DYNAMIC ATM TRAFFIC CONTROL USING
FEEDBACK AND TRAFFIC PREDICTION

Technical Report CSE-93-05

Yu Gong

Department of Computer Science and Engineering
Auburn University
Auburn University, AL  36849-5347

March 15, 1993
Dynamic ATM Traffic Control Using Feedback and Traffic Prediction

Yu Gong
Department of Computer Science and Engineering
Auburn University
Auburn, AL 36849

gong@eng.auburn.edu, Tel: (205) 844-6316, Fax: (205) 844-6329

Abstract

This paper focuses on the congestion problem arising in broadband wide area networks using asynchronous transfer mode (ATM). The ATM replaces the conventional window-based flow and congestion control by an open-loop mechanism, consisting of a connection admission control (CAC) and a usage parameter control (UPC). This so-called preventive congestion control (PCC) principle does not suffice since in reality effective CAC and UPC are both hard to achieve. Recently, the feasibility of using feedback to assist traffic controller is investigated. It has been found that feedback is useful in the long run to alleviate congestion. This general conclusion, although theoretically important, does not address the practical concern as to how the network behaves when it is adjusting itself. In practice, it is desirable that the mid-term, even short term congestion be controlled using certain type of feedback. The work reported in this paper is motivated by this practical concern. A new feedback based dynamic traffic control mechanism (called balanced mechanism) is proposed and compared with the pure PCC and an existing feedback based mechanism, known as EFCN. It is shown that the balanced mechanism outperforms both PCC and the EFCN.

Technical Report No. 93-05

Key Words: ATM, congestion control, closed-loop congestion control, EFCN, feedback.
1 Introduction

Due to the unscalable property of end-to-end propagation delay (governed by the speed of light), high-speed wide area networks (WANs) have a large amount of in-transit data over communication links [1]. For example, assume link speed is 622 Mbps, \(^1\) the end-to-end propagation delay is 23.5 milliseconds, \(^2\) and each packet (called a cell) consists of 53 bytes, \(^3\) the maximum number of cells between a source and a destination can be as high as 34,474. The large amount of in-transit traffic determines that the classic window-based congestion and flow control mechanisms [3, 4] to be ineffective. (In this paper, we refer to “congestion and flow control” as “traffic control”.) On this issue, Kleinrock [5] has good explanations.

To cope with the dilemma associated with the large delay-bandwidth product, the CCITT standard recommendation for asynchronous transfer mode (ATM) [2] selects a preventive congestion control (PCC) principle. Under this principle, traffic controllers are placed at the network boundary. Two mutually dependent functions, called connection admission control (CAC) and usage parameter control (UPC), are defined. The CAC is used during the connection setup phase to determine if the Quality of Service (QoS) requirement for a new traffic can be met. Once a connection is set up, the UPC enforces the traffic behavior to adhere with its target parameters, such as peak rate, average rate, burstness, etc.

If both CAC and UPC were accurate, the PCC principle would be sufficient to solve the ATM traffic control problem. In reality, however, the contrary is often true. Kurose [6] surveys existing analytical methods for the design of CAC functions. He classifies them into four categories: tightly controlled, approximate, bounding, and observation-based approaches, and shows that each of these approaches has problems. As such, the CAC can at best “coarse-tune” network performance. During the communication phase, the UPC must “fine-tune” the QoS. Previous research shows that this fine-tuning is also difficult to achieve [7, 8, 9, 10]. Therefore, the original PCC method leaves loopholes to possible uncontrolled congestion, reflected by failure to guarantee the QoS requirements.

Mukherjee et. al. [11] and Fendick et. al. [12], independently proved that feedback from network can be used to adjust source rates toward asymptotic (desirable) levels. As expected, the convergence requires a long transient delay. They considered binary congestion-noncongestion type feedback, which is first proposed in 1986 by Ramakrishnan et. al. [13] for the DECBIT protocol. This binary feedback scheme, known as explicit forward congestion notification (EFCN), is now adopted by the CCITT working group XVIII into the ATM standard recommendation.

As the field progresses toward deployment of the ATM technique, many practical issues become increasingly urgent, such as “what is the transient performance of EFCN?”; “is the binary

---

\(^1\)This transmission speed is specified in the CCITT standard recommendation [2] as one of network-network interface characteristics for ATM.

\(^2\)This is approximately the signal propagation delay across the continental U.S. on fiber-optic transmission media.

\(^3\)This is the ATM cell length as specified in [2].
feedback sufficient?”, and “what is the best way for adjusting source rate in response to feedback?”, etc. In this paper, we focus on these more practical issues. A new feedback-based traffic control mechanism (referred to as the balanced mechanism) is proposed and compared with pure PCC without feedback and the EFCN mechanisms. In the comparisons, transient behaviors during the periods of load changes are observed. We show that the balanced mechanism outperforms both pure PCC and the PCC with EFCN.

