Traffic Control in Unicast ATM ABR Service using Adaptive Approach

Trilok Chand Aseri\textsuperscript{1} and Dr. Deepak Bagai\textsuperscript{2}

\textsuperscript{1}Sr. Lecturer
Department of Computer Science & Engineering
Punjab Engineering College (P.E.C.), Deemed University
Sector-12, Chandigarh-160012 (India)
E-mail: a_trilok_chand@yahoo.com

\textsuperscript{2}Assistant Professor
Department of Electronics & Electrical Communication Engineering
Punjab Engineering College (P.E.C.), Deemed University
Sector-12, Chandigarh-160012 (India)
E-mail: dbagai@yahoo.com

Abstract

This paper addresses the problem of rate control for Available Bit Rate (ABR) service class in Asynchronous Transfer Mode (ATM) networks. An adaptive neuro-fuzzy mechanism based on Adaptive Network Fuzzy Inference System (ANFIS) for allocating rates in ABR service has been proposed and compared with the fuzzy technique called as Fuzzy Explicit Rate Marking (FERM). To achieve this, a neuro-fuzzy ANFIS controller has been built and its control actions are compared with FERM. Simulations are carried out. Network throughput and average cell delay at the terminal end; and percentage cell drop at the switches are calculated.

Key-Words: - Traffic Control, Asynchronous Transfer Mode, Available Bit Rate, Neuro-fuzzy

1. Introduction

Asynchronous Transfer Mode (ATM) is a high speed packet switching technology which is used to transmit voice, video and data over high speed links using short fixed size cells consisting of 48 bytes of payload and 5 bytes of header \cite{1}. ATM networks offer various service classes such as Constant Bit Rate (CBR), real time-Variable Bit Rate (rt-VBR), non real time – Variable Bit Rate (VBR), Available Bit Rate (ABR), Unspecified Bit Rate (UBR) and Guaranteed Frame Rate (GFR). Of these, Available Bit Rate (ABR) service has been introduced to support highly bursty traffic data applications.

Designing effective traffic control techniques in ATM ABR service have been difficult because of the nature of the traffic. The traditional techniques \cite{2} of traffic control can be supplemented with a variety of novel control techniques such as Computational Intelligence (CI) and Artificial Intelligence (AI). One such computational intelligence method called as FERM (Fuzzy Explicit Rate Marking) \cite{3}, which uses fuzzy logic, has been successfully used to address the problem of traffic control. But the controller developed using FERM has to be tuned according to the
environment manually which can be a cumbersome task as parameters of the membership functions has to be adjusted.

Therefore, there is a need of effective method for tuning the membership functions so as to minimize the output error measure or maximize performance index. Some soft computing technique, which has the power of representation of fuzzy logic and is able to adapt or fine-tune the knowledge base, can be used. ANFIS (Adaptive Network Fuzzy Inference System) [4] as a neuro-fuzzy algorithm and one of the best function approximator gives us an alternative approach to design a controller. It requires an initial rough idea of membership functions and the training data, which is nothing but set of inputs and desired responses for that particular situation. ANFIS then adjusts the parameters using its hybrid learning approach, which otherwise we would have done manually. This work basically aims at demonstrating that ANFIS can be used as a better alternative to conventional ATM fuzzy controller [3][5][6][7], as it reduces the amount of effort needed to adjust the shape of the membership functions or fine tune it.

The rest of the paper is organized as follows. Implementation details regarding the controller are given in section 2. Section 3 illustrates the simulation of a network model in NIST simulator. Training and simulation results are given in section 4. Section 5 concludes the paper.

2. Implementation Details

2.1 Architecture of ANFIS

For building a Fuzzy Inference System (FIS), we have to specify the fuzzy sets, fuzzy operators and knowledge base. But, for constructing an ANN for an application, the user needs to specify the architecture and learning algorithm [8]. Integrated neuro-fuzzy system combines the advantages of ANN and FIS. While the learning capability is an advantage from the viewpoint of FIS, formation of knowledge base is an advantage from the viewpoint of ANN. One such neuro-fuzzy algorithm called Adaptive Neuro Fuzzy Inference System (ANFIS) [4] has been used to build neuro-fuzzy controllers. ANFIS is a feedforward type adaptive network, which is functionally equivalent to an FIS. It basically consists of five layers, having adaptive and non-adaptive nodes.

