A Framework for Statistical Multiplexing onto a Variable-Bit Rate Output Channel

Chan-weng Lai and Soung C. Liew
Department of Information Engineering, The Chinese University of Hong Kong

***Abstract*— This paper investigates soft multiplexing, an issue that arises in VP-based ATM networks. The main distinguishing feature of this form of multiplexing (as opposed to traditional statistical multiplexing) is that the output channel is a *variable-bit rate* VP. The concepts of work conservation and greediness were introduced to study soft multiplexing systematically. We assume that the overall multiplexed output traffic is regulated and policed by a Leaky Bucket. Before allowed to access the output channel, each input traffic stream is also controlled by a Leaky Bucket. It is shown that although strict performance guarantee can be provided to each input traffic stream by independent Leaky-Bucket operation, output-channel utilization can be rather low, and it can be improved significantly by if we allow some interactions among the Leaky Buckets of the input streams. In particular, we show that two simple soft-multiplexing schemes, excess token passing and token borrowing, can achieve higher output-channel utilization, better delay and loss performance without sacrificing guarantee to each session.**

Keywords— **ATM Networks, Multiplexing, Leaky Buckets, Flow Control, Packet Networks, VP/VC**

I. Introduction

The Asynchronous Transfer Mode (ATM) has been accepted as the mechanism for transporting information in Broadband Integrated Services Digital Networks (B-ISDN). Optical transmission links with transmission rate in the range of Mb/s to Gb/s are expected to be widely used in these networks. Many services (including compressed video), on the other hand. will only need from several kb/s to a few Mb/s of bandwidth. If a virtual circuit is used to carry each information stream, a physical link will then contain a huge number of simultaneous virtual circuits, as depicted in Fig. 1(a). It is prohibitive to have to monitor, control, and manage the traffic flows of thousands of virtual circuits on each link. To tackle this problem, the concept of virtual channels (VC) and virtual paths (VP) has been conceived [1, 2].

A VP is a collection of VC's to be transported and switched as a unit. The resulting network is structured into a hierarchy of three levels: VC, VP, and transmission levels [1, 2]. The VP-level network interconnects VC switching nodes, where VC's are multiplexed onto VP's. At VP switching nodes, VP's are in turn multiplexed onto physical links without the VC-level routing and control information being examined. By managing the network at the VP level, the VC level can concentrate on service handling.

As illustrated in Fig. 1(b), the two-layer arrangement presents a new multiplexing problem at the VP level. Specifically, unlike a physical link, a VP does not have a hard limit on the bit rate, and if it behooves the network to do so, the bit rate of the VP can be allowed to vary. One may to so, for instance, to deal with the dynamic changing of traffic conditions in the network. Thus, as far as the VC's contained in the VP are concerned, they are being multiplexed onto a variable-bit rate output channel. This will be referred to as *soft multiplexing* in this paper. Most existing multiplexing disciplines, such as Generalized Processor Sharing (GPS)[3, 4], Round Robin (RR)[5] and Stop-and-Go Queueing[6, 7, 8], assume fixed-rate output channel. Multiplexing onto a fixed-rate output channel will be called *hard multiplexing*.

In this paper, we assume that the input traffic of each VP (output traffic of a soft multiplexer) is monitored and controlled by a Leaky-Bucket[9, 10, 11]. We also assume that an appropriate hard multiplexing discipline (e.g. GPS multiplexing [3, 4]) will be used to multiplex VP's onto physical links in a way that guarantees performance on each VP provided the input traffic of each VP conforms to the pre-agreed Leaky-Bucket parameters. Thus, we shall only confine our attention to the performance issues within the soft multiplexer.

We explore several soft multiplexers that use a Leaky Bucket for the control of the traffic of each VC (each input to the soft multiplexer). In the simplest case, all these Leaky Buckets work independently to provide strong performance guarantee. The drawback is that the VP bandwidth can not be shared efficiently among the VC's. Two schemes are proposed to achieve efficient bandwidth sharing while maintaining a certain degree of performance guarantee to each VC. The first employs *excess token passing* in which excess tokens of a Leaky Bucket will be passed to others. The second employs *token borrowing* in which busy VC's can borrow tokens from others.

In Section II, some fundamental concepts of soft multiplexing are introduced. In Section III, a soft multiplexer with independent Leaky Buckets is considered. Sections

IV and V discuss the mechanisms and the effects of excess token passing and token borrowing. Finally, the advantages of the proposed mechanisms are verified by simulations in Section VI.

II. Fundamental Concepts of Soft Multiplexing

Let us first review hard multiplexing. A physical link with transmission rate C is either in idle state or busy state. It transmits at rate C in busy state and at zero rate in idle state. The unused bandwidth (transmission capacity) during the idle state cannot be saved for future usage. As shall be explained, in Leaky-Bucket-controlled soft multiplexing, the unused bandwidth can be saved (up to a certain degree) for later use.

