Asymptotic Resource Consumption in Multicast Reservation Styles

Danny J. Mitzel
mitzel@catarina.usc.edu
Computer Science Department
University of Southern California
Los Angeles, CA 90089
and
Hughes Aircraft Company
P.O. Box 92919
Los Angeles, CA 90009

Scott Shenker
shenker@parc.xerox.com
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Abstract

The goal of network design is to meet the needs of resident applications in an efficient manner. Adding real-time service and point-to-multipoint multicast routing to the Internet's traditional point-to-point best effort service model will greatly increase the Internet's efficiency in handling point-to-multipoint real-time applications. Recently, the RSVP resource reservation protocol has introduced the concept of "reservation styles", which control how reservations are aggregated in multipoint-to-multipoint real-time applications. In this paper, which is an extension of [9], we analytically evaluate the efficiency gains offered by this new paradigm on three simple network topologies: linear, m-tree, and star. We compare the resource utilization of more traditional reservation approaches to the RSVP reservation styles in the asymptotic limit of large multipoint applications. We find that in several cases the efficiency improvements scale linearly in the number of hosts.

1 Introduction

The goal of network design, in its most concise formulation, is to meet the needs of resident applications[1] in an efficient manner. The key concept here is efficiency. Most application needs can be met merely by providing sufficient bandwidth. However, bandwidth is not free and so networks should be designed to meet application performance goals while using bandwidth efficiently.

The Internet, and other similar packet-switched network architectures, offer point-to-point best-effort service. The network takes packets from a single source and delivers them to a single destination. Each packet is given best-effort service, which means that no guarantees are made as to when and whether it will be delivered. Operationally, best-effort service is typically implemented using FIFO service at network switches with no admission control or resource reservation; that is, sources need not notify the network before transmitting data, and no resources are set aside for any particular flow.

This architecture, with its point-to-point best-effort service model, has supported a wide variety of data applications in an extremely efficient manner. Applications like remote login (e.g., Telnet), file transfer (e.g., FTP), and electronic mail are handled much more efficiently in this architecture than they are in, say, X.25 or in the telephone network.

However, there are application requirements that this service model does not handle efficiently. For instance, applications like interactive video and voice often have stringent real-time requirements on the delivery of packets. To be useful to these application, packets must arrive within some delay bound (see [12] for a fuller discussion of these requirements). There has been significant recent work [2, 5, 6, 7, 8, 10, 14] showing that by carefully scheduling packets, and utilizing admission control to prevent network overload, one can achieve such real-time delay bounds in a packet-switched environment. It is widely believed that providing such delay bounds only to those flows which need them, by using resource reservation and nontrivial scheduling algorithms, is much more efficient than continuing to use FIFO scheduling and merely adding enough bandwidth to provide low delay service to all flows.

Notice that reservations change the concept of resource consumption in the network. Without reservations, the usage of resources in these networks is tied directly to sending packets; i.e., if you haven't sent any packets you haven't consumed any resources since you haven't denied or affected anybody else's service. With reservations, admission control will deny access if there are not sufficient unreserved resources available; reservations, even if unused, can therefore prevent other flows from reserving resources. Thus, reservations themselves can be seen as consuming resources somewhat independently[2] from the actual usage of those reser-

[1] By resident applications we mean those applications using the network, not those applications residing in the network.

[2] The degree of independence depends on the nature of real-time service provided. Service which provides worst-case delay bounds must base all admission control decisions on the reservation parameters without regard for the actual usage. Service which provides lower quality assurances, like the predictive service in [2, 12], can base ad-

vations. In this paper, we will focus on the reservation of resources rather than their actual use. Although audio and video applications often do not have a fixed quality of service requirement (i.e., they can operate over a broad range of data rates providing varying degrees of perceived quality), we assume that a reservation mechanism is required to ensure minimal signal quality levels.

Another class of applications whose requirements are not handled efficiently by the Internet's point-to-point best-effort service are those that require the same data to be sent to several receivers. This occurs in teleconferences and remote lectures, where voice and video from one individual go to many other participants. Using the traditional point-to-point service, a separate packet is sent to each receiver (we call this simultaneous unicasts); multiple copies of packets are sent over the overlapping portions of the routes to individual receivers. Multicast routing [4, 13] solves this problem by having the network, whenever the routes diverge, send a single copy along each path. As we quantify later, this leads to tremendous efficiency gains. In fact, multicast (as embodied in the Mbone [1]) has been crucial in enabling the widespread distribution of video and voice in broadcasting Internet Engineering Task Force meetings. Broadcasting these meetings, which at times have several hundred listeners, would simply have been impossible without multicast given the current limited bandwidth on many Internet links.

