DRES: Network Resource Management using Deferred Reservations

Samphel Norden and Jonathan Turner
Applied Research Lab,
Dept. of Computer Science,
Washington University, St.Louis,
Email:{samphel,jst}@arl.wustl.edu

Abstract— **In this paper, we consider the problem of resource reservation for networks. Current approaches for resource reservation in Integrated Service Networks adopt an all-or-nothing approach, where partially acquired resources must be released if resources are not available at all routers on the chosen path. Furthermore, under high load, end-systems must retry requests repeatedly leading to inefficient allocation and increased traffic. We propose a new approach called Deferred REServation (DRES) that substantially improves performance (reduces the overall flow rejection probability and increases link utilization) over the all-or-nothing reservation approach. Flow admissibility is increased by deferring requests at routers for a limited period of time until resources are available. Analytical and simulation results confirm the performance benefits of our approach.**

I. Introduction

Continuous streaming media applications such as video-conferencing, video-on-demand and IP telephony are becoming increasingly popular in the internet. Packet loss, delay variations and bandwidth scarcity make the current internet unsuitable for reliable, guaranteed delivery of multimedia services. Real-time applications require resources to be available in advance and must be shielded from transient performance degradation. The traditional best effort Internet thus must be modified to support QoS. A key mechanism to provide guaranteed services is resource reservation. A reservation protocol (e.g.: RSVP [2]) is essential for negotiating resources and for providing QoS guarantees.

When using per-flow based resource reservation for QoS, traditional per-flow resource reservation schemes use a static, all-or-nothing approach to reserve resources for flows. For example, RSVP or ATM UNI [10] based signalling mechanisms either obtain resources at all routers in a route, or tear down the connection even if a single router does not have sufficient resources at the time the request is processed. Even if resources become available immediately afterward, the reservation is not admitted and the user (or the end system) is not informed of that fact. Not even information about the potential future availability is returned to the user, to give an estimate on when to retry the request. Under heavy load, this can lead to the end-user retrying requests repeatedly, requiring the network to propagate and process signalling messages, increasing the network processing overhead. Moreover, signalling messages are typically assigned higher priorities, which could potentially disrupt other services.

In this paper, we study an alternative approach to resource allocation called *Deferred REServation (DRES)*, which enables routers to defer rejection of reservation requests when there are insufficient end-to-end resources. This approach has a real world parallel in the “call-waiting” feature of telephone networks that allows a new call to be “held” until an existing call terminates. The defer period is intelligently computed by the routers so as to increase the probability of a successful reservation, taking advantage of the available knowledge of the network state. We show by simulation that deferring reservations can yield significant performance gains over the all-or-nothing approach, offering improvements in link utilization, and flow admission probability, especially under network overload.

The rest of this paper is organized as follows. Section II motivates DRES, and describes the network model and the basic DRES approach. Section III presents an analytical model for DRES that quantifies performance for a single link scenario. We extend this model for bursty request arrivals and present simulation results in Section III-A and Section IV. Section V presents related work in this area. We conclude in Section VI.

Supported by NSF Grant ANI-9714698

II. Deferred Reservation

A. Motivation

To reiterate, standard resource reservation approaches using ATM signalling or RSVP, attempt to reserve resources at all hops on a path, failing which, the reservation is rejected. We seek to improve performance by not rejecting but deferring a request when resources are insufficient, so as to allow the request to be admitted at a later time.

Figure 1 shows a space-time diagram that highlights the essential difference between the traditional (called NDRES for No Deferred REServation) and DRES modes of reservation. Two reservation requests arrive as shown. Both DRES and NDRES allow req_1 to reserve resources. However, in NDRES, req_2 fails at hop_2 and the resources are released. With DRES, the reservation is deferred for a period and is able to obtain resources after a short delay, allowing req_2 to be completed. Specifically, deferring helped since resources that were allocated to some existing flow were released after the flow terminated. An NDRES session would have to repeatedly poll for resources until it succeeds. This leads to high overhead for the application and increased traffic on the network, as well as lower utilization.

