

Advance reservations for predictive service in the Internet

Mikael Degermark¹, Torsten Köhler^{1,2}, Stephen Pink^{1,2}, Olov Schelén²

¹ Department of Computer Science, Luleå University, S-971 87 Luleå, Sweden; {micke,olov}@sm.luth.se

² Swedish Institute of Computer Science, PO Box 1263, S-164 28 Kista, Sweden; {steve,tk}@sics.se

Abstract. We extend a measurement-based admission control algorithm suggested for predictive service to provide advance reservations for guaranteed and predictive service, while retaining the attractive features of predictive service. The admission decision for advance reservations is based on information about flows that overlap in time. For flows that have not yet started, the requested values are used, and for those that have already started measurements are used. This allows us to estimate the network load accurately for the near future. To provide advance reservations we ask users to include durations in their requests. We present simulation results to show that predictive service with advance reservations provides utilization levels significantly higher than those for guaranteed service, and comparable to those for predicted service without advance reservations. Those utilization levels are reached without any preemption of other admitted flows. Finally, we discuss how to setup advance reservations over multiple hops in the Internet using resource reservation setup protocols.

1 Introduction

Real-time multimedia applications will share future networks with traditional data applications. To provide quality of service (QoS) for real-time applications, it is likely that resource reservations will have to be made in the network. Current resource reservation protocols allocate resources just before communication begins, e.g., ST-2 [14] and various ATM signaling protocols reserve resources during connection establishment. This model of communication may not fit the needs of future network users [12].

Resource reservations should be optional and decoupled from the starting time of the session. One should be able to reserve resources prior to or during a network session, depending on when a specific service is needed. Users may know far in advance of their needs and would like to plan their activities by making advance reservations to ensure that they are not blocked by the network's admission control mechanism. Imagine some users with busy schedules in

different time zones who want to have an important teleconference on a resource-limited network at an agreed time in the near future. They should be allowed to make an advance reservation, given that they know when and for what duration their teleconference will take place.

In this paper, we will look at an important candidate for an admission control algorithm, originally proposed in [8] and later refined in [9], for a service called predictive service [3]. Predictive service provides QoS for applications that can tolerate some loss, such as real-time digital audio and video applications that can adjust their playback points in response to jitter in the network. The efficiency gain of predictive service comes from allowing more flows into the network than guaranteed service, thus providing more sharing and lower cost. The architecture described in [3] supports guaranteed and predictive service, but not advance reservations.

The possibility of making advance resource reservations should be a part of a communications architecture to provide better service to the users. Whether advance reservations are actually needed depends on future resource scarcity. Where resources are plentiful, not even immediate reservations may be necessary, but where resources are scarce enough to justify reservations at all, it makes sense to be able to make them in advance. In this paper, we will show that advance reservations can be provided by the network with little overhead.

2 Framework

The service model and the admission control algorithm suggested in this paper are extensions of those presented in [3] and [9]. In [3], the proposed network service interface offers guaranteed service, predictive service and best effort (ASAP) service. The service interface relies on token bucket traffic shaping; the source specifies the bucket size b and the token generation rate r . Guaranteed service provides a minimum transmission rate, and therefore the queuing delay bound becomes the bucket size divided by the rate. Predictive service provides K different service classes with widely spaced target delay bounds D_i and it is suggested that the target bounds are spaced by an order of magnitude. The bounded quantity is the queuing delay per hop, so it is necessary to add up

the target delay bounds at each hop to find the upper bound on the total queuing delay.

To support this service interface, a scheduling algorithm is presented in [3]. The guaranteed service traffic is scheduled with weighted fair queuing (WFQ) [4], so that each guaranteed service client has a separate WFQ flow. All the predictive service flows and ASAP traffic share the spare bandwidth in a pseudo-WFQ flow, called flow 0. The available bandwidth for flow 0 is therefore $\mu - \hat{\nu}_G$, where μ is the link bandwidth and $\hat{\nu}_G$ is the measured bandwidth usage for all guaranteed flows over the link. Inside flow 0, there are a number of strict priority classes: one class for each target delay bound and ASAP traffic at the lowest priority. The strict priority scheme implies that queuing delay experienced by higher priority classes will be conveyed to lower priority classes.

Admission control is performed in each switch along the path of a flow. Admission requests will be carried to the switches by an end-to-end resource reservation protocol.

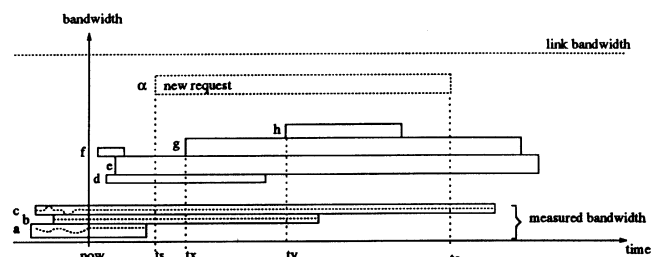
Note that our use of the term “guaranteed service” in this paper is adopted from [3]. There are other ways to provide guaranteed service which may give better utilization, e.g., jitter-EDD [5].

3 Duration intervals

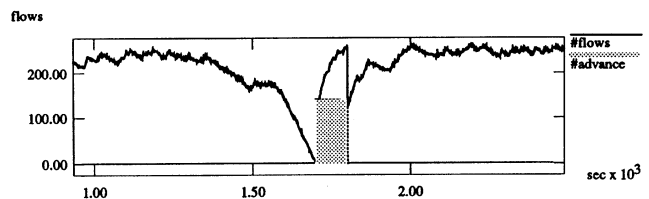
To achieve an efficient scheme for advance reservations without preemption, we have extended the service interface so that each admission request includes a duration interval: $I = [t_s, t_e]$, where t_s is the starting time of the requested service and t_e is the finishing time of that service. Knowledge of these intervals is necessary for the admission control algorithm to determine which requests overlap and when the reserved resources will be released. As a special case, requests for immediate admission will specify *now* as their starting time. Note that a duration interval refers to the resource allocation only, sending packets outside the interval is possible, but as no resources are allocated to those packets, they will receive a lower service quality.

