A neural network approach to effective bandwidth characterization in ATM networks

Srinivasan Ramaswamy
Newbridge Networks Corporation
600 March Road, Kanata, Ont., Canada K2K 2E6, tel: (613) 591-3600, fax: (613) 599-3673, e-mail: srini@ca.newbridge.com

Pawel Gburzynski
Department of Computing Science, University of Alberta
615 GSB, Edmonton, AB, Canada T6G 2H1, tel: (403) 492-2347, fax: (403) 492-1071, e-mail: pawel@cs.ualberta.ca

Abstract
Many call admission control schemes for ATM-type networks focus on the cell loss rate as the exclusive QoS metric and therefore base their *Eb* (*Effective bandwidth*) schemes on cell-loss rate approximations. We use simulation data to train an adaptive logic network (ALN) to estimate cell loss *and* delay; these estimates can then be used to compute effective bandwidths to satisfy both cell loss and delay. Results indicate that the ALN model is simple, computationally efficient, and sufficiently accurate for practical use.

Keywords
Neural networks, ATM networks, call admission, bandwidth characterization

1 INTRODUCTION

Many call admission schemes in ATM networks are based on the concept of *effective bandwidth*. The effective bandwidth of N aggregated sources is generally less than N times the effective bandwidth of a single source (this phenomenon is known as *statistical multiplexing gain*). Most of the effective bandwidth research has focused on the cell loss rate (*Clr*). Anick, Mitra and Sondhi (1982) present a well-known *fluid-flow* model (AMS) for the case of N identical On/Off sources and an infinite buffer. The cell loss rate for a buffer of size S is approximated by the probability that the occupancy of the infinite buffer exceeds S. Guerin, Ahmadi and Naghsineh (1991) propose a simplified version of the AMS model. A simple *binomial* approach is described by Murase, Suzuki, Sato and Takeuchi (1991): the authors' proposal is to approximate the cell loss rate by the probability that the combined cell rate of all N sources exceeds the link capacity. Rege (1994), Sykas, Vlakos, Tsoukatos and Protonotarios (1993) compare these and other methods. One limitation of them is the low accuracy for high loss probabilities (e.g., $> 10^{-6}$) and/or low buffer sizes. Another limitation is the lack of delay prediction, which in some applications is not less critical than the loss prediction.

We consider using an adaptive logic network (ALN) for estimating cell loss and delay. The results are compared with the results produced by two other approaches: multivariate non-linear regression (REG) and the AMS model. We have extended the latter to predict the delay as well; we refer to this as the D-AMS model. The ALN model is shown to predict *Eb* fairly accurately for a wide range of buffer size (1–20 times the burst length), cell loss (0.1–10^{-6}) and delay (1–5000 cells). This is the non-linear region of interest since the analytical methods (Anick et al. 1982, Guerin et al. 1991) are inaccurate at high cell loss.

2 EXPERIMENTAL SETUP

2.1 Models

The traffic sources requesting admission to a queue are likely to have similar peak rates.* Therefore, to simplify the problem at hand, we limit ourselves to calls with identical traffic descriptors (homogeneous QoS requirements are common to all *Eb* schemes mentioned above).

Each traffic source has an *On* and an *Off* state. When in the *On* state, the source generates cells at a deterministic rate of Pcr cells/s; no cells are generated during the *Off* period. The *On* and *Off* periods are exponentially distributed with means t_{ON} and t_{OFF}, respectively. The N sources are independent, but have identical mean *On* and *Off* periods. The mean burst length, *Bl*, is given by $Pcr \times t_{ON}$, and the average-to-peak-rate ratio, Av/Pk, is given by $\frac{t_{ON}}{t_{ON}+t_{OFF}}$.

2.2 Factors affecting delay and cell loss

The average delay *Del* and cell loss *Clr* are influenced by the number of calls N multiplexed at the queue, the characteristics of a single call (i.e., mean burst length *Bl* and average-to-peak-rate ratio Av/Pk), and the system

*When it is required to multiplex sources with widely differing peak rates and/or QoS requirements, round-robin scheduling among multiple queues is generally preferred.

parameters (i.e., buffer size S and service rate Bw). Decina and Toniatti (1990) mention that the effective bandwidth for sources whose peak rate is greater than roughly one-tenth of the link bandwidth is influenced by the burst length. The peak rate for the sources under consideration is assumed to be at least 10 times smaller than the link rate. This has the advantage of eliminating the need to consider Bl while obtaining an inverted model for Eb. The peak rate only affects the scale of operation without changing the relative effect of the other parameters. Eb is the service rate Bw normalized w.r.t. Pcr and N. Its value lies between Av/Pk and 1.

The reason why we choose Bl and Av/Pk as the parameters representing the characteristics of a single call (rather than using t_{ON} and t_{OFF} directly) is the following. Suppose that we obtain via a simulation study a formula for Eb in terms of Clr, t_{ON}, t_{OFF}, S and N. If we scale the peak rate used in the study by a factor of 10 – to find the service rate for another call type with the same characteristics – we will have to scale t_{ON} and t_{OFF} as well. Our formula will no longer apply since it was obtained using a call with different t_{ON} and t_{OFF}. On the other hand, if we choose Bl and Av/Pk as the independent variables, these will remain the same after the peak rate is scaled. Hence we can use the formula exactly as it is.

We chose three of the five factors for a full-factorial study – S, Eb, N. The traffic generated by a single source can be viewed as a video session (Hyman, Lazar and Pacifici 1991), with $t_{ON} = 25$ms, $t_{OFF} = 35$ ms and $Pcr = 14\,150$ cells/s. This results in $Bl = 353$ and $Av/Pk = 0.417$.

2.3 Training set

The levels chosen for the three factors mentioned before are shown in table 1. When the number of calls is large, linear approximations are expected to be accurate (figure 1 left) and hence we do not consider more than 25 calls. The same observation applies when the buffer size is much larger than Bl (figure 1 right) and so we only consider a maximum buffer size of 5000. Since Bl is 353 cells, the maximum value of S/Bl is 22. As we were mainly interested in predicting Eb – its value lies in $[Av/Pk, 1]$ – we used a fine granularity of 0.02. We found that when Eb was close to its minimum value of 0.417 (corresponding to Av/Pk), the steady-state values of Clr and Del were quite difficult to obtain accurately, even with very long simulations. Therefore, we opted to use a minimum Eb of 0.45.