An application that can benefit from deferring is a wireless access environment which requires mobile handoff where the base station of one cell must successfully handoff to the next cell before the signal weakens and the connection is dropped. This maps to a scenario where there is a defer bound during which a request reservation must be admitted. From the overall perspective of the user and the service provider, we have the following benefits of deferring: *Improved Utilization, Reduced processing overhead, and Minimal user effort to obtain a reservation.*

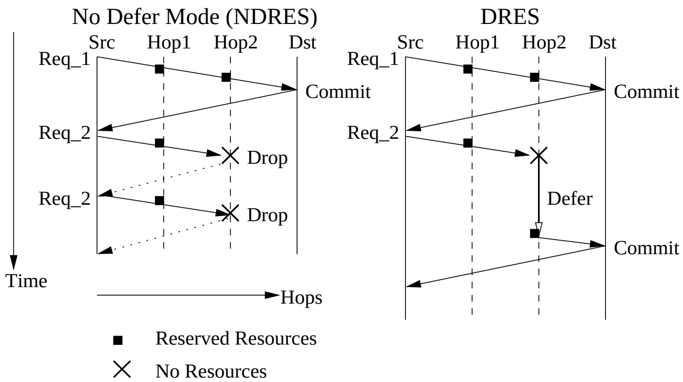

Fig. 1. NDRES vs DRES

B. Basic DRES Protocol

DRES is a 2-phase resource reservation protocol. For the rest of this discussion, we will assume that the QoS for a flow is characterized by bandwidth. The key idea of DRES is to use deferring as a mechanism to accommodate flows that can tolerate initial set-up delays. When a user makes a new flow reservation, the router first locally verifies if sufficient link bandwidth or capacity is available. Typically, link bandwidth can either be in a committed (allocated) or a free state. DRES adds a third state: "reserved" where capacity is allocated at local routers but there is no end-to-end commitment.

When the bandwidth requested (B_i) is not available at an intermediate link, a request is not rejected. Instead, it is queued and deferred at that hop. By delaying requests, DRES can improve the admission probability, since it is likely that by waiting, resources can be committed to the request due to the expiry of an ongoing reservation, or due to the freeing of "reserved" resources. Deferred flows and reserved resources are flushed after a timeout period.

The defer period (T_d) is the key parameter and can be viewed from a user and a network perspective. The user could specify T_d as a QoS constraint. Typically, the longer one defers a request, the better the chance of it being admitted. However, users typically will not be willing to wait for an unbounded time period and will give up on a request that is delayed for too long. There are two perspectives with respect to the defer bound T_d. We can assume that the defer bound is known in advance. However, our results can be applied to situations where reservation requests are maintained until users abandon them. Alternatively, the network operator can set T_d as a matter of policy. The value may vary with network loading conditions or as a function of reservation characteristics or the user's service class.

Description: Figure 2 defines the behaviour of routers implementing the basic DRES protocol along a given reservation path. Arrival of a request (Event A) results in verifying that the request can be admitted. If an end-to-end reservation is possible, then a positive acknowledgement (ACK) is sent by the last hop router.

A. Request Arrival $req_i(B_i)$

IF ($B_i > L_{free}$) set T_d^i;
ELSE
Allocate B_i; Forward req_i
If (Last Hop Router)
Send ACK to source;

B. Timer Expiry: // $T_d^i == 0$
Send NACK back to source; Release resources at all hops along path;
C. Flow Expiry:
Release resources at all hops; Admit jobs from deferred queue;

Fig. 2. Base DRES Algorithm

If not, the request is deferred. When a router receives an ACK, it forwards the request all the way to the source confirming the reservation. If a timer expires (Event B), a negative acknowledgement (NACK) is sent. If an existing admitted flow expires (Event C), then we perform admission control for the first deferred request. L_{free} denotes the available link capacity. The same algorithm is executed independently at every node, and information is only exchanged by the reservation request, acknowledgement, and release messages described above. There are many choices for how to serve waiting reservations. Different choices can have a profound impact on throughput and fairness. In this paper, we focus on attempting to admit those requests that have bandwidth less than the link capacity.

