Neural-network connection-admission control for ATM networks

R.-G.Cheng
C.-J.Chang

Indexing terms: Neural networks, Connection-admission control, ATM networks

Abstract: ATM connection-admission control (CAC) using neural networks offers improvement over conventional CAC but creates some difficulties in real operation, such as complicated training processes. This is because ATM traffic characteristics are quite diverse, and quality of service (QoS) and bandwidth requirements vary considerably. A neural-network connection-admission control (NNCAC) method which can overcome these difficulties by preprocessing neural-network input parameters is proposed. The NNCAC method introduces a unified metric for input-traffic parameters by utilising robust analytical results of the equivalent-capacity method. It diminishes the estimation error of the equivalent-capacity method, due to modelling, approximation and unpredictable statistical fluctuations of the system, by employing the learning capability of a neural network. The method further considers the congestion status parameter and the cell loss probability, which provides insight information about the system. Simulation results revealed that the proposed NNCAC method provided a 20% system-utilisation improvement over Hiramatsu's neural-network CAC scheme and a 10% system-utilisation improvement over the fuzzy-logic-based CAC scheme, while maintaining QoS contracts. It was also found that the NNCAC method provided utilisation comparable with that of the NFCAC scheme but possessed a lower cell loss probability. NNCAC is suitable for designers who are not familiar with fuzzy-logic control schemes or have no ideas about the requisite knowledge of CAC.

1 Introduction

Asynchronous transfer mode (ATM) is a key technology for integrating multimedia services in high-speed networks. Because of bursty traffic characteristics and various quality of service (QoS) and bandwidth requirements for these multimedia services, an ATM network must have an appropriate traffic control, which includes connection-admission control (CAC) and congestion control, to guarantee the QoS for existing calls as well as to achieve high system utilisation.

Conventional CAC and congestion-control schemes [1–7] based on mathematical analyses provide robust solution for different traffic environments at steady state. However, their design and implementation, which directly utilise capacity estimation and buffer thresholds, suffer from some fundamental limitations. One of the limitations is that, because of the difficulty in acquiring complete statistics on traffic input to ATM networks, it is not easy to determine accurately the effective thresholds or the equivalent capacity. The rationale and principles underlying the nature and choice of thresholds or equivalent capacity under dynamic conditions are unclear [2, 8]. The decision process is full of uncertainty. In other words, because of modelling, approximation and the unpredictable statistical fluctuations of the system, decision error always accompanies these control schemes and results in performance degradation.

Recently, neural networks have been applied widely to deal with traffic-control-related problems in ATM networks [9, 12]. The self-learning capability of the neural network is used to characterise the relationship between input traffic and system performance. In [9], Hiramatsu proposed a connection-admission controller which uses a neural network. This admission controller employed the offered traffic characteristics, QoS requirement and actual network-operation-performance measures to decide whether to accept or reject a call-setup request. The results showed that the neural network learned a complicated boundary for a call-acceptance decision. In [11], Tran-Gia and Gropp investigated different aspects of using neural networks to perform CAC. The numbers of active connections of each traffic class were selected as the inputs to the neural network for connection-acceptance decision. Numerical results showed that the neural-network approach yielded significant benefits. In [12], it was reported that a neural network used to produce a feedback-control signal adaptively was able to alter the source rate to relieve congestion. The control signal achieved an optimum in the sense that it maximised the performance-measure function defined by the authors. However, in most of the proposed neural-network approaches, all users for each kind of service were selected as the input parameters. Thus, the neural-network dimensions and the learning time required will increase as the number of traffic types grows; the

IEE Proceedings online no. 19971088

Paper first received 18th June and in revised form 20th December 1996

The authors are with the Department of Communication Engineering, National Chiao Tung University, Hsinchu, Taiwan 300, Republic of China

system complexity would be increased by system upgrades. The application of neural networks to CAC would thus be limited to simplistic traffic environments, such as limited traffic types, simplified traffic sources etc.

