FLOW CONTROL FOR ABR SERVICE IN ATM NETWORKS USING MODULAR NEURAL NETWORKS

Wee-Seng Soh & Chen-Khong Tham
Dept of Electrical Engineering,
National University of Singapore,
Singapore 119260.
E-mail: wssoh@ee.nus.sg *&* eletck@nus.sg

Most congestion control schemes only react to congestion after it has taken place, which may be too late in high-speed ATM networks. In this paper, we present an enhanced version of the 'intelligent congestion control scheme' by Siu & Tzeng [1] using modular and hierarchical neural networks for predicting the congestion status of an ATM switch. The fast learning and accurate predictions obtained with this architecture is shown to produce small average queue length while maintaining a high throughput. Experiments which compared the performance of this enhanced scheme with the original scheme show that the ability to predict congestion becomes increasingly important when link distances are longer.
Keywords: ATM Flow Control, Modular Neural Networks

1 Introduction

Asynchronous Transfer Mode (ATM) is a high-speed packet switching technology for the broadband integrated services digital network (B-ISDN), in which various kinds of communication services such as voice, video and data are transferred over high-speed links. This technology is gaining acceptance as the backbone high-speed network of the future. However, there are some open issues in the standardization of ATM technology, one of which is the problem of congestion control.

In ATM networks, there are two main classes of traffic: guaranteed and best-effort. Guaranteed services, such as Constant Bit Rate (CBR) and Variable Bit Rate (VBR), require explicit quality of service (QoS) guarantees, e.g. bounded cell delay and cell loss probability. These services will declare information such as the required QoS and their own traffic parameters at call setup time. For these services, admission control and bandwidth allocation are the main means of congestion control.

Another service category currently under development is Available Bit Rate (ABR), which supports data networking applications. Most of these applications have vague requirements for throughputs and delays, which are often expressed as ranges of acceptable values. Only best-effort service is provided for ABR. Congestion control for the ABR service is much more difficult to implement due to the bursty nature of data traffic.

Mathematical analysis usually provide useful results only in steady state situations and when restrictive assumptions are satisfied. However, in real applications, traffic conditions change frequently and the environment is non-stationary. Neural network approaches which do not require precise models of the network processes have been successfully used for traffic management and congestion control [2,3,4,5].

2 Existing Flow Control Schemes

The ATM Forum [6] has finally adopted the closed-loop rate-based mechanism as the flow control strategy after considerable debate. Rate-based schemes use feedback information from the network to specify the maximum rate at which each source can emit cells into the network on every Virtual Connection (VC). Specifically, a source sends forward Resource Management Cells (RM-cells) to the destination periodically, which then returns a backward RM-cell for every forward RM-cell received. These backward RM-cells carry feedback control information from the network to the source.

Many rate-based schemes have been proposed. Examples of such schemes are Forward Explicit Congestion Notification (FECN) [7], Backward Explicit Congestion Notification (BECN) [8] and Proportional Rate Control Algorithm (PRCA) [9]. Unfortunately, most schemes suffer from problems due to VC starvation. A VC which passes through a greater number of congested links will tend to have lower allowed cell rate (ACR) than others, and the VC is said to be 'beaten down'. This results in extreme unfairness. In the next section, we present a more robust technique developed by Siu & Tzeng [1], which resolves the undesirable features of existing rate-based schemes. Our neural network-based technique will be a direct enhancement of this technique.

3 Intelligent Congestion Control Technique

The key idea of this scheme is that each congested switch will estimate the 'optimal' cell rate on each VC using a first-order filter. The advantage of this scheme is that it does not require per-VC queuing or accounting, yet it can address the 'beat down' problem found in the other rate-based schemes.

Specifically, a source in each VC will periodically send an explicit RM-cell containing its current ACR to the destination. Using this information, an intermediate switch can then use a simple first-order filter to iteratively estimate the 'optimal' cell rate for each VC. The minimum estimated rate among all rates computed at the switches, which is also known as the Explicit Rate (ER), will then be conveyed to the source via a backward RM-cell. The

source, on the other hand, will decrease its rate on every cell transmitted until it receives a backward RM-cell from the destination. It will then modify its ACR accordingly using the ER.