A user that does not know how long a session will be should be able to use a default value or a value derived from a personal profile. If a resource allocation turns out to be shorter than needed, it will be possible to renegotiate by calling admission control again. If the new request is granted, the duration of the resource allocation is extended. If not, the session is in danger of losing service quality when the resource allocation expires. Default intervals could be renegotiated automatically just before they expire.

If a resource allocation turns out to be longer than needed, other advance reservation requests might unnecessarily be rejected. Fortunately, the unused resources can be given to those who are requesting immediate admission. This is because the measurement procedure of predictive service automatically detects unused capacity once a flow is active. Therefore, over-reservation has little impact on the total utilization as long as there are some immediate requests for admission. In addition, there is an option for users to explicitly cancel the allocation before it expires.



1



2

Fig. 1. Snapshot of reservations

Fig. 2. Plot of number of active flows when there is a large block of flows admitted in advance

4 Admission control decision for advance reservations

The admission decision for predictive service is based on requested rates for flows that have not yet started and on measured rates for currently active flows. If there are no advance reservations and a request for immediate admission arrives, our extended conditions give the same result as the conditions stated in [9].

Figure 1 is a snapshot of admitted flows in a time/bandwidth diagram. Flows a, b and c are currently active and we have measurements of their rates and maximum delays, which are used as predictions of their future behavior. When a new admission request arrives, admission is granted if the new flow would not cause any delay bounds to be violated or bandwidth limits to be exceeded. The admission conditions (Sect. 4.1) only consider flows that overlap with the new flow (b,c,d,e,g,h), using measured bandwidth if they have started or, otherwise, bandwidth requests; we call this the *estimated* bandwidth. The conditions are checked at all points where new flows begin (t_s , t_x and t_y).

For reservations in the distant future, the number of currently active overlapping flows is small and admission decisions are based mainly on requested rates. In the near future, the number of currently active overlapping flows is probably large and admission decisions are based mainly on measured values. So, in the distant future, the admission criteria are conservative, but as time proceeds more overlapping flows will become active and we obtain better estimates of bandwidth usage. Thus, as we get closer in time to the point at which a flow with an advance reservation is to begin, we have a more accurate knowledge of the network load and more flows can be admitted. Requests for immediate reservation can fill up the remaining bandwidth.

4.1 Admission criteria

A client may request admission for predictive service in one of the classes 1 to K (where class- K packets are scheduled at the lowest priority level), or for guaranteed service. The following notation will be used in the formulas¹ describing our admission criteria:

- $\nu_{G(t)}$ estimated bandwidth for guaranteed flows at time t ,
- $\nu_{P(t)}$ estimated bandwidth for predictive flows at time t ,
- $\nu_{P_i(t)}$ estimated bandwidth for flows in predictive class i at time t ,
- $R_{G(t)}$ requested bandwidth for guaranteed flows at time t ,
- $\hat{D}_j$ measured delay in predictive class j ,
- $B_j(t)$ bucket size sum for not yet started flows in predictive class j .

Predictive service: When a client requests service in predictive class k for a flow α , shaped by token bucket filter $(r_k^\alpha, b_k^\alpha, I^\alpha)$ the admission control algorithm performs the following checks:

- Determine if the bandwidth usage, after adding the new load r_k^α , will exceed the available link capacity $v\mu$ during the requested interval I^α :

$$v\mu > \max_{t \in I^\alpha} (r_k^\alpha + \nu_{G(t)} + \nu_{P(t)}) . \quad (1)$$

The available link capacity, $v\mu$, is determined by the link capacity μ and the link utilization target v , that is tunable.

- Determine whether the worst possible behavior of the new flow and the other flows that have not yet started can cause violation of delay bounds for predictive service classes k through K .

The worst case is when all predictive service flows flush their entire token buckets simultaneously in one burst. The resulting queue will be emptied according to the available bandwidth.

- Check the delay bound, D_k , of the same priority level:

$$D_k > \max_{t \in I^\alpha} \left(\hat{D}_k + \frac{b_k^\alpha + \sum_{i=1}^k B_i(t)}{\mu - \nu_{G(t)} - \sum_{i=1}^{k-1} \nu_{P_i(t)}} \right) \quad (2)$$

- check the delay bound of the lower priority levels, i.e., D_j where $k < j < K$.

$$D_j > \max_{t \in I^\alpha} \quad (3)$$

$$\left(\frac{\hat{D}_j (\mu - \nu_{G(\text{now})} - \sum_{i=1}^{k-1} \nu_{P_i(\text{now})}) + (b_k^\alpha + \sum_{i=1}^j B_i(t))}{\mu - \nu_{G(t)} - \sum_{i=1}^{k-1} \nu_{P_i(t)} - r_k^\alpha} \right) .$$

Guaranteed service: When a client requests guaranteed service for a flow α shaped by $(r_G^\alpha, b_G^\alpha, I^\alpha)$, the admission control algorithm first performs the total bandwidth check expressed in condition 1, then the following checks are performed:

- Determine whether the requested bandwidth of all guaranteed service flows will exceed link capacity:

$$v\mu > \max_{t \in I^\alpha} (r_G^\alpha + R_{G(t)}) . \quad (4)$$

¹ These formulas are extensions of those presented in [9]

