Reserving for Future Clients in a Multipoint Application—Why and How?

Pratyush Moghé, *Student Member, IEEE*, and Izhak Rubin, *Fellow, IEEE*

***Abstract*—As applications become more sophisticated, we believe that the problem of guaranteeing their service commitment will get much harder. Currently applications requiring real-time media quality of service (QoS) are set up using a *flow-reservation* model. In this model, the network admits each flow (or connection) within an application separately using RSVP, asynchronous transfer mode (ATM) signaling, or some other mechanism. This model is restrictive because it cannot inherently deliver guarantees on future client additions to an ongoing application. We claim that such guarantees are going to be required to build sophisticated multipoint and multiparty applications that involve a dynamic number of clients and a real-time sharing between clients. Without such guarantees, existing clients within an application can be subjected to context disruption or unreliable communication. To reduce this form of degradation, we propose the idea of *application reservation* where resources for an entire application are reserved at application setup. This includes resources for the clients expected to add to the application in future.**

Application reservation can be implemented as an overlay to existing flow-based reservations. In this paper, we present the architectural features of such a framework. As part of the service interface, applications declare space-time dynamics of future client arrivals in addition to traditional descriptors. A new class of QoS measures that quantifies the type of degradation mentioned above is also conveyed. The application manager checks to see if the application can be admitted. After the application is admitted, the manager monitors clients as they arrive, to ascertain if they conform to the declared service interface parameters. Conforming clients have resources reserved for them, so they can be simply added to the application. We review these architectural components and address the nontrivial problem of managing the nonconforming clients.

***Index Terms*—Application reservation, ATM, flow-based reservations, internet application management, RSVP.**

I. Introduction

As applications grow sophisticated [1], we believe that the problem of guaranteeing their service commitment will become much harder. Currently, there is an intensive effort in trying to guarantee media-level quality of service (QoS) to flows [or connections in the asynchronous transfer mode (ATM) world] within applications. Thus far, each application has been treated as a conduit of flows, where a flow represents resources required by a client[1] within the application. In this model, flows are assumed to have independent service requirements and consequently independent fates. For instance, almost all the reservation schemes discussed in the literature focus on reserving flows, one at a time [2]–[5]. This approach might be fine for conventional applications involving a fixed set of clients. It is quite restrictive, however, when used to implement an application where clients can join or leave the application during its lifetime, and where clients depend on each other for mutual information exchange. With pure flow-based reservations, there is no guarantee that new clients can add to an ongoing application. If there is insufficient bandwidth at the instant a client requests to add, the flow admission procedures will reject it. Note that this does not affect just the client that wishes to add to the application. Since the application is an entity shared by other clients, some of the existing clients may depend on the adding client. By blocking the adding client, these existing clients may experience service degradation in the form of context disruption, or worse. If an existing client depends critically on the blocked client, say for information transfer, it may even abort the application. To reduce the likelihood of this type of service degradation in applications that manifest it, we have recently proposed the idea that application managers should be able to reserve bandwidth for clients that are expected to add in future to the applications [6]. This is tantamount to reserving for an entire application (*application reservation*) as opposed to reserving in units of flows (*traditional flow-based reservation*). The paradigm of reserving applications can be applied to several client-server based services, conferences, collaborations, distributed computation and simulations, etc.

In this paper, we discuss some of the engineering problems that arise in implementing application reservation. First, what constitutes the service interface for such applications? That is, how are such applications specified? What additional information does the interface have to convey in order to have an entire application reserved? If the admission of applications is done analogous to flow-based reservations, we also require to declare to the network the maximum tolerable service degradation. In fact, this raises another question. How is this service degradation quantified? Recall that this type of service degradation is experienced by new clients (when they are blocked) as well as by existing clients in the form of context

Manuscript received January 23, 1996; revised August 13, 1996.

P. Moghé was at the University of California, Los Angeles, when this work was done. He is now with the Network and Service Management Research Department, Bell Laboratories, Holmdel, NJ 07733 USA.

I. Rubin is with the Department of Electrical Engineering, University of California at Los Angeles, CA USA.

Publisher Item Identifier S 0733-8716(97)02276-2.

[1]Throughout this paper, we refer to "client" as a participant in an application. This usage is not restricted to the "client-part" of a client-server interaction. A client could be any process or human entity that requires communication resources.

disruption. Hence, its quantifying metrics are distinct from traditional metrics such as delay, jitter, or packet loss. To reinforce the distinction, we call the new type of service degradation *client-level degradation.* Second, assuming the service interface is designed, how are these applications admitted? Third, it is obvious that an *a priori* specification of future client arrivals is imprecise. In that case, some of the client arrivals may conform to the service interface specification and some may not! Naturally we need to figure out which do and which do not. Algorithms that classify client arrivals as conforming or nonconforming are referred to as *conformance-definition* algorithms. Such algorithms need to be designed. In the first part of the paper, we briefly review a set of solutions to these three problems [6]. As far as possible, these solutions reuse the techniques developed for handling similar problems at the flow layer.

We then focus on the problem of deciding what to do with the nonconforming client arrivals. For brevity, we refer to these clients as *violators.* This problem turns out to be crucial. Since the service interface parameters are likely to be imprecise, it is possible that a substantial number of future client arrivals may be classified as violators. If these violators are not treated carefully, the network can end up with the same level of client-level degradation it set out to reduce, and in addition, incur the higher overhead of application reservation. We consider two broad solutions to manage the violators. In the first scheme, each violator is simply blocked, under the premise that it does not deserve an entry because it violates the contract. In the second scheme, each violator spawns a new application request. The spawned application is admitted if there is sufficient bandwidth in the network. We show through analysis that the first scheme, while seemingly fair, is sensitively dependent on the design of the conformance definition as well on the precision of the service interface. To compare these schemes, we use them to implement application reservation in the presence of uncertainty in the parameters of the service interface. An application is assumed to specify an absolute worst-case client-level degradation metric via its service interface. The network has to ensure that the application perceives a client-level degradation better than this metric. We assume simplistically that the uncertainty in the service interface parameters is modeled by a random variable (rv) of known distribution. Our main result is as follows. We show that, under certain conditions, the network can guarantee the appropriate levels of client-level degradation to the application with both the schemes but at different costs. The first scheme can deliver guarantees by inflating some of the service interface parameters in order to reserve more bandwidth than otherwise required. It is thus wasteful in terms of reserved bandwidth. The second scheme requires additional admission procedures since each violator spawns an independent application request. This causes a significant increase in the setup messages exchanged and hence increases the network traffic. The total cost of implementing application reservation depends on the reserved bandwidth as well as on the setup network traffic. We formulate a simple optimization problem, where the objective is to minimize the total implementation cost subject to a feasible client-level degradation. Under certain conditions, we show that a hybrid scheme that shares some features of both these schemes can minimize the overall implementation cost.

Note that we do not claim that the particular methodology we present for application reservation is the only way to solve the problem. This is just the first step toward understanding the issues. The important point made in this work is that application reservation is a generalization of flow-based (or connection-based) reservations. It can benefit sophisticated applications by ensuring reliable and low-latency joins of future clients. Given that most networks will likely implement some form of flow reservation, application reservation should be designed as an overlay to the flow reservation schemes. With this in mind, we have presented a framework that comprises four steps. The first step is the conceptual design of the service interface. The second step deals with the resource allocation problem within the network. The third step involves the design of the conformance-definition algorithms. The final problem is to make the framework robust in the presence of uncertainty in the service interface specification. This last step is the focus of this paper. This particular problem does not appear to have easy general answers and its design impacts all the aspects of the framework—including the perceived client-level degradation and the overhead of implementation. What we have presented here is a possible way to solve this problem under a restricted set of assumptions.

The paper is organized as follows. Section II reviews the first three problems of application reservation posed above. In Section III, we analyze different schemes to manage the violators. In Section IV, we synthesize a scheme that minimizes the cost of application reservation under certain conditions. We conclude after discussing the related work in the literature in Section V.

II. Application Reservation

We now present some specific solutions to the problems posed in the earlier section. We do not make strong assumptions about the type of network [Internet protocol (IP) or ATM or any other kind] but we do assume that there is a mechanism to guarantee media-level QoS through flow (or connection) reservations. We begin by discussing the elements of the service interface in Section II-A. We review the admission strategy in Section II-B. In Section II-C, we outline a method (called *conformance definition*) to classify new clients that arrive to an ongoing application. New clients are classified either as conforming clients or as violators.

