Traffic and Congestion Control in ATM Networks Using Neuro-Fuzzy Approach

Suriti Gupta
College of Technology and Engineering
Udaipur-313001

Vinod Kumar
College of Technology and Engineering
Udaipur-313001

ABSTRACT
In this paper, a neuro-fuzzy based Call Admission Control (CAC) algorithm for ATM networks has been simulated. The algorithm presented employs neuro-fuzzy approach to calculate the bandwidth require to support multimedia traffic with QoS requirements. The neuro-fuzzy based CAC calculates bandwidth required per call using measurements of the traffic via its count-process, instead of relying on simple parameters such as the peak, average bit rate and burst length. Furthermore, to enhance the statistical multiplexing gain, the controller calculates the gain obtained from multiplexing multiple streams of traffic supported on separate virtual (i.e, class multiplexing).

Keywords
Call Admission Control (CAC), ATM networks, Neuro-fuzzy control.

1. INTRODUCTION
Asynchronous transfer mode (ATM) is a key technology for integrating broad-band multimedia services (B-ISDN) in heterogeneous networks. The ATM forum has specified several service categories in relation to traffic management in an ATM network. The available bit rate (ABR) service [1] guarantees a zero-loss cell rate if the source obeys the dynamically varying traffic management signals from the network. It is primarily driven by telecommunications companies and is a proposed telecommunications standard for Broadband ISDN. ATM is a connection-oriented protocol that supports both connection oriented and connectionless services with Constant Bit Rate (CBR) and Variable Bit-Rate (VBR) traffic characteristics. ATM is a high speed packet switching technique using short fixed length packets called cells. Fixed length of cells simplifies the design of an ATM switch at which high switching speed is involved. The selection of short fixed length cells reduces the delay, and most significantly the jitter, for delay sensitive services such as voice and video. ATM is essentially a connection-oriented technique, though it has been envisioned as a basis for supporting all services, connectionless as well as connection oriented. ATM is a method of breaking up data into 53 byte cells and transmitting them from place to place on a network over a series of switches.ATM provides the flexibility to integrate the wide variety of service including video telephony and audio. ATM connection can be multiplexed deterministically or statistically. Allocating resources in accordance to the peak rate do deterministic multiplexing. Here congestion is eliminated and delay as well as delay variation, is small. While this approach is safe, average bandwidth utilization is poor, in general and this scheme is not much different from the SDH scheme. In statistical multiplexing, the peak rate of all the connection is above the link capacity. Using this type of multiplexing, one attempts to solve the unused bucket problem of Synchronous Transfer Mode (STM), by statistical multiplexing several connection on the same link, based on he traffic characteristics.

In other words, if a large number of connections are there, all of them may be assigned the same link in the hope that statistically they will not burst at the same time. If some of them do burst simultaneously, there is sufficient elasticity that the burst can be buffered up and put in subsequently available free buckets. This is called statistical multiplexing, and it allows the sum of peak bandwidth requirements of all the connections on the link to even exceed the aggregate available bandwidth of the link under certain conditions of discipline. This was impossible in the STM network and this is the main advantage of the ATM networks. But there is finite probability that at any time, offered input at the switch is higher than the link capacity. This can partly upset the buffering at the input. However, a longer duration of congestion might result in cell loss and higher delay. Through statistical multiplexing, ATM offers improved bandwidth utilization. However, it is exactly statistical multiplexing that creates some other problems like congestion.Since the ATM forum decided to use a closed-loop rate based congestion control scheme as the standard for the ABR service [2], several feedback control schemes, were proposed in the literature [3-6] and without neural and fuzzy logic. There is no mathematical analysis provided to demonstrate the performance of these controllers in terms of QoS.In this paper, a neuro-fuzzy based Call Admission Control (CAC) algorithm for ATM networks has been simulated. The algorithm presented employs neuro-fuzzy approach to calculate the bandwidth require to support multimedia traffic with QoS requirements. The neuro-fuzzy based CAC calculates bandwidth required per call using measurements of the traffic via its count-process, instead of relying on simple parameters such as the peak, average bit rate and burst length. Furthermore, to enhance the statistical multiplexing gain, the controller calculates the gain obtained from multiplexing multiple streams of traffic supported on separate virtual (i.e, class multiplexing). In order to simplify the design and obtain small reaction time, the controller is simulated using a hierarchical structure of a bank of small size, parallel ANN units. Each unit is a feed forward back propagation ANN that has been trained to learn the complex non-linear function relating different traffic patterns, with the corresponding received capacity. The neuro-fuzzy approach is effective in achieving more accurate results.

