ATM Connection Admission Control Using pRAM Based Artificial Neural Networks

Francesco Balestrieri[*] Pantelis L. Panteli Vasillios Dionissopoulos

Trevor G. Clarkson

e-mail:pantelis.panteli@kcl.ac.uk

DEPARTMENT OF ELECTRONIC ENGINEERING

KING'S COLLEGE LONDON

Keywords

ATM, CAC, pRAM, neural networks

Abstract

As they learn from observed data, neural networks have found many applications in communication networks. Here, a neural network for Call Admission Control (CAC) based on the Probabilistic-RAM (pRAM) neuron model is presented. Because pRAM neural networks can be implemented in hardware easily and at a low cost, they make excellent controllers in the ATM environment. The performance of such a controller is analysed through simulations, and the results are compared with the equivalent capacity of connections CAC algorithm. The results show that, the proposed hardware-based controller guarantees the required quality of service and at the same time provides an improvement in network utilization.

1 Introduction

Among the tasks that ATM must be able to fulfil is the ability to meet the demands for service quality posed by the different service classes. These demands are negotiated during the call-setup procedure and, based on the ability of the network to satisfy these requirements without deteriorating the quality of service (QoS) provided to already established connection, a call is accepted or rejected. On the other hand, the efficient utilization of network resources is a primary concern to the network provider, thus a balance has to be maintained between the QoS and network utili-

[*] University of Pisa (Italy), at King's College London sponsored by the EU SOCRATES exchange programme

zation. This is done through traffic control mechanisms, such as Connection Admission Control (CAC).

CAC procedures decide whether a new connection should be accepted or rejected. Traditionally, this is done by defining an analytical model of the network and then applying mathematical techniques in order to evaluate the effect of the traffic configuration on the QoS. In most cases the resulting calculations are too complex to be performed in real time, and an approximation is needed at the price of reduced network throughput. Also, the accuracy of the decision depends on the accuracy of the analytical model.

One of the alternatives proposed is the use of neural networks. Due to their ability to learn from observed data, neural networks can be trained to model the non-linear relationship between the traffic characteristics and the QoS. The knowledge acquired by the neural network can then be used in real time to generate the CAC decision. Because neural networks are adaptive and rely on observed data rather than on an analytical model of the system at hand, the resulting scheme is robust, efficient and capable of modifying itself to reflect changes in traffic behaviour.

To evaluate the suitability of a neural-network based CAC scheme, however, the issues of hardware realization and of its cost must be considered. In this work, a neural admission controller based on the Probabilistic-RAM (pRAM) neuron model is presented. Neural networks built using this neuron model have all the advantages discussed above, but most important can be easily implemented with conventional RAM memory, with the result of high cost-effectiveness.

2 pRAM Based Artificial Neural Networks

An artificial neural network, or simply a neural network, is an information processing system whose structure was inspired by research on the human brain. It differs from traditional computing systems in that, rather than having a single, complex processing unit, it is built of many simple processing units interconnected in a network-like structure. The basic processing unit of a neural network is called a *neuron*, in analogy with the neurons found in the human brain. A neuron is a multiple-input, single-output non linear circuit.

2.1 The pRAM Neuron Model

A N-input pRAM (N-pRAM) has N binary inputs $u_1, \dots, u_n$ and one binary output a. Like in conventional memories, the input vector $\mathbf{u}=[u_1, \dots, u_n]$ selects one out of 2^N locations $\alpha_\mathbf{u}$, whose

content is a continuous value between 0 and 1. The selected α_u represents the probability that the neuron *fires*, which means that the output *a* is 1. The α_u 's are referred to as *weights*.
Given an input vector *i*, the probability that the pRAM fires is then

$$\Pr(a=1 \mid i)=\alpha_i \tag{1}$$

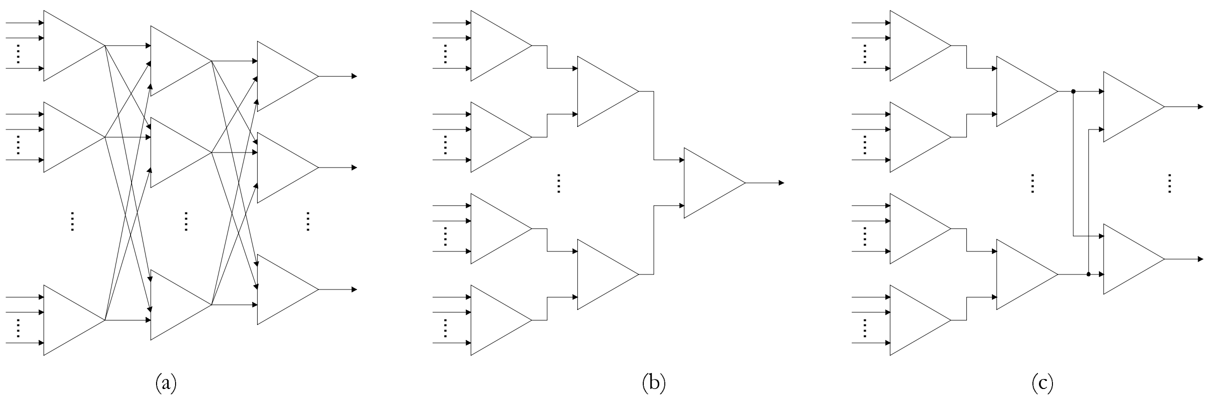

Figure 1: pRAM network topologies: feed-forward fully interconnected (a), pyramidal (b), trapezoidal (c).

or more generally

$$\Pr(a=1 \mid i)=\sum_u \alpha_u \prod_{j=1}^{N}\left(i_j u_j+\bar{i}_j \bar{u}_j\right) \tag{2}$$

where for any real number z, $\bar{z}=1-z$.
The output *a* is then 1 with a probability whose distribution is governed by the 2^N variables α_u. Apart from providing a realistic model of a biological neuron, this allows for a large number of behaviours and high nonlinearity [2].

2.2 PRAM network topologies

As with any other neuron model, pRAM neurons are organized in networks. The number of neurons in the network and the network topology determine the capabilities of the network itself, and also influence the speed of learning. Different applications usually require different network architectures, and the right network architecture is found with a trial-and-error procedure where the performance of different networks is assessed for the particular problem in order to find the most suitable one.

The main types of network architectures used with pRAM neurons are the fully-connected feedforward network and the pyramidal network (fig. 1). The pyramidal network is usually preferred as it proved to be suitable for most applications with the advantage of greater simplicity.

2.3 Learning in pRAM networks

A simple and efficient reinforcement learning technique has been developed for the pRAM, in which the environment generates a binary reward signal $r \in \{0,1\}$ and a binary penalty signal $p \in \{0,1\}$. At each step of the learning procedure only the weight that was addressed by the input vector is updated. At time t+1 the new weight $\alpha_u(t+1)$ is given by $\alpha_u(t+1) = \alpha_u(t) + \Delta\alpha_u(t)$, where $\Delta\alpha_u(t)$ is obtained by the *learning rule*

$$\Delta\alpha_u(t) = \rho((a-\alpha_u)r + \lambda(\bar{a}-\alpha_u)p)(t) \times \delta_{u,i} \tag{3}$$

where $\rho \in [0,1]$ is defined as the *learning rate* and $\lambda \in [0,1]$ as the *reward-to-penalty ratio*.
The effect of equation (3) is that when a reward is received (r=1) the probability α_u of obtaining the same output a with the same input vector u increases, whereas the same probability decreases if a penalty is received (p=1). The Kronecker delta $\delta_{u,i}$ ensures that only the location accessed at time t is updated.
Other training algorithms than the one described above can be used. A version of the popular back-propagation technique, for example, has been defined for the pRAM[2].