The paper is organized as follows. In the next section, Section 2, we review the previous work and discuss the motivation behind our new feedback-based method. In Section 3, this new scheme is described, together with the rationale behind all design choices. The performance evaluation is conducted through computer simulation. The simulation model is described in Section 4, and the numerical results are discussed in Section 5. We conclude this paper in Section 6.

2 Previous Work and Motivations

2.1 The Inherent Problem with Pure PCC

Butto et. al. [7] and Rathgeb [8] analyzed many existing UPCs, including Leaky Bucket [14], Jumping Window [8], Moving Window [8], and Exponentially Weighted Moving Average [8]. They concluded that these functions can be used to effectively enforce the peak rate parameter, but have problems in policing the average rate. Borgonovo and Fratta [10] explained that the reason lies in the latency for the controller to measure the traffic parameter. One possible solution to this latency problem is by the use of a buffer and a scheduler, altogether called a traffic shaper [15]. The use of shapers makes the traffic controller less dependent on the current usage parameters, at the expense of excessive cell delays. In general, using shapers is not a good idea for policing real-time traffic, such as interactive voice and video.

As Kurose pointed out in [6], how the traffic pattern changes within the network and what this change contributes to QoS are still unknown. With these uncertainties, most traffic control methods tend to be conservative, leaving a large portion of network power untapped. In this case, using network feedback becomes a viable solution approach to not only enhance network utilization but also maintain the QoS.

2.2 The Inherent Problem with Feedback

The most salient problem with feedback in a WAN is its long delay. In case the network condition does not change drastically during the delay, feedback may still be useful. Many protocols achieve this “elasticity” through very conservative approaches.

In the 4.3+ BSD TCP/IP protocol, [16], also known as Slow Start, a traffic controller uses an implicit congestion feedback to dynamically adjust the window size: if the acknowledgement is significantly delayed, the source times out and immediately reduces the window size to 1, thus
allowing only one outstanding packet. After a period of no congestion, the window size is increased by 1 each time. The effectiveness of the Slow Start protocol is evaluated by Jacobsen [16] and Keshav [17]. Its obvious conservativeness is justifiable since this protocol is to be used mainly on the existing networks with low speed, such as the Internet. The packet size on the Internet is also much larger than the cell size in the future ATM-based WANs.

In the less conservative DECBIT protocol [13], source rate is reduced by 50% upon reception of a congestion feedback. When no congestion happens, the source rate is allowed to increase, each time by a small constant. In this paper, as in many others, the rate adjustment method in the DECBIT protocol is referred to as MDAI, meaning "multiplicative decrease, additive increase". Note that the MDAI acts much faster in reducing load than increasing it. Thus, it is still a quite conservative method.

2.3 Definition of Feedback Delay

In the Slow Start protocol, reckoning the amount of feedback delay is impossible. This, however, is relatively straightforward, in the DECBIT protocol.

The DECBIT protocol uses explicit feedback, i.e., the EFCN. In this scheme, a switch along a virtual connection generates congestion notification when its queue length passes a predetermined threshold. The notification is written into a bit field in every cell passing through the switch. Therefore, it is the destination node that receives the congestion notification. It then relays the information to the source in the form of a signal (the original design in DECBIT used piggybacking).

Definition 1 (The Logical Generation Time (LGT) of Feedback): We define the logical generation time of feedback as the earliest time after which cells from a source cannot possibly affect the value of the feedback.

Suppose feedback A is received by a traffic source, it reflects the network condition caused by the source's cells sent before LGT(A). In other words, feedback A does not convey the network condition since LGT(A).

Definition 2 (The Physical Generation Time (PGT) of Feedback): The physical generation time of feedback is the time when the feedback physically appears in the network, and begins its journey to the source.

By introducing this last definition, we wish to clarify the fact that even before feedback is physically generated, it may have been delayed already. This latency is primarily due to the overhead involved for switches to detect the congestion, and in the case of EFCN, the latency for the congestion notification to reach the destination node.

To clearly illustrate the relationship between LGT and PGT, we show the source and destination time axes in Figure 1. The feedback scheme is assumed to be the EFCN. In Figure 1, three important reference points are identified.
1. At $t_s$, a tagged cell that will later carry an explicit congestion notification is sent into the network.

2. At $t_d$, the tagged cell arrives at the destination and a signal is sent to the source.

3. At $t_f$, the feedback arrives at the source.

The shaded area in Figure 1 corresponds to propagation of cells being sent before the tagged cell.