The concept and operation of a Leaky Bucket is simple [9, 10, 11]. Each Leaky Bucket is associated with two parameters : the bucket size δ and the token generation rate ρ. Tokens are generated at a constant rate ρ, and a cell must acquire a token before it can be transmitted on the virtual connection. Unused tokens are accumulated in a bucket that can hold at most δ tokens. When the bucket fills up, any newly generated but unused token will be discarded. The maximum transmission rate of a Leaky Bucket can either be bounded or unbounded. In the simplest case, which is assumed in this paper, the maximum transmission rate is unbounded and is infinite when the bucket is not empty, and it is ρ when the bucket is empty. The practical meaning of the former is that the output rate can match the maximum aggregate input arrival rate when the bucket is not empty. By saving up tokens in the bucket, bandwidth usage can be deferred.

We say that a traffic with rate function $R(t)$ satisfies the burstiness constraint of a virtual connection (imposed by the Leaky Bucket), or according to Cruz's notation[13, 14], $R \sim (\delta, \rho)$ if

$$\int_x^y R(t)dt \leq \delta + \rho(y - x) \qquad (1)$$

for all x, y satisfying $y \geq x$. It can be easily seen that traffic with $R(t)$ satisfying (1) can be immediately transmitted when regulated by a Leaky Bucket with parameters δ and ρ. Also, note that $O \sim (\delta, \rho)$, where O is the output traffic of a Leaky Bucket with parameter δ and ρ.

Multiplexing systems are usually described using the terminology of queueing theory. Arriving packets are considered as work load arrivals and the multiplexer is a server serving such work load. A server is said to be *work conserving* if it is busy (i.e. transmitting packets) whenever there are packets waiting in the system[12]. This definition of work conservation has an underlying assumption that the server only serves at a constant rate(i.e. the transmission rate is fixed) and such servers, named *hard servers* in this paper, are either in idle state or busy state. However, when the output channel is Leaky-Bucket controlled, the server should be a *soft server* which is capable of serving at variable rate.

The definition of work conservation needs to be reexamined in a soft server. The basic idea of work conservation is not to waste server capacity when there is demand for service. In a Leaky-Bucket controlled system, server capacity can be saved up as tokens in the bucket and used later. Thus, unless the bucket overflows, no server capacity is wasted. Thus, a soft server is said to be *work conserving* if *no token will be wasted whenever there are packets waiting in the system.* This definition reduces to work conservation of a hard server when the bucket size is 0. Table 1 illustrates the service rate, S, of a work conserving soft server under different states when there are packets waiting. Note that the service rate S should not be confused with the maximum transmission rate of the VP, which is the upper bound on the service rate. In this paper, we assume for simplicity that the maximum transmission rate is infinite when the bucket is not empty, and it is the token generation rate ρ when the bucket is empty. We have the freedom to choose S, and the server is work-conserving if the choices conform to the Table 1's specification.

Table 1.

state	bucket state	service rate
1	full	$S \geq \rho$
2	not full & not empty	$S =$ any rate
3	empty	$S \leq \rho$

We see that in states 2 and 3, no token will be wasted even if $S < \rho$ because the unused tokens can be accumulated in the non-full bucket. In state 1, if there are packets waiting, no token will be wasted because $S \geq \rho$, and bucket overflow is not possible. Tokens will only be wasted when there is no backlog and the bucket is full, but this is unavoidable.

Another important concept is that of a *greedy* soft server. A soft server is said to be greedy if *the server will serve at infinite rate when there are tokens in the bucket and at token generation rate ρ when the bucket is empty.* In other words, the server always serves at maximum transmission rate of the VP. Table 2 lists the service rate, S, of a greedy soft server under different states. An arrival will consume a token immediately if one is available, and co-existence of tokens and cell backlog is impossible. Thus a greedy server minimizes the average delay. Note that given the same input traffic patterns, all greedy systems have identical average queue length and waiting times, regardless of the order in which the packets are served. According to the above definitions, a greedy soft server is also work conserving because the bucket will never be full (i.e. no token will be wasted) while some cells are waiting. In addition,

when the bucket size is 0, work conserving and greedy servers are the same. Since a physical link can be considered as a virtual connection with zero bucket size, work conservation and greediness for a physical link server are therefore also the same.

Table 2.

state	bucket state	service rate
1	full	$S = \infty$
2	not full & not empty	$S = \infty$
3	empty	$S = \rho$

For a multiplexer whose output traffic is policed by a Leaky Bucket before it enters the VP, cell delay and loss can be separated into three parts: those in the multiplexer, those in the VP Leaky Bucket and those in the VP. As mentioned earlier, by forcing the aggregate output traffic A to conform the burstiness constraint of the VP (i.e. $A \sim (\delta, \rho)$), there will not be any loss and delay due to the second part and the third part is also bounded. In this paper, we only examines multiplexer whose output conforms to the constraint. Therefore, an efficient soft multiplexer should aim at minimizing the first part. In addition, a certain degree of fairness and performance guarantee must be maintained among the inputs being multiplexed.