2.4 The Integrating pRAM

An extension of the pRAM that operates on real-valued inputs and outputs, called the integrating pRAM (i-pRAM), has been presented in [5]. The structure of the i-pRAM is the same as that of the pRAM: what is different is the way in which inputs are presented to the neuron.

Let us consider a single real number $x \in [0,1]$. The number x can be approximated by a quantity x^* obtained by averaging a pulse stream of length R:

$$x^* = \frac{1}{R}\sum_{r=1}^{R} i(r) \qquad (4)$$

where *i(r)* is a binary value which takes the value 1 with probability equal to *x*. If we present *N* of such pulse streams on each of the *N* pRAM inputs, the pRAM will generate at the output another pulse stream of length *R* that can be averaged in the same way as before to obtain a real number

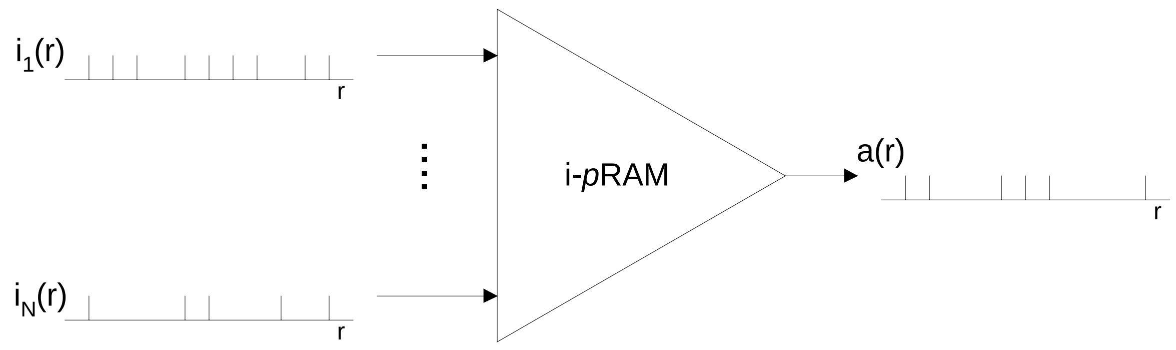

Figure 2: Input and output sequences of an i-*p*RAM.

$y \in [0,1]$, which is the real output of the i-pRAM.

The learning rule (3) needs to be modified to work in the i-pRAM case. Over the period *R*, at most *R* different locations will be addressed by the different binary input vectors resulting from the conversion of the real values in pulse streams. We define $X_{\underline{u}}$ as the number of times that a location $\alpha_{\underline{u}}$ has been addressed. The quantity $X_{\underline{u}}$ gives a measure of how much the weight $\alpha_{\underline{u}}$ contributed in generating the output *y*, and thus indicates how much $\alpha_{\underline{u}}$ should be affected by the training step. Having introduced $X_{\underline{u}}$, the learning rule (3) becomes:

$$\Delta\alpha_{\underline{u}}(t) = \rho\left(\left(a - \alpha_{\underline{u}}\right)r + \lambda\left(\bar{a} - \alpha_{\underline{u}}\right)p\right)(t) \times X_{\underline{u}}(t) \qquad (5)$$

2.5 Hardware realization of pRAM networks

2.5.1 Hardware model of the pRAM neuron

As it was said in section 2.1, the step from the abstract definition of the pRAM neuron model to its hardware implementation is indeed short.

The weight memory can be implemented with an ordinary bank of RAM, with a number of bits per location sufficient to guarantee enough precision for the weights. Common 16-bit commercial memories are suitable for this purpose [2].

The only other components needed are a random number generator and a comparator (Fig.3).

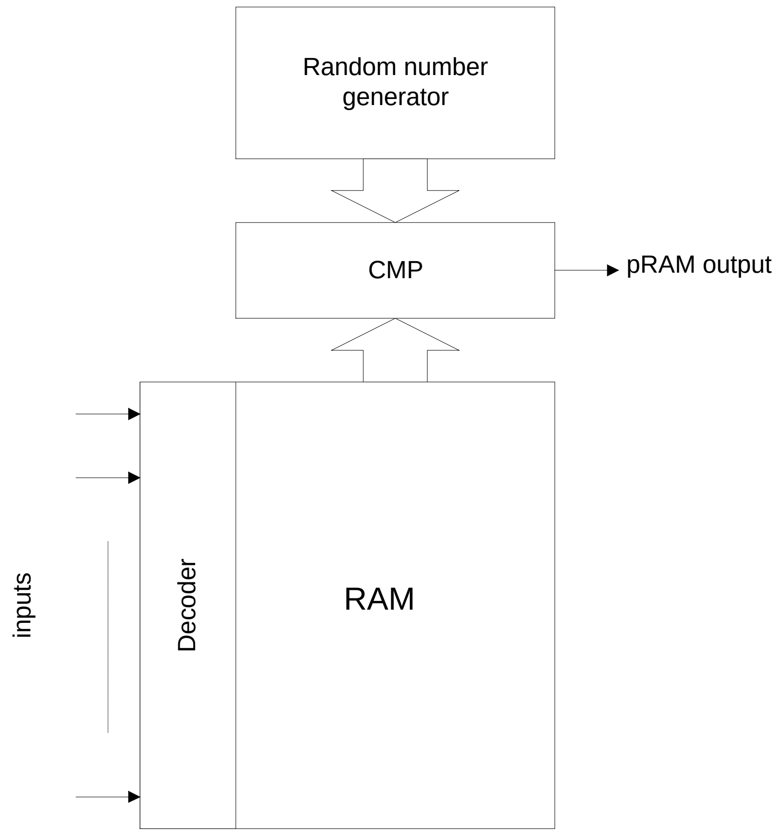

Figure 3: Hardware realization of a pRAM neuron.

When an input is presented to the pRAM, the content of the corresponding memory location is extracted and it is compared with the outcome of the random number generator. If the random number is less than or equal to the weight, the pRAM output is 1, otherwise it is 0.

2.5.2 The pRAM-256 VLSI Neural Network Processor}

The current generation of hardware pRAM was presented in [4]: it is a VLSI processor, called pRAM-256, that implements 256 neurons on one chip. Its main features are[3]:

- ♦ 256 pRAMs, 6 inputs each
- ♦ Configurable connections between pRAMs
- ♦ On-chip Reinforcement Learning Unit, implementing the learning rule (3)

- Learning can be global (i.e. the reinforcement signal is broadcast to all neurons) or local (i.e. each neuron receives a different reinforcement signal)

- A non-learning cycle for all 256 pRAMs takes 0.154ms at 33 MHz

- A learning cycle for all 256 pRAMs takes 0.246ms at 33 MHz

- External static RAM used for efficient weight storage

- The number of 256 neurons is enough for most applications. If a larger network needs to be built, up to five chips can be connected in parallel to provide a larger number of neurons.

Thanks to the on-chip learning unit and to the configurable connections, the pRAM-256 provides a complete hardware solution that can be embedded in stand-alone controllers without the need of software support, still offering the flexibility of a software solution.

3 pRAM Networks for CAC

Connection admission control has been one of the first problems in ATM control to be addressed through the use of neural networks. The ability of neural networks to adapt to changing traffic situations combined with prediction capabilities have resulted in very efficient schemes [1, 7-12, 15]. The main advantages obtained through the use of neural networks in CAC are:

- A NN based controller does not depend on a particular traffic model, as opposed to analytical CAC techniques.

- A NN based controller is adaptive, and can be retrained to reflect changes in the ATM network, like the introduction of a new service or modifications in the network topology.

- A NN based controller is fast, and once its structure is defined its speed does not depend on the complexity of the relation to be learned.

- Because it extracts information from the observation of traffic, a NN based controller does not require the user to specify a detailed traffic descriptor.