III. Performance of a DRES Multiplexor

In this section, we study the performance of DRES on a single link. If we have exponential interarrival times, exponential holding times (reservation duration), and exponential defer times, there is a simple analytical model that can be used to calculate the flow blocking or rejection probability (See Figure 3). In this model, the transition rates from states i with $i > n$ to states $i-1$ reflect the premature departures from the defer queue caused by reservations which exceed their defer time bound while waiting in the defer queue. The symbol σ denotes the rate at which these early departures occur, where $\sigma = \frac{1}{T_d}$, where T_d is the average defer bound. For constant defer time bounds (the case we are most interested in), this model over-estimates the rejection probability. Unfortunately, it is difficult to model the constant defer time case exactly, since the state of the system must include the time that each waiting flow reservation has left in its defer timer. We have developed an analytical approximation for the constant defer time case that provides a good estimate for the flow rejection probability.

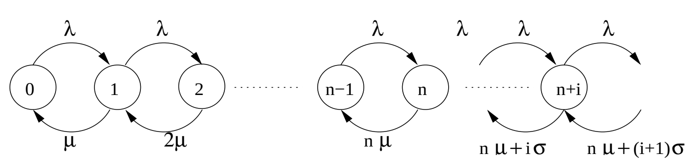

Fig. 3. Constant Defer Time Markovian Model

Consider a DRES multiplexor in which there is a finite defer queue of length k, and flow reservations are queued so long as

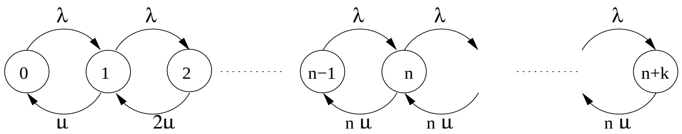

Fig. 4. Defer Queue Approximation Markovian Model

Term	Explanation
λ	Flow arrival rate
μ	Flow completion rate
p_i	Steady state probability of being in State i
δ	Link bandwidth of each flow
n	Number of Servers ($1/\delta$)
T_d	Defer time bound
n	Number of flows supported by link
$k = T_d \cdot n \cdot \mu$	Size of defer queue
MHT	Flow Duration ($\frac{1}{\mu}$)
$\rho = \frac{\lambda}{n \cdot \mu}$	Offered load

TABLE I
NOTATION

there is an empty slot in the queue and reservations stay in the queue until they are accepted (no early departures). If $k = 1 + \lfloor n \cdot \mu \cdot T_d \rfloor$, where T_d is the defer bound, then a reservation is enqueued if and only if the expected waiting time when it arrives is less than or equal to the defer time bound. As we show below (by comparison with simulation), this system provides a close approximation to the original DRES multiplexor with constant defer times. However, unlike the original model, it has a simple analytical model shown below.

Table I summarizes the notation we use in the analysis below. Our goal is to develop a closed form for the flow rejection probability, which is given by p_{n+k}, since flows that arrive in this state are rejected. We can derive the probability of being in state m using standard methods [8].

$$p_m = p_0 \cdot \prod_{i=0}^{m-1} (\frac{\lambda_i}{\mu_{i+1}}) \tag{1}$$

$$= p_0 \cdot \prod_{i=0}^{m-1} (\frac{\lambda}{\mu \cdot min\{i+1, n\}}) \tag{2}$$

Thus we get:

$$p_m = \begin{cases} p_0 \cdot (\frac{\lambda^m}{\mu^m}) \cdot \frac{1}{m!} & m \le n \\ p_0 \cdot (\frac{\lambda^m}{\mu^m}) \cdot (\frac{1}{n^{m-n}}) \cdot \frac{1}{n!} & m > n \end{cases} \tag{3}$$

where:

$$p_0 = \frac{1}{1 + \sum_{m=1}^{n} \frac{(\lambda/\mu)^m}{m!} + \frac{1}{n!} \cdot \sum_{m=n+1}^{n+k} (\frac{(\lambda/\mu)^m}{n^{m-n}})} \tag{4}$$

Figure 5 shows how the flow rejection probability obtained using this model compares to the simulation results of the basic DRES model. We see that our model provides a fairly accurate estimate of the rejection fraction. We will now extend the model