In summary, a good soft multiplexer should achieve the following goals: 1) $A \sim (\delta, \rho)$; 2) low loss and delay in multiplexer; 3) high utilization of output capacity; 4) guaranteed performance of each session. Work conservation and/or greediness can be imposed to achieve 1), 2) and 3), while individual Leaky Buckets for regulating the inputs (see next section) can be used to achieve 4). As will be seen later, there are tradeoffs among these goals.

III. Soft Multiplexer with Independent Leaky Buckets

Fig. 2 shows the structure of a soft multiplexer which consists of a number of independent Leaky Buckets, one for each of the N input VC's, and a FCFS multiplexer. Outputs from these Leaky Buckets are multiplexed onto the VP with burstiness constraint (δ, ρ) in FCFS bases. Let the bucket size and token-generation rate of the ith Leaky Bucket be denoted by δ_i and ρ_i. When setting $\sum_{i=1}^{N} \delta_i = \delta$ and $\sum_{i=1}^{N} \rho_i = \rho$, it is easy to show that the aggregate output traffic of the multiplexer conforms to the burstiness constraint of the VP, (δ, ρ). To prove this, consider the following. Denote the Leaky Bucket of the ith session as LB_i and the output traffic rate function of LB_i as R_i. Then $R_i \sim (\delta_i, \rho_i)$ or

$$\int_x^y R_i(t)dt \leq \delta_i + \rho_i(y - x) \tag{2}$$

for all x and y satisfying $y \geq x$. If we sum inequality (2) for $i = 1 \ldots N$, we get

$$\int_x^y \sum_{i=1}^{N} R_i(t)dt \leq \delta + \rho(y - x) \tag{3}$$

which implies $R_1 + R_2 + \cdots + R_N \sim (\delta, \rho)$. Thus delay and loss in the VP Leaky Bucket is zero and delay and loss in VP are bounded.

This independent operation is analogous to fixed-rate allocation in hard multiplexing where there is no bandwidth sharing among sessions. With both soft and hard multiplexing, throughput of each session i is guaranteed at rate ρ_i. The implementation is also straightforward. An obvious drawback of fixed-bandwidth allocation, however, is poor utilization of network bandwidth. For example, in soft multiplexing, an input may be idling with a overflowing bucket while another active input may suffer undue delay and loss.

We shall use fluid-flow approximation [3] for our discussion in this paper. For illustration, let us consider a situation in which there are two sessions, s_1 and s_2, as depicted in Fig. 3. There are two graphs, one for each session. In each graph, the darkened solid line corresponds to the cumulative arrivals of the session, the lower line corresponds to the cumulative number of tokens accepted into the bucket (all tokens must be accepted into the bucket before they can be acquired by cells), the upper line equals to the lower line plus the token originally in the bucket at time 0. At time t_3, s_1 has run out of tokens (i.e., $A_1(t) \geq \delta_1 + K_1(t)$). While s_1 suffers a delay from t_3 to t_4, there are extra tokens stored in LB_2. In principle, this delay is avoidable and can be shortened because many tokens of s_2 are being wasted in the interval $(0, t_2)$ because of bucket overflow.

To examine whether this soft server is work conserving, we define $l_i(t)$ to be the number of tokens in LB_i at time t and $l(t)$ to be the number of tokens in the VP Leaky Bucket at time t. Then $\sum_{i=1}^{N} l_i(t) < l(t)$ when any of the N Leaky Buckets overflows and some others are not full. This means that $l(t) - \sum_{i=1}^{N} l_i(t)$ tokens in the VP Leaky Bucket can never be used, or in other words, certain bandwidth of the VP is wasted. Thus, waste of tokens in soft multiplexer implies waste of tokens in the VP Leaky Bucket. Since it is possible that some sessions are backloged while others are wasting tokens, this multiplexer is not work conserving.

IV. Excess Token Passing

This section explores the use of excess token passing to minimize the number of tokens wasted in the above system. The mechanism of excess token passing is simple. Whenever a bucket in the soft multiplexer is filled up, a newly generated token will be passed to others instead of being discarded.

In token passing, $R_i \sim (\delta_i, \rho_i)$, for $i = 1 \ldots N$ no longer holds. However, the output of the soft multiplexer should still satisfy the VP burstiness constraint, as argued below. Cells can be released into the VP only if it can be granted a token. Therefore, the total number of output cells within a time period cannot exceed the number of tokens in the system at the beginning of the period plus the number of tokens generated during the period. Therefore,

$$\begin{aligned}\sum_{i=1}^{N}\int_{x}^{y} R_i(t)dt &\leq \sum_{i=1}^{N} l_i(x) + \sum_{i=1}^{N}\rho_i(y-x) \\ &\leq \sum_{i=1}^{N}\delta_i + \sum_{i=1}^{N}\rho_i(y-x) \\ &= \delta + \rho(y-x)\end{aligned}$$

An excess-token passing soft multiplexer still provides the same guarantee to sessions as the one with independent Leaky Buckets does. In addition, a session i's backlog will always be cleared at rate $\geq \rho_i$. Because newly generated tokens in the multiplexer will be discarded if and only if all Leaky Buckets overflow, $\sum_{i=1}^{N} l_i(t) = l(t)$, for all t. Thus, the bandwidth of the VP will be wasted if and only if all Leaky Buckets in the soft multiplexer overflow. Since no cell will be waiting when all buckets in soft multiplexer overflow, excess-token passing multiplexers are work conserving.