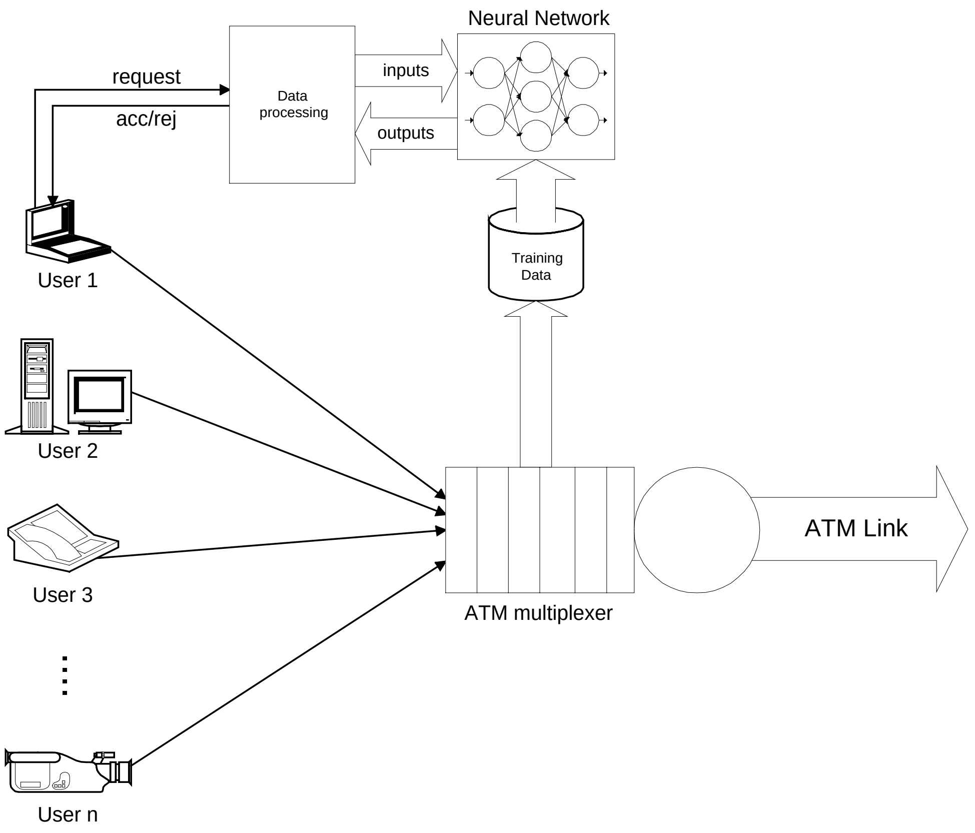

Figure 4: Block diagram of a neural CAC controller.

The general scheme for CAC using neural networks is shown in fig.4: before beginning to transmit, a user asks for permission to the admission controller, that interrogates the neural network in order to evaluate the effect of the new connection on the resulting QoS. At the same time, training data is collected by observing the buffer status and relating this information to the current network situation.

Most of the neural networks applications to CAC refer to the classical multilayer perceptron model, without addressing a particular hardware solution. The issue of hardware implementation, however, must be considered in order to judge how and at which cost a particular neural network implementation can be integrated in an ATM switch.

The pRAM neuron model described in section 2 can be easily implemented with common digital components, and as its main building block is an array of conventional RAM, its implementation is also very cost effective. For these reasons, and for its good properties in terms of learning speed, it makes an excellent candidate for the ATM environment where cost and speed are critical issues.

3.1 Definition of the pRAM Admission Controller

The first step in the definition of the neural admission controller is deciding the problem representation, or in other words which information must be presented to the neural network and how, and which information are received from the neural network. Particular care must be taken in making this choice, because it influences the learning ability of the neural network.

3.1.1 Choice of Inputs and Outputs

There is good agreement [8,11] that the neural network should have as many inputs as there are traffic classes, each input representing the number of open connections of each class. That is, if we have N traffic classes we define the *call vector* $\underline{c} = [c_1, \cdots, c_N]$, c_i is the number of traffic sources of class *i* that are allowed to transmit.

For what concerns the output, a possible solution is to make the neural network directly generate the CAC decision. In this case the neural network would act as a classifier, one class being the acceptable call vectors (i.e. those for which the required QoS is guaranteed) and the other the unacceptable ones. However, such a choice implies that the neural network needs to be retrained whenever the QoS requirement changes.

A better choice is to make the neural network estimate the expected value for the QoS parameter given a certain call vector: this way, to decide if a call vector can be accepted or not, the estimated QoS is compared with a threshold corresponding to the QoS objective, and if the latter is changed it is sufficient to modify this threshold.

With such a choice of inputs and outputs, the neural network acts as a function approximator, the target function being the correspondence between the QoS parameter and the call vector. As a confirmation of the validity of such a choice, it should be noted that several analytical CAC techniques (for example the one in [13]) attempt to calculate an approximation of the same function.

3.1.2 Input and Output Transformations

It is necessary to operate some kind of transformation on the inputs before feeding them to the neural network, and on the outputs after retrieving them. This is not only because the i-pRAM works in the range [0,1], but also because we need to convert the target function to a form that is easier to learn.

Each input to the i-pRAM is an integer number, that must be converted to a real number in the range [0,1]. As the inputs are equally spaced in their range, a simple normalization is enough for this task. Thus, if c_i is the component of the call vector we are considering, the corresponding input to the i-pRAM x_i is:

$$x_i = \frac{C}{PCR} + \frac{c_i}{\frac{C}{SCR} - \frac{C}{PCR}} \qquad (6)$$

where *C* is the link capacity (in cells/sec) and *SCR* and *PCR* are the Sustainable and Peak Cell Rate, and we are considering the Cell Loss Rate (CLR) as the QoS objective. In other words, we take as the minimum number of connection the one given by the peak rate allocation (as any number of connections below this would never generate any losses), and as the maximum the one given by the average (sustainable) rate allocation (as any number of connections above this would make the network unstable, because the aggregate offered load would exceed the link capacity).

Let us consider as a QoS parameter the Cell Loss Rate (CLR). As the loss rate is already in the range [0,1], it could appear that no output transformation is needed. However, the pRAM does not have the precision required to approximate the very small values, for example 10^{-9}, of loss rates that occur in an ATM network. As we are only interested in the order of magnitude of the CLR, we can train the i-pRAM network to learn the logarithm of the loss rate instead. Again, we must normalize this value in the range [0,1] by choosing a minimum and maximum value: the maximum is obviously log(1)=0, while the minimum is the lowest CLR requirement, say $\log(10^{-12}) = -12$. As the i-pRAM generates positive numbers, we must take the opposite of the log(CLR). The output transformation is then:

$$\log(CLR) = -\frac{y}{12} \qquad (7)$$

where *y* is the output of the i-pRAM.

Another choice that must be made at this stage is which pRAM network topology to use. pRAM networks have been successfully used as function approximators in their continuous version, the i-pRAM. In particular, a two-layer pyramidal network of 8-ipRAMs (fig.5) has been used in ATM traffic shaping [14] and head-of-line control [16]. Also in our experiments this topology proved to be a good compromise between simplicity and accuracy.