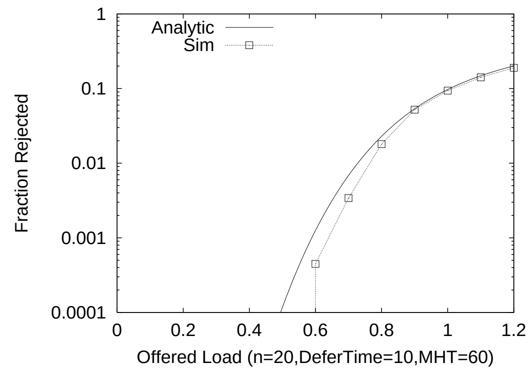

Fig. 5. Comparing the Analytical/Simulation Models for Multiplexor

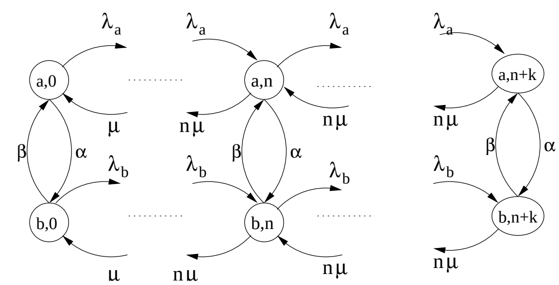

Fig. 6. Bursty Markov Model

to handle bursty request arrivals.

Bursty Arrival Model: We now generalize the analytical model developed in the previous section to handle bursty flow arrival patterns. In particular, we allow the flow arrival rates to alternate between two values λ_a and λ_b. The arrival rate persists at each value for an exponentially distributed period of time. Specifically, the arrival rate of λ_a persists for an average time period of $\frac{1}{\alpha}$, and the arrival rate of λ_b persists for an average period of $\frac{1}{\beta}$. This leads to the Markov chain shown in Figure 6. For this model, we have states characterized by two variables (r,i) where $r \in \{a, b\}$ and i is the number of flows that are either using the link or are waiting in the defer queue. The flow rejection probability in this case is given by $p_a(n+k) + p_b(n+k)$.

The balance equations for the Markov chain can be derived directly from Figure 6. Using standard methods, one can show that $p_a(0), p_b(0)$, satisfy the following set of equations,

$$p_a(0) = \frac{\beta S_{bb} - \alpha S_{ab}}{(\alpha + \beta)(S_{aa} \cdot S_{bb} - S_{ab} S_{ba})} \tag{5}$$

$$p_b(0) = \frac{\alpha S_{aa} - \beta S_{ba}}{(\alpha + \beta)(S_{aa} \cdot S_{bb} - S_{ab} S_{ba})} \tag{6}$$

where $S_{aa}, S_{ab}, S_{ba}, S_{bb}$ are functions of the transition rates only. Given $[p_a(0), p_b(0)]$, the remaining steady state probabilities can be calculated from the balance equations.

A. Performance Results

In all of the subsequent results, we compare the following schemes:
DRES(x): where x represents the defer bound;
NDRES: No Deferring (Traditional reservations (ATM/RSVP))

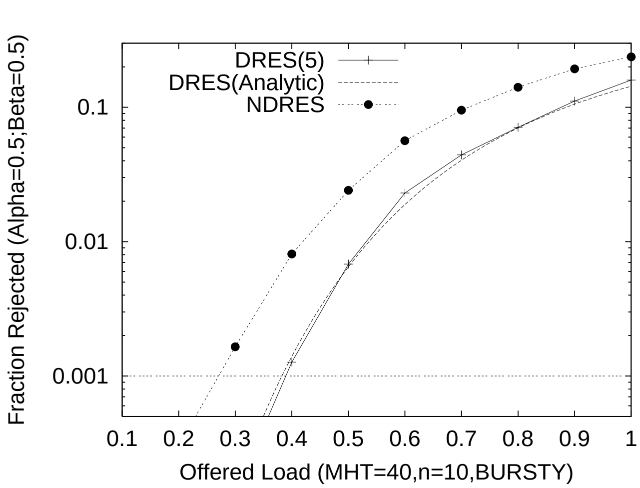

Fig. 7. Inter-burst Time = 0.1 x Call Duration(DeferTime=5)

