Report Number: WUCS-01-25

2001-01-01

Performance of Deferred Reservations in Data Networks

Samphel Norden and Jonathan Turner

This paper studies the performance of deferred resource reservation in data networks. Conventional resource reservation protocols, such as PNNI and RSVP adopt an all-or-nothing approach, where partially acquired resources must be released if resources are not available at all links on the chosen path. During periods of high network load, this leads users to retry requests repeatedly, adding control traffic at exactly the time when the network's capacity to process that control traffic is exhausted. Deferred REServation (DRES) can significantly improve performance by reducing the overall call rejection probability, allowing more traffic to be carried, using the same resources.... **Read complete abstract on page 2.**

Performance of Deferred Reservations in Data Networks

Samphel Norden and Jonathan Turner

Complete Abstract:

This paper studies the performance of deferred resource reservation in data networks. Conventional resource reservation protocols, such as PNNI and RSVP adopt an all-or-nothing approach, where partially acquired resources must be released if resources are not available at all links on the chosen path. During periods of high network load, this leads users to retry requests repeatedly, adding control traffic at exactly the time when the network's capacity to process that control traffic is exhausted. Deferred REServation (DRES) can significantly improve performance by reducing the overall call rejection probability, allowing more traffic to be carried, using the same resources. Call admissibility is increased by deferring requests at routers for a limited period of time until resources become available. The paper includes an analysis of the performance of a DRES multiplexor, for Poisson and bursty reservation arrival processes, and simulation results for substantial network configurations, using several different QoS methods.

Performance of Deferred Reservations in Data Networks

Samphel Norden and Jonathan Turner

WUCS-01-25

September 2001

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

Performance of Deferred Reservations in Data Networks

Samphel Norden, Jonathan Turner*†

Abstract

This paper studies the performance of *deferred resource reservation* in data networks. Conventional resource reservation protocols, such as PNNI and RSVP adopt an all-or-nothing approach, where partially acquired resources must be released if resources are not available at all links on the chosen path. During periods of high network load, this leads users to retry requests repeatedly, adding control traffic at exactly the time when the network's capacity to process that control traffic is exhausted. Deferred REServation (DRES) can significantly improve performance by reducing the overall call rejection probability, allowing more traffic to be carried, using the same resources. Call admissibility is increased by deferring requests at routers for a limited period of time until resources become available. The paper includes an analysis of the performance of a DRES multiplexor, for Poisson and bursty reservation arrival processes, and simulation results for substantial network configurations, using several different QoS routing methods.

1 Introduction

This paper describes a deferred reservation mechanism for providing quality of service guarantees to reservation-oriented applications in the Internet. Conventional resource reservation protocols (RSVP [2] in the Internet and PNNI [11] in ATM networks) adopt an all-or-nothing approach to resource reservation. If any link on the path to the destination lacks the resources to support a given reservation request, the request fails, forcing the user to give up or try again later. Deferred reservations provide another option, allowing the network to delay its response to a reservation request for a short

*Samphel Norden and Jonathan Turner are with the Department of Computer Science, Washington University, St.Louis, Email:{samphel,jst}@arl.wustl.edu.

†This work is supported in part by NSF grant ANI-9714698.

period of time, in anticipation of the needed resources becoming available, as other sessions terminate and release them. The use of reservation deferral can significantly reduce the likelihood of reservation failure, particularly in situations where each reservation constitutes a significant fraction of a single link's resources. This can be a common occurrence in wireless access settings and at the typically constrained interfaces linking ISP networks and enterprise networks.

This paper presents a specific deferred reservation protocol (DRES) and studies its performance on both single links and more complex network configurations. DRES is a sender-initiated reservation protocol in which reservation requirements are used to guide routing decisions. We study DRES performance in the context of several different QoS routing protocols and show that the choice of routing protocol can have a significant impact on performance.

Figure 1 shows a space-time diagram that highlights the essential difference between conventional resource reservation (referred to as NDRES for No Deferred REServation) and DRES. 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 reservation would have to repeatedly poll for resources until it succeeds. This leads to high overhead for the end-user and increased traffic on the network. DRES can increase network utilization, reduced processing overhead, and reduce the user effort needed to obtain a reservation.

DRES can be implemented using a simple 2-phase resource reservation protocol. When a user initiates a new flow reservation, the reservation is propagated through the network using a suitable reservation routing protocol. Each router along the path determines if the required bandwidth is available on the link to the next router in the path, and if so, reserves the resources and propagates the reservation request. If the required bandwidth is not available, the request is timestamped and placed on a *defer queue* for the link. As other flows terminate and release their reserved bandwidth, those resources are allocated to pending reservation requests. If a deferred reservation is not satisfied within its *defer bound*, it is removed from the defer queue and all resources reserved for it by previous routers are released. The time periods spent by a deferred reservation in different hops are summed as the reservation request propagates. The reservation fails only when the

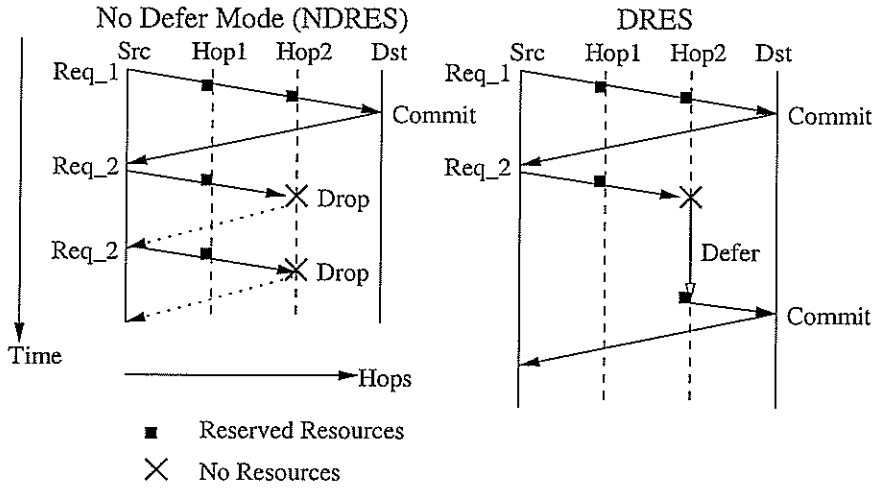

Figure 1: NDRES vs DRES

sum of the times spent waiting at all hops exceeds the defer bound.

The defer period (T_d) is a key parameter and could potentially be specified by either the user or the network. The user could specify T_d as a QoS constraint. Typically, the longer one is willing to have a request deferred, 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. A network operator might also set T_d as a matter of policy. The value might vary with network loading conditions or as a function of reservation characteristics or the user's service class. In this paper, we focus on the simplest case of a fixed defer bound, which is uniformly applied to all reservations.

2 Performance of a DRES Multiplexor

In this section, we study the performance of DRES on a single link capable of supporting up to n reserved flows at one time. 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 2). 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.

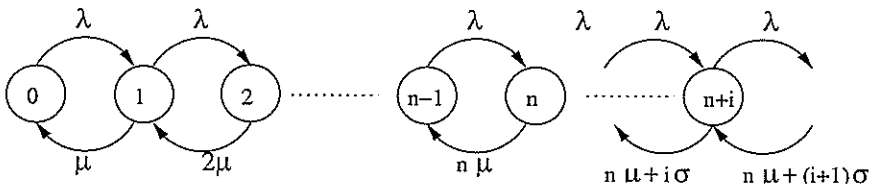

Figure 2: Constant Defer Time Markovian Model

Consider a DRES multiplexor in which there is a finite defer queue of length k, and flow reservations are queued so long as there is an empty slot in the queue and reservations stay in the queue until they are accepted (no early departures). If $k = 1 + \lfloor n\mu T_d \rfloor$, 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. This multiplexor has a simple analytical model shown in Figure 3. The steady-state probabilities p_i are easily determined using standard methods and the reservation rejection probability is just the steady state probability of state $n+k$, p_{n+k}. Such an analysis yields the following equations.

