

Resource sharing in advance reservation agents

Olov Schelén and Stephen Pink

Computer Science and Electrical Engineering

Luleå University of Technology, SE – 971 87 Luleå, Sweden

Tel: +46 920 91000, Fax: +46 920 72801, Email: {olov, steve}@cdt.luth.se

Abstract: We present an architecture where clients can make end-to-end resource reservations through agents. For each domain in the network, there is an agent responsible for immediate and advance admission control. Reservations from different sources to the same destination domain are aggregated as their paths merge toward the destination.

We show that network resources can be shared between immediate and advance reservations without being pre-partitioned. Admission control for immediate reservations use information about resources to be allocated for advance reservations in the near future. An important parameter in the admission control algorithm is the so called *lookahead time*, i.e., the point at which we actually start making resources available for approaching advance reservations by rejecting immediate requests. In our model, preemption of immediate reservations is made in cases where the admission control cannot make resources available through rejection of immediate requests. The risk of preemption can be varied by changing the lookahead time. We explore, with simulations, the effects of providing advance reservations with this model. The results show the cost in terms of resource utilization, rejection probability and preemption probability.

Keywords: QoS, advance reservations, resource sharing, agents, admission control, simulation, virtual leased lines

1 Introduction

Real-time applications such as audio and video conferencing may need resource reservations in the network to perform well. However, many of the current resource reservation proposals allow only immediate reservations, i.e., reservations at the time the session begins, while real-time events often are scheduled and advertised in advance. Clients making immediate reservations for scheduled events will have to judge the appropriate time for making reservation requests to be accepted in due time. When resources are scarce, clients may start making reservations earlier than needed to make sure that they obtain resources in due time. This leads to sub-optimal scheduling of network resources. By offering reservations in advance, clients obtain better support for planned activities and providers obtain information for scheduling resources over time.

Support for advance reservations is also important between different providers. For scalability reasons, it is not desirable that each end-to-end admission request results in call-specific state for each link along the path. Therefore, providers will negotiate bulk resources among each other for carrying aggregated admission-controlled traffic over several domains. In this context, advance reservations are suitable both for providing time-limited contracts and for providing bandwidth that varies at given time intervals, e.g., over hours of the day and days of the week.

We have designed a reservation architecture where immediate and advance admission control is performed by *agents* [15, 17]. The provided service corresponds to *virtual leased lines* between domains. An advantage with an agent-based approach is that reservations can be provided without having to maintain reservation state in routers. Reservations can be made for both senders and receivers. In addition, reservations can be made for remote locations, thereby supporting third party reservations and nomadic

computing, i.e., reservations for places where the host will be moving in the future. Our architecture can aggregate admission requests along the path towards a destination. This results in admission control for the aggregate being performed for individual links. In section 2, we give the background on our agent architecture.

The focus of this paper is on resource management for individual links. In section 3, we describe admission control for immediate and advance reservations [16]. The objective is to share resources between immediate and advance reservations. In section 3.1, we describe some architectural considerations about locating admission control in agents. In section 3.2, our service model is described. In this model resources for advance reservations will be made available by *rejecting* immediate requests for a short period before resources are to be allocated, or ultimately by *preempting* service for flows that were granted immediate access. Preemption means that a flow of packets using the reservation loses its service quality, i.e., the flow will still be given best-effort service (preemption is defined in section 3.2). In section 3.3, we define the notions *bookahead* and *lookahead*. In section 3.4, we define the admission control criteria. In section 4 we show, with simulations, the effects of advance reservations in terms of resource utilization, rejection probability and preemption probability.

2 The agent architecture

In [17] and [15], we present an architecture where clients can make resource reservations through *agents*. For each domain in the network there is an agent responsible for *admission control* (figure 1). Each agent knows the topology and static link resources in its domain. The agent is an end-system that is configured for passively participating in a link state routing protocol (e.g., OSPF) where each participating router has an identical topological database over the domain. Thus, an agent obtains the topological database with little signaling overhead. Link state advertisements are sent from the nearest router to the agent. Agents retrieve link properties, such as static bandwidths, by querying routers seen in the topology map. For this, we use a network management protocol (e.g., SNMP). Queries are done at startup and when topology changes are detected by the routing protocol. Routers need no extensions to allow agents to build a resource map through the management protocol.

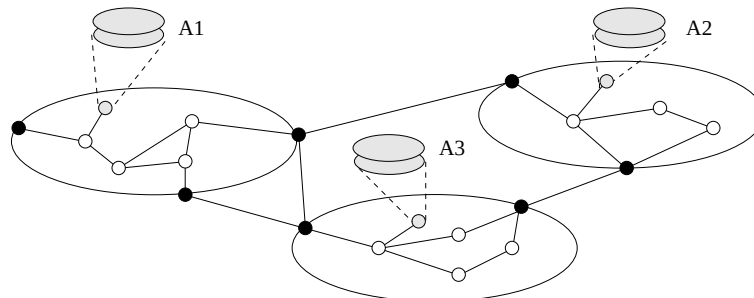


Figure 1: Reservation agents and their domains

Packets using reserved resources are *marked* by applications or edge routers. Core routers handle marked packets by classifying them into a small number of predefined service types. Similar ideas have been proposed in the *differentiated services* area [2, 12]. In the differentiated services model, packets are marked to obtain either lower drop probability or higher strict priority. Here, we abstract from these details and use the term *priority packets* to denote marked packets, and the term *priority traffic* for all priority packets collectively. A key point of differentiated services and our agent architecture is to avoid signaling state and expensive per-packet processing in routers. To allow resources to be managed in separate agents, we rely on router mechanisms for service differentiation on the packet level.

Independently of which packet scheduling model is used for differentiating service in routers, there must be an admission control architecture to make sure that the amount of priority traffic is low enough

to receive good quality of service. One issue is whether the admission control architecture should reserve resources over the links that are actually going to be used, or just limit the overall rights to send priority traffic. A scheme that does not consider exactly where traffic will go cannot give reliable commitments without very low utilization. Thus, it is likely that such a scheme would need to gamble and accept occasional service loss.

In this work we focus on agent-based admission control where the objective is to administer resources exactly, i.e., admission control in an agent maintains information about reserved resources on each link in its domain. Agents perform immediate or advance admission control without involving the routers. The architecture offers a service comparable to unidirectional *virtual leased lines*. Reserved resources can be used by one or many applications sending data between the domains connected by a virtual leased line.