This paper proposes a neural-network connection-admission-control (NNCAC) method for an ATM traffic controller. The NNCAC method retains the benefits of the two approaches mentioned above while reducing their drawbacks. It adopts a multilayer feedforward neural network with preprocessed inputs: an equivalent bandwidth, a congestion-status parameter and a cell-loss probability. The equivalent bandwidth is the first input and is obtained by transforming the traffic characteristics (usually described by three traffic parameters peak bit rate, average bit rate and mean peak-rate duration) of a new call into a unified metric. This transformation can reduce the dependence of the neural-network control mechanism on traffic types; it greatly reduces the dimensions of the NNCAC method and saves a large percentage of the learning time. The second input is the congestion-status parameter generated by a congestion controller. Congestion control is so correlated with CAC that it should be taken into account. One of the most frequently used congestion-control methods is the buffer-threshold method, in which network congestion sounds an alarm when a queue length exceeds some predefined threshold. Network congestion is then averted by regulating the traffic flow of incoming sources according to a congestion-status parameter. The last input is the cell loss probability, a measure of QoS, which is used as a system-performance feedback to indicate how effective the system-control performance is. The NNCAC method uses the back-propagation training algorithm to adjust link weights and to learn the proper call-acceptance decision boundary from training data. Simulation results reveal that the proposed NNCAC provides system utilisation superior to Hiramatsu's neural-network CAC scheme [9] and fuzzy-logic-based CAC scheme [13], while maintaining QoS contracts. It was also found that the NNCAC method provides a utilisation comparable with that of the NFCAC scheme [14].

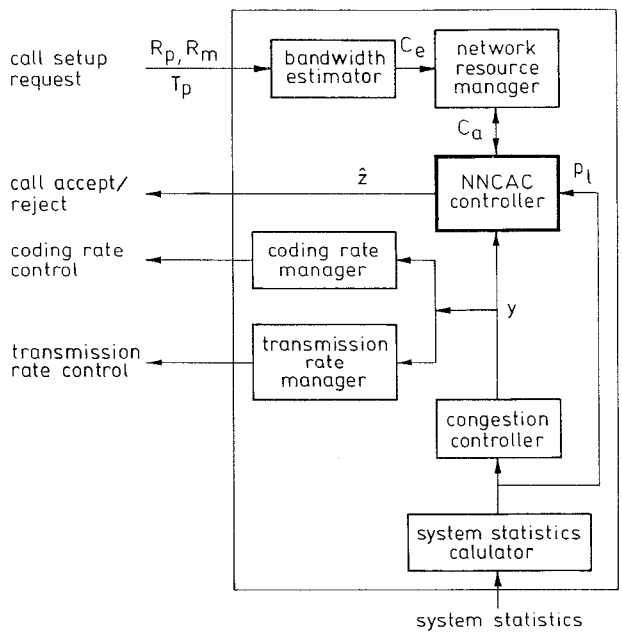

Fig. 1 *Possible realisation of an ATM traffic controller*

2 Neural-network connection-admission control

A functional block diagram of the ATM traffic controller is depicted in Fig. 1. As shown, the traffic controller includes a system-statistics calculator, a congestion controller, a coding-rate manager, a transmission-rate manager, a bandwidth estimator, a network-resource manager and a neural-network connection-admission controller. The system-statistics calculator measures the queue length q, the queue-length-variation rate Δq and the cell-loss probability p_l of each queue. The parameters q, Δq and p_l are output to the congestion controller. The congestion controller receives parameters q, Δq and p_l of specific queues and applies a congestion-control method to obtain the congestion-status parameter y. The coding-rate manager outputs a signal to traffic sources which increases or decreases the rate at which cells are generated in response to the congestion-status parameter. The transmission-rate manager, however, outputs a signal which blocks (suspends) or unblocks (resumes) delivery of cells from those sources which cannot tolerate any data losses. The bandwidth estimator receives parameters from a traffic source which requests admission into the ATM network node and estimates its equivalent bandwidth C_e. These parameters are peak bit rate (PBR), average bit rate (ABR) and peak-bit-rate duration (PBRD), denoted by R_p, R_m and T_p, respectively. The network resource manager calculates the bandwidth currently available for allocation, denoted by C_a, according to the equivalent bandwidth required by existing connections. When a new request for connection admission with bandwidth C_e is accepted, a new value of C_a is generated by subtracting C_e from the old value of C_a. Conversely, when an existing connection with bandwidth C_e is disconnected, a new value of C_a is updated by adding C_e to the old value of C_a. Initially, C_a is set to 1. The NNCAC controller is a neural network which receives the bandwidth C_a currently available for allocation, the congestion status parameter y and the cell loss probability p_l as input parameters. Based on the three parameters, the NNCAC controller generates a decision signal $\hat{z}$ and sends the signal back to the new connection to indicate acceptance or rejection of the new call request. In this paper, fuzzy implementations of the congestion controller and the bandwidth estimator are used. Details of these implementations can be found in [13].