Structure of ANFIS used is shown in Figure 1. To reflect different adaptive capabilities, both square and circle node symbols have been used. A square node (adaptive node) has parameters, while a circle node (fixed node) has none. To achieve a desired input–output mapping, parameters are updated according to given training data using a hybrid approach of gradient descent and a Recursive Least Square Estimator (RLSE), based learning procedure.

ANFIS as an ATM traffic controller has been thought of as consisting of two inputs and one output. Input pair is comprised of average queue length (AveQLen), which is calculated over an averaging interval, and change in average queue length (DeltaQ), which is nothing but queue growth rate. Output given by neuro-fuzzy controller is Fractional Flow Rate (FFR), which is the value between 0 and 1. Each input variable is further described by its membership functions or adjectives shown in Figure 1. There are no membership functions corresponding to output variable, FFR, these are not needed as ANFIS automatically maps the right value, which it learns while training. ANFIS needs only a rough idea of the membership functions of the input linguistic variable, which it refines during training using hybrid learning method. Description of adjectives of the linguistic variables is described in the Figure 1 with the help of different hatches. Rule base chosen over here is based on the concept of
allowing maximum flow, whenever it is possible; this is clearly visible from the rule base shown in Figure 2.

Figure 1: Structure of ANFIS used as a neuro-fuzzy controller

Figure 2: Rulebase used in ANFIS controller
Chosen specifications of each layer are as follows,

Layer I: Every node in this layer is a square node, which computes the degree of membership of the input. Every node has a function of type given in equation (1),
\[ O^1_i = \mu_{A_i}(x), \]  
where, \( x \) is the input to node \( i \) and \( A_i \) is the linguistic label associated with this node function. Each node is using bell-shaped membership function as given in equation (2),
\[ \mu_{A_i}(x) = \frac{1}{1 + \left( \frac{x - c_i}{a_i} \right)^2}, \]  
where, \( \{a_i, b_i, c_i\} \) is a parameter set. The parameters in this layer are referred to as premise parameters. The hybrid algorithm adjusts these premise parameters to achieve the optimal shape of the member functions.

Layer II: Every node in this layer is the circle node labeled \( \prod \), that multiplies the incoming signals and sends the product out, as shown in equation (3),
\[ O^2_i = w_i \times \mu_{A_i}(x_i) \times \mu_{B_i}(x_j), \]  
Where, \( i=1\ldots46; \ j, k=1\ldots5 \)  
Product T-norm operator has been used to perform fuzzy AND operation (Algebraic product as given in equation (4)).
\[ T_{mn}(a,b) = ab \]  

Layer III: Each node in this layer is a circle node labeled \( \mathcal{N} \) (normalized). The ratio of ith rule firing strength to the sum of all rules’ strengths is calculated over here, given in equation (5),
\[ O^3_i = \frac{w_i}{\sum_i w_i}, \]  
Where \( i=1 \) to \( 46 \)

Layer IV: Every node \( i \) in this layer is a square node with a node function as given in equation (6),
\[ O^4_i = \bar{w}_i f_i = \bar{w}_i (p, x_1 + q, x_2 + r_i) \]  
where, \( \bar{w}_i \) is the output of layer 3 and \( \{p, q, r_i\} \) is the parameter set, \( f_i \) is referred to as consequent parameters, \( x_1 \) is AveQLen (Average Queue Length) and \( x_2 \) is DeltaQ (Change in Queue Length). The parameters \( \{p, q, r_i\} \) are adjusted through RLSE (Recursive Least Square Estimator).

Layer V: Single fixed node in this layer is labeled \( \sum \). The overall output is computed as the summation of all incoming signals, as given in equation (7),
\[ O^5_i = y = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \]  

2.2 Modules of Neuro-Fuzzy Controller

Neuro-fuzzy controller is realized in two parts: ANFIS controller trainer and ANFIS controller driver. The trainer module trains the ANFIS controller according to the supplied training data. It does that by using the hybrid learning method of gradient descent and least square estimator. This module basically requires rough initial premise parameters for the membership functions of the layer 1 (as shown in Table 1) and supplied training data (as shown in Figure 3). Consequent parameters are learnt during forward pass of the hybrid learning algorithm using recursive least square estimator (RLS) and premise parameters are learnt using gradient descent method. The pseudo code for the implemented hybrid method is given in Figure 4. Output of the ANFIS-trainer is a text file whose contents are shown in Table 2, which can be used by the ANFIS-driver program as its input. The pseudo code for the ANFIS driver program is given in Figure 5.
Table 1: Initial parameter list

