A Simple Adaptive SVC Caching Scheme for Voice Trunking Over ATM (VTOA) Applications

Yetik Serbest
SBC Technology Resources Inc.
Austin, Texas

Sangkyu Park, Aimin Sang
Electrical and Computer Engineering
Univ. of Texas at Austin

Abstract—**In Voice Trunking over ATM (VTOA) technology, the tandem is replaced by three components: Trunk Inter-Working Function (T-IWF), Control and Signaling Inter-Working Function (CS-IWF), and ATM Network. This new architecture calls for re-examination of the signaling channel delay issues, which are well-studied and well-specified for TDM networks. In VTOA, the ATM network together with the inter-working functions act as a virtual tandem switching system, which is called "ATM-based distributed virtual tandem switching system." The aim is that the performance of ATM-based trunking network should be at least as good as TDM-based network. The cross-office delay budget must be shared among the interworking functions and the ATM network. Hence, the time for the ATM network to establish a switched virtual connection (SVC) is stringent.**

In this paper, we propose a simple adaptive SVC caching scheme to offload the high processing capacity burden on ATM switches. After a SVC is established in a usual way for a call, it is not torn down after the conversation is over. Instead, it is kept alive for variable duration (i.e., delayed release) with the expectation that there would be another call request for the same terminating end office during that time. The caching duration is adaptively changed with call arrival rate and call setup delay in the ATM network in order to stay within the delay budget specified by the requirements. The adaptability and the convergence of the algorithm are also shown by extensive simulations related to practical scenarios.

Keywords—**VTOA, SVC, Setup Latency, Caching, TDM, SS7, Cross-office Delay, Post-dial Delay.**

I. Introduction

In today's voice network, end offices are connected via tandem trunks or direct trunks or both. Each trunk is a DS0 ($64kbps$) that is transmitted between the switching offices in a time division multiplexed manner. Each end office connects to its neighboring end offices and the tandem office using separate trunk groups. A simple logical diagram of today's Time Division Multiplexing (TDM) based trunking network is seen in Fig. 1, where many transport level details are skipped. The trunk groups are forecasted and pre-provisioned with dedicated bandwidth, which may lead to inefficiency and high operations cost.

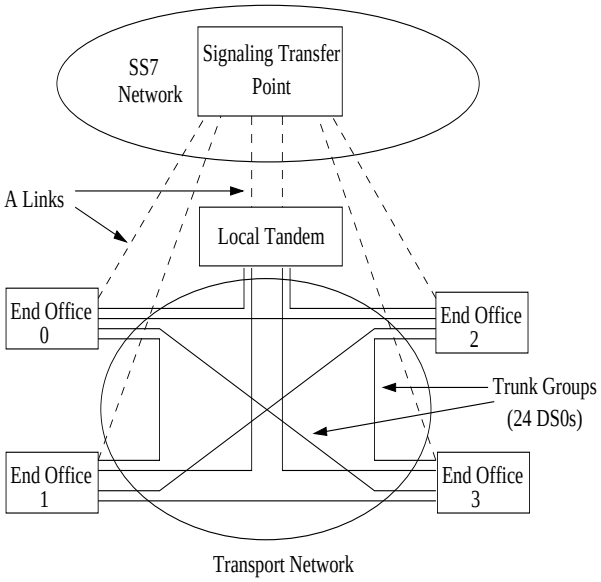

Fig. 1. A Logical Diagram of the TDM Circuit-Switched Voice Network

Recently, a new way of voice trunking using ATM technology has been proposed ([7], [2], [9]). In this scheme, voice trunks from end office Class 5 switches are converted to ATM cells by a device called Trunk Inter-Working Function (T-IWF). Refer to [10] for extensive summary of technical details for TDM-ATM conversion. The T-IWFs are distributed to each end office, and are controlled by a centralized Control and Signaling Inter-Working Function (CS-IWF). The CS-IWF performs call control functions as well as the conversion between the narrowband Signaling System No.7 (SS7) protocol and the broadband signaling protocol .

The T-IWFs, the CS-IWF, and the ATM network form ATM-based distributed virtual tandem switching system, which is depicted in Fig. 2. The term "distributed" refers to the fact that the tandem functions are carried out in part by the T-IWFs that are located in the end offices in a distributed manner. The term "virtual" means that the ATM-

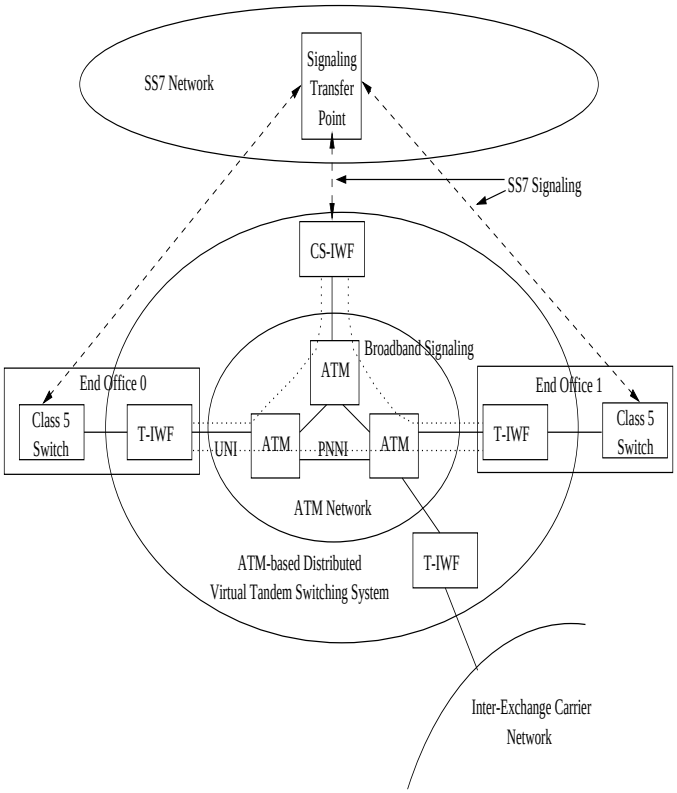

Fig. 2. An ATM-based Trunking Network (VTOA Architecture)

based trunking network is functionally equivalent to the traditional TDM tandem switching system. In addition, the trunks are no longer DS0 time slots that are statically provisioned. Instead, the trunks are realized through dynamically established switched virtual connections (SVCs) using PNNI QoS routing protocol, which eliminates the need to provision separate trunk groups to different destinations as is done in TDM-based trunking networks.

The new three-component architecture (i.e., T-IWFs, CS-IWF, and ATM network) calls for re-examination of the signaling channel message processing, which are well-studied and well-specified ([6], [4]) for TDM networks. The aim is that the performance of ATM-based trunking network should be at least as good as that of TDM-based network.