The NNCAC controller is a multilayer feedforward neural network. The neural network possesses an ability to approximate a perfect connection-acceptance decision function from input/output data pairs $\{X, Z\}$. Consider a feedforward neural network NNCAC(X, W), where X represents an input vector and W represents a set of weight vectors updated by some learning rules; denote the call acceptance decision function by $Z = f(X) : D \subseteq R^{n_i} \rightarrow R^{n_o}$, where D is a compact metric space on R, and n_i (n_o) is the input- (output-) space dimension. The Stone–Weierstrass theorem [15] showed that NNCAC(X, W) can be trained to approach $f(X)$ asymptotically as much as possible. In other words, the NNCAC controller formulates the CAC problem as a pattern recognition problem: upon recognition of the input pattern X, a yes/no decision must be made to accept/reject a connection request.

A backpropagation learning algorithm [16] is here used to train the NNCAC controller to solve such pat-

tern recognition problems. Let $X(i)$ denote the vector randomly sampled from D and used as an input to the NNCAC controller at time instant t_i; let NNCAC[$X(i)$, W] = $\hat{z}(i)$ denote the corresponding decision of the NNCAC controller, and let $f[X(i)] = z(i)$ denote the desired decision. The objective of the backpropagation learning algorithm is to minimise the decision error E by recursively adjusting its weight in each layer, where E is defined as

$$E = \frac{1}{2}\|\text{NNCAC}[X(i), W] - f[X(i)]\|^2 = \frac{1}{2}[\hat{z}(i) - z(i)]^2 \quad (1)$$

Consider an M-layer feedforward neural network. Each layer has a number of processing elements called neurons which are fully interconnected via adaptive weights. Neurons in the input layer (layer $k = 1$) do not process the input data; they simply store input-data values. Neurons in the hidden layers ($2 \leq$ layer $k \leq M - 1$) and output layer (layer $k = M$) perform two operations. The jth neuron in the kth layer, for example, first calculates a weighted sum, denoted by $S_j^{(k)}$, of all outputs $o_i^{(k-1)}$ of the $(k-1)$th layer. $S_j^{(k)}$ is given by

$$S_j^{(k)} = \begin{cases} x_j & \text{if } k = 1 \\ \sum_{i=1}^{n_{k-1}} w_{ji}^{(k)} o_i^{(k-1)} & \text{if } 2 \leq k \leq M \end{cases} \quad (2)$$

where x_j is the input variable of the jth neuron in the input layer, n_{k-1} is the number of neurons in layer $(k-1)$, and $w_{ji}^{(k)}$ is the weight of the link connected from the ith neuron in layer $(k-1)$ to the jth neuron in layer k. After that, the neuron further transforms $S_j^{(k)}$ into output $o_j^{(k)}$ via an activation function $G(\cdot)$. $o_j^{(k)}$ is expressed as

$$o_j^{(k)} = \begin{cases} S_j^{(k)} & \text{if } k = 1 \\ G[S_j^{(k)}] & \text{if } 2 \leq k \leq M \end{cases} \quad (3)$$

The adjustment of weights is based on a steepest-descent algorithm [16]. It can be expressed as

$$w_{ji}^{(k),new} = w_{ji}^{(k),old} - \eta \frac{\partial E}{\partial w_{ji}^{(k)}}\Big|_{w_{ji}^{(k)} = w_{ji}^{(k),old}} \quad (4)$$

where η is a gain term which determines the learning rate of the link weight. η is usually set equal to a positive constant less than unity. To obtain the partial derivative for the quadratic error E, an error term produced by the jth neuron in layer k, denoted by $\delta_j^{(k)}$, is obtained from

$$\delta_j^{(k)} = -\frac{\partial E}{\partial S_j^{(k)}}, 1 \leq k \leq M, 1 \leq j \leq n_k \quad (5)$$

It was shown in [16] that the error signals $\delta_j^{(k)}$ can be computed according to a recursive procedure of the generalised delta learning rule [16] described as follows:

$$\delta_j^{(k)} = \begin{cases} G'[S_j^{(k)}] \sum_l \delta_l^{(k+1)} w_{lj}^{(k+1)} & \text{for } 2 \leq k \leq M-1 \\ (z - \hat{z}) G'[S_j^{(k)}] & \text{for } k = M \end{cases} \quad (6)$$

