Session Directories and Scalable Internet Multicast Address Allocation

Mark Handley
USC Information Sciences Institute
mjh@isi.edu

Abstract

A multicast session directory is a mechanism by which users can discover the existence of multicast sessions. In the Mbone, session announcements have also served as multicast address reservations - a dual purpose that is efficient, but which may cause some side-affects as session directories scale.

In this paper we examine the scaling of multicast address allocation when it is performed by such a multicast session directory. Despite our best efforts to make such an approach scale, this analysis ultimately reveals significant scaling problems, and suggests a new approach to multicast address allocation in the Internet environment.

1 Introduction

A multicast session directory is a mechanism by which users can discover the existence of multicast sessions, and can find sufficient information to allow them to join a multicast session. Such a session is minimally defined by the set of media streams it uses (their format and transport ports), by the multicast addresses and scope of those streams. A session directory distributes this and additional descriptive information by periodically multicasting announcements so that receivers can decide which sessions they would like to join.

Since the early days of the Mbone, session directories have been used to perform both session advertisement and multicast address allocation. Thus session announcement messages have also served as multicast address reservations - a dual purpose that is efficient, but which may cause some side-affects as session directories scale. We will examine the scaling of multicast address allocation when it is performed by such a multicast session directory.

Of critical interest both for session announcement and for multicast address allocation is the scope of sessions - which part of the network the data from the session will reach. There are two mechanisms for scope control in the Mbone: TTL scoping and administrative scoping. In this paper we concentrate primarily on TTL scoping, as this is the principle mechanism in use today.

Despite our best efforts to make such an approach scale, this analysis ultimately reveals significant scaling problems, and suggests a new approach to Internet multicast address allocation.

TTL Scoping

When an IPv4 packet is sent, an IP header field called Time To Live (TTL) is set to a value between zero and 255. Every time a router forwards the packet, it decrements the TTL field in the packet header, and if the value reaches zero, the packet is dropped[1]. With unicast, TTL is normally set to a fixed value by the sending host and is intended to prevent packets looping forever.

With IP multicast, TTL can be used to constrain how far a multicast packet can travel across the MBone by carefully choosing the value put into packets as they are sent. However, as the relationship between hop-count and suitable scope regions is poor at best, the basic TTL mechanism is supplemented by configured thresholds on multicast-capable links and tunnels. Where such a threshold is configured, the router will decrement the TTL, as with unicast packets, but then will drop the packet if the TTL is less than the configured threshold. When these thresholds are chosen consistently at all of the borders to a region, they allow a host within that region to send traffic with a TTL less than the threshold, and to know that the traffic will not escape that region.

Scoping Requirements

For a session announcement, the primary scoping requirements are that the session announcement is heard at all the places where the data for the session can be received, and that the announcement is not heard in places where the session cannot be received. These requirements are most easily satisfied by simply multicasting session announcements with the same scope as the session they describe.

For multicast address allocation, the primary scoping requirement is that no multicast address is allocated in such a way that the session using it (and hence the session announcement) can clash with the same address being used by another session. Here, TTL scoping and administrative scoping give us significantly different problems.

Administrative scoping is a relatively simple problem domain in that, barring failures, two sites communicating within the scope zone will be able to hear each other's messages, and no site outside the scope zone can get any multicast packet into the scope zone if it uses an address from the scope zone range.

TTL scoping suffers from an asymmetry problem - an address, either in use or being announced with the same scope as the session it describes, will not be detected outside the scope zone, but sites outside the scope zone can use the same address to get data into the scope zone. This makes multicast address allocation for TTL scoping hard. We would like to be able to use the same multicast address in multiple non-overlapping scope zones as the address space is limited and we envisage a large number of locally scoped sessions will be in use, but when choosing an address we cannot be sure that it is not in use behind some smaller TTL threshold that would clash with the session for which we are allocating the address.

This paper presents a range of solutions for allocating addresses within the context of TTL scoping. The solutions do not prohibit the use of administrative scoping; indeed the simpler solutions work

[1] The IP specification also states that TTL should be decremented if a packet is queued for more than a certain amount of time, but this is rarely implemented today.

well for administrative scope zone address allocation. However, as efficient address allocation for TTL scoping is the harder problem we shall initially concentrate on the issues it raises.

2 Multicast address allocation

Multicast addresses may be well-known addresses which are used for years, but most multicast groups are only used for a single purpose such as a conference or game and then not needed again. In IPv4, there are 2^{28} (approximately 270 million) multicast addresses available. Over time, the total number of multicast sessions is likely to greatly exceed the address space, but at any one time, this is not likely to be a problem so long as addresses are allocated in a dynamic fashion, re-used over time, and so long as scoping is used to allow the same address to be in use simultaneously for multiple topologically-separate local sessions.

As Mbone session directories such as sdr[5] are already advertising the existence of multicast sessions and their addresses to the appropriate scope zones, we have traditionally leveraged this distribution process as a part of a multicast address allocation mechanism.

There are many alternative approaches that might be taken, but if the session directory approach can be made to scale, it has many advantages including simplicity, ease of deployment and lack of dependence on any additional third-party infrastructure. We shall examine what may be achieved by an entirely distributed multicast address allocation scheme based on the existing session announcement architecture. Afterwards in summary we will examine alternatives including how this may be combined with a dynamic hierarchical scheme.