- Determine that the delay bounds of each predictive service class is still observed when the remaining bandwidth is decreased (estimated bandwidth for guaranteed flows and for predictive classes with higher priority is subtracted from the link bandwidth).

$$D_j > \hat{D}_j * \max_{t \in I^\alpha} \left(\frac{\mu - \nu_{G(\text{now})} - \sum_{i=1}^{j-1} \nu_{P_i(\text{now})}}{\mu - r_G^\alpha - \nu_{G(t)} - \sum_{i=1}^{j-1} \nu_{P_i(t)}} \right), \quad 1 \leq j \leq K . \quad (5)$$

4.2 Operation of admission control algorithm

Figure 2 illustrates how our admission control algorithm operates. At time 1700 a large number of sources start. These sources, which were admitted in advance, have reserved all of the available bandwidth and all finish at time 1800. There is a background of sources asking for immediate admission with predictive service. The figure clearly shows that the number of active flows goes down to zero just before time 1700, because admission control rejects new flows to honor the resource commitments to the previously admitted sources, and because active sources finish. At time 1700 the number of active flows increases sharply as the previously admitted sources begin to transmit. As the aggregate traffic is measured and found to consume only part of the resources needed to accommodate the worst case according to the token bucket parameters of the admission requests, more sources can be admitted and the flow count increases further.

4.3 Renegotiations

If a request is made for extending the duration of an already active flow, it would be very conservative to base the admission criteria on peak-rate values obtained from the token bucket specification. Instead the available measurements of the flow could be used to predict its future behavior. This can be done by checking the conditions 1–3, replacing the bucket rate r_k^α with the measured rate and replacing the bucket depth b_k^α with zero (so that it has no effect on the admission decision). This means that the future behavior of any active flow is given by the measured values only. The conditions need to be checked at the points in the requested interval where previously admitted flows will start. This ensures that the extended duration will be granted only if no previously admitted reservations are jeopardized.

5 State requirements

To make the admission control decision, the original admission control algorithms [8, 9] need the current bandwidth use for all guaranteed flows plus maximum delay and bandwidth use for each predictive service class. These are measured values and per-packet processing is needed to maintain them.

The information needed to make advance admission decisions in a switch is summarized in Figs. 3 and 4. Figure 3 shows the cumulative requested bandwidth admitted to flows

that have not yet started. Figure 4 shows the predicted bandwidth use of currently active flows. Present measurements are used as predictions of future bandwidth use for those flows. We need to keep the state corresponding to these diagrams for the guaranteed flows collectively and for each predictive service class individually. Note that the state corresponding to Fig. 3 need to be accessed only when making an admission decision, while the state corresponding to Fig. 4 are measured values and per-packet processing is needed to maintain it.

A straightforward implementation for advance reservation would keep an amount of state proportional to the number of active flows plus the number of flows reserved in advance. Aggregation methods, however, can decrease the amount of state needed for the admission control decision: flows that start or cat finish at the same time, or at nearly the same time, can be treated collectively.

There is a tradeoff between the amount of state saved by aggregating requests and the flexibility of making requests. A simple way to aggregate requests is to use time slots. Duration intervals may then start and finish only at certain points in time. A disadvantage with this scheme is internal fragmentation: clients may have to reserve longer intervals than they will actually use.

Any scheme for resource allocation in a packet-switched network requires that switches can find the appropriate resource allocation when a packet is received, so that the packet can be scheduled correctly. It seems inevitable that this mapping requires an amount of state linear in the number of active reservations if resource reservations can be made for individual flows. For instance, the values that identify flows with reservations (addresses, port numbers, flow identifiers, etc.) will typically be kept by the switch.

To summarize, regardless of whether advance reservation is supported or not, predictive service requires that a switch keeps and accesses on a per-packet basis, an amount of state that is no more than linear in the number of active reservations. To support advance reservations, our admission control algorithm needs to keep an additional amount of state not more than linear in the number of admitted flows that have not yet started. This additional state needs to be accessed only when admission decisions are being made.

6 Simulations

Our simulations aim to show that adding advance reservation capability to the admission control algorithm for predictive service does not decrease utilization levels very much. This implies that advance reservations for predictive service give significantly higher utilization levels than advance reservations for guaranteed service.

The simulated network topology is a single link, and the source models are simple on/off models with exponentially distributed on and off times. We have done simulations of scenarios with immediate reservations only, and with both immediate and advance reservations. We have also examined the effects of aggregating state for active flows with similar finishing times. Finally, as it is likely that advance reservations will be made a long time in advance and have longer

durations than immediate reservations, we have studied how such scenarios affect utilization levels.

6.1 Simulated topology

The simulated topology is a single 10 Mbit/s bottleneck link connecting two routers. A number of sources are connected to one of the routers with links of infinite bandwidth. All sources send data to a sink connected to the other router. Our data comes from the upstream router R (Fig. 5).

6.2 Source model parameters

We use three kinds of sources, all generate packet trains at some peak rate p . The train length is exponentially distributed with mean N . The time between packet trains is also exponentially distributed with mean I . The ratio between the peak and average rate, p/a , can be calculated from those values.

All sources regulate their output with a token bucket filter with token generation rate r and bucket depth b . Each token is worth 1000 bits, which is equal to the packet size; sending one packet consumes one token. If the bucket is empty the packet is queued until a token is available. The token bucket filters in Table 1 are designed to always have a token available when the source wants to output a packet.