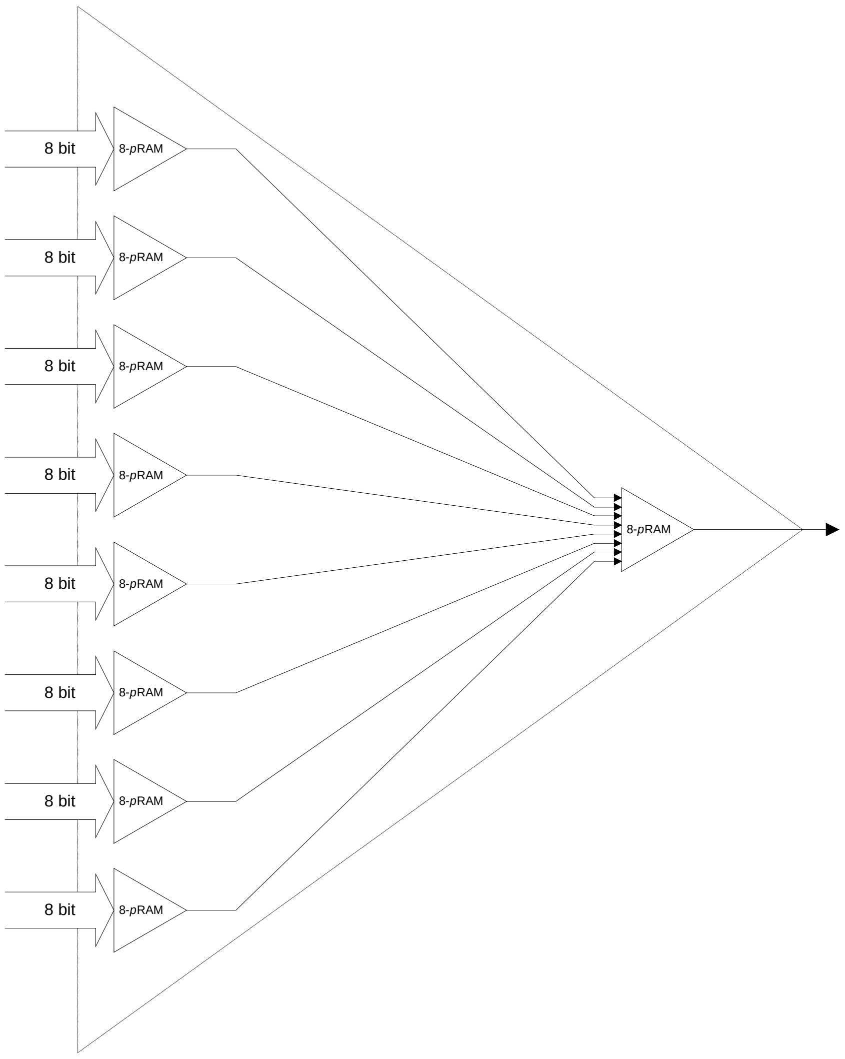

Figure 5: Neural network topology.

3.1.3 Training

In order for the neural network to correctly apprehend the relation between the QoS parameter and the call vector, training data must be collected during the normal ATM network operation. The purpose of this data collection is to relate the value of the QoS parameter with the call vector that generated it. The way in which this data collection is performed is different for the various QoS parameters.

Let us take the CLR as the QoS parameter. Ideally, to correctly relate the CLR to the number of connections we would need to fix the number of connections until the network reaches the steady state and only then compute the average value of the loss rate. In the practical case, however, the number of connections varies unpredictably and it is not possible to fix a given number of connections without interfering with the users' requests.

It has been pointed out [13] that in a broadband network the arrival rate of data is much higher than the arrival rate of calls: moreover, the time between two successive connection requests is usually enough for the CLR to reach a value close to the steady-state value.

For this reason, to collect the training data we calculate the CLR between two successive call requests and we relate it to the call vector observed at the beginning of the interval. As we could obtain, for the same call vector, different values for the CLR in different observation intervals, we take as a final measurement the average of those values.

A method to collect data for Cell Transfer Delay (CTD) approximation has been proposed in [11]. In this method, when a cell enters the buffer it is stamped with the current time and the call vector. Then, when the packet exits the buffer and is transmitted on the link, the CTD is calculated as the difference between the exit time and the insertion time, and it is related with the stamped call vector. Again, an average of the measurements obtained for the same call vector is

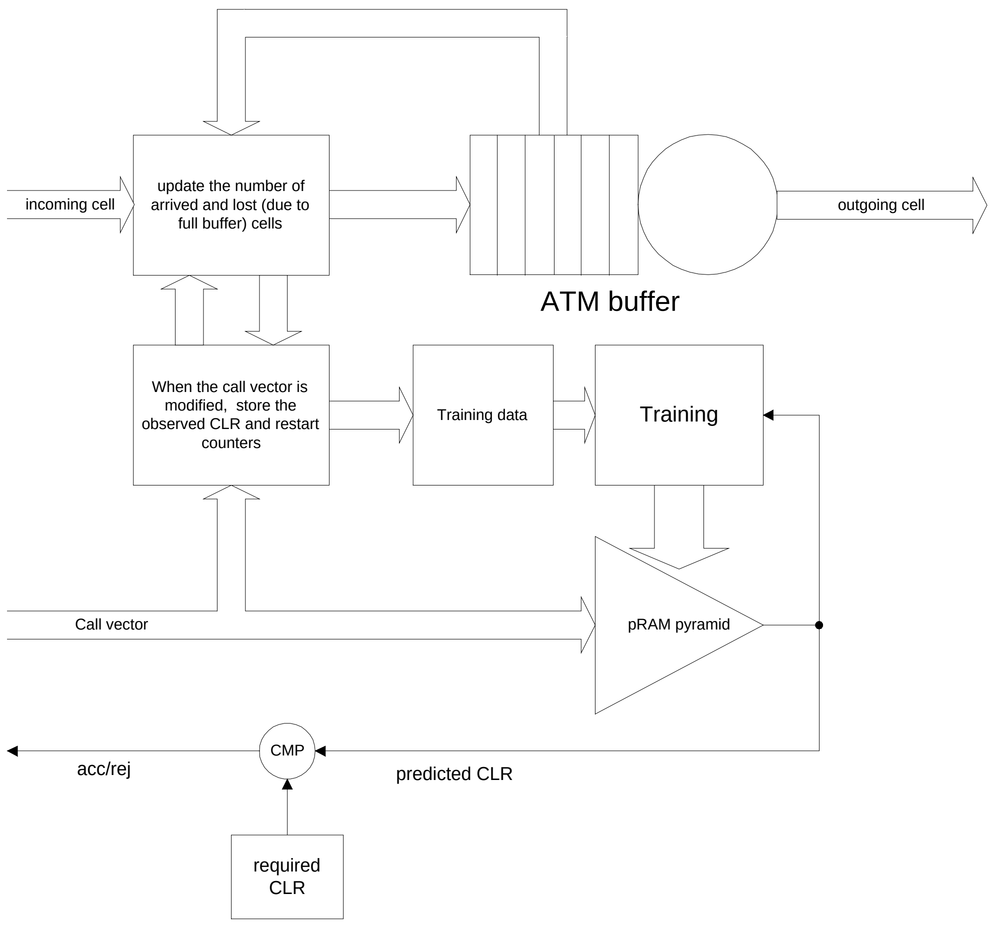
incoming cell
update the number of arrived and lost (due to full buffer) cells
outgoing cell
ATM buffer
When the call vector is modified, store the observed CLR and restart counters
Training data
Training
Call vector
pRAM pyramid
CMP
acc/rej
predicted CLR
required CLR

Figure 6: Data collection for CLR approximation.

taken as a final value.

This way, the neural network learns to approximate the delay introduced by the buffer: as the other factors that contribute to the overall delay (for example, the propagation delay) are fixed, this estimate is enough to predict the CTD introduced at the local switch.

3.2 Integration with an ATM Switch

The diagram of Fig.4 is general, and does not take into account the fact that access to an ATM network takes place through a switching element. In the practical case, the information about the QoS and traffic parameters of the new connection must be extracted from the traffic descriptor contained in the connection request that is embedded in the flow of cells. Only when the request has been accepted by all the switches along the path the connection is effectively established.