There are other frameworks, i.e., IETF Integrated services [20, 7] and RSVP [21], that can set up both unicast and multicast reservations end-to-end. However, the provided service is not the same as for virtual leased lines, since reservations are made for individual data flows (i.e., application data streams). In that model, per-flow reservations are maintained in the routers. This results in scaling problems due to reservation state and packet classification overhead in routers. This model inherently relies on per-flow semantics and therefore it is hard to perform aggregation over different administrative domains. Thus, RSVP and Integrated Services may be too heavy-weight for the backbones.

In differentiated service, on the other hand, there is low overhead in routers. Our goal is to design an architecture that performs scalable resource management. Currently, our service model provides only unicast virtual leased links. When resources are managed by agents according to this service model, there are opportunities to aggregate reservations as they merge toward a destination domain. In this section, we give an overview of the admission control architecture and focus on the aggregation issues.

2.1 Direction of admission requests

An admission request can be directed to any agent, e.g., an agent that manages a user's account. Each agent sets up reservations between any two points in the network by invoking other agents. Thus there is *no difference* between reservations for *sending* and *receiving* data, or reservations made by a *third-party*.

Each reservation request contains a bandwidth to be reserved, a source and a destination address. The source address determines the point where the reservation starts (i.e., the point where traffic using the reservation will enter) and the destination address determines the point where it ends. An agent receiving a request first considers whether the starting point is in its domain. If the starting point is *not* in its domain, it finds an agent closer to the starting point and repeats the request with that agent. Technically speaking, it is possible to directly identify the agent that is responsible for the source domain, but in practice clients may prefer using a particular agent. Also, agents may prefer having business relations with only a few adjacent agents. This would require that requests are made in several steps. The result is that accounting information is aggregated, i.e., the mechanism helps keep down the number of separate charges. In figure 2, it is shown how a request is issued by node Dx for reserving resources from Ax to Dx.

If an agent finds that the starting point given in the request is in its domain, admission control is performed on the links from the starting point towards the destination, defined by the routing protocol. This is possible since all routers and the agent have the same topology map of the domain. Thus, an agent can perform admission control on the links from the starting point to an edge router in its domain.

An agent cannot perform admission control beyond its domain. Therefore, if the destination is outside of the agent's domain it must request a reservation with the neighboring agent, giving the edge point where traffic will cross the borders as the source address of the reservation (figure 2). Without distinguishing between reservations to different destinations, it would be impossible for agents to grant reservations to other domains. This is because there is no way to make sure that there are sufficient reservations along any path in other domains further downstream (unless we accept very low utilization for priority traffic).

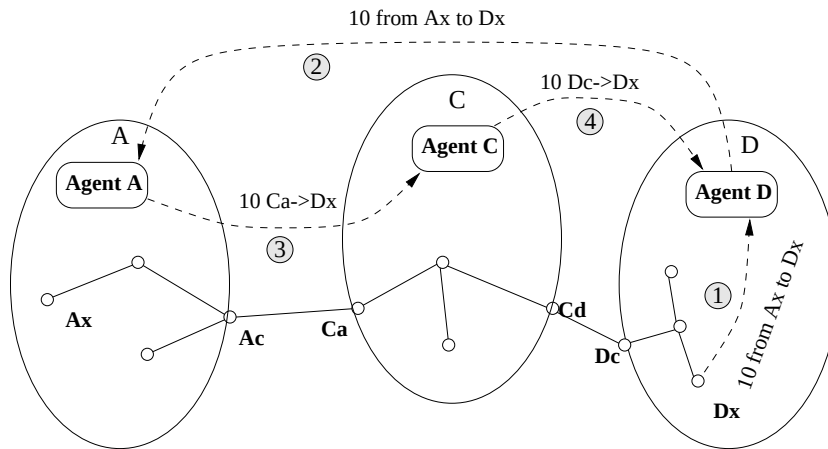


Figure 2: A receiver Dx requesting a reservation to receive data from Ax

2.2 Aggregation

If reservations were always done “on demand” there would be signaling between all neighboring agents along the path for each end-to-end admission request. Our architecture supports several ways of aggregating reservation state. Aggregation takes place in the agents because routers are not part of the admission control process. The motivation for aggregation may be less when reservation state is stored in agents with plenty of memory and disk. However, a per-flow reservation model would have scaling problems both in terms of state and processing cost in agents. Therefore it is important to show that agents can provide exact resource reservations with good scaling properties.

In [17] we describe how aggregation is done. The key ideas are the following: first, reservations from different sources to the same destination are aggregated as their paths merge toward the destination. Second, an agent in charge of resources at the final destination can generalize reservations for specific end points so that they are valid for any end point in the destination domain, thereby allowing more aggregation.

2.2.1 Merging reservations

To grant requests spanning many routing domains, there must be a commitment from all agents along the path involved in the request. When an agent is making a reservation with a neighboring agent, the source address included in the request determines the point (router interface) where the traffic will cross the borders, i.e., enter the domain of the neighboring agent. The neighboring agent does not care where traffic came from originally. Thus, priority packets can use reserved resources as long as the destination address matches the reservation. This allows agents to aggregate traffic with different sources into one single reservation made with a neighboring agent.

An end-to-end resource reservation can be seen as a number of consecutive *funnels* (figure 3). Priority packets poured into any of these consecutive funnels use reserved resources along the path to the destination. Agents may aggregate several requests with different source addresses into the same funnel, as long as all the requests specify the same destination. Agent C (in fig. 3) sets up a funnel starting at its edge by reserving bulk resources (40 units) with neighboring agent D to a distant destination and then aggregate requests for resources between domains C and D into the established funnel. This bulk reservation implies that downstream agents only keep state for the aggregate. The agent that decides to aggregate into a bulk reservation keeps information about its individual commitments constituting the

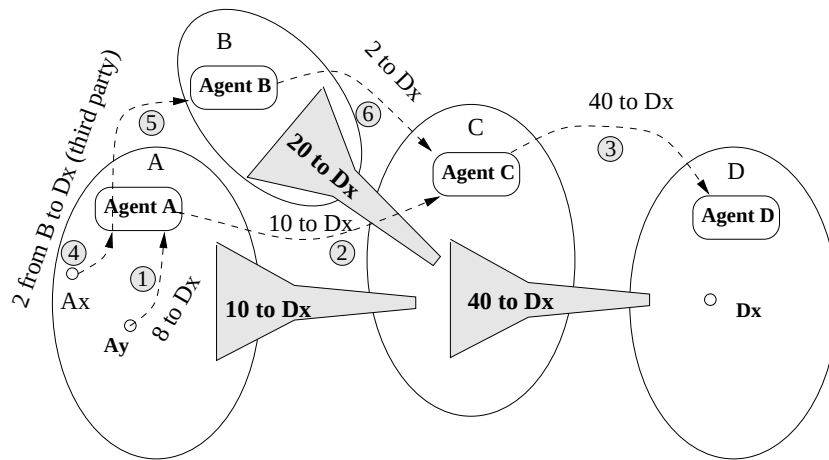


Figure 3: Funnel and aggregate reservations for one destination

aggregate. Aggregation by merging reservations towards each destination can be done with full control over the resources. The funnels form a so called *sink-tree* with the root towards the destination. At each merging point, the reserved bandwidth on the outgoing link will be equal to, or larger than, the sum of the reserved bandwidth on the incoming links.