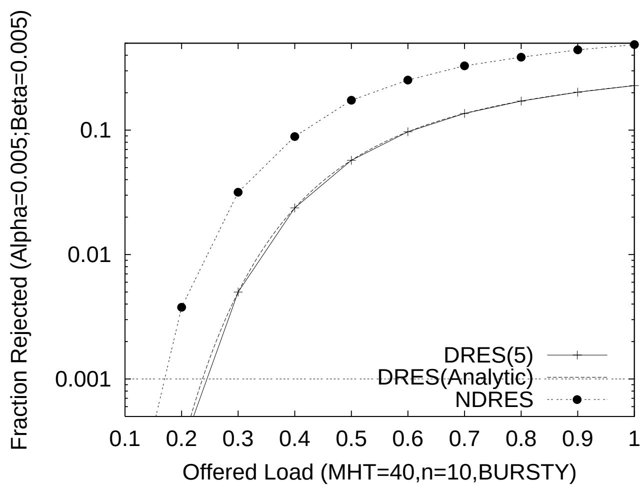

Fig. 9. Inter-burst Time = 10 x Call Duration(DeferTime=5)

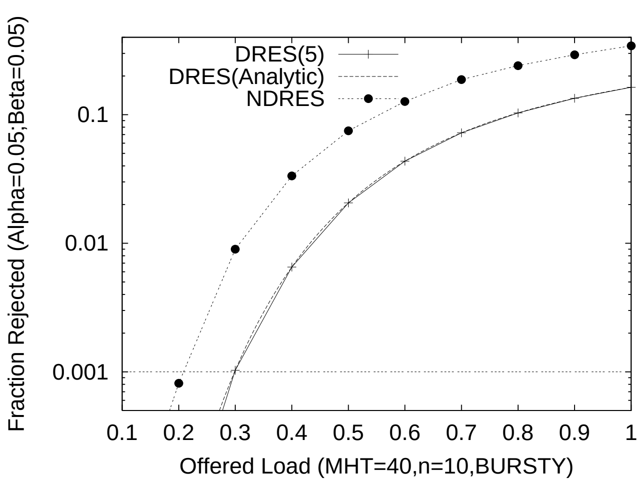

Fig. 8. Inter-burst Time = 1 x Call Duration(DeferTime=5)

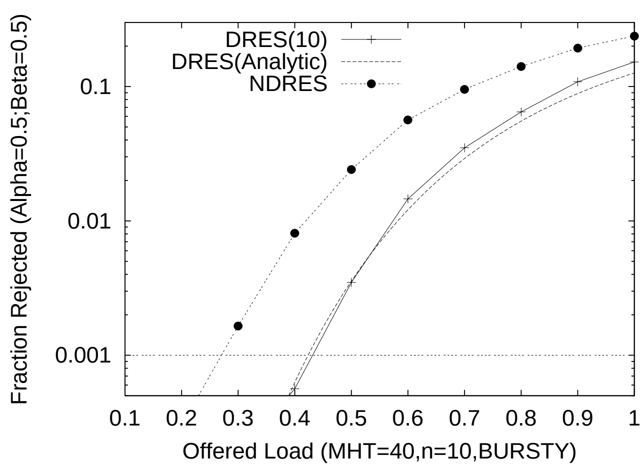

Fig. 10. Inter-burst Time = 0.1 x Call Duration (DeferTime=10)

We evaluated the rejection probability of three schemes, NDRES, and DRES under bursty traffic conditions using both analysis and simulation for the case where $\lambda_a = 0$ (ON/OFF). Simulated DRES closely matched the performance of the analytical model as shown in Figures 7-10.

