policies,” IEEE Trans. on Communications, June 1980.
[16] F. Kamoun, L. Kleinrock, “Analysis of Shared Storage in a Computer Network Node Environment under General Traffic Conditions,” IEEE Trans. on Communications, July 1980.
[17] P. Kermani, L. Kleinrock, “Virtual Cut-Through: A New Computer Communication Switching Technique,” Computer Networks, Vol.3, 1979.
[18] A. Kesselman , Y. Mansour, “Harmonic Buffer Management Policy for Shared Memory Switches,” Theoretical Computer Science, September 2004.
[19] 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.
[20] S.H. Low, “Equilibrium Bandwidth and Buffer Allocations for Elastic Traffics,” IEEE/ACM Trans. on Networking, June 2000.
[21] M. Riedmiller, H. Braun, “A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm,” In Proc. of the IEEE Intl. Conf. on Neural Networks 1993.
[22] W. Stallings, “Data and Computer Communications, Sixth Edition” Prentice-Hall, 2000.
[23] A.K. Thareja, A.K. Agarwala, “On the Design of Optimal Policy for Sharing Finite Buffers,” IEEE Trans. on Communications, June 1984.
[24] D. Tipper, M.K. Sundareshan, “Adaptive Policies for Optimal Buffer Management in Dynamic Load Environments,” In Proc. of IEEE INFOCOM, 1988.
[25] R. L. Watrous, “Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization,” Technical Report MS-CIS-87-51 LINC LAB 72, University of Pennsylvania, Philidelphia, June 1986.
[26] 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.
[27] J.P. Yang, “Performance Analysis of Threshold-Based Selective Drop Mechanism for High Performance Packet Switches,” Performance Evaluation, May 2004.
[28] H. Yousefi’zadeh, “A Neural-Based Technique for Estimating Self-Similar Traffic Average Queuing Delay,” IEEE Communications Letters, October 2002.
[29] H. Yousefi’zadeh, E.A. Jonckheere, J.A. Silvester, “Utilizing Neural Networks to Reduce Packet Loss in Self-Similar Teletraffic Patterns,” In Proc. of IEEE ICC, 2003.
[30] 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.
[31] H. Yousefi’zadeh, E.A. Jonckheere, “Dynamic Neural-Based Buffer Management for Queuing Systems with Self-Similar Characteristics,” IEEE Trans. Neural Networks, September 2005.