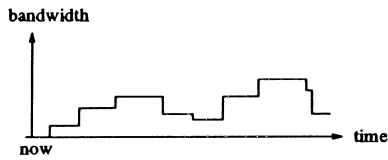
All source parameters are listed in Table 1. In the table, D is the maximum delay for a guaranteed flow, calculated from the token bucket parameters. D_j is the requested delay bound when the source asks for predictive service. The router supports two predictive service classes, one with a delay bound of 16 ms and the other with a delay bound of 160 ms.

6.3 Flow generation

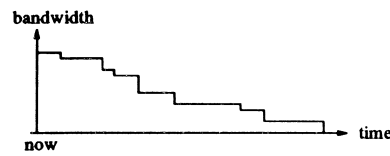
Sources ask for admission according to a Poisson process; the times between admission requests are exponentially distributed with a mean of 400 ms. In all simulations, the offered load is about 2.4 times larger than the capacity of the link, so most admission requests are rejected.

In the first set of simulations, the requested duration intervals are exponentially distributed with a mean of 300 s. For sources that ask for admission in advance, the times between the admission request and the start of the duration interval, henceforth called the *booking times*, are exponentially distributed with a mean of 5 min.

In the second set of simulations, part of the load, henceforth called the *uniform* load, comes from sources with durations that are uniformly distributed in an interval $\pm 50\%$ of their mean. These sources use booking times that are also uniformly distributed in an interval $\pm 50\%$ of their mean. The offered uniform load is kept constant by decreasing the number of admission requests when durations are large, e.g., when durations have a mean of 30 min the number of admission requests are six times fewer than when durations have a mean of 5 min.



3



4

Fig. 3. Cumulative requested bandwidth

Fig. 4. Predicted bandwidth use of currently active flows

Table 1. Source model parameters

Model name	Model parameters				Token bucket		Delay bounds	
	p pkts/s	I ms	N pkts	p/a	r tkns/s	b tkns	D ms	D_j ms
Exp 1	64	325	20	2	64	1	16	16
Exp 2	1024	90	10	10	320	50	160	160
Exp 3	10^7	584	9	76000	512	80	–	160

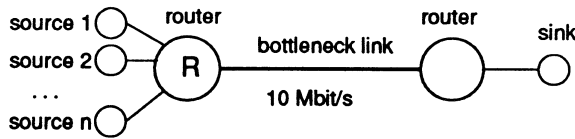


Fig. 5. Simulated topology

A source requests admission by sending a setup packet containing the desired service type and token bucket parameters towards the destination. If all routers along the path grant admission, the source transmits during the requested interval and then it stops.

6.4 Measuring process

The measuring process estimates current bandwidth utilization $\hat{v}$, and experienced maximum delay $\hat{D}$ in the same way as in [9]. When deciding whether to admit an advance reservation, the algorithm needs estimates of bandwidth utilization in the future, e.g., at time t in Fig. 6. This is done by continually estimating current bandwidth utilizations and using these as predictions of future utilization. The estimates are obtained by a straightforward extension of the measuring process in [9]. The packet rate of every active flow is sampled; these rates are then used as in [9] to obtain estimates of bandwidth utilization between finishing points of flows. The sum of the rates of the bottom three flows in Fig. 6 are used to estimate $\hat{v}$ between $f3$ and $f4$, i.e., the finishing points of flows 3 and 4 from the bottom. This procedure ensures that the estimates are conservative in the distant future and accurate in the near future where many currently active flows will still be active.

In a straightforward implementation, calculating the estimates is linear in the number of flows. To avoid keeping track of every individual flow and to reduce the overhead in calculating the estimates, we have experimented with aggregating flows that finish at about the same time. The bandwidth utilization of the aggregated flows is then estimated collectively.

6.5 Simulation results

To verify our simulation environment, we first replicated some relevant results from [9] in our simulator. In this first set of simulations, all sources in a single simulation conformed to the same source model and all requested immediate admission for the same type of service. The results of these simulations are summarized in Table 2 under IMM. In Table 2, *util* is the utilization of the bottleneck link and *delay* is the maximum experienced queuing delay. *# sources* are averages of flow counts; *act* is the average number of sources that were transmitting. The *measuring params* are the size of the T and S windows used in the measuring process (see [9]).

The utilization target v was 90% in all simulations. All simulations ran for at least 3000 s simulated time. The data in Table 2 comes from the second half of the simulated time. Visual inspection confirmed that no startup transients remained at that time.

In the simulations with advance reservations, 50% of the sources asked for immediate admission and 50% for admission in advance. The choice was random. All sources conformed to the same model and asked for the same type of service. These simulation results are summarized in Table 2 under ADV. There, *adv* is the average number of sources that were transmitting and were admitted in advance, and *adm* is the average number of sources that were admitted in advance but have not begun transmitting.

The GRA simulations are similar to ADV, the only difference being the measuring process: all flows that finish within the same A seconds are aggregated and measured collectively. This also implies that, for purposes of admission control, finishing times are rounded upwards to the nearest A seconds. In the table, A is the granularity of the measuring process.