A straightforward observation is that those cells transmitted before the tagged cell affect the network condition, and cause the congestion notification to be (later) written into the tagged cell. In fact, at time $t_s$ when the tagged cell is being transmitted, the network condition has already been affected since all previous cells have entered the network. Therefore, the feedback should be considered already logically formed at time $t_s$ even though no physical feedback can be generated before $t_d$ ($t_d > t_s$).

The discussion so far indicates that the feedback delay, referred to as the age of feedback, should be defined as follows.

**Definition 3 (The Age of Feedback):** The age of feedback is the difference between the current time and the LGT of the feedback.

Clearly, the age of feedback in EFCN is a constant, equal to a round-trip delay, i.e., $t_f - t_s$.

### 2.3.1 Possible Changes of QoS during the Age of Feedback

Next, we consider possible changes to the QoS during the interval $(t_s, t_f)$ in Figure 1. Since the feedback does not convey information regarding what happens during its age, factors contributing
to the changes must be additionally considered in the rate adjustment mechanism. Three most important factors are as follows.

1. Interfering cells from other connections.

2. The source rate.

3. The transmission rate, i.e., the rate for the cells to pass the UPC and enter the network.

Unfortunately, the knowledge about interfering cells from other connections is theoretically impossible to obtain, although information regarding the other two factors can be collected locally. Practically, however, good guesses about the interfering traffic can be used. This way, the rate adjustment functions are built upon heuristics.

Here we mention that there is an important reason for distinguishing between the source rate and the transmission rate. The difference between the two corresponds to the cell loss rate at UPC (see Figure 1). This penalty must not be neglected in rate adjustment since it a part of the QoS over the entire virtual connection.

3 A Feedback-Based Traffic Control Mechanism

A traffic controller is, in fact, a rate tuner. Under the pure PCC principle, it monitors the amount of traffic that enters the network based only upon its locally collected information. In the case of feedback-based schemes, the feedback, obtained remotely, is also used in the rate adjustment process. In this section, we explain the two important aspects of a feedback-based mechanism, namely, the feedback and the rate adjustment function.

3.1 The Feedback

Different from the EFCN which uses binary feedback, the traffic controller proposed in this paper uses continuous feedback to be defined momentarily. A special dedicated cell, called feedback cell (FC) is issued periodically by the controller in order to probe the condition along a virtual connection. This FC corresponds to the tagged cell in the last section. As explained there, the LGT for the feedback in an FC is the time when the FC is being transmitted. This conclusion is helpful to the rate adjustment function because the age of feedback can be tracked quite easily.

Naturally, for timely adjustment of source rate in response to changes in network condition, one would like to issue a large number of FCs to bring in more feedback information. However, we expect multiple FCs would introduce a large amount of overhead, thus decrease the effectiveness of the mechanism. The overhead includes processing overhead within the controller, multiplexers, and switches to handle the multiple FCs, and the bandwidth overhead allocated to FCs. Perhaps the more serious problem with multiple FCs is that they tend to cluster together over a period of time.
In this case, an additional function within the controller is needed to detect the FC clustering, and spread them apart. Again, an overhead has to be introduced.

Because of the potential problems with multiple FCs, in this paper, only one FC is used. The controller works in a cyclic manner as follows.

1. Issue an FC.
2. Wait for the feedback.
3. Receive feedback and adjust rate.
4. Go to 1.

The feedback is the maximum switch utilization along a virtual connection. To obtain this information, the FC is initialized to 0 when it is being transmitted. Upon receiving an FC, a switch compares its own utilization with the value already stored in the FC. If the former is larger, the FC is updated with the new value. Here the switch utilization is defined as the ratio of switch’s queue length to the queue capacity.

Depending upon how much processing capacity a switch has, the utilization can be either computed instantly upon reception of the FC (introducing the lowest overhead but largest variation), or be averaged over a period of time (requiring more processing power in switches but is more accurate). In this paper, the latter method is assumed: we assume all switches are capable of continuously observing its queue utilization over each controller cycle. Because of the averaging effect, the possibility for erroneous feedback is minimized.

3.2 Rate Adjustment

Figure 2 shows the block diagram for the internal of the controller. The controller consists of two Leaky Bucket UPCs arranged in tandem, the first one is for policing the average rate and the second one for the peak rate. For a cell to enter the network, it must pass both policers.

Each of the policers has a token pool whose content (called tokens) is generated at a constant rate. When a cell is generated, it consumes one token. In case the token pool is empty, the cell is dropped. As shown in Figure 2, token rates for the two policers are $r_a$ and $r_p$, respectively. Moreover, both policers have finite token pool sizes, denoted by $K_a$ and $K_p$.