The fluid flow diagram in Fig. 4 gives us a clearer picture of excess token passing. Comparing with Fig. 3, fewer tokens are wasted, because in the interval (t_1, t_2), excess tokens of s_2 are transferred to s_1, and also in (t_3, t_4), excess tokens of s_1 are transferred to s_2. When s_1 suffers a delay in independent case (Fig. 3), both of the sessions can be served without any delay after adopting excess token passing.

This system appears much like General Processor Sharing(GPS) multiplexing [3] if the share of excess tokens passed to session i is $\frac{\rho_i}{\sum_{j\in B}\rho_j}$, where B is the set of sessions having a non-full bucket. One characteristic of GPS is that unused bandwidth of an idle session can be shared by active sessions. This is also the main spirit of excess token passing. The main difference between them is that a session in our system may save up tokens for later bursts. Only excess tokens from a full Leaky Bucket are passed to other sessions. A soft multiplexer with excess token passing can be described as follow. Each of the N sessions is associated with a positive real number, $\phi_1, \phi_2, \cdots, \phi_N$. Let $T_i(\tau, t)$ be the number of tokens received by session i in $(\tau, t]$. A soft multiplexer with excess token passing ensures that:

$$\frac{T_i(\tau,t)}{T_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}, j = 1, 2, \cdots, N. \qquad (4)$$

for a session i which has a non-full token pool in the interval $(\tau, t]$. We can see that every session i is guaranteed an effective token generation rate of $\rho_i = \frac{\phi_i}{\sum_j \phi_j}\rho$. By setting ϕ_i's, we can distribute the tokens generated to sessions according to the their individual requirements. The larger ϕ_i, the larger share of token session i gets. If we set bucket size to zero, system with excess token passing is equal to GPS systems (i.e. $\delta_i = 0, \forall i$ and $\delta = 0$).

V. Token Borrowing

While excess token passing minimizes the number of tokens wasted, token borrowing aims at improving the delay/loss performance in soft multiplexer. In token borrowing, whenever the bucket of a session becomes empty and backlog occurs, it may borrow tokens from others and return them later. For the same reason we mentioned in excess token passing, the aggregate output of a system with token borrowing satisfies the burstiness constraint of the VP, too.

Figure 5 illustrates the conceptual structure of a token-borrowing soft multiplexer. On top of the Leaky Buckets is attached a coordinator which monitors the states of all Leaky Buckets and moves tokens among the buckets. This coordinator maintains a vector $B = (B_1, B_2, \cdots, B_N)$ in which B_i is an integer indicating the number of tokens session i has borrowed if B_i is negative, or the number of tokens session i has lent out if B_i is positive. A session with $B_i > 0$ is called a *lender* and a session with $B_i < 0$ is called a *borrower*. Initially, B_i $(i = 1, \ldots, N)$ will be set to 0. A real number h, $0 \leq h \leq 1$, is used to indicate whether tokens can be borrowed from a session. If the number of tokens in bucket i is less than $h * \delta_i$, session i stops lending tokens to others. The following illustrates how token borrowing works:

Token Borrowing Mechanism :

If session j begins to be backloged, the coordinator will try to transfer to it a token from a leaky bucket i which has more than $h * \delta_i$ tokens. Then, B_j will be decremented by 1 and B_i will be incremented by 1. The system may evolve to a situation in which all buckets $i(i = 1 \cdots N)$ contain less than $h * \delta_i$ tokens. At that time, the coordinator will stop transferring tokens around.

Token Returning Mechanism :

A newly generated token for a borrower j will be passed to the coordinator if there is no backlog for session j. The coordinator will return this token to one of the lenders, say session i. Then it decrements B_i by 1 and increments B_j by 1. A borrower may also be forced to return the newly generated token if a lender begins to be backloged when all buckets i $(i = 1, \ldots, N)$ contain less than $h * \delta_i$ tokens.

Let us now consider the fluid flow graphs in Fig. 6 to illustrate the mechanism of token borrowing. A differ-

ence to the previous fluid flow graph is that the lower line indicates $K_i(t)+L_i(t)$, where $L_i(t)$ is the number of tokens borrowed in at time t if it is positive, or the number of tokens lent out at time t if it is negative. When session 1 runs out of tokens at time t_1, the coordinator will let it borrow from session 2. Therefore, $K_2(t)+L_2(t)$ decreases in the interval (t_1, t_2) since tokens are being lent out. During this interval, B_1 decreases and B_2 increases. After t_2 (end of burst of session 1), newly generated tokens of session 1 will be returned to session 2 with adjustment of B_1 and B_2 until both B_1 and B_2 become 0. We can see that both of the sessions can be served without delay/loss.