Adding multicast and real-time service will transform the Internet from a point-to-point best effort network into one that also offers point-to-multipoint real-time service. This will greatly expand the range of applications whose needs can be efficiently met by the Internet. We should note, however, that there are some real-time applications that are best described as multipoint-to-multipoint; an example of this is an n-way teleconference where every participant needs to see and hear every other participant. Such applications can be dealt with as a set of independent point-to-multipoint applications; when the paths from two different sources to the same receiver overlap, resources are reserved for both sources independently. This approach is typically sufficient when the traffic from these sources, and the receiver's desire to see that traffic[3], is relatively independent. However, as first recognized in [15], when the sources are not independent this approach leads to inefficiencies. The RSVP resource reservation protocol [15] introduced the concept of "reservation styles". Reservation styles are different ways of aggregating the resource requirements for each source on a single link. Table 1 summarizes the reservation styles investigated in this study. The detailed definition of these reservation styles, and the mechanisms that implement them, have been described elsewhere [15, 16].

Our purpose in this paper is to evaluate the extent to which these reservation styles increase efficiency; in particular, we focus on asymptotic resource usage as n (the number of participants in the multipoint application) gets large. An earlier work [9] discussed these issues in a more informal manner; the present work is an attempt to quantify these savings in a more systematic and rigorous fashion. We find that the RSVP reservation styles achieve very significant savings for large n. Thus, if multipoint-to-multipoint applications represent a sizable portion of the future network load then it will be important to include these reservation styles in the basic Internet service model.

We look at two cases where these reservation styles are useful. The first case is "self-limiting" traffic, where the very nature of the application leads to correlations in the traffic patterns from the different sources. An example of this is an audio conference, when the social prohibition of simultaneously speaking means that rarely will more than one or perhaps a few speakers be active at any one time. The second case is "channel selection" where receivers want the option of receiving data from any of the available sources, but will never want to receive data from more than one (or a few) at a time. Each of these cases violate the "independence" assumption, and thus the reservations for traffic from separate senders should not necessarily be treated independently.

We investigate the asymptotic resource savings achieved by these RSVP reservation styles in three simple network topologies (see Figure 1): linear, m-tree, and star. These topologies are not meant, of course, to be realistic. Rather, by restricting ourselves to simple and tractable topologies, our intent is to provide a more rigorous and systematic analysis of resource consumption in large multipoint-to-multipoint applications. Most of our results are analytical, although we do rely on simulations for some quantities which have so far defied direct calculation.

This paper has 5 sections. We first, in Section 2, define the basic resource consumption model and describe the three simple topologies. We then, in Sections 3 and 4, describe the two reservation styles and analyze, through both calculations and simulations, their asymptotic resource consumption. We conclude with a summary of our results in Section 5.

2 Network Model

We consider multipoint-to-multipoint applications running on a set of n network hosts. Each host is both a sender of data and a receiver of data. Each host generates an equivalent traffic stream, which consumes (or at least requires the reservation of) some given amount of bandwidth. The quantity of interest is the total reserved bandwidth needed to support a given size application. We will set the amount of bandwidth reserved to be the unit of bandwidth, so that every independent reservation consumes one unit of bandwidth[4]. All reservations are unidirectional in nature.

We consider the three network topologies depicted in Figure 1: linear, m-tree, and star. While none of these topologies are particularly good models for a real network, they do represent a wide spectrum of possibilities. Many of our results are relatively independent of topology, which suggests that perhaps our results are relevant to more general networks. Each link is bi-directional, with separate reservations for bandwidth in each direction. We consider the capacity of each link to be unlimited.

Each source sends its data to all other hosts. Routing is done via multicast. Since these are acyclic topologies, there is no ambiguity in the routes. There is a multicast distribution tree from each source to all other hosts. Similarly, there is a reverse tree going from each receiver to all other hosts; this describes the paths taken by data arriving at that host. In our topologies, the distribution tree and the reverse tree are always identical. In fact, for all hosts they both are the

mission control decisions at least in part on actual usage.

[3] By this we mean that my desire to see source A is independent of my desire to see source B.

[4] Note that we are using a rather primitive model of reservations, using only bandwidth to describe the reservation. In practice, the flow specification [11, 13] will likely be somewhat more complex.