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

Thus we get:

$$p_m = \begin{cases} p_0 \cdot (\frac{\lambda}{\mu})^m \cdot \frac{1}{m!} & m \leq n \\ p_0 \cdot (\frac{\lambda}{\mu})^m \cdot (\frac{1}{n^{m-n}}) \cdot \frac{1}{n!} & m > n \end{cases} \tag{2}$$

where:

$$p_0 = \frac{1}{1 + \sum_{m=1}^{n} \frac{(\lambda/\mu)^m}{m!} + \frac{1}{n!} \cdot \sum_{m=n+1}^{n+k} (\frac{(\lambda/\mu)^m}{n^{m-n}})} \tag{3}$$

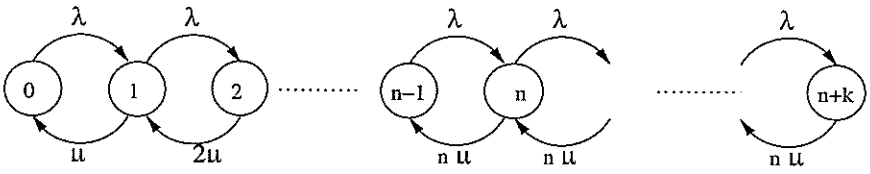

Figure 3: Defer Queue Approximation Markovian Model

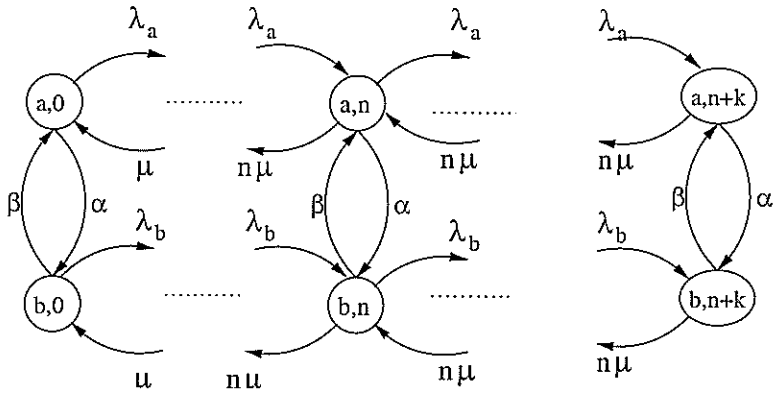

Figure 4: Bursty Markov Model

From Figure 5, we see that our model provides a fairly accurate estimate of the rejection fraction. The legends *{DRES(Mux,Sim),DRES(Mux)}* refer to the simulation and analytical models respectively, and *NDRES(Mux)* refers to NDRES on a single link (multiplexor).

Bursty Arrival Model: We now generalize our analytical model to handle bursty flow arrival patterns. In particular, we allow the flow arrival rates to alternate between two values λ_a and λ_b using two coupled markov chains A and 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 $1/\alpha$, and the arrival rate of λ_b persists for an average period of $1/\beta$. This leads to the Markov chain shown in Figure 4. 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 $p_a(n+k)+p_b(n+k)$, where $p_a(i)$ and $p_b(i)$ are the steady state probabilities of being in state (a, i) and (b, i), respectively.

The balance equations for the Markov chain can be derived directly from Figure 4. Using standard methods, one can show that $p_a(0), p_b(0)$, satisfy

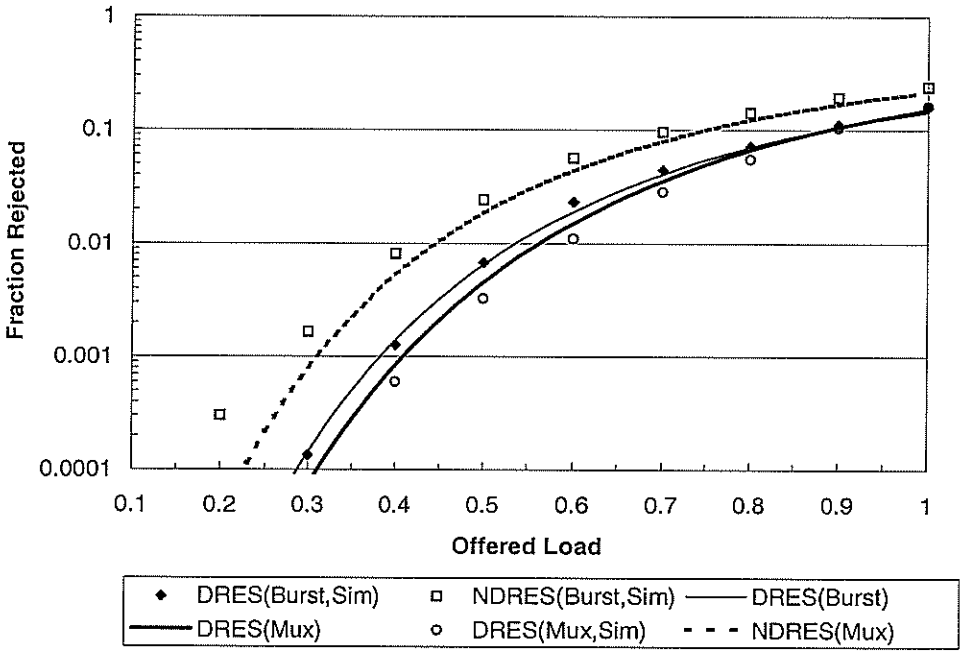

Figure 5: Performance of Analytical/Simulation models on a link

the following set of equations,

$$p_a(0) = \frac{\beta S_{bb} - \alpha S_{ab}}{(\alpha+\beta)(S_{aa} \cdot S_{bb} - S_{ab}S_{ba})} \tag{4}$$

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

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.

Figure 5 shows results comparing DRES and NDRES with bursty traffic, where the simulation and analytical models are denoted by *DRES(Burst,Sim)* and *DRES(Burst)* respectively. These results are for a link fraction of 10%, a defer bound of 5 and mean holding time (MHT) of 40. where $k = 2$, $\lambda_a = 0$, $\alpha = \beta$ and $1/\alpha + 1/\beta$ equal to one tenth of the mean reservation duration for the bursty model. The chart shows results from both the analysis and simulation for DRES, confirming the accuracy of the analytical model. More significantly, the chart shows that at a rejection fraction of 10^{-3}, DRES can carry almost 50% more traffic than NDRES. This is significant because a well-engineered network will be designed to maintain a

small rejection fraction under normal traffic conditions. The results show that DRES can significantly increase the traffic carried by a given link at an acceptable rejection fraction. (The choice of .001 as a target rejection fraction is somewhat arbitrary, but is an appropriate choice for reservation blocking.) Additional numerical results for DRES multiplexors can be found in [13, 19].

3 Performance of DRES on a Path

In this section, we discuss simulation results for DRES on a simple path, as shown in Figure 6. The path comprises six routers and the performance is measured for requests going from the host at the first router to the host at the last router. At each intermediate router, we introduce cross traffic that goes one hop away. The reservation bandwidth is a uniform random variable in the range $[0, LF]$, where LF is the *Link Fraction*, the amount of the link's capacity consumed by a single reservation. The cross traffic requests are generated using an ON/OFF model [9] with exponentially distributed ON and OFF times. During each ON period, a geometrically distributed number of requests are generated with a mean N at a fixed rate of p requests per time unit. The OFF time is an exponentially distributed value with mean I. This gives an average request generation rate of $I/N + 1/p$. The values of I,N and p are 100,10 and 1 respectively. The duration of the cross traffic reservations is 20% of the duration of the end-to-end reservations. The call duration (MHT) is 60 time units. In addition to DRES and NDRES, we consider an NDRES variant ($ND{+}k$), where the request is retried up to k times during the defer period (so the time between retries T_d/k).