2. OVERVIEW OF NEURAL NETWORKS (NN)
A NN is a parallel distributed information processing structure consisting of processing elements (which can posses a local memory and can carry out localized information processing operations) interconnected via unidirectional signal channel called synapses. Each processing element has a single output connection that branches (“fans out”) into as many collateral

connections as desired; each carry the same signal- the processing element output signal. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is, it must depend only on the current values of the input signals arriving at the processing element via impinging connections and on values stored in the processing elements local memory. An abstract model of a neuron is shown in Fig. 1.

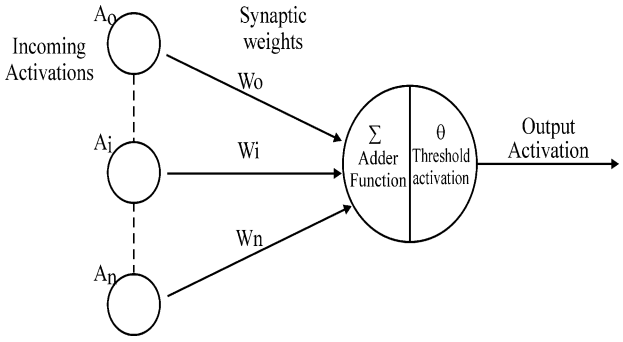

Fig 1: Abstract Neuron Model

A fuzzy system is basically made of a fuzzifier, a defuzzifier, an inference engine, and a rule base as shown in Fig. 2. The role of the fuzzifier is to map the crisp input data values to fuzzy sets defined by their membership functions depending on the degree of "possibility" of the input data. The goal of the defuzzifier is to map the output fuzzy sets to a crisp output value. It combines the different fuzzy sets with different degrees of possibility to produce a single numerical value. The fuzzy inference engine defines how the system should infer through the rules in the rule base to determine the output fuzzy sets.

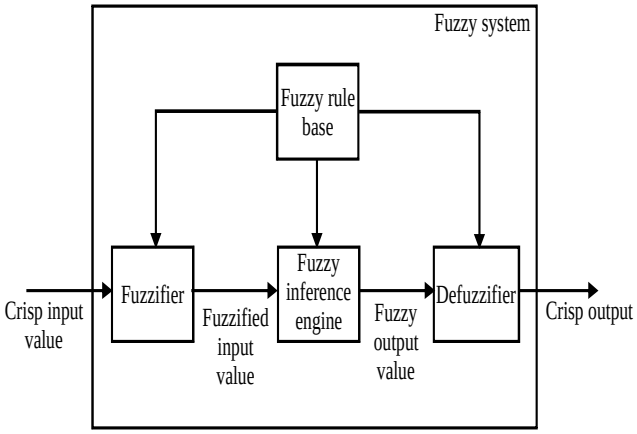

Fig 2: Block Diagram of a Fuzzy System

A binary threshold divides the buffer space in two parts. Below or equal to the threshold level, every arriving cell is given entry to the network and above the threshold every cell is rejected [7]. The change from entry in the network to rejection is abrupt. A gradual change is more intuitive here. This can be done with fuzzy threshold. For defuzzification, with the set such as shown in Table 1, weighted average is used. A typical example is given in the following section.

Table 1. A typical de-fuzzification

SET	L	B	M	A	H
I	0.05	0.25	0.50	0.75	0.95
II	0.05	0.20	0.40	0.60	0.80
III	0.05	0.20	0.40	0.70	0.95
IV	0.075	0.175	0.35	0.60	0.95
V	0.05	0.10	0.25	0.5	0.95

Where, L : Low Set; B : Below Medium Set; M : Medium Set; A : Above Medium Set; H : High Set

2.1 The CAC Algorithm

The objective of admission controller is to keep the service quality under the requested value by rejecting some of the call setup requests, while connecting as many calls as possible.