Parameter h is used to control the trade-off between performance and guarantee. When $h = 1$, no tokens will be lent out and the system is analogous to the one with independent Leaky Buckets. As mentioned in section II, this system provides strong guarantee but marginal performance. When $h = 0$, a backloged session can borrow freely from others whenever there are some non-empty buckets. Therefore, mean delay and loss are minimized. However, QoS of lenders may be affected by borrowers. Fig. 7 shows a situation in which session 2 is affected by a misbehaving session that always transmits at its peak rate. Without token borrowing, session 2 can be served without delay. However, after adopting token borrowing with $h = 0$, delay occurs whenever a burst with rate $C_2 > \rho_1 + \rho_2 = \rho$ arrives. This is because session 2's bucket will always be empty due to session 1's borrowing, and when a burst of session 2 arrives, although session 1 is forced to return its newly generated tokens, the total token generation rate $\rho = \rho_1+\rho_2 < C_2$. Session 2's performance can be guaranteed only if $C_2 \leq \rho_1 + \rho_2$.

In general, lender i can be guaranteed if the following is satisfied when lender i begins to be backloged:

$$\rho_i + \rho_R(t) \geq C_i \tag{5}$$

where $\rho_R(t)$ is the token returning rate at time t.

Assume that there are N sessions, let $B(t)$ represent the set of borrowers and $NB(t)$ be number of borrowers at time t. Similarly, the set of lenders and the number of them at time t are represented by $L(t)$ with $NL(t)$. In the worst case, all $NL(t)$ lenders become backloged. The newly generated tokens of the $NB(t)$ borrowers will be shared by the $NL(t)$ lenders and a lender i can have a token returning rate of

$$\rho_R(t) = \frac{\sum_{i \in B(t)} \rho_i}{NL(t)} \tag{6}$$

assuming the returned tokens are distributed evenly among all the lenders. So the lenders can be guaranteed if

$$C_i < \rho_i + \frac{\sum_{i \in B(t)} \rho_i}{NL(t)} \tag{7}$$

In the extreme case, there is only one borrower and all others are lenders. If we assume all the sessions i are homogeneous with peak rate $C_i = C$ and $\rho_i = \rho'$ for all i, the criterion becomes

$$\rho' > \frac{N-1}{N} C \tag{8}$$

If (8) is satisfied, lenders can be guaranteed. However, this is a rather stringent requirement and the allocated rate ρ' must be very close to the the peak rate C.

Another method to guarantee the lenders is the detection of misbehaving sessions. By examining the vector B, the coordinator can find out those sessions that have borrowed an exceeding number of tokens from others. The coordinator can send warning signals to the sources or refuse to transfer tokens to these sessions.

Excess token passing can work together with token borrowing. In this situation, the resulting soft multiplexer is work conserving. If $h = 0$, a session can borrow from others whenever there are some non-empty buckets and a cell will be buffered only when all the buckets are empty. In other words, packets will be served immediately if there are some tokens. Therefore, the resulting soft multiplexer is also greedy.

VI. Simulation Results

The simulation experiments assume there are three homogeneous sessions modeled by the two-state Markov chain in Fig. 8 with both mean on period and mean off period equal to $\frac{1}{60}s$. During an on-period, cells would arrive with rate $S = 24000 cells/s$. Thus the mean arrival rate r is about $5Mb/s$. We set δ_i to be the mean burst length, which is 400 cells, for all i. A buffer with size 400 was attached to each Leaky Bucket. Since we are only interested in the range $r < \rho_i < S$, we may vary ρ_i in (12000, 24000), for all i. In Fig. 9(a) and (b), the cell-loss probability and delay of the token-passing scheme are compared with those of the independent Leaky-Bucket scheme. We see that for the same load $(\frac{r}{\rho_i})$, both cell-loss probability and delay can be improved by one to two orders of magnitude with token passing. For a fixed loss-probability or delay requirement, the sustainable load of each session can be higher. It means that, for a fixed set of multiplexed sessions, a smaller VP is needed. This is rather encouraging in that we can improve the performance a lot without losing fairness.

The effects of token borrowing are presented in Fig. 10(a) and (b). When the threshold of token borrowing $h = 1$, the resulting soft multiplexer is the same as an independent Leaky-Bucket system. Fig. 10(a) shows that cell-loss probability can be improved significantly when we change h from 1 to 0.99. Although improvements can still be observed when h is decreased further, they becomes progressively less significant. When $h = 0.99$,

a session will not lend out tokens unless its token pool is 99% full. Therefore, with $h = 0.99$, a session is highly guaranteed. it is therefore interesting that we can improve the cell-loss probability by one order of magnitude while maintaining a high degree of guarantee. This result shows that when some sessions are running out of tokens, it is very likely that there are other sessions which have buckets that are more than 99% full. If token borrowing had been disabled, the newly generated tokens of these buckets would have been discared with high probability. Token borrowing allow us to put these tokens to good use by transferring them to the needy sessions. This is similar to the effect of excess token passing except that, in token borrowing, the tokens are lent out rather than transferred and a borrower must later return what he has borrowed.