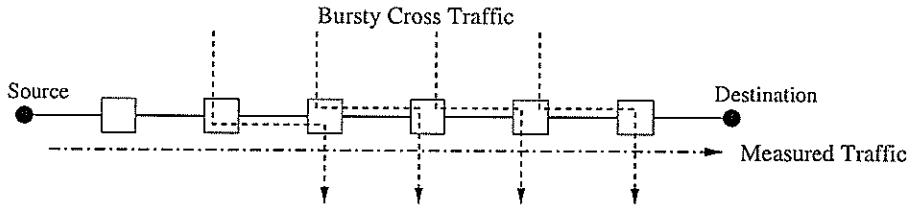

Figure 6: Path Topology

Figures 7-8 show the rejection fraction for link fractions of 0.05 and 0.1. When $LF = .05$, the path can carry about 40% more end-to-end traffic with DRES than it can with NDRES, while maintaining a rejection probability of .001. When $LF = .1$, the path can carry five times as much

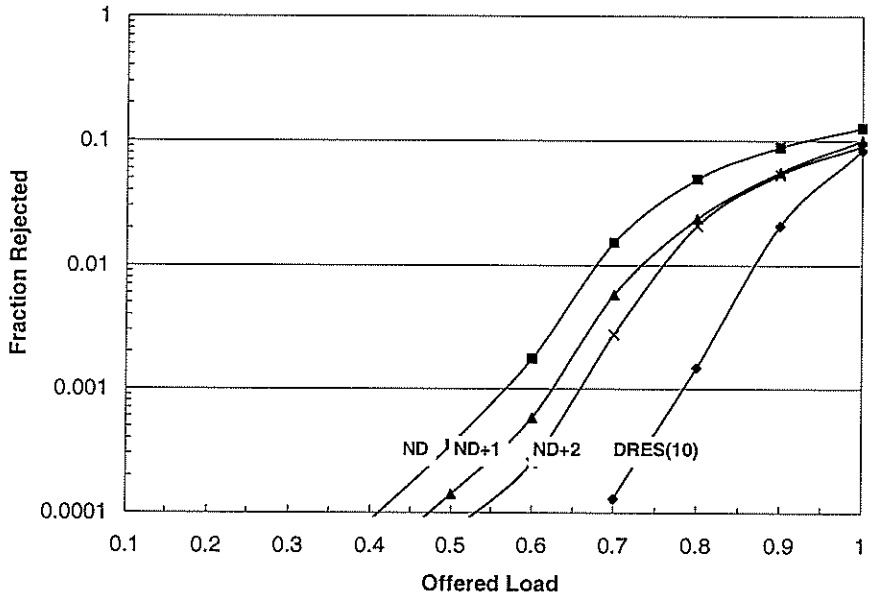

Figure 7: Rejection Fraction for a Path (LF = 0.05)

end-to-end traffic with DRES as it can with NDRES, while maintaining a rejection probability of .001. The addition of retries to NDRES yields significant improvements, but if the number of retries is limited to 1 or 2, the performance still falls short of what DRES provides. While more retries yield further improvements, they also increase network control overhead substantially.

Figure 9 shows how the performance of DRES is affected by the defer bound, ($TD(x)$ represents DRES with a defer bound x). Reducing the defer bound from one sixth of the mean reservation duration, to half that, reduces by 30% the amount of traffic that can be carried while maintaining a rejection fraction of .001. Note that even when the defer bound is just 3% of the reservation duration, the amount of traffic that can be carried with DRES is more than 3 times the amount that can be carried with NDRES (assuming a target rejection fraction of .001).

4 Routing and DRES

A network that supports flow reservation must select a suitable path for a flow before it can reserve the required resources. This is also true for networks that support deferred reservation. In this section, we define several

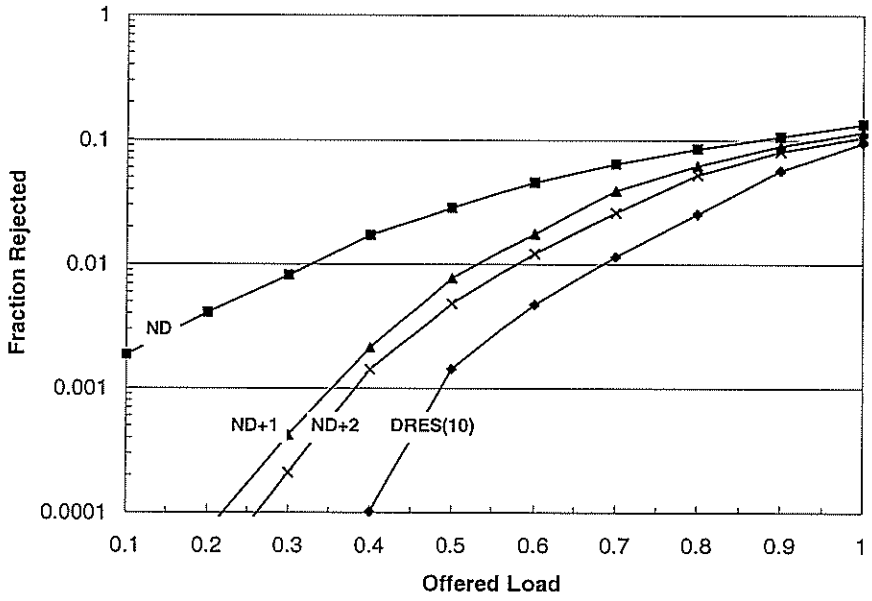

Figure 8: Rejection Fraction for a Path (LF = 0.1)

QoS routing algorithms that can be used in networks supporting flow reservation, including deferred reservation. The effect of the different routing algorithms on the performance of DRES and NDRES is evaluated in the next section.

4.1 Bandwidth Distance Propagation Algorithm (BDP)

The *Bandwidth Distance Propagation* (BDP) algorithm is a distance-vector style of algorithm, which propagates information about the length and bottleneck bandwidth associated with available paths to each destination. Reservation requests are forwarded to a next hop router selected using this information. While, BDP shares with other distance-vector algorithms the possibility of creating routing loops, looping reservation requests are easily detected, since the reservation packet includes a list of the routers on the path.

In the BDP algorithm, neighboring routers exchange information about paths to each network destination. The basic information element stored at a router is a tuple of the form (d, h, λ, β). Such a tuple signals the existence of a path to destination d through the next-hop router h, that has path length λ and bottleneck bandwidth of β from h to the destination. Based

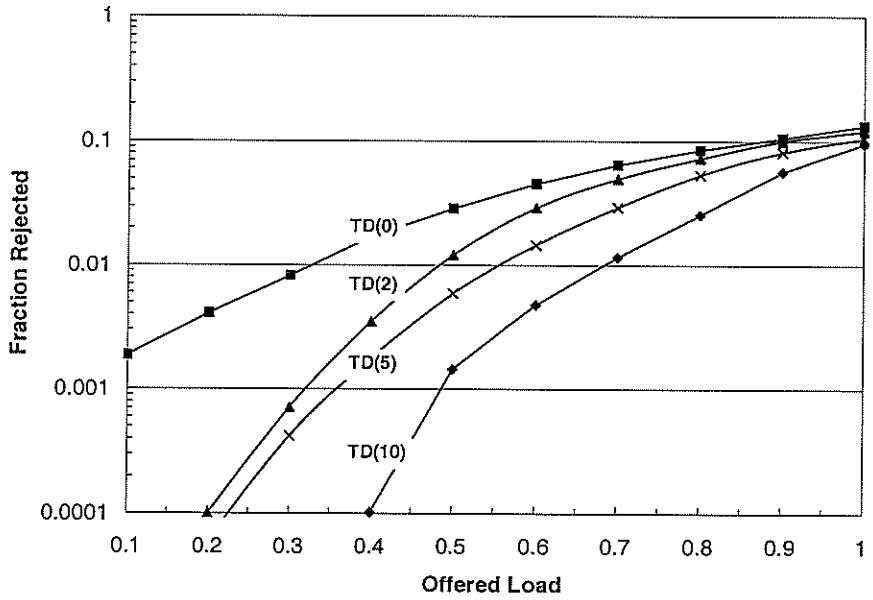

Figure 9: Effect of Defer Bound on Rejection Fraction (LF = 0.1)

on this, a reservation request for destination d may be forwarded to h if the reservation bandwidth is no more than λ and no more than the available bandwidth on the connecting link to h. The path length is the sum of the lengths of the individual links on the path, not simply the hop count. A router may have several tuples for a given destination.

A router with a tuple (d, h, λ, β) will "propagate" the tuple to all of its neighbors except h (the exclusion of h helps prevent the creation of routing loops). A tuple (d, h, λ, β) is propagated by a router x to its neighbor y as the tuple $(d, x, \lambda + \lambda_1, \min\{\beta, \beta_2\})$ where λ_1 is the length of the link joining x and y, and β_2 is the available bandwidth on the link joining x and h.