A. Service Interface

Applications that require application reservation are assumed to declare information through the service interface. This information can be classified into *descriptors* and *QoS measures.* Fig. 1 illustrates a possible interface. Application descriptors consist of *client-level descriptors* and *flow descriptors.* The idea is that client-level descriptors capture space-time dynamics of future client arrivals and the dependence between the clients. Flow descriptors quantify the traffic model for each flow in the application. Application QoS measures consist of

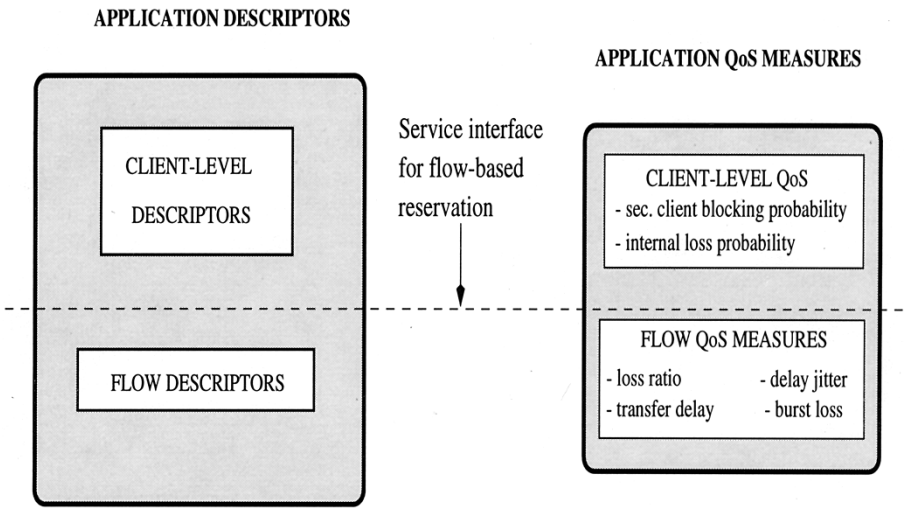


Fig. 1. Application service interface.

client-level QoS measures and *flow QoS measures.* Client-level QoS quantifies worst-case acceptable client-level degradation described in Section I. We will motivate two client-level QoS measures in the next section. Flow QoS measures represent the worst-case acceptable degradation of media within a flow. Note that the application service interface encompasses the service interface used for just flow-based reservations (lower shaded half of Fig. 1).

1) Client-Level Descriptors: We now outline a candidate list of client-level descriptors first proposed in [7]. These are motivated by the fact that when an application is being set up, information about the clients *present* in the application at that time may be known completely. However, the application may not know deterministically *when* or *where* future clients might add to it. Thus, the service interface assumes stochastic space-time descriptors for their arrival. These descriptors also convey information about the *binding* between existing and future clients. Since the basic assumption is that clients add to an ongoing application in order to participate in a shared information transfer, blocking a new client can affect the service of an existing client. This dependence of service between existing and future clients is captured by client-level binding.

For convenience, we call the clients present during application setup as *primary* clients and the future clients as *secondary* clients. Without loss of generality, we assume that the application knows the superset of locations from where secondary clients can add. These locations are represented by a set of access nodes $\Im$. These access nodes could be end-routers in an IP network or enterprise switches in an ATM network. Given that clients can join and leave the application any time during its evolution, we assume the application terminates when all its clients leave. We now outline a list of client-level descriptors.

1) Number of primary clients at each access node i, $i \in \Im$.
2) Mean client holding time, $1/\mu$.
3) Maximum number of clients simultaneously carried in the application, $K_{\max}$.
4) Minimum lock-out time, T_{lo}, the initial time interval when no secondary clients are expected to arrive into the application.
5) Parameters for the space-time dynamics of secondary-client arrivals
   a) maximum batch size of global arrivals,[2] $BA_{\max}$. That is, no more than $BA_{\max}$ clients are expected to arrive into the application simultaneously;
   b) peak rate of global batch arrivals, λ_p, where we define $T_p = 1/\lambda_p$;
   c) minimum time interval between global bursts of batches, T_I. In general, secondary-client batches can arrive in a bursty fashion. A burst is a succession of global batches, where consecutive batches arrive T_p time-units apart. T_I is the minimum time interval between consecutive bursts;
   d) spatial parameter, p^i ($0 \leq p^i \leq 1$), is the fraction of global secondary clients that arrive at access node i. $\sum_{i \in \Im} p^i = 1$.
6) Visibility information between clients. Visibility captures the information flow between clients. For each source client within the application, it establishes the set of receiving clients.
7) Binding between existing and future clients is quantified by a binding descriptor, expressed simply in the form of a matrix $\underline{\gamma} = \{\gamma(i, j), i, j \in \Im\}$. Parameter $\gamma(i, j)$ represents the binding between clients of the application that are incident at access nodes i and j; it is interpreted as the probability with which an existing client at node j gets affected critically due to secondary-client blocking

[2] There are two constructs worth noting here. First, the notion of *global* and *local* secondary-client arrivals. Secondary clients that arrive at a particular access node i are referred to as local arrivals at node i. The aggregated secondary-client arrival stream to the entire (or global) application object constitutes global arrivals.

The *batch* construct is used to model situations where secondary clients can join in groups that are distributed arbitrarily among the access nodes. We assume that near-simultaneous arriving secondary clients are grouped into a batch.

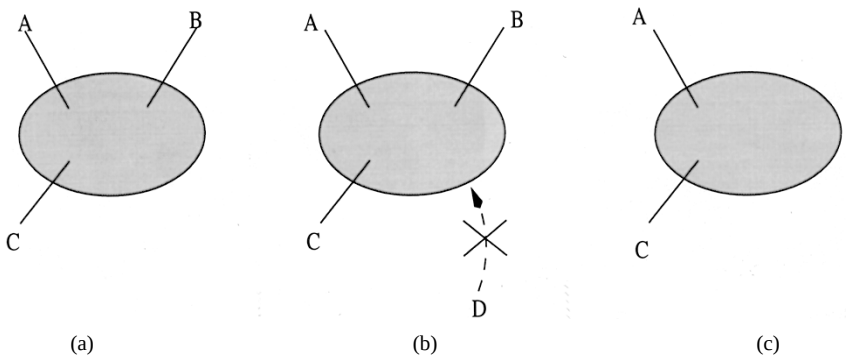


Fig. 2. Client-level degradation could manifest as internal loss: (a) application set up between clients A, B, and C, (b) secondary client D arrives later. Network blocks it from adding to the application, and (c) existing client B's service degrades critically, and it aborts the application. This is called "internal loss" of client B.

at node i. (This will be further clarified in Section II-A2.)

Let us now consider an example of a practical application that can be specified using these descriptors.

Example: Consider a conferencing application (with strong client-level binding) where a set of moderators begins the conference. After an initial testing period, a group of clients is invited. A collaborative session is then completed, whereupon a few clients depart. Depending on the outcome of the session (i.e., with some probability), a fresh set of clients is invited to join the existing set of clients. After completing another collaborative session, all the clients leave and the application ends.

In our model, the clients constituting the moderator set are called primary clients. Clients that add on later (in two groups) are secondary clients. The minimum duration specified for the initial testing period is the lock out time T_{lo}. The maximum client-size of the groups joining is $BA_{\max}$. The minimum duration between the arrival times of the groups is T_I (this is the minimum duration expected for the first collaborative session). The mean holding time is $1/\mu$. The spatial parameter p^i is estimated simply as the ratio of the mean number of secondary clients expected at access node i to the total number of secondary clients expected. This parameter does not capture the temporal correlation between the collaborative sessions but is purposely kept simple enough. More accurate parameters may be added to improve the characterization. The advantage of our simple model is that it is scalable. If more groups were to arrive at unexpected locations, this would manifest as a distorted spatial parameter (as long as the locations lie in the set of access nodes). Note that applications being considered in this paper are assumed to exhibit a strong intensity of client-level binding. For such applications, we believe it is reasonable to assume that all the likely access nodes are known in advance.

2) Client-Level QoS Measures: Thus far, we have not mentioned how we quantify client-level degradation. To motivate this, consider Fig. 2. Let us assume that a shared application is set up between clients A, B, and C based on the resources they require (as determined by traditional flow reservation schemes). After a while, when client D arrives and requests to be added to the application, sufficient bandwidth might not be available to accommodate it. If this happens, client D gets blocked. Obviously, blocking of client D is a measure of the client-level degradation of the application. In general, we represent such blocking by a measure called *secondary-client blocking*, and define it as the probability that a secondary-client arrival to an ongoing application gets blocked because of lack of available network bandwidth. Secondary-client blocking is clearly a candidate client-level QoS measure.

Client-level QoS measures could also comprise additional metrics that are more subjective than blocking probability. For instance, one or more of the existing clients A, B, and C could get affected by blocking of client D. Whether or not an existing client is critically affected depends on the intensity of client-level binding between the client and the adding client. In Fig. 2, we indicate that client B gets critically affected and aborts the application. We call this phenomenon *internal loss.* Internal loss probability is the probability that an existing client aborts an application due to the blocking of a new client. Since internal loss probability quantifies client-level degradation in applications, it is also a candidate client-level QoS measure.

B. Admission

Assuming an application specifies the descriptors and QoS measures, the application manager implements an end-to-end admission scheme that checks if the required amount of resources are available. If they are, it reserves bandwidth for secondary clients as long as they arrive in accordance with the declared client-level descriptors. Otherwise, it rejects the application request. For lack of space, we do not discuss the admission method. This is covered in detail in [6], where an admission scheme is designed to work with traditional flow-based admission schemes.

C. Classifying Secondary Clients

Assuming an application is admitted, the network reserves bandwidth for its secondary clients as long as they arrive according to the declared client-level descriptors. Clients that conform to the descriptors can be simply added to the application without undergoing an admission check. Clients that do not conform are called violators. Algorithms (called

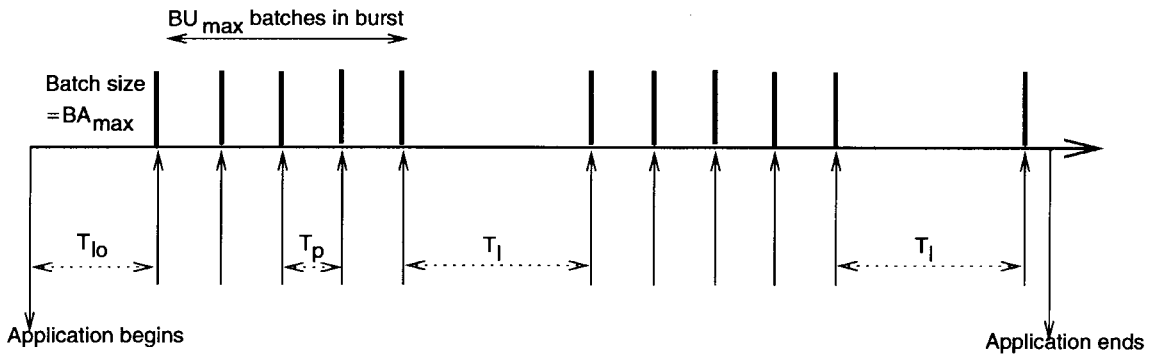


Fig. 3. Worst-case global secondary-client arrival process.

conformance-definition algorithms) are used to distinguish between conforming clients and violators. We now discuss the design of these algorithms in some detail because this impacts the problem of managing the violators.

Conformance definition is developed in two stages—1) a worst-case arrival model (global and local) is developed for the secondary client arrivals and 2) leaky-bucket based algorithms are designed to identify if the actual arrivals "exceed" this worst-case arrival model.

1) Worst-Case Arrival Model: The worst-case *global* arrival model is shown in Fig. 3. Secondary clients arrive globally as batches of $BA_{\max}$ clients each. The maximum number of consecutive batches arriving at peak rate $1/T_p$ is the burst size $BU_{\max}$. Between consecutive bursts is a spacing interval of T_I time units. The parameter $BU_{\max}$ can be derived in terms of the remaining client-level descriptors.

From the model in Fig. 3, we can compute the following parameters for global secondary-client arrivals in the worst-case:

$$\text{Sustained global batch rate } \lambda_s = \frac{BU_{\max}}{T_p \cdot (BU_{\max} - 1) + T_I} \quad \left(T_s \triangleq \frac{1}{\lambda_s}\right)$$

$$\text{Sustained global batch size } BA_s = BA_{\max}$$

$$\text{Sustained global client rate } CA_s = BA_{\max} \cdot \lambda_s.$$

To characterize secondary-client arrivals at access node i, we define the following *local* parameters:

$$g^i = P\{\text{batch arrives at node } i \mid \text{batch arrives globally}\}$$
$$\lambda_s^i = \text{Sustained batch rate at node } i$$
$$BA_s^i = \text{Sustained batch size at node } i \text{ (w.r.t. global arrivals)}$$
$$\overline{BA}_s^i = \text{Sustained batch size at node } i \text{ (w.r.t. batches that arrive at node } i)$$
$$CA_s^i = \text{Sustained client rate at node } i.$$

Assuming each secondary client within a batch arrives independently at access node i, the local parameters can be readily computed:

$$g^i = 1 - (1 - p^i)^{BA_{\max}}$$
$$\lambda_s^i = \lambda_s \cdot g^i = \frac{BU_{\max} \cdot g^i}{T_p \cdot (BU_{\max} - 1) + T_I}, \quad \left(T_s^i \triangleq \frac{1}{\lambda_s^i}\right)$$
$$BA_s^i = BA_{\max} \cdot p^i$$
$$\overline{BA}_s^i = \frac{BA_{\max} \cdot p^i}{g^i}$$
$$CA_s^i = \lambda_s^i \cdot \overline{BA}_s^i.$$

Using a result proved recently [8], the worst-case local arrival model at access node i can be shown to be a periodic, deterministic, on–off process. During the *on* period, secondary clients arrive as batches, with $BA_{\max}$ clients in each batch. The burst size BU^i is determined so that the sustained secondary-client arrival rate matches the computed value of CA_s^i. That is, BU^i has to satisfy

$$\frac{BA_{\max} \cdot BU^i}{T_p(BU_{\max} - 1) + T_I} = CA_s^i.$$

Thus

$$BU^i = \left\lceil \frac{CA_s^i \cdot \{T_p(BU_{\max} - 1) + T_I\}}{BA_{\max}} \right\rceil.$$

2) Conformance-Definition Algorithms: The overall conformance definition procedure is indicated in Fig. 4. An arriving global secondary-client batch is first tested to see if admitting it violates the parameter $K_{\max}$. The global batch is then checked for conformance with respect to the global batch algorithms. The global batch algorithms are of two types—global batch-rate algorithm and global batch-size algorithm. The global batch-rate algorithm checks to see if an arriving global batch is conforming with respect to the worst-case declared arrival rate of global batches. That is, if an arriving global batch arrives "too soon," it is declared to be nonconforming, and all its component local batches are also declared to be nonconforming. The basic algorithm that defines this conformance policy is a leaky-bucket algorithm called *secondary-client batch-rate algorithm*

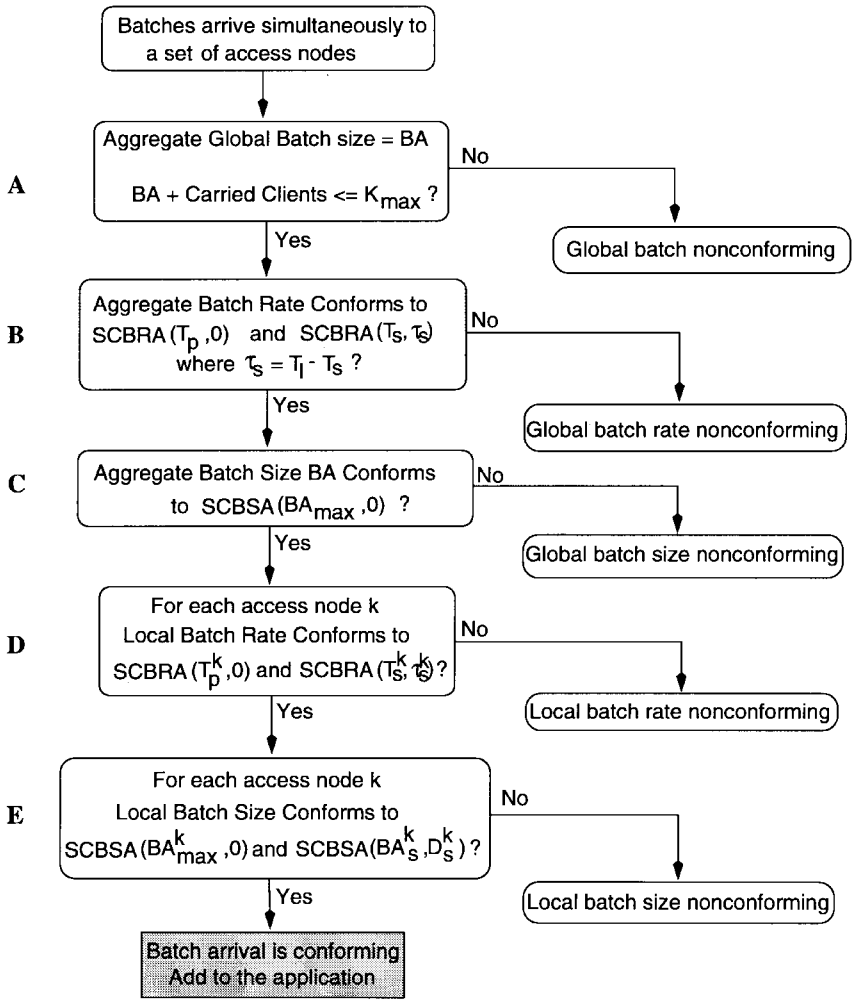


Fig. 4. Conformance-definition procedure.

(SCBRA). The global batch-rate algorithm is made up of a pair of SCBRA algorithms—one that checks for peak rate and another for the sustained rate. The global batch-size algorithm checks to see if an arriving global batch is conforming with respect to the worst-case declared global batch size. If an arriving global batch is "too large," it is declared to be nonconforming, and all its component local batches are also declared to be nonconforming. The basic algorithm that defines this conformance policy is a leaky-bucket algorithm called secondary-client batch-size algorithm (SCBSA).

If a global batch is conforming with respect to its arrival rate as well as its batch-size, its component local batches are checked for conformance. This means that, at each access node, the local batch arrival is checked by an additional set of batch-rate and batch-size algorithms. These algorithms are referred to as local batch-rate and batch-size algorithms. These algorithms are also based on the SCBRA and SCBSA algorithms. An arriving batch that is declared to be conforming both globally and locally is added to the application without an end-to-end admission procedure. Both the SCBRA and SCBSA algorithms are applications of the basic leaky-bucket schemes proposed elsewhere for access control [9]. Fig. 5 illustrates their flow-charts. SCBRA (T, τ) is a virtual scheduling or continuous-state leaky bucket algorithm that operates with an interval parameter T and a limit parameter τ. Fig. 5(a) illustrates the use of this algorithm to define the conformance of ith arriving secondary-client batch depending on the *time* $t_a(i)$ it arrives. SCBSA(B, D) also works with two parameters, an increment parameter B and a limit parameter D. The algorithm presented in Fig. 5(b) detects if the ith arriving secondary-client *batch-size,* $N_{BA}(i)$, is conforming with respect to these parameters.

The global and local checks in Fig. 4 are designed based on these algorithms and based on the worst-case arrival model derived in Section II-C1. The details of this design are outside the scope of this paper, and may be found in [6].

III. Managing the Violators

Having designed techniques to identify the violators, the question remains: What do we do with the violators? The simplest strategy is to block (reject) them. We call this *hard-decisioning.* While this scheme seems to be fair (violators do not adhere to the descriptors declared by the application at setup), it can be tricky to implement since descriptors for space-time dynamics of secondary clients are rarely precise. Thus, a substantial number of secondary clients can be declared as violators. Hard decisioning causes all

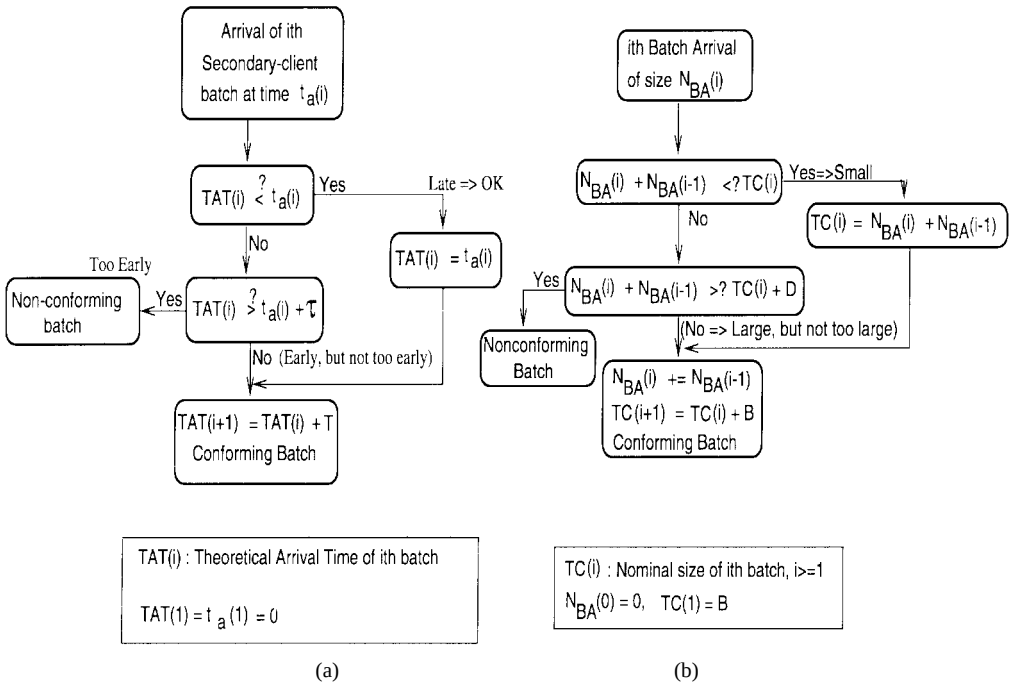


Fig. 5. SCBRA (T, τ) and SCBSA (B, D) algorithms.

such violators to get blocked. As mentioned earlier, blocking secondary clients could also induce existing clients to abort the application. Clearly, hard-decisioning can degrade the client-level QoS seriously. In Section III-A, we examine analytically the impact of uncertainty in the client-level descriptors on the client-level QoS under hard-decisioning. In Section III-B, we discuss an alternative scheme where each violator's addition request is treated as an application independent of the ongoing application. In Section IV, we compare the costs involved in implementing both these schemes.

A. Hard-Decisioning Scheme

In this section, we wish to evaluate the effect of a small deviation in the client-level descriptors on the client-level QoS measures, secondary-client blocking ($\Phi_{\text{sec-blk}}$) and internal loss probability ($\Phi_{\text{in-loss}}$), under hard-decisioning. To introduce deviation in the descriptors, we assume that the *declared* spatial parameter p^i for access node i, is not necessarily equal to the *actual* spatial parameter, p^i_{new}, where $\sum_{i\in\Im} p^i_{\text{new}} = 1$. The remaining descriptors are assumed to be correct[3] and the application is assumed to abide by them. Thus all the global secondary-client arrivals are in accordance with the worst-case global secondary-client arrival model in Fig. 3 and are declared conforming by boxes A, B, and C of the procedure in Fig. 4. The local secondary-client arrivals can, however, be nonconforming with respect to the local batch-rate and batch-size algorithms (boxes D and E in Fig. 4). The reason for choosing the spatial parameter for deviation is that it is a crucial descriptor that governs the space-time dynamics of secondary-client arrivals. It is also the descriptor that is least likely to be known with certainty. We now present the key analytical idea in determining how frequently the local secondary clients are declared to be nonconforming, as a function of the deviation in the spatial parameter. The details are described in [10].

Fig. 6 indicates the approach used. A particular access node i is selected. An infinite population model is assumed for the secondary-client arrivals. The local batch-rate algorithm is modeled as a single *rate leaky-bucket,* while the local batch-size algorithm is modeled as a single *size leaky-bucket.* An arriving secondary-client batch is nonconforming if it is declared to be nonconforming by at least one of the two leaky buckets. All the clients belonging to a nonconforming batch are considered nonconforming and get blocked. The contents of the two leaky buckets are modeled as a joint Markov chain, and its stationary distribution is obtained. The client-level QoS measures, $\Phi_{\text{sec-blk}}$ and $\Phi_{\text{in-loss}}$, are derived from this distribution using Markov renewal theory. The details can be found in the Appendix. In the next section, we discuss the numerical results based on this analysis.

1) Numerical Results: We now illustrate, with the help of an example, the effect of deviation in the client-level descriptors on the derived client-level QoS measures. Consider an application with three access nodes 1, 2, and 3. Suppose that the corresponding spatial parameters (p^1, p^2, p^3) are declared to be (0.5, 0.3, 0.2). Further suppose that the actual spatial parameter exceeds the declared spatial parameter only at access node 3, and that this excess is made up for by reducing the spatial parameters symmetrically across access nodes 1 and 2. Let DEV be the fractional deviation of the spatial parameter at node 3. That is, $DEV = (p^3_{\text{new}} - p^3)/p^3$.

Fig. 7 illustrates the effect of DEV on secondary-client blocking at each access node. The descriptor values are listed in Fig. 7. As expected, blocking at node 3 ("P_sec_blk at node

[3]We neglect the possibility that an arriving client can be nonconforming *because* it arrives at a node outside the access node set. This possibility can be reduced by extending the set of access nodes to include all plausible nodes.

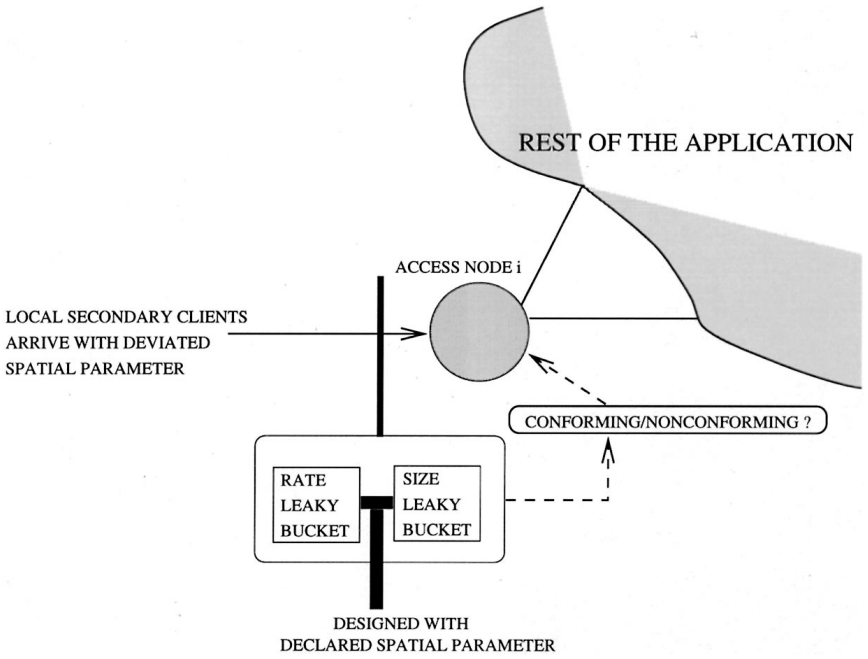


Fig. 6. Setup used to evaluate hard-decisioning at access node i.

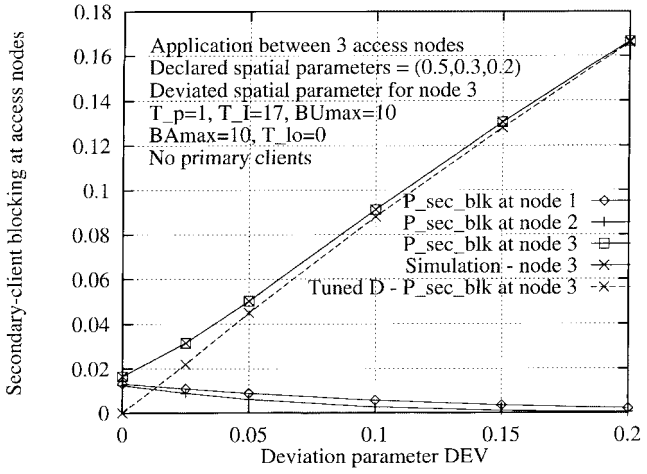


Fig. 7. Effect of deviated spatial parameter on $\Phi^{i}_{\text{sec-blk}}$.

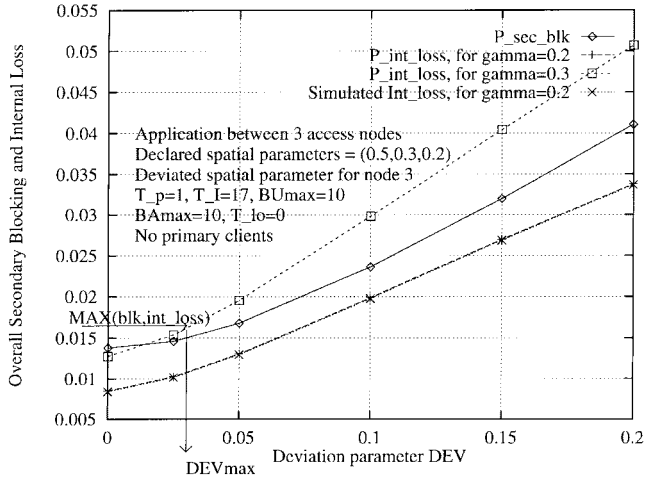


Fig. 8. Effect of deviation on $\Phi_{\text{sec-blk}}$ and $\Phi_{\text{int-loss}}$.

3") is a linear function of DEV for sufficiently large values of DEV. When DEV is small, however, the conformance-definition algorithm tends to unfairly block arrivals at all the access nodes. For example, blocking is positive even when the deviation is zero. The reason for this is that the limit parameter, D_s^i, that is used to design the local batch-size algorithm, has been sized assuming infrequent bursts. D_s^3 can be tuned till this behavior is corrected. Fig. 7 shows this as the broken curve "Tuned D-P_sec_blk at node 3." However, tuning is tricky in general, since D_s^i is restricted to integer values. In Fig. 7, we also compare the secondary-client blocking with an untuned D_s^3 as calculated analytically and as determined by simulation. The two plots are seen to match fairly well, and this verifies our analytical methodology.

Fig. 8 examines the deviation on the overall secondary-client blocking and internal loss in the application. We assume an untuned limit parameter. The internal loss probability is observed to be a strong function of deviation as well as the binding descriptor. Two values for the binding descriptor [$\gamma(i, j) = 0.2$ and 0.3, between all nodes] are considered, and as expected, the internal loss scales linearly with the binding descriptor. As indicated, Fig. 8 can be used to calculate the largest acceptable deviation $DEV_{\max}$ for a given service interface. This value of deviation (on the x-axis) corresponds to the lower of the maximum tolerable secondary blocking and internal loss values (on the y-axis).

We can summarize the analysis in this section as follows: *Under hard-decisioning, the client-level QoS is sensitive to the deviation in the client-level descriptors as well as to the design of the conformance-definition algorithms.*

B. Spawning Applications

An alternative method to deal with a violator is to model it as a part of a *new* application. That is, each violator x that

arrives to an application $\mathcal{A}$ "spawns" a request for a separate application $\mathcal{A}_x$. $\mathcal{A}_x$ models the resources required by client x so it can communicate with other clients. $\mathcal{A}_x$ is referred to as a *spawned application;* it is distinct from the application $\mathcal{A}$. Client x is admitted provided the resources required by the spawned application are available. Note that the spawned application is potentially quite complex, since it involves not just the existing clients when x arrives, but also the future clients that x wants to communicate with. Thus the spawned application can be just as complex as the original application; it has primary and secondary clients of its own.

Our objective in the next section is to compare hard-decisioning and the spawned application schemes based on their resource costs, assuming imprecise client-level descriptors. The modeling and admission strategies for spawned applications are treated in more depth in [10].

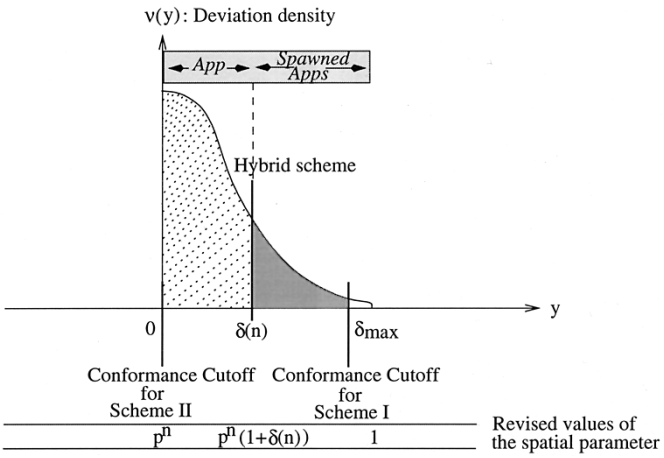


Fig. 9. Deviation and conformance cutoff.

IV. Optimal Implementation

We would now like to compare the hard-decisioning and spawned application schemes by evaluating their implementation cost. As mentioned earlier, all realistic applications suffer from uncertain or deviated descriptors. Let us assume that an application requires stringent client-level QoS guarantees even when its descriptors are deviated. For simplicity, let us assume that only the local spatial parameters are deviated. We restrict our attention to a single access node n. Let $\Phi^n_{\max}$ represent the largest acceptable value of secondary-client blocking probability at access node n. This value is assumed to ensure that the internal loss is less than the worst-case acceptable internal loss.

Let us assume simplistically that the excess[4] deviation in the local spatial parameter (p^n) is represented by a rv of known density function ν. Fig. 9 illustrates a possible density function. At zero deviation, the spatial parameter corresponds to the declared value p^n. At deviation y, the corresponding revised value of spatial parameter is $p^n_{\text{new}} = p^n(1+y)$. The maximum possible value of deviation, $\delta_{\max}$, is obtained by setting $p^n_{\text{new}} = 1$. That is, $\delta_{\max} = (1-p^n)/p^n$. Let us examine the use of hard-decisioning and spawned application schemes to implement application reservation.

1) *Scheme I*
   This approach is the straightforward application of hard-decisioning. We overallocate the spatial parameter so as to reduce the probability that an arriving secondary client is nonconforming—all violators are blocked (as in hard-decisioning). To make this idea concrete, let us define the deviation corresponding to the cutoff between the conformers and the violators as the *conformance cutoff* point $\delta(n)$. In Scheme I, the conformance cutoff is set close to $\delta_{\max}$. In particular, the smallest cutoff $\delta(n)$ that keeps the secondary-client blocking below $\Phi^n_{\max}$ is chosen. The spatial parameter declared by the service interface is thus modified to the revised value: $p^n_{\text{new}} = p^n[1+\delta(n)]$. All subsequent violators get blocked. No spawned applications are required.
2) *Scheme II*
   Another possible approach is to apply the spawned application scheme. The idea is retain the value of the spatial parameter declared by the service interface, p^n. Thus, the conformance cutoff point $\delta(n)$ is set equal to zero. Violators are set up as spawned applications. Note that it is acceptable if a small fraction of these violators are blocked, as long as the secondary-client blocking at node n stays below $\Phi^n_{\max}$.

In comparing[5] these strategies, we note that Scheme I possibly allocates flows that are never used. This scheme is thus wasteful in terms of the cost of reserved bandwidth. On the other hand, in Scheme II, almost every violator spawns an application. Implementing a spawned application is expensive in terms of both reserved bandwidth and the setup cost (also called signaling cost) involved in the end-to-end admission procedure. If a spawned application reuses the flows allocated to the original application, the cost of reserving bandwidth can be reduced. However, the setup costs (these include the computational costs) incurred in the end-to-end admission procedure remain, and can be considerable. Thus, a comparison between Schemes I and II involves a tradeoff between the cost of reserving bandwidth and the cost of setup. We define a "minimum-cost" scheme as one that minimizes the combined cost of reserving bandwidth and setup when implementing the application and the subsequent spawned applications, subject to the constraint that the secondary-client blocking probability be less than $\Phi^n_{\max}$. In the next section, we show that a hybrid scheme that shares some features of both these schemes can be tuned to achieve just such a condition.

A. Selecting Conformance Cutoff

Let us construct a hybrid scheme designed around a particular value of conformance cutoff point $\delta(n)$, where $\delta(n) \in$

[4]For simplicity, we ignore the possibility of negative deviation. Our cost analysis holds if a bimodal density function is used to model positive and negative deviation in the spatial parameter.

[5]We ignore the increased likelihood of the application being denied admission in Scheme I, or the possibility of a spawned application being blocked in Scheme II because of lack of available bandwidth. We also neglect internal loss—a reasonable assumption if it affects both the schemes.

$[0, \delta^{\max}]$. The service interface is implemented using the revised spatial parameter $p^n_{\text{new}} = p^n[1 + \delta(n)]$. All arrivals that are conforming with respect to p^n_{new} have flows reserved for them at application setup. Most violators can spawn applications. A small number can be blocked, as long as the secondary-client blocking probability stays below $\Phi^n_{\max}$. It is easy to see that the hybrid scheme tends to Scheme I when $\delta(n)$ increases toward $\delta^{\max}$, while it tends to Scheme II when $\delta(n)$ reduces to zero. Refer to Fig. 9.

We now compute the cost of implementing this scheme as a function of the conformance cutoff point $\delta(n)$. We then pose the question: is there an optimal cutoff, $\delta^*(n)$, that minimizes the cost of implementation? If so, such a scheme is the *minimum-cost* scheme.

1) Cost Formulation as a Function of $\delta(n)$: Before we formulate the costs, we need to define a link-level component of a flow called *flowlet.* A flow has an end-to-end meaning; typically it is tagged by a unique combination of source and receiver addresses and ports. A flowlet is associated with a particular link, and it represents the resources required by the flow on that link. Thus, a flow can be thought of as the cocatenation of its flowlets.

Bandwidth costs, $C_{bw}[\delta(n)]$ and $C^{SA}_{bw}[\delta(n)]$: We define $C_{bw}[\delta(n)]$ and $C^{SA}_{bw}[\delta(n)]$ to be the number of flowlets required by the application and by all the spawned applications, respectively, as a function of $\delta(n)$. We assume that the flow between a source client and its receiving clients is routed in the form of a single multicast tree rooted in the access node of the source client.

$$\begin{aligned} C_{bw}[\delta(n)] &= \text{Number of flowlets carried in the application due to conforming secondary clients that exceed spatial parameter } p^n \\ &= \text{Secondary-client arrival rate} \times \text{Mean holding time} \times \text{Number of flowlets on the source multicast tree rooted in node } n \\ &= \lambda_s\, BA_{\max}\{p^n[1+\delta(n)] - p^n\} \cdot \frac{1}{\mu} \cdot L^n \\ &= \lambda_s\, BA_{\max} p^n\, \delta(n)\, \frac{1}{\mu}\, L^n. \end{aligned}$$

Here, L^n denotes the number of flowlets required by a secondary client at node n. L^n can be approximated by the total number of links on the source multicast tree rooted in node n. That is, $L^n = 1 + \sum_{h \geq 2} L^n_h$, where $L^n_h =$ Number of links at a distance of h hops from node n on its source multicast tree.

Before we can derive the bandwidth cost for the spawned applications, we need to compute what fraction of violators can be blocked. Clearly, the maximum acceptable blocking of an arbitrary secondary client arrival at node n is $\Phi^n_{\max}$. Also, $\int_{y=\delta(n)}^{\delta_{\max}} \nu(y)\, dy$ represents the fraction of secondary-client arrivals at node n that are nonconforming. Since conforming secondary clients are always admitted, the maximum acceptable blocking of violators is $\Phi^n_{\max} / \int_{y=\delta(n)}^{\delta_{\max}} \nu(y)\, dy$. We will assume that the actual blocking of violators is set equal to this maximum value since this clearly reduces the overall cost.

$$\begin{aligned} C^{SA}_{bw}[\delta(n)] &= \text{Aggregate number of flowlets required by all the spawned applications} \\ &= \text{Number of spawned applications} \times \text{Number of flowlets in each spawned application} \\ &= \text{Rate of spawning applications} \times \text{Mean holding time of spawned application} \times L^n \\ &= \left[1 - \frac{\Phi^n_{\max}}{\int_{y=\delta(n)}^{\delta_{\max}} \nu(y)\, dy}\right] \cdot \int_{y=\delta(n)}^{\delta_{\max}} \{\lambda_s BA_{\max} \times p^n[(1+y) - (1+\delta(n))]\, \nu(y)\}\, dy \times \frac{1}{\mu} \times L^n \\ &= \left[1 - \frac{\Phi^n_{\max}}{\int_{y=\delta(n)}^{\delta_{\max}} \nu(y)\, dy}\right] \cdot \lambda_s BA_{\max} p^n \cdot L^n \cdot \frac{1}{\mu} \times \int_{y=\delta(n)}^{\delta_{\max}} [y - \delta(n)]\nu(y)\, dy. \end{aligned}$$

Setup cost, $C_{sig}[\delta(n)]$ and $C^{SA}_{sig}[\delta(n)]$: We define $C_{sig}[\delta(n)]$ and $C^{SA}_{sig}[\delta(n)]$ to be the number of setup flowlets required to implement the application and all the spawned applications, respectively, as a function of $\delta(n)$.

Suppose that the setup cost is dominated by the computational cost of the end-to-end admission procedure and the communication cost of carrying setup messages. Then, we can ignore the setup cost incurred by the conforming secondary clients in an application, since they only require to be allocated flows, without requiring an end-to-end admission procedure. Let us assume simplistically, that $4L^n$ setup flowlets are required to exchange setup messages that set up and tear down an application. That is, $C_{sig}[\delta(n)] = 4L^n$, is assumed to be independent of $\delta(n)$. This is a very simplistic modeling of the setup cost; more sophisticated models may be used alternatively [13].

In the worst case, each spawned application also requires $4L^n$ setup flowlets. Thus

$$\begin{aligned} C^{SA}_{sig}[\delta(n)] &= \text{Number of spawned applications} \times 4L^n \\ &= \left[1 - \frac{\Phi^n_{\max}}{\int_{y=\delta(n)}^{\delta_{\max}} \nu(y)\, dy}\right] \cdot \lambda_s BA_{\max} \cdot p^n \cdot \frac{1}{\mu} \times \int_{y=\delta(n)}^{\delta_{\max}} [y - \delta(n)]\nu(y)\, dy \times 4L^n. \end{aligned}$$

Overall cost $C[\delta(n)]$: If α and β are the weights for the application flows and for the setup flows, respectively, the overall cost C is formulated as

$$C = \alpha C_{bw}[\delta(n)] + \alpha C^{SA}_{bw}[\delta(n)] + \beta C_{sig}[\delta(n)] + \beta C^{SA}_{sig}[\delta(n)].$$

Note that the routing (or switching) costs are ignored because they are a function of the actually required resources, not a function of what is allocated at setup and are hence independent of $\delta(n)$. Substituting the components, neglecting the terms independent of $\delta(n)$, and canceling the multiplier $\lambda_s B A_{\max} p^n L^n/\mu$, we have

$$C[\delta(n)] = \alpha\delta(n) + (\alpha+4\beta)\cdot\left(1 - \frac{\Phi^n_{\max}}{\int_{y=\delta(n)}^{\delta_{\max}} \nu(y)\,dy}\right) \times \int_{y=\delta(n)}^{\delta_{\max}} p[y-\delta(n)]\nu(y)\,dy. \quad (1)$$

Clearly, an optimal $\delta(n)$, if it exists, is one that minimizes $C[\delta(n)]$, where $\delta(n) \in [0, \delta_{\max}]$.

2) Minimum-Cost Scheme—Example: We consider a particular case where the deviation is distributed according to a truncated exponential distribution. That is, density function $\nu(y) = \sigma/(1-e^{-\sigma\delta_{\max}})e^{-\sigma y}$, where $y \in [0, \delta_{\max}]$, and $\sigma > 0$. To simplify matters, we also impose the condition that $\Phi^n_{\max} = 0$, i.e., we assume that no secondary-client blocking is tolerated. Such a specification can be used by applications with very stringent client-level QoS requirements.

Equation (1) can now be rewritten as

$$C[\delta(n)] = \alpha\delta(n) + (\alpha+4\beta) \times \int_{y=\delta(n)}^{\delta_{\max}} [y-\delta(n)]\,\frac{\sigma e^{-\sigma y}}{1-e^{-\sigma\delta_{\max}}}\,dy.$$

After integration and simplification, and again discarding the terms that are independent of the variable $\delta(n)$, we obtain the cost function

$$C[\delta(n)] = \left[\alpha + \frac{(\alpha+4\beta)\,e^{-\sigma\delta_{\max}}}{1-e^{-\sigma\delta_{\max}}}\right]\delta(n) + \frac{\alpha+4\beta}{1-e^{-\sigma\delta_{\max}\sigma}}\,e^{-\sigma\delta(n)}.$$

Let R denote the cost-ratio β/α. R is a measure of the relative costs of a single setup flowlet versus application flowlet. Expressing C in terms of R we have

$$C[\delta(n)] = \left(1 + \frac{1+4R}{1-e^{-\sigma\delta_{\max}}}\,e^{-\sigma\delta_{\max}}\right)\cdot\delta(n) + \frac{1+4R}{1-e^{-\sigma\delta_{\max}\sigma}}\cdot e^{-\sigma\delta(n)}. \quad (2)$$

Here, the cost $C[\delta(n)]$ has been redefined in units of setup cost α. Note that C is a strictly convex[6] function of $\delta(n)$ over the interval $[0, \delta_{\max}]$. We can very quickly find the optimal point $\delta^*(n)$ with the following procedure.

1) Find a stationary point $\delta^s(n)$ that solves the equation $[d/d\delta(n)]\,C[\delta(n)] = 0$, $\forall\delta(n) \in \Re$. A closed form expression for the root is $\delta^s(n) = -(1/\sigma)\ln(1 + 4R\,e^{-\sigma\delta_{\max}})/(1+4R)$.

[6]If $\Phi^n_{\max} > 0$, $C[\delta(n)]$ is not necessarily convex. In this case, a modified Newton method [14, pp. 105, 186–187] or a bisection method [15, pp. 293–302] can be used to find a local minimum.

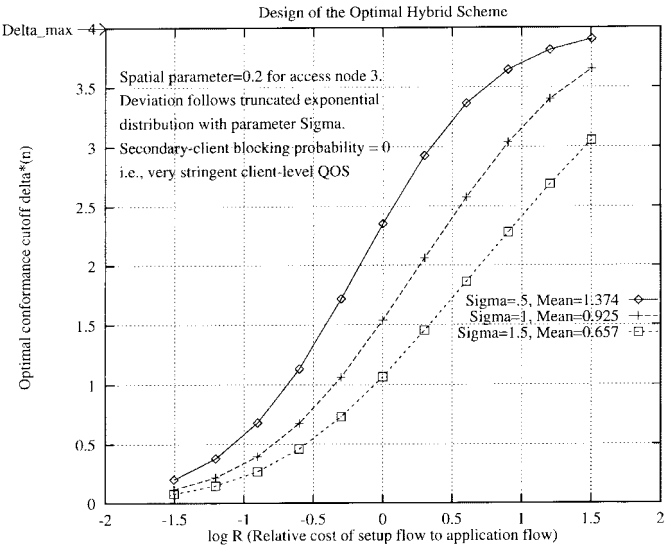


Fig. 10. Optimal implementation for the hybrid scheme.

2) a) If $\delta^s(n) \notin [0, \delta_{\max}]$, one of the extremum points 0, or $\delta_{\max}$, is clearly the solution. Then, if $C(0) < C(\delta_{\max})$, $\delta^*(n) = 0$, else $\delta^*(n) = \delta_{\max}$.
   b) Else $\delta^*(n) = \delta^s(n)$.

Fig. 10 illustrates the optimal cutoff $\delta^*(n)$ as a function of the logarithm of R and the parameter σ. As expected, for a fixed value of R, the optimal cutoff increases as σ reduces or equivalently as deviation density falls off less rapidly. For a small value of σ ($\sigma = 0.5$), the optimal cutoff is seen to be most sensitive to changes in R when $\log R$ is close to zero. This is the region where R is close to unity, that is, the relative costs of application flows and setup flows are of the same order. The optimal cutoff is seen to be a weak function of R, when the relative costs are skewed significantly.

V. Related Work

We are not aware of any published work that addresses the problem of application reservation in our context. Recently there has been some work in the area of *advance reservations* [16]–[20]. The idea is to extend the current limitation of flow-based reservations [5] where flows have to initiate an admission procedure just *before* they become active. With advance reservations, flows can reserve resources for a future duration. This problem has a strong flavor of "booking" to it. Advance reservations have been shown to work with predictive service type [21], [22] as well as with guaranteed service. Note that fundamentally this problem is quite different from ours, though its implementation may involve some similar techniques. The key difference is that the advance reservations are still at the *flow layer* unlike our work which is constructed on an *application basis.* Since advance reservations treat flows as isolated entities, they cannot model client-level degradation or guarantee client-level QoS to applications. In fact, advance reservations can be thought of as a special case of application reservation, where the application consists of a single flow corresponding to a secondary client wanting to book resources.

VI. Summary

As applications become more sophisticated, their "QoS expectations" also increase in terms of sophistication. To clarify this, consider the current breed of applications being deployed in the Internet. The QoS expectations of most applications involve the reliable, real-time delivery of media flows. Considerable effort has been applied in order to specify and guarantee the flow QoS; recent developments reassure us that this problem is essentially close to being solved by a combination of protocols. The next breed of applications on the internet will demand *flow QoS and more.* Examples of such applications are conferencing, groupware (collaborations), distributed computation, and many other custom client-server based applications. These applications do not necessarily have a fixed number of clients; clients can join and leave applications during their evolution. For a truly seamless information exchange, the network should be able to guarantee that once an application begins, all the clients that wish to add to the application can indeed do so. Without such a guarantee, future clients can get blocked. This degrades the service from the viewpoint of future clients because they cannot participate in the application at the "precise" moment they need to. In addition, by blocking such clients, the network also risks degrading the service for the existing clients in the application. In some of these applications, the existing clients can depend on the future clients for a certain aspect of the service. By blocking these adding clients, the existing clients can suffer context disruption or worse. If the existing clients critically depend on the blocked clients, and their service is no longer meaningful, they can abort the application. This particular type of service degradation that adding and existing clients undergo is very different from the traditional flow degradation. We call it *client-level degradation* to emphasize the distinction. Current flow-reservation protocols can guarantee a cap on flow degradation by ensuring a threshold on flow QoS measures like loss, delay, jitter etc. However, they cannot guarantee a threshold on client-level degradation. To enable such guarantees, we have proposed the concept of *application reservation* where the network reserves resources for an entire application and not just for isolated flows.

In this paper, we outline an architectural framework for application reservation. The central idea is to implement application reservation as an overlay to existing flow-based reservation. The framework comprises the following four parts.

1) In the first part, we design the service interface for application reservation. The service interface specifies the application's resource demand via client-level descriptors and flow descriptors. Client-level descriptors express when and where new clients are expected to add to the application, and how clients depend on each other. Flow descriptors quantify the traditional flow traffic model. The service interface also specifies the application's expected quality of service via client-level QoS and flow QoS measures. Client-level QoS measures quantify the worst-case acceptable client-level degradation, while the flow QoS measures quantify the flow-level degradation for each existing client within the application.
2) The application manager checks to see if this resource demand is available, and if so, admits the application.
3) After the application is admitted, new clients that arrive are monitored by algorithms to see if they conform to the declared client-level descriptors. If they do, they are simply added to the application without running an admission check. If they do not, they are declared to be *violators.*
4) We design a strategy for managing the violators. It is crucial to design this carefully, otherwise the entire framework for application reservation can cause more harm than good. (Since the client-level descriptors are expected to be imprecise or uncertain, there could be many violators.) We study two extreme strategies. In the *hard-decisioning* scheme, the violators are blocked under the premise that they do not deserve an entry since they violate the contract. In the *spawned application* scheme, each violator spawns an independent application request that models the resource requirements from its viewpoint alone. The violator is admitted if there are sufficient resources for the spawned application. When the service interface is imprecise, the first scheme wastes reserved bandwidth, while the second one incurs large setup overhead in terms of network traffic. We design a hybrid scheme that shares some of the features of these two schemes. Under certain conditions and for a known characterization of the uncertainty in the client-level descriptors, this scheme can be tuned to achieve the minimum cost of implementation.

Appendix
Deriving Client-Level QoS

Suppose that a particular access node i is selected. We assume that time is slotted into slots of duration T_p. Also, we approximate the conformance-definition algorithm parameters T_I, T_s^i, τ_s^i (defined earlier) to be integer multiples of T_p. Let one cycle of *on* and *off* periods of worst-case global secondary-arrivals constitute a *frame* of length N_f. Then $N_f = [T_I + (BU_{\max} - 1) \cdot T_p]/T_p$ slots. The actual probability with which a local batch arrives at any slot in the on period is $g_{\text{new}}^i = 1 - (1 - p_{\text{new}}^i)^{BA_{\max}}$.

Define the binomial operator $\mathcal{B}()$, where $\mathcal{B}(p, n, k) \triangleq \binom{n}{k} p^k (1-p)^{n-k}$. Let ξ_j^i be the probability that a local batch of size j arrives at node i at any slot in the on period, and let $\xi_{j,\text{batch}}^i$ denote the probability of the same event, conditioned on at least one client arrival at node i. Obviously, $\xi_j^i = \mathcal{B}(p_{\text{new}}^i, BA_{\max}, j)$, for $0 \le j \le BA_{\max}$ and $\xi_{j,\text{batch}}^i = \xi_j^i/(1 - \xi_0^i)$, for $1 \le j \le BA_{\max}$.

A. Local Conformance-Definition Algorithms

Since the global descriptors of the application are *correct*, we can ignore the peak-rate SCBRA. The remaining SCBRA can be equivalently [9] thought of as a rate leaky-bucket (RLB). The parameters of the leaky-bucket can be expressed in terms of the SCBRA parameters. Let us define two parameters,

$I = T_s^i/T_p$, and $L = \tau_s^i/T_p$. We then define the discrete-time process $R = \{R_n, n = 0, 1, \cdots\}$ over the discrete state space $= \{0, 1, \cdots, I + L - 1\}$, where R_n represents the size of the RLB leaky-bucket at time index n, not including a possible batch arrival at n. RLB evolves according to the following two rules—1) For a local batch arrival at time n, if $R_n \leq L$, the batch is *conforming* with respect to rate, and R_n increments by an amount I, otherwise the batch is *nonconforming* with respect to rate, and R_n remains unaltered. 2) RLB contents decrement by one every slot as long as $R_n > 0$.

Analogous to the above description, the local batch-size algorithm can be represented by a single size leaky-bucket (SLB). The SLB is represented by two parameters: $B = BA_s^i$, and $D = D_s^i$. We define the discrete-time process $S = \{S_n, n = 0, 1, \cdots\}$ over the discrete state space $= \{0, 1, \cdots, D\}$, where S_n represents the size of the SLB leaky-bucket at time index n, not including a possible batch arrival at n. SLB contents evolve according to the following rules—1) Let a local batch arriving at time n consist of $\tilde{BA}$ clients, where $0 < \tilde{BA} \leq BA_{\max}$. If $S_n + \tilde{BA} \leq B + D$, the batch is declared *conforming* with respect to size, and S_n increments by an amount $\tilde{BA}$, otherwise the batch is *nonconforming* with respect to size, and S_n remains unaltered. 2) SLB contents decrement by an amount equal to $\min(B, S)$ one slot-time after each on period slot. In the next section, we model the contents of SLB and RLB as a joint Markov chain.

B. Analysis

Define the discrete-time process $Z = \{Z_n, n = 0, 1, \cdots\}$ over the discrete state space $= \{0, 1, \cdots, N_f - 1\}$, where $Z_{n+1} = (Z_n + 1) \cdot \mathrm{mod} N_f$, and $Z_0 \triangleq 0$. Here, Z_n is a labeling index that uniquely identifies the position of a local secondary-client batch arrival in the frame. The *on* period is characterized by $Z_n = \{0, 1, \cdots, BU_{\max} - 1\}$, while the *off* period is characterized by $Z_n = \{BU_{\max}, \cdots, N_f - 1\}$.

Lemma: The joint process (R, S, Z) is a periodic, discrete-time Markov chain, with a period equal to N_f.

The proof is straightforward, and we omit it. Let us assume that the transition probability function P of the chain (R, S, Z) is derived [10]. The chain (R, S, Z) is irreducible and has a finite state space Ω. Hence, it is positive recurrent [11] and has a unique stationary distribution, π^i, that is determined by numerically solving the linear equations: $\sum_{(a,b,c)\in\Omega} \pi^i_{abc} P^{rsz}_{abc} = \pi^i_{rsz}$, $\forall (r, s, z) \in \Omega$, and $\sum_{(r,s,z)\in\Omega} \pi^i_{rsz} = 1$.

C. Client-Level QoS Measures: $\Phi_{\text{sec-blk}}$, $\Phi_{\text{in-loss}}$

To compute $\Phi_{\text{sec-blk}}$, we define a few more terms

$$\begin{aligned}
\Phi^i_{\text{sec-blk}} &\triangleq \text{P}\{\text{Client is blocked at node } i\}\\
\Phi^i_{\text{sec-blk}}(z) &\triangleq \text{P}\{\text{Arrival at slot } z \text{ (at node } i\text{) is nonconforming}\}\\
\Phi^{\text{batch},i}_{\text{nonconf}}(k, z) &\triangleq \text{P}\{\text{Batch of size } k \text{ at slot } z \text{ (at node } i\text{) is nonconforming}\}.
\end{aligned}$$

Since the probability that an arriving batch is conforming is the joint probability that it is conforming with respect to both RLB and SLB, the probability that an arriving batch is nonconforming is

$$\Phi^{\text{batch},i}_{\text{nonconf}}(k, z) = 1 - \sum_{r=0}^{L} \sum_{s=0}^{B+D-k} \pi^i_{rsz} \cdot N_f \quad \text{for } z = 0, \cdots, BU_{\max} - 1,\ k = 1, \cdots, BA^i_{\max}.$$

A secondary-client arrival is nonconforming if it belongs to a nonconforming batch. Thus

$$\begin{aligned}
\Phi^i_{\text{sec-blk}}(z) &= \sum_{j=1}^{BA^i_{\max}} \text{P}\{\text{Client at slot } z \text{ belongs to } j\text{-sized batch, batch is nonconforming}\}\\
&= \frac{\sum_{j=1}^{BA^i_{\max}} \xi^i_j \times \Phi^{\text{batch},i}_{\text{nonconf}}(j, z) \times j}{\sum_{k=1}^{BA^i_{\max}} k\, \xi^i_k}.
\end{aligned}$$

Blocking at a node i is obtained by averaging $\Phi^i_{\text{sec-blk}}(z)$ over all the slots in the on period, so that

$$\Phi^i_{\text{sec-blk}} = \frac{\sum_{z=0}^{BU_{\max}-1} \Phi^i_{\text{sec-blk}}(z)}{BU_{\max}}.$$

The overall secondary-client blocking probability, $\Phi_{\text{sec-blk}}$, is simply the weighted average of the blocking at all the access nodes. □

To derive $\Phi_{\text{in-loss}}$, we define

$$\begin{aligned}
\Phi^i_{\text{in-loss}} &= \text{P}\{\text{Existing client at node } i \text{ aborts the application}\},\\
\mathcal{F}^i &= \text{Fraction of the clients carried at node } i, \text{ (assuming no internal loss)}.
\end{aligned}$$

Then, $\Phi^i_{\text{in-loss}} = 1 - \prod_{l=1\cdots|\Im|} (1 - \Phi^{\boldsymbol{l}}_{\text{sec-blk}} \cdot \gamma[l][i])$. Internal loss probability is

$$\Phi_{\text{in-loss}} = \sum_{i=1}^{|\Im|} \Phi^i_{\text{in-loss}} \times \mathcal{F}^i. \tag{3}$$

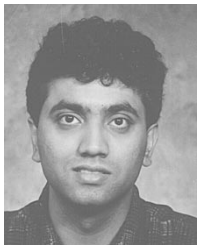
In [10], we outline an analytical procedure to derive $\mathcal{F}^i$. $\mathcal{F}^i$ is the ratio of the mean number of clients carried in the application at node i (ignoring internal loss) to the mean number of clients carried in the entire application (ignoring internal loss). To compute $\mathcal{F}^i$, we model the number of clients in the application and at the access node i as a joint stochastic process. Under certain conditions, this stochastic process can be shown to be a semi-regenerative process. Its distribution can be derived using the *key renewal theorem* [12]. Assuming $\mathcal{F}^i$ is determined, the internal loss probability can be computed using (3). □

ACKNOWLEDGMENT

The authors would like to thank A. Kalavade for a thorough critique of the paper.

REFERENCES

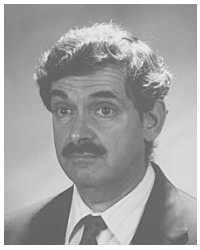
[1] J. Grudin, "Computer-supported cooperative work: History and focus," *Computer*, vol. 27, no. 5, pp. 19–26, May 1994.
[2] C. Topolcic, "Experimental internet stream protocol, Ver. 2 (ST-II)," RFC 1190, Oct. 1990.
[3] ITU-T Draft Recommendation Q.2931, *Edinburgh TD 155, B-ISDN, Digital Subscriber Signalling System no. 2, User Network Interface Layer 3 Specification for Basic Call/Connection Control.* Geneva, Switzerland, June 13–21, 1994.
[4] ATM Forum 94-0998, in *Signaling Subgroup Meeting Notes, UNI 4.0 Features*, Ottawa, Canada, Sept. 26–29, 1994.
[5] L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP)—Version 1 functional specification," R. Braden, Ed., Internet Draft, *draft-ietf-rsvp-spec-12.ps*, May 6, 1996.
[6] P. Moghé and I. Rubin, "Enhanced call—A paradigm for applications with dynamic client-membership and client-level binding in ATM networks," *IEEE/ACM Trans. Networking*, vol. 4, pp. 615–628, Aug. 1996.
[7] ——, "Enhanced call—A paradigm for multipoint applications with dynamic client-membership in ATM networks," in *Proc. GLOBECOM '95*, Singapore, Nov. 1995, pp. 723–727.
[8] B. T. Doshi, "Deterministic rule based traffic descriptors for BISDN: Worst case behavior and connection acceptance control," in *Proc. GLOBECOM '93*, 1993, pp. 1759–1764.
[9] ATM Forum, "ATM layer specification," *ATM User-Network Interface Specification*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[10] P. Moghé, "Admission management of applications with dynamic client-membership and client-level binding," Ph.D. dissertation, UCLA, 1995.
[11] P. Hoel, S. Port, and C. Stone, *Introduction to Stochastic Processes.* Houghton Mifflin, 1972, pp. 66–67.
[12] E. Cinlar, *Introduction to Stochastic Processes.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[13] M. H. Ammar, S. Y. Cheung, and C. M. Scoglio, "Routing multipoint connections using virtual paths in an ATM network," in *Proc. IEEE INFOCOM '93*, 1993, pp. 98–105.
[14] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization.* New York: Academic Press, 1981.
[15] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C.* Cambridge: Cambridge Univ. Press, 1988.
[16] W. Reinhardt, "Advance reservation of network resources for multimedia applications," in *Proc. ICAWA '94*, Germany, Oct. 1994.
[17] D. Ferrari, A. Gupta, and G. Ventre, "Distributed advance reservation of real-time connections," in *Proc. 5th Int. Workshop Network Operating Syst. Support Dig. Aud. Vid.*, Apr. 1995.
[18] L. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, and H. Wittig, "Issues of reserving resources in advance," in *Proc. 5th Int. Workshop Network Operat. Syst. Support Dig. Aud. Vid.*, Apr. 1995.
[19] M. Degermark, T. Kohler, S. Pink, and O. Schelen, "Advance reservations for predictive service," in *Proc. 5th Int. Workshop Network Operat. Syst. Support Dig. Aud. Vid.*, Apr. 1995.
[20] A. Gupta, "Improved performance for multi-party communication through advance reservations," in *Proc. 6th Int. Workshop Network Operat. Syst. Support Dig. Aud. Vid.*, Apr. 1996.
[21] D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," in *Proc. ACM SIGCOMM '92*, Aug. 1992, pp. 14–26.
[22] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks," in *Proc. ACM SIGCOMM '95*, 1995.



Pratyush Moghé (S'88) received the B.E. degree in electronics and telecommunications at the College of Engineering, University of Poona, Pune, India, in 1988, the M.S degree in electrical engineering from Clemson University, Clemson, SC, in 1990, and the Ph.D. degree from the University of California, Los Angeles, in 1995.

He is a Member of Technical Staff in the Network and Service Management Research Department at Bell Laboratories, Holmdel, NJ. He spent the summer of 1990 with the Network Architectures and Services Department at GTE Laboratories, Waltham, MA. His research interests include toolkits for application management and network management. He is also studying the problem of rapid prototyping of systems that implement adaptive and switchable network protocol stacks.

Dr. Moghé received the IAF Trophy Best Student Award in 1988 and the UCLA Fellowship in 1990.



Izhak Rubin (S'69–M'71–SM'83–F'87) received the B.Sc. and M.Sc. degrees from the Technion-Israel Institute of Technology, Haifa, Israel, in 1964 and 1968, respectively, and the Ph.D. degree from Princeton University, Princeton, NJ, in 1970, all in electrical engineering.

Since 1970, he has been on the faculty of the University of California, Los Angeles (UCLA) School of Engineering and Applied Science where he is currently a Professor in the Electrical Engineering Department. He has had extensive research, publications, consulting, and industrial experience in the design and analysis of commercial and military computer communications and telecommunications systems and networks. At UCLA, he leads a large research group. He also serves as the President of IRI Computer Communications Corporation, a leading team of computer communications and telecommunications experts engaged in software development and consulting services. During 1979 to 1980, he served as the Acting Chief Scientist of the Xerox Telecommunications Network. He was the co-chairman of the 1981 IEEE International Symposium on Information Theory, the Program Chairman of the 1984 NSF-UCLA workshop on Personal Communications, the Program Chairman for the 1987 IEEE INFOCOM Conference, and the Program Co-chair of the IEEE 1993 Workshop on Local and Metropolitan Area Networks. He is the Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS and of the *ACM/Baltzer journal on Wireless Networks*. He has contributed chapters to texts on telecommunications systems and networks.