Reservation Style	**Description**
Independent Tree	A separate and independent reservation is allocated for each source distribution tree. Per-link reservation is based on the number of upstream senders (N_{up_src}).
Shared Tree	A shared reservation is allocated on each link in the distribution mesh for use by any source. Per-link reservation is based on the number of upstream senders, limited by the number of simultaneous sources that will transmit at any one time ($MIN[N_{up\_src},\ N_{sim\_src}]$).
Chosen Source	A separate and independent reservation is allocated along the distribution tree from each source to only the set of receivers that are currently tuned into that source. Per-link reservation is based on the number of upstream senders that have been selected by at least one downstream receiver ($N_{up\_sel\_src}$).
Dynamic Filter	A set of shared resources is allocated on each link to accommodate the maximal downstream resource demand. Each reservation has a receiver-controlled "filter" allowing dynamic selection among sources. Per-link reservation is based on the number of upstream senders, limited by the number of independent reservations required to allow all downstream receivers to make independent source selections ($MIN[N_{up\_src},\ (N_{down\_rcvr} * N_{sim\_chan})]$).

Table 1: Summary of Reservation Styles

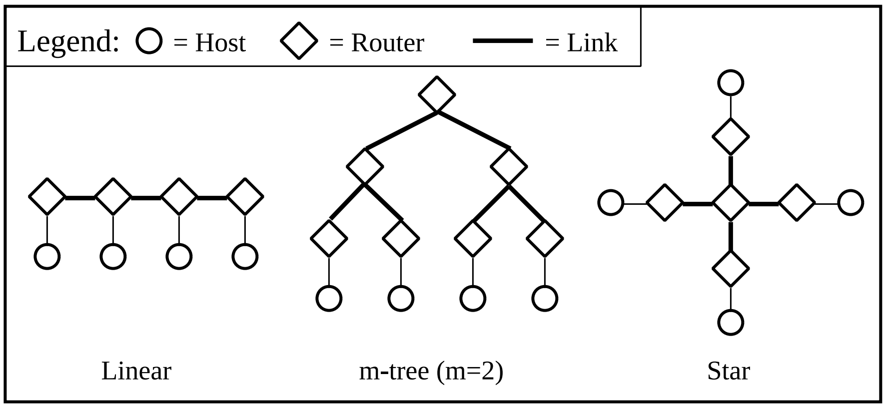

Figure 1: Network Topologies

entire network in all of our topologies (although links may be traversed in different directions in different trees, each link is traversed exactly once in each tree). A distribution mesh is the union of the distribution trees. For our networks, the distribution mesh is always the entire network with every link traversed in both directions.

For each network topology, we consider a network with n end hosts and let the network grow as the number of hosts does. There are several quantities that will be relevant to our later analysis. For a given size network n, we can consider:

Total Links L The total number of links in the topology.

Diameter D The maximum host-host distance, in numbers of hops.

Average Path A The average host-host distance, in numbers of hops. This does not count a host connecting to itself.

Let us now briefly discuss the three topologies. The linear topology has $n-1$ links, with each connecting two hosts. Clearly $D = L = n-1$. A slightly more nontrivial calculation[5] reveals that $A = \frac{n+1}{3}$.

In the m-tree topology, the hosts are at the leaves of the tree and the tree has a constant branching ratio of m. Here, $n = m^d$ where d is the depth of the tree. The longest path is one that traverses to and from the root of the tree, so $D = 2d = 2\log_m n$. We also have $L = \frac{m}{m-1}(n-1)$, and $A = \frac{2[(m-1)n\log_m n - n + 1]}{(n-1)(m-1)}$.

The star configuration has a central hub, and there is a link connecting each host to this hub. Here, $D = A = 2$, and $L = n$. Notice that the star topology is merely the limiting case of the m-tree topology with $d = 1$ and $m = n$.

These results are summarized in Table 2. For later comparisons, we now compute the resource usages of multicast and simultaneous unicasts. The quantity of interest is the total number link traversals, which counts each time

[5] Here, as elsewhere in the paper, we will spare the reader the derivation of these essentially combinatoric formulae. Also, all of these formulae are only valid for $n > 1$ and for values of n that represent a complete topology. This is not an issue for the linear or star topologies, but is relevant for the m-tree, where $n = m^d$ are the only valid values for n.

a packet traverses a link as a separate use of the link. To do this computation (and later computations), we define two quantities for a given direction along a given link[6]:

N_{up_src} is the number of upstream sources that include the link in their multicast distribution tree.

N_{down_rcvr} is the number of downstream hosts that receive data along this link.

For the topologies we consider, these two numbers must always sum to n, $N_{up\_src} + N_{down\_rcvr} = n$, since every link is on every distribution tree. Furthermore, considering the reverse direction of the link merely reverses these two numbers.

When using simultaneous unicasts, the total number of packets traversing a particular link is given by $N_{up\_src} * N_{down\_rcvr}$. When using multicasts, the total number of packets traversing a particular link is given by N_{up_src}, because duplication for different receivers is eliminated.