Once these error-signal terms have been determined, the partial derivative for the quadratic error can be computed directly by

$$\frac{\partial E}{\partial w_{ji}^{(k)}} = \frac{\partial E}{\partial S_j^{(k)}} \frac{\partial S_j^{(k)}}{\partial w_{ji}^{(k)}} = -\delta_j^{(k)} o_i^{(k-1)} \quad (7)$$

and the update rule for the backpropagation algorithm is then given by

$$w_{ji}^{(k),new} = w_{ji}^{(k),old} - \eta \frac{\partial E}{\partial w_{ji}^{(k)}} = w_{ji}^{(k),old} + \eta \delta_j^{(k)} o_i^{(k-1)} \quad (8)$$

The procedure for setting up a training-data table is described as follows. Consider an ATM network element with multimedia-source users; the ith new user declares its traffic parameters (i.e. peak rate, mean rate and peak-rate duration). These parameters along with system performance statistics (i.e. queue length, cell-loss probability etc.) are converted into preprocessed parameters (i.e. available bandwidth C_a, congestion status parameter y and cell loss probability p_l) and then fed into the NNCAC controller (see Fig. 1). The NNCAC controller outputs a decision signal $\hat{z}$ to indicate acceptance or rejection of the call request. The correctness of the decision signal $\hat{z}$ will be verified by a desired output z which is obtained by

$$z = U(P_{QoS} - \bar{p}_l) \quad (9)$$

where $\bar{p}_l$ is the moving average of the next m-step measurements of the cell loss probability, P_{QoS} is the QoS requirement of the cell loss probability, and $U(x)$ is a unit step function defined as

$$U(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

In other words, if the resulting $\bar{p}_l$ satisfies the QoS requirement P_{QoS} (i.e. $\bar{p}_l \leq P_{QoS}$), then $z = 1$, denoting that the call-setup request should be accepted; otherwise $z = 0$, denoting that the call-setup request should be rejected.

As shown in Fig. 2, the desired output z, along with C_a, y, p_l and $\hat{z}$ for the ith new call, are written into the ith location of the training-data table. A sequence of training data is then collected and stored. According to these training data, the NNCAC controller can be offline-trained to approximate the complicated call-acceptance-decision function.

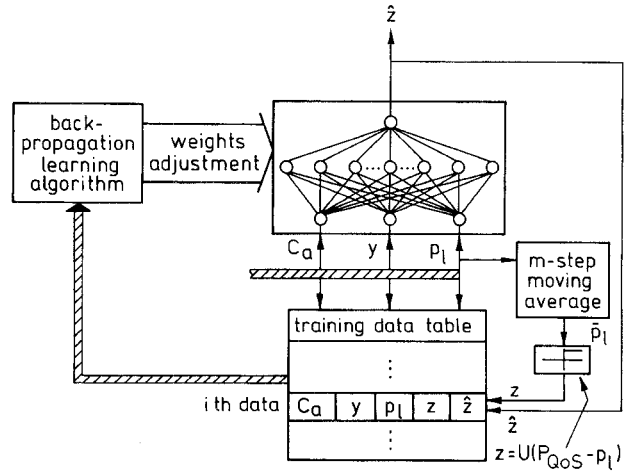

Fig.2 *NNCAC training-data table setup procedure*

However, in a real operation, traffic characteristics will change over time and the call-acceptance-decision function must be modified accordingly to prevent errors. The susceptibility of the NNCAC controller to crashes caused by propagation of decision errors implies that a further online training procedure must be provided. For online learning, the training data are updated from time to time to capture the dynamic