Four parameters (i.e., $r_a$, $r_p$, $K_a$, and $K_p$) can be used to monitor traffic submitted into the network. The effectiveness of all the parameters is not the same. First, Rathgeb [8, Figures 3 and 4] showed that changing token pool size has lesser impact on the cell loss probability than modifying token rate. Therefore, in our work, we assume both pool sizes are fixed. Second, as concluded in earlier research [7, 8], the peak rate parameter can be quite accurately enforced using the LB policer. Therefore, in our work, we do not allow the token rate $r_p$ in the peak rate policer to change. (The $r_p$ should be set to the target peak rate according to [7, 8].)
As the result, only one parameter (the token rate $r_a$ in the average rate policer) remains, which can be modified based on the feedback. We assume that the range for modifying $r_a$ is $[r_m, r_M]$. A method for tuning $r_a$ within this interval is described in the remainder of this subsection.

Consider the time axis as shown in Figure 3. Time $\tau_i$ ($i = 1, 2, \ldots$) in Figure 3 is the moment when the $i$th FC enters the network. The interval $[\tau_i, \tau_{i+1}]$ is referred to as the $i$th cycle. The following notations are defined for the $i$th cycle.

- $\tau_i$: the token rate in the average rate policer during the $i$th cycle. Note that, for the notational simplicity, here we drop the subscript “a” as we used earlier.
- $\eta_i$: the feedback available at the end of the $i$th cycle.
- $s_i$: the average source rate during the $i$th cycle.
\( t_i \) : the average transmission rate during the \( i \)th cycle.

At the beginning of the \( i \)th cycle (time \( \tau_i \) in Figure 3), five pieces of information regarding what happened in the previous two cycles are used to estimate the token rate \( \tau_i \) for cycle \( i \). These five performance measures are \( t_{i-2} \) from cycle \( i - 2 \) and \( \tau_{i-1} \), \( \eta_{i-1} \), \( s_{i-1} \), and \( t_{i-1} \) from cycle \( i - 1 \). They are surrounded by circles in Figure 3.

The estimate for \( \tau_i \) is denoted by \( \hat{\tau}_i \). It is subject to the following restriction.

\[
\tau_m \leq \hat{\tau}_i \leq \tau_M
\]

The estimation process proceeds by first estimating, for cycle \( i \), the cell rates \( \hat{s}_i \) from the source and \( \hat{t}_i \) into the network. The minimum for the two estimates is taken as \( \hat{\tau}_i \), i.e.,

\[
\hat{\tau}_i = \min\{\hat{s}_i, \hat{t}_i\}
\]

Note that the real rates \( s_i \) and \( t_i \) always satisfy \( s_i \geq t_i \), but this relation does not necessarily hold for their estimates.

The rationale behind equation (2) is that \( \hat{s}_i \) and \( \hat{t}_i \) are computed below with conflicting QoS goals: \( \hat{s}_i \) is estimated for the purpose of minimizing the source cell loss probability, while \( \hat{t}_i \) is computed to satisfy QoS requirement within the network.

To achieve the smallest cell loss probability, \( \hat{s}_i \) must be close to the real average rate. The following estimation is reasonable.

\[
\hat{s}_i = s_{i-1}
\]

This estimation is based upon the fact that the length of each cycle is long enough so that the measured rate \( s_{i-1} \) is very close to the source's average rate.

The target network QoS is difficult to define because it is related to all connections in a complicated manner. Therefore, we propose an addition to the ATM admission control: when a user requests a virtual connection, a target VC utilization (denoted by \( \eta^* \)) is negotiated between the user and the network. During the communication phase, the traffic controller compares the feedback with \( \eta^* \) to determine how badly the VC is congested. This way, the traffic controller does not have to observe the QoS within the network, which can be difficult to obtain.

To compute \( \hat{t}_i \), several cases must be distinguished (see Figure 3).

Case 1 (\( \eta_{i-1} > \eta^* \)): In this case, the network is considered being congested in cycle \( i - 1 \). (Note that the congestion condition had already been generated at time \( \tau_{i-1} \), the LGT of the feedback \( \eta_{i-1} \).) Therefore, in cycle \( i \), the token rate \( \tau_i \) must be reduced. The extent of this rate deduction is dependent upon whether the source has already submitted less traffic in the previous cycle. Two subcases are further distinguished below.