2.1 IPRMA

Van Jacobson has partially described[9] a scheme for multicast address allocation called Informed Partitioned Random Multicast Address Allocation (IPRMA). This is intended to allow a session directory instance to locally generate a multicast address with minimal chance that this multicast address will conflict with another multicast address already in use.

Schemes like IPRMA depend on the address allocator knowing a large proportion of the addresses already in use. Information about each existing session is multicast with the same scope as the session. Session directories use an announce/listen approach to build up a complete list of these advertised sessions, and a multicast address is chosen from those not already in use. However, as different sessions have different scopes, an announcement for a local session at one site will not reach all other sites, so the same address can then also be chosen for a global session at another site leading to an address clash. IPRMA attempts to avoid this by partitioning the address space based on the TTL of the session.

The general principle of IPRMA is illustrated in figure 1. This illustrates the probability of allocating a particular multicast address. The area under each of the segments of the curve is one. For a particular TTL, only one of the segments of the curve is valid, as illustrated in figure 2.

Thus, although a session directory at a particular location can only see sessions advertised that will reach its location, and cannot see sessions advertised locally[2] elsewhere, the partitioning of the address space prevents a new global allocation clashing with an existing local allocation elsewhere.

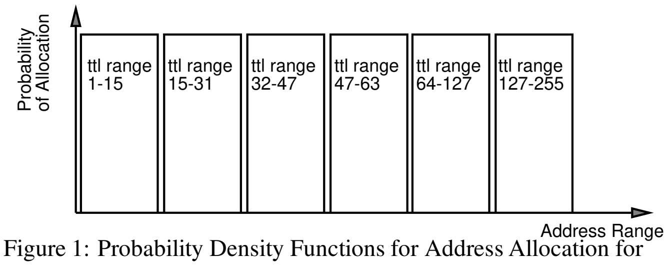

Figure 1: Probability Density Functions for Address Allocation for each of 6 TTL ranges

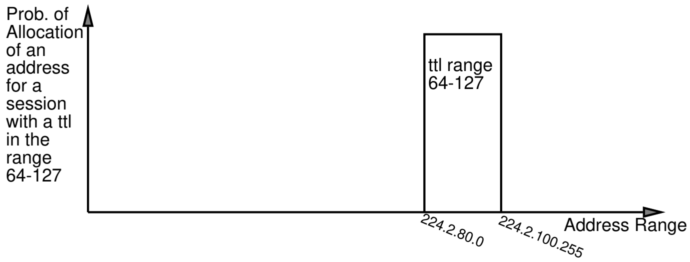

Figure 2: Probability Density Function for Address Allocation of sessions with TTL 64-127

The problem with partitioning the address space in this way is that some partitions may be virtually empty, and others will be densely occupied. If the session advertisement mechanism is perfect and all sites within a scope band can see all sessions advertised within that band, then we can fully populate a scope band. However this ideal is not achievable in practice for a number of reasons. In particular, packet loss causes delays in discovering new sessions advertised elsewhere. Any such delay means the same address can be allocated at more than one site. Another problem is that inconsistencies between TTL zone boundaries and IPRMA partition boundaries may mean that not all sites allocating addresses within a partition can see all the other addresses in use in that partition. This is illustrated in figure 3.

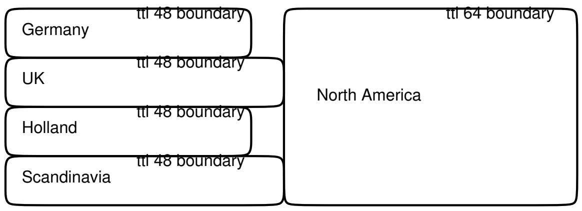

Figure 3: An example of inconsistent TTL boundary policies

In the current Mbone, boundaries between most countries are at TTL 64, but within Europe, the boundaries between countries are at TTL 48. The boundaries into and out of Europe are at TTL 64. This allows some groups to be kept within each country by sending at TTL 47 and some groups to be kept within Europe by sending at TTL 63. In the US, no TTL 48 boundaries exist, and so no TTL 47 sessions are used. Now if there is an IPRMA partition that covers

[2] locality is determined by the scope of the session, which in turn is determined by the TTL of the session

the range 33-64 (which would be appropriate for the North America region) then both Europe-wide sessions and UK-only sessions fall into the same partition. However, a session directory running in Scandinavia would not see the UK TTL 47 sessions, and might cause a clash when allocating a Europe-wide TTL 63 session.

Splitting the IPRMA address space into a larger number of ranges reduces this problem, but also reduces the number of addresses available in each range. The TTL allocations are not evenly distributed throughout the possible TTL range, and in fact occur at only a few discrete values. Splitting the available address range into a set of fixed ranges means that many of those ranges are empty whilst a few are full.

As there are delays in one site discovering that another site has announced a session, IPRMA randomly assigns addresses from the relevant partition. Using a random mechanism is necessary because we do not know which sites with which we are likely to clash and do not know how many of them there are. Using a purely random allocation mechanism within a scope band would lead to an expected address clash when approximately the square root of the number of available addresses in the scope band are allocated. Figure 4 illustrates the probability of a clash for allocations from an address space of 10,000. This is the well known "birthday problem", so named because the probability of their being two children in the same class with the same birthday is high for typical class sizes.