Through this process, a router x may receive many tuples for a given destination d. A small number of these are retained. The selection of tuples to retain is done as follows. First, we discard a tuple (d, h, λ, β) if λ exceeds a fixed bound $M(x, d)$ on the allowed length of paths from x to d. The removal of these tuples, prevents reservations from using excessively long paths during periods of heavy load. Next, if one tuple *dominates* another, then we discard the dominated tuple. A tuple $(d, h_1, \lambda_1, \beta_1)$ dominates a tuple $(d, h_2, \lambda_2, \beta_2)$ if $\lambda_1 \leq \lambda_2$ and $\beta_1 \geq \beta_2$. The tuples that remain after dominated tuples are dropped are called dominant tuples. If there are too many dominant tuples for a given destination we drop some of them, using

a simple heuristic procedure.

The length and bottleneck bandwidth coordinates of a set of dominant tuples can be plotted on a two-dimensional coordinate system with length on the horizontal axis. The points define a "stair-case" function, and any subset of the points defines another. We select a subset of points, of the target size, that minimizes the difference between the two functions, that is, one that minimizes the difference in the areas under the two curves. Such a subset is straightforward to compute.

To select the next hop for a reservation request that originated at router x and is going to destination d, the BDP algorithm first determines the set of *viable tuples* for the reservation. A tuple (d, h, λ, β) is viable if the reservation bandwidth is no larger than the available bandwidth on the connecting link to h, and the length of the path constructed so far, plus λ is no larger than $M(x, d)$. From the set of viable tuples, the algorithm selects the tuple with the smallest length that has a bottleneck bandwidth that is at least as large as the reservation bandwidth. If there is no such viable tuple, it selects the one with the largest bottleneck bandwidth. The required bandwidth is reserved on the link to the next hop for the selected tuple and the reservation request is forwarded to the next hop. If there are no viable tuples then the reservation is rejected in the NDRES case. In the DRES case, the reservation is deferred until some tuple for the destination becomes viable. This will most often occur as a result of some flow ending and releasing bandwidth on an incident link, but it may also occur as a result of a new tuple being received from a neighboring router. Of course, if the defer bound is reached before any tuple becomes viable, then the reservation is rejected.

4.2 Least Combined Cost Routing (LCC)

The Least Combined Cost Routing (LCC) algorithm selects a route for a flow reservation by computing a least cost path at the source router where the reservation request first enters the network, then forwarding the reservation request along this path, reserving resources at each hop. If at some point on the path, the selected link does not have sufficient capacity for the reservation, then the reservation is rejected in the NDRES case, and deferred in the DRES case. The cost metric used in the path computation takes into account both the distances spanned by the links and the amount of available bandwidth relative to the reservation bandwidth. It requires an underlying routing information distribution algorithm, to periodically update the necessary link state information.

Term	Explanation
B	Available bandwidth on a link
R	Reservation bandwidth
$L[u,v]$	Length of link joining routers u and v
M	Bandwidth "margin" where $M = B - R$

Table 1: Notation for Cost Metric

The cost metric is motivated by the observation that whenever the network is lightly loaded, paths should be selected to minimize the sum of the link costs, since this minimizes the network resources used, and the network delay. When some links are heavily loaded, we want to steer traffic away from those links, even if our most recent link state information indicates that they have enough capacity to handle the flow reservation being setup. The reason for avoiding such links is that in the time since the last link state update, the link may have become too busy to handle the reservation. Rather than risk setting up the reservation on a path with a high likelihood of failure, we would prefer a longer path with a smaller chance of failure.

Table 1 lists several key pieces of notation used in the link cost expression shown below which is referred to as the Combined Cost Metric (CCM).

$$C[u,v] = \{L[u,v] + \alpha(\max(0, g - M))^{\beta}\} \times R$$

The three parameters, *alpha*, *beta* and g determine how the cost of a heavily loaded link increases. Specifically, if the link's *margin* (the amount of available bandwidth remaining after subtracting the bandwidth required by the reservation) is greater than g, then the cost of the link is equal to its length. If its margin is less than g, then its cost increases as the margin shrinks (note the margin may be less than zero). G should be chosen to reflect the likelihood that in the time between the last link state update and the arrival of a reservation, that the link has become too busy to handle the reservation. Specifically for margins of g or greater, the probability of making a bad route selection based on stale link state information should be small, say 1-5%. If the average reservation bandwidth is a small fraction of the link bandwidth, then a reasonable choice for g would be 5 times the average reservation bandwidth. The parameter β determines how rapidly the cost grows as the margin drops. In the simulation results reported in the next section, β is set to 2, giving quadratic growth.

To determine a reasonable choice for the scaling parameter α, consider the appropriate cost increment in a situation where the margin is equal to

zero. Note, that in the time since the last link state update, the "true" margin may have either increased or decreased. If we assume that both possibilities are equally likely, then the added cost when the margin is zero should balance the cost of the two different "incorrect" routing decisions that are possible. A decision to use a path with a zero margin link is incorrect, if that link no longer has enough bandwidth to accommodate the reservation. A decision to not use a path with a zero margin link is incorrect if the link actually does have sufficient capacity for the reservation. The cost of the first type of incorrect decision is that the reservation is rejected (in the NDRES case) or that the reservation is deferred (in the DRES case) and possibly rejected later. The cost of the second type of incorrect decision is that a longer, higher cost path is used, wasting network resources. This added cost is $\alpha g^{\beta} R$. We equate the cost of rejecting a reservation request to the cost of the resources that the reservation would use if it were accepted and used a minimum length route. If this minimum route length is D, then the cost of rejecting the reservation is DR. Setting this equal to $\alpha g^{\beta} R$ and solving for α gives $\alpha = D/g^{\beta}$. To avoid the implied requirement to calculate D and α for each reservation, we simply specify α based on a typical value of D. Note that it is reasonable to use different values of α in the DRES and NDRES cases, since the cost of making an incorrect decision to use a path containing a zero margin link that is no longer able to accept the reservation is smaller in the DRES case, than in the NDRES case. We report results both for the case of equal values of α and the case in which a smaller value is used for DRES.

We do not consider the effects of attempting to re-route a reservation request, when a link selected by the routing algorithm turns out to have insufficient capacity, since in the DRES case, it's not clear how one would decide between rerouting and deferring. While one can certainly make a case for rerouting in the NDRES case, we have found that adding rerouting to NDRES yields very small improvements under the traffic conditions studied.

4.3 Parallel Probe Algorithm (PP)

In both the BDP and LCCR algorithms, routing information is distributed throughout the network to facilitate the routing of reservation requests. One drawback of this approach is that routers must maintain a great deal of information, much of which is never used. Indeed, if no reservation consults a particular piece of routing information before the next update replaces it, then that piece of routing information served no purpose, and the effort spent to create it was wasted.

The Parallel Probe (PP) algorithm takes a different approach. Rather than maintain a lot of dynamic routing information, it sends *probe packets* through the network to collect routing information as it is needed. This means that no extraneous routing information must be maintained. Only that information that is relevant to the selection of paths for actual flow reservations is required.

The PP algorithm uses a precomputed set of paths for each source-destination pair. Probe packets are sent in parallel on all of these paths to the destination, and are intercepted by the last hop router. As the probe packets pass through the network, each router on the path inserts a field specifying the available bandwidth on its outgoing link. This operation is simple enough to be implemented in hardware, allowing probes to be forwarded at wire speed.

When the probe packets reach the last hop router, it selects the best path for the flow, based on the information received. Each probe packet includes a field indicating how many probes were sent, allowing the last hop router to easily determine when it has received all the probes, in the normal case where all probes are received. If one or more probes is lost, the last hop router will proceed following a timeout. The last hop router selects the shortest path for which the bottleneck bandwidth is at least equal to the reservation bandwidth, if there is one or more such path. If there is no such path, the reservation is dropped in the NDRES case. In the DRES case, the last hop router selects the path with the largest bottleneck bandwidth and continues.

If the last hop router selects a path with a large enough bottleneck bandwidth to handle the reservation, it sends a reservation message back along the selected path to the origination point, reserving resources as it goes. If in the short time since the probe packet was forwarded, a link has become too busy for the reservation, the reservation attempt fails and all reserved resources are released.

If the last hop router selects a path that does not have a large enough bottleneck bandwidth for the reservation (this may occur with DRES), it sends a message back to the originating router, indicating which path was selected, but not reserving any resources. The originating router sends the reservation request along the selected path in the forward direction. At any link where the available bandwidth is not sufficient, the reservation is deferred.