Case 1.a (\( t_{i-1} < t_{i-2} \)): In this case, the source has already submitted lighter traffic to the network during the age of the feedback. Therefore, \( \hat{t}_i \) can be estimated using an optimistic
Figure 4: The Optimistic Rate Adjustment Function

\[
\hat{t}_i = \min\{t_{i-1}, f_{\text{opt}}[\eta_i-1, \phi_1(\eta_{i-1})]\}
\]

where

\[
f_{\text{opt}}(\eta, y) = (r_m - r_M) \frac{\log(y + 1 - y \eta)}{\log(y + 1)} + r_m
\]

and

\[
\phi_1(\eta) = (C_M - C_m) \frac{1 - \eta}{1 - \eta^*} + C_m
\]

In equation (6), \(C_m\) and \(C_M\) (where \(C_m \leq C_M\)) are two parameters explained shortly. The key to this estimation is the heuristic function (5). This function is plotted in Figure 4 with various values of \(y\). Figure 4 shows that using this function, the estimated transmission rate does not decrease drastically when \(\eta\) increases. So, we call this function an **optimistic function**.

Figure 4 also shows that the larger the value of \(y\), the more slowly the function decreases with increasing \(\eta\); therefore, more optimism is built into the function. For this reason, we call the argument \(y\) the **degree of confidence** for the optimism. In our mechanism, this degree of confidence also changes with the feedback \(\eta\) in an interval \([C_m, C_M]\), as shown in (6).

**Case 1.b \((t_{i-1} \geq t_{i-2})\):** In this case, more traffic was submitted into the network. Most likely, the congestion condition is not changed at time \(\tau_i\). Therefore, \(\hat{t}_i\) is computed using a conservative function.

\[
\hat{t}_i = \min\{t_{i-1}, f_{\text{con}}[\eta_{i-1}, \theta_1(\eta_{i-1})]\}
\]
where

\[ f_{\text{con}}(\eta, y) = \frac{1}{1-e^{-y}} \left[ (\eta_M - \eta_m) e^{-y\eta_M} + \eta_m - \eta_M e^{-y} \right] \]  

(8)

and

\[ \theta_1(\eta) = (C_M - C_m) \frac{\eta - \eta^*}{1 - \eta^*} + C_m \]  

(9)

The function (8) is plotted in Figure 5. Note that using this estimating function, a small increase of \( \eta \) can incur a large rate deduction. Thus this function is called conservative function. Similar to Case 1.a, equation (9) provides the degree of confidence for the conservativeness in the interval \([C_m, C_M]\).

**Case 2 (\( \eta_{i-1} \leq \eta^* \))**: In this case, the network may not be congested at all in the \((i-1)\)th cycle. We further distinguish two subcases.

**Case 2.a \((t_{i-1} < t_{i-2})\)**: In this case, the source has submitted lighter traffic to the network during the age of the feedback; therefore, \( \hat{t}_i \) is computed using the following optimistic function.

\[ \hat{t}_i = f_{\text{opt}}[\eta_{i-1}, \phi_2(\eta_{i-1})] \]  

(10)

where

\[ \phi_2(\eta) = (C_M - C_m) \frac{\eta - \eta^*}{\eta^*} + C_M \]  

(11)

Different from equation (4) in Case 1.a, the estimated rate in (10) is not subject to the upper bound \( t_{i-1} \). Thus, the token rate may increase for the next cycle, which is not allowed in Case 1.a. Furthermore, the degree of confidence is also computed differently.
Case 2.b (\(t_{i-1} \geq t_{i-2}\)): In this case, the transmission rate in cycle \(i - 1\) has been increased. However, the effect of this rate increase is not known yet at time \(\tau_1\). Therefore, a conservative estimation is used in this case.

\[
\hat{t}_i = \min\{t_{i-1}, f_{\text{con}}[\eta_{i-1}, \theta_2(\eta_{i-1})]\}
\]

(12)

where

\[
\theta_2(\eta) = (C_M - C_m) \frac{\eta}{\eta^*} + C_m
\]

(13)

4 Simulation Model

Table 1 lists all the parameters used in the simulator. They are categorized based on the three types of important components: source, controller, and switches. The parameters related to the controller have been explained in the last section. In this section, the parameters associated with the other components are described.

Table 1: Parameters in the Simulator

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>(\mu_0, \mu_1, \lambda_0, \lambda_1)</td>
</tr>
<tr>
<td>Controller</td>
<td>(K_a, K_p, r_a \in [r_m, r_M], r_p, [C_m, C_M])</td>
</tr>
<tr>
<td>Switch (i)</td>
<td>(1 \leq i \leq n): (L_i, d_i, p_i, x_i(j))</td>
</tr>
</tbody>
</table>

The end-to-end virtual connection (VC) model is shown in Figure 6. The parameters in Figure 6 have the following meanings.
$n$: the total number of switches.

$L_i$: the capacity of the queue in switch $i$ along the virtual connection. The server of each queue has a constant service rate of one cell per unit time, and it follows the FCFS discipline.