The hybrid system of token passing and borrowing has also been studied. In Fig. 11(a) and (b), the upper solid line is for the independent Leaky Buckets case. The middle three lines correspond to the case of token borrowing and are reproduced from Fig.10. The lower three lines correspond to the hybrid system. We see that, in the latter case, the curves keep shifting downwards until $h = 0$. Note that when $h = 0$, the soft multiplexer is greedy and the best average performance can be achieved. However, there is less guarantee for each session.

We now examine the fairness of soft multiplexers by considering a scenario in which one of the sessions misbehaves and injects unexpected long bursts into the VP. Fig. 12, the upper bunch of curves corresponds to the misbehaving session and the lower bunch of curves corresponds to the normal sessions. We see that, even though the misbehaving session would borrow tokens from the normal sessions, all sessions are better off when excess tokens passing and token borrowing are enabled. This is true even in the case of $h = 0$ when the guarantee to each session is reduced to its minimum. This is interesting because the independent Leaky-Bucket operation is supposed to provide protection of a session from misbehaving sessions, and yet our results indicate that this protection does not necessarily leads to better guaranteed performance. It should be emphasized, however, that the loss-probability performance shown is an average over a long time period. Further study is needed to better understand the dynamics of the interactions of the sessions at particular time instants.

VII. Conclusions

This paper has investigated soft multiplexing, an issue that arises in VP-based ATM networks. The main distinguishing feature of this form of multiplexing is that the output channel is a variable-bit rate channel. We have proposed a modification of the concept of work conservation (a well-known concept in traditional statistical multiplexing) for use in soft multiplexing and have introduced the concept of greedy soft multiplexers. These concepts allow us to classify and study different multiplexing schemes systematically. We focused on soft multiplexers with input traffic streams regulated by individual Leaky Bucket and the multiplexed output traffic policed by a Leaky Bucket. The underlying assumption is that as long as the output traffic conforms to the prenegotiated contract specified by the Leaky-Bucket parameters, the network is obligated to meet certain performance guarantees on the output traffic. The issue then is reduced to the study of the performance within the multiplexer. Two performance-enhancing soft multiplexing schemes, excess token passing and token borrowing, have been investigated. It has been shown that by allowing tokens to be transferred among the input Leaky Buckets according to need in a systematic fashion, significance performance improvement can be achieved without sacrificing guarantee to each session.

References

[1] M. Kawarasaki, B. Jabbari, "B-ISDN Architecture and Protocol" *IEEE J. select. Areas Commun.*, Vol.9, No.9, Dec. 1991, pp.1405-1415.

[2] K. Sato, S. Ohta, I. Tokizawa, "Broad-Band ATM Network Architecture Based on Virtual Paths" *IEEE Trans on Commun.*, Vol.38, No.8, Aug. 1990, pp.1212-1222.

[3] A. K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks : The Single-Node Case," *IEEE/ACM Trans. on Networking*, Vol.1, No.3, June 1993, pp.344-357.

[4] A. K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in integrated Services Networks : The Multiple Node Case," *Proc. IEEE INFOCOM'93*, Mar. 1993, pp.521-530.

[5] J.B. Nagle, "On Packet Switches with Infinite Storage" *IEEE Trans on Commun.*, Vol.35, No.4, Apr. 1987, pp.435-438.

[6] S. J. Golestani, "Congestion-Free Transmission of Real-Time Traffic in Packet Networks" *Proc. IEEE INFOCOM'90*, San Francisco, Jun. 1990, pp.527-536.

[7] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management" *Proc. ACM SIGCOMM'90*, Vol.20, No.4, Sept. 1990, pp.8-18.

[8] S. J. Golestani, "Duration-Limited Statistical Multiplexing of Delay-Sensitive Traffic in Packet Networks" *Proc. IEEE INFOCOM'91*, Bal Harbor, Fl., Apr. 1991, pp.323-332.

[9] J. S. Turner, "New Direction in Communications or Which Way to the Information Age?" *Proc. Int. Zurich Sem. Digit. Commun. '86*, pp.25-32.

[10] M. Butto, E. Cavallero and A. Tonietti, "Effectiveness of the Leaky Bucket Policing Mechanism," *IEEE J. select. Areas Commun.*, Vol.9, No.3, Apr. 1991, pp.335-342.

[11] N. Yin, M. G. Hluchyj, "Analysis of the Leaky Bucket Algorithm for On-Off Data Source" *Proc. GLOBECOM'91*, pp.254-260.

[12] L. Kleirock, *Queueing System*, Vol.2, Computer Application, John Wiley, pp.1141-1147.

[13] R. L. Cruz, "A Calculus for Network Delay, Part 1 : Network Elements in Isolation" *IEEE Tran. on Information Theory*, Vol.37, No.1, Jan 1991, pp.114-131.