Before describing the NN controller, the admission control algorithm has been discussed. Consider an ATM link with capacity C_{link} b/s and assume multiple sources with multiple classes of traffic (i.e., different traffic characteristics and QoS requirements). The measure of QoS is Cell Loss Rate (CLR). Also consider that there are N virtual path supports sources with similar class of traffic. For example, two types of video traffic sources with different QoS will be supported on separate virtual paths. Also, if two traffic sources share that same QoS but have different characteristics each will be supported on a separate virtual path.The reason for this classification is to simplify the CAC design since it is easier to designs smaller size NNs, each trained to learn a specific traffic pattern, rather than a complex architecture capable of learning a large set of traffic patterns.Presented CAC algorithm can be used in case of static or dynamic capacity allocation. In case of static capacity allocation, the capacity of each virtual path will be permanently assigned by the network management layer. The decision maker in this case will make a decision to accept or reject the call based on the following equation:

$$\alpha_i = C_{vpi} + X - C_{ai} - C_{new}$$

Where

α_i - output value from the CAC (If this value is greater than or equal to zero, then the new call will be accepted otherwise it will be rejected).

C_{vpi} - Fixed capacity assigned for virtual path (I) by the network layer.

C_{link} – Total allocated capacity for all virtual paths on the link. Module II is used to calculate this capacity from the used capacity of each individual virtual paths

C_{ai} - Actual capacity allocated on virtual path (I). Module I is used to calculate this capacity from traffic measurement.

C_{new} – Capacity calculated for a new call.

X - Gain due to class multiplexing, and is calculated using the following equation:

$$X = \frac{\sum C_{vpi} - C_{link}}{N}$$

Where N is the number of virtual paths.

In the case of dynamic capacity allocation, the capacity of each virtual path will be requested from the network management layer dynamically. The decision maker in this case will generate a request for the network layer to increase (decrease) the capacity required for the virtual path based upon requests from incoming calls requesting bandwidth over this virtual path. The capacity required for the virtual path to accept an incoming call is determined from :

$$C_{vpi} = C_{ai} - C_{new}$$

If the request for C_{vpi} is granted, then the call is accepted otherwise it is rejected.

2.2 Neural Network Module I

In this module, multiple three-layered back propagation NNs were trained to learn the relationship between the characteristics of the cell arrival process, the QoS, and the capacity required to support it. The inputs to the NN are former two, while the later is the output. The arrival process is measured via its count process, which keeps account of the number of cells arriving in consecutive time periods. The NNs have been trained to capture the correlations that exist among cells arrivals. The choice of measurement period and the sampling period has been decided by the need to teach the NN the correlations of the arrival process The QoS was presented via cell loss rate, which also indicates the size of the buffer. The output from the module is then an accurate estimate of the bandwidth, which is required to support the input arrival process. This estimate is accurate regardless of the type of the input traffic or the number of multiplexed sources. This is because the design is scalable and parallel in the sense that the NN module is composed of a bank of n parallel smaller NN units, each is trained to learn the characteristics of a certain class of traffic with similar bandwidth requirements. To add on a new service or a new type of multimedia traffic, one has to add on a simple NN unit to the already existing bank. The NN module is represented by

$$\text{Capacity} = NN_f \{H(I), [W]\}$$

Where, "Capacity" represents the output from the NN model, NN_f denotes the NN transfer function, [W] represents the weight matrices of the hidden and output layers, and [H (I)] is an input vector with K elements. Each element is a value of cells count process N(0, T_s) during a sampling period T_s, measured over a measurement period Tm where

$$T_m = KT_s$$

The two parameters T_m and T_s were determined to capture the correlations of the traffic while keeping the size of the NN simple. For example a large value of T_s would imply a smaller size input vector [H(I)] hence, a smaller number of input Processing Elements (PEs) and simple NN structure. However, the price is that the NN would not be able to capture the statistical characteristics of the traffic, leading to poor estimate of the required bandwidth. We have taken T_s equal to the duration of one video frame (1/30s) and T_m to be 600 frame duration which implies 600 inputs to the NN