$d_i$: link propagation delay between switches $i$ and $i + 1$, $i = 1, 2, \ldots, n$. (Switch $n + 1$ is the destination node.)

$x_i(j)$: the probability of cross traffic cells' interarriving distance to be $j$ in switch $i$, i.e., the cross traffic intercell distance is assumed to follow a discrete general distribution.

$p_i$: cell deflecting probability after switch $i$, i.e., a cell in switch $i$, after having been served, will not stay on the virtual connection with probability $p_i$. In the simulation, these deflecting cells are discarded.

The traffic source is modeled by a two state MMPP as shown in Figure 7. This model has been shown to be versatile in approximating many types of real sources [18]. In the MMPP model, a two-state continuous-time Markov chain is defined. The state transition rates are $\mu_0$ and $\mu_1$. In state 0 (1), cell arrivals follow the Poisson distribution with rate $\lambda_0$ ($\lambda_1$).

The peak rate and the average rate of the MMPP model can easily be computed. The peak rate $s_p$ is simply the larger of $\lambda_0$ and $\lambda_1$, i.e.,

$$s_p = \max\{\lambda_0, \lambda_1\}$$

and the average rate $s_a$ is given by

$$s_a = \frac{\mu_1}{\mu_0 + \mu_1}\lambda_0 + \frac{\mu_0}{\mu_0 + \mu_1}\lambda_1$$

5 Results and Discussion

Three traffic control methods, the pure PCC, the EF-CN with MDAI, and the balanced mechanism proposed in this paper, are compared in five scenarios. In the first three scenarios, the virtual
connection is persistently congested, moderately congested, and not congested at all, respectively. Furthermore, loads to all switches do not change in these three scenarios. In the last two scenarios, the network condition changes at the end of the fifth round-trip time from no congestion condition to heavy congestion (in scenario 4) or from heavy congestion condition to no congestion (in scenario 5).

Scenario 1: a heavily congested virtual connection

Some of the parameters selected for this scenario are shown in Tables 2. Four switches are assumed in this case; all have identical distributions (not shown in Table 2) for interarriving time of cross traffic cells. The expected cross traffic intercell distance is 0.192. Therefore, the cross traffic load on each switch is \(1/(1 + 0.192) = 0.84\), a somewhat heavy load. The discrete general distribution also allows many cross traffic cells to arrive at the same time, i.e., a batch arrivals model. The batch size follows a geometric distribution with mean 4.76. Note that in Table 2, the buffer size in each switch is 5. As such, buffers in all switches may become saturated from time to time, generating potential congestion.

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>(\mu_0 = 0.4, \mu_1 = 0.4, \lambda_0 = 0.333, \lambda_1 = 0.067)</td>
</tr>
<tr>
<td>Controller Peak Rate</td>
<td>(K_p = 10, r_p = 0.333)</td>
</tr>
<tr>
<td>Average Rate</td>
<td>(K_a = 100, r_a \in [0.05, 0.25])</td>
</tr>
<tr>
<td>Switches ((n = 4))</td>
<td></td>
</tr>
<tr>
<td>Switch 1</td>
<td>(L_1 = 5, d_1 = 500, p_1 = 0.5)</td>
</tr>
<tr>
<td>Switch 2</td>
<td>(L_2 = 5, d_2 = 500, p_2 = 0.0)</td>
</tr>
<tr>
<td>Switch 3</td>
<td>(L_3 = 5, d_3 = 500, p_3 = 0.8)</td>
</tr>
<tr>
<td>Switch 4</td>
<td>(L_4 = 5, d_4 = 500, p_4 = 0.5)</td>
</tr>
</tbody>
</table>

In both EFCN and the balanced mechanisms, the threshold switch utilization \(\eta^*\) is targeted at 0.5. Because the cross traffic cells' deflecting probability in switch 2 \((p_2)\) is 0, the utilization of switch 3 becomes persistently higher than 0.5. In Figure 8, the utilization of switch 3 over the first 15 round-trip time is plotted. Note that this utilization is also the feedback in the balanced mechanism, since all other switches have utilizations much lower than that of switch 3. Figure 8 shows that the two mechanisms EFCN(MDAI) and the balanced mechanisms do not differ significantly in terms of the maximum switch utilization: the congestion persists no matter which mechanism is being used in this case.

However, other performance measures show that although the balanced mechanism cannot eliminate the persistent congestion, it is able to improve the QoS for cells both from the virtual connection (called VC cells) and from the cross traffic (called CT cells). This conclusion is sum-
marized from the results shown in Figures 9–12 where the VC cell loss probability, CT cell loss probability, VC cell delay, and CT cell delay versus switch index, respectively, are plotted.