150 million cells were simulated in a single experiment. A total of $10 \times 7 \times 19 = 1330$ experiments were run using SMURPH (Gburzynski 1996). The experiments were repeated with a different seed for the random number generator to improve the reliability of the delay and cell loss estimates.

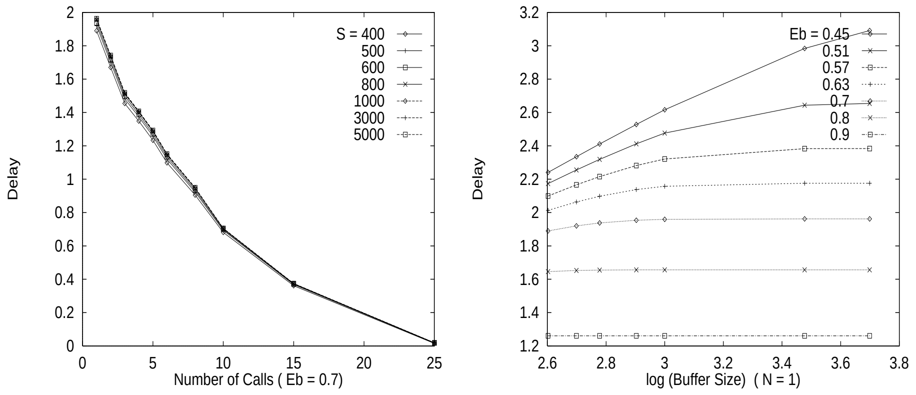

Figure 1 Delay vs. N (left), delay vs. S (right)

2.4 Test set

The levels chosen for the three factors above are shown in table 2. The values of Bl and Av/Pk were the same as in the training set. The peak cell rate (Pcr) was 1000 cells/s – to verify that the ALN and regression models correctly predict delay and cell loss for peak rates different from that used in the training set. The mean on and off time was suitably altered to maintain Bl and Av/Pk at their values in the training set. The levels for the factors have been chosen to test the models on both interpolation and extrapolation. As in the case of the training set, a 150 million cells were simulated in each experiment and the experiments were repeated to obtain better estimates.

Figure 2 (left) shows the variation of Del with Eb for varying N and $S =$

Table 1 Levels of factors for training set

Factor	*Levels*	*Number of levels*
N. of calls	1–5, 6, 8, 10, 15, 25	10
Buffer size	400, 500, 600, 800, 1000, 3000, 5000	7
Service rate	0.45–0.67 in steps of 0.02, 0.67–0.9 in steps of 0.03	19

Table 2 Levels of factors for test set

Factor	*Levels*	*Number of levels*
N. of calls	1, 2, 3, 7, 9, 12, 20, 30	8
Buffer size	370, 550, 900, 2000, 4000, 10000,	6
Service rate	.44, .48, .52, .56, .6, .65, .7, .75, .8, .85, .9, .95	12

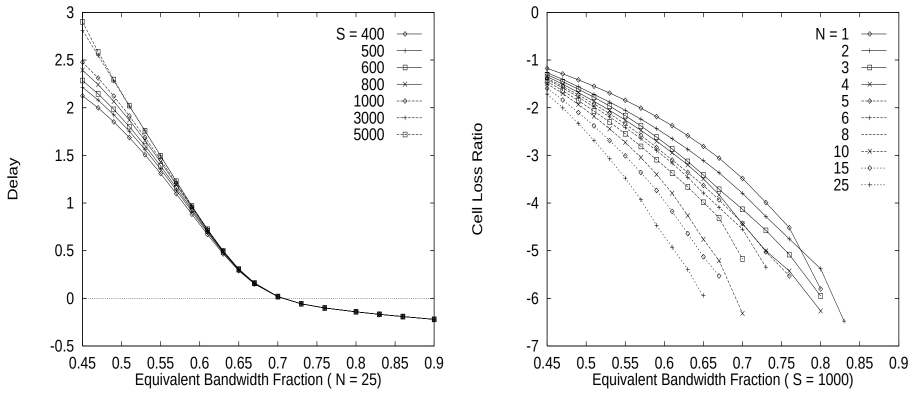

Figure 2 Delay vs. *Eb* (left), *Clr* vs. *Eb* (right)

1000, while figure 2 (right) shows *Clr* as a function of *Eb*. The qualitative information in these figures was used in deriving the regression and ALN models for delay.

2.5 Regression

In this section, we present an equation relating the delay and *Clr* to the three factors mentioned in section 2.2. We use for this the SPSS tool-box from the MATLAB package (see SPSS (1992)).

First, we tried to obtain a regression equation for the delay in terms of various linear combinations of the three factors with no transformations on the factors or on the delay. Figure 3 (left) shows the scatter plot for the predicted values versus the actual values and it can be seen to be highly non-linear.

After the non-linearities were removed, regression was able to explain 99.78% of the variance in the delay (figure 3 right shows the final stage). The regression equation was

$$\begin{aligned} log(Del) \; = & \\ & 1.2946 \times log(S/Bl) \; + \; 5.2768 \times Eb \; + \\ & 1.5555 \times log(N) \; - \; 1.6749 \times log(S/Bl) \times Eb \; - \\ & 3.6725 \times Eb \times log(N) \end{aligned} \tag{1}$$

where *Eb* denotes the service rate.