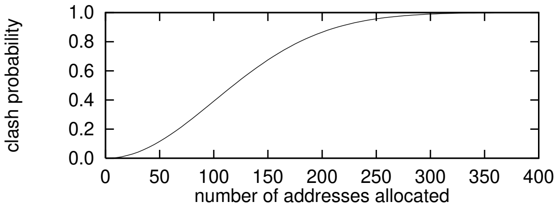

Figure 4: Probability of an address clash when allocating randomly from a space of 10,000

IPRMA's mechanism is not purely random. Addresses that an address allocator knows are in use are not chosen, so the random allocation is only from those addresses which are unallocated and from those which the session directory has failed to inform the address allocator are in use. Thus the probability of an address clash is dependent on how well IPRMA's partitions match the TTL boundaries in use and on how good a view the address allocator has of the sessions already allocated. If the address allocator is using session announcements to discover address usage, and has been running continuously, then the accuracy of the address allocator's model is dependent on the mean propagation delay (taking into account packet loss) and the rate of creation of session advertisements.

2.2 Simulation-based Comparison of Algorithms

To illustrate the different algorithms in a more realistic setting, we took a map of the real Mbone as gathered from the *mcollect*[3][4] network monitor, and built a simulation model of the Mbone topology including all the TTL thresholds and DVMRP routing metrics in use. The mcollect data is not a complete mapping of all of the Mbone because some mrouters do not have unicast routes to the mwatch daemon, but it represents a large proportion of mrouters in use. Any disconnected subtrees of the network were removed, and the resulting connected graph includes 1864 distinct nodes.

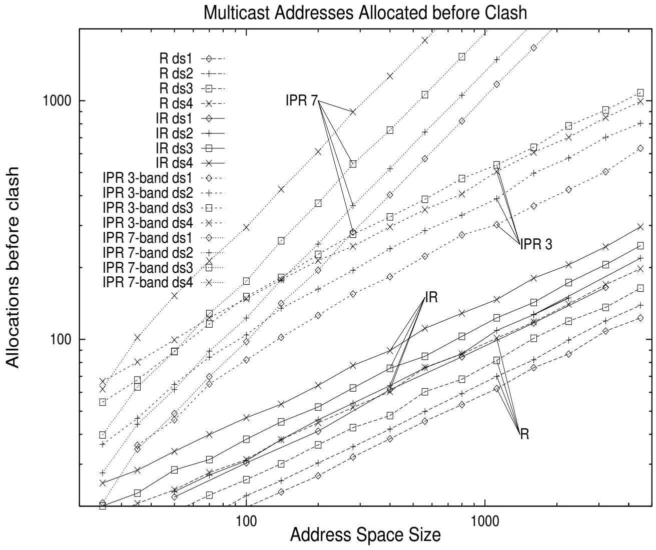

Figure 5: Simulations of address allocation algorithm performance

Nodes in this graph were chosen at random as the originator of a session, and the TTL for the session was chosen randomly from the following distributions:

ds1 {1,15,31,47,63,127,191}

ds2 {1,1,15,15,31,47,63,127,191}

ds3 {1,1,1,1,15,15,15,15,31,47,63,127,191}

ds4 {1,1,1,1,1,1,1,1,15,15,15,15,15,15,31,31,47,47,63,63,127,191}

Although these TTL distributions are not based on realistic data, they help illustrate the way that local scoping of sessions helps scaling, even where it defeats the informed allocation mechanisms.

Four algorithms were tested:

R - pure *random allocation*

IR - *informed random allocation*. An address is not allocated if it is seen in another session announcement

IPR 3-band - *informed partitioned random allocation* with 3 allocation bands separated at TTLs 15 and 64

IPR 7-band - *informed partitioned random allocation* with 7 allocation bands separated at TTLs 2, 16, 32, 48, 64 and 128

IPR 3-band illustrates the effect of imperfect partitioning as discussed in reference to figure 3. IPR 7-band is basically perfect partitioning in this case, as no two different TTL values from the distribution fall into the same band.

In this simulation we assume no packet loss, and this gives unrealistically good results for the informed schemes. We will look at the effects of loss later. Routing is performed using the DVMRP routing metrics, and scoping achieved using the TTL thresholds configured in the Mbone as reported by mcollect.

The results of this simulation are shown in figure 5 on a log/log graph. As can be seen, random (R) and informed random (IR)

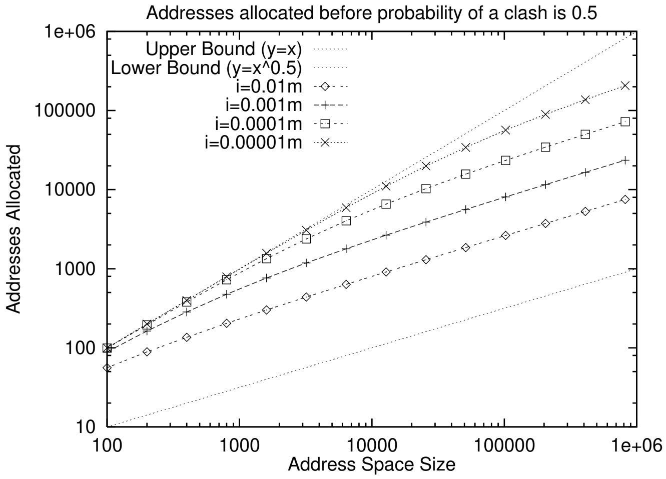

Figure 6: Addresses allocated in one IPRMA partition such that the probability of a clash is 0.5

achieve a mean allocation of $O(\sqrt{n})$ before an address clash occurs, where n is the number of addresses available. Also interesting is that informed-random is not a great improvement on random allocation.