In these four Figures, the initial token rate in the average rate policer \( r_\alpha \) is selected as 0.167, slightly lower than the average source rate 0.2. In light of the existing congestion along the VC, this initial token rate may not be appropriate, i.e., it may need to be further lowered. In the case of pure PCC, the token rate, once selected, cannot be changed during the entire communication phase, while in the other two mechanisms, changes can be made based on the feedback. The balanced mechanism clearly shows its advantages in Figures 9–12 on both cell loss and delay performance for both VC and CT cells. Our explanation for this phenomenon, as well as for many other cases, is that the balanced mechanism more closely follows the statistic dynamics of the network condition. Cells injected into the network using this mechanism are less likely to be dropped than in the EFCN(MDAI) case.

**Scenario 2: a moderately congested virtual connection**

In this scenario, three changes are made on the parameter settings of Scenario 1. First, we change the target switch utilization \( \eta^* \) from 0.5 to 0.8. Second, the distance between adjacent switches is increased from 500 time units to 1000; therefore, links can function better as cell buffers. Third, the deflection probability in switch 2 is now changed from 0 to 0.8; thus only 20% cross traffic cells served by switch 2 can actually arrive at switch 3. Except for these changes, all other parameters are the same as in Scenario 1. As the result of these modifications, the virtual connection become less congested.

Numeric results show that the switch having the highest utilization is still switch 3. The change of its utilization over time is plotted in Figure 13. Figure 13 shows that, in general, the balanced mechanism generates slightly higher utilization than the EFCN(MDAI). As the result, the balanced mechanism may have poorer QoS due to the potential congestion.

However, Figures 14–17 show that the contrary is true: the balanced mechanism gives rise to much better cell loss and delay performance than the EFCN(MDAI) for both VC and CT cells. It is also better than the pure PCC.

**Scenario 3: no congestion along the virtual connection**

We now change the parameter settings of Scenario 2 to make the VC even less congested, i.e., we double the buffer capacity in each switch to change it from 5 to 10. All other parameters are the same as in Scenario 2.

As one may expect, the virtual connection is extremely lightly loaded. Under the congestion-free condition, the optimal token rate in the average rate policer is 0.2, which is the average rate of the source. The token rate \( r_\alpha = 0.167 \) is selected for this scenario. It is close to the optimal rate.
Also as expected, Figures 18–21 show that the pure PCC mechanism performs the best in terms of maintaining QoS in the network. However, the balanced mechanism is not much worse than the pure PCC, while the EFCN(MDAI) is.

However, the pure PCC can outperform the feedback based mechanisms, only under the condition that the parameter settings in the controller is selected appropriately. In reality, to accurately configure the policers is often difficult to achieve. As such, a feedback mechanism that performs similarly with the optimal pure PCC would be attractive. The balanced mechanism appears to be such a choice.

**Scenario 4: from no congestion to heavy congestion**

To test the responsiveness of the feedback-based mechanisms, we allow switch 2's deflection probability to change from 0.8 to 0 at the end of the fifth round-trip time. The initial token rate $r_a$ in the average rate policer is 0.1. All other parameters are the same as in Scenario 2.

In the pure PCC mechanism, the pessimistic token rate ($r_a = 0.1$) leads to unnecessary cell loss at the policer during the first five round-trips when no congestion exists along the VC; but it protects the switches during the period of congestion after the fifth round-trip. As the result, the pure PCC mechanism still performs the best in terms of protecting QoS in switches, as shown in Figures 22–25. This protection, however, is at the expense of the QoS over the entire virtual connection, as shown in Table 3. Table 3 shows that the PCC offers the best protection of switches, and the EFCN(MDAI) offers lenient control at the controllers but poor performance in switches. The balanced mechanism balances the QoS in the two types of components.

<table>
<thead>
<tr>
<th>MECHANISM</th>
<th>CELL LOSS PROBABILITY</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P.R. Policer</td>
<td>A.R. Policer</td>
<td>Switches</td>
<td>Total Cell Loss along VC</td>
</tr>
<tr>
<td>PCC</td>
<td>0</td>
<td>0.492</td>
<td>0.033</td>
<td>0.525</td>
</tr>
<tr>
<td>EFCN(MDAI)</td>
<td>0</td>
<td>0.345</td>
<td>0.086</td>
<td>0.431</td>
</tr>
<tr>
<td>Balanced</td>
<td>0</td>
<td>0.465</td>
<td>0.046</td>
<td>0.511</td>
</tr>
</tbody>
</table>

Finally, the change of the token rate $r_a$ in the two feedback-based mechanisms in the first 15 round-trip time is plotted in Figure 26. This figure shows that when the VC is not congested, the token rate in the balanced mechanism increases faster towards the optimal value 0.2 than the EFCN(MDAI). The increase also stops precisely at the optimal point, while the token rate in the EFCN(MDAI) still increases unnecessarily after the optimal point has been passed. When the VC begins to be congested at the end of the 5th round trip, the token rate in the balanced mechanism is promptly reduced to 0.06 after 2 more round-trip time, while the EFCN(MDAI) is not able to do so until the end of the 11th round-trip. Starting from the end of the 11th round-trip, the token
rate in the balanced mechanism is held fixed for the remainder of the connection. However, the same rate in the EFCN(MDAI) slowly increases and shows no sign of convergence.