Figure 4 shows the same stages for cell loss ratio. The variation in cell loss rate was much more difficult to capture, mostly because in a large number of experiments, the observed *Clr* was 0. Since the logarithmic transformation is not defined in this case, we had to eliminate these experiments from consideration. Consequently, only 509 experiments were available for regression. Also,

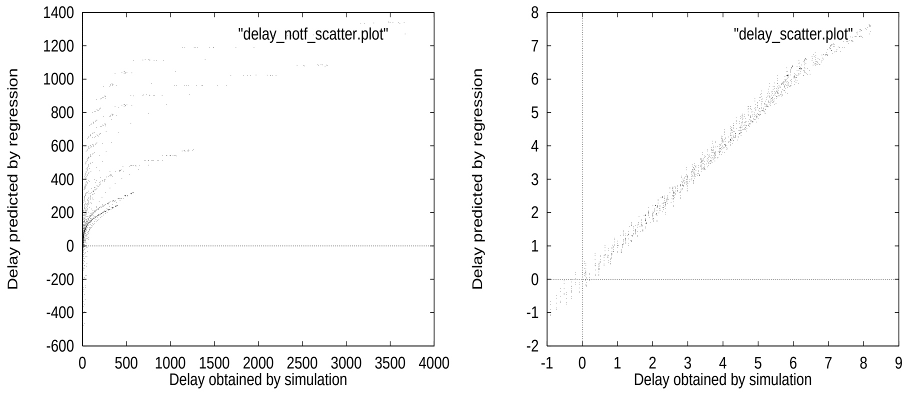

Figure 3 Delay scatter: before transformation (left) and after transformation (right)

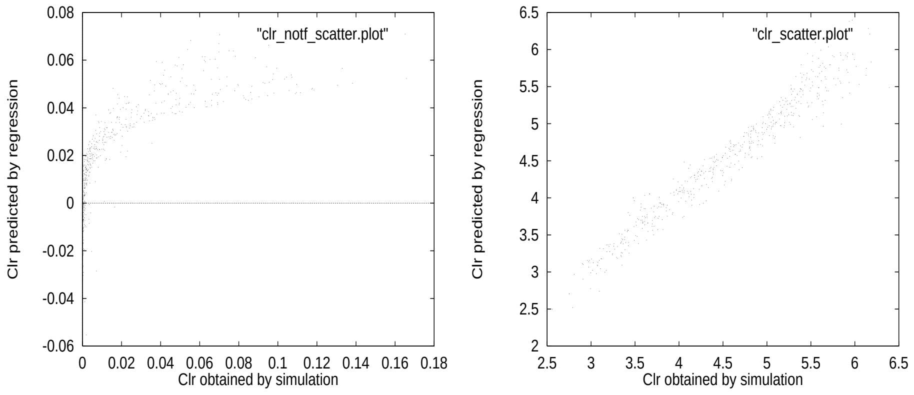

Figure 4 *Clr* scatter: before transformation (left) and after transformation (right)

a highly non-linear transformation had to be applied on the *Clr* to improve the quality of regression. The regression equation obtained for *Clr* was

$$
\begin{array}{lll}
log(log^2(Clr^2)) & = & \\
 & -0.6433 \times log(S/Bl) & - \\
 & 14.519 \times log(Eb) & + \\
 & 0.8781 \times log(N) & + \\
 & 2.8648 \times log(S/Bl) \times log(Eb) & + \\
 & 0.2471 \times (S/Bl)^2 & - \\
 & 0.2910 \times N^2 &
\end{array}
\qquad (2)
$$

It can be seen in figure 4 that the predicted *Clr* is still fairly non-linear with respect to the actual *Clr*. It is this non-linearity that lowers the quality of the effective bandwidth prediction from *Clr*, even though 99.82% of the variation has been explained by regression. Note that the equation for *Clr* requires 6 terms whereas that for delay required only 5.

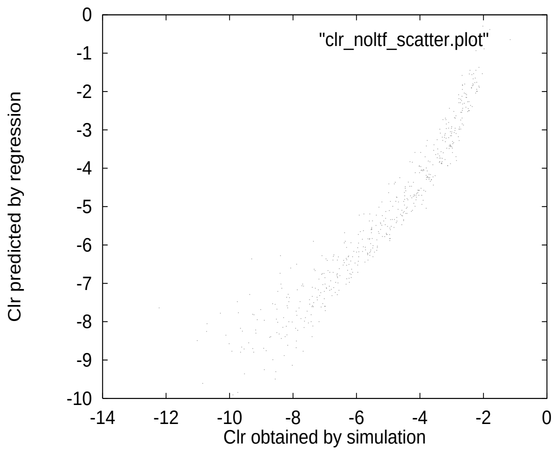

Figure 5 *Clr* scatter (log transformation)

The reason for the complicated transformation on *Clr* in equation 2 can be seen by comparing the scatter plot in figure 5 with that in 4. In figure 5, the predicted values are obtained by merely using the log transformation on the *Clr*.

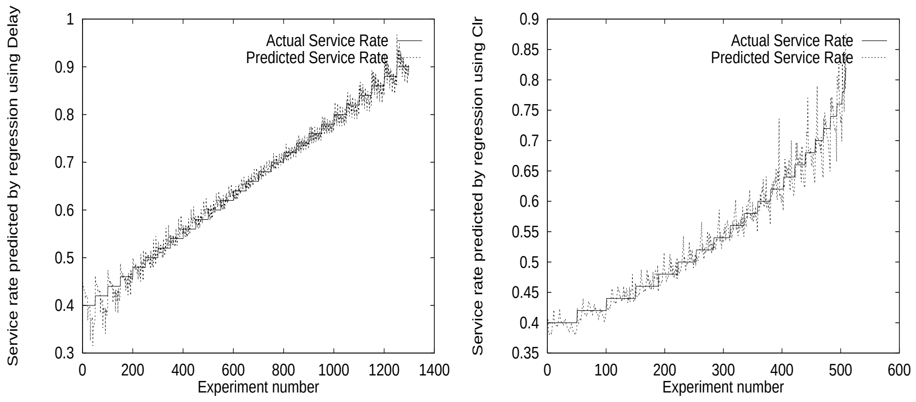

Figure 6 Bandwidth prediction: delay (left), *Clr* (right)

Equations obtained by inverting equations 1 and 2 were used to predict the service rate requirement for a given value of delay, the buffer size to burst length ratio, and the number of calls. Figure 6 (left) plots the values for service rate obtained from the inverted regression equation against the values used in the simulation. It can be seen that the fit is better for the service rate obtained from the delay equation. Figure 6 (right) shows the service rate predictions for *Clr.* *

3 ADAPTIVE LOGIC NETWORKS (ALN)

An Adaptive Logic Network (ALN – Armstrong and Thomas (1996), Armstrong and Thomas (1998)) maps vectors of real values in Euclidean n-dimensional space to boolean values. The first layer of computing units consists of linear-threshold perceptrons that output 1 only if an inequality of the form $w_0 + w_1 \times x_1 + w_2 \times x_2 + \ldots + w_n \times x_n >= 0$ is satisfied. The coefficients w_i of the expression are called the weights of the unit. The boolean outputs of the first layer of units are combined by a tree expression of AND and OR operators of arbitrary fan-in to produce the output of the ALN.