<table>
<thead>
<tr>
<th>α/ρ₁</th>
<th>β₁/ρ₂</th>
<th>γ₁/ρ₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>437</td>
<td>6.77</td>
<td>-138</td>
</tr>
<tr>
<td>184</td>
<td>4.81</td>
<td>932</td>
</tr>
<tr>
<td>246</td>
<td>3.94</td>
<td>-340</td>
</tr>
<tr>
<td>20.7</td>
<td>1.84</td>
<td>-78.7</td>
</tr>
<tr>
<td>31.4</td>
<td>1.87</td>
<td>3.85</td>
</tr>
<tr>
<td>29.7</td>
<td>1.59</td>
<td>100</td>
</tr>
<tr>
<td>202</td>
<td>8.77</td>
<td>351</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3: Supplied training data

Table 2: Final parameter list

<table>
<thead>
<tr>
<th>α/ρ₁</th>
<th>β₁/ρ₂</th>
<th>γ₁/ρ₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>433</td>
<td>6847935027</td>
<td>7.9337142511</td>
</tr>
<tr>
<td>187</td>
<td>6527801635</td>
<td>4.5276680737</td>
</tr>
<tr>
<td>244</td>
<td>2556577779</td>
<td>2.7269522987</td>
</tr>
<tr>
<td>174</td>
<td>0.2912356433</td>
<td>0.2355740339</td>
</tr>
<tr>
<td>0.1910494491</td>
<td>3.5991702884</td>
<td>-81.4225717168</td>
</tr>
<tr>
<td>31.7</td>
<td>7105873156</td>
<td>6.0454768807</td>
</tr>
<tr>
<td>20.2</td>
<td>0.111331348</td>
<td>2.7473340640</td>
</tr>
<tr>
<td>202.2</td>
<td>2061780457</td>
<td>5.9714678601</td>
</tr>
<tr>
<td>0.0000617662</td>
<td>0.0001618797</td>
<td>0.0999380129</td>
</tr>
<tr>
<td>0.0000278193</td>
<td>-0.0004369902</td>
<td>0.0191458458</td>
</tr>
<tr>
<td>0.00001876990</td>
<td>-0.0000800919</td>
<td>0.0081245585</td>
</tr>
<tr>
<td>0.00004230476</td>
<td>-0.0002004445</td>
<td>0.3944335662</td>
</tr>
<tr>
<td>0.0000217555</td>
<td>0.0005799140</td>
<td>0.0096750790</td>
</tr>
<tr>
<td>0.0000250039</td>
<td>0.0001319311</td>
<td>0.0006106552</td>
</tr>
<tr>
<td>-0.00000391762</td>
<td>-0.00000362783</td>
<td>0.0562003190</td>
</tr>
<tr>
<td>-0.00000351942</td>
<td>-0.0002257514</td>
<td>0.0001816664</td>
</tr>
<tr>
<td>-0.0000183435</td>
<td>-0.0000585601</td>
<td>0.2140135859</td>
</tr>
<tr>
<td>-0.00000382226</td>
<td>0.0042675208</td>
<td>0.0003537161</td>
</tr>
<tr>
<td>0.0000036698</td>
<td>0.0015591074</td>
<td>0.0003837857</td>
</tr>
</tbody>
</table>

Figure 3: Supplied training data
The pseudo code for the implemented hybrid method is as follows:

```c
/* ANFIS Controller Trainer Routine */
ANFIS_Trainer (Num_of_Epochs, Num_of_Training_Data)
{
    /* Get initial premise parameter */
    Get_parameter (INIT_PARA_FILE);
    /* Get Training Data */
    Get_data (TRAIN_DATA_FILE, Num_of_Training_Data, training_data_matrix);
    For (i; i< Num_of_Epochs; i++)
    {
        /* Consequent parameters are updated using RLS */
        Forward_Pass (Num_of_Training_Data, training_data_matrix);
        /* delta is updated using gradient descent */
        Backward_Pass (Num_of_Training_Data, training_data_matrix);
    }
    /* Update premise parameters in 2st layer */
    Update_parameter (step_size);
}
```