Fig.7 illustrates how the pRAM controller could be situated inside an ATM switch with output buffering: every output link has its own buffer, and each buffer has a dedicated pRAM controller. If, as it is often the case, a separate buffer is used for the different QoS requirements, again a pRAM controller must be used for each buffer.

The use of the term, pRAM controller, means the hardware-based neural network used in this research. In the simulation and results, the pRAMs are used in the integrating mode, as i-pRAMs, but the term pRAM is used for readability.

4 Simulation and Results

The performance of the proposed controller is analysed through simulations conducted in OPNET (OPtimized Network Engineering Tools).

The first step of our simulation experiments is to collect data for successive pRAM training. To do this, we run the simulation without any admission control, in order to explore all the possible traffic configurations, even those that would cause congestion in the ATM network. The system in question is an ATM node with a buffer size of 10 cells, where sources of a single class are multiplexed. The sources are modelled using the Interrupted Bernoulli Process (IBP) [13]. The peak and average rates, normalized to the link capacity as in [16], are P=0.05 and D=0.01 and the mean burst length is B=80.

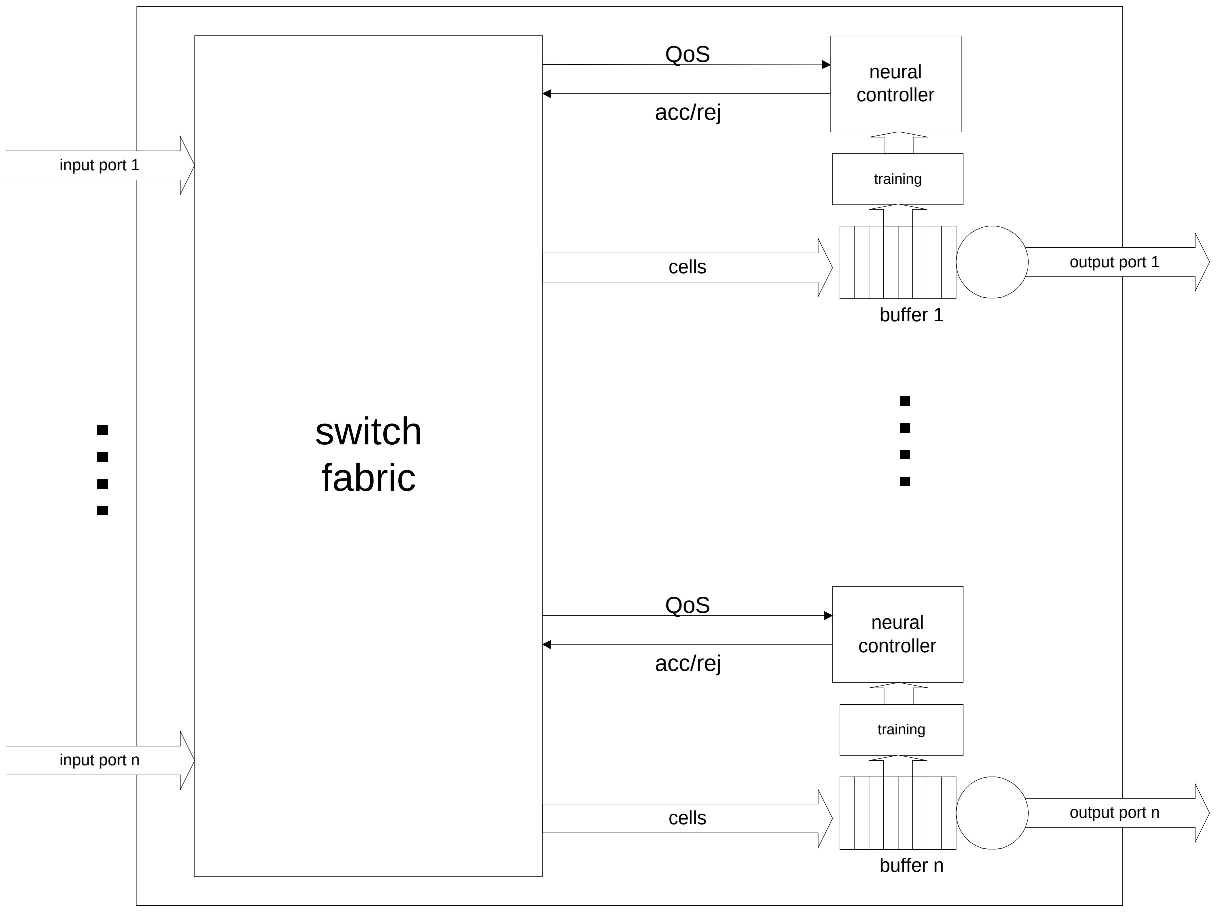

Figure 7: Integration with an ATM switch.

Call requests are generated at a constant rate of 5 calls/sec in the high traffic state, and 0.01 calls/sec in the low traffic state. The duration of the high and low traffic states is 16 seconds, and the initial number of active connections is 20, so that the number of requests oscillates between 1/P=20 (peak allocation) and 1/D=100 (average allocation).

The simulation enables us to verify the validity of our method to measure the CLR. The simulation was performed for a simulated time of 5000 seconds with a link capacity of 50 Mb/s. The measured CLR is then compered with the values obtained using the analytical computation of the CLR according to the method proposed in [13]. The results of this comparison are shown in fig.8.

The graph shows that the measured CLR closely follows the analytical calculation for high values of the loss rate, becoming less accurate for lower values. The reason for this is that in order to measure very low values of the CLR we need to run the simulation for a long time: for example, if we need 100 losses to measure the CLR with reasonable accuracy, for a CLR of 10^{-8} we must generate at least 10^{10} cells. With a link capacity of 50Mb/s this means running the simulation for at least 84800 simulated seconds *only* for the traffic configurations that generate that particular CLR. In a real ATM network this would not be a problem, because the measure would be obtained during normal network operation; in our simulation, however, we had to stop the accuracy to a CLR of 10^{-6}, also taking into account the fact that the measurements of the lowest CLRs are less reliable.

Having collected the data, we now train the pRAM network to learn the relationship between the CLR and the number of connections. Figure 9 shows the training error obtained with the learning rates ρ=0.1 and λ=0.01: as it can be seen, the error goes below 4% after only 80 training epochs.

Figure 10 compares the pRAM approximation with the CLR measurement. Again, the area around the CLR value of 10^{-6} is less reliable.

At this point we can use the trained pRAM to predict the variation of the CLR as the network evolves. As we want to show how the pRAM predicts the loss rate in various traffic situations, we still leave the network uncontrolled. This time call requests are generated with exponentially distributed inter-arrival times: the traffic configuration (i.e. the variation of the number of calls) is shown in figure 11.

Figures 12 and 13 respectively show the prediction of the CLR at every connection initiation or termination and the prediction of the average CLR. It can be seen that the pRAM prediction is close to the actual measurement.

Finally, we use the pRAM prediction of the CLR to generate the CAC decision. The maximum number of connections admitted by the pRAM admission controller for different buffer sizes and a CLR requirement of 10^{-4} is given in figure 14, and it is compared with the maximum number of connections admitted by the equivalent capacity method [6].

Fig.15 shows the average CLR measured in the buffer when the maximum number of connections is admitted by the pRAM admission controller, with a statistical confidence of 90%: it is evident that the CLR requirement of 10^{-4} is respected. The CLR obtained with the equivalent capacity method was 0 in all situations, and is not shown in the graph. This is in accordance with the fact that the equivalent capacity method is highly conservative, especially with small and moderate buffer sizes [16].