One can also view the ALN in terms of the real-valued function it represents. A functional computation can be derived from the ALN by taking combinations of linear functions where the combining operators are the maximum and minimum of functions. If x_n is the output variable of the ALN, then weights of the first layer of units are normalized to have $w_n = -1$ and the inequality of a unit is turned into a function of the form $x_n = w_0 + w_1 \times x_1 + w_2 \times x_2 + \ldots + w_{n-1} \times x_{n-1}$.

The tree of maximum and minimum operators has the same form as the tree of OR and AND operators respectively. The linear functions have weights which are adapted based on training data consisting of vectors $x_1, \ldots, x_n$ that represent the function graph. The algorithm is like least squares fitting of linear pieces to data points, where a linear piece is only active for a subset of the training points. Given $x_1, \ldots, x_{n-1}$, a subtree of a node contains the active linear piece if its value (computed using the maximum and minimum operators according to the subtree) is less (or, respectively, greater) than the value of any other subtree for an AND (or, respectively, OR) node.

The software that was used in the experiments reported on below refines the piecewise linear functions produced by the above ALN by inserting quadratic fillets at each junction of two linear pieces so that the overall function is continuously differentiable.

ALNs have several advantages for ATM traffic characterization:

- The normalized weights of the active linear pieces are partial derivatives of the output variable with respect to the input variables. Hence the partial derivatives of the learned functions can be directly controlled.
- If an ALN represents a function $x_n = \mathrm{f}(x_1, \ldots, x_{n-1})$ which is monotonic

*Note that adjacent points belong to different experiments and are not related. However, they are joined by lines as this makes it easier to distinguish the predicted values from the simulation values.

increasing in some variable x_i, then a functional computation can be derived from that ALN which computes the corresponding function inverse: $x_i = g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. This uses the coefficients of the same linear pieces, combined in a different way.

- The ALN does not require the predictor variables to be scaled or normalized. This speeds up the model development process as well the use of the model for prediction.

Note that most performance functions are naturally monotonic. Forcing the learned function to be monotonic or convex makes it difficult for the function learned by an ALN to be influenced by the noise in training points; hence overtraining, which prevents good generalization in other neural networks, can be avoided in many cases.

3.1 Choosing the epsilon values

The ALN software allows the user to specify a smoothing parameter (ϵ) associated with each variable that expresses the half-length of an interval which has to be covered by each point in a training set in each axis. Increasing ϵ has the effect of smoothing the function in the direction of the variable, but the network cannot discriminate between points separated by less than ϵ.

In the case of the effective bandwidth problem, we have three input variables: Eb, $log(S)$ and N. The minimum value of the epsilon, $\epsilon_{i,min}$, for an input variable, i, is given by half the smallest interval, $I_{i,min}$, between two adjacent levels of that variable in the training set. The maximum value for epsilon, $\epsilon_{i,max}$, is given by $I_{i,min}$. This prevents over-smoothing of the learned function.

Table 3 Epsilons for D-ALN and C-ALN

Predictor Variable	**Smallest interval**	$\epsilon_{i,min}$	$\epsilon_{i,max}$
Eb	0.02	0.01	0.02
$log(S)$	0.1	0.05	0.1
N	1	0.5	1

To prevent overtraining, it is customary to evaluate at periodic intervals the trained ALN on the test set. This has the disadvantage that the final evaluation on the test set does not really test the generalization of the ALN, since the ALN 'has seen' the test set. Instead, we divide the training set into two portions. Set 1 contains the simulation data for $Eb = 0.45, 0.49, 0.53, \ldots$ and is used as the training data. Set 2 contains the $Eb = 0.47, 0.51, 0.55, \ldots$

After every every set of ten passes (epochs) through the training data (set 1), the trained ALN is evaluated on set 2. The test data is used only when a satisfactorily trained ALN is obtained. It must be noted though that this partitioning of the training data has the effect of changing ϵ_{min} and ϵ_{max} for Eb to 0.02 and 0.04, respectively.

Overtraining is also reduced by constraining the slope of Del w.r.t. the other variables; in this case, delay was constrained to decrease monotonically as Eb or N increases as well as to increase monotonically as a function of buffer size.

3.2 ALN model for cell delay

The inputs to the D-ALN are S, Eb, N and Del. Because of the large ranges of S and Del, we chose to train the D-ALN on $log(S)$ and $log(Del)$, respectively. Since the ALN is scale-invariant, we did not have to normalize the inputs as is usually done with other neural networks.

The ALN software allows the user to specify ϵ_{op} for the output variable. Whenever the root mean square error (RMSE) on the training set is greater than ϵ_{op}, the ALN automatically grows in size to reduce the error. If the ϵ_{op} is set to an unnecessarily small value, the resulting ALN may become very large without corresponding reduction in error on the test set. Too large a value of epsilon may prevent the ALN from learning satisfactorily due to inadequate size.

In order to determine the optimum epsilon on the output variable, ϵ_{del}, for the D-ALN, three different values were tried. Table 4 shows the RMSE on the training set and the average relative error (ARE) on set 2 for each value of epsilon. From the table, we can see that the lowest ARE (0.73%) is is

Table 4 Choosing the output epsilon for D-ALN

Epoch	$\epsilon = 0.05$	$\epsilon = 0.01$	$\epsilon = 0.003$
10	<0.0273, 1.28%>	<0.0271, 1.16%>	<0.0271, 1.16%>
20	<0.0162, 0.84%>	<0.0142, 0.8%>	<0.0142, 0.8%>
30	<0.0139, 0.79%>	<0.0115, 0.79%>	<0.0115, 0.78%>
40	<0.0128, 0.79%>	<0.0096, 0.77%>	<0.0095, 0.75%>
50	<0.0112, 0.81%>	<0.0084, 0.75%>	<0.0085, 0.75%>
60	<0.0114, 0.88%>	<0.0079, 0.76%>	<0.0079, 0.79%>
70	<0.0108, 0.78%>	<0.0070, 0.73%>	<0.0070, 0.78%>
80	<0.0110, 0.97%>	<0.0070, 0.8%>	<0.0068, 0.84%>
90	<0.0112, 0.97%>	<0.0064, 0.81%>	<0.0064, 0.87%>
100	<0.0106, 1.05%>	<0.0058, 0.79%>	<0.0060, 0.84%>

achieved for $\epsilon = 0.01$. Hence ϵ_{del} is chosen as 0.01. The table also shows the point at which training, if continued, would result in poor generalization. For example, when $\epsilon = 0.01$, ARE decreases continuously for the first 70 epochs and then increases. This indicates that training should be stopped after 70 epochs.