Sending a packet from each source to each destination without using multicast involves $n(n-1)A$ link traversals; the data from a single source travels over $n-1$ paths, of average length A, and there are n such sources. Using multicast involves merely nL link traversals, since no link is traversed more than once by any packet, and all links are traversed in exactly one direction in each multicast tree. Thus, the ratio of $(n-1)A$ to L is an estimate of resource savings due to multicast. For the linear network, these savings are $O(n)$, for m-trees the savings are $O(\log_m n)$, and for a star the savings are $O(1)$.

We should note that these savings are calculated for link traversals of data. In the rest of the paper we are interested in savings in terms of reserved resources. Reservation styles do not affect the actual number of link traversals, only the resources reserved.

3 Self-Limiting Applications

We now consider self-limiting applications. These are multipoint-to-multipoint applications which have application-level constraints that inhibit data sources from transmitting simultaneously. As we mentioned before, an audioconference is one example of this, where the social inhibitions tend to discourage simultaneous speaking.[7] Another rather different example is satellite tracking. Here there are a number of large antennae, and when the satellite is within their range the data is downloaded and then sent to the other sites. If the ranges of the antennae do not overlap so the satellite is only within range of a single antenna at any one time, then the traffic is self-limiting because two sources are never active simultaneously. More generally, we can describe a self-limiting application by the maximal number of sources that will transmit at any one time; we will denote this quantity by N_{sim_src}.

The traditional approach is to make separate and independent reservations for each distribution tree. We will call this the *Independent Tree* approach. On every link, the number of units of bandwidth reserved is given by N_{up_src}. The total bandwidth reserved over the whole network is given by $nL = n(n-1)A$, since there is an independently reserved path from every sender to its $n-1$ receivers, and there are n senders.

However, as was noted in [15], the application requirements are entirely met if, on every link, there are merely sufficient reserved resources to accommodate the maximum number of upstream sources that will simultaneously transmit. Thus, on each individual link we need only reserve $MIN[N_{up\_src}, N_{sim\_src}]$, which is in contrast to the N_{up_src} units reserved by the Independent Tree approach. RSVP has defined a reservation style which we call[8] *Shared*, in which only this necessary reservation of $MIN[N_{up\_src}, N_{sim\_src}]$ is made; these resources are shared between the upstream sources in the sense that traffic from any of them can use this reserved bandwidth.

For simplicity, we focus on the case where $N_{sim\_src} = 1$, so in the Shared reservation style each link has either 0 or 1 units of bandwidth reserved. The difference in reservation styles can then be concisely captured by noting that the Independent reservations are based upon the *sum* of all links in all distribution trees, while the Shared reservations are based upon the *union* of the links across the distribution mesh. In all of our topologies the mesh consists of all the links traversed in both directions, whereas each distribution tree consists of all links traversed in only one direction. Thus, the ratio of bandwidth consumed between these two approaches is always $\frac{n}{2}$ in our three topologies. We have summarized these results in Table 3.

In fact, whenever the distribution mesh is acyclic the ratio of Independent to Shared resource usage is exactly $\frac{n}{2}$. Consider the distribution tree from source A, and assume that it does not touch some link (in either direction) in the distribution mesh. Since this link is in the distribution mesh, it must lie on the path between two sources; assume that this link is on the path from B to C. Then, the path from A to B to C to A contains a cycle, since it can only traverse the missing link once (going from B to C). Thus, if the distribution mesh is acyclic then every distribution tree touches every link once and only once. It then directly follows that the distribution mesh touches every link in both directions. Therefore, whenever the distribution mesh is acyclic the ratio of resource usage is exactly $\frac{n}{2}$. Note that in cyclic networks this result need not hold. For instance, in a fully connected network, the Independent and the Shared resource demands are exactly the same.

For purposes of comparison, it is interesting to note that multicast's advantage over simultaneous unicasts ranged from $O(n)$ in linear networks, to $O(\log_m n)$ in m-tree networks, to $O(1)$ in star networks. In contrast, the shared reservation style has an advantage of $\frac{n}{2}$ in all networks with acyclic distribution meshes. Observe also that the results for the Shared and Independent reservation styles are consistent with the intuition that the resource requirements of Independent scale as $O(nL)$ whereas those of Shared scale as $O(L)$.

4 Channel Selection

4.1 Definition of Reservation Styles

The other class of applications we consider are "channel selection" applications [3]. These are applications in which

[6] Even though links are bidirectional, when referring to the reservations along a link we typically are referring to a single direction.

[7] Note that a vidoconference is not self-limiting, since video is independent of what other participants are doing.