Figure 1 shows the outline of the intelligent congestion control technique proposed by Siu & Tzeng - the detailed pseudocode is not reproduced here due to space constraint. Note that each queue of a switch has a variable $MACR$ (Modified ACR), which contains the estimated 'optimal' cell rate value.

1. For every N data cells transmitted from a source, an RM(ACR,ER) cell is sent, where ER is initialised to be the maximum ACR of the source.

2. After transmitting each cell, a source decreases its ACR by a fraction (1/Multiplicative Decrease Factor (MD)).

3. Upon receiving each RM(ACR,ER) cell, a source increases its ACR by a certain fixed rate. However, the new value cannot exceed ER.

4. A destination returns a backward RM(ACR,ER) cell upon receiving each forward RM(ACR,ER) cell from the source.

5. When a non-congested switch receives an RM(ACR,ER) cell from the source,
$$MACR \leftarrow MACR + \beta \times (ACR - MACR).$$

6. When a congested switch receives an RM(ACR,ER) cell from the source and if $(ACR < MACR)$,
$$MACR \leftarrow MACR + \beta \times (ACR - MACR).$$

7. Suppose a congested switch receives an RM(ACR,ER) cell from the destination. If the current queue length exceeds the fast-down queue threshold (DQT), i.e. queue is very congested, it replaces ER in the RM-cell by $\min(ER, \gamma \times MACR)$, where γ is a fast reduction ratio; otherwise, if $(ACR > MACR)$, it replaces ER in the RM(ACR,ER) cell by $\min(ER, MACR)$.

Figure 1: Outline of intelligent congestion control technique proposed by Siu & Tzeng.

For the above technique, a switch is considered to be congested when the

queue length of the switch increases over a period of time. However, in high-speed networks, it may be too late to take a corrective action after congestion has already occurred. Queue lengths may build up rapidly and exceed the buffer size, resulting in undesirable cell loss. We propose a neural network-based approach as an enhancement of the intelligent congestion control technique. By making use of the prediction capability of neural networks, our proposed technique is able to predict a congestion before it actually occurs, and take the necessary corrective actions.

4 Neural Networks

Neural networks, which are trained using the *supervised learning* paradigm, are useful for acquiring a non-linear mapping between an input pattern and target values, especially when an unknown relationship exists between them. Learning is done by presenting a set of examples consisting of input-output pairs of values to the neural network and subsequently modifying its internal parameters. A neural network can also generalise from the set of examples with which it was trained and produce reasonable output values even when an input pattern it has not seen before is presented.

In the area of high-speed networking, neural networks have been applied in Connection Admission Control (CAC) [2], flow control and routing [4], ATM switch control [10] and bandwidth prediction for variable bit rate video. The most commonly used neural network architecture is the Multi-Layer Perceptron (MLP) network trained using the error back-propagation algorithm [11].

The basic approach for prediction using neural networks is to construct a mapping between current and future values. For the case of flow control, the neural network is trained by presenting sets of an input pattern which reflects the buffer status - this can consist of current and previous cell arrival rates (CAR) as well as the change in buffer queue length - and a target value which is the observed queue length at some time in the future. To predict congestion, an input pattern which reflects the current buffer status is presented. The required output will then be an estimate of the future queue length.

5 Modular and Hierarchical Neural Networks

Although single or monolithic function approximators, such as MLP networks, are able to model most functions, an alternative, especially when complex functions need to be modelled, is to use multiple models for regression. This method involves the construction of several different predictors and then combining them to give improved prediction accuracy. In the rest of this section,

we present a modular and hierarchical neural network architecture which can meet both objectives of high prediction accuracy and fast learning.