Next, we try to determine the optimum epsilon for each of the input variables in a similar manner. For each variable, the values tried are ϵ_{min}, ϵ_{max} and $(\epsilon_{min} + \epsilon_{max})/2$. The epsilon with the lowest ARE is then chosen.

The software tries to lower the RMSE to a specified level, rms_{min}. Setting rms_{min} to a value much below ϵ_{op} has little advantage. On the other hand, setting rms_{min} to ϵ_{op} may not result in a satisfactorily trained ALN. Table 5 shows the RMSE and ARE for three different values of rms_{min}. It is clear

Table 5 Choosing rms_{min} for D-ALN

Epoch	$rms_{min} = 0.01$	$rms_{min} = 0.005$	$rms_{min} = 0.001$
10	<0.0277, 1.2%>	<0.0277, 1.2%>	<0.0277, 1.2%>
20	<0.0143, 0.71%>	<0.0143, 0.71%>	<0.0143, 0.71%>
30	<0.0112, 0.69%>	<0.0112, 0.69%>	<0.0112, 0.69%>
40	<0.0103, 0.71%>	<0.0103, 0.71%>	<0.0103, 0.71%>
50	<0.0086, 0.72%>	<0.0086, 0.72%>	<0.0086, 0.72%>
60	-	<0.0076, 0.77%>	<0.0076, 0.77%>
70	-	<0.0070, 0.84%>	<0.0070, 0.84%>
80	-	<0.0068, 0.85%>	<0.0068, 0.85%>
90	-	<0.0064, 0.82%>	<0.0064, 0.82%>
100	-	<0.0063, 0.90%>	<0.0063, 0.9%>

that there is little advantage in choosing $rms_{min} < 0.01$. It can also be seen that training should be stopped after 30 epochs.

Figure 7 compares the estimates of the D-ALN on the training and test sets with simulation results. Since the test set was generated for $Pcr = 1000$ cells/s while the training set used $Pcr = 14\,150$ cells/s, the figure shows that the trained ALN can be used to predict delays for other values of Pcr fairly accurately as long as Bl and Av/Pk remain the same.

3.3 ALN model for cell loss

The inputs to the C-ALN are the same as those to the D-ALN. The ALN is trained on $log(Clr)$. The points in the training set for which $Clr = 0$ are eliminated. This results in the C-ALN being trained on 899 data points, which is still adequate considering that we have only three input variables.

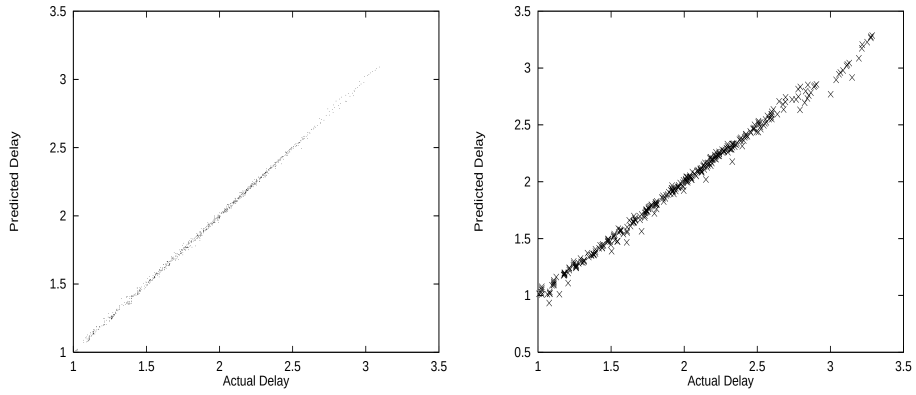

Figure 7 D-ALN model predictions: training set (left) and test set (right)

As in the case of the D-ALN, the training was done on one half of the training set, with the other half being used to check for overtraining. The learned function was constrained by specifying that the *Clr* is non-decreasing for an increase in Eb, N, or S.

The optimum epsilons for Eb, $log(S)$ and N were obtained, in each case being equal to the respective minimum epsilons. The optimum output epsilon, ϵ_{clr}, was found to be 0.05. The best setting for rms_{min} was also 0.05. Training was stopped after 40 epochs.

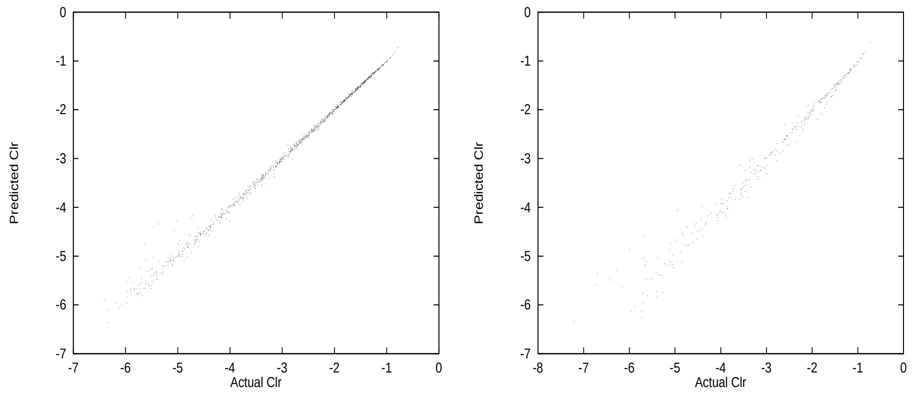

Figure 8 C-ALN model predictions: training set (left) and test set (right)

Figure 8 compares the predictions of the C-ALN model on the training and test sets with simulation results. As in the case of *Del*, the results obtained by evaluating the trained model on the test set show that the ALN can be used to predict *Clr* for different values of *Pcr*. Section 4 presents a numerical comparison of the *Clr* predictions by the C-AMS, C-REG and C-ALN models.