Informed Partitioned Random with 3 bands does significantly better than Informed Random, but still only achieves a mean allocation of approximately $O(\sqrt{n})$ before a clash occurs for larger values of n.

Informed Partitioned Random with 7 bands (perfect partition placement) achieves an optimal mean allocation of $O(n)$, and with the TTL distributions used, is limited by higher scope bands filling completely.

2.3 Effects of Announcement Delay and Loss on Perfectly Informed Partitioned Random Allocation

The simulations above ignore the effects of delay in the session announcement mechanism. To take this into account, we need to have more information about the length of sessions and of the effectiveness of the announcement protocol.

Let us assume that the mean length of a session is 2 hours, that the mean advance announcement time is 2 hours, that the mean end-to-end delay across the Mbone is 200ms, that the mean packet loss is 2% and that each announcement is resent every 10 minutes. Allowing for packet loss, these figures give a mean end-to-end delay approximated by (0.98*0.2)+(0.02*600)= 12 seconds. Given that a session is advertised for a mean of 4 hours, approximately 0.1% of sessions currently advertised are not visible at any time.

Thus given perfect partitioning for IPRMA, the probability of a clash in any given partition is determined as follows. Let n be the number of addresses potentially available in the partition, m be the number of addresses currently allocated, and i be the number of addresses invisibly allocated.

$$i = 0.001m$$

The probability, c, of any single new address allocation not clashing with an existing address is thus given by:

$$c_m = \frac{n-m}{n+i-m}$$

If we assume the total number of sessions allocated, m, is a constant, and that no session is advertised for less than 10 minutes, then the probability, p_m, of no clashes occurring within the mean lifetime of a session is given by:

$$p_m = \left(\frac{n-m}{n+i-m}\right)^m \quad (1)$$

With an address space of 65536 addresses partitioned into 8 equal regions, and even distribution of sessions (as seen from each site) across the TTL regions, IPRMA gives us a total of approximately 16496 concurrent sessions as seen from each site before the probability of a clash exceeds 0.5. Figure 6 shows a graph (computed from equation 1) of the address space size within a partition against the number of addresses allocated in that partition before the probability of a clash within any four hour period exceeds 0.5. Results are given for several different values of i. As can be seen, the address space packing is good for small partitions, but gets worse as the size of the partition increases. Clearly the efficacy of the announcement protocol is of paramount importance, as shown by the significantly better results with smaller values of i.

These numbers serve to illustrate the performance of IPRMA under near perfect conditions. As figure 6 shows, even IPRMA only manages to allocate $O(\sqrt{n})$ before the probability of a clash becomes significant when loss rates are higher because its limiting factor is the random element introduced to cope with failures of the announcement mechanism. The curve given by $i = 0.00001m$ is probably an upper bound on the performance of IPRMA as this is approximately the value given with zero packet loss and a 200ms end-to-end delay. However, we can come close to this curve by not announcing sessions at a constant interval, but starting from a high announcement rate (say a 5 second interval) and exponentially backing off the rate until a low background rate is reached. Combined with local caching servers so that new session directory instances get a complete current picture, and assuming a mean loss rate of 2%, repeating the announcement 5 seconds after it is first made gives a mean delay of about 0.3 seconds, and hence $i = 0.00005m$. Note also that many sessions are announced for much longer than the four hours assumed here.

It is important to note how the address allocation mechanism and the address announcement mechanism need to be closely coupled in this architecture. Small changes to the announcement model can greatly affect the scalability of the address allocation model.

In this analysis, it has been assumed that sessions are evenly allocated across the TTL regions and the IPRMA regions are pre-allocated and coincide precisely with the TTL boundaries in use on the Mbone. However, making these assumptions in an address allocation tool would be a dangerous path to take, as changes in TTL boundary policy could make the pre-allocation of address space highly sub-optimal.

An alternative approach is to try and make the partitions initially small and numerous, but to make their size adapt depending on the sessions already allocated.

2.4 Adaptive Address Space Partitioning

If we do not know in advance which TTL values will be used by sessions and we do not know the distribution of sessions between those partitions, then it makes sense to make the partitions themselves adaptive.