2.2.2 Prefix aggregation

Requests that specify different destinations cannot easily be aggregated into a funnel in the general case. If we allow funnels to split into different directions further downstream, upstream agents must keep information about which destinations are involved and how resources are divided between them. This means that the aggregate no longer can be seen as one unit. However, if a set of destinations can be identified by one common identifier, and there are sufficient resources available in each branch to service the whole aggregate, then split outs can be scalable and exact.

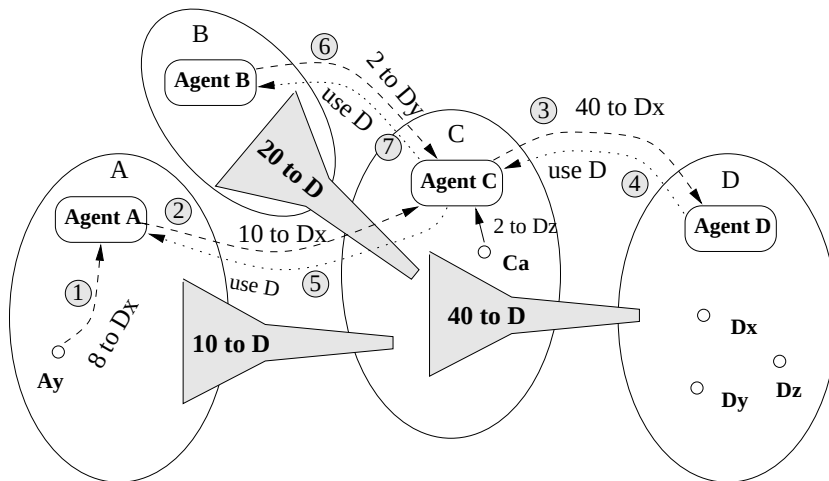


Figure 4: Funnel and generalized destinations

With this argument, we allow aggregation to be increased by *generalizing* a reservation as valid for

all destinations within the same destination domain. This means that a funnel provides resources to all endpoints within the destination domain, i.e., resources in the destination domain must be sufficient to ensure that priority packets get expected service independently of how they are routed within that domain. It is only the agent responsible for the destination domain that can judge if resources are sufficient for allowing generalization. Consider the example in figure 4. Agent D can judge if the resources in D are sufficient to let incoming funnels be valid for the whole domain. Thus, it must be agent D that decides when a reservation request for a particular destination node within D can be replaced with a reservation for any node in domain D.

In IP networks an *address prefix* is often used to represent a domain (individual nodes within the domain are distinguished by the remaining suffix). An address prefix is represented by an IP address and an associated mask telling how many bits are part of that prefix. An agent generalizes a request by replying with an *address prefix* to be used as a destination address for the reservation instead of the original address included in the request (figure 4). The aggregation that results from generalizing a reservation is also called *prefix aggregation*.

Clients or agents that have been granted a generalized reservation can use it for any destination matching the prefix of the reservation. Agents providing generalized reservations must handle the committed bandwidth of priority traffic to any destination in its domain. A conservative agent would therefore generalize reservations up to an aggregated value equal to the narrowest link. In practice we believe that this can be useful for well provisioned local area networks.

In the Internet, agents must be able to handle occasional topology changes (e.g., node and link failures), as well as pathological changes (e.g., route-flaps) [13]. Agents listen to the routing protocol and detect routing changes on average as fast as any router in the domain. When changes occur, agents can schedule resources along new paths.

3 Admission control for immediate and advance reservations

To obtain a modular design, advance reservations should, as far as possible, be managed separately from immediate reservations. One way of separating the admission control for immediate and advance reservations would be to partition the resources so that one part is available for immediate reservations and another part for advance reservations. However, this would result in two logically separate networks with the problem of partitioning resources according to current demand. Therefore, we have chosen to share resources between immediate and advance admission control. To allocate resources safely we have extended the immediate admission control to consider the near future concerning the resources that will be allocated for advance reservations. There are three major consequences of these results. First, the resources can be managed efficiently without suffering from badly chosen resource partition sizes. Second, the immediate admission control scheme can be kept fast and simple by not having to consider advance reservations in the far future. Third, we have the architectural freedom of locating the admission control for immediate and advance reservations at separate places in the network without suffering undue communication overhead.

3.1 Parameter-based or measurement-based?

Admission control algorithms can be classified as either parameter-based or measurement-based. The metrics used in *parameter-based* admission control are *traffic specifications* and static link resources. By aggregating the given metrics, such a scheme can offer absolute or statistical performance bounds [6, 7], often enforced by the packet scheduler (e.g., WFQ or EDF). Parameter-based admission control can be used for immediate reservations and, by adding timing information, for advance reservations.

The metrics used in *measurement-based* admission control involve also *measurements* of used resources [10] [20]. Such a scheme can detect reserved resources that are not utilized and grant those resources to other requests. The benefit of measurement-based admission control is better resource utilization. On

the other hand there is no guarantee that reserved resources are available when needed. The scheme may also be sensitive to synchronization effects. Suppose there are many events scheduled at a given time and many clients start reserving immediate resources just before that time. As there is little traffic, more reservations will be allowed. When all events start, the network will be over-allocated with poor service as a result.

In our agent scheme, we use only parameter-based admission control for a number of reasons. First, measurement-based admission control can be efficiently implemented only in routers or switches (where traffic measurement can be done directly). When admission control is in the agent, the signaling overhead for traffic measurements may be prohibitive. Second, traffic measurements provide little information about the future and are therefore not suitable for advance reservations [4]. Third, we want to provide a service model where reservations give a trusted level of service for people that are prepared to pay for it. The problems of lower utilization for reserved traffic are mitigated in a packet-switched network as unused resources always can be used by best-effort traffic. Thus the overall network utilization does not suffer very much from parameter-based admission control.