Now after determining the size of the inputs, the number of hidden layer Es has to be determined. There can be infinite number of traffic patterns (whether from single or multiple sources) that can cover the range of bandwidth from 0 to 200 Mb/s. This implies a huge number of PEs in the hidden layer, adding to the complexity of the NN architecture. Knowing that the number of patterns that a NN can deterministically recognize is equal to twice the number of its weight. Thus, to make the NN architecture simple, a hierarchical structure using banks of parallel NN units has been employed. The hierarchical structure has been used also in , and proved to be effective in learning different features of patterns that the single NN approach . Another advantages of hierarchical approach is that the processing elements of the hidden layer only have to process a limited number of traffic patterns one step at a time, therefore the hierarchical model can provide capabilities that otherwise would not be possible to deliver.To employ the hierarchical model, the elements of the count process input vector have been divided into two levels. The first level represents the cell arrival rate in 20 frames together with their equivalent bandwidth, whereas the second level contains 30 elements representing the bandwidth output from the first level and the required bandwidth for the whole traffic.Equivalently, these 20 elements contain the information that represents the cell arrival rates in 600 frames of video traffic. However, there still problem with the neural network model. Since the number of inputs for a single NN is 20 and range of these inputs should cover a bandwidth range from 0 to 200 Mb/s. This means that a large number of traffic patterns would still be learned by the neural network implying a massive number of neurons in the hidden layer.To further simply the design, the traffic into multiple ranges according to its bandwidth requirement and design a neural network unit to learn each range. Therefore, a NN has been assigned for traffic requiring bandwidth from 0 to 20 Mb/s, another neural network for traffic requiring bandwidth from 20 to 40 Mb/s and so on until 200 Mb/s. Because of this an extra important feature, which is flexibility to extend and build upon, it has been achieved. Since all that is required to add more traffic patterns with new requirements is to add an extra NN unit, or more, according to the new requirements.Fig. 3 shows the model used to calculate the bandwidth from the common traffic. This model has tow phases of operation, the training phase and the operation phase. In the training phase, the first of NNs were trained to recognize the relationship between a count process vector consisting of 20 elements, and the capacity requirements for these 20 elements. In the operation phase, outputs from Module I represent the allocated bandwidth for traffic based on its count process vector. The comparators are used to allow only one output at one time from different NNs in the first level. The function of the comparators is to compare whether the output capacity from the NNs falling within its specific range, then this output will be passed to the OR function. Otherwise the comparator will generate a zero output. The shift register in his model will hold the inputs to each neural network level.

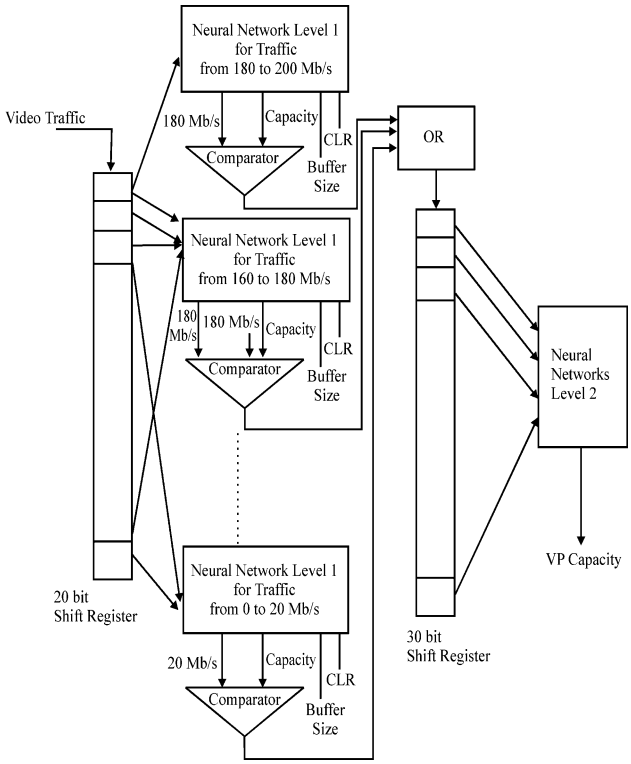

Fig 3: Block diagram of Module I

The second level of NN learns the relationship between the 30 outputs from the previous level and the equivalent capacity assigned for the traffic. Hence the number of inputs for all NNs in this level is kept to 30 inputs.

2.3 Neural Networks Module II

The purpose of Module II is to include the gain achieved from statistical multiplexing of multiple virtual paths each with different class of traffic and bandwidth. For example, consider two virtual paths VP1 and VP2, each is supporting a certain number of calls, assuming that the bandwidth computed for each virtual path, from Module I, is 10 Mb/s and 34 Mb/s, respectively. The total assigned capacity for both virtual paths (I) is not 44 Mb/s (the sum) but the value that is less than that, say 40 Mb/s. Hence, the 4 Mb/s gained is called class multiplexing gain. Module II also uses a three-layered backpropagation NN to learn the relationship between its inputs and outputs. A NN is Module II is responsible to learn the relationship between the capacity assigned for each virtual path (from Module I) and the actual assigned capacity for these paths on the links. For two virtual paths, the structure of NN is simple. It has two inputs representing the bandwidth assigned for two virtual paths, four hidden neurons, and one output which is the required capacity to support these paths. The CAC also computes the requested increase (or decrease) in the capacity for each virtual path based upon suitable entries such as a desired cell loss rate per virtual path.