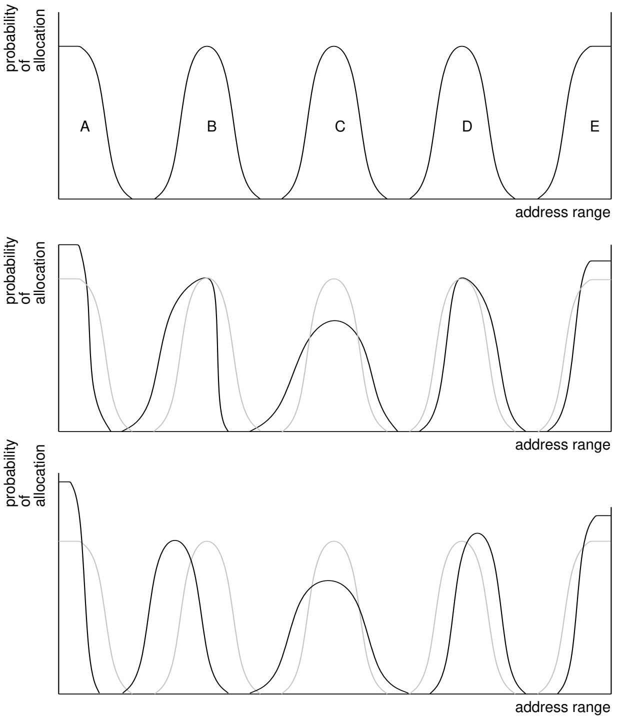

Figure 7: Two options for adaptive address space partitioning

Initially the address range is divided into even sized partitions, as shown in figure 7a. As some of the partitions start to become densely occupied whilst others are sparsely occupied, it is necessary to adapt the size of the partitions. As the size of some partitions increases, it is necessary to correspondingly reduce the size of other partitions to make space.

This can be done in one of several ways, two of which are illustrated in figure 7. In this example, many sessions which fall into TTL range C are allocated, and C needs to expand. Eventually C starts to use addresses that were originally allocated from range B. At this point, unless all sites have similar information about which addresses are in use, there is a possibility of clashes occurring between new sessions in the more widely scoped range and existing sessions in the less widely scoped range.

Deterministic Adaptive Address Space Partitioning

The major failing of adaptive IPRMA, as described above and in [9], emerges from one of two circumstances.

One way is for two or more TTLs of sessions fall into the same IPRMA address partition. This results in some lower TTL sessions in that partition not being visible at sites wishing to advertise a higher TTL session falling into the same partition.

Alternatively, a densely packed partition may expand at one site to overlap a lower TTL partition at another site. The higher TTL partition is constrained in its growth at one site by a large number of allocations in a lower TTL partition. At another site, these lower TTL allocations are not visible, so the higher TTL partition sees different constraints to its growth, resulting in an overlap with the densely packed partition at the first site.

Both of these situations result in a failure of the "informed" mechanism in IPRMA. The first situation can be prevented by increasing the number of partitions, but at the expense of exacerbating the second situation as partitions start smaller and therefore need to grow more to avoid address clashes caused by delays in announcement propagation. As figure 6 showed, even small failures in the informed mechanism have a significant effect on the number of addresses that can be allocated without a clash occurring.

If we assume that the announcement protocol communicates all sessions announced at a particular TTL to all the sites reachable at that TTL with no delay, then *there is a deterministic solution to the above problem*.

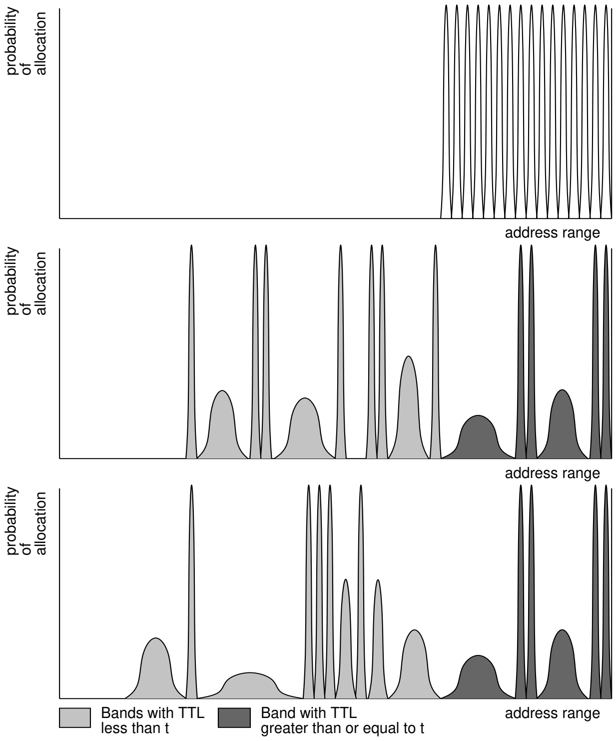

Figure 8: Illustration of Deterministic Adaptive IPRMA

To achieve this, we partition the address space into sufficient partitions that only one frequently used TTL value falls into each address range. Under these circumstances, the original adaptive IPRMA can fail because the size and/or position of an IPRMA partition is affected by both higher and lower TTL partitions, and space is wasted by empty partitions. However, if we assume a perfectly reliable announcement protocol[3], then any site wishing to allocate a session S with a TTL of x can see the session announcements for all sessions that S can clash with that have TTLs greater than or equal to x. Thus, in our variant of IPRMA, every site bases the position and size of the partition corresponding to TTL x only on session announcements for sessions with a TTL greater than or equal to x. This ensures that no clash can occur due to the failings above. It requires that the initial partition sizes are very small, and that partitions are initially clustered at the end of the space corresponding to maximum TTL. Figure 8a illustrates an initial starting partitioning

[3]The Session Announcement Protocol (SAP)[6] is of course not perfectly reliable, but packet loss affects the categories much less than individual address allocations, so the simplification is not unreasonable in this case

before any addresses have been allocated. Figures 8b and c illustrate the IPRMA partitioning at two sites after a significant number of addresses have been allocated. These two sites can communicate with a TTL of t or greater.