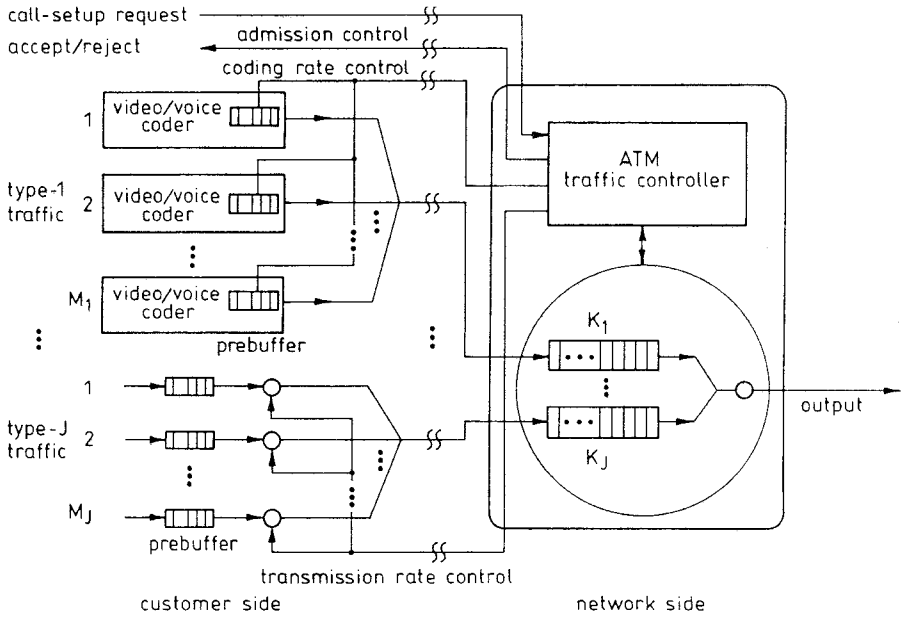

Fig. 3 *System model*

behaviour of the system. To learn an accurate decision function (or equivalently, an optimal weight W^*) by backpropagation, it is necessary to train NNCAC over the training-data input space in random sequences. Here, a leaky pattern table (LPT) selection method [9] is used to update the learning table and to select proper data for online training. Use of the LPT selection method enables the NNCAC controller to capture the dynamic behaviour of network fluctuations as soon as possible and to reduce decision errors [9].

3 Simulation results

3.1 Simulation model for an ATM network

The system model of an ATM-network element containing an ATM traffic controller is shown in Fig. 3. The ATM-network element has a total bandwidth of $C = 1$, in which a C_j fraction of the total bandwidth is allocated to the jth type of traffic, $\Sigma_{j=1}^{J} C_j \leq 1$, and it consists of J types of queues in which the jth queue with length K_j is for the jth type of QoS, $j = 1 \ldots J$. The traffic controller receives a request from a traffic source during the connection-setup phase and responds to the traffic source with a decision signal either to accept or to reject the request. In addition, the traffic controller periodically monitors the network-traffic load. Based on this monitoring, the traffic controller averts congestion by sending congestion-status signals to the traffic sources. After receiving the congestion-status signals, the traffic sources adjust the rate at which they deliver cells to the network. The cell rate can thus be adjusted by either altering the encoding process or suspending/resuming cell production.

In the simulations, it is assumed that there is delay-sensitive traffic and delay-insensitive traffic. Delay-sensitive traffic such as voice and video services is called type-1 for which $QoS_1 = 10^{-5}$; delay-insensitive traffic such as transactional data-bearing services is called type-2 for which $QoS_2 = 10^{-6}$. Two separate finite buffers of length $K_1 = K_2 = 100$ cells are provided and capacities of $C_1 = 0.8$ and $C_2 = 0.2$ are assumed.

The cell-generation process for a video coder is assumed to have two motion states: one is the low-motion state for interframe coding rate and the other is the high-motion state for intraframe coding rate [17].

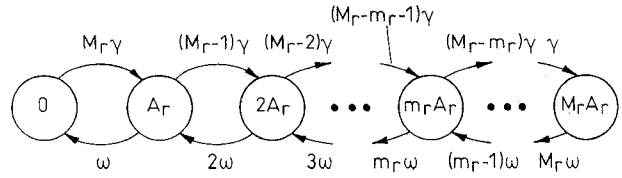

Fig. 4 *State-transition diagram for interframe coding $\lambda_r(t)$*

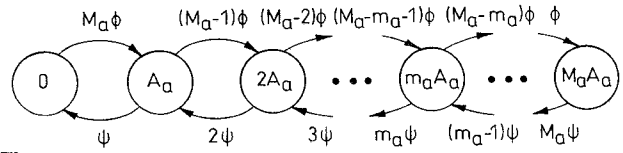

Fig. 5 *State-transition diagram for difference state $\lambda'_a(t)$*

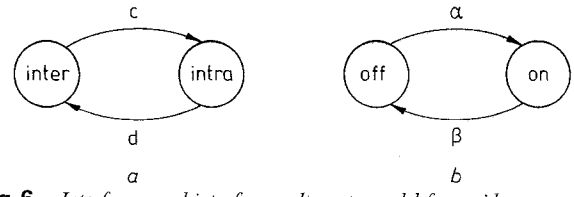

Fig. 6 *Interframe and intraframe alternate model for a video source and state-transition diagrams of a voice source*
a Interframe and intraframe alternate model
b State-transition diagrams of a voice source