The performance of the C-ALN model on the test set is not as good as for the D-ALN model. The maximum relative error on the test set (19.9%) is almost twice that of the D-ALN (11.8%). We can also see a larger scatter at low loss ($< 10^{-6}$) in figure 8. This is primarily because of the smaller training set due to the missing elements corresponding to zero *Clr*. These values could be replaced by more accurate estimates from longer simulation runs at high service rates and large buffer sizes.

Figure 9 (left) compares D-ALN model predictions for $N = 30$ (a value not in the training set) with simulation values. Figure 9 (right) shows the ability of the C-ALN model to extrapolate accurately.

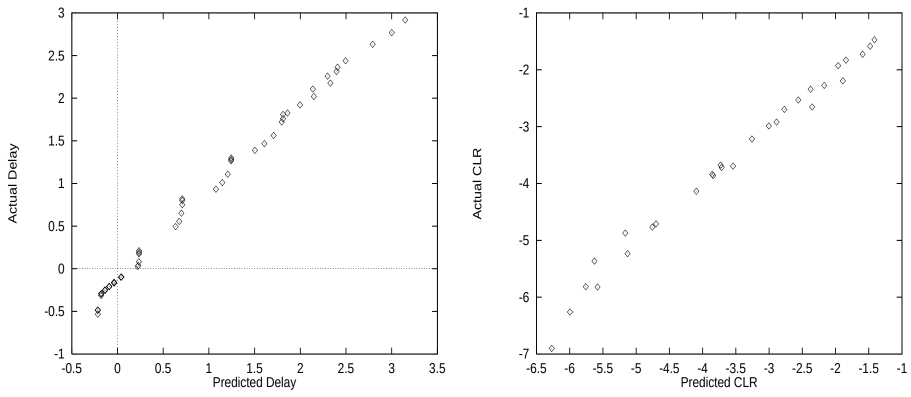

Figure 9 Extrapolation for $N = 30$: D-ALN model (left) and C-ALN model (right)

3.4 Residual analysis for ALN models

Since the ALN uses the ordinary least squares (OLS) principle to determine the orientation of the linear pieces, the trained ALN is subject to the assumptions inherent in the use of OLS.

The assumptions made in the case of OLS are (Gunst and Mason 1980):

- Predictor variables are non-stochastic and measured without error. Since the predictor variables are controlled inputs to the simulation, this statement is true.
- Model error terms follow a normal probability distribution. This can be seen to be approximately true from the histogram plots of $log(Del)$ and $log(Clr)$ residuals (figure 10).
- Any two errors are independent of each other. The presence of correlation reduces the reliability of the model. To verify this, we plot the residuals against the values obtained by simulation (figure 11).

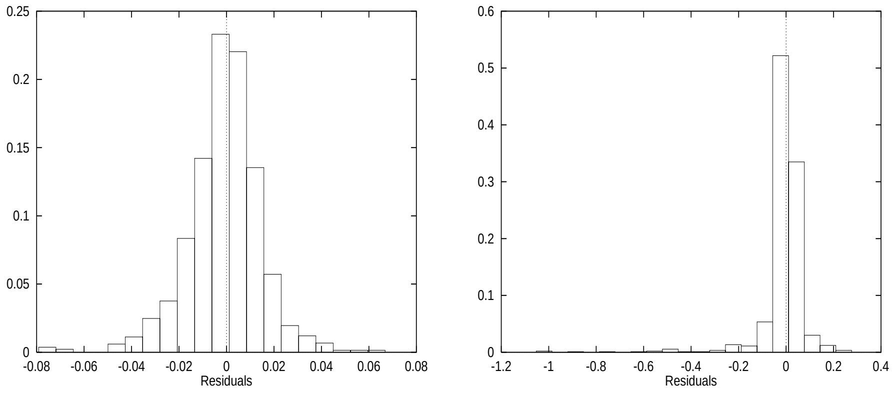

Figure 10 Histograms of residuals: D-ALN (left) and C-ALN (right)

- Model error terms have zero means, are uncorrelated, and have constant variances. The first of these can be seen to be true by observing that the standard deviation limits appear equidistant from the horizontal axis (figure 11). The second condition is generally true for databases compiled from controlled laboratory experiments. The third assumption is discussed below.

From the $log(Del)$-residuals plot, we can see that the errors are randomly distributed on either side of the horizontal axis, indicating that there is no systematic error. Since the log function is nonlinear for values of the abscissa < 10, we have eliminated these values from the plot. In any case, we are more interested in delay predictions at higher delays.

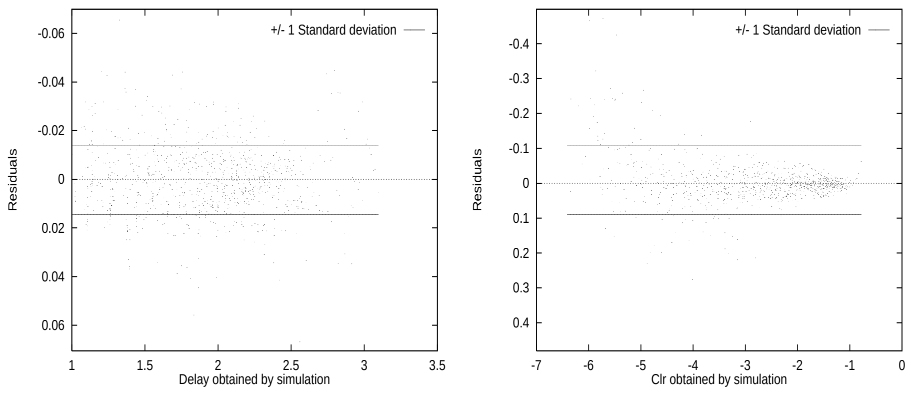

Figure 11 ALN residuals plots: log(Del) (left) and log(Clr) (right)

The $log(Clr)$-residuals plot shows that the residuals are fairly random for $Clr > 10^{-6}$. For lower cell loss, the simulation results themselves are inaccurate. The residuals are seen to increase in magnitude as $log(Clr)$ decreases below -4, leading us to conclude that there is a small amount of heteroscedasity (unequal variances).