[8] The terminology of the reservation styles in RSVP is somewhat in flux, so here we adopt a somewhat independent terminology to avoid direct inconsistencies. The shared reservation style is currently called wildcard-filter in [16].

Topology	L	D	A
Linear	$n-1$	$n-1$	$\frac{n+1}{3}$
m-Tree	$\frac{m}{m-1}(n-1)$	$2\log_m n$	$\frac{2[(m-1)n\log_m n-n+1]}{(n-1)(m-1)}$
Star	n	2	2

Table 2: Topological Properties

Topology	Number of Reservations		Ratio
	Independent	Shared	
Linear	$n(n-1)$	$2(n-1)$	$\frac{n}{2}$
Tree	$\frac{nm(n-1)}{m-1}$	$\frac{2m(n-1)}{m-1}$	$\frac{n}{2}$
Star	n^2	$2n$	$\frac{n}{2}$

Table 3: Resource Allocation for Self-Limiting Applications with $N_{sim\_src} = 1$.

the traffic from each sender is independent, but the receiver only wishes to receive data from a limited number of senders at any one time. The eponymous example is that of television, where one wants access to many channels but only wants to receive one at a time. Similarly, large multiparty videoconferences are sometimes an example of this, in that a receiver may be unable to accommodate data streams from all active participants simultaneously, but desires the ability to dynamically select a subset of the sources to receive at any time. This restriction on the number of simultaneous sources may be due to bandwidth limitations, display or codec hardware, or the inability of the user to assimilate information from all sources concurrently. In general, we can characterize a channel selection application by the maximum number of channels N_{sim_chan} it wishes to receive at any one time.

From the user's perspective there are two alternative service models: assured channel selection and non-assured channel selection. In assured channel selection, the user is guaranteed that the resources will be available to view the selected channel. Thus, assured channel selection involves pre-reserving resources for all of the channels. Assured channel selection is the service that is appropriate for the examples cited above. In non-assured channel selection, no such guarantee is made, and the request can be denied by admission control. We consider this case only because it provides a convenient lower bound to the resources required for assured channel selection. There is a tradeoff between the extra assurance of the assured service model, and its presumably higher cost due to extra resource consumption; one of the goals in this section is to examine quantitatively this extra resource consumption. We now describe the reservations required to support these assured and non-assured services.

The most direct way to support this non-assured service is to make a new reservation every time a new channel is selected (and then to tear down the old reservation). We will call this the *Chosen Source* reservation style, since it only reserves for the currently chosen sources. Resources are reserved along the distribution subtree from each source to the set of receivers that are currently tuned into that source. The trees from different sources are independent. Thus, the reserved amount on a link is given by $N_{up\_sel\_src}$ which is the number of senders upstream that have been selected by at least one downstream receiver. The Chosen Source reservation style, because it only reserves for the currently selected sources, provides a lower bound for the resource consumption required by assured service. We can provide assured service using two different reservation styles.

The traditional way to provide assured channel selection is to reserve independent trees for each source, which is just the Independent reservation style discussed in Section 3. This provides sufficient resources for all sources to simultaneously arrive at the receiver. The receiver can then switch between channels by selecting the desired incoming stream. The channel selecting, or filtering of incoming data, is done entirely at the receiver (much like the signals for all TV channels arrive at the cable set-top box, and the tuner selects one).

RSVP [15] introduced the idea, inspired by a comment from Jon Crowcroft [3], that this selecting or filtering process can occur within the network rather than just at the receiver. Each reservation on a link is accompanied by a *filter* that determines which packets get to use the reserved resources. One of the novel aspects of RSVP is that even while the reservation is fixed this filter can change dynamically in response to signals from the receivers. RSVP offers a *Dynamic Filter* reservation style which reserves enough bandwidth on each link so that the receiver can always select, without failure, any set of N_{sim_chan} sources. RSVP provides a mechanism whereby receivers inform their upstream routers which sources they wish to receive, and the filters are then set to only allow packets from those sources to pass. The resource requirements can be expressed as follows; on every link the amount reserved is given by $MIN[N_{up\_src}, (N_{down\_rcvr} * N_{sim\_chan})]$, recalling that N_{down_rcvr} is the number of downstream hosts that receive data (from any source) along the link (i.e., the number of receivers for which this link is in the reverse tree), and N_{up_src} is the number of upstream sources that include the link in their multicast distribution tree.

This formula merely expresses the insight that one need not reserve more channels than the number of upstream sources, nor more than the maximal number of downstream requests. As one can see directly from the expression for per-link reservations, on every link the resources required for Dynamic Filter reservations is bounded above by the

Independent reservation and below by the Chosen Source reservation.