6.6 Discussion

These simulations clearly show that predictive service with advance reservations provides higher network utilization than guaranteed service with advance reservations. They also

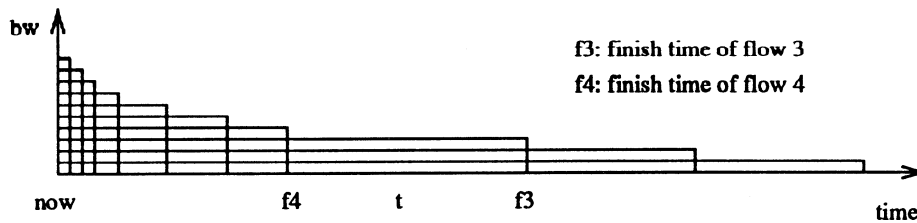


Fig. 6. Estimated future bandwidth use for currently active flows

Table 2. Simulation results, durations exponentially distributed

Name	Model	Serv	util %	delay (ms)	# sources (avg)			Measuring params		
					act	adv	adm	$T(s)$	$S(s)$	$A(s)$
IMM	Exp1	G	45	2.6	140	–	–	5.0	0.80	–
ADV	Exp1	G	44	2.3	137	137	282	5.0	0.80	–
IMM	Exp1	P	78	2.3	244	–	–	5.0	0.80	–
ADV	Exp1	P	68	1.9	213	160	265	5.0	0.80	–
GRA	Exp1	P	70	1.9	219	164	249	5.0	0.80	32
GRA	Exp1	P	75	2.5	232	153	240	5.0	0.80	128
IMM	Exp2	G	28	9.3	28	–	–	1.0	0.12	–
ADV	Exp2	G	27	8.4	27	27	109	1.0	0.12	–
IMM	Exp2	P	76	37.0	75	–	–	1.0	0.12	–
ADV	Exp2	P	59	13.4	58	37	124	1.0	0.12	–
GRA	Exp2	P	54	11.1	54	37	122	1.0	0.12	32
GRA	Exp2	P	50	11.5	49	33	123	1.0	0.12	128

show that adding advance reservation capability to predictive service decreases with bandwidth utilization. The levels are not much lower though for smooth traffic, but for bursty traffic, the utilization level decreases more. When the fraction of sources asking for advance admission is lower the decrease in utilization is lower, e.g., in simulations when 10% of the sources (instead of 50%) ask for admission in advance, the utilization is 69% for the burstier Exp2 traffic.

The reason for the decrease in utilization is that advance reservations will block requests for immediate admission. This blocking effect is larger when the sources are bursty, since the token bucket parameters are larger. Moreover, when token buckets are deep, the admission decision is based on delay considerations more than on available bandwidth. To make a good admission decision in this case, the algorithm would need to know how much each flow contributes to the current queuing delay. This would enable the algorithm to estimate future delay, since it knows which flows will be active at any future time. Instead, the algorithm uses the current delay as an estimate of future delay. Since this is a very conservative estimate, network utilization suffers.

An interesting and somewhat surprising result is that, when there was aggregation of flows in the measuring process, utilization increased for the smooth traffic generated by Exp1 sources, but decreased for the burstier Exp2 sources. It is easy to see that utilization might decrease, since the durations of active reservations are virtually extended by the aggregation in the measuring process. This gives less room for new reservations. The simulation results are not conclusive as to why aggregation increases utilization for the smooth Exp1 sources.

An interesting observation is that, for guaranteed service, sources asking for immediate admission are almost com-

pletely shut out by sources asking for admission in advance. This is due to the fact that, for guaranteed service, the admission decision is based on requested values only, regardless of estimated bandwidth use. The sources asking for admission in advance are admitted first, and so can starve out sources asking for immediate admission, since no bandwidth is freed when the sources with guaranteed service begin to transmit.

6.7 Booking time and duration

These simulations investigate how booking times and durations are related to utilization. Table 2 shows that there is virtually no decrease in utilization levels for guaranteed service when there are advance reservations, so we concentrate on predictive service. No effort was spent in finding good parameters for the measuring process, so it is likely that utilizations levels could be improved somewhat. Jamin et al. [9] discussed the effects of changing various measurement parameters.

In this and all following simulations, 60% of the load has zero booking times and durations that are exponentially distributed with a mean of 5 min. Of the offered load, 40% is uniform as explained in Sect. 6.3. The parameters in Tables 3, 4, and 5 refer to the uniform load only. All sources in a simulation obey the same source model, and all request predictive service. Simulations ran for 7500 s simulated time and the reported results come from the last 1500 s. No delay bounds were violated.

Table 3 gives utilization levels for a simulation series where all booking times were zero, i.e., all requests are for immediate reservation. This allows comparison with later results using non-zero booking times. Table 3 generally shows a tendency of increasing utilization as the durations of the uniform load increase. This is expected, as the number of

Table 3. Zero booking time

Mean duration (min)	5	10	15	30	45	60	75
Exp1 util (%)	79	81	82	78	78	83	83
Exp2 util (%)	78	79	80	80	81	81	81
Exp3 util (%)	40	42	43	46	45	47	46

Table 4. Effect of booking time. Mean duration is 10 min

Mean booking time (min)		5	15	30	45	60	75
Exp1	util (%)	80	79	79	78	79	78
	#adv	162	136	132	133	134	132
Exp2	util (%)	76	76	76	76	77	76
	#adv	36	26	26	25	25	25
Exp3	util (%)	16	19	20	19	21	21
	#adv	30	15	15	15	15	15

active flows with long durations increase and there are fewer dips when flows leave.

The uniform load in the next set of simulations have duration means of 10 min. The booking times vary. Results are reported in Table 4, which lists utilization levels, *util*, and average number of active reservations that were booked in advance, *#adv*. Table 4 show that utilization levels are independent of the length of the booking time when the booking time is longer than the duration intervals. As there is no overlap with active flows at the time of admission, the number of flows that can be admitted in advance is determined by how much will fit into the link given the token bucket parameters. Simple calculations on the token bucket parameters and comparison with the reported *#adv* numbers confirm this. The utilization levels for the smoother Exp1 and Exp2 models decrease only a few percent compared to the simulations with zero booking time. For the Exp3 model, utilization levels decrease dramatically, from 42% to 21%, when durations are 75 min. Note, however, that the utilization level would be less than 2.7% if guaranteed service were requested. The utilization decreases because of the blocking that occurs just before reservations made in advance become active; see Fig. 2 for an illustration of this effect.