The rate of intraframe coding is further divided into two parts: the first part has the same rate as the interframe coding and the second part, called difference coding, is the difference rate between intraframe coding and interframe coding. The interframe coding and the difference coding are all modelled as discrete-state Markov-modulated Bernoulli processes (MMBP) with basic rates A_r and A_a, as shown in Figs. 4–6. Let $\lambda_a(t)$, $\lambda_r(t)$ and $\lambda'_a(t)$ denote the respective cell-generation rates from the video coder for intraframe coding, interframe coding and difference coding at time t. Clearly, $\lambda_a(t) = \lambda_r(t) + \lambda'_a(t)$ The process of $\lambda_r(t)$ is an (M_r+1)-state birth–death Markov process. The state-transition diagram for $\lambda_r(t)$ uses the label m_rA_r to indicate the interframe-coding cell-generation rate of a state and uses the labels $(M_r - m_r)\gamma$ and $m_r\omega$ to denote the respective transition probabilities from state m_rA_r to state $(m_r + 1)A_r$, and from state m_rA_r to state

$(m_r - 1)A_r$. Similarly, the process for $\lambda'_a(t)$ is an $(M_a + 1)$-state birth–death Markov process. The state-transition diagram for $\lambda'_a(t)$ uses the label m_aA_a to indicate the additional cell-generation rate of a state due to intraframe coding and uses the labels $(M_a - m_a)\phi$ and $m_a\psi$ to denote the respective transition probabilities from state m_aA_a to state $(m_a + 1)A_a$, and from state m_aA_a to state $(m_a - 1)A_a$. The video source alternates between interframe and intraframe, depending on the video-source activity factor. There is a transition rate c in the interframe state and a transition rate d in the intraframe state. The values of γ, ω, M_r, A_r, ϕ, ψ, M_a, A_a, c and d can be obtained from the traffic variables R_p, R_m and T_p [3, 7].

The cell-generation process for a voice call is modelled by an interrupted Bernoulli process (IBP) [3, 4]. During the ON ('talkspurt') state, voice cells are generated at rate A_v; while during the OFF (silence) state, no cells are generated. A voice source has a transition rate of α in the OFF state, and a transition rate of β in the ON state, as shown in Fig. 6*b*.

As for the data source, there are high-bit-rate and low-bit-rate data services, and the generations of high-bit-rate data cells and low-bit-rate data cells are characterised by Bernoulli processes with rates of θ_1 and θ_2, respectively. Also, the holding times for video, voice, high-bit-rate data and low-bit-rate data are assumed to be exponentially distributed.

For a video-source-arrival process, it is assumed that $R_p = 3.31 \times 10^{-2}$, $R_m = 1.10 \times 10^{-2}$ and $T_p = 0.5$s, which give $M_r = M_a = 20$, $A_r = 1.34 \times 10^{-3}$, $A_a = 3.15 \times 10^{-4}$, $\gamma = 3.77 \times 10^{-6}$, $\omega = 5.65 \times 10^{-6}$, $\phi = \psi = 2.83 \times 10^{-5}$, $c = 5.65 \times 10^{-6}$ and $d = 5.09 \times 10^{-5}$. For a voice source arrival process, it is assumed that $R_p = 4.71 \times 10^{-4}$, $R_m = 2.12 \times 10^{-4}$ and $T_p = 1.35$s, which give $A_v = 4.71 \times 10^{-4}$, $\alpha = 1.71 \times 10^{-6}$ and $\beta = 2.09 \times 10^{-6}$. For a high-bit-rate data source, it is assumed that $R_p = 7.36 \times 10^{-2}$, $R_m = 7.36 \times 10^{-3}$ and $T_p = 3.14 \times 10^{-2}$s, which give $\theta_1 = 0.1$, and for a low-bit-rate data source, it is assumed that $R_p = 3.68 \times 10^{-2}$, $R_m = 7.36 \times 10^{-4}$ and $T_p = 2.88 \times 10^{-2}$s, which give $\theta_2 = 0.02$. The mean holding time is 60 min for a video service, 3 min for a voice service and 18 s for both high- and low-bit-rate data services. Without loss of generality, we may assume that the values of R_p and R_m have been normalised by the network capacity.