3. CAC SIMULATION

First, the number of NN units and their size has to be decided. The three layer backpropagation NN have been used. From the block diagram of Module I (Fig. 3) it is clear that there are ten NN blocks of level 1, each of them have twenty input layer neurons and one output layer neurons. Number of neurons in the hidden layer is taken to be forty. These ten units have been trained for the ranges 0 – 20 Mb/s, 20 – 40 Mb/s, upto 180 – 200 Mb/s. The training data has been generated by using autoregressive Markov model. The NN unit of level 2 has thirty input neurons as in Fig. 3 and one output layer neuron. In this unit, sixty neurons in the hidden layer have been considered. This unit has been trained for the range 0 – 200 Mb/s. Since only two virtual paths, VP1 and VP2 have been considered, thus the NN of Module II has two neurons in the input layer. It has one output, thus one output neuron is there and four neuron in the hidden layer have been taken.

The NN of Module II has been trained to learn the relationship between the capacity assigned for each virtual path (From Module I) and actual assigned capacity for these paths on the link. For the training purpose of NN units of size 20, 40, 1, 2500 sets of data has been used, each set or vector consists of 21 elements out of which first 20 elements are input to the NN and the last one is target or desired output. While for NN units of size 30, 60, 1, 3000 sets are used. In this case each set have 31 elements out of which 30 are input and the 31st one is the target or the desired output. For the NN of module II 2500 sets are used. Each set of which have three elements, first two are input and the last one is the desired output. After completing training part twelve weight files are obtained which was used in the simulation of CAC model.

4. RESULTS AND DISCUSSION

The training of feed forward backpropagation NNs, simulation of autoregressive Markov process, and CAC algorithm has been described. The details of constraints are given in Table 2. That is, the range of capacity for which NNs is trained, learning parameters, number of iteration, and final error.It is clear form the training profiles that the convergence his occurred and the NNs have stabilized. From this weights files are obtained, which contain the optimum weights of the connections of the NNs, and are used in testing or operation phase of CAC model.

5. CONCLUSION

In this paper, mathematic analysis and simulations are made using popular and empirical stochstatic method to model problem. At first we analyzed using Markov Process, BOX-MULLER algorithm and then we use Feedforward backpropagation network. Finally, to discuss the results of CAC model, simulated, we consider some examples in which the virtual path capacities are assigned to be C_{vp1} and C_{vp2} and request for new connection is made with capacity C_{new}. CAC model will now calculate the actual capacity allocated on the virtual paths C_{a1} and C_{a2}, and also the link capacity C_{link} and made decision to accept or reject the call on respective link or path.

6. REFERENCES

[1] F. Bonomi and K. Fendick, "The ratebased flow control framework for the ABR ATM service," IEEE Network, vo1.9, n0.2, pp. 25-39, 1995.

[2] F. Bonomi and K. Fendick, "The rate-based flow control framework for the ABR ATM service," IEEE Nerwork, vo1.9,no.2, pp. 25-39,1995.

[3] B. Chen, Y. Zhang, J. Yen, and W. Zhao, "Fuzzy adaptive connection admission control for real-time applications in ATM-based heterogeneous networks," Jotirnal of Intelligent and Fmq~ Systems, vo1.7, no.2, 1999.

[4] R.-G. Cheng, C.-J. Chang, and L.-F. Lin, "A QoS provisioning neural fuzzy connection admission controller for multimedia high-speed networks,"

IEEG'ACM Transaction on Networking, ~01.7, no.1, pp. 1 1 1-121, 1999.

[5] A. Hiramatsu, "ATM communications network control by neural networks," IEEE Transaction on netiraf networks, vol.l,no.l,pp.122-130, 1990.

[6] T.V.Lakshman, P. P. Mishra and K.K. Ramakrishnan, "Transporting compressed video over ATM networks with explicit-rate feedback control," IEEE/ACM Transactions on Networking, Vol. 7,No. 5, October 1999.

[7] Parag Jain, Sandip Vijay and S. C. Gupta "Fuzzy Congestion Control Scheme in ATM Networks". Global Journal of Computer Science and Technology, Vol. 9 Issue 5 (Ver. 2.0), January 2010 PP 68.