3.2 Service model

To support advance reservations, information about time must be added to the reservation requests. Therefore, advance reservation requests include a *duration* interval: $I = [t_s, t_e]$, where t_s is the starting time of the requested service and t_e is the finishing time of that service. This information could be automatically collected from a site where scheduled events are advertised, or generated by the client making a reservation. This is indifferent to our architecture.

Immediate reservations are open ended and do not specify a duration. A consequence of allowing open ended immediate reservations together with advance reservations is that there may be cases where resources are over-allocated. This is resolved by *preempting* immediate reservations to free resources for advance reservations. Preemption is used to ensure that there are always resources to honor advance reservations. Flows that lose reserved resources through preemption will be given best-effort service. We have chosen *not* to include a service for immediate requests that are *guaranteed* never to be preempted. Instead, we aim at keeping the preemption probability low, so that the user need not worry about preemption when asking for admission. A fundamental reason for this decision is that indefinite non-preemptible service would block resources indefinitely and no advance reservations of those resources could therefore be made. Furthermore, resources that are reserved in advance for some time in the far future cannot be given to indefinite sessions in the meantime unless we have the option of preempting those sessions when resources are needed. Thus, we have the following service model:

- Durations must be specified for advance reservations. A duration includes the start time and the finish time for the requested service.
- Immediate reservations do not specify durations. Such requests are serviced immediately (if possible) and can be preempted later by flows that reserved in advance.
- Preemption means that a flow loses its service quality; however, it will still be serviced at a best-effort level.
- In order to manage resources safely there may be limits imposed on minimum and maximum bookahead times for advance reservations.
- The admission control for advance reservations may support finding the earliest point for which a requested service can be granted.

There are other proposals for advance reservations that do allow indefinite (open ended) resource reservations. To provide indefinite sessions without having preemption, resources could be set aside for this

service exclusively. In the Tenet suite [5], resources are partitioned to support advance reservations and non-preemptible immediate sessions simultaneously. With that solution there will be some fragmentation due to it being hard to maintain the partitioning such that the demand for the two different services are met at each instant. In our architecture we can support indefinite sessions by partitioning resources. However, in this paper, we will investigate whether or not we can avoid some problems with resource fragmentation by sharing resources between immediate and advance reservations and instead allow for statistically rare cases of preemption.

There are also proposals relying on statistical guarantees of immediate and advance reservations where both immediate and advance reservations are preemptible. In [18] a model is presented where the call with the shortest bookahead time is preempted in case of congestion. The model relies on the theory of effective bandwidth to calculate the chance of a call being active. In our model we deal with aggregated bulk reservations rather than per-call reservations. For our model we have chosen to use deterministic guarantees in the admission control for advance reservations. Our belief is that advance reservations, once accepted, should not be preempted. Preemption of advance reservations is acceptable only when there are failures in the basic infrastructure. This consideration is made in a context where there is a lot of unplanned network traffic that use immediate reservations and an even larger amount of bursty traffic that uses no reservation at all. In this context resources will be well utilized.

3.3 Bookahead and lookahead

When immediate reservations and advance reservations share resources and the duration of each immediate reservation is unknown, there is a risk of over-allocation that can be resolved with preemption. The probability of preemption can be controlled by either monitoring immediate requests when admitting advance reservations, or vice versa, by monitoring advance reservations when admitting immediate requests.

In practice, however, the risk of preemption for immediate requests can not be controlled very well by advance admission control. When advance reservations are requested far in advance, the information obtained from monitoring present traffic or present immediate reservations will not be useful [4]. Most of the active flows would have finished at the crucial time anyway. It is only when advance reservations are requested with very short notice that it would be valuable to know about present traffic. Since advance reservations are provided to meet the demand for scheduled sessions primarily, we believe that most reservations will be requested on a notice much longer than the average session length. We denote the time from making an advance reservation until the resources are to be available as the *bookahead time*. The bookahead time is set by the clients asking for admission and varies from request to request. In our admission control we may impose a minimum bookahead time. Then, advance admission control need not consider the immediate reservations state at all.

The risk of preemption can be effectively controlled by immediate admission control. By monitoring advance reservations for the near future when admitting immediate requests, we can control the risk of preemption. If there are insufficient resources available in the near future, the request could be rejected instead of serviced. It is up to the provider to decide whether rejection or preemption is preferable in these cases. Rejecting immediate requests because the resources are reserved in the near future means that the overall utilization will go down to make room for the advance reservations. We denote the pre-allocation time, i.e., the time for starting to set aside resources for advance reservations as the *lookahead time* into the advance reservation state. Setting the lookahead time is an internal issue to the admission control scheme. As shown in simulations (section 4), the lookahead time can be tuned to obtain reasonable utilization and preemption probability.

3.4 Admission control criteria

Figure 5 is a snap-shot of admitted flows in a time/bandwidth diagram. Flows a, b and c are immediate reservations. For immediate reservations we add up the aggregate bandwidth. When a new admission

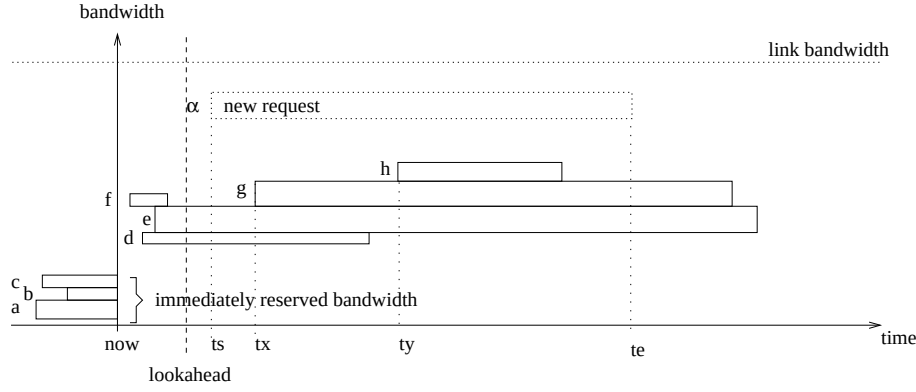


Figure 5: Snap-shot of reservations

request α arrives, admission is granted if the new reservation would not cause the aggregate bandwidth to exceed the link bandwidth. For advance reservations, the admission conditions consider other advance reservations that overlap with the new flow (d,e,g,h). The conditions are checked at all points where new flows begin (t_s , t_x and t_y). For immediate requests, the admission conditions consider the current aggregate of immediate reservations and the advance reservations that are active within the lookahead time (d,e,f). The following notation is used in describing our admission criteria:

R	Aggregate bandwidth reserved by current immediate reservations
$A(t)$	Aggregate bandwidth reserved in advance at time t
$look$	The lookahead time used at immediate admission
μ	The link bandwidth

Immediate reservations: When a client requests an immediate reservation α with bandwidth r^α , the admission control algorithm performs the following checks:

- Determine if the bandwidth usage, after adding the new load r^α , will exceed the available link bandwidth μ during the lookahead time.

$$\mu > \max_{t \in look} (r^\alpha + R + A(t)) \quad (1)$$

Advance reservations: When a client requests an advance reservation α with bandwidth and duration (r^α, I^α) , the admission control algorithm performs the following checks:

- Check whether the starting time is acceptable as a bookahead time. The minimum bookahead time is configurable to any value larger than the lookahead time.
- Check whether the finishing time is acceptable. It should be larger than the starting time. Admission control may also set a limit on how far in the future reservations can be granted.
- Determine whether the aggregate bandwidth of advance reservations exceeds the link bandwidth at some point during the requested duration:

$$\mu > \max_{t \in I^\alpha} (r^\alpha + A(t)) \quad (2)$$

3.5 Reservation state

As described in section 2.2, our model has mechanisms to minimize the amount of reservation state. Nevertheless, it is important to have data structures that manage reservations efficiently and allow quick admission control. For immediate admission control, we need to keep information about the current reserved aggregate bandwidth. We also need access to the advance reservation state within the lookahead time. For advance admission control, we need to keep the aggregated bandwidth reserved in advance over time. For this we have two alternative data structures. Our first data structure keeps exact starting times and finishing times for each admission request. The amount of state is proportional to the number of reservations maintained. The other data structure uses time slots where each reservation is extended to occupy a number of time slots. For each slot, the aggregate reserved bandwidth can be found. This data structure has fixed size and depends only on the maximum bookahead time allowed and the slot length.

Also, we must be able to find a victim for preemption among the immediate reservations. For this we need information about individual immediate reservations. However, potential victims can be chosen when they are admitted. It is enough to record a small number of potential victims corresponding to the amount of resources to preempt.

4 Simulations

The objective of our simulations is to explore the effects of allowing reservations in advance. The performance measures are *total utilization*, *rejection probability*, and *preemption probability*. We vary the *fraction* of the total link resources used for *advance reservations* and the *lookahead time* into advance reservation state at admission control for immediate requests. Admission decisions are parameter-based, i.e., based on traffic specifications and static link resources. For simplicity we use the requested bandwidth as the only traffic parameter. This study is for admission control over a single link only.

4.1 Simulation parameters

- The duration of immediate requests, i.e., the call hold time, is exponentially distributed with a mean of 300 time units.
- The duration of advance requests, is also exponentially distributed with a mean of 300 time units.
- The bookahead time for advance requests, i.e., how far in advance the advance requests are made, is exponentially distributed with a mean of 20 000 time units, where the tail is cut at 60 000 units (i.e., there is a maximum bookahead time).
- The total link capacity is 45 000 resource units.
- The resources asked for in each request are uniformly distributed in [1..3000] resource units.
- The inter-arrival time of admission requests is exponentially distributed with a mean of 11.76 time units. This number was selected to obtain a reasonable rejection probability, where around 6% of the requests are rejected. The average offered load is 85% of total capacity.
- When preemption is performed, the immediate request with the largest requested capacity is picked as a victim (which would encourage making requests in advance when large capacity is needed).

Note that the same parameters for duration and resource requirements are used for both immediate and advance reservations. This is to isolate the impact of reserving in advance versus reserving immediately. Should we have different parameters for the two classes, other effects would dominate the outcome in terms of utilization, rejection probability and preemption probability (the impact of having other parameters

is discussed in section 4.6). The bookahead time is chosen to be sufficiently large so it has little impact on the currently active immediate reservations.

The simulations are run many times for each parameter set, with different seeds for the random variables. The 90% confidential intervals are computed and presented as error bars in the plots (although the confidence is often so tight that the error bars appear as dots).

4.2 Advance reservation requests

We vary the fraction of the admission requests that are made in advance between zero and one. The upper bound on resources available for advance reservations equals the total link capacity, resulting in advance reservations being seldom rejected when a small fraction of the requests are made in advance (Figure 6).

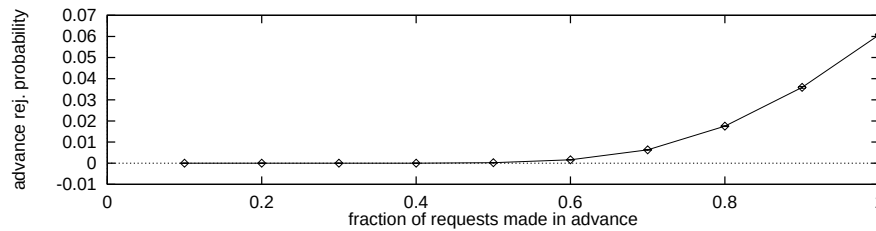


Figure 6: Rejection probability for advance requests

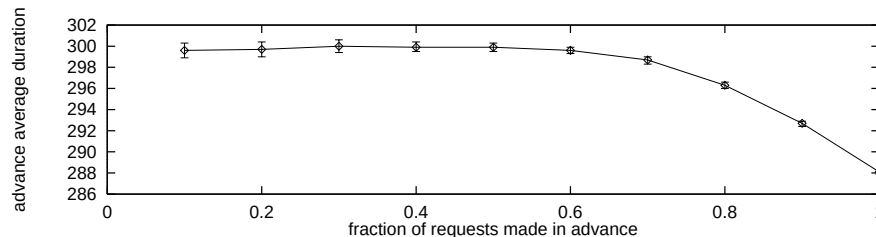


Figure 7: Average duration for accepted advance requests

Figure 7 shows that the average duration of the granted requests goes down as there is a higher fraction of advance reservations. When resources are scarce, advance reservations will result in some fragmentation in time, and there is a greater chance that requests for short durations are accepted. This effect does not occur for immediate requests since admission is made without considering durations (in fact, durations are unknown).