2.4.1 Determining Adaptive IPRMA Partitioning

To implement Adaptive IPRMA, parameters need to be defined. These include the number of initial partitions and their TTL ranges, the shape of partitions' probability density function, and the desired occupancy of a partition.

Initial Partitioning

We can choose either an initial partitioning based on values currently in use in the Mbone, or one that will work for any TTL boundary allocation policy. The latter is desirable if the overheads of choosing such a partitioning are sufficiently small.

The only initial partitioning that will work for *all possible* boundary allocation policies is to partition the address range into one partition for every TTL value. Of course this is undesirable.

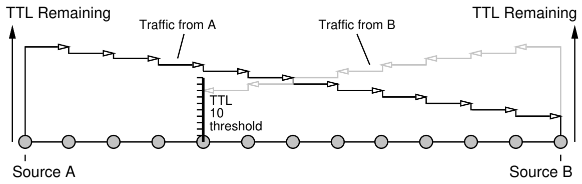

Figure 9: Potential Asymmetry due to TTL threshold

However, the TTL of packets is decremented at each multicast router traversed. If all TTLs were in use, we could not assume that if site A can see site B's TTL x traffic, site B will be able to see site A's TTL x traffic, which would make conferencing difficult. This is because there may be a high threshold boundary separating A and B, and not equidistant from them as in figure 9. At low TTL values this effect is small, but for large scopes with higher TTLs and hop counts, the effect is more pronounced.

In the real MBone, this problem is explicitly avoided by sending traffic with TTL $y - 1$ when it is intended to stay within a zone with boundary threshold y. This allows sufficient leeway for each hop to decrement the TTL while still ensuring that traffic still passes any thresholds internal to the region. If the TTLs chosen for data traffic do not suffer from this problem, then neither will the session announcements describing this traffic. Thus at high TTL values, many closely spaced TTL ranges which are also close to TTL threshold border values (and therefore need to be in separate partitions) will not occur.

Allocating one partition per TTL value is necessary at very low TTLs, but because of the way thresholds are used, it is unnecessary at high TTLs. The general guideline here is that the TTL range allocated to an address space partition must not be significantly greater than the typical hop count of sessions advertised at that TTL, but if boundary values are consistently chosen world-wide, then the TTL range need not be significantly smaller that the maximum multicast hop count.

Figure 10 shows the distribution of hop-counts available in the real Mbone, built from the mcollect network map by taking each mrouter

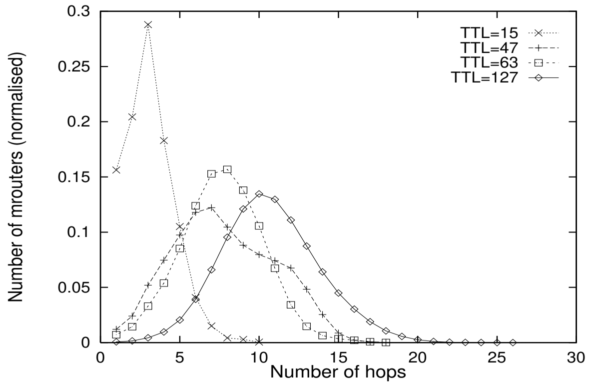

Figure 10: MBone hop count distribution for several TTL scopes

and calculating a histogram of number of mrouters against distance from that mrouter for each of four commonly used TTLs. The graph shows the combined histogram for all potential sources. TTL 47 is unusual because it is used to separate countries in Europe, but is not usually used as a boundary elsewhere in the world, where TTL 47 traffic will behave just like TTL 63 traffic.

The hop count curves in figure 10 give typical figures:

TTL	Most frequent hop count	Maximum hop count	Example Usage
255		32	DVMRP metric infinity
127	10.6	26	Intercontinental
63	7.7	18	International
47	7.0	18	National
16	3.1	10	Local

In general, the expected hop counts are approximately proportional to the TTL as this is of primary importance to network managers when setting up a TTL boundary. If our scheme is to cope with any *likely* boundaries, the size of the highest TTL band (up to TTL 255) should be less than the DVMRP infinite routing metric of 32. Boundary values are not always chosen consistently, so we also wish to build in a margin of safety to the partition sizes.

Given this, the number of TTL values, n, allocated to a partition with lowest TTL t, with a margin of safety m, is given by the following, with n rounded up to the nearest integer:

$$n = \frac{32t}{255m}$$

Choosing a margin of safety of 2 gives 55 partitions, as shown in figure 11. This will work well for existing partitioning and for any likely future partitioning.