Two kinds of cell loss probability for type-i traffic are considered: source loss probability $p_{s,i}$ due to selective discarding on the customer side, and node loss probability $p_{n,i}$ due to blocking on the network side. Thus, the overall cell loss probability $p_{l,i}$ for type-i traffic is defined as

$$p_{l,i} = \kappa p_{s,i} + p_{n,i}, \quad i = 1, 2 \tag{11}$$

where κ is used to indicate the significance of the node loss over the source loss. $\kappa = 0.8$ is here assumed because selectively discarding cells at the source should have less effect on information retrieval than blocking cells at the node.

3.2 Simulation results and discussion

Fig. 7 shows the utilisation of an ATM-network element employing the NNCAC method proposed in this paper, the fuzzy-logic-based CAC scheme proposed in [13], Hiramatsu's neural-network CAC scheme proposed in [9] and the NFCAC scheme proposed in [14]. In the simulations, a three-layered neural network with 30 hidden nodes, and a backpropagation learning algorithm was used in both Hiramatsu's neural-network CAC and NNCAC. The activation function $G(x)$ was $1/(1 + e^{-x})$ in the hidden layer and $U(x)$ in the output layer.

It was found that the NNCAC method produced a processing gain in utilisation over Hiramatsu's neural-network CAC scheme by preprocessing of the input parameters. This was because the training error for Hiramatsu's neural-network CAC scheme cannot easily be reduced in such a complicated traffic environment. The technique of preprocessing was also employed by the fuzzy-logic-based CAC scheme and the NFCAC scheme to improve the controlled performance. However, the NNCAC method produced about a 10% improvement in utilisation over that of the fuzzy-logic-based CAC scheme by reducing the estimation error of the equivalent capacity method through self-learning. In comparison with the NFCAC scheme, both the NNCAC method and the NFCAC scheme provided similar system utilisations in the steady state because they both adapted their behaviour to the system by means of the learning capability of the neural network. However, the design of the NFCAC scheme requires much more knowledge than the design of the NNCAC method. The NNCAC method would be a proper choice for designers who are not familiar with fuzzy-logic-control schemes, or lack the requisite knowledge of CAC.

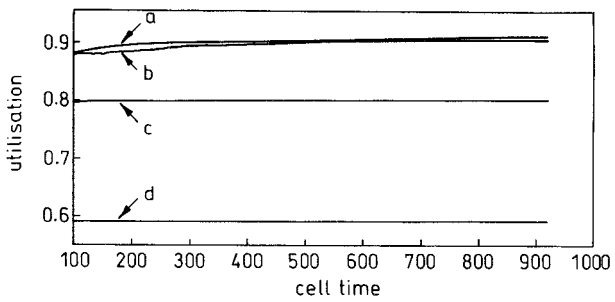

Fig.7 *System utilisation*
Unit = 10^6 cells
a NFCAC
b NNCAC
c Fuzzy-logic-based CAC
d Hiramatsu's CAC

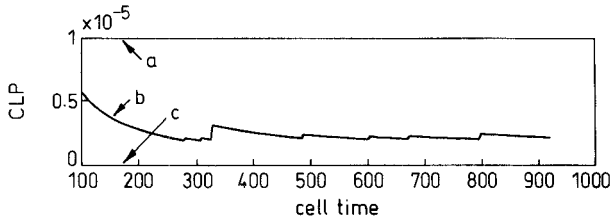

Fig.8 *Cell-loss probability (CLP) for type-1 traffic*
Unit = 10^6 cells
a QoS
b NFCAC
c NNCAC, fuzzy-logic-based CAC, Hiramatsu's CAC

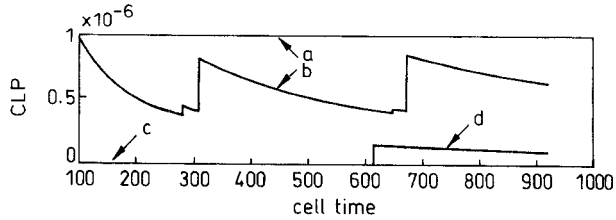

Fig.9 *Cell-loss probability (CLP) for type-2 traffic*
a QoS
b NFCAC
c Fuzzy-logic-based CAC, Hiramatsu's CAC
d NNCAC

Figs. 8 and 9 show the cell-loss probabilities of ATM networks employing different control schemes. The QoSs of the two types of traffic were all guaranteed for the NNCAC method, the fuzzy-logic-based CAC scheme, and Hiramatsu's neural-network CAC scheme. However, the NNCAC method maintained a larger margin between the QoS requirement and its cell loss probability than the NFCAC scheme.