Figures 7-9 represent three scenarios where the inter burst arrival time ($\frac{1}{\alpha} + \frac{1}{\beta}$) is varied from 0.1,1 and 10 times the flow duration. We chose $\alpha = \beta$ and a flow duration of 40 time units which implies $\alpha = 0.5, 0.05, 0.005$ for the three experiments. We chose a defer time of 5 times units which is 12.5% of the flow duration, and a fixed flow bandwidth of 10% of the total link capacity. As the burst duration increases (decreasing α), the performance degrades as expected. However, in all the plots, we see that DRES significantly outperforms NDRES. In particular, in a system designed for a rejection fraction of 0.001, DRES can carry 48% more traffic than NDRES when $\alpha = 0.5$, 43% more when $\alpha = 0.05$, and 38% more when $\alpha = 0.005$. Comparing Figures 9 and 10, we see that we can get significant additional improvement by increasing the defer time from 5 to 10. Figure 10 shows DRES with a defer bound of 10 units outperforming NDRES by *60%*.

IV. Performance of a simple DRES Network

We study the performance of the bursty model on a ring topology, in which 20 routers are connected together in a ring, and each router has a traffic source associated with it. Each source generates flow reservations for the destination that is 5 hops away, and the reservation bandwidth is uniformly distributed with a maximum bandwidth of 5% of the link. Figure 11 shows the performance of DRES with a defer bound of 10 units supporting *twice* the load compared to NDRES at a rejection probability of 10^{-3}, while DRES with a bound of 5 units shows a 60% gain over NDRES. At a higher probability of 10^{-2}, DRES(10) has a 50% gain and DRES(5) has a 37% gain over NDRES. As we increase the burst duration to 40 units in Figure 12, the performance benefits decrease to around *15%* for DRES(10) at a rejection fraction of 10^{-3} and *11%* at a rejection fraction of 10^{-2}.

Additional experiments were performed [12] on topologies such as the parking lot topology (which studies the impact of cross-traffic on deferring). We also investigated performance of DRES on a typical ISP topology using AT&T's worldnet network. We also performed experiments with different traffic classes (small and large bandwidth requests). In addition,

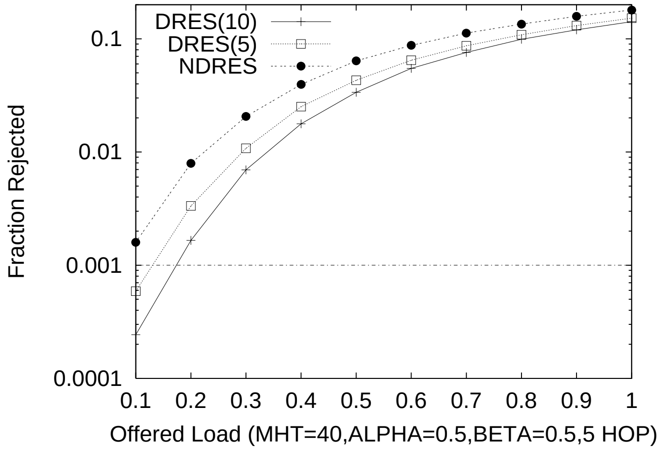

Fig. 11. (Ring Topology)Inter-burst Time = 0.1 x Call Duration

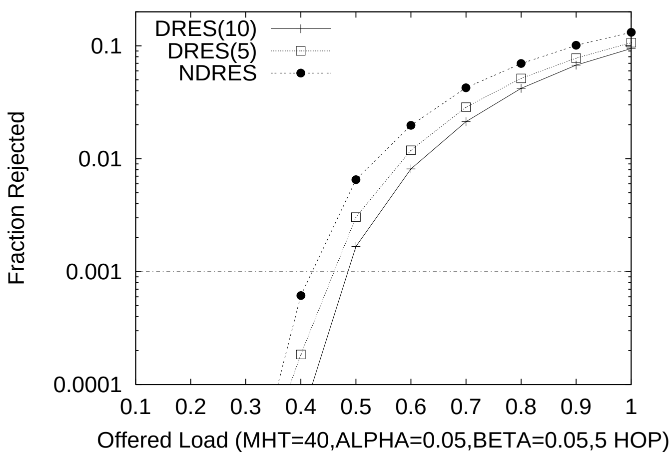

Fig. 12. (Ring Topology)Inter-burst Time = 1 x Call Duration

we also have preliminary results on the impact of QoS routing on DRES. We find that DRES provides a significant percentage gain under all these different scenarios. Due to space constraints, these results are shown in [12].