5.1 Hierarchical Mixtures of Experts (HME)

The Hierarchical Mixtures of Experts (HME) [12] is an architecture which is based on the principle of *divide-and-conquer* in which a large and difficult function approximation problem is broken into several smaller and easier-to-solve sub-problems. This involves dividing the input space into a nested set of regions and fitting surfaces to the data that fall in these regions. A separate function-approximator or 'expert' network is assigned through competition to each of these regions. Each expert network may itself be composed of competing sub-experts.

When the HME was used in time series prediction tasks[13,14], results which were better than leading methods, including those using MLP networks[15], can be obtained. In this paper, the HME is used for predicting the buffer queue length based on past arrival patterns. These predictions form the basis of the proposed congestion controller.

5.2 Description of the HME architecture

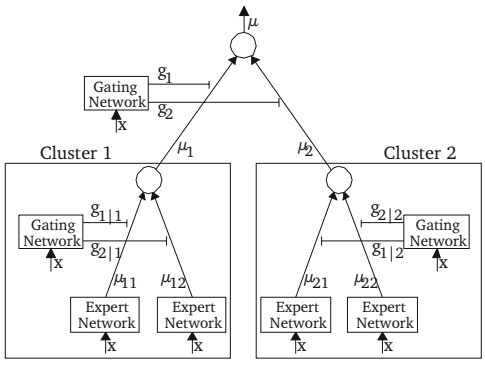


Figure 2: A two-level Hierarchical Mixtures of Experts architecture.

A two-level HME architecture, which consists of expert networks and gating networks, is shown in Figure 2. The mathematical framework presented in this section is based on this two-level tree-like architecture - it is straightforward

to extend it to arbitrarily deep structures. Only the main features will be described here (see Jordan & Jacobs [12] for details).

The case in which the architecture as a whole is performing regression is considered here. Within the architecture, the expert networks perform regression in regions of input space, whereas the gating networks essentially perform classification over the entire input space since their task is to select the appropriate expert network in order to produce the output value. The observed input-output pairs used for training the HME can be denoted as $(\mathbf{x}, \mathbf{y})$.

The various parts of the architecture will now be described. Expert network (i, j) produces output $\boldsymbol{\mu}_{ij}$ as a linear function of the input $\mathbf{x}$, i.e. $\boldsymbol{\mu}_{ij} = \mathbf{U}_{ij}\mathbf{x}$, where $\mathbf{U}_{ij}$ is a weight matrix of dimensionality $n \times m$ in the case where there are m elements in the input vector and n elements in the output vector.

The top-level gating network produces intermediate variables ξ_i, also as linear functions of the input vector: $\xi_i = \mathbf{v}_i^{\mathrm{T}}\mathbf{x}$, where $\mathbf{v}_i$ is a weight vector. Subsequently, the i^{th} output of the gating network is produced by the 'softmax' activation function $g_i = \frac{e^{\xi_i}}{\sum_k e^{\xi_k}}$, where the g_i values are positive and sum to one for each $\mathbf{x}$. This is the mechanism which enables gating networks to perform classification. Likewise, the outputs of the lower-level gating networks are defined as follows: $\xi_{ij} = \mathbf{v}_{ij}^{\mathrm{T}}\mathbf{x}$, and $g_{j|i} = \frac{e^{\xi_{ij}}}{\sum_l e^{\xi_{il}}}$, where $g_{j|i}$ is the j^{th} output of the i^{th} gating network at the second level of the architecture.

The output vector at each non-terminal node of the tree is the weighted sum of the outputs of expert networks below that node, which for the second level is $\boldsymbol{\mu}_i = \sum_j g_{j|i}\boldsymbol{\mu}_{ij}$. Finally, the output at the top level of the tree is $\boldsymbol{\mu} = \sum_i g_i\boldsymbol{\mu}_i$. Since the values of g and $\boldsymbol{\mu}$ depend on the input vector $\mathbf{x}$, the final output value of the architecture is a non-linear function of $\mathbf{x}$.