4.4 Qualitative Comparison

The three protocols presented above represent a wide range of approaches. At one extreme, we represent hop-by-hop protocols such as RIP in *BDP*. The *LCC* protocol represents protocols based on Dijkstra's shortest path algorithm such as OSPF. The *PP* approach represents a hybrid multi-path routing scheme. We examine these protocols with respect to essential routing metrics such as the *call setup overhead*, *message overheads*, and *Router processor complexity*. In terms of the *call setup time*, *BDP* adopts a hop-by-hop approach and at each stage the next hop is decided from a viable set of tuples using a very simple procedure. The *PP* algorithm also has low setup time since probes simply query hardware port processors using precomputed paths. While there is additional processing at the last hop, the procedure effectively takes a round trip time. The *LCC* protocol has the longest call setup time since it computes the shortest path on-demand. From the *message overhead* perspective, both BDP and LCC send a single reservation request on the chosen path. The *PP* algorithm sends probes on k paths incurring a slightly higher overhead. However, since no resources are reserved in the forward pass, this does not lead to any wastage. *Router processor complexity* refers to the complexity in processing information from other routers as well as processing reservation requests. *LCC* has relatively low complexity requirements where residual link bandwidth is the only state information that needs to be exchanged. While *PP* probes interact directly with hardware, this is still a necessity in order to do fast processing. Processing state information from updates is a negligible task since probes perform that service. *BDP* has a large overhead comparatively, since it needs to maintain tuples, in particular the bottleneck capacity on multiple paths. However, by performing hop-by-hop routing, BDP seeks to eliminate the stale information that could be used to guide the decisions for the *PP* and *LCC* schemes.

5 Simulation Results

This section presents performance results for DRES in more complex network contexts, where routing plays a significant role. We present results for two network configurations. The first is a 10×10 torus. While not particularly representative of real networks, it provides a simple, uniform topology with a rich set of alternate paths to choose from. Because of its uniformity, it provides a more neutral setting than do more realistic network topologies. The second network configuration was chosen to be more representative of

a real wide area network. This network has nodes in each of the 20 largest metropolitan areas in the United States. The traffic originating and terminating at each node is chosen to be proportional to the population of the metro area served by the node, and the traffic between nodes is also chosen, based on the populations of the two nodes. This leads to the sort of uneven traffic distribution that is typical of real networks. The links in the network are also dimensioned to enable them to carry the expected traffic. Dimensioning links to have appropriate capacity is important for a realistic study of routing, since a badly engineered network can easily distort the results, leading to inappropriate conclusions about the relative merits of different routing algorithms. Further details on the national network configuration are given in section 5.2.

5.1 Results for the Torus Configuration

This section reports results on a 10×10 torus. Reservation requests arrive at the same rate at every node. The interarrival times and the reservation holding times are exponentially distributed. Uniform reservation bandwidths are used (that is, all reservations have the same bandwidth). All requests go between pairs of nodes that are three rows and three columns apart, leading to a completely uniform traffic distribution.

There are a number of numerical parameters that are varied in the simulations. Each parameter has a *default value*. Whenever one parameter is varied in a given chart, the other parameters are assigned their default values. The default value of the link fraction is .1. The default value of the defer bound is $1/4^{th}$ the reservation holding time (MHT). For LCC, the default update period (time between state updates) is $MHT/2$. For BDP, the default update period is $5 \times MHT$ (BDP uses a larger update period to compensate for the larger volume of information that must be passed in each update with BDP). The default bound on the number of tuples propagated by BDP for each destination is 6. The default number of alternate paths on which PP sends probes is 6. DRES based schemes are represented as *solid lines*, while NDRES based schemes are represented as *dashed lines* in all subsequent figures. The vertical line on the plots indicates the *default value* when that value is plotted on the the x-axis.

Figure 10 provides a set of baseline results for all three routing algorithms, for both DRES and NDRES. The chart shows that the routing algorithms have a very significant impact on performance, with PP performing substantially better than LCC and BDP for both DRES and NDRES. At a rejection fraction of .001, DRES carries about 20% more traffic than

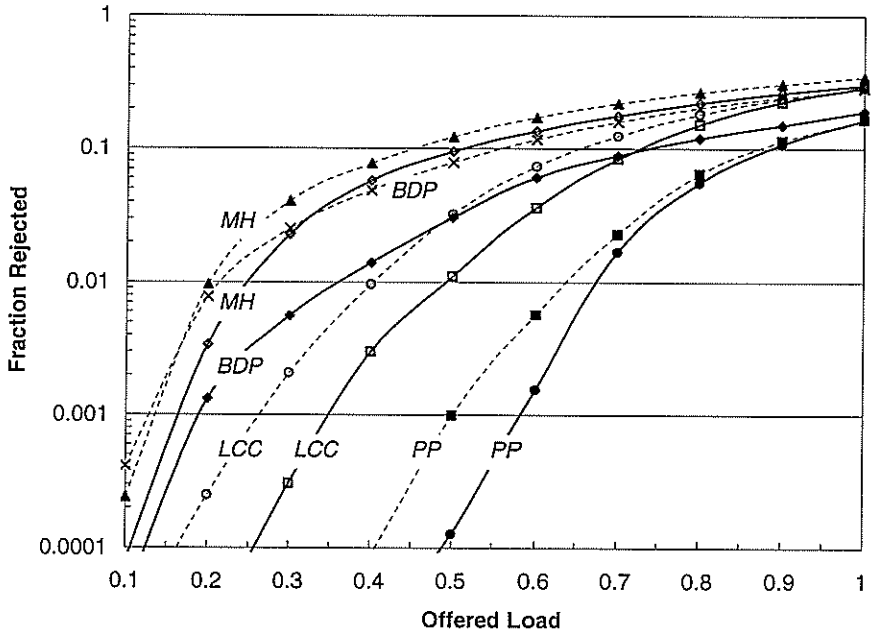

Figure 10: Rejection Fraction (Torus)

NDRES, assuming both use PP.

We define the *threshold* for a given routing algorithm to be the offered load at which the rejection fraction equals is .001. Figure 11 shows how the threshold grows as the defer bound is increased from zero, for LCC and PP. The variation for BDP is the most significant (133% increase when defer bound equals MHT). However, BDP operates at a lower threshold than both PP and LCC.

We define the *DRES gain* for a given routing algorithm as the ratio of the threshold in the DRES case to the threshold in the NDRES case. Figure 12 shows how the DRES gain increases with link fraction for LCC and PP. Both PP and LCC show a significant gain for DRES over NDRES at larger link fractions (35% at a link fraction of 0.2). BDP provides the maximum gains of nearly 80% at the same link fraction, though it operates at a lower threshold as before. We see that deferring plays an important role especially when the requested bandwidth is a significant portion of the link capacity.

Figure 13 shows how the threshold for LCC and BDP drops as the update period increases. The interesting point is that while LCC drops significantly from a *threshold* of 0.7 to 0.2, BDP remains relatively constant. Hop-by-hop routing avoids the use of stale link information and allows BDP to be

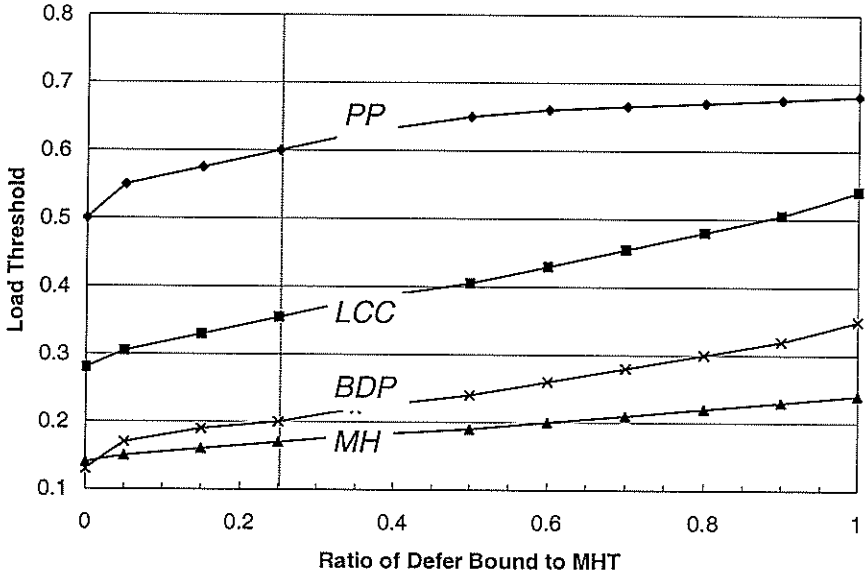

Figure 11: Variation of Threshold with Defer Time(Torus)

relatively immune to the update period. At large update periods ($2 \times MHT$), BDP outperforms LCC. At the default update period of 30 time units, for a given routing scheme, DRES carries approximately 40% more traffic.