Pseudo code of forward pass used above, which calculates new consequent parameters using recursive least square estimator (RLS) is given as follows:

```c
/* Forward Pass */
Forward_Pass (Num_of_Training_Data, Training_data_matrix)
{
    For (i; i< Num_of_Training_Data; i++)
    {
        /* Pass the ith input pair (i.e. Average Queue Length and Change in Queue Length) to the adaptive network */
        Put_input_data (i, training_data_matrix);
        /* Calculate node outputs from layer 1 to layer 3 */
        Calculate_output (From Layer 1 to Layer 3);;
        /* Get the desired output corresponding to above training data */
        Target = Get_Desired_Output (training_data_matrix);
        /* Get new set of consequent parameters using recursive least square (RLS)*/
        New_Parameters = RLS (Target, training_data_matrix, New_Parameters);
    }
    /* Put the parameters identified by RLS algorithm into adaptive network */
    Put_parameter (New_Parameters);
}
```

Pseudo code for backward pass used in ANFIS driver program, backward pass uses gradient descent method to update premise parameters:

```c
/* Backward Pass */
Backward_Pass (Num_of_Training_Data, training_data_matrix)
{
    For (i; i< Num_of_Training_Data; i++)
    {
        /* Pass the ith input pair (i.e. Average Queue Length and Change in Queue Length) to the adaptive network */
        Put_input_data (i, training_data_matrix);
        /* Calculate node outputs from layer 4 to layer 5 */
    }
}
```

Figure 4: Pseudo-code for hybrid method
2.3 Applying Neuro-Fuzzy Controller

Neuro-Fuzzy controller is applied at the output buffers of the ATM switch as shown in Figure 6. It takes QLen (Average queue length) and DeltaQ (change in average queue length) as its input. Queue length is averaged over an interval called as an Averaging Interval (AI). The queue growth rate or DeltaQ, is basically the difference between the QLen of two consecutive AIs. Output of controller is FFR (Fractional Flow Rate), which is the value between 0 and 1. This FFR is multiplied with the link cell rate to get the ER (Explicit rate), which is then conveyed through the returning RM cell or BRM (Backward RM) cell to the sources as the new rate. The sources then adjust their ACR (Allowed Cell Rate) according to the new ER conveyed (equation (8)).

$$\text{ER} = \text{FFR} \times \text{Link Cell Rate} \quad \text{(8)}$$

where, ER is Explicit Rate (Mb/sec), FFR is Fractional Flow Rate (between 0 and 1), Link Cell Rate is maximum speed of output physical link.

To work, neuro-fuzzy controller needs a network configuration file, which is nothing but a file containing matrix, with rows representing source node number and columns representing destination number. Configuration matrix as shown in Figure 7 is supplied to the neuro-fuzzy controller as a text file. This configuration corresponds to the network shown in Figure 1. All zeros represent that there is no link between source and destination nodes and all ones represent that there is a link between source and destination nodes of the adaptive network. The configuration file (shown in Figure 7) corresponds to the implemented ANFIS controller.
Neuro-fuzzy controller also takes initial parameter list of its adaptive nodes in the form of the text file as shown in Table 1. The parameter set which corresponds to the premise parameters are \{ai, bi, ci\} and parameter set which corresponds to the consequent set are \{pi, qi, ri\}.

Figure 6: Application of fuzzy controller within ATM switch

Figure 7: Configuration matrix of the network supplied to NF controller
2.4 Integrating with the Simulator

To test the algorithms, a simulator called as NIST ATM/HFC simulator has been used [9]. Switch, consisting of ANFIS and FERM as controllers, has been implemented as a component of simulator. Neuro-fuzzy controller has been applied at the output ABR buffer of the switch, so is FERM as a conventional fuzzy controller. ABR output buffer is maintained per physical port of the switch. There is only one integrated input buffer per port for all kinds of services, which ATM supports, but the cells are segregated into buffers according to the services as shown in Figure 8. Switch contains four functions mainly responsible for switch’s processing:

sw_myreceive: It accepts the incoming cells and puts them into the input queue maintained per port and enforces the discard policy if the input size is increasing the input queue threshold. The threshold policy it uses is random.

sw_proc_slot_time: It is the clock of the switch, and its call time depends upon the speed of the switch. It is regularly called after certain number of ticks.

sw_demux: This function is called by sw_proc_slot_time() to demultiplex the input queue maintained per port into ABR, VBR and CBR queues at the corresponding output ports.

sw_schedule_output: This function is called by the sw_proc_slot_time() to schedule the output of the queues at the output the output port according to the priority of the service i.e. first CBR queue is processed and then VBR queue is processed and then the ABR queue is processed.

Neuro-fuzzy controller takes the cells from the sw_demux and calculates the new explicit rate and sets it in the outgoing cells. It then passes the cell to the sw_schedule_output for the priority wise dispatch. Pseudo code of the integrated neuro-fuzzy controller is given in Figure 9.