5.3 Training the HME: a Maximum Likelihood estimation problem

The HME architecture can also be viewed as a statistical model of the observed data, which is assumed to be generated by the environment through a mixture of regressive processes. The HME training task becomes one of identifying the parameters of the expert networks which function as regressive processes $P_{ij}(\mathbf{y})$, and gating networks which determine the mixture coefficients g_i and $g_{j|i}$. The posterior probabilities h_i and $h_{j|i}$ that each branch of the architecture generated the target vector $\mathbf{y}$ can also be evaluated using Bayes' formula [12].

The basic approach to training the HME is as follows: first, a likelihood function which gives the likelihood that the HME architecture generated the observed target value is formulated; second, the parameters of the HME architecture are adapted so as to increase or maximise this likelihood measure, i.e

the training task is essentially to solve a maximum likelihood (ML) estimation problem.

5.4 Online training with Recursive Least Squares (RLS)

From this point onwards, training the HME becomes an optimization problem. In this paper, a second order method based on Newton-Raphson, known as the Recursive Least Squares (RLS) algorithm, is used.

For the expert networks, the online update rule is given by the following equation

$$\mathbf{U}_{ij} \leftarrow \mathbf{U}_{ij} + h_{ij}(\mathbf{y} - \boldsymbol{\mu}_{ij})\mathbf{x}^{\mathrm{T}}\mathbf{R}_{ij} \tag{1}$$

where $\mathbf{R}_{ij}$ is the inverse covariance matrix for expert network (i, j), and $h_{ij} = h_i h_{j|i}$. This matrix is updated recursively according to

$$\mathbf{R}_{ij} \leftarrow \lambda^{-1}\left[\mathbf{R}_{ij} - \frac{\mathbf{R}_{ij}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{R}_{ij}}{\lambda[h_{ij}]^{-1} + \mathbf{x}^{\mathrm{T}}\mathbf{R}_{ij}\mathbf{x}}\right] \tag{2}$$

where λ (set to 0.99) is a *forgetting factor* to enable the networks to track time-varying parameters since the posterior probabilities change over time. The update equations for the top and lower-level gating networks have a similar form with $\ln h_i$ and $\ln h_{j|i}$ respectively replacing $\mathbf{y}$ as the target value to be learnt, and the weighting terms 1 and h_i respectively replacing h_{ij}.

6 HME-based Intelligent Congestion Control

The HME architecture is integrated into Steps 5, 6 and 7 of the intelligent congestion control technique described in Figure 1. Each output queue of a switch has a HME-based controller. Its role is to predict the buffer queue length at the next sampling instant if no corrective action for congestion is taken in the current sampling instant, using an input pattern made up of the current and previous values of the cell arrival rate at the ATM switch. Learning takes place in real-time: the output of the HME is compared with a threshold Q_t to determine the predicted congestion status; this decision affects the subsequent input patterns and target values, which in turn affect future HME predictions and congestion decisions. However, note that the HME is trained only after a 'non-congested' prediction is made to ensure that a consistent mapping is formed. This is because a forecasted congestion will cause the sources to reduce their respective ACRs, causing them to become smaller than otherwise. Figure 3 shows the algorithm for congestion prediction using the HME.

Once the predicted queue length $J(t)$ at the next sampling instant is available, it is compared with the threshold queue length Q_t as well as the fast-down

1. Start of control cycle. Set $t = 0$.

2. Get cell arrival rate $p(t)$ from multiplexer
   (number of cells arrived in the previous time interval Δt).

3. Get buffer queue length $l(t)$ from multiplexer
   (obtained by sampling the buffer queue length at time t).

4. If $t \geq k - 1$, form new cell arrival pattern $P(t)$:
   $$P(t) = [p(t), p(t-1), \ldots, p(t-k+1)]$$

5. Pass $P(t)$ to HME and ask for $J(t)$, which is the prediction of buffer queue length at the next sampling instant.

6. Determine predicted congestion status: (Q_t is the threshold queue length)
   if $J(t) > Q_t$, set $c(t)$ to 1, i.e. *congested* = *TRUE*
   if $J(t) > DQT$, *very congested* = *TRUE*
   else *very congested* = *FALSE*
   else set $c(t) = 0$, i.e. *congested* = *FALSE*