5.2 Results for National Network

This section reports results for a national network configuration which has nodes for each of the 20 largest metropolitan areas in the United States. The network topology is shown in Figure 14.

The link capacities were chosen using a constraint-based network design methodology developed in [16, 17, 18]. The method selects link capacities that are sufficient to carry any traffic configuration that satisfies certain traffic constraints. Here, the traffic constraints used to dimension the links includes constraints on the total traffic originating ($\alpha(u)$) and terminating ($\omega(u)$ at each node u. The traffic constraints also include limits on the traffic between any pair of nodes $\mu(u,v)$as given in Equation 6.

$$\mu(u,v) = c \cdot min\{f(u,v)\alpha(u), g(u,v)\omega(v)\} \tag{6}$$

where

$$f(u,v) = \frac{\omega(v)}{\sum_{w \neq u} \omega(w)} \quad g(u,v) = \frac{\alpha(u)}{\sum_{w \neq v} \alpha(w)} \tag{7}$$

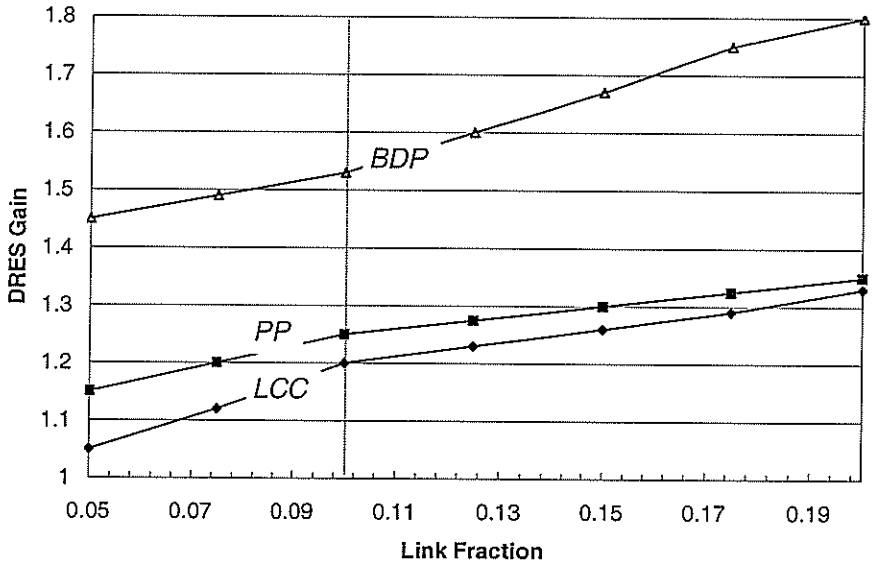

Figure 12: Variation of DRES Gain with Link Fraction (Torus)

The parameter c is a constant called the relaxation factor, and was set to 1. For all u, we let $\alpha(u) = \omega(u)$ with the values chosen to be proportional to the population of the metropolitan areas.

The constraint-based design method takes a topology and the set of traffic constraints and uses linear programming to find a set of link capacities sufficient to handle all traffic conditions satisfying the traffic constraints, assuming the traffic is routed using shortest paths. This process results in a wide range of link capacities, with the largest capacity link being 32 times as large as the smallest.

In the simulations, reservation requests arrive at each node, at rates that are proportional to the population of the area served by the node. The interarrival times and the reservation holding times are exponentially distributed. Uniform reservation bandwidths are used. The destination of each reservation is chosen randomly, but with the choice weighted by the relative population size of the possible destinations. As with the torus, there are several numerical parameters that are varied in the simulations. Each parameter has a *default value*. Whenever one parameter is varied in a given chart, the other parameters are assigned their default values. The default value of the link fraction is .05. In these charts, the link fraction is the ratio of the reservation bandwidth to the bandwidth of the smallest capacity link in the network. The default value of the defer bound is $1/4^{th}$ the mean holding

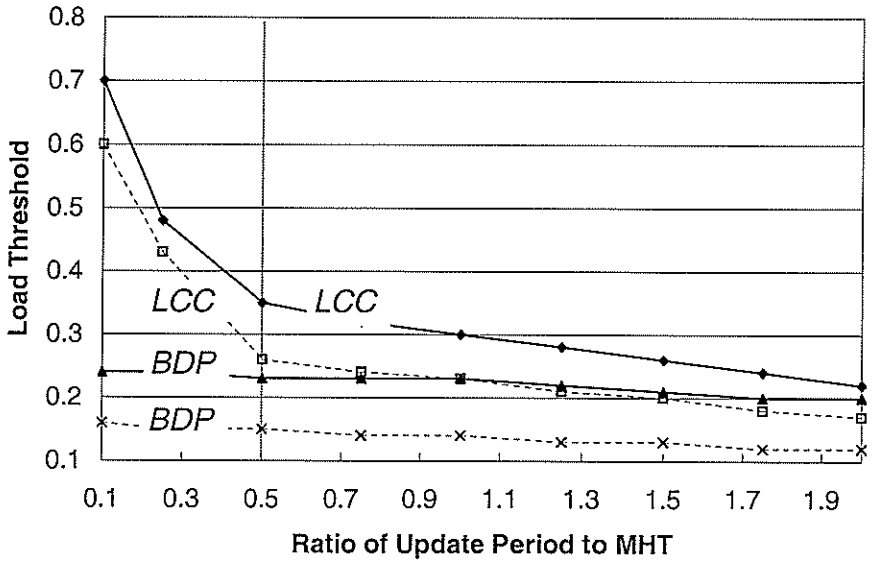

Figure 13: Variation of Threshold with Update Period (Torus)

time (MHT). The default update period (time between state updates) is $MHT/2$ for both LCC and BDP. The default bound on the number of tuples propagated by BDP for each destination is 6. The default number of alternate paths on which PP sends probes is 6. As before, *solid* and *dashed* lines represent the DRES and NDRES based schemes respectively. We refer to this topology as the *ISP* topology in the results.

Figure 15 provides a set of baseline results for all three routing algorithms, for both DRES and NDRES. The chart also includes results for a simple non-QoS routing algorithm, which always uses the path with the smallest number of hops denoted as *MH*. The chart shows that in the national network, as in the torus, the routing algorithms have a very significant impact on performance. We see that BDP performs considerably better than on the torus and that LCC out-performs PP for NDRES. It appears that the fixed set of paths available to PP limits routing flexibility, causing a higher rejection fraction in the NDRES case. This does not affect DRES as much, since the ability of the reservation to wait, makes it less sensitive to the limited routing flexibility.