The final simulations keep the mean booking time fixed at 15 min, and the duration means vary from 5 to 75 min. Table 5 list the results. There, *#adv* is the number of active flows that were reserved in advance and *#advstarts* is the number of such flows that started within the 25-min interval.

Table 5 shows that longer duration intervals increase utilization levels. The reason appears to be that the number of starts of reservations that were booked in advance decreases as the length of duration intervals increase. The number of accompanying dips then also decreases. Utilization levels for the bursty Exp3 sources fluctuate significantly due to the uneven distribution of starting times of advance reservations. A plot of the utilization level and number of starting advance reservations per time unit clearly show this phenomenon, the utilization level drops significantly before a high concentration of starts.

When booking times are shorter than durations, the number of active flows that were admitted in advance will increase. This does not appear to have much effect on utilization. A 20,000-s simulation, where durations had a mean of

Table 5. Effect of duration. Mean booking time is 15 min

Mean duration (min)	5	10	15	30	45	60	75
Exp1	util (%)	69	79	82	84	74	85
	#adv	130	144	153	187	205	219
	#advstarts	628	342	244	152	109	93
Exp2	util (%)	71	76	78	79	80	80
	#adv	26	26	28	38	50	52
	#advstrats	133	69	52	31	24	19
Exp3	util (%)	11	19	24	28	30	35
	#adv	15	15	19	29	42	50
	#advstrats	80	42	29	27	22	21

75 min and booking times were large, resulted in 36% utilization for Exp3 sources. Compare this to the 37% utilization, when booking times had a mean of 15 min (Table 5).

To conclude, these simulations show that when durations of advance reservations are long, utilization levels are about the same as the levels without advance reservations, unless token bucket parameters are very large, in which case the levels decrease somewhat due to blocking before the starting times of reservations booked in advance. When booking times are longer than a few minutes, utilization levels are not related to the length of the booking time. Utilization does depend on the duration of advance reservations, longer duration gives higher utilization. There were no violations of delay bounds in any simulation.

7 Setup protocols for advance reservations

To provide an end-to-end advance reservation service in an internetwork, it is necessary to have a protocol which is able to set up reservations in advance across multiple hops. So far, we have been concerned only with admission control into a single hop. There have been some protocols suggested for setting up reservations in a packet-switched network, including ST-2 [14], RSVP [16], Q.2931 [17] and RCAP [1]. The first two protocols were designed to provide reservations in the Internet. One difference between them is that ST-2 relies on *hard* state and RSVP on *soft* state. Another difference is that in ST-2 the sender makes the reservations and in RSVP it is the receiver. For our purposes, *hard state* means that the reservation state in the routers, once set up, will remain in those routers until explicitly torn down. By *soft state* we mean that reservation state must be refreshed periodically and will timeout otherwise. In practice, this means that, as routes change, the reservation state in RSVP will timeout and be rebuilt along the new path.

7.1 Setting up advance reservations using RSVP

With some minor changes, RSVP [16] can be used to set up advance reservations in almost the same way as it is used for setting up immediate reservations. To establish an advance reservation for a multi-party session, the senders have to announce their session by periodically sending announcement messages (in RSVP terms, “PATH” messages) down a multi-cast tree. Receivers respond to those announcements by sending reservations toward the senders. The resources

have at this point been reserved for some time in the future. At the time the session starts, resources are allocated and the service to each session participant increases from best effort to the requested quality.

For reservations made far in advance, there is potentially a very large number of path and reservation messages that must be sent before the session begins. To reduce overhead, the frequency of sending these messages should start at a low level and increase as the time of the session approaches. RSVP could support advance reservations efficiently, while allowing the admission control algorithm and measuring process to aggregate session state, if the following two minor changes are made:

- To support advance reservations, the flow specification carried by RSVP path and reservation messages should include session durations. A sender will state a duration for the session and the receivers are free to reserve any interval within that duration. Since senders may lengthen or shorten durations, special wildcard durations can be used by the receivers to follow the changes made by the sender.
- To cancel a reservation, RSVP should provide the original flowspec in the interface between RSVP and the admission control mechanism. This is because our admission control algorithm aggregates requests for sessions of similar duration to save state and for measuring purposes. We propose that RSVP provide the original flow specification when making a call to the admission control mechanism to delete a session.

Setting up advance reservations with RSVP is attractive because relatively small changes are needed to the protocol. Also, RSVP seems to be a setup protocol that will be widely used in the Internet. However, there are two problems with the scheme.

First, both senders and receivers need to be present when the reservation is made and during the whole period before the reserved interval starts. The actual application need not be present, but some agent on each involved host needs to join the multi-cast group and tell the local RSVP agent to start sending and receiving path and reservation messages. If hosts fail or are temporarily cut off from the network, the reservation may timeout. Even worse, the sender and/or receiver hosts may not even be present when the reservation needs to be made². A truly flexible resource setup protocol should allow resource reservations by third parties as well as by receivers and/or senders.

Second, RSVP currently does not prevent route changes. Changed routes will cause the reservation messages of existing reservations to pass through new routers, which may or may not have sufficient resources to grant the reservations. If not, some reservations will be rejected, which introduces an ordering problem in rebuilding advance reservation state. Thus, earlier reservations can be shut out by later reservations whose request happens to arrive first at the router in the new path. Such behavior will decrease the utility of advance reservations. If you cannot trust an advance reservation to be present at session time, why bother to make it? There

appears to be a strong connection between a resource reservation and the path over which it holds.