**Scenario 5: from heavy congestion to no congestion**

In this last scenario, we use the same parameter settings in Scenario 4, except that the deflection probability of switch 2 changes from 0 to 0.8 at the end of the fifth round-trip time. Therefore, the virtual connection is heavily congested in the first five round-trip time, and then the congestion disappears.

The QoS performance follows the similar pattern as we have shown in Scenario 4. Thus we do not repeat the previous conclusion. Here we only plot the changes of the token rate \( r_a \) in Figure 27. This figure shows that when the network changes from heavy congestion to no congestion, the token rate in the average rate policer is able to increase to the vicinity of the optimal 0.2. However, the EFCN(MDAI) increases its token rate indefinitely until the permissible upper limit \( r_M \) is reached. This fact shows that the balanced mechanism can more accurately police the source rate.

**6 Conclusion**

In this paper a balanced feedback-based congestion control mechanism is proposed and evaluated through computer simulation. The design of this new mechanism is motivated by the observation that the existing traffic control mechanisms, such as the pure preventive congestion control (PCC) or the feedback-based explicit forward congestion notification with the so-called multiplicative decrease additive increase rate adjustment method (EFCN(MDAI)), do not fully make use of available information. The pure PCC does not follow the feedback at all, and the EFCN(MDAI) does not take feedback delay into consideration. The proposed mechanism put both (i.e., feedback and its delay called *feedback age*) in a synergistic whole, and balances the traffic control actions using an optimistic and a conservative functions.

The three mechanisms, the pure PCC, the EFCN(MDAI), and the balanced mechanism, are compared. The conclusions are as follows.

- The balanced mechanism is superior to the EFCN(MDAI) in protecting the QoS (including cell loss and delay) along a virtual connection. This protection is not only offered to cells belonging to the connection but also to cross traffic.

- The balanced mechanism also achieves lower cell loss probability than the EFCN(MDAI) over the entire virtual connection. This includes cells lost in both switches and policers.

- As the congestion condition changes within the network, the balanced mechanism is able to adjust the token rate in the LB policer to appropriate level both promptly and accurately.
The EFCN(MDAI) mechanism suffers from long latency, rate fluctuation, and inaccurate rate adjustment.

- In all scenarios examined in our work, we have found that there are optimal parameter settings that make the pure PCC outperform all other mechanisms. This supports the original concept of using the pure PCC as the foundation of ATM. The optimal settings, however, are closely related to the existing network condition, thus they are either difficult to obtain or subject to change. In reality, a feedback-based mechanism that is able to closely follow the change of the optimal setting is needed. We show in this paper that the balanced mechanism is such a mechanism.

References


Figure 8: Maximum Switch Utilization along VC (Scenario 1)

Figure 9: VC Cell Loss Probability along the Connection (Scenario 1)
**Figure 10:** CT Cell Loss Probability along the Connection (Scenario 1)

**Figure 11:** VC Cell Delay along the Connection (Scenario 1)
Figure 12: CT Cell Delay along the Connection (Scenario 1)

Figure 13: Maximum Switch Utilization along VC (Scenario 2)
Figure 14: VC Cell Loss Probability along the Connection (Scenario 2)

Figure 15: CT Cell Loss Probability along the Connection (Scenario 2)
Figure 16: VC Cell Delay along the Connection (Scenario 2)

Figure 17: CT Cell Delay along the Connection (Scenario 2)
Figure 18: VC Cell Loss Probability along the Connection (Scenario 3)

Figure 19: CT Cell Loss Probability along the Connection (Scenario 3)
Figure 20: VC Cell Delay along the Connection (Scenario 3)

Figure 21: CT Cell Delay along the Connection (Scenario 3)
Figure 22: VC Cell Loss Probability along the Connection (Scenario 4)

Figure 23: CT Cell Loss Probability along the Connection (Scenario 4)
Figure 24: VC Cell Delay along the Connection (Scenario 4)

Figure 25: CT Cell Delay along the Connection (Scenario 4)
Figure 26: Change of the Token Rate (Scenario 4)

Figure 27: Change of the Token Rate (Scenario 5)