7. Save $P(t)$ (only the previous and current $P(t)$ values need to be saved).

8. If $t \geq k$ AND $c(t-1) = 0$, i.e. *congested* = *FALSE* at $t-1$, train HME with the following input-output pair:
   input pattern: $P(t-1)$ target value: $y(t) = l(t)$

9. Repeat the cycle: set $t \leftarrow t + 1$ and go to Step 2.

Figure 3: Algorithm for congestion prediction using the HME. The various parameter values can be found in Section 7.3. *Note*: t refers to the number of sampling intervals Δt.

queue threshold DQT. Congestion is predicted if $J(t)$ exceeds Q_t, and extreme congestion is predicted if $J(t)$ exceeds DQT (note that $DQT > Q_t$). For the latter case, the estimated 'optimal' rate will be reduced drastically.

A three-level HME with eight expert networks at the bottom level was used. The learning algorithms presented in Section 5.4 are used with the appropriate extensions to the three-level case.

We have made use of 'pattern tables', which store previously observed training patterns, similar to what Hiramatsu [2] has proposed for the case of

neural network-based Connection Admission Control (CAC) schemes. Two pattern tables are used, one for congested events, i.e. queue length is larger than threshold queue length Q_t, and another for non-congested events. Patterns observed at each interval Δt are inserted into the appropriate table, and when either table is full, the oldest pattern is replaced with the current observed pattern.

In our proposed congestion control scheme, the current input-output pattern is always used for training the HME in each interval. After that, during the same interval, training is repeated using M number of patterns selected from the pattern tables. This ensures that the neural network utilises new information as soon as it becomes available, while not 'forgetting' older patterns. Consequently, the controller is able to react accordingly to a sudden change in the traffic situation. For each of the M patterns, the congested-event pattern table is selected with a fixed probability P_t, and the non-congested-event table with probability $(1-P_t)$. A training pattern is then chosen randomly from the selected table and used for training the HME. This permits the neural network to be trained with different input patterns selected from the input domain in a random order and can help it to achieve faster and more accurate learning (Haykin[16]).

7 Experimental Details

7.1 ATM network simulator

The NIST ATM Network Simulator[17] provides an interactive ATM network modelling environment with a graphical user interface. It enables the user to: (1) create different network topologies, components and applications; (2) measure simulated network activity at different points in the network; and (3) modify component and application parameters to evaluate their effects. The core of the simulator is a discrete event simulation kernel. We added additional modules to enable it to exchange data with neural network processes using UNIX shared memory, as well as to implement the traffic source and congestion controller described in this paper.

7.2 Network topology used

The network topology used in the experiments is a metropolitan area network (MAN) as shown in Figure 4. Two sets of link distances are used (see Table 1). The first set is the same as those used by Siu & Tzeng[1]. The second set, shown in brackets, is used to highlight the advantages of prediction capability.

Table 1: Link distances for the MAN shown in Figure 4

Link	Distance	Link	Distance
L1A, L2A	50 km (100 km)	L4A, L4B	5 km (70 km)
L2A, L2B	25 km (90 km)	L5A, L5B	1 km (60 km)
L3A, L3B	10 km (80 km)	BB	50 km (100 km)

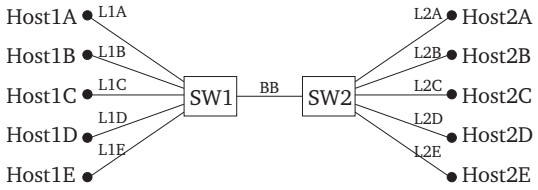


Figure 4: MAN used for the experiments.

7.3 Traffic sources and parameters used

In the simulation, all sources are assumed to be persistent sources, i.e. they will attempt to transmit cells at their highest possible speed (ACR). This is to better illustrate the fairness of the scheme [1]. The various parameters used in the simulation are shown in Table 2.

7.4 Experiments

For each set of link distances shown in Table 1, two experiments were performed:

1. the original intelligent congestion control scheme proposed by Siu & Tzeng as described in Section 3 (**S&Z**); and
2. the HME-based scheme we proposed, as described in Section 6 (**HME**).