7.2 Soft versus hard state setup protocols

Hard-state-based resource reservation protocols avoid the problem of changing routes, but at the cost of flexibility. Some routing changes that are due to failures are unavoidable. When a hard-state-based resource reservation protocol encounters such a routing change, its only available action might be to reestablish the connection on a new route or tear down the connection. Reestablishing the connection on the new route has the same consequences for the user as RSVP's way of rebuilding state on the new path: reservations could be rebuilt in such a way as to deny service to users that had earlier reservations. On the other hand, tearing down the connection completely and not reestablishing a new one seems an inflexible way of handling the situation if a new route in fact exists. Only an "all-or-nothing" user would desire such a service.

Route "pinning" has been suggested for RSVP to avoid situations where the route protocol "flaps" between two different paths without a failure on either one. This could happen, for example, when, while routing, table calculations attempt to converge after a change. Such "flapping" would needlessly disturb the QoS for an application. Route pinning would allow RSVP to keep a path stable after a reservation is made by continuing to forward packets on the initial path, even though the route has changed. This might help the situation where advance reservations would otherwise need to be rebuilt along new paths. In a proposed scheme (pers. comm.), a *pin* flag is added to the reservation messages of RSVP. When the pin flag is present, the path towards the source is pinned down as long as the pin state is refreshed. If real routing failures occurred, pinned routes would be freed and the RSVP path would migrate to the new route. Thus, RSVP would be able to take advantage of the robust features of Internet routing (such as being able to route around "damage") while preserving QoS in the face of "flapping" routes.

One way to avoid the situation of having resource reservations rebuilt in a new order and denying service unfairly is to have reservations timestamped to reflect the original time reservations have been made. Although this would add more complexity, it could in principle be implemented in a soft-state-based as well as hard-state-based setup protocol.

Thus, RSVP's soft-state style of resource reservation would seem to have advantages over setup protocols that use hard state. The major problem for setting up resource reservations in advance with such protocols is the demand that senders and/or receivers must be present at booking time for these schemes to work. This is a problem for both hard- and soft-state-based setup protocols. Thus, a third-party approach to making advance reservations on behalf of potentially absent senders or receivers would seem to be needed.

8 Related work

The distinguishing feature of our work is that we offer an admission control algorithm that allows advance reservations

² This point was made by David Clark

for predictive service without preemption. The algorithm results in significantly higher utilization levels for predictive service than for guaranteed service.

There are few papers on advance resource reservation in the literature. The earliest mention of advance reservations in packet-switched networks we have found is from 1992 [12].

The work closest to ours is [6], where advance reservations are for a kind of guaranteed service. Their simulations show network-wide utilization gains, since conference sessions (that share resources) receive priority by being reserved in advance. All schemes for advance reservation where reservations are merged, including ours if used together with, e.g., RSVP, would show similar gains. In their scheme, a separate partition of the link is allocated to advance reservations. If such partitioning were used for predictive service, there would be small or no utilization gains compared to guaranteed service, because partitioning prevents allocating unused resources to immediate requests.

Wolf et al. [15] describe a general model for resource reservation in advance. Admission control algorithms are not suggested.

A new scheme called RBONE for making advance reservations in the Mbone (i.e., the Internet multi-cast backbone) has recently been presented [2]. The key idea is to assign priorities to sessions that are pre-registered. Resource reservation is done as the flow becomes active and RSVP reservation requests carry priorities. The receivers learn which priority to use by an out-of-band mechanism. The routers give resources to the requests with the highest priorities, possibly by preempting reservations with lower priorities. The scheme is independent of routing, and routers do not keep much state for advance reservations, except for what is needed to authenticate reservation requests. A disadvantage is that preemption is necessary. The scheme is geared towards large and dense sessions. When few hosts participate in each session, it is necessary that the agents assigning priorities know the topology of the network in order to provide an adequate response to registrations. Users, for example, may want to be notified if their registrations are likely to be preempted.

9 Further work

We have used simple on/off source models. Recent studies [10, 13, 7] have shown that real data traffic and VBR video traffic are self-similar, which means that our simple traffic models can give results that are too optimistic. Jamin et al. [9] show by simulation that predictive service works well for some self-similar traffic models when flows enter and leave frequently. However, when reservations are made a long time in advance and traffic is self-similar, current estimates of bandwidth and delay can be poor estimates of the future. A natural continuation of our work is to simulate using self-similar traffic.

Another continuation would be to study larger topologies. This has been done by Jamin et al. [8] for the original predicted service admission control algorithm. The concern here is that it is possible for existing advance reservations in parts of a multihop path to block new admission requests

along that path, so that utilization decreases intolerably. The reservation setup protocol and its timing parameters are important here, since partially installed reservations that are later rejected in another part of the path could introduce severe blocking under high load [11]. This problem is not specific to predicted service, since reservations for other kinds of service would also suffer to some degree from such phenomena.

10 Conclusions

We have shown how the predictive service admission control algorithm developed in [8] and [9] can be extended to support advance reservations, provided that requests for admission specify the duration of their reservation. Our work shows that it is possible to have advance reservations without preemption.

The extended admission control algorithm proposed in this paper relies on knowledge of which flows overlap in time with the flow that requests advance reservation, measuring those overlapping flows that are active, and assigning the requested rate to the flows that have not yet started. Thus, more requests for a certain duration of time can be granted as we get closer to that duration of time, increasing sharing and lowering cost for those flows that occupy that duration. We have also suggested ways to minimize the amount of state information necessary to provide advance reservations and to simplify the measuring process that estimates future bandwidth use.