A similar effect can be observed concerning the average bandwidth given to the granted requests. If the offered load is increased, the average bandwidth for granted requests will go down as the rejection probability goes up. For immediate reservations clients see a considerable decrease in the chance of succeeding with large reservations as the offered load is increased. However, for advance reservations the rejection probability depends on the bookahead time. Therefore, clients can increase their chance of succeeding with large requests by reserving further in advance. This effect is clearly seen in our simulations. However, as the bookahead time is chosen by clients it is not easy to draw any conclusions on *how* early one should make an admission request. The appropriate bookahead time depends on when

other clients choose to book ahead. The only effect that can be claimed for sure is that the chance of having a request granted for a given duration increases strictly with the bookahead time.

4.3 Total utilization

We define the *total utilization* as being the ratio of the total reserved bandwidth versus the total bandwidth of the link. This metric considers how large part of the total resources that are reserved. The metric does not include increased link utilization due to best-effort traffic, nor drops in link utilization due to reserved resources not being utilized.

Figure 8 shows that the total utilization goes down as the fraction of advance reservations increases. There are two reasons for this. First, we start allocating resources ahead of the starting point of the durations for advance reservations. The interval during which resources are allocated ahead of time is denoted the *lookahead* time. Technically speaking the lookahead time is used by admission control for *immediate* requests to judge whether there are any advance reservations needing the resources within that lookahead time. If so, the immediate request will be rejected. Thus, having a longer lookahead time results in more immediate requests being rejected and thereby a lower utilization. Having a shorter lookahead time means that the network will more often be over-allocated, resulting in more immediate requests being preempted (as we will show in section 4.4).

The second reason for lower utilization is that fragmentation in time is introduced when advance reservations are allowed. There is little chance that other advance requests ask for the small fragments that may be available between already-granted advance reservations. Consequently, a decrease in utilization can be observed as the rate of advance reservations is increased, even though the lookahead time is zero. When the lookahead is zero, admission control for immediate requests does not consider any resources reserved in advance.

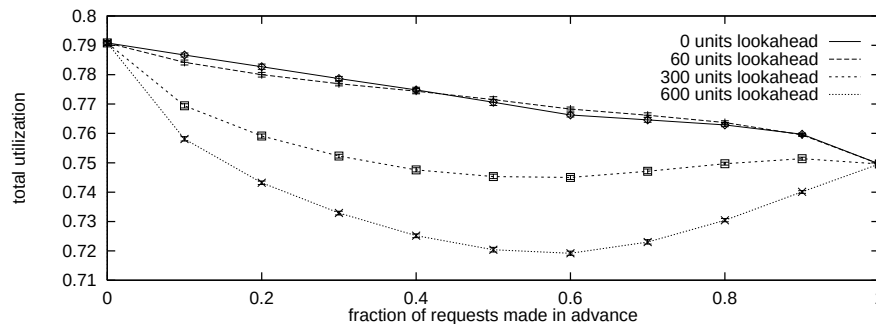


Figure 8: Total utilization

This may sound like bad news for providing advance reservations, but in fact similar drops in total utilization would be observed as soon as there are scheduled events introduced in the network, even though users were to make only immediate reservations. Users would still be likely to request resources before events begin to make sure that they have resources allocated in time. The only difference is that the control is with the clients instead of with the network. The degree of over-allocation in time that clients choose would be a consequence of social behavior in an environment with scarce resources. In some situations we can expect that the utilization drops in an uncontrolled way. When there are scheduled events, advance reservations can improve the service and the utilization by ensuring that resources are allocated at an appropriate time.

Measurement-based immediate admission control could alleviate the problem of clients over-allocating in time by measuring traffic on the link. However, we cannot get any useful measurements for scheduled

events until the traffic starts. Thus, we must choose between staying with the traffic specification or gradually losing the resource allocation as the measurement procedure finds that there is no traffic. When there are several events scheduled at the same time, there is a risk that allocated resources are insufficient to give acceptable service.

4.4 Rejection and preemption probability

Figure 9 shows that the rejection probability for immediate reservations increases as the fraction of advance reservations is increased. However, the change in rejection probability for immediate requests is primarily an effect of picking the victims among a smaller set of immediate requests as the rate of advance reservations increases. If we instead monitor the total rejection probability (Figure 10) over both immediate and advance requests, we get a much lower risk of rejection (note that the scales are different). Perhaps this is a better reflection of overall user satisfaction.

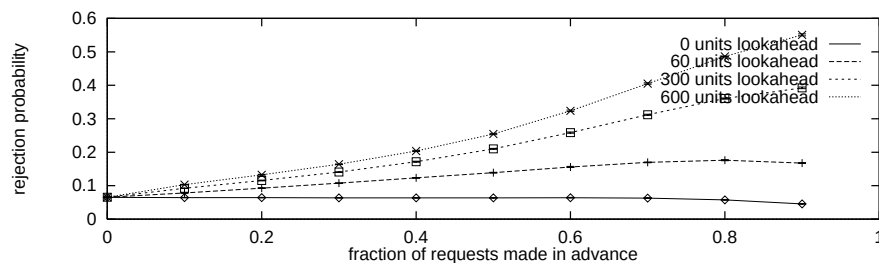


Figure 9: Rejection probability for immediate requests

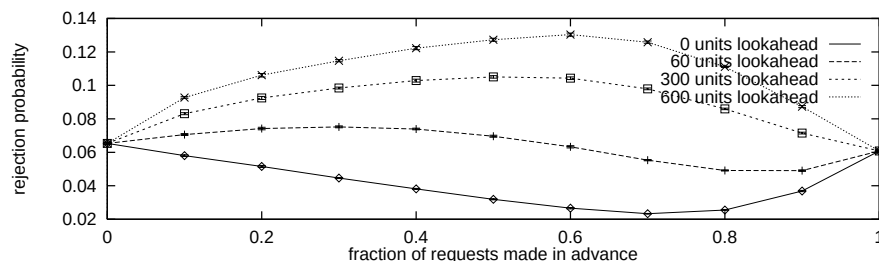


Figure 10: Total rejection probability (for immediate and advance requests)

We can see that the rejection probability is about the same when we have only immediate requests as it is when we have only advance requests, i.e., around 6 %. The decrease in total utilization (Figure 8) is explained by the fact that the average duration for accepted requests goes down (due to fragmentation in time) with increasing demand for advance reservations (Figure 7).

Figure 11 shows that the preemption probability increases as the fraction of advance reservations is increased. Again, part of the explanation is that all preemption victims are picked within the immediate requests and as the fraction of immediate requests goes down we get a higher risk of preemption for this service class. In Figure 12, we show the preemption probability over the total number of served reservations.