2.5 Sizing partitions

Given that the above partitioning can be regarded as "sufficient" in the sense that it will result in an IPRMA allocation scheme having information about all the addresses in use in each partition, there are many factors that might be used to determine the size of a partition, including packet loss rates, session duration statistics, the details of the session announcement mechanism. If we have information about these, then we can use figure 6 or a derivative of it to determine a safe minimum size for the address space in a partition given the number of sessions currently allocated in that partition.

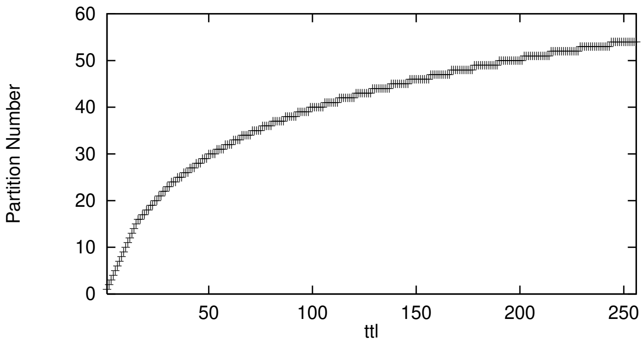

Figure 11: Mapping of TTL values to IPRMA partitions

However, measuring packet loss rates and to derive such a minimum size adaptively is likely to cause problems because we need every site to come up with the same answer to such a calculation. Session duration statistics might be a different matter, as all sites in a partition should be seeing the same sessions, but requiring a session directory to continually keep such statistics is probably more work than is necessary because it will be masked by uncertainties about loss rate and about statistics about the change rates of partitions.

For our variant of IPRMA to work well, it must be able to cope with any "flash crowds" it is likely to see. This requires an additional margin to be able to cope, and a small gap between partitions with sessions in them so that partitions can move in response to such allocation bursts without "colliding" with neighbouring partitions.

2.6 Simulations of Adaptive IPRMA

With static IPRMA (see figure 5), a reasonable method to examine scalability is to simulate filling up the address space until a clash occurs. With adaptive variants of IPRMA, such an approach is not useful as these schemes are designed to provide good address space utilisation in an environment where the number of sessions and their distribution is to some extent likely to be stable.

Simulating steady state behaviour is more difficult, as we need some definition of steady state, and also criteria for deciding whether the address allocation scheme is performing acceptably. As a criterion for acceptable performance we chose the following:

> *An address allocation scheme is acceptable if during the mean lifetime of a session the probability of an address clash anywhere in the world is less that 50%.*

This criterion is convenient, but somewhat arbitrary. The choice of a 50% probability of a clash is not important except that we have to choose some threshold. A mean session lifetime was chosen because this means we do not need to consider session lifetime in our criteria - we need only de-allocate and re-allocate as many sessions as we started with.

Thus to simulate performance of these algorithms, we use the following method and the same real-mbone topology as before:

1. Allocate n sessions with TTLs chosen from the appropriate distribution and sources chosen at random without regard for address clashes.
2. Re-allocate the addresses using the algorithm being tested so that no clashes exist.
3. Remove one existing session chosen at random.
4. Allocate a new session.
5. Repeat from 3 until n sessions have been replaced keeping score of the number of address clashes.

This process is repeated 100 times to obtain a mean value for each choice of the address space size and each choice of n to produce a table of clash probabilities. The precise value of n for each address space size where the probability of a clash exceeds 0.5 is discovered by using a median filter to remove remaining noise. Although a search algorithm can be used to locate the approximate range of n for each address space size, this is still an expensive simulation to run to any degree of accuracy or scale.

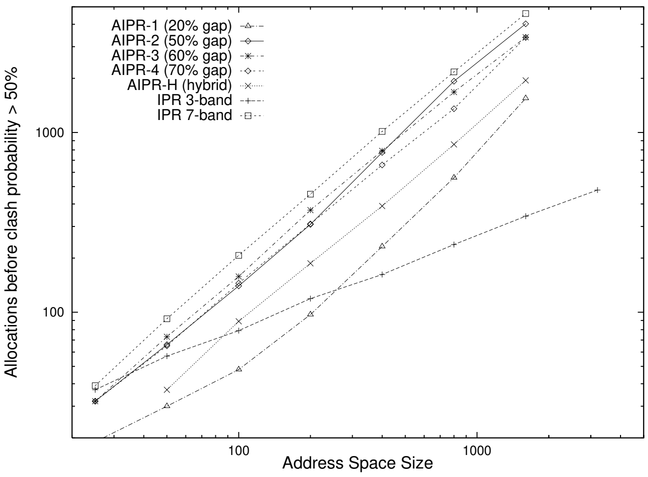

Figure 12: Steady state behaviour of Adaptive Informed Partitioned Algorithms

Figure 12 shows the results of running this simulation with various algorithms. The TTL distribution is DS4 as in figure 5. The algorithms shown are:

AIPR-1 - an adaptive informed partitioned random algorithm as illustrated in figure 8. In this case, the bands are rectangular, 20% of the address space is evenly allocated to inter-band spacing, and the target band occupancy is 67%. The initial band allocation allocates only a single address to each band.

AIPR-2, AIPR-3, AIPR-4 - as AIPR-1 except 50%, 60% and 70% respectively allocated to inter-band spacing.

AIPR-H - an adaptive informed partitioned random algorithm forming a hybrid between IPRMA-7 and AIPR-1. 20% of the address space is evenly allocated to inter-band spacing, and the target band occupancy is 67%. The initial band allocation occupies the upper 50% of the address space.

IPR 3-band, IPR 7-band - the static 3-band and 7-band informed partitioned random algorithm as used in figure 5. We use them here as a control experiment to compare static and dynamic partitioning schemes.

As before, IPR 3-band and IPR 7-band use static allocation bands based on TTL. No gap between the bands is required.

AIPR-1, AIPR-2, AIPR-3 and AIPR-4 allocate bands depending on the number of sessions in the band. Each band begins with only a single address and expands with the goal of 67% occupancy[4]. Bands are initially positioned at the top of the address space, and higher TTL bands which expand "push" lower TTL bands down the address space. AIPR-1 allocates 20% of the available address space to inter-band gaps. AIPR-2 allocates 50% of the space to inter-band gaps, AIPR-3 60% and AIPR-4 70%. These gaps are needed to absorb natural variations in band occupancy, where a higher TTL band can expand and move down the address space potentially causing clashes with old addresses in lower TTL bands.

AIPR-H is a hybrid between IPR-7 and AIPR-1. It has 7 bands as in IPR-7. These bands are initially positioned so that they occupy the top 50% of the address space with 20% of the space being used for inter-band gaps. When a high TTL band expands, it pushes downwards, but the band below it does not move downwards unless the occupancy is greater than 67%. If the occupancy is less than 67% the band is reduced in width.

Of the adaptive schemes, AIPR-3 performs best in this simulation. This result was somewhat surprising as so much of the address space is reserved for gaps between the allocation bands. However, the explanation for this is that this behaviour is due to the nature of the simulation. As session originators are chosen at random and TTLs are chosen randomly from the DS4 distribution, the number of highest TTL sessions does not fluctuate beyond what would be expected from the variation in number of high TTL sessions. However the lower TTL sessions not only change globally in number, but also change in their location, and this leads to large variations in number of low TTL sessions *visible from a particular site*. We postulate that this behaviour does not accurately represent what happens in the real world, where a particular community chooses a TTL for their sessions and the number of sessions that community creates varies within more restricted bounds than would be the case in our simulator. Thus the adaptive schemes which assume some degree of stability in the number of *visible* sessions in any particular band are actually performing surprisingly well given the nature of the simulation, but the rapid variations in visible sessions are reflected in the need for excessively large inter-band gaps. This is also reflected in the relative improvement of AIPR-1 and AIPR-2 as the address space increases. With a larger number of addresses allocated, the relative size of the fluctuations in lower TTL bands is reduced, and the algorithms start to perform somewhat better. It would be interesting to simulate larger address spaces than the 1600 addresses that are simulated here, but to test n addresses allocated, the simulation requires $O(n^3)$ time and $O(n^2)$ space (or $O(n^4)$ time and $O(n)$ space), and this makes simulating larger spaces not feasible with the resources available. [5].

The nature of clustering of session allocations in the real world is not well understood, and so devising an appropriate simulation of this is somewhat difficult. Producing an upper bound is somewhat simpler, and can be done by replacing a session advertised from a site with a particular TTL with a session advertised from the same site with the same TTL. This is not a particularly interesting simulation as it doesn't test the adaptation mechanism itself, but merely the limits to how far the mechanism can adapt. In the case of the static schemes, this tests the point at which one band in the scheme typically becomes full. For the adaptive schemes, this typically tests the point at which the TTL=1 band runs out of address space at some point in the topology.

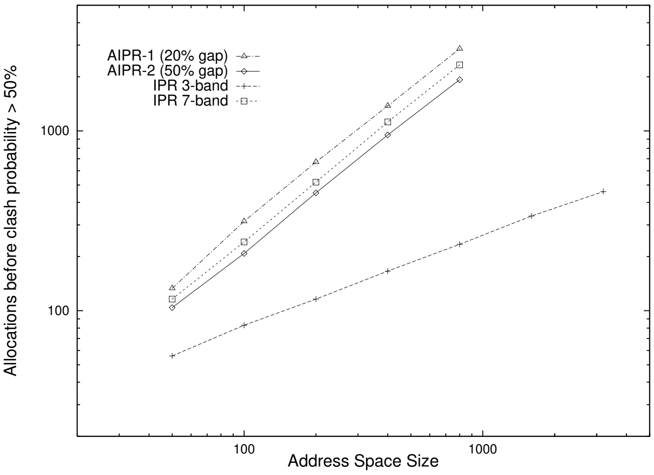

Figure 13: Upper-bound on steady state behaviour of Adaptive Informed Partitioned Algorithms

Simulations of this upper bound are shown in figure 13. As would be expected, AIPR-1 with 20% of the space allocated to bandgaps does the best of the schemes simulated, and considerably better that AIPR-2 (50% allocated to bandgaps). The static scheme IPR-7 still performs well, but as noted, this scheme is not practical unless the precise TTL values to be used are known in advance.

Without more knowledge about the nature of session clustering as multicast sessions become more common, it is difficult to go beyond the bounds we have explored here for adaptive address allocation schemes. We have shown that with Deterministic Adaptive IPRMA the number of allocations scales linearly with the