4 Concluding remarks

This paper presents a neural-network connection-admission-control NNCAC method for ATM networks. The NNCAC method uses three preprocessed input parameters to simplify the training process and to improve the controlled performance. It supports a unified metric for input-traffic characterisation by utilising robust analytical results from the equivalent capacity method and then diminishes the estimation error of the equivalent capacity method by employing the learning capability of a neural network. Simulation results showed that our proposed NNCAC method provided a 20% system utilisation improvement over Hiramatsu's neural-network CAC scheme, and 10% system utilisation improvement over the fuzzy-logic-based CAC scheme [13], while maintaining QoS contracts. It was also found that the NNCAC method provided a utilisation comparable with that of the NFCAC scheme [14] and possessed a lower cell loss probability. The NNCAC method is suitable for designers who are not familiar with fuzzy-logic control schemes, or lack requisite knowledge of CAC.

5 Acknowledgment

This work was supported by the National Research Council, Taiwan, under contract NSC 85-2213-E009-075.

6 References

1 GUÈRIN, R., AHMADI, H., and NAGHSHINEH, M.: 'Equivalent capacity and its application to bandwidth allocation in high-speed networks', *IEEE J. Select. Areas Commun.*, 1991, **9**, (7), pp. 968–981
2 MURATA, M., OIE, Y., SUDA, T., and MIYAHARA, H.: 'Analysis of a discrete-time single-server queue with bursty input for traffic control in ATM networks'. IEEE GLOBECOM '89, 1989, pp. 1781–1787
3 BAE, J.J., and SUDA, T.: 'Survey of traffic control schemes and protocols in ATM networks', *Proc. IEEE*, 1991, pp. 170–189
4 YIN, N., LI, S.Q., and STERN, T.E.: 'Congestion control for packet voice by selective packet discarding', *IEEE Trans. Commun.*, 1990, **38**, pp. 674–683
5 ATAI, A.: 'A rate-based feedback traffic controller for ATM networks'. IEEE ICC'94, 1994, pp. 1605–1615
6 JONG, S.G., and HEE, K.: 'Congestion control with double threshold in ATM networks'. IEEE GLOBECOM '94, 1994, pp. 595–599
7 ROBERTS, J.W.: 'Variable-bit-rate traffic control in B-ISDN', *IEEE Commun. Mag.*, September 1991, **29**, pp. 50–56
8 BONDE, A.R., and GHOSH, S.: 'A comparative study of fuzzy versus "fixed" thresholds for robust queue management in cell-switching networks', *IEEE/ACM Trans. Netw.*, 1994, **2**, (4), pp. 337–344
9 HIRAMATSU, A.: 'ATM communications network control by neural networks', *IEEE Trans. Neural Networks*, 1990, **1**, (1), pp. 122–130
10 'Neuralcomputing in high-speed networks', *IEEE Commun. Mag.*, 1995, **33**, (10)
11 TRAN-GIA, P., and GROPP, O.: 'Performance of a neural net used as admission controller in ATM systems'. IEEE GLOBECOM '92, 1992, pp. 1303–1309
12 TARRAF, A.A., HABIB, I.W., and SAADAWI, T.N.: 'Congestion control mechanism for ATM networks using neural networks'. ICC '95, 1995, pp. 206–210
13 CHENG, R.G., and CHANG, C.J.: 'Design of a fuzzy traffic controller for ATM networks', *IEEE/ACM Trans. Netw.*, 1996, **4**, (3), pp. 460–469
14 CHENG, R.G., and CHANG, C.J.: 'A neural-net based fuzzy admission controller for an ATM network'. IEEE INFOCOM '96, 1996, pp. 777–784
15 COTTER, N.E.: 'The Stone-Weierstrass theorem and its application to neural nets', *IEEE Trans. Neural Netw.*, 1990, **1**, (4), pp. 290–295
16 RUMELHART, D.E., HINTON, G.E., and WILLIAMS, R.J.: 'Learning internal representation by error propagation' *in* 'Parallel distributed processing: explorations in the microstructure of cognition' (MIT Press, Cambridge, 1986), vol. 1, chap. 8
17 GALL, D.L.: 'MPEG: a video compression standard for multimedia applications', *Commun. ACM*, 1991, **34**, (4), pp. 46–58