We now proceed to analyze the asymptotic resource consumptions of these various reservation styles. For simplicity, we choose $N_{sim\_chan} = 1$, so every receiver receives only one channel at a time. With this choice, the reservation in the two directions on a link are identical, since reversing directions merely reverses the meanings of N_{up_src} and N_{down_rcvr}.

4.2 Assured Channel Selection Alternatives

We now compare the two assured channel switching reservation styles. The Independent reservation case was already considered in Section 3; recall that the total resource consumption is given by nL. The Dynamic Filter reservation style requires a reservation of $MIN[N_{up\_src}, N_{down\_rcvr}]$ on every link; the presence of the MIN function makes this style somewhat harder to characterize and compute.

In the linear case, the reservation needed on a link in the i'th position is $MIN[i, n-i]$. For n odd, this sums to $\frac{n^2-1}{2}$, and for n even it sums to $\frac{n^2}{2}$ (in Table 4 we only show the result for n even).

For the m-tree topology, the expression $MIN[N_{up\_src}, N_{down\_rcvr}]$ reduces to the number of hosts below the link on the tree (assuming the root of the tree is "up"). There are m^i links at depth i in the tree, and there are m^{d-i} nodes below the links at depth i. Thus, the resource consumption at every level is just $2m^d$ (the factor of 2 is because each link has two directions). Since there are d levels, the total resource consumption is just $2dm^d$. In terms of n, this becomes $2n \log_m n$.

The star topology result can be calculated by merely setting $m = n$ in the m-tree result, yielding a resource consumption of $2n$.

These results are summarized in Table 4. They are consistent with the intuition that the worst case of Chosen Source, and hence Dynamic Filter, scales as $O(nD)$, in contrast to Independent scaling as $O(nL)$.[9]

4.3 Dynamic Filter vs. Non-assured Selection Overhead

As mentioned earlier, our interest in the Chosen Source reservation style is primarily because it represents the minimal resources needed to support the currently selected sources. Thus, we can use this reservation style to quantify the overhead incurred in providing the extra assurance in the assured selection service (as opposed to the non-assured service provided by the Chosen Source style). However, the total resource requirements for Chosen Source depend not only on network topology and participant distribution[10] but also on the set of sources selected by each receiver. When describing the resource consumption of Chosen Source, we therefore need to characterize the set of source selections. Consequently, we define three classes of Chosen Source behavior; *worst case* (CS_{worst}) occurs when all receivers correlate their source selections to maximize the total resource consumption; *average case* (CS_{avg}) is the average result when each receiver performs an independent and random source selection; and *best case* (CS_{best}) is achieved when receivers correlate their source selections to minimize the total resource consumption. For each of these Chosen Source behaviors we compute, either through analysis or simulation, the total resource consumption and compare these asymptotic resource requirements with those of the Dynamic Filter reservation style.

4.3.1 Chosen Source Worst Case (CS_{worst})

The worst case for Chosen Source results when each receiver selects a distinct source (resulting in no overlap in distribution trees) such that the set of selections maximizes the total point-to-point distance. For the linear topology, CS_{worst} is obtained when each receiver selects the host $\frac{n}{2}$ distance away (assuming for convenience that n is even). The total resource requirements for this case can be easily calculated to be $\frac{n^2}{2}$. For m-tree, CS_{worst} is obtained when each receiver selects a host that requires traversal of the root node, this results in $2D$ link reservations per source giving a total requirement of $2nD = 2n \log_m n$. For the star topology, CS_{worst} is obtained whenever each receiver selects a distinct source, resulting in $2n$ reservations.

These results are reproduced in Table 5. Surprisingly, for all the topologies studied the ratio of CS_{worst} to Dynamic Filter is always exactly 1. That is, in these topologies providing assured channel selection requires absolutely no additional resources when compared to the worst case of the non-assured channel selection. We do not yet know how fully general this result is. We do know that it does not hold for the fully connected network (where Dynamic Filter requires $n(n-1)$ reservations and CS_{worst} requires only n).

4.3.2 Chosen Source Average Case (CS_{avg})

We now consider the average case performance of the Chosen Source reservation style when all receivers make an independent and random source selection. We have been unable to solve this case exactly, and so instead we use simulation to compute CS_{avg}.

Our experimental methodology was to simulate each of the three network topologies for various values of n. For each value of n we performed random source selection for each receiver; selecting a Chosen Source from among the $n-1$ other participants with uniform probability. Then we calculated the exact number of link reservations required by the Chosen Source reservation style. We repeated this process multiple times and used the sample mean to predict CS_{avg}. Even though the total number of permutations for source-receiver selection grows as $(n-1)^n$ we found that repeating the random source selection process just 500 times for each n resulted in an estimate of CS_{avg} with less than 1% relative error at a 95% confidence level.