4 COMPARISON OF DELAY AND CELL LOSS PREDICTIONS

We compare here the delay and cell loss predictions by the AMS, ALN and REG models against the delay values obtained from the simulations.

Table 6 makes a numerical comparison of the delay predictions based on the average and maximum *relative* errors* in the test and training sets. From the table, we can see that the D-ALN performs the best on both the test and training sets while the D-REG model comes a close second. The average error of the D-ALN on the test set is almost half that of the D-REG model and only one-third that of D-AMS. The similarity of the average errors on the test

Table 6 Numerical comparison of delay predictions

Method	*Training set*		*Test set*	
	Maximum error	*Average error*	*Maximum error*	*Average error*
D-AMS	14.6%	3.3%	16.1%	3.0%
D-REG	11.9%	1.95%	12.2%	2.3%
D-ALN	4.93%	0.6%	11.8%	1.32%

and training sets for both the simulation-based models indicates that we have been successful in curtailing model overtraining.

From the scatter plots, it appears that the C-ALN model does very well at high cell loss, but its performance declines at lower cell loss. The C-REG model appears more consistent. Both models clearly perform better than the C-AMS model.

Table 7 makes a numerical comparison of cell loss predictions based on the average and maximum *relative* errors in the test and training sets. We see that the C-ALN is considerably better than the analytical model, with its average error on the test set being only one-eighth that of the latter. The C-REG model is closer to the C-ALN model in terms of average error, but its maximum error is much larger. This indicates that the simulation-based models can be fairly accurate while offering the benefit of being computationally inexpensive.

*The errors for the delay and cell loss models are computed by comparing against simulation results for $log(Del)$ and $log(Clr)$, respectively.

Table 7 Numerical comparison of cell loss predictions

Method	*Training set*		*Test set*	
	Maximum error	*Average error*	*Maximum error*	*Average error*
C-AMS	56.5%	21.7%	61.7%	24.0%
C-REG	17.1%	3.0%	32.4%	5.2%
C-ALN	19.28%	1.27%	19.9%	3.85%

5 APPLICATION: EFFECTIVE BANDWIDTHS

In the preceding section, we examined three methods of estimating the cell loss and mean delay in our system. We now apply those methods to effective bandwidth computation. We compute effective bandwidths that satisfy cell loss requirements, mean delay requirements, and both. Two requirement sets are considered: the first requirement set ($Clr \leq 10^{-2}$, delay ≤ 100) corresponds to a situation with stringent delay and jitter requirements but relatively high tolerance for loss (e.g., real-time video); the second requirement set ($Clr \leq 10^{-4}$, delay ≤ 1000) corresponds to the more common case where low loss is required, but delay can be tolerated. Throughout this section, we set $Pcr = 14\,150$, $t_{ON} = 0.025$, and $t_{OFF} = 0.035$ (this yields an Scr (sustained rate) of $0.417 \times Pcr = 5896$). The bandwidths are plotted per source and relative to the Pcr, and are thus in $[0.417, 1]$.

Of the three methods, only the ALN could be explicitly inverted; to obtain effective bandwidths from the regression function, and in the AMS model, we used a simple binary iteration approach. We found that generally, accurate results can be obtained in only 5–10 iterations. The declared range of input values for the ALN was $[0.4, 1]$, and the inverted ALN therefore gave values in $[0.4, 1]$, sometimes returning values slightly below the Scr.

5.1 Effective bandwidth vs. number of calls

We compute effective bandwidths as a function of the number of calls, which we vary from 1 to 20. The buffer size S is fixed at 5000. The results are plotted in figure 12 (requirement set 1 on the left, requirement set 2 on the the right). There are six lines per requirement set: for each of the three approximation methods there is a line for the delay-based effective bandwidth and another for the cell-loss-based effective bandwidth. The effective bandwidth needed to satisfy both delay and cell loss requirements is simply the maximum of the two lines.

We note that the delay requirement is dominant in requirement set 1. The three estimates for delay-based effective bandwidth are very close to each other

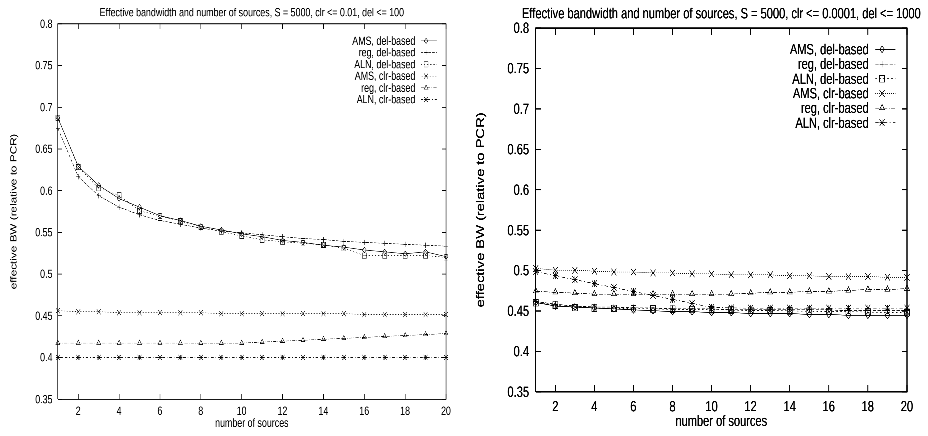

Figure 12 Effective bandwidth vs. N

and show the typical decaying shape (indicating a statistical multiplexing gain). The estimates for the loss-based effective bandwidth are close to the Scr even for $N = 1$ and do not change when N is increased. In this case, it is clear that the ALN underestimates the effective bandwidth, because the estimate is about 5% below the Scr.

For requirement set 2, the loss requirement dominates slightly. As expected, the tighter loss requirements increased the loss-based effective bandwidth, and the less stringent delay requirements reduced the delay-based effective bandwidth.

5.2 Effective bandwidth vs. buffer size

We compute the effective bandwidths as a function of the buffer size, which we vary from 1 to 10 000. The number of calls N is fixed at 10. The results are plotted in figure 13 (requirement set 1 on the left, requirement set 2 on the right). Once again, there are six lines per requirement set: for each of the three approximation methods, there is a line for the delay-based effective bandwidth and another for the cell-loss-based effective bandwidth.