It is clear that increasing the lookahead time makes admission control more conservative, resulting in lower preemption probability at the cost of lower utilization and higher rejection probability. By

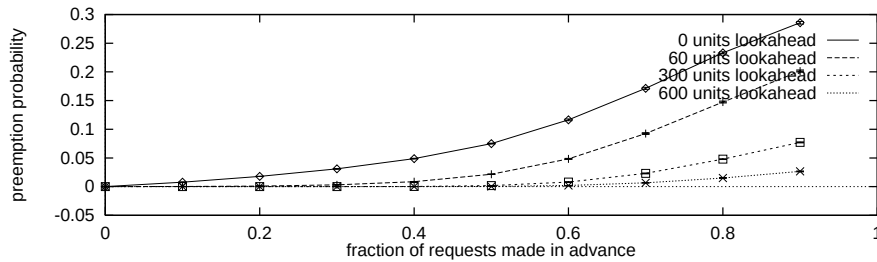


Figure 11: Preemption probability for immediate requests

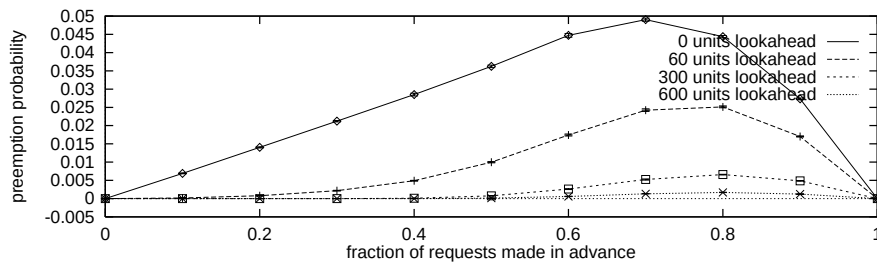


Figure 12: Total preemption probability (for immediate and advance requests)

decreasing the lookahead time, we allow the system to be more over-booked resulting in higher preemption probability. The fact that the system is allowed to be over-booked explains why the total rejection probability (Figure 10) goes down when we use zero lookahead and there is a suitable fraction of advance reservations.

4.5 Lookahead time

Given a desirable target for preemption probability, the lookahead time can be conservatively computed by considering the bandwidth that must be allocated for each advance reservation and the amount of bandwidth that is returned per time-unit by immediate reservations ending. However, at reasonable load, it is likely that there are some unallocated resources that should be considered as well. In our simulations, between 75% and 79% of the resources are allocated on average, which explains why it is possible to allocate resources for advance reservations with a total preemption probability below 5%, even when using zero lookahead.

In previous examples we kept the lookahead times constant as the fraction of advance reservations was increased. We have also tried to use exponentially distributed lookahead times but we found a strict performance degradation compared with constant lookahead time. If there are problems in meeting the requirements for high utilization and low preemption probability, we may be forced to have a limit imposed on resources available for reservations in advance. As an example, we now fix the advance request rate to 40% of the total number of requests (i.e., the average is bounded and there is no firm bound) and we plot the performance measures as functions of the lookahead time.

Figures 13, 14, and 15, show the effects. It is clear that the preemption probability falls steeply as the lookahead time is increased. In most systems, like cellular phone systems, preemption is considered bad and should be avoided. In a connectionless packet switched network, preemption is not as bad since generally the packets are still served best-effort. We believe, however, that the relative weight of having

low preemption is still high. Provided that we have the average duration for resource reservations (assuming exponentially distributed session lengths), the fraction of advance reservations, and the individual weights (set by the provider) for utilization, blocking and preemption, we can find a suitable lookahead time.

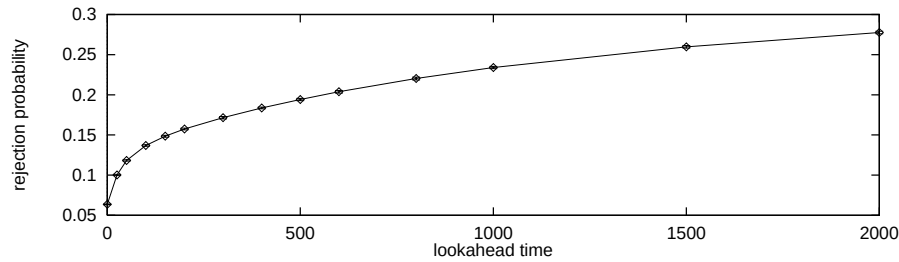


Figure 13: Rejection probability for immediate requests, at 40% advance request ratio

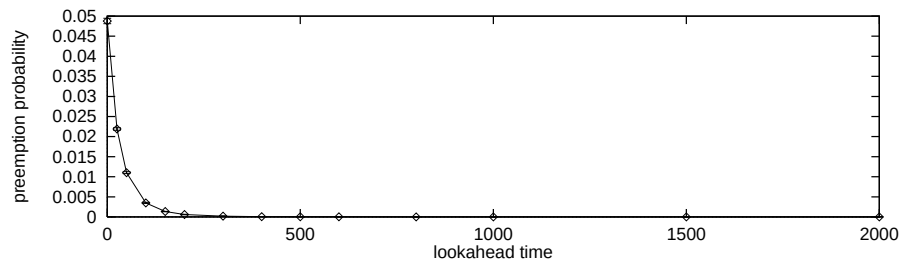


Figure 14: Preemption probability for immediate requests, at 40% advance request ratio

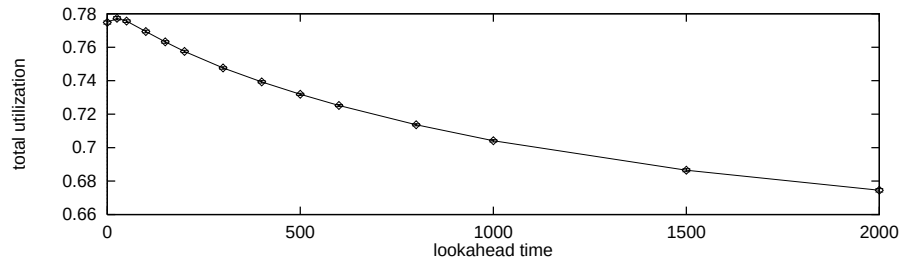


Figure 15: Utilization for immediate requests, at 40% advance request ratio

With a 40 % advance reservation request rate, we can, for example, use a lookahead of 50 time units, resulting in 77.6% of the resources being allocated under a 1.1% immediate request preemption probability (i.e., 0.44% total preemption probability). By using a lookahead of 100 time units, we get a utilization of 76.9% at a preemption probability of 0.35% for immediate requests. This is to be compared with 79% of the resources being allocated when there are no advance reservations. The offered load is in both cases 85% of the total capacity and the total rejection probability is around 7%.