[14] R.L. Cruz, "A Calculus for Network Delay, Part II : Network Analysis" *IEEE Tran. on Information Theory*, Vol.37, No.1, Jan. 1991, pp.132-141.

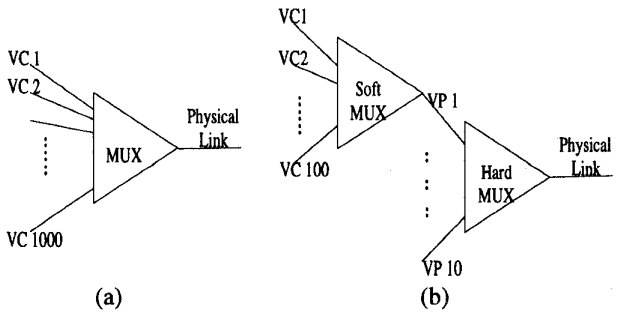

Figure 1 (a) Traditional multiplexing
(b) Two-layer multiplexing

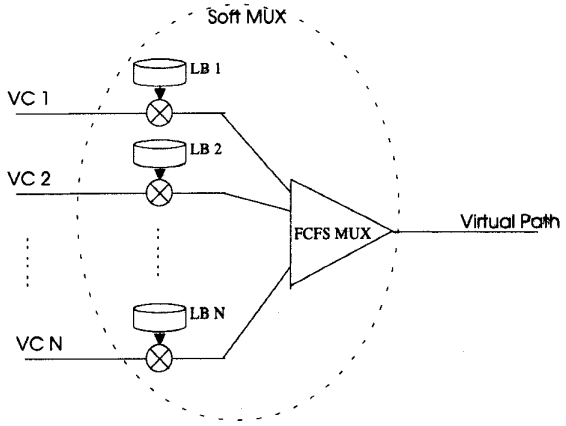

Figure 2
Soft multiplexer with Leaky Buckets

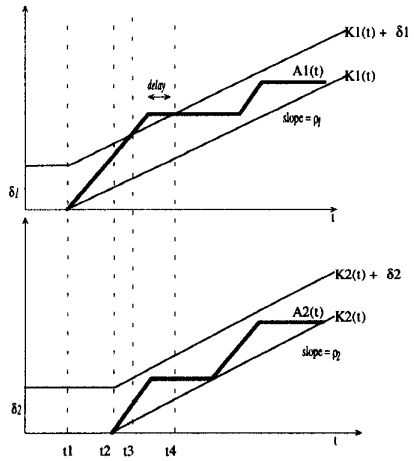

Figure 3
Fluid flow diagram for soft MUX
with with independent Leaky Buckets

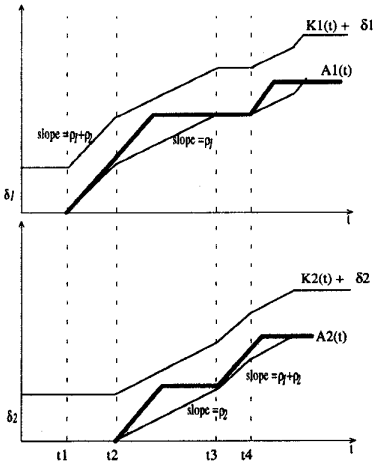

Figure 4
Fluid flow diagram after adopting
excess token passing

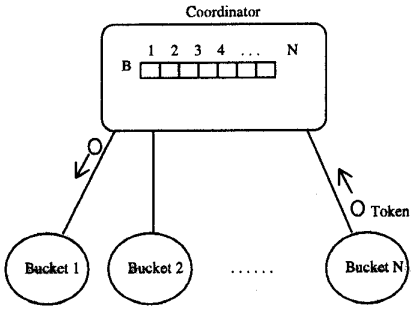

Figure 5
Structure of soft MUX
token borrowing

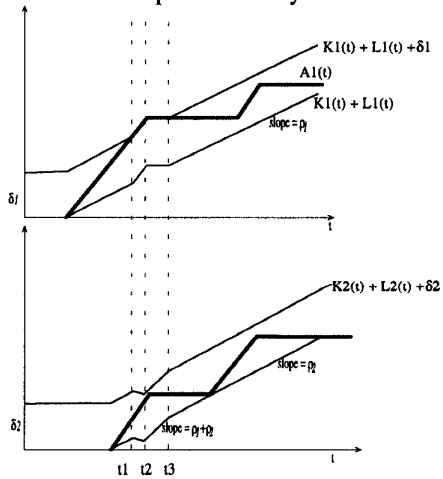

Figure 6
Fluid flow diagram after adopting
token borrowing

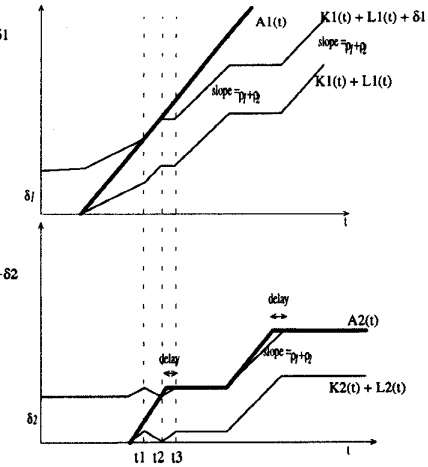

Figure 7