The results which we are interested in are: average queue length, throughput (i.e. total number of cells received at the destinations), and link utilisation. These are discussed in the next section.

8 Results

Figure 5(a) shows the plot of buffer queue length and link utilisation vs time for S&Z1. As we can see, it is effective in maintaining the queue length well below 20 while having almost full link utilisation. Figure 5(b) shows the corresponding results for HME1. The largest peak at the initial stage occurs

Table 2: Simulation parameters

Parameter	Values
Sampling interval Δt	100μs
Buffer size of switch multiplexer	100
Size of pattern table	1000
Pattern table selection probability P_t	0.5
Size of input pattern k	10
Threshold queue length Q_t	5
Number of patterns selected from pattern table M	5
Fast-down queue threshold DQT	50
Link Speed	155 Mbps
Initial cell rate ICR	7.75 Mbps
Multiplicative Decrease Factor MD	256
Fast reduction ratio γ	1/4
Peak cell rate PCR	155 Mbps
Additive increase rate AIR	0.053 Mbps
Minimum cell rate MCR	0.155 Mbps
Number of data cells between RM-cells N	31
First-order filter parameter β	1/16

because the HME has not learnt the traffic pattern yet. It can be seen that the HME-based controller is able to keep the queue length well below 10 after it has learnt the traffic pattern. By comparing the average queue length for the above two cases in Table 3, we see that the HME-based controller is nearly twice as effective in terms of keeping queue length small. However, it has a slightly smaller throughput in this case. By careful observation of Figure 5(b), we see that there is a slight dip in link utilisation between $t = 0.02$s to $t = 0.03$s. This corresponds to the stage where the HME has not fully learnt the traffic pattern, causing it to predict congestion excessively, thus accounting for the lower throughput.

The advantages of the proposed HME-based controller are better understood when the distances between the sources and the ATM switch are increased, as in the second set of link distances used. Figure 5(c) shows the plots for S&Z2. Because of increased distance between the sources and the switch, backward RM-cells take longer time to convey the estimated 'optimal' cell rates to the sources. In this case, the disadvantage of having no prediction capability becomes obvious. As we can see, the buffer queue length exceeds the fast-down queue threshold DQT quite frequently, causing the ACRs to be