The values for delay-based effective bandwidth computed by the three methods exhibit a characteristic *step* form: the delay requirements can be satisfied by any bandwidth if the buffer size is less than the delay requirement; in our figure, the dots are plotted close to the sustained cell rate line. Once the buffer size is relatively large, adding buffers leaves the mean delay (and therefore the delay-based effective bandwidth) at a constant level. Between these two extremes, there is a narrow range of buffer values where an increase in buffer size leads to an increase in mean delay. The reason for this increase is a simultaneous sharp decrease in the cell loss rate, resulting in the delay of cells that would otherwise be dropped.

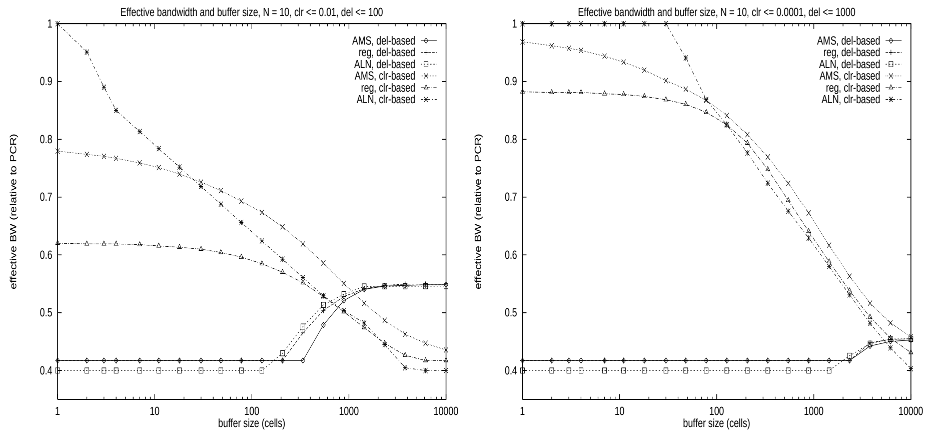

Figure 13 Effective bandwidth vs. S

Comparing the graphs for loss-based and delay-based effective bandwidth, we note that the cell loss requirement dominates for small buffer sizes and that the delay requirement dominates for larger buffer sizes. The crossover point between these two regions depends on the requirements and on the traffic characteristics. There is no increase in statistical multiplexing gain once the cross-over point has been reached.

The three estimates for loss-based effective bandwidth are divergent when effective bandwidth is high (i.e., when the buffer size is small), and somewhat closer when effective bandwidth is low. This can be explained by the fact that the regression function and the ALN are based on simulations of buffer sizes ≥ 400, and are therefore inaccurate when buffer sizes are small. In the case of delay-based effective bandwidths, the three estimates are very close to each other. The loss-based curves have a shape similar to those obtained by Rege (1994).

6 CONCLUSIONS

We have discussed a simple and reasonably accurate scheme for predicting the delay and cell loss when a number of bursty sources are multiplexed at a link with finite buffer space. Unlike other schemes, our model uses the number of sources N as an input, leading to delay and loss computation times that are independent of the number of sources being multiplexed. Though we have used On/Off bursty sources in order to compare the results with other effective bandwidth schemes, the techniques presented here can be extended to complex sources that cannot easily be modeled analytically, e.g., aggregate LAN traffic, MPEG traffic.

The ALN model has the advantage over other neural networks that the same network that is trained to predict cell loss or delay can be inverted to predict effective bandwidth instead. This means that the ALN can make effective bandwidth predictions as quickly as it can make delay or loss predictions. Since the evaluation time is very small – of the order of milliseconds – ALNs are highly suitable for use in real-time CAC.

Note that although the regression model is only somewhat worse than ALN, it was very painful to build (mostly by educated trial and error), it cannot be easily reversed, and it is completely useless (must be rebuilt practically from scratch) for a traffic pattern with different characteristics.

ALN implementation in hardware is easy because the ALNs are composed of AND gates, OR gates and simple linear threshold elements. This can result in an increase in evaluation speeds by an order of magnitude or more.

REFERENCES

Anick, D., Mitra, D. and Sondhi, M. M.: 1982, Stochastic theory of data–handling system with multiple sources., *The Bell Technical Journal* **61**(8), 1871.

Armstrong, W. W. and Thomas, M.: 1996, *Handbook of Neural Computation – Adaptive Logic Networks(Section C1.8)*, Oxford University Press.

Armstrong, W. W. and Thomas, M.: 1998, Atree 3.0 Educational kit for Windows 3.x, 95, NT. available from http://www.cs.ualberta.ca/~arms.

Decina, M. and Toniatti, T.: 1990, On bandwidth allocation to bursty virtual connections in ATM networks, *ICC '90* **3**, 844–851.

Gburzynski, P.: 1996, *Protocol Design for Local and Metropolitan Area Networks*, Prentice–Hall, New Jersey.

Guerin, R., Ahmadi, H. and Naghsineh, M.: 1991, Equivalent capacity and its application to bandwidth allocation in high-speed networks., *IEEE Journal on Selected Areas in Communications* **9**(7), 968.

Gunst, R. and Mason, R. (eds): 1980, *Regression Analysis and its Application — A Data–Oriented Approach*, Marcel Dekker Inc., New York.

Hyman, J., Lazar, A. and Pacifici, G.: 1991, Real–time scheduling with quality of service constraints., *IEEE Journal on Selected Areas in Communications* **9**(7), 1052–1063.

Murase, T., Suzuki, H., Sato, S. and Takeuchi, T.: 1991, A call admission control scheme for ATM networks using a simple quality estimate., *IEEE Journal on Selected Areas in Communications* **9**(9), 1461.

Rege, K. M.: 1994, Equivalent bandwidth and related admission criteria for ATM systems–a performance study., *International Journal of Communication Systems* **7**, 181.

SPSS: 1992, *SPSS for Windows Advanced Statistics Release 5*, SPSS Inc., Chicago, IL.

Sykas, E. D., Vlakos, K. M., Tsoukatos, K. P. and Protonotarios, E. N.: 1993, Congestion control–effective bandwidth allocation in ATM networks., *IFIP Transactions C [Communication Systems]* **C–14**, 65.