Our simulations show that predictive service with advance reservations provides higher network utilization than guaranteed service with advance reservations. They also show that adding advance reservation capability to predictive service decreases bandwidth utilization. The levels are not very much lower though, and they are still significantly higher than for guaranteed service. The decrease in utilization is due to the fact that advance reservations will block sources asking for immediate admission. This blocking effect is larger for bursty sources which request more resources and when durations of advance reservations are short.

We have also discussed how to set up advance reservations in the Internet. RSVP can, with minor changes, be used to set up reservations in advance. There are, however, problems with this solution and we suggest that a more general mechanism might be more appropriate.

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

References

1. Banerjee A, Mah B (1991) The real-time channel administration protocol. In: Proc 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Springer-Verlag, Berlin, pp 160–170
2. Braden B, Casner S : RBONE – Resource administration for the MBONE. Presentation at 32nd IETF, Danvers, Massachusetts. URL for slides: <ftp://ftp.ietf.cnri.reston.va.us/ietf-online-proceedings/95apr/area.and.wg.reports/tsv/rsvp/rsvp.casner.slides.ps>

3. Clark D, Shenker S, Zhang L (1992) Supporting real-time applications in an integrated services packet network: architecture and mechanism. *Rapid Communication Rev* 22:4
4. Demers A, Keshav S, Shenker S (1990) Analysis and simulation of a fair queueing algorithm. *J Internetworking: Res Exper* 1:3-26
5. Ferrari D (1992) Real-time communication in an internetwork. *J High-Speed Networks* 1:79-103
6. Ferrari D, Gupta A, Ventre G (1995) Distributed advance reservation of real-time connections. In: *Proc 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pp 15-26
7. Garrett MW, Willinger W (1994) Analysis, modeling and generation of self-similar VBR video traffic. In: *Proc ACM SIGCOMM '94*, pp 269-280
8. Jamin S, Clark D, Shenker S, Zhang L (1993) Admission control algorithm for predictive real-time service. In: *Proc 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, Springer-Verlag, Berlin, pp 349-356
9. Jamin S, Danzig P, Shenker S, Zhang L (1995) A measurement-based admission control algorithm for integrated services packet networks. *Comput Communication Rev* 25(4), pp 14-26
10. Leland WE, Taq Q MS, Willinger W, Wilson DV (1992) On the self-similar nature of Ethernet traffic. *Comput Communication Rev* 23(4)
11. Mitzel DJ, Estrin D, Shenker S, Zhang L (1996) A study of reservation dynamics in integrated services packet networks. In: *Proc IEEE INFOCOM '96*, San Francisco, Calif, 2: 871-879
12. Partridge C, Pink S (1992) An implementation of the revised Internet stream protocol. *Internetworking Res Experience* 3:27-54
13. Paxson V, Floyd S (1994) Wide-area traffic: the failure of Poisson modeling. *Comput Communication Rev* 24(4), pp 257-268
14. Topolcic C (1990) Experimental Internet Stream Protocol, Version 2 (ST-II). RFC 1190
15. Wolf L, Delgrossi L, Steinmetz R, Schaller S, Wittig H (1995) Issues of reserving resources in advance. In: *Proc 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, New Hampshire, pp 27-37
16. Zhang L, Deering S, Estrin D, Shenker S, Zappala D (1993) A new resource ReSerVation protocol. *IEEE Network Mag* 7:8-18
17. Broadband Integrated Service Digital Network (B-ISDN) Digital Subscriber Signaling System No. 2 (DSS2) user network interface layer-3 specification for basic call/connection control (1994) ITU-T Recommendation Q.2931. International Telecommunication Union, Geneva



STEPHEN PINK is the manager of the Computer and Network Architecture research laboratory at the Swedish Institute of Computer Science and Professor of Computer Science at Luleå University in Sweden. He has a Bachelor's degree from the University of Minnesota, a Master's degree from Cornell University, and a Ph.D from the Royal Institute of Technology in Stockholm. Before coming to Sweden, he was a faculty member at the University of Rochester and worked in the computer and communication industry in Massachusetts. He has been the program chair of the ACM SIGCOMM 1994 conference, been on

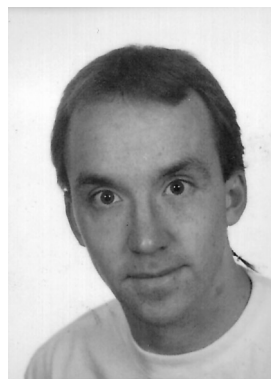
the program committee of a number of workshops and conferences, and is on the editorial board of *IEEE Network Magazine*, the *Journal of High-Speed Networking*, and the *Journal of Internetworking*.



TORSTEN KÖHLER is a member of the Computer and Network Architecture research laboratory at the Swedish Institute of Computer Science and a Ph.D student at Luleå University in Sweden. His interests are network and transport protocols for high-speed and mobile networks. He has a Master's degree from Humboldt University Berlin, Germany. Before coming to Sweden, he was a visiting scientist at the IBM European Networking Center in Heidelberg, Germany and a researcher at Technical University of Berlin.



OLOV SCHELEÉN is currently a Ph.D student at Luleå University, Sweden. His interests are in multimedia networking and resource reservations. He received his MS (1985) and his Licentiate (1988) in Computing Science from Luleå University. He was a lecturer in Computer Science and Networking (until 1994).



MIKAEL DEGERMARK is a Ph.D student in Computer Science at Luleå University in Sweden. His interests include network and transport protocols for multimedia networking and mobility. He has a Master's degree and a Licentiate degree, both from Luleå University.