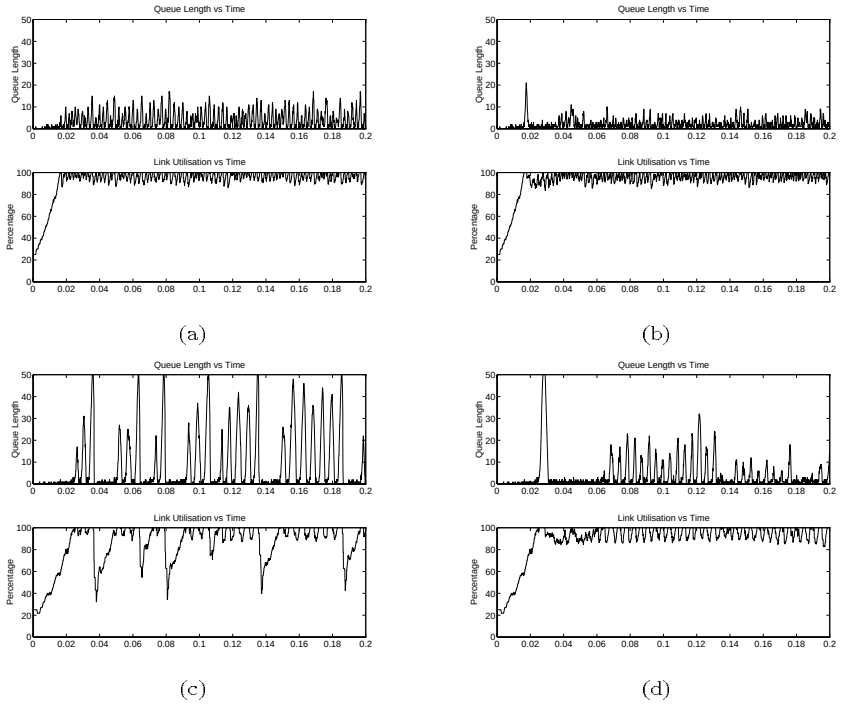


Figure 5: The buffer queue length of multiplexer and the link utilisation throughout the simulation period of 200ms for the following experiments: (a) Link distance set 1 without HME (S&Z1). (b) Link distance set 1 with HME (HME1). (c) Link distance set 2 without HME (S&Z2). (d) Link distance set 2 with HME (HME2).

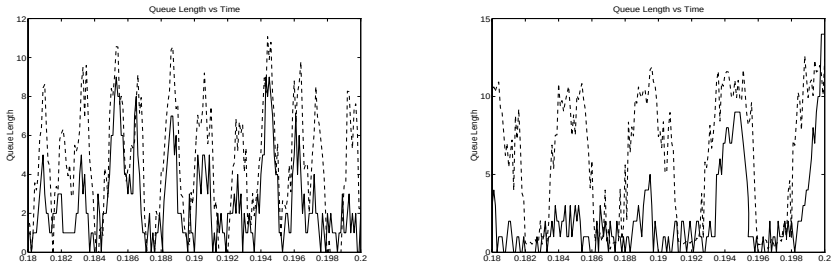
slashed by large proportions. Link utilisation is affected drastically at such instances.

Figure 5(d) shows the plots obtained for HME2. Similar to Figure 5(b), there is a large peak in queue length at approximately $t = 0.03$s as the HME has not learnt the traffic pattern yet. Between $t = 0.03$s to 0.06s, there is a distinct period with extremely low queue length and reduced link utilisation. However, the HME improved rapidly, causing the link utilisation to be maintained at near optimum level after this transient period. Throughout the remaining simulation period, we see that queue length is well below the fast-down queue threshold DQT. Also, average queue length from $t = 0.14$s to 0.2s is apparently lower than that from $t = 0.06$s to 0.14s while maintaining the same level of link utilisation, meaning that the HME has become more specialised. From

Table 3: Throughput and average queue length at the end of 200ms of simulation time.

Congestion Control Scheme	Total number of cells received at destination	Average queue length
Link set 1 without HME (S&Z1)	67767	3.9103
Link set 1 with HME (HME1)	67017	2.1033
Link set 2 without HME (S&Z2)	60481	8.8762
Link set 2 with HME (HME2)	63465	3.9153

Table 3, it is observed that the HME-based controller has an average queue length less than half that of the S&Z scheme. Moreover, its throughput is approximately 5% higher. Note that in all cases, there are no cell loss.



(a) (b)

Figure 6: Predicted and actual queue length vs time for $t = 0.18$s to 0.2s for the following experiments: (a) Link distance set 1 with HME (HME1). (b) Link distance set 2 with HME (HME2). (Note: the dashed lines are the predicted values, while the solid lines are the actual values).

Figure 6 shows the predicted and actual queue length vs time for both sets of link distances with HME-based controller. The time period $t = 0.18$s to 0.2s is specially chosen because it corresponds to the period when HME has fully learnt the traffic patterns. From both plots, it can be seen that the predicted queue length wraps the peaks of the actual queue length most of the time, meaning that the HME is able to foresee a sudden increase in queue length quite well. It is interesting to note that both predicted queue length and actual queue length have similar trends. Because a predicted congestion would have reduced the sources' ACRs, the actual queue length is lower than the predicted queue length most of the time.

9 Discussions

For the case of HME-based controllers, we see that queue length increases to a large peak at the initial stage, followed by a transient period of less than optimum link utilisation. This corresponds to the period before HME has learnt, because we have used random initial weights for the HME. In actual applications, the combination of both on-line and off-line training may be necessary. The initial weights of HME should be set to near optimal values, which could be obtained by off-line training using simple simulation models. Another way to reduce this transient effect is to force the controller to look into actual queue length during the initial stage, i.e. if actual buffer queue length is greater than the threshold value Q_t, actions for congestion have to be taken even if the HME s output indicates non-congestion.