![Switch architecture with neuro-fuzzy controller within ATM switch](image)
3. Simulation Model

3.1 Description of the Model

ATM network model used, shown in Figure 10, consists of three switches. It has been designed to capture interference between the traffic traveling a different number of hops and interference of the background traffic of higher order services like VBR competing for the resources and their effect on the total throughput and cell delay. ABR sources are acting at first and second switch; VBR is interfering the normal flow of ABR traffic at third switch. There is single common buffer per port at the input of the switch and at the output port queuing takes place according to the service (i.e. for ABR and VBR, we have different buffers, see Figure 8). VBR is taken as the background traffic, rest all sources are ABR sources. There are ten 3-hop ABR batch sources, five 2-hop ABR batch sources and four 1-hop VBR batch sources in the model. These ABR and VBR sources are connected to the Broadband Terminal Equipment (BTE).

3.2 Assumptions and Parameter Settings

Queuing is assumed to occur at both input and output buffers.

It is assumed that ABR buffers can accommodate 1024 cells and VBR buffer can accommodate 200 cells. A simple priority mechanism is adopted within simulator for servicing ABR and VBR queues, first VBR queue is checked if there are no cells in the VBR queue, ABR queue is serviced then.

Inter-switch distances are assumed to be of 10 Km.

Lengths of access links connecting the terminals to the ATM switches are taken as 2 Km long in the model.
All the links are assumed to have 155Mb/sec transmission speed i.e. link cell rate would be 155 Mb/sec.

All ABR sources are assumed to be of batch type. The mean number of cells that are to be sent are taken to be 2000, also the mean interval between the bursts is specified to the ABR sources, which is taken to be equal to 5 msec.

VBR sources are also batch sources. Mean number of cells, which are generated are taken to be 500, with mean interval of 5 msec between bursts. The load increases from 20% of the link rate to 150% of the link cell rate.

Fuzzy controller, using FERM as a traffic control algorithm, has been tuned according to above environment manually. Neuro-fuzzy controller, using ANFIS, has been given the training data which is in compliance with the thinking of rulebase and tuning objectives i.e. to maximize the flow rate whenever possible and the possible tradeoff between end to end cell delay and overall throughput, of the 3 hop traffic.

Figure 10: Network model used in simulator

4 Results and Discussions

4.1 Training Results of Neuro-Fuzzy Controller

Implemented ANFIS trainer has been used to train the adaptive network with training data shown in Figure 3. The training data has been selected in conformance with the ideology of rulebase of neuro-fuzzy Controller (see Figure 2), i.e. to allow maximum flow whenever it is possible. ANFIS has been trained with 66 training data (\{Qlength, DeltaQ\}, \{FFR\}) for 1000 epochs, after which RMSE (Root Mean Square Error) value nearly becomes constant (refer Figure 11), which is because of the limitation of neural network. The output of ANFIS trainer is parameter-set in a text file (shown in Table 2), which is given as input to ANFIS driver. All the possible outputs corresponding to all the combination of
inputs have been generated and a control surface has been plotted as shown in Figure 12. The control surface reflects all the set of actions a controller can take in any circumstance i.e. with the possible set of inputs. Generated control surface of neuro-fuzzy controller nearly mimicked the control surface of fuzzy controller using FERM (refer Figure 13), which is already tuned manually according to the expected control actions. ANFIS trainer program is supplied with initial membership functions as shown in Figure 14 and Figure 15. These membership functions are tuned by the training program using ANFIS’s hybrid learning method. The tuned membership functions are shown in Figure 16 and Figure 17. Comparison with initial membership functions easily suggests the changes it has gone during training. Final parameters are listed in Table 2.

![Figure 11: Curve generated between RMSE versus no. of epochs, after training of neuro-fuzzy controller](image)

![Figure 12: Control surface of neuro-fuzzy controller using ANFIS](image)

![Figure 13: Control surface of fuzzy controller using FERM](image)
4.2 Simulation Results and Analysis

Both controllers, fuzzy and neuro-fuzzy controller, have been integrated with the switch as shown in Figure 6 and Figure 8 and this switch is included in the selected network (see Figure 10). Finally, simulations are carried out and network throughput and average cell delay are calculated at the terminal end i.e. at the BTE (Broadband Terminal Equipment). To calculate it, separate code has been embedded to the code of BTE and at the interspaces of 1000 cells, network throughput (Mb/sec) and average cell delay (msec) are logged and then graphs are plotted between average cell delay and network throughput as shown in Figure 18 and Figure 19. Similarly, percentage cell drop at switch S3 and switch S2 has been calculated.