Rather than displaying CS_{avg}'s absolute performance, we show how CS_{worst} compares to CS_{avg}.[11] In Figure 2 we plot the ratio of the simulated CS_{avg} resource requirements against those of CS_{worst} for the linear, m-tree (with $m = 2$ and $m = 4$), and star topologies. Note that the ratio appears to asymptotically approach a nonzero constant for all

[9] While we expect that the worst case of Chosen Source will scale as $O(nD)$ in more general topologies, we doubt that Dynamic Filter will continue to be equal to the worst case of Chosen Source in more general topologies.

[10] These completely determine the resource requirements for the Independent and Dynamic Filter reservation styles.

[11] Note that CS_{worst} is equivalent to Dynamic Filter thus this also represents the ratio in performance of Dynamic Filter assured channel selection to average case Chosen Source non-assured channel selection.

Topology	Number of Reservations		Ratio
	Independent	Dyn Filter	
Linear	$n(n-1)$	$\frac{n^2}{2}$	$\frac{2(n-1)}{n}$
Tree	$\frac{mn(n-1)}{m-1}$	$2n\log_m n$	$\frac{m(n-1)}{2(m-1)\log_m n}$
Star	n^2	$2n$	$\frac{n}{2}$

Table 4: Resource Allocation for Assured Channel Selection with $N_{sim\_chan} = 1$.

Topology	Number of Reservations			Ratio	
	CS_{worst}	CS_{avg}	CS_{best}	CS_{avg}/CS_{worst}	CS_{best}/CS_{worst}
Linear	$\frac{n^2}{2}$	$O(n^2)$ simulation	n	0.53 simulation	$\frac{2}{n}$
Tree	$2n\log_m n$	$O(n\log_m n)$ simulation	$\frac{m(n+1)-2}{m-1}$	0.77 simulation	$\frac{m(n+1)-2}{2n(m-1)\log_m n}$
Star	$2n$	$O(n)$ simulation	$n+2$	0.82 simulation	$\frac{n+2}{2n}$

Table 5: Resource Allocation for Non-Assured Channel Selection with $N_{sim\_chan} = 1$

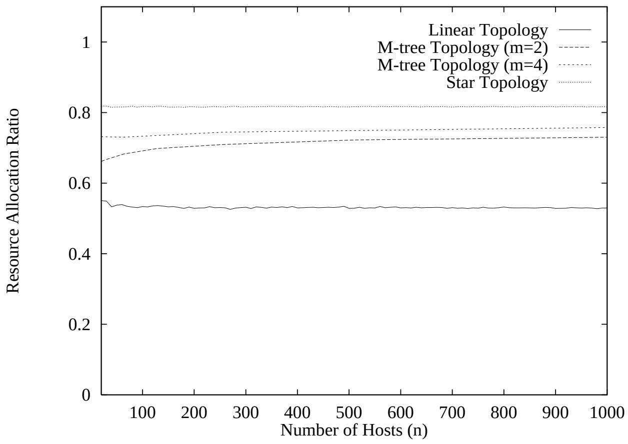

Figure 2: Ratio of Chosen Source Average and Worst Case for Selected Topologies

topologies investigated (the constant depends on the topology, but in each topology the ratio appears to asymptote to a constant). This implies that Dynamic Filter overallocates only a fixed percentage of resources when compared to Chosen Source average case. Thus, not only does the assured service of Dynamic Filter not require any overallocation when compared to the worst case of the non-assured service of Chosen Source, but it also only requires a fixed percentage of overallocation when compared to the average case of the non-assured service of Chosen Source. Again, we do not know how general these results are, but they do hold in each of our three topologies.

4.3.3 Chosen Source Best Case (CS_{best})

Chosen Source best case is when the source selections minimize the total resources required. This occurs when all receivers but one select the same source (a receiver cannot select itself as its source) and the exceptional receiver selects a nearest source. This results in a single multicast distribution tree from the source to all but one of the receivers plus the path from the exceptional receiver to its nearest neighbor. Thus the total resources required are $L + 1$ for the linear topology and $L + 2$ for the m-tree and star topologies.

For each topology, CS_{best} scales as $O(n)$. In contrast, Dynamic Filter scales as $O(n^2)$ in the linear topology and as $O(n \log_m n)$ in the tree topology (and as $O(n)$ in the star topology). Therefore, only when comparing Dynamic Filter to the best case of Chosen Source in the linear and tree topologies do we find an asymptotic scaling advantage for Chosen Source. The extent of this advantage scales as $O(D)$, since Dynamic Filter scales as $O(nD)$ and Chosen Source best case scales as $O(n)$.