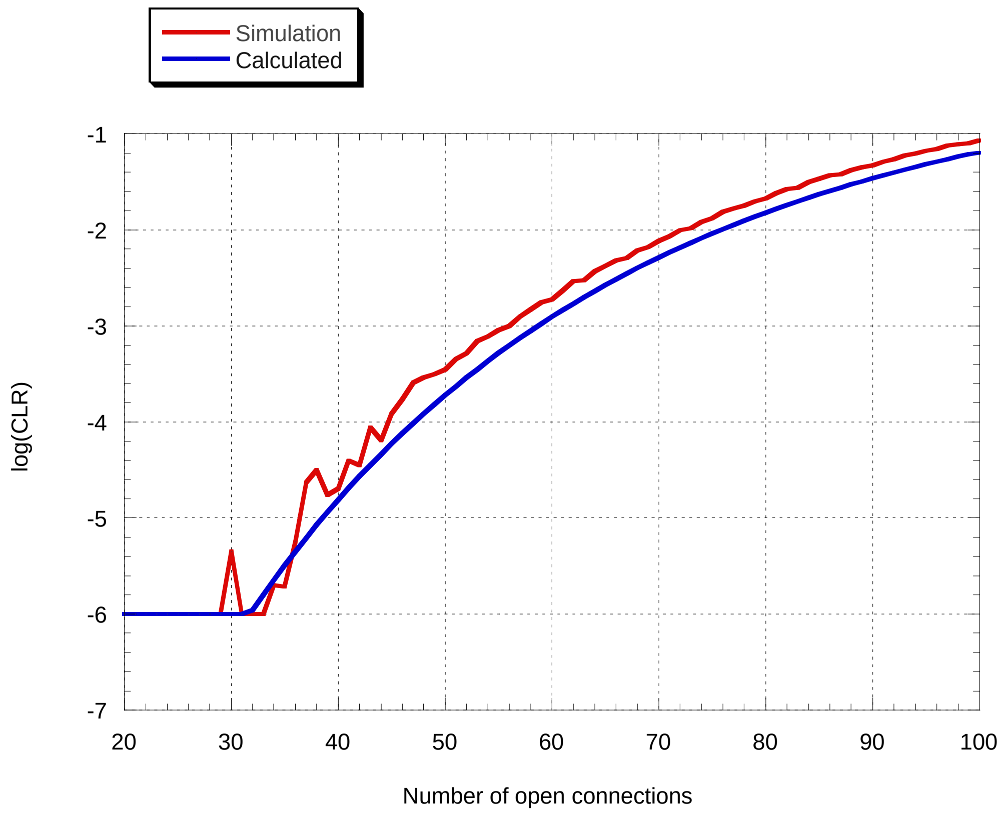

Figure 8:

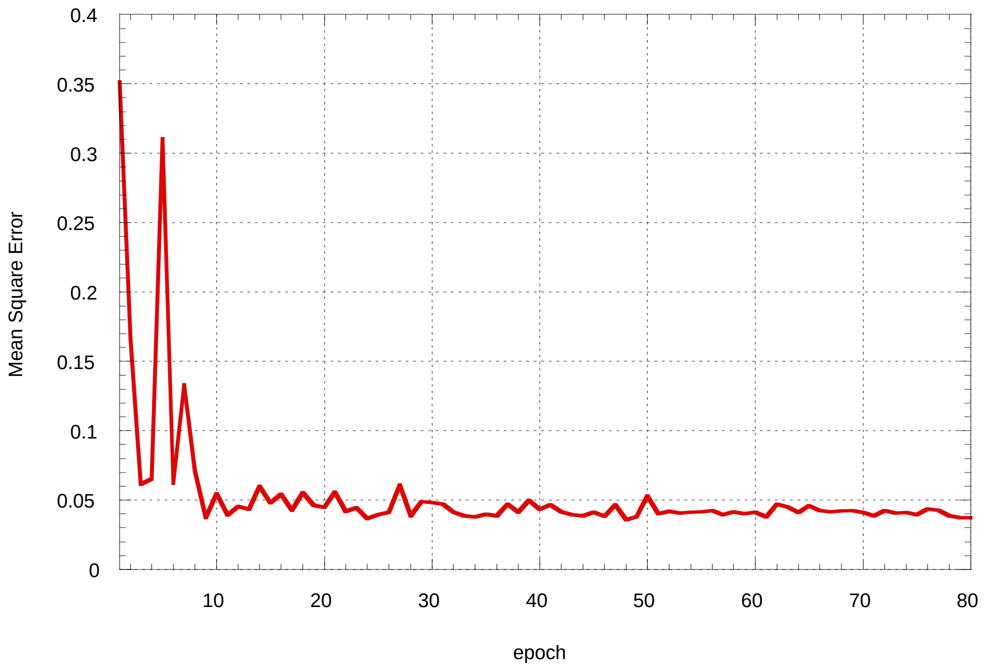

Figure 9:

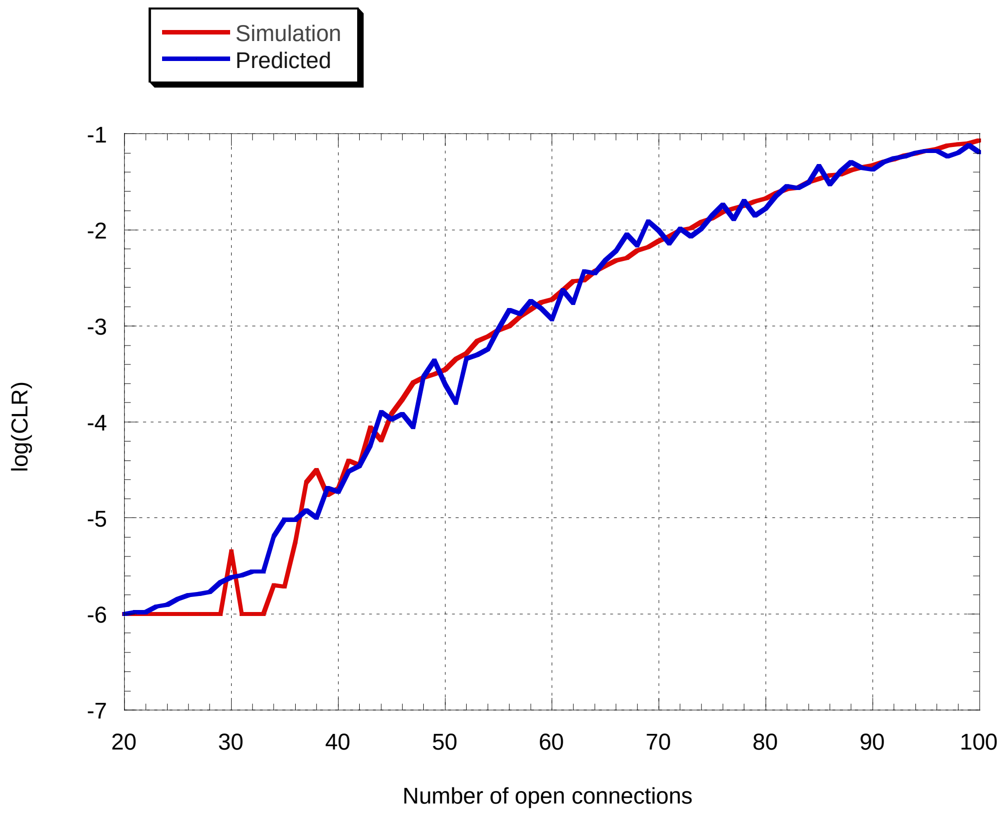

Figure 10:

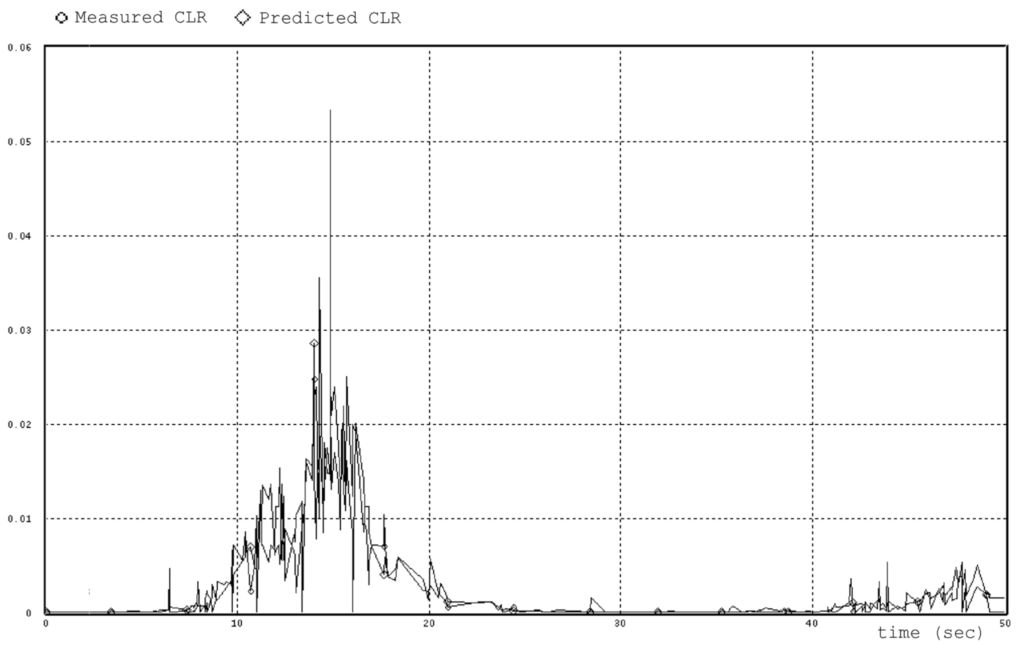

Figure 11:

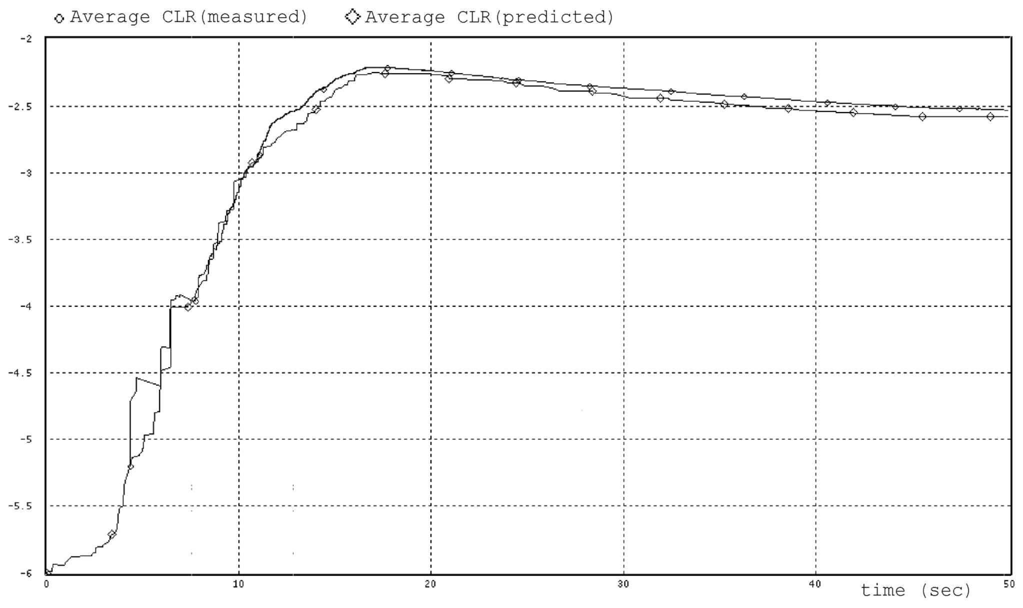

Figure 12:

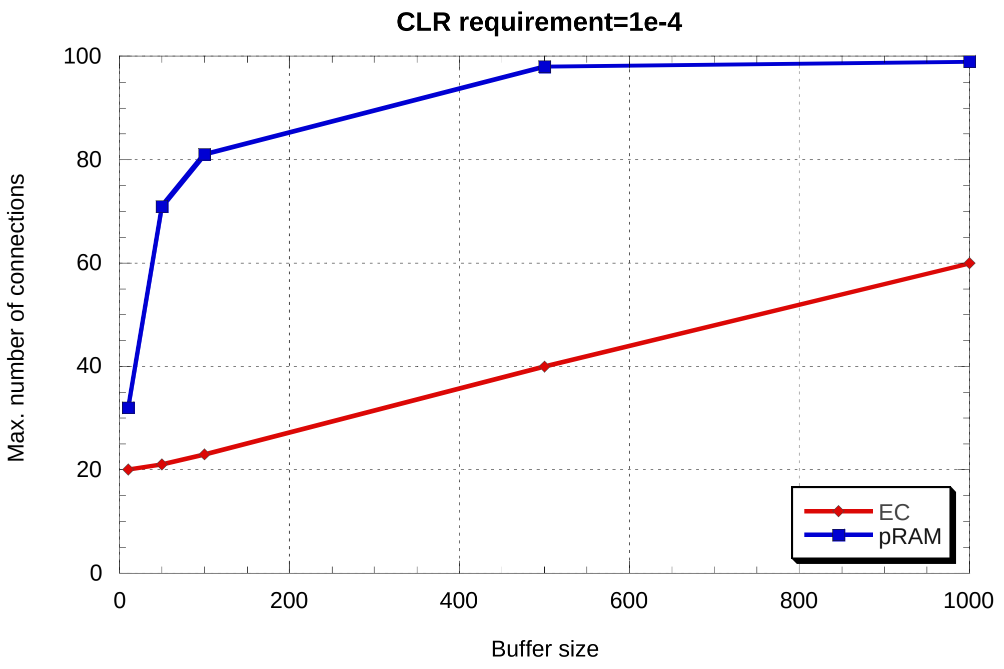

Figure 13:

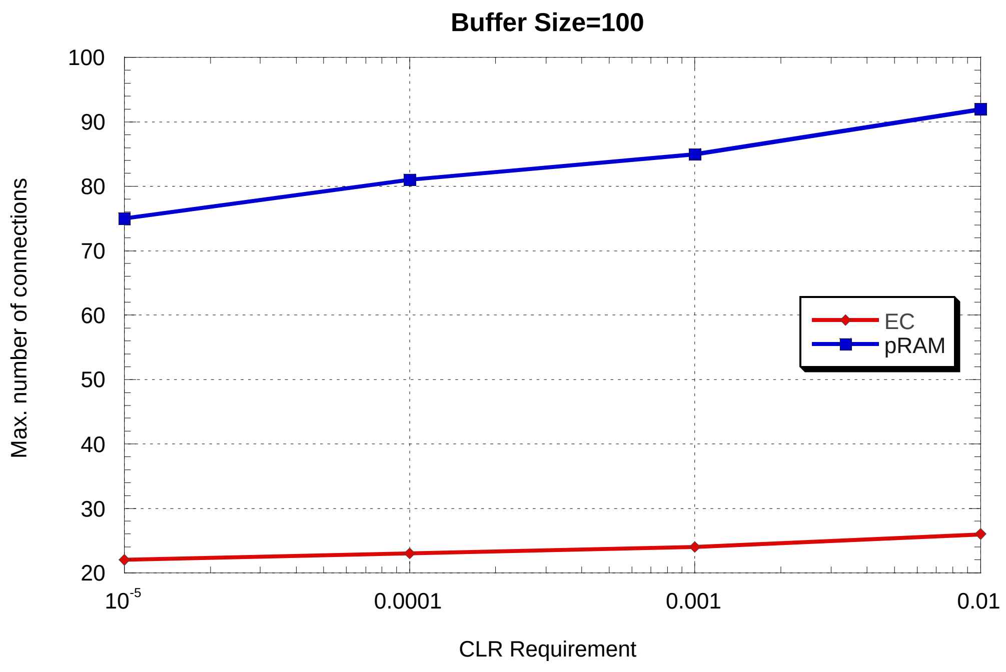

Figure 14:

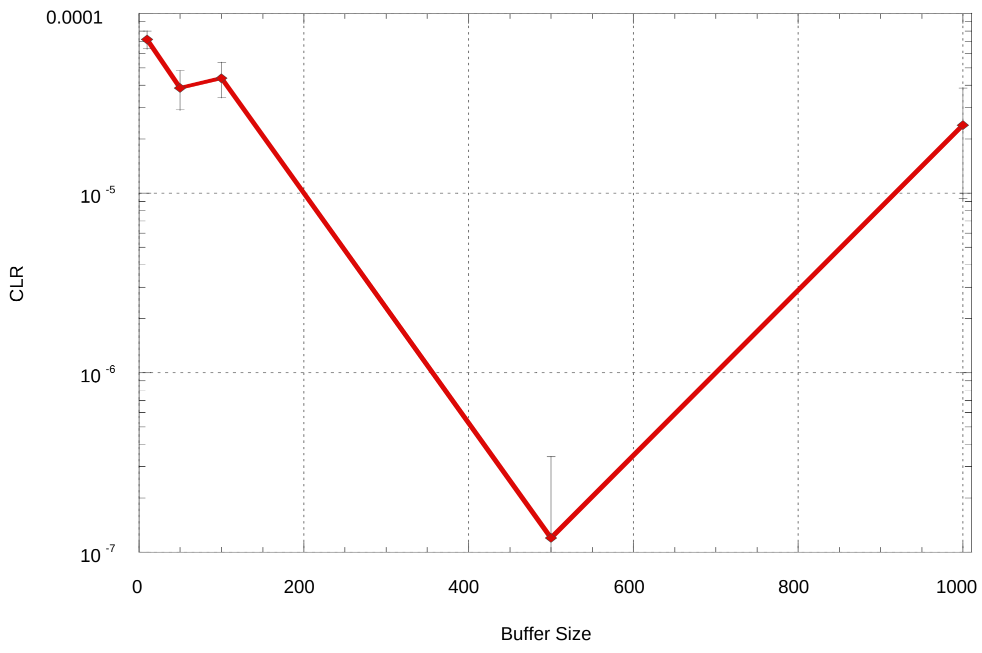

Figure 15:

Conclusions and Further Work

The simulation results show how the proposed admission controller successfully approximates the variation of the QoS parameter while the ATM network evolves. The pRAM-based neural network learns the relation between the CLR in the ATM buffer and the number of admitted connections quickly and with good accuracy, with the additional advantage of easy and inexpensive hardware realisation.

The network performance obtained using the proposed admission controller shows a good improvement over that obtained with the equivalent capacity method; in particular, the neural controller is less conservative, resulting in better network utilization while still meeting the QoS requirement.

There are several ways in which this work could be improved. First of all, the system should be tested with other traffic models than the IBP, and especially with a traffic model that better reproduces video sources. Also different ATM network topologies should be used, in order to test the effect of the aggregation of traffic at intermediate switches.

Different pRAM networks should also be tested, especially for the case of multiple classes, as well as different choices of inputs and outputs.

Acknowledgments

We would like to thank all the members of the Networks and Communications Research Group for their helpful discussion and advice, and especially Mr. Godfrey Onyiagha, Dr. Denise Gorse and Dr. Chris Christodoulou for their invaluable suggestions. Participation to the SOCRATES exchange programme was made possible by the interest of Prof. Luciano Lenzini from the department of Information Engineering in Pisa, and Mrs. Janine Lajudie, exchange coordinator for the department of Electronic Engineering at King's College London.

References

[1] R. Bolla, F. Davoli and M. Marchese. "A Simple Model for Cell Loss Probability Evaluation in an ATM Multiplexer". In *Fourth Workshop on Performance Modelling and Evaluation of ATM Networks*, pp. 51/1-51/9, July 1996.

[2] T. G. Clarkson, D. Gorse, J. G. Taylor and C. K. Ng. "Learning Probabilistic RAM Nets Using VLSI Structures". *IEEE Transactions on Computers*, n. 41, vol. 12, pp. 1552-1560, December 1992.

[3] T. G. Clarkson, p*RAM-256 VLSI Neural Network Processor Data Sheet*, http://crg.eee.kcl.ac.uk/clarkson/pram.html, Department of Electronic Engineering, King' s College London, November 1995.

[4] Clarkson T G, Ng C K & Guan Y, The pRAM: an adaptive VLSI chip. IEEE Transactions on Neural Networks, Special Issue on Neural Network Hardware, Vol. 4, No 3, 408-412, 1993.

[5] D. Gorse and J. G. Taylor. "A Continuous Input RAM-Based Stochastic Neural Model". *Neural Networks*, n. 4, pp. 657-665, 1991.

[6] R. Guérin, H. Ahmadi and M. Naghshineh. "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks". *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 968-81, 1991.

[7] I. Habib, A. Tarraf and T. Saadwi. "A neural network controller for congestion control in ATM multiplexers". *Computer Networks and ISDN Systems*, n. 29, pp. 325-334, 1997.

[8] H. Hiramatsu. "ATM Communications Network Control by Neural Networks". *IEEE Transactions on Neural Networks*, n. 1, vol. 1, pp. 122-130, March 1990.

[9] H. Hiramatsu. "Integration of ATM Call Admission Control and Link Capacity Control by Distributed Neural Networks". *IEEE Journal on Selected Areas in Communications*, n. 9, vol. 7, pp. 1131-1138, September 1991.

[10] H. Hiramatsu. "Training Techniques for Neural Network Applications in ATM". *IEEE Communications Magazine*, pagg. 58-67, Ottobre 1995.

[11] R. J. T. Morris and B. Samadi. "Neural Network Control of Communications Systems". *IEEE Transactions on Neural Networks*, n. 5, vol. 4, pp. 639-649, 1994.

[12] J. E. Neves, M. J. Leitão and L. B. Almeida. "Neural Networks in B-ISDN Flow Control: ATM Traffic Prediction or Network Modelling?". *IEEE Communications Magazine*, pagg. 50-56, Ottobre 1995.

[13] C. G. Onyiagha, X. Krasniqi and T. G. Clarkson. "Probabilistic RAM Neural Networks in an ATM Multiplexer". In *Solving Engineering Problems with Neural Networks, Proc. Int. Conf. Engineering Applications of Neural Networks*, pp. 229-232, June 1996.

[14] C. G. Onyiagha, X. Krasniqi and T. G. Clarkson. "Adaptive Access Control of ATM Traffic Using Neural Networks. *IEEE Globecom*, pp. 1:201-205, November 1996.

[15] Y. Park and G. Lee. "Applications of Neural Networks in High-Speed Communication Networks". *IEEE Communications Magazine*, pagg. 68-74, Ottobre 1995.

[16] H. G. Perros and K. M. Elsayed. "Call Admission Control Schemes: A Review". *IEEE Communications Magazine*, pagg. 82-91, Novembre 1996.

Figure Captions

Figure 1: pRAM network topologies: feed-forward fully interconnected (a), pyramidal (b), trapezoidal (c).