V. Related Work

There have been numerous proposals to perform resource reservation in Integrated Service networks. In this section, we highlight some of the prominent ones that are relevant to our approach. YESSIR [1], a resource reservation protocol for RTP traffic has a concept of *partial reservations* which are made when there are insufficient resources at routers. However, in such cases, YESSIR simply informs the source about the insufficient bandwidth and pushes the problem of deciding whether to reduce the requested bandwidth or to drop the flow altogether to the end-host. YESSIR does not offer end-to-end guarantees with its notion of *partial reservations*. More importantly, having a combination of best-effort reservations along with actual reservations (as results from partial reservations), results in a large number of flows of poor quality as mentioned by the authors, which is unacceptable for providing QoS guarantees.

Recent work on advance reservations [6], [3], [4], [5], [9] propose mechanisms which differentiate between an *immediate* reservation where the flow duration is not known, and an *advance* reservation where the duration is known, and the reservation process is initiated far ahead of time. References [6], [5] describe mechanisms which allow the preemption of *immediate* flows, to allow *advance* reservations to access the link. In such cases, the QoS is downgraded for *immediate* reservations. Also, *immediate* reservations could be dropped so as to accommodate reservations in the future leading to no guarantees for *immediate* reservations. The Tenet real-time protocol suite [7] partitions resources into advance reservations and non-preemptible normal reservations, which can lead to fragmentation.

VI. Conclusions

In this paper, we highlighted the problems with traditional approaches of RSVP/ATM-style reservations under high network load. When resources are momentarily unavailable, the session initiator needs to manually retry, leading to higher network and signalling processing overhead. To resolve these problems, we introduced DRES, a way of intelligently delaying RSVP/ATM-style resource reservation requests for a short defer time. Using simulation, we showed that this significantly improves the chance of admissibility. Depending on the load and situation, these improvements can be quite significant. We believe that introducing such a mechanism into a resource reservation system would enhance network quality as perceived by the users, while reducing processing load at the network nodes due to moving from a polling-based system to an ”interrupt-driven” paradigm. We are currently investigating the impact of QoS routing, as well as mechanisms to predict the defer time so as to only defer when necessary.

References

[1] Pan P., and Schulzrinne H. “YESSIR: A simple reservation mechanism for the internet”, *Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, July, 1998.

[2] Braden B., Zhang L., Berson S., Herzog S., and Jamin S. “Resource Reservation Protocol (RSVP) - Version 1 Functional Specification”, *In RFC 2205*, September 1997.

[3] Schill A., Kuhn S., and Breiter F. “Resource reservation in advance in heterogeneous networks with partial ATM infrastructures”, *Proceedings of IEEE INFOCOM’97*, March 1997.

[4] Wolf C. L., *et. al*. “Issues of reserving resources in advance”, *Proceedings of NOSSDAV’95*, April 1995.

[5] Greenberg A. G., Srikant R., and Whitt W. “Resource sharing for book-ahead and instantaneous-request calls”, *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, February 1999.

[6] Schelen O., and Pink S. “Sharing resources through advance reservation agents”,*Proceedings of IWQoS’97*, May 1997.

[7] Banerjea A., *et. al*. “The tenet real-time protocol suite: Design, implementation, and experiences”, *Technical Report TR-94-059*, Department of Computer Science, University of California at Berkeley, 1994.

[8] Kleinrock L. “Queuing systems, Volume 1: Theory”, *John Wiley & Sons*, 1975.

[9] Guerin R., and Orda A. “Networks with advance reservations: the routing perspective”, *Proceedings of INFOCOM’2000*, March 2000.

[10] ATM Forum. “ATM User-Network Interface Signaling Specification v4.0”, 1995.

[11] Hafid A. “Providing a scalable video-on-demand system using future reservation of resources and multicast communications”, *Proceedings of IWQoS’97*, May 1997.

[12] Norden S. “DRES: Internet resource management using deferred reservations”, *Technical Report WU-CS-01-06, Department of Computer Science, Washington University*, http://www.arl.wustl.edu/∼samphel/wu-cs-01-06.ps