10 Conclusion

In this paper, we have proposed a HME-based intelligent congestion control scheme, which is a direct enhancement of the scheme proposed by Siu & Tzeng[1]. It is shown to be superior in keeping queue length low, while maintaining high throughput in an ATM network. For the case where distances between sources and the switch are large, the ability to predict congestion becomes increasingly important, as it would be too late to take corrective actions after congestion has already occurred. The introduction of a HME-based controller into an ATM switch increases the complexity of the switch, but it may be worthwhile, considering the benefits it can bring. The experiments described in this paper have assumed persistent sources, i.e. all sources will attempt to transmit cells at their highest possible speed [1]. In practice, data traffic is highly bursty. Our future work will investigate the performance of the HME-based scheme with such traffic.

References

1. K.Y. Siu and H.Y. Tzeng. Intelligent congestion control for ABR service in ATM networks. *ACM SIGCOMM'95 Computer Communication Review*, pp. 81-106, 1995.
2. A. Hiramatsu. ATM communications network control by neural networks. *IEEE Transactions on Neural Networks*, 1(1):122-130, 1990.
3. A. Hiramatsu. Training techniques for neural network applications in ATM. *IEEE Communications Magazine*, pp. 58-67, Oct 1995.

4. J.E. Neves, M.J. Leitão, and L.B. Almeida. Neural networks in B-ISDN flow control: ATM traffic prediction or network modeling? *IEEE Communications Magazine*, pages 50-56, Oct 1995.
5. A.A. Tarraf, I.W. Habib, and T.N. Saadawi. Intelligent traffic control for ATM broadband networks. *IEEE Communications Magazine*, pp. 76-82, Oct. 1992.
6. ATM Forum, Traffic Management Specification Version 4.0, AF-TM-0056.000, April 1996.
7. B. A. Makrucki. Explicit Forward Congestion Notification in ATM networks. *Proceedings of TriComm'92*, Feb 1992.
8. P. Newman. Backward Explicit Congestion Notification for ATM Local Area Networks. In Proc. *IEEE GLOBECOM'93*, pp. 719-723, December 1993.
9. L. Roberts, B. Makrucki, T. Tofigh, A. W. Barnhart, B. Holden, G. Fedorkow, J. Daigle, M. Hluchyi, H. Suzuki, G. Ramamurthy, P. Newman, N. Giroux, R. Kositpaiboon, S. Sathe, G. Garg, and N. Yin. Closed-Loop Rate-Based Traffic Management. *ATM Forum Contribution 94-0438R1*, July 1994.
10. T.X. Brown. Neural networks for switching. In B. Yuhas and N. Ansari, editors, *Neural Networks in Telecommunications*, pp. 11-36. Kluwer, 1994.
11. D. E. Rumelhart, G. E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, ch. 8, pp. 318-362. Bradford Books/MIT Press, Cambridge, MA, 1986.
12. M.I. Jordan and R.A. Jacobs. Hierarchical Mixtures of Experts and the EM algorithm. *Neural Computation*, 6:181-214, 1994.
13. S.R. Waterhouse and A.J. Robinson. Non-linear prediction of acoustic vectors using Hierarchical Mixtures of Experts. In *Proceedings of Neural Information Processing Systems (NIPS) 7*, 1995.
14. W.S. Soh. Time-series Prediction using Modular Neural Networks. Technical Report, National University of Singapore, Singapore, May 1996.
15. A.S. Weigend, B.A. Huberman, and D.E. Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1(3):193-209, 1990.
16. S. Haykin. *Neural Networks - A Comprehensive Foundation*, Chapter 6. Macmillan, NJ, USA, 1994.
17. N. Golmie, A. Koenig, and D. Su. The NIST ATM Network Simulator. Technical Report NISTIR 5703, National Institute of Standards and Technology, USA, Aug 1995.