The signaling channel processing requirements are well-known for TDM circuit-switched voice network. ITU-T Q.766 ([6]) and Bellcore GR-1364-CORE ([4]) are examples among many others. These specifications dictate the cross-office delay requirements for processing of SS7 messages. The actions needing to be taken in each office are clearly defined upon reception of a particular SS7 message. For a normal tandem trunk call flow, the originating end office sends an Initial Address Message (IAM) to the tandem switch through SS7 network. The IAM message includes routing address of the tandem office, calling telephone number, called telephone number, and Trunk ID. The tandem switch has a mean processing delay budget of 180 ms as specified in [6] (360 ms for 95th percentile) to process the IAM message and to reserve a trunk in the trunk group that is pre-established to the terminating end office.

In VTOA architecture, the end offices and the virtual tandem (i.e., CS-IWF) communicate through SS7 network as seen in Fig. 2, the same way the switching offices do in TDM-based trunking networks. However, control/signaling and through-connect establishment (a SVC through the ATM network) functions reside in different entities, CS-IWF and ATM network (T-IWF as well) respectively. The coordination of these different components adds new message exchanges into the picture. As a result, these three components should share the 180 ms (mean) budget, as they are considered to be a unique entity, i.e., a virtual tandem switching system. In this architecture, there are two options for CS-IWF upon reception of an IAM message. First is to order either originating or terminating T-IWF for initiation of an ATM connection and wait for an "ATM SVC Established" message before sending the IAM message to the terminating end office. Second is to send the IAM message to the terminating end office at the same time it sends a request to either T-IWF for an ATM connection establishment. It is expected that the ATM connection will be ready before the reception of Address Complete Message (ACM), which indicates that ringing is applied to the callee and the through-connect should be established in the tandem. The second option provides more time for the establishment of a SVC through ATM network. However, a SVC may very well go through several ATM switches. The ATM switches generally have reasonably large figures for call setup latency. Even though there can be some exceptions, it would be unreasonable to assume so, since the latency numbers of the new switches are yet to be tested, and already deployed ATM switches can be assumed to serve for years to come. In other words, for either option we need a scheme for fast SVC setup through ATM network to stay in the standardized delay budget limits.

In this paper, we propose an adaptive SVC caching scheme to overcome the limitation of ATM switches posed above. The idea is simple: delayed release of SVCs. In this scheme, an already established SVC is not released as soon as the conversation is finished (that is, when either side hangs up). Instead, it is kept alive for a variable duration, what we call the caching time, with the expectation that during that time another call request for the same ter-

minating end office would arrive. The caching duration is adaptively changed with call arrival rate and call setup delay experienced in ATM network in order to stay within the delay budget specified by the requirements. Thus, the processing load of ATM network is constantly monitored and the caching time is changed accordingly. Intuitively, the caching time should be increased when the call setup time exceeded the budget, and should be decreased when the call setup time were less than the requirement.

In our analysis, we first derive a relation between the processing delay and the caching time. Poisson call arrivals and Exponential call holding times are assumed. In the algorithm, we utilize this derivation to determine the new caching time. Hence, once we measure the average call setup delay, we can determine the appropriate caching time from the simple relation we derived.

In the paper, we also study adaptability and convergence of the algorithm with extensive simulation study. We find that this simple algorithm is successful to track the changes in the processing load of ATM network (call setup delay) and in the call arrival rate. In our study, we try different distributions (Gaussian and Weibull) for call setup delay. The efficiency versus delay budget allocation tradeoff is studied as well.

The main contribution of the paper is that by a simple caching algorithm implemented at the edges of the broadband network (at the T-IWFs), call processing load of ATM network can be decreased.

The remaining part of this paper is organized as follows: In the next section, the algorithmic design is given. The tuning parameters and their roles in the scheme are also discussed in detail. The statistical analysis of practical scenarios in Section III indicates wide applicability of the proposed caching scheme. The implications of our findings on the VTOA architecture are discussed in Section IV. The paper ends with a conclusion in Section V.

II. Proposed SVC Caching Scheme

Before we give the details of the SVC caching algorithm, let us motivate it by providing the intuition behind it. As given in Section I, the main idea is the delayed release of SVCs. That is, even after the conversation is over, the established SVC is to be kept alive for an adaptive duration hoping that during that time there could be a call request for the same destination, hence, the same SVC could be recycled. The urging point here is to reduce the cost of call processing in ATM network. The algorithm is designed to adaptively determine the caching duration. The heart of the algorithm is the adaptation process, which is the main topic of this section.

In VTOA, network operator (carrier) will decide on the delay budget for the ATM network portion of ATM-based distributed virtual tandem switching system. This decision would very well be a compromise with respect to processing power of ATM switches. For a given delay budget of d_{budget} (i.e., the requirement for mean call processing time), the mean SVC setup latency in the ATM network should be kept below d_{budget}. Otherwise, the call setup latency requirement for voice networks would be violated. In return, there might be some unwanted consequences such as increase in impatient hang-ups and re-attempts due to escalated post-dial delay.

Since T-IWFs at the edge of ATM network initiate SVC setup in VTOA architecture, it is the T-IWF where the SVC caching scheme is implemented to enforce the d_{budget} requirement. To do so, it needs to track SVC setup latency in ATM network. The stipulation is that the ATM network is to carry integrated services not just the voice traffic. Hence, other SVC services (for instance, ADSL and Frame Relay) might impose further processing burden on ATM network. Besides, the ATM network is a black box to T-IWFs. The T-IWFs are unaware of ATM topology. The ATM network utilizes PNNI QoS routing protocol to find a path for the SVC to be established. The PNNI checks transport level resources (such as bandwidth and buffer) to accommodate toll quality voice service. Thus, the PNNI is blind to signaling channel resources (call processing capacity). Therefore, the SVC setup latency in ATM network should be measured separately. At this point, we should add that a recent proposal to incorporate setup latency to predict crankbacks in the PNNI protocol can also be used [3]. However, it requires some changes in the protocol. On the other hand, our approach can be implemented at the edges of the network and imposes no changes in the network protocol itself.

The processing load in ATM network is expected to be time-varying. The T-IWF should probe the changes in call processing load of ATM network by deploying a measurement scheme to estimate SVC setup latency of ATM network. This can be done easily as follows: by measuring the time elapsed between UNI “SETUP” message sent and “CONNECT” message received as depicted in Fig. 3. Since the activity of voice traffic changes by time of the day (work or off-work hours) and by community of interest (business or residential), each T-IWF should keep a separate measurement to every other T-IWF.

Intuitively, there is a monotonous relationship between

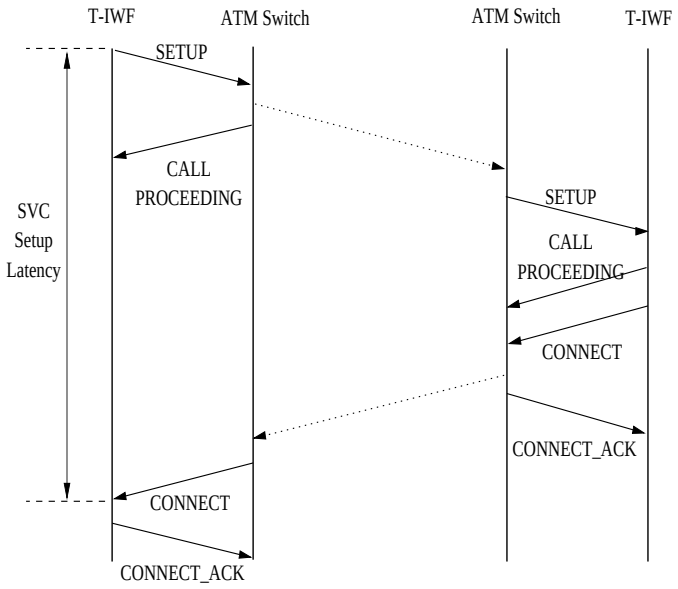

Fig. 3. SVC Setup Connection Messages

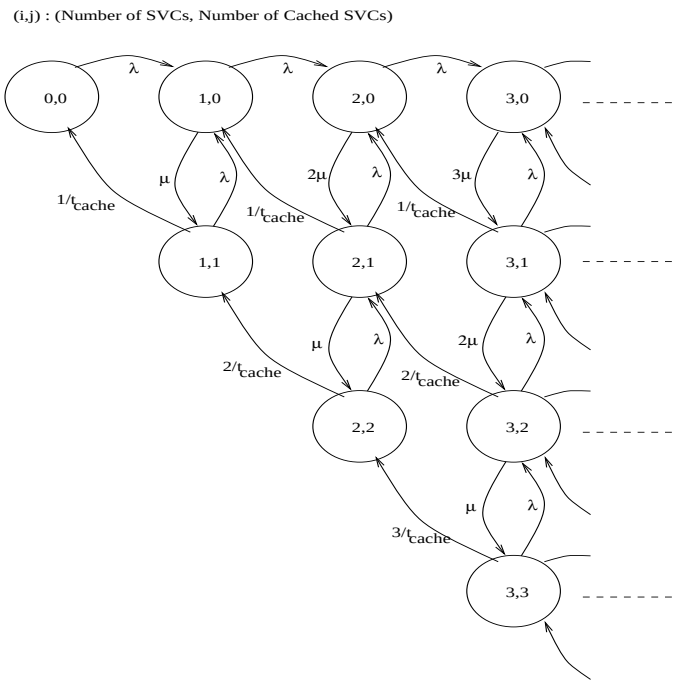

Fig. 4. State Transition Diagram of SVC Caching

mean SVC setup latency and the caching duration. As caching duration increases (decreases), the mean SVC setup latency decreases (increases). For instance, the more time a SVC is cached, the higher probability a call request is accommodated, that is, a cached SVC is hit. Our assumption is that the SVC setup latency for a call, which is taken care of by a cached SVC, is zero.

Next, an explicit relation between mean SVC setup latency and caching time is derived. In the analysis, we consider calls with Poisson arrival rate λ, and Exponentially distributed independent holding times with mean $\frac{1}{\mu}$. We can construct a Markov Chain, in which the state is represented by (number of SVCs, number of cached SVCs) pairs, where the number of SVCs include all established connections, whereas the number of cached SVCs is a counter that represents SVCs that are in the cache and that do not carry any traffic at the moment. The upper limit for the number of SVCs is the total number of trunks (DS0s), represented by N_{trunk}, coming out of the end office switch. Only some portion of trunks is allowed to be cached due to trunk efficiency concerns as well as due to SVC needs of other services. Therefore, there is also an upper limit, represented by N_{cache_limit} for the number of cached SVCs. The discussion on how to choose an appropriate N_{cache_limit} is given later in this section.

The cached SVCs, if they are not recycled, are released after the caching duration t_{cache} expires. Although t_{cache} is constant for every adaptation period, in the analysis we assume that it is Exponentially distributed.

In Fig. 4, the state transition diagram of the Markov Chain is depicted. The steady state distribution can be found numerically by Gauss Seidel method given in [8]. Consequently, it is straightforward to find mean call setup latency of the caching scheme d_{post_cache} as shown in (1). The mean SVC setup latency in the ATM network is represented by d_{setup}. One should note that when there is cached SVC in the system, the setup latency of a new call is zero. Hence, only the states with no cached SVCs (i.e., $\pi(i,0)$) are contributors to the calculation.

$$d_{post\_cache} = \sum_{i=0}^{N_{trunk}} \pi(i,0) d_{setup} \qquad (1)$$

Since the construction of the state transition matrix of the Markov Chain in Fig. 4 is cumbersome, we developed an approximation. For this approach, the calls are first served by an $M/M/\infty$ queuing system. That is, there are infinite number of trunks available (no call blocking) for SVC establishment. Every SVC upon completion of its call enters the caching system, which is represented by another $M/M/\infty$ queuing system. In the caching system, SVCs are served (i.e., released) by an Exponential server with a mean period of $t_{cache\_mean} = f(t_{cache}, \lambda, \mu)$. We know that $t_{cache\_mean} \in [0, t_{cache}]$ due to cache hits. Since it is hard to come up with an exact expression for t_{cache_mean}, we heuristically make the following approximation: $t_{cache\_mean} \approx \beta t_{cache}$, $\beta \in (0, 1]$. Again, once we determine the steady state distribution, which is Poisson in this case ([1]) as seen in (2), we can easily determine

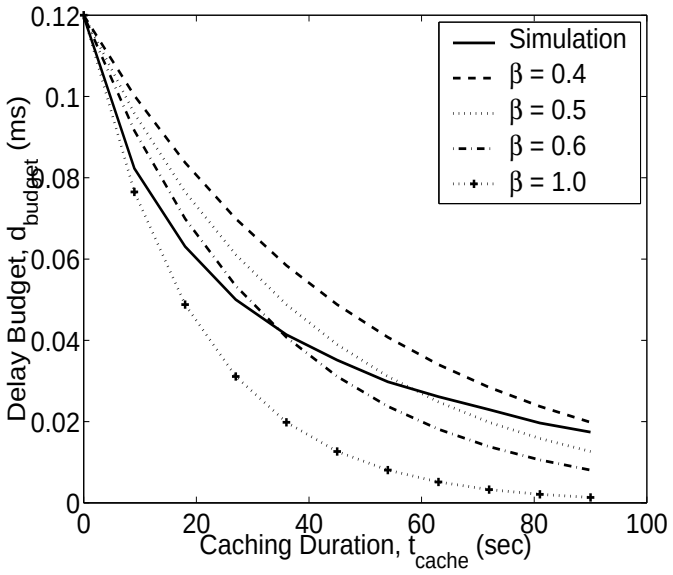

Fig. 5. Delay Budget vs Caching Duration

the mean call setup latency as given in (3). The infinite size of the queuing systems in this approximation is a reasonable one. In practice, the number of trunks is designed to be huge in order to have a very small blocking probability ($\approx 10^{-5}$). Although a certain percentage ($\approx 10\%$) of the total trunks is allowed for caching in practice due to efficiency concerns, the number of cachable SVCs is still big, considering that the total number of trunks in the end offices today is larger than 4000.

$$\pi(i) \approx (\lambda\beta t_{cache})^i \frac{\exp(-\lambda\beta t_{cache})}{i!},\ i = 0, 1, \ldots \quad (2)$$

$$d_{post\_cache} \approx \pi(0) d_{setup} \approx d_{setup} \exp(-\lambda\beta t_{cache}) \quad (3)$$

The approximation given in (3) furnishes a very simple and useful relation among caching time, t_{cache}, call arrival rate, λ, and allocated delay budget d_{budget} shown in (4).

$$t_{cache} \approx \frac{1}{\lambda\beta} \log(\frac{d_{setup}}{d_{budget}}) \quad (4)$$

In Fig. 5, the closeness of the approximation to the simulation result is shown as an example for various β values. For this example, $\lambda = 0.5\ calls/sec$, $\mu^{-1} = 90\ sec$ and $d_{setup} = 120ms$. As it is seen, the more close to one (zero) β is, the more aggressive (conservative) the approximation is. One important point is that there is a monotonous relation between delay budget (d_{budget}) and caching time (t_{cache}) as expected.

The important question at this point is "How do we determine N_{cache_limit}?" There are two steps in this process. First, we need to find the optimum cache duration t^*_{cache} by (4). By the term "optimum", we mean the unique t_{cache} value that is found from (4) for a given SVC setup latency requirement d_{budget}. The assumption here is that there is a reasonably accurate estimation of the call arrival rate λ. At this step, one should make a good judgement on SVC setup latency d_{setup} in ATM network as well. The estimation of d_{setup} depends on many factors such as the overall call arrival rate to the network (and its distribution therein), the network topology, the expected number of ATM switches to be involved in the call. The ATM switches have different latency figures for different call arrival rates they are exposed to. For instance, an ATM switch could have $10ms$ SVC setup latency for $50\ calls/sec$ and $30ms$ for $100\ calls/sec$. In practice, the ATM network is designed in such a way that the call arrival rate to a single ATM switch is kept below a required value. In addition, a constraint of maximum number of ATM switches for a call to traverse can be imposed in topology design. In light of these observations, there are many engineering concerns to be dealt with for the first step. In the second step, the probability $\pi(N_{cache\_limit})$ to run out of cachable SVCs (i.e., to hit the upper limit of the number of cached SVCs) is to be decided. Once again, $\pi(N_{cache\_limit})$ is an engineering parameter to be tuned. It is upto the network operator to decide on how frequently the cache limit N_{cache_limit} could be hit. After $\pi(N_{cache\_limit})$ is given, N_{cache_limit} can be found from Erlang-B formula. Note that λ and t^*_{cache} are known from the first step.

This simple approximation (4) along with the monotonous property emphasized above provides the main stepping stone of the adaptive algorithm, where the caching time (t_{cache}) is adaptively changed with the latency experienced in the ATM network and the call arrival rate. Every measurement interval ($n^{th}\ T_{MI}$) the caching time (t_{cache}) is calculated as in (5), where $\hat{\lambda}$ is the estimate of the mean call arrival rate, and $\hat{d}_{setup}$ is the estimate of the mean call setup delay in the ATM network.

$$t_{cache}(n) = \frac{1}{\beta\hat{\lambda}(n-1)} \log\left(\frac{\hat{d}_{setup}(n-1)}{d_{budget}}\right) \quad (5)$$

In (5), $\hat{\lambda}$ and $\hat{d}_{setup}$ are obtained by measurements and are filtered every T_{MI} as shown in (6). The parameter β in (5) is a predetermined constant between zero and one as explained before.

$$\begin{aligned} \hat{\lambda}(i) &= (1-w)\hat{\lambda}(i-1) + w\hat{\lambda}(i), \\ \hat{d}_{setup}(i) &= (1-w)\hat{d}_{setup}(i-1) + w\hat{d}_{setup}(i) \end{aligned} \quad (6)$$

The filtering operation is needed to increase the stability of the algorithm, hence, to reduce the effect of high frequency components in the measurements. The weight w in (6) determines the time constant of the low-pass filter. The bigger w is, the more responsive the algorithm is. If w is too large, the filter will not diminish the effect of transient changes in $\hat{\lambda}$ and $\hat{d}_{setup}$. On the other hand, the smaller w is, the more stable the algorithm is. In other words, if w is set too low, the algorithm responds too slowly to changes in the actual call arrival rate and call setup delay. In the analysis, w is equal to 0.1.

Note that the aim of the caching scheme is to keep the mean call setup latency (d_{post_cache}) below the requirement (d_{budget}). d_{post_cache} includes both cache hits (zero latency) and normal SVC setup latency (d_{setup}) when an SVC has to be set up through ATM cloud. The caching time t_{cache} found from (5) automatically gurantees that $d_{post\_cache} \leq d_{setup}$ due to (3), given that an appropriate β is used.

To summarize, every established SVC is kept alive (i.e. cached) for a duration of t_{cache} which is determined by (5). t_{cache} is adapted to the changes of mean call arrival rate (λ), and mean call setup latency (d_{setup}) in the ATM network by measuring both variables. Every end office's T-IWF carries out these procedures for every other terminating end office. When a new call request arrives and if there is already a cached SVC for the destined end office, the same SVC is utilized for this new call without the need to perform another SVC setup procedure. The algorithm chooses the oldest cached SVC, in case there are more than one cached SVC for the same destination.

One should note that for this scheme to work, every SVC should have a unique identification. The originating T-IWF should notify the terminating T-IWF of the identification of the cached SVC. The protocol to do that is out of the scope of this study.

III. Statistical Analysis

This section gives place to the simulation results obtained from the application of the SVC caching scheme defined in Section II to a realistic voice network found in a big metropolitan area.

In the simulations, we focus on a single end office and assume that call blocking probability in ATM network is zero. This assumption seems unreasonable at first. However, it is a realistic case especially when the carriers design their ATM networks to have zero-blocking capacity for VTOA applications.

The ATM cloud is a black box represented by a SVC setup latency distribution in this study. This simplification is necessary to avoid the simulation of every node in the network as well as PNNI routing protocol. In addition, the cross traffic for every other destination source pair should be simulated. As a result, the complexity could be huge, especially to simulate large metropolitan area networks with many end offices and with relatively large number of ATM switches in the broadband backbone. For this reason, the SVC setup latency experienced in ATM network is characterized by different distributions representing different load conditions. We experimented with Gaussian and Weibull distributions.

We assume that there are N_{EO} end offices. The aggregate call arrival rate to the end office of interest is distributed among the destination end offices uniformly. The uniform distribution is chosen to test the worst-case performance of the caching scheme. If, in fact, call requests focus on certain destinations (e.g. community of interest), the SVC caching scheme will perform better, that is, there will be more cache hits overall.

To measure the efficiency of the caching algorithm, we define a new performance metric ρ, which is given in (7). As seen from its definition, ρ is the ratio of average duration of SVCs utilized (carried voice traffic) to the total duration of SVCs utilized or cached (kept alive after the conversation is over). In (7), $m_{busy,i}$ ($m_{idle,i}$) represents the number of utilized (cached) SVCs for the i^{th} end office. For instance, if a SVC carries 80 second duration of traffic and then is cached idle for 20 seconds, the efficiency of this SVC is 80%. Obviously, the ideal condition is when $\rho = 1$. The closer ρ is to 1, the more successful the caching scheme is. In other words, ρ is the success measure of the caching scheme.

$$\rho(t) = \frac{\sum_{i=0}^{N_{EO}} \int_0^t m_{busy,i}(\tau)\, d\tau}{\sum_{i=0}^{N_{EO}} \left(\int_0^t m_{busy,i}(\tau)\, d\tau + \int_0^t m_{idle,i}(\tau)\, d\tau \right)} \quad (7)$$

To see the viability of the caching scheme proposed in Section II, we performed extensive simulations. By experimental study, we seek answers to the following questions. First of all, "Does the caching scheme provide the

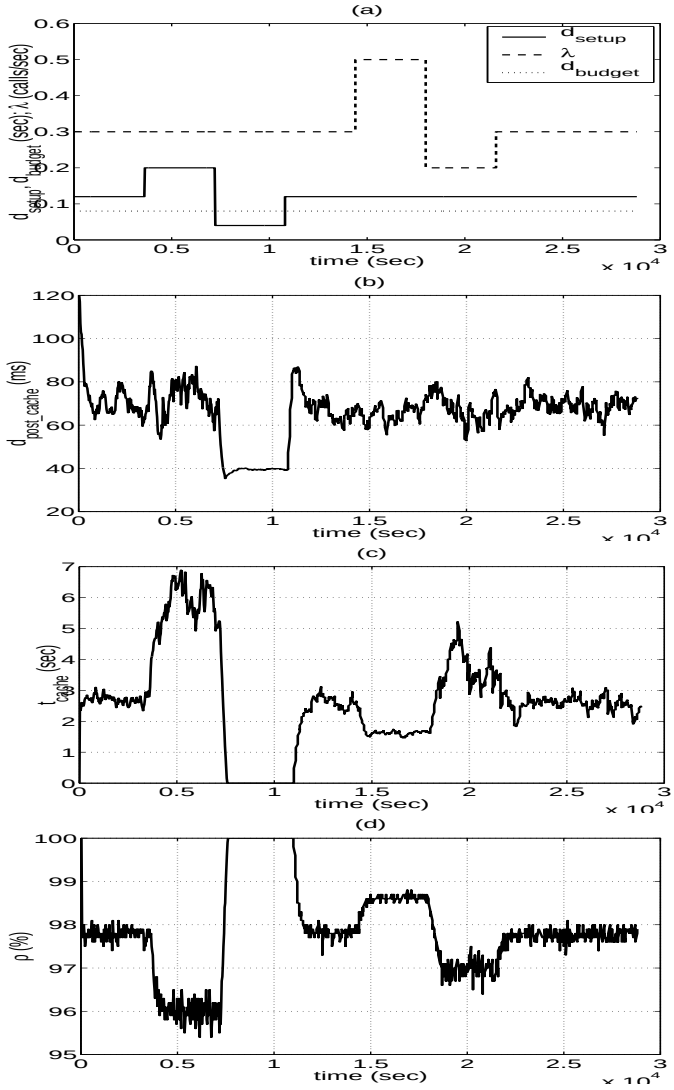

Fig. 6. Performance of the Caching Scheme: $N(d_{setup}, 5ms)$ for SVC Setup Latency

the ultimate goal of keeping the call setup latency below the required value?" Secondly, since online measurement is used for d_{setup} and λ, "Does it adapt to the changes in d_{setup} and λ?" The other important question is "How efficient is the scheme?"

In Fig. 6, the results of the first simulation scenario are shown. The tuning parameters of the caching scheme and the system parameters are as follows: $\beta = 0.5$, $w = 0.1$, $d_{budget} = 80ms$, $N_{EO} = 50$, $N_{trunk} = 4000$, $N_{cache\_limit} = 400$, and $\mu^{-1} = 90sec$. We set the measurement interval T_{MI} as $30sec$. T_{MI} is determined in such a way that there are sufficient number of measurement samples to make reasonable estimations on d_{setup} and λ. We consider Gaussian distribution ($N(d_{setup}, d_\sigma)$) for SVC setup latency in ATM network and Poisson call arrivals (λ). To address the second question posed above, we changed $N(d_{setup}, d_\sigma)$ and λ over time, which is depicted in Fig. 6a. As it is seen, while $\lambda = 0.3calls/sec$ (per each terminating end office), the evolution of $N(d_{setup}, d_\sigma)$ is as follows: $N(120ms, 5ms) \rightarrow N(200ms, 5ms) \rightarrow N(40ms, 5ms) \rightarrow N(120ms, 5ms)$. In the second part of the simulation, SVC setup latency distribution is kept as $N(120ms, 5ms)$, while call arrival rate λ is changed from $0.3calls/sec$ to $0.5calls/sec$, then from $0.5calls/sec$ to $0.2calls/sec$ and finally from $0.2calls/sec$ back to $0.3calls/sec$.

As illustrated in Fig. 6b, the caching scheme keeps the call setup latency (d_{post_cache}) below the requirement ($d_{budget} = 80ms$). During sudden changes in d_{setup} or in λ temporary violations occur. This can be overcome by using a smaller β than 0.5. In Fig. 6c, the evolution of t_{cache} is shown. Whenever the difference of $d_{setup} - d_{budget}$ increases or λ decreases, t_{cache} increases. As $d_{setup} - d_{budget}$ increases, the algorithm enlarges t_{cache} to improve the cache hits. Thus, the number of normal SVC setup through ATM network is reduced. As a result, the mean call setup latency d_{post_cache} is kept below d_{budget}. When λ decreases, t_{cache} has to be increased as well. This is necessary to keep the cache hits constant, as cache hits declines due to reduced λ, if t_{cache} is not augmented. The other noticeable observation from Fig. 6c is that $t_{cache} = 0$ when $d_{setup} < d_{budget}$. Obviously, when SVC setup latency in ATM network is smaller than the requirement, there is no use of caching scheme. The efficiency of the caching scheme is considerably high (e.g. bigger than 96%) as shown in Fig. 6d. From the evolution of t_{cache} and ρ, one should note that whenever t_{cache} increases ρ decreases or vice versa. The reason is that increase in t_{cache} means escalation in idle SVC durations on the average.

Throughout the simulation, we keep track of the number of cached SVCs in the system. The point is to see if the pre-determined N_{cache_limit} is sufficient. We observe that N_{cache_limit} is never reached. Moreover, the maximum number of cached SVCs stays below 200, which is much smaller than $N_{cache\_limit} = 400$. As a side note, we want to reiterate that, N_{cache_limit} is decided by considering the tradeoff between mean call setup delay requirement (d_{budget}) and efficiency of the caching scheme (ρ). If N_{cache_limit} is reached frequently, this indicates that more SVCs need to be cached in order to satisfy d_{budget} constraint.

The caching scheme constantly probes SVC setup latency (d_{setup}) in ATM network as well as call arrival rate (λ). Therefore, measurement errors may dampen the effectiveness of the algorithm. To test the effect of the mea-

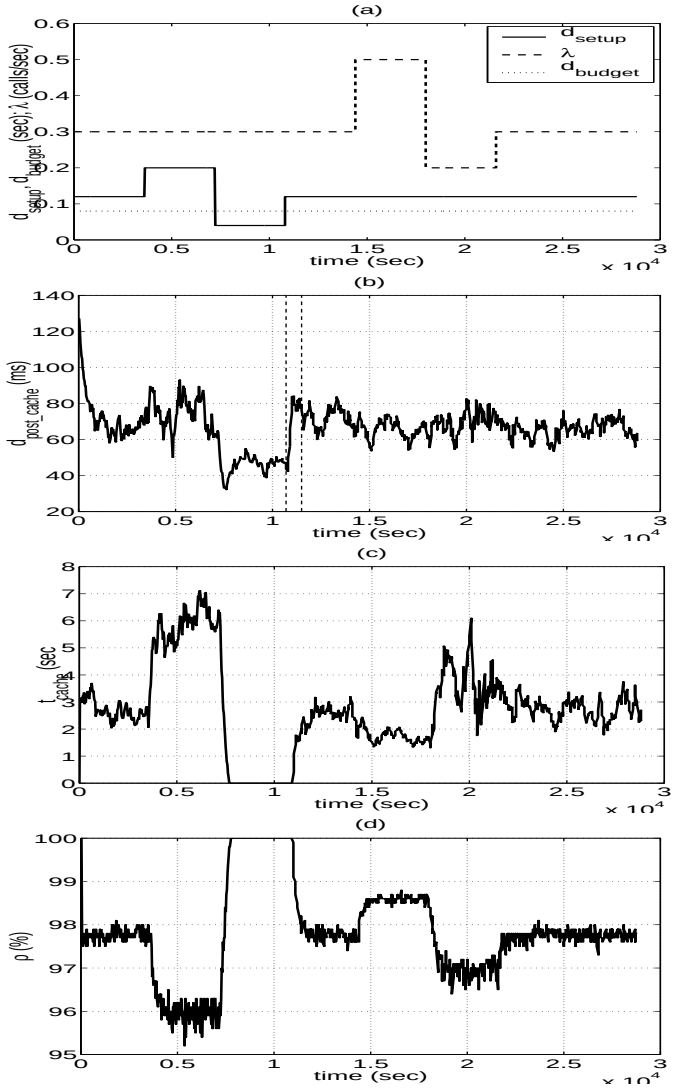

Fig. 7. Performance of the Caching Scheme: $N(d_{setup}, 50ms)$ for SVC Setup Latency

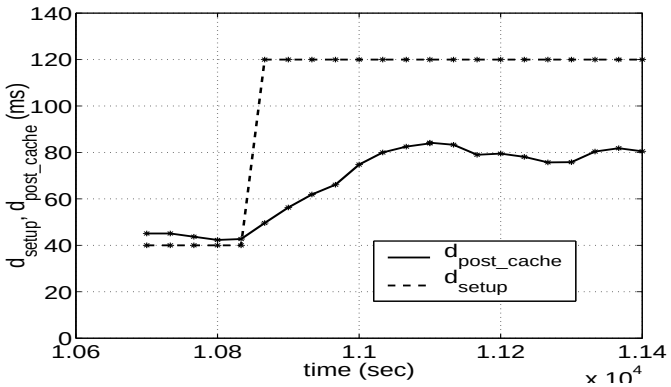

Fig. 8. Convergence Rate of the Caching Scheme

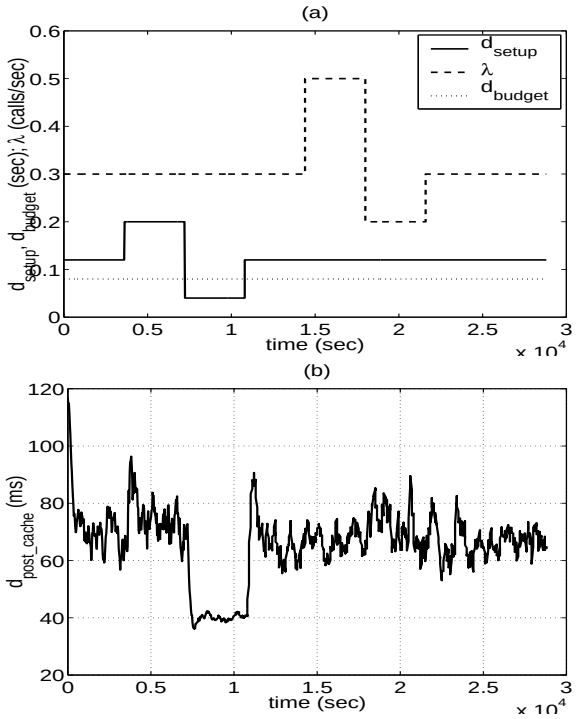

Fig. 9. Performance of Caching Scheme: Weibull Distributed SVC Setup Latency

surement errors on the performance of the algorithm, we increase the standard deviation of Gaussian distribution for SVC setup latency. While $d_\sigma = 5ms$ in the previous scenario, $d_\sigma = 50ms$ (ten fold increase) here. The rest of the parameters are the same as the first simulation scenario. As the simulation results shown in Fig. 7, the algorithm is quite robust, and increased variance almost has no effect on the performance. At this point, we should note that the size of the measurement interval T_{MI} has a great impact on the estimation of d_{setup} and λ. If T_{MI} is kept unreasonably small, there will be insufficient number of samples to make a satisfactory mean estimation.

The convergence time of adaptive schemes is an important criteria. To show how fast the adaptation of the algorithm is, the area between two vertical dashed lines in Fig. 7b is enlarged and is shown in Fig.8. In the figure, every point represents a measurement interval (T_{MI}), which corresponds to $30sec$. As depicted in the figure, upon sudden change in d_{setup} (from $40ms$ to $120ms$), d_{post_cache} reaches the requirement ($d_{budget} = 80ms$) in 6 steps, which means $180sec$. It is important to note that the convergence rate of the algorithm depends on many factors. The weight w used in filters, the measurement interval T_{MI} are first ones to consider. Obviously, w could be made bigger or a smaller duration for T_{MI} could be used to increase the convergence rate. However, one should keep in mind that, the former leads to instability (oscillations) due to increased sensitivity to transient changes, whereas the latter has the same effect because of insufficient statistics collection.

Up to now, we made use of Gaussian distribution for SVC setup latency in ATM network. Next, we carry out a simulation to observe the effect of Weibull distribution. In this scenario, $\beta = 0.5$, $w = 0.1$, $d_{budget} = 80ms$, $N_{EO} = 50$, $N_{trunk} = 4000$, $N_{cache\_limit} = 400$, $\mu^{-1} = 90sec$, and $T_{MI} = 30sec$. We changed mean and standard deviation (d_{setup}, d_{σ}) of the Weibull distribution and λ over time, which is depicted in Fig. 9a. As it is seen, while $\lambda = 0.3calls/sec$ (per each terminating end office), the evolution of (d_{setup}, d_{σ}) is as follows: $(120ms, 43.6ms) \rightarrow (200ms, 72.7ms) \rightarrow (40ms, 14.5ms) \rightarrow (120ms, 43.6ms)$. In the second part of the simulation, $d_{setup} = 120ms$ and $d_{\sigma} = 43.6ms$, while call arrival rate λ is changed from $0.3calls/sec$ to $0.5calls/sec$, then from $0.5calls/sec$ to $0.2calls/sec$ and finally from $0.2calls/sec$ back to $0.3calls/sec$. As seen in Fig. 9b, performance of the caching scheme is consistent with the previous observations made for Gaussian case. This is actually an expected result. Our algorithm is based on mean estimation and does not depend on distribution. The important condition here is to select an appropriate measurement interval T_{MI}.

The simulation results show that there is a tradeoff between efficiency ρ of the caching scheme and SVC setup latency d_{setup} of ATM network with respect to delay budget d_{budget} allocated. That is, $d_{setup} - d_{budget}$ is the important factor to determine the efficiency ρ of the scheme. The bigger $d_{setup} - d_{budget}$ is, the less efficient the scheme is. Intuitively, the caching scheme has to increase the caching duration (hence, the number of cached connections) in order to meet the small delay requirement. As we elaborated before, the delay budget will highly depend on the processing capacity of the ATM switches. Thus, the efficiency also distinguishes the call processing performance of the ATM switches.

Let us now convey over this message by an example. In the example, we have three kinds of ATM switches. Each ATM switch has different SVC setup latency. Consequently, the ATM network consisted of these switches will have different SVC setup latency. The assumption is that the SVC setup processing delay in ATM network with the first kind of switch is $80ms$ (mean), with the second one is $120ms$, whereas with the third one is $200ms$. That is, the first ATM switch has a good SVC setup performance, and the third one has a poor call processing performance. In the simulations, the total call arrival rate is $10\ calls/sec$, and again the calls are distributed uniformly to the destination end offices, where there are 100 of them. We obtained the efficiency for each $d_{budget} \in [5ms, 200ms]$. The results, that are shown in Fig. 10, can be interpereted in

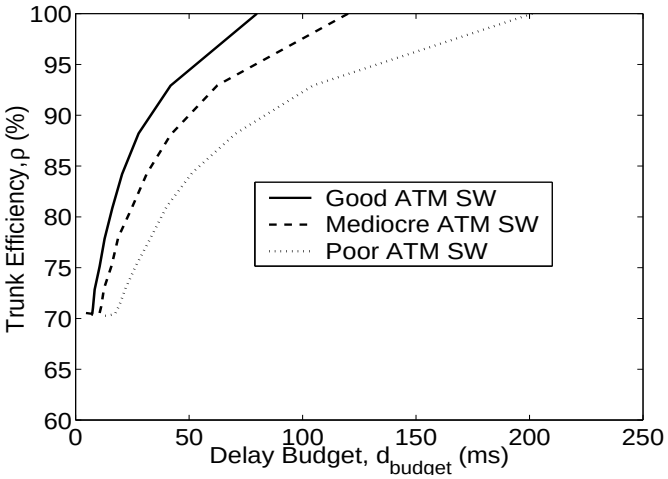

Fig. 10. SVC Caching Tradeoff: Efficiency vs Delay Budget

two ways. First, for a given delay budget, one can find the maximum reachable efficiency with the corresponding ATM switch. Second, for a required efficiency, the delay budget, that should be allocated, can be determined. For instance, when the delay budget is $50ms$, the efficiency of the caching scheme with good, mediocre, and poor ATM switches in the backbone is 94%, 90%, and 84% respectively. On the other hand, for the target efficiency of 95%, the delay budget allocations for good, mediocre, and poor ATM switches should be $55ms$, $80ms$, and $135ms$ in sequence. One should notice that the efficiency stabilizes beyond a certain delay budget value. For instance, the efficiency for the poor ATM switch case remains almost constant for the delay budget of smaller than $20ms$. Actually, for a fixed call arrival rate, there will always be cache hits beyond a delay budget value, no matter how small it gets. This is because the bigger $d_{setup} - d_{budget}$ gets, the bigger t_{cache} becomes to satisfy the d_{budget} requirement. For very small d_{budget} values (d_{setup} is fixed), t_{cache} becomes so big that the efficiency (ρ) turns out to be insensitive to d_{budget} due to sustained cache hits.

IV. Discussions

In this paper, we focus on SVC setup latency in ATM network. Obviously, one immediate solution could be to construct an overlay PVP (Permanent Virtual Path) network in the ATM backbone, for only end points of VPs require call processing and transit nodes do not involve in the establishment of the SVCs. The design of VP networks has been well studied and there are many proposed optimization algorithms in the literature. However, the efficient management of VP networks is still a challenging task practically. Although constructing elastic VPs [11],

which resizes itself with the changing traffic conditions, is a promising solution, there is no standard way of changing the capacity of VPs automatically. Moreover, the carriers do not want to commit to proprietary solutions. Besides, the SVCs are rerouted automatically by PNNI routing protocol without interference from the management system in case of failures in ATM network. For management and operations purposes, this feature makes the SVCs highly appealing to the carriers. On the other hand, as we posed in Section I, a SVC caching scheme is necessary especially for ATM backbone with ATM switches that have big SVC latency figures.

We have two choices for SVC caching: horizontal (time dimension) scheme and vertical (space dimension) scheme. In the horizontal scheme, which is the way we pursue in this study, the caching duration is the controling parameter. As for the vertical scheme, depending on the estimation of the call arrival rate, adaptive number of SVCs, n_{cache}, are pre-established ready to be used. Hence, n_{cache} could also be adaptively adjusted with the changing call requests, as time proceeeds. Actually, both approaches would provide similar results, as adaptation of t_{cache} and n_{cache} would intuitively have the same effect on SVC setup latency. Hence, as n_{cache} increases, mean SVC setup latency decreases.

The drawback of the vertical scheme is the lack of decomposability. One can analyze the vertical scheme by constructing a Markov Chain (with the assumption of Poisson arrivals and Exponential holding times) where the state is represented by (number of connections, number of pre-established connections) tuples. Since the Markov Chain is not decomposible, the only way to adjust n_{cache} with the changing traffic conditions is to do numerical analysis on the newly constructed Markov Chain as λ (call arrival rate) estimations change over time. By doing so, an appropriate n_{cache} can be found according to the call arrival rate measurements. Therefore, it is hard, if not impossible, to find a simple explicit momotonic relation between the number of pre-established SVCs (n_{cache}) and SVC setup latency experienced in the network. Additionally, this approach bears a high processing burden for practical realizations. As a result, we choose to adaptively adjust the caching duration (i.e., horizontal scheme).

The explicit monotonic relation between call setup latency and the caching time (4) assisted us to derive a simple mechanism to adapt the caching time (5) which tracks the traffic (call arrival rate) and network (call processing load of the network) conditions. The simplification mentioned in Section II facilitated to derive this revealing relation. However, the monotonous property is an intuitive one. In the absence of this facilitating explicit relation, other highly effective adaptive schemes could be used. For instance, Least-Mean-Square algorithm ([5]) is a good candidate.

In this study, as repeated in several places, the focus is to satisfy the mean cross-office delay requirements set for the TDM voice networks. However, the $95th$ and $5th$ percentile values (assuming Gaussian distribution) are also described in the standards. For instance, $5th$ percentile value shows that there will be 5% call clipping (impatient hang-ups), in which case the network resources are wasted. These requirements can be incorporated to the caching scheme described in this study as well. One obvious approach could be to take the most stringent requirement (i.e., $5th$ percentile) into consideration instead of the mean. In that case, all the requirements would be met. Clearly, an appropriate measurement interval should be selected in order to have sufficient number of delay samples in order to validate the Gaussian distribution assumption for the cross-office delay. The tradeoff here is the efficiency. The carriers can change the target requirement (mean or $5th$ percile or $95th$ percentile) as the real clipping measurements become available. Thus, the requirement can also be an engineering parameter to be tuned.

V. Conclusion

In this paper, a simple adaptive SVC caching scheme is defined for the VTOA applications. The motivation is based on the observation that SVC establishment through an ATM cloud might take longer than the specifications dictated in the standards of today's voice networks.

Call processing capacity in the ATM network is treated as a scarce resource. Hence, the idea is to recycle already established SVCs more than once. To do so, a delayed release of a SVC mechanism is used. In this scheme, a SVC is not torn down after the users stop the conversation (hang up), instead it is to be kept alive for an adaptive duration (caching time), hoping that there would be another call request to the same destination. Thus, call processing for a new SVC establishment is eliminated.

We devised a simple monotonic relation between caching time and mean call setup latency. By exploiting this dependence, we come up with an adaptation scheme for the caching time. In the algorithm, the mean call arrival rate as well as mean call setup latency in the ATM network is measured constantly to determine the appropriate caching duration in order to meet the requirement of

the mean call setup latency. The stability and efficiency of the algorithm are examined with extensive simulation study.

REFERENCES

[1] D. Bertsekas and R. Gallager, *Data Networks*, Second Edition, Englewood Cliffs, NJ, Prentice-Hall, 1992.

[2] M. S. Chambers, H. Kaur, T. G. Lyons, and B. P. Murphy, "Voice over ATM," *Bell Labs Tech. Journal*, Vol. 3, No. 4, Oct-Dec 1998.

[3] E. Felstaine, R. Cohen, and O. Hadar, "Crankback Prediction in Hierarchical ATM Networks," *Proceedings of INFOCOM'99*, New York City, March 1999.

[4] GR-1364-CORE, "LSSGR: Switch Processing Time Generic Requirements, Section 5.6," *Bellcore Generic Requirements*, Issue 1, June 1995.

[5] S. Haykin, *Adaptive Filter Theory*, Third Edition, Upper Saddle River, NJ, Prentice-Hall, 1996.

[6] Q.766, "Specifications of Signaling System No.7 ISDN User Part," *ITU-T Recommendation*, Geneva, March 1993.

[7] T. So, R. Wilson, and K. Caves, "Signaling Requirements for Interconnecting PSTNs over ATM," *ATM Forum Contribution 98-0843*, Nashville, Dec. 1998.

[8] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton, NJ, Princeton University Press, 1994.

[9] K. K. Whang, W. Fink, B. Krishnamurthy, I. Lin, D. M. Rouse, and H. V. Sorensen, "Voice over PacketStar Gateway Solution for Service Provider Networks," *Bell Labs Tech. Journal*, Vol. 3, No. 4, Oct-Dec 1998.

[10] D. J. Wright, "Voice over ATM: An Evaluation of Implementation Alternatives," *IEEE Communications Magazine*, Vol. 34, No. 5, May 1996.

[11] J. Yan, "Adaptive Configuration of Elastic High-Speed Multiclass Networks," *IEEE Communications Magazine*, Vol. 36, No. 5, May 1998.