5 Summary

In this paper we studied the asymptotic resource consumption of various RSVP reservation styles in three simple topologies. To our knowledge, this is the first analytic comparison of the relative merits of these approaches. For self-limiting applications, the Shared reservation style achieves savings of $\frac{n}{2}$ over the traditional Independent reservation style in any topology with an acyclic distribution mesh. For channel selection applications, the Dynamic Filter reservation style achieves substantial savings over the Independent reservation style in the m-tree and star topologies. More surprisingly, the Dynamic Filter reservation style uses exactly the same resources as the worst case of the Chosen Source reservation style, and appears to be only a constant factor worse than the average case of the Chosen Source reservation style. These results suggest that, at least for large multipoint applications, the RSVP reservation styles of Shared and Dynamic Filter offer substantial savings in resource consumption over the traditional Independent reservation style, and that the assured channel selection service does not incur asymptotically diverging overallocation when compared to the non-assured service.

These results, of course, relied on many simplifications. We hope, in future work, to explore variations on the various models such as considering $N_{sim\_chan} > 1$ and $N_{sim\_src} > 1$, and allowing the number of senders and receivers to be different. More importantly, these results were derived on over-simplified topologies. It is important to explore to what extent they apply to real networks. This question is more subtle than it first appears, since our results describe the large n limit of a network, where both the network and the number of resident hosts are growing. Two questions must be addressed. How can one characterize "real" networks? Certainly randomly generated networks are no more real than the simple topologies considered here. To some extent, real networks are the product of chaotic growth at the edges and planned growth in the interior. Assuming one can characterize more realistic networks, how can one explore the asymptotic limit? Should one hold the density fixed, or the ratio of the diameter to number of hosts, or is there some other criterion? In these simple topologies the answer was clear, but in more general networks this is a completely unexplored issue. These are issues we hope to return to in future work; they are relevant not just to the present study but to any investigation which depends on network topology.

6 Acknowledgments

We wish to thank Deborah Estrin for her helpful comments on this paper, and to gratefully acknowledge both her and Lixia Zhang for collaborating with us on an earlier version of this work.

References

[1] Casner, S., Deering, S., "First IETF Internet Audiocast," *ACM SIGCOMM Computer Communication Review*, vol. 22, no. 3, July 1992.

[2] D. Clark, S. Shenker, and L. Zhang. *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism* In **Proceedings of SIGCOMM '92**, pp 14-26, 1992.

[3] J. Crowcroft, personal communication.

[4] Deering, S., "Multicast Routing in a Datagram Internetwork," Technical Report STAN-CS-92-1415, Stanford University, December 1991.

[5] D. Ferrari and D. Verma. *A Scheme for Real-Time Channel Establishment in Wide-Area Networks,* In **IEEE JSAC**, Vol. 8, No. 3, pp 368-379, April 1990.

[6] S. J. Golestani. *A Stop and Go Queueing Framework for Congestion Management*, In **Proceedings of SIGCOMM '90**, pp 8-18, 1990.

[7] J. Hyman, A. Lazar, and G. Pacifici. *Real-Time Scheduling with Quality of Service Constraints,* In **IEEE JSAC**, Vol. 9, No. 9, pp 1052-1063, September 1991.

[8] C. Kalmanek, H. Kanakia, and S. Keshav. *Rate Controlled Servers for Very High-Speed Networks,* In **Proceedings of GlobeCom '90**, pp 300.3.1-300.3.9, 1990.

[9] Mitzel, D., Estrin, D., Shenker, S., Zhang, L., "An Architectural Comparison of ST-II and RSVP," to appear in *Proceedings of IEEE Infocom '94*, June 1994.

[10] A. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks,* In **Technical Report LIDS-TR-2089**, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1992.

[11] Partridge, C., "A Proposed Flow Specification," *Internet Request for Comments*, RFC-1363, September 1992.

[12] S. Shenker, D. Clark, and L. Zhang, "A Scheduling Service Model and a Scheduling Architecture for an Integrated Services Packet Network", preprint, 1993.

[13] Topolcic, C., "Experimental Internet Stream Protocol: Version 2 (ST-II)," *Internet Request for Comments*, RFC-1190, October 1990.

[14] D. Verma, H. Zhang, and D. Ferrari. *Delay Jitter Control for Real-Time Communication in a Packet Switching Network,* In **Proceedings of TriCom '91**, pp 35-43, 1991.

[15] Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D., "RSVP: A New Resource ReSerVation Protocol," *IEEE Network Magazine*, September 1993.

[16] Zhang, L., Braden, R., Estrin, D., Herzog, S., and S. Jamin, *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification,* RFC in preparation, 1993.