Figure 16 shows how the threshold grows as the defer bound is increased from zero, for LCC, PP and BDP. The cross-over between LCC and PP reflects the fact that LCC performs better than PP for NDRES. As before, the defer bound has a significant impact on the BDP algorithm. The threshold

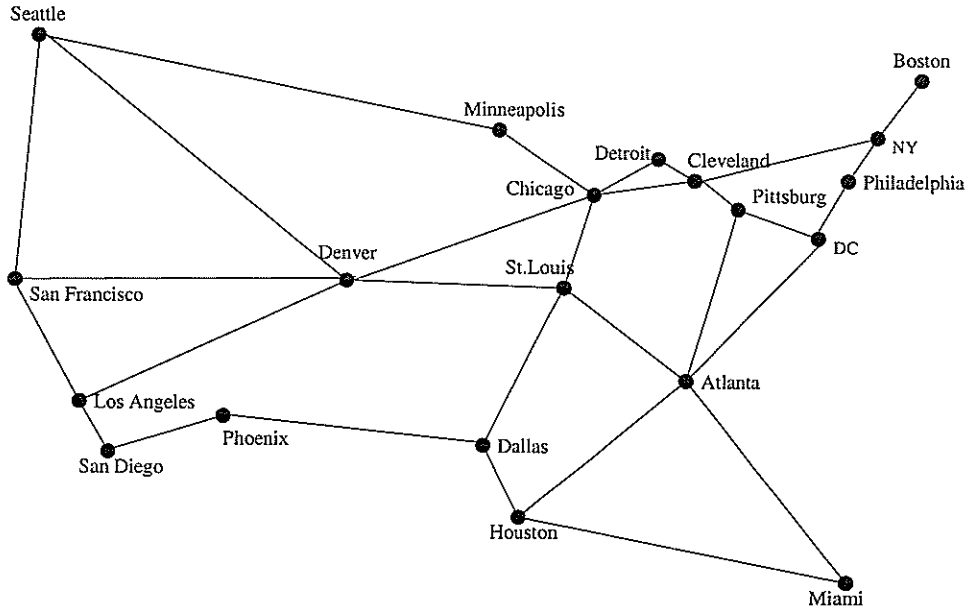

Figure 14: National Network Topology

almost doubles for BDP as compared to a 40% gain for the PP and LCC algorithms. This suggests that larger defer bounds have a greater impact for hop-by-hop protocols than for source routing protocols. Consider a path picked by the PP algorithm compared to the next hop picked by BDP. If the path cannot accommodate the reservation, a larger bound means that resources are held at multiple hops until the defer bound expires. However, BDP routes around the blocked link and deferring facilitates this process.

Figure 17 shows how the DRES gain increases with link fraction for BDP, LCC and PP. We see that all three schemes show greater gain at higher link fractions similar to the results for the torus, with LCC showing the maximum improvement of 20%. Again, larger link fractions translate to larger performance gains for DRES (30-50% gains at a link fraction of 0.2 for the PP algorithm compared to 10-35% gains at the default link fraction of 0.05).

Figure 18 shows how the threshold for BDP and LCC drops as the update period increases. The results show that for the national network, BDP is relatively insensitive to the update period. LCC however falls by about 35% at an update period that is twice the call duration. However, the drop is not as significant as the result for the torus (See Figure 13) since the national network has been engineered to support traffic routed on the shortest path.

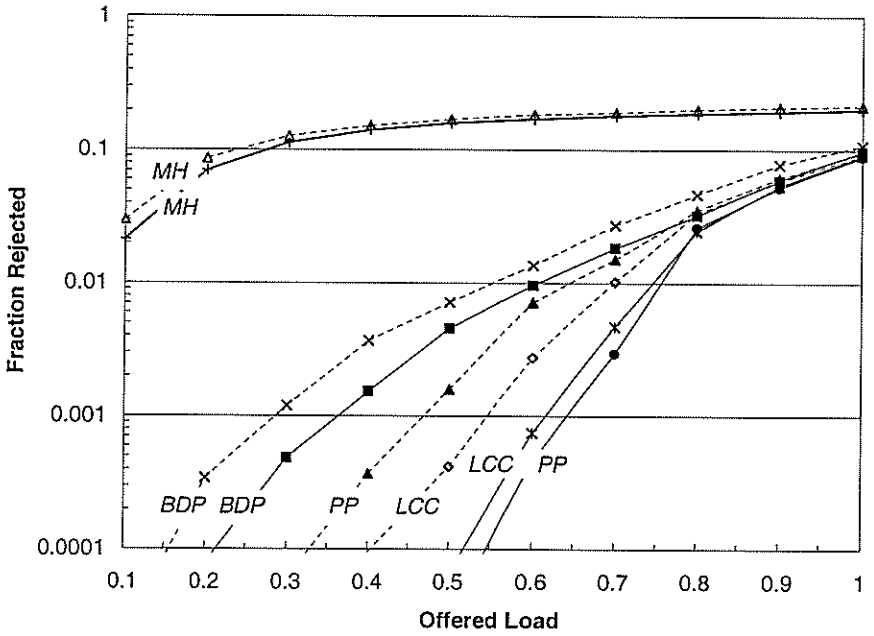

Figure 15: Rejection Fraction (ISP)

Figure 19 shows how the parameter α in the LCC algorithm affects the threshold. The poor performance at small values of α shows clearly that the cost adjustment done by LCC when the margin is small has a significant impact on performance. We note that the performance is relatively insensitive to the value of α beyond .5 and that the default value of α is close to the optimal choice.

5.3 Summary

The results presented above suggest that while DRES does provide significant benefits over a diverse set of topologies, the choice of routing protocol is crucial. For perspective, the 20 node ISP topology costs tens of billion dollars based on our cost assumptions. 20-40% improvement (as shown by DRES) can translate to significant savings. In addition, DRES shows greater gains at large link fractions. This suggests its applicability in access links at backbone networks which are bottleneck link candidates whereby reservations when aggregated together can occupy a substantial link fraction. The PP algorithm emerges as a strong candidate for QoS routing independent of DRES. Its relatively negligible reliance on link state updates highlights its

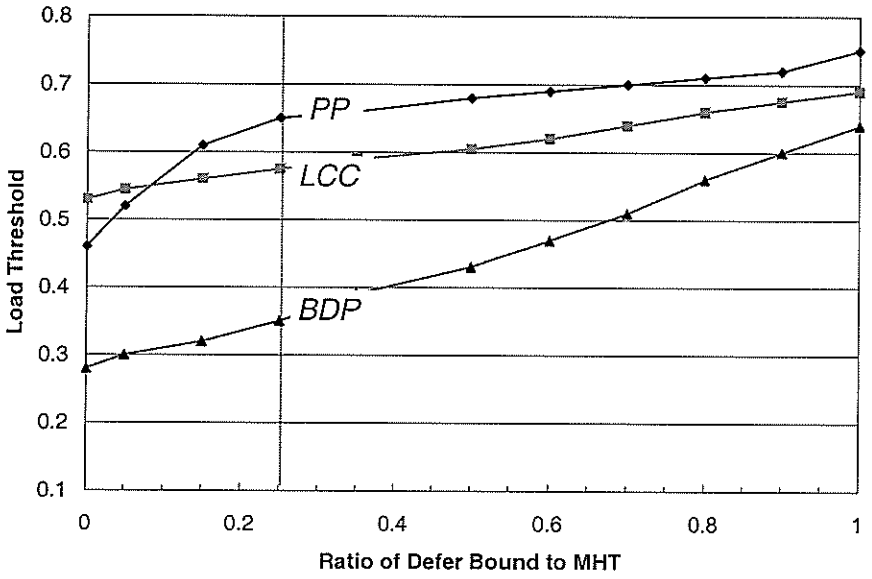

Figure 16: Variation of Threshold with Defer Time (ISP)

robustness under fluctuating network conditions. The LCC algorithm also performs better than current shortest path approaches and can be easily deployed. The BDP algorithm when coupled with a reasonable defer bound provides a practical alternative since it is relatively immune to link state updates compared to LCC.

6 Related Work

There have been numerous proposals to perform resource reservation in data 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.

Recent work on advance reservations [3, 4, 5, 6] propose mechanisms

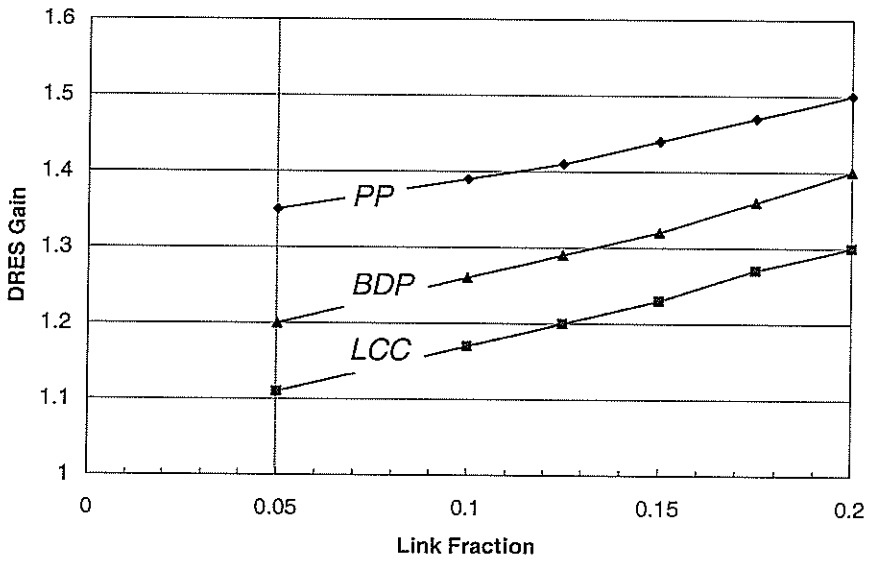

Figure 17: Variation of DRES Gain with Link Fraction (ISP)

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. [5, 6] 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 can be dropped so as to accommodate reservations in the future leading to no guarantees for *immediate* reservations. [14] describes preliminary mechanisms for evaluating the cost of integrating QoS routing with advance reservations.