Cell Delay versus Throughput
The offered load to the network has been increased gradually, with increase in
time to 150% of the link capacity. Therefore, the plotted graph shows the relation between cell-delay and network throughput as the load is increased from 20% to 150% of the link capacity. The plotted graphs for the two controllers are shown in Figure 18 and Figure 19. The results are given as follows:

Graph in Figure 18 clearly shows that the network throughput increases with the increase in offered load, till all the resources of the network don’t get exhausted. At this point it can be seen that throughput is nearly touching 115 Mb/sec mark, but after that switches cannot accommodate any more cells in their buffers, hence the overflow of cells is there, which leads to cell loss. Therefore, a decline in throughput and an increase in cell delay is observed at this stage.

Initially, rate of decrease of network throughput is more but the control actions of neuro-fuzzy controller tends to steady the fall in throughput and controls till long periods, and the throughput nearly remain constant for long period, but as the load increases, throughput falls again and controller steadied it around average throughput value of 60 Mb/sec.

At the end, there are a lot of vertical oscillations, these are basically because of the queuing effect within the switches and actions by the neuro-fuzzy controller to bring down the buffer level to the acceptable level. Cell delay increases with the increase in load because of increased queuing within network switches.

Plotted graph of neuro-fuzzy controller, shown in Figure 18, is nearly the same as the graph of manually tuned fuzzy controller, shown in Figure 19. These are in compliance with their control surfaces shown in Figure 12 and Figure 13 respectively. Control surface governs all set of actions to be taken by a particular controller. In actual, neuro-fuzzy controller has mimicked the manually tuned fuzzy controller.

Percentage Cell Drop

Percentage cell drop for the switches of the network has also been plotted with respect to time. This parameter suggests how severe is the congestion and how effective are the control actions taken by the fuzzy controllers to control congestion i.e. reduce cell loss.

Percentage Cell Drop at Switch S3

Graph of percentage cell drop with respect to simulation time for switch S3 is
shown in Figure 20. Nearly, at 80 msecs of simulation, queue length of the switch S3 overshoots the set maximum limit of 1024, which leads to cell loss. At this stage, the congestion has just start to happen. This cell loss keeps on increasing with time, as offered load to the network also increases. But due to the control actions of neuro-fuzzy controller, cell loss rate keeps on decreasing. It can be seen from Figure 20, the slope of graph keeps on decreasing (reflecting cell loss rate is decreasing). For switch S3, this curve is very smooth because of the presence of higher order background traffic, which is because of the fact, that, neuro-fuzzy controller will not be able to act upon ABR queue unless higher order traffic is serviced. Therefore it takes time for the curve to bend i.e. rate of change of percentage cell drop to decrease, see Figure 20.

Percentage Cell Drop at Switch S2

For switch S2 in network (shown in Figure 10), graph has a similar logic as that of switch S3 but there are lot of oscillations as shown in Figure 21. These are basically because, traffic controllers here are able to act immediately, as there is no higher order traffic, and set proper ER (Explicit Rate) to the new incoming cells, which further convey these to the sources. Hence, it leads to the reduction of flow rate quickly. Therefore the cell dropping reduces substantially, so there is drop in percentage cell drop as shown in Figure 21. But because offered load increases, therefore cell dropping is there again and increases till the fresh set of action are taken by the neuro-fuzzy controller. These actions cause oscillations at switch S2. From the Figure 21, it is easily visible that, at the end of simulation, neuro-fuzzy controller was able to reduce the cell loss at switch S2. At switch S1, no congestion found throughout the simulation period.

5. Conclusion

Traffic control remains a critical issue and a high priority, especially given the growing size, demand, and speed of the increasingly integrated services network. A large number of different schemes have been proposed for traffic control in ATM ABR
service. In this paper, an adaptive neuro-fuzzy rate control scheme based on ANFIS (Adaptive Network Based Fuzzy Inference System) has been proposed for the Available Bit Rate (ABR) service class in an ATM network and compared with the fuzzy rate control scheme FERM (Fuzzy Explicit Rate Marking). ANFIS has been used to fine tune the membership functions adaptively. This neuro-fuzzy system provides a better alternative way to design a controller. It adjusts the parameters at the database level according to the desired responses, supplied as a training data, which otherwise we would have done manually. The control actions of neuro-fuzzy controller nearly mimicked the manually tuned fuzzy controller.

References