Fluid flow diagram illustrating
the unfairness to lenders

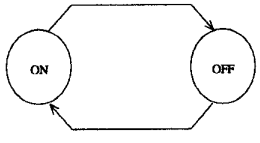

Figure 8
2-state markov-chain for
on-off sources

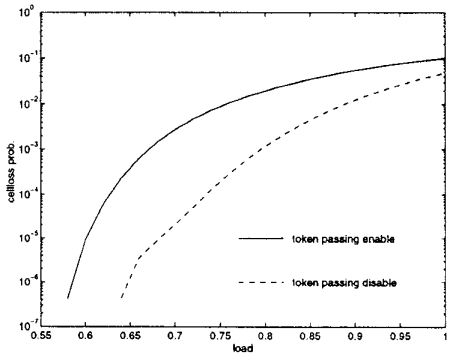

Figure 9(a) Cell Loss of Excess Tokens Passing

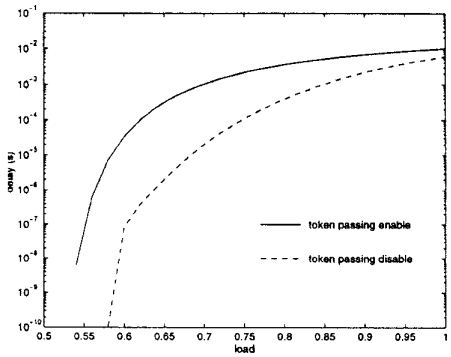

Figure 9(b) Delay of Excess Token Passing

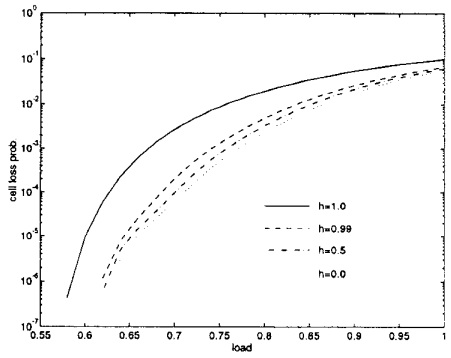

Figure 10(a) Cell Loss of Token Borrowing

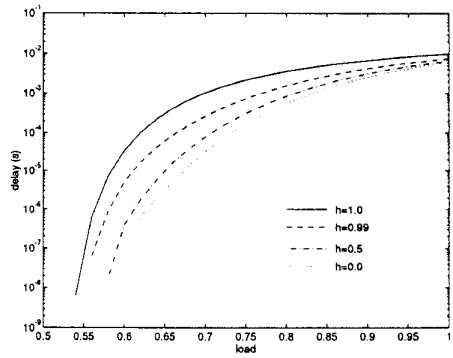

Figure 10(b) Delay of Token Borrowing

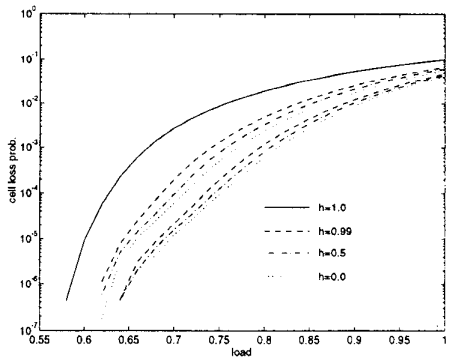

Figure 11(a) Cell Loss of Both Excess Token Passing & Token Borrowing

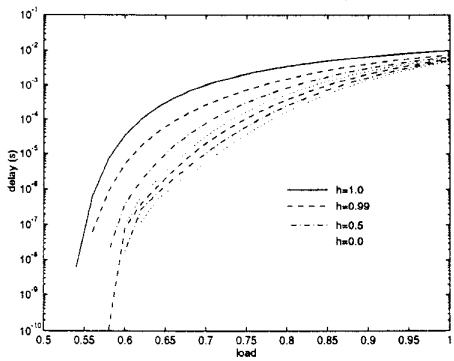

Figure 11(b) Delay of Both Excess Token Passing & Token Borrowing

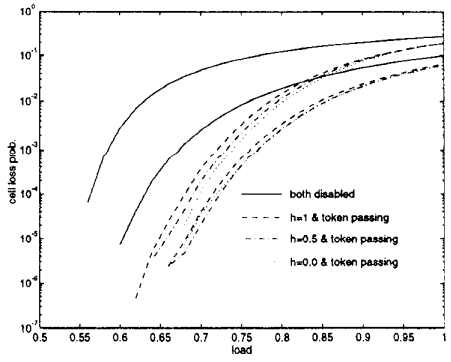

Figure 12 Effect of a Misbehaving Session to Normal Sessions