Prior work in route selection algorithms are typically of two categories *flooding* and *preferred-neighbour*. The former approach involves flooding all routers to find a route. Examples of this approach include [8] which is a distributed route selection mechanism for real-time messages. This scheme sends upto $2m$ messages where m is the number of network links. Not only is this a huge overhead, but each message reserves resources resulting in a potentially high blocking probability. [7] proposes *selective probing* of links as part of a distributed routing mechanism which does hop-by-hop forwarding.

One approach that is a hybrid variation of *flooding* and *preferred-neighbour* approaches is the scheme proposed in [15] for real-time traffic. In this scheme, probes are sent on k alternate paths that simultaneously use k met-

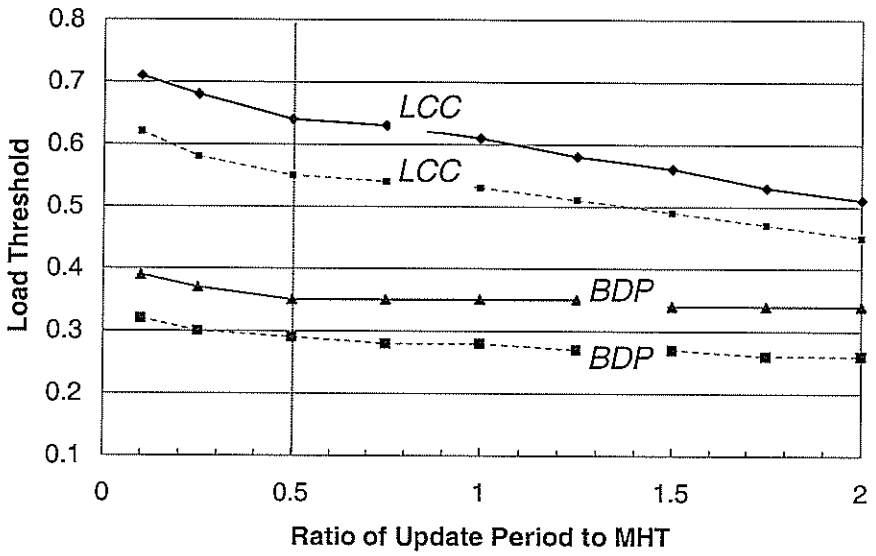

Figure 18: Variation of Threshold with Update Period (ISP)

rics such as hop count to find the best path. This protocol has substantial complexity as it attempts to evaluate k alternate metrics between ID's, raising concerns about scalability. Also, the k probes each reserve resources on the path effectively blocking other requests on each of those paths. Furthermore, in a network where there is significant sharing of links between the alternate paths, resources may be reserved at common links for each of the k metrics creating bottlenecks that increase the blocking probability further.

7 Conclusions

In this paper, we have studied the performance of deferred reservations in data networks. The results show that the use of deferred reservations can lead to significant improvements in network performance. We note that in large networks, even a 10% improvement in performance can yield cost savings worth billions of dollars.

The results also demonstrate the substantial impact that QoS routing algorithms can have on both DRES and NDRES. The three algorithms studied represent a diverse range of different approaches. The parallel probe algorithm appears to be the most effective of the three. While its implementation does require support from the underlying hardware, for updating probes, it requires no background state update, making it an interesting alternative to

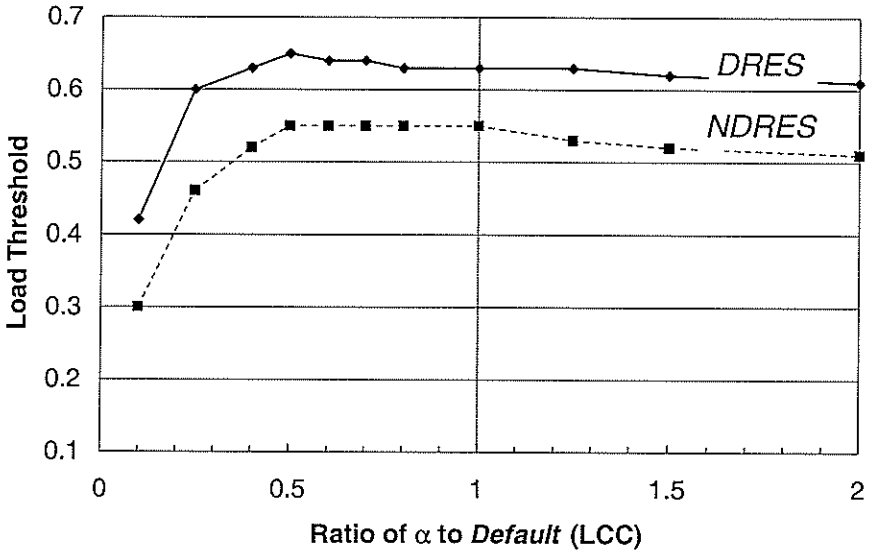

Figure 19: Variation of Threshold with LCC Parameter α (ISP)

more conventional protocols.

References

[1] Pan P., and Schulzrinne H. "YESSIR: A simple reservation mechanism for the internet", *NOSSDAV'98*, 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", *Proc. of IEEE INFOCOM'97*, March 1997.

[4] Wolf C. L., *et. al.* "Issues of reserving resources in advance", *Proc. 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",*Proc. of IWQoS'97*, May 1997.

[7] Chen S., and Nahrstedt K. "Distributed quality-of-service routing for next generation high speed networks based on selective probing", *Proc. IEEE LCN'98*, pp.80-89, 1998.

[8] Shin K. G., and Chou C. "A distributed route-selection scheme for establishing real-time channels", *Proc. High Performance Networking*, pp. 319-330, 1995.

[9] Jamin S., *et. al.* "A measurement based admission control algorithm for integrated services packet networks", *IEEE/ACM Transactions on Networking*, December 1996.

[10] Kleinrock L. "Queuing systems, Volume 1: Theory", *John Wiley & Sons*, 1975. .

[11] ATM Forum. "ATM Private Network-Network Interface Signaling Specification v1.0", 1996.

[12] Labovitz C., et. al. "Internet routing instability", *Proc. of SIGCOMM'97*, August 1997.

[13] Norden S. "DRES: Internet resource management using deferred reservations", *Tech Report WU-CS-01-06, Dept. of Computer Science, Washington University*, http://www.arl.wustl.edu/~samphel/wu-cs-01-06.ps

[14] Guerin R., and Orda A. "Networks with advance reservations: the routing perspective", *Proc. of INFOCOM'2000*, March 2000.

[15] Manimaran G., et. al. "A new distributed route selection approach for channel establishment in real-time networks", *IEEE/ACM Transactions on Networking*, vol. 7, no. 5., October 1999.

[16] Fingerhut J. A. "Approximation algorithms for configuring nonblocking communication networks, *Doctoral Dissertation, Dept. of Computer Science, Washington University*, May 1994.

[17] Fingerhut J. A., Suri S., and Turner J. S. "Designing Least-Cost Nonblocking Broadband Networks," *Journal of Algorithms* 1997, pp. 287-309.

[18] Ma H., Singh I., and Turner J.S. "Constraint based design of ATM networks, an experimental study", *Tech. Report, WUCS-97-15*, Dept. of Computer Science, Washington University, 1997.

[19] Norden S. and Turner J. S. "DRES: Network resource management using deferred reservations", *Proc. of GLOBECOM'01*, November 2001.