If there were reservation agents without supporting lookahead in the immediate admission control algorithm, the results would be comparable with having zero lookahead (the network must support preemption and non-blocking resource allocation that can be used by the advance reservation agent). In that case, we have 5% preemption probability for immediate requests when 40% of the admission requests are made in advance.

4.6 Evaluation

Our simulations show the cost of advance reservations when they use the same holding-times and have the same resource demands as immediate reservations. We do this to isolate the impact of reserving in advance versus reserving immediately. By using different parameters for advance and immediate reservations, utilization increases when there are more advance reservations. Given a certain offered load, it is easy to show that there is a benefit from making large reservations in advance. It is generally easier to obtain a large utilization if we start packing the largest requests into the time-bandwidth diagram shown in figure 5. In reality, we expect that users needing large resources will tend to make reservations in advance to avoid being blocked.

5 Conclusion

In this paper we claim that resources can be more accurately allocated over time by supporting advance reservations for scheduled events. The price of supporting advance admission control is a small but controlled decrease in network utilization. This is compared with users making immediate reservations for their scheduled events. When resources are scarce, users tend to ask for admission ahead of time to make sure that they have resources granted in due time for their own scheduled event. Users make their decision with little information about other scheduled events and with little concern about the overall utilization of the network. In some situations, the effective utilization and the probability of having reservation requests granted can drop without control.

Our simulations show the effects of providing advance reservations. We assume that we allocate resources at some point for each reservation and the utilization is measured in terms of how much we can allocate. We do not consider drops in link utilization due to reserved resources not being used.

In our model, resources are shared between advance and immediate admission control, i.e., there is no pre-partitioning of resources. The most important parameter in the admission control algorithm is the so-called *lookahead* time, i.e., the point at which we actually start making the resources in the routers available for approaching advance reservations. We show how a suitable lookahead time can be found through simulations.

The effects of advance reservations can be expressed in terms of rejection probability, preemption probability, and total utilization. In the simulations presented, we can allocate 79% of the link resources when there are no advance reservations and around 77% of the resources when 40% of the admission requests are made in advance, and there is a 0.4% preemption probability for immediate requests. The offered load is in both cases 85% of the total capacity and the total rejection probability is around 7%.

Acknowledgments

This work was supported by a grant from the Center for Distance Spanning Technology (CDT), Luleå, Sweden, and Ericsson Radio Systems, Sweden.

References

- [1] A. Banerjee, D. Ferrari, B. A. Mah, and M. Moran. The tenet real-time protocol suite: Design, implementation, and experiences. Technical Report TR-94-059, University of California at Berkeley, Berkeley, California, November 1994.

- [2] D. D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. Technical report, MIT Lab for Computer Science, Boston, Massachusetts, November 1997.
- [3] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: architecture and mechanism. In *Proceedings of SIGCOMM*, pages 14–26, Baltimore, Maryland, August 1992. ACM. *Computer Communication Review*, Volume 22, Number 4.
- [4] M. Degermark, T. Köhler, S. Pink, and O. Schelén. Advance reservations for predictive service. In *Proceedings of NOSSDAV*, Lecture Notes in Computer Science, pages 3–14, Durham, New Hampshire, April 1995. Springer.
- [5] D. Ferrari, A. Gupta, and G. Ventre. Distributed advance reservation of real-time connections. In *Proceedings of NOSSDAV*, Lecture Notes in Computer Science, pages 15–26, Durham, New Hampshire, April 1995. Springer.
- [6] V. Firoiu, J. Kurose, and D. Towsley. Efficient admission control for edf schedulers. In *Proceedings of IEEE Infocom*, San Fransisco, California, March 1996.
- [7] R. Guerin, C. Partridge, and S. Shenker. Specification of guaranteed quality of service. Request for Comments (Proposed Standard) 2212, Internet Engineering Task Force, October 1997.
- [8] A. Gupta, W. Howe, M. Moran, and Q. Nguyen. Resource sharing for multi-party real-time communication. In *Proceedings of IEEE Infocom*, Boston, Massachusetts, April 1995.
- [9] S. Herzog. Rsvp extensions for policy control. Internet Draft, Internet Engineering Task Force, November 1996. Work in progress.
- [10] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *Proceedings of SIGCOMM*, Cambridge, Massachusetts, September 1995.
- [11] D. J. Mitzel, D. Estrin, S. Shenker, and L. Zhang. A study of reservation dynamics in integrated services packet networks. In *Proceedings of IEEE Infocom*, San Fransisco, California, March 1996.
- [12] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. Internet draft, Bay Networks, LBNL and UCLA, November 1997.
- [13] V. Paxson. End-to-end routing behavior in the internet. In *Proceedings of SIGCOMM*, pages 25–38, Palo Alto, California, August 1996. ACM.
- [14] W. Reinhardt. Advance reservation of network resources for multimedia applications. In *2nd International Workshop on Advanced Teleservices and High Speed Communication Architectures (IWACA'94)*, Heidelberg, Germany, October 1994.
- [15] O. Schelén and S. Pink. An agent-based architecture for advance reservations. In *IEEE 22nd Annual Conference on Computer Networks (LCN'97)*, Minneapolis, Minnesota, November 1997.
- [16] O. Schelén and S. Pink. Sharing resources through advance reservation agents. In *Proceedings of IFIP Fifth International Workshop on Quality of Service (IWQoS'97)*, New York, May 1997.
- [17] O. Schelén and S. Pink. Aggregating resource reservations over multiple routin domains. In *Proceedings of IFIP Sixth International Workshop on Quality of Service (IWQoS'98)*, Napa Valley, California, May 1998.
- [18] D. Wischnik and A. Greenberg. Admission control for booking ahead shared resources. In *Proceedings of IEEE Infocom*, San Fransisco, California, March 1998.
- [19] L. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, and H. Wittig. Issues of reserving resources in advance. In *Proceedings of NOSSDAV*, Lecture Notes in Computer Science, pages 27–37, Durham, New Hampshire, April 1995. Springer.
- [20] J. Wroclawski. Specification of the controlled-load network element service. Request for Comments (Proposed Standard) 2211, Internet Engineering Task Force, October 1997.
- [21] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource ReSerVation protocol. *IEEE Network Magazine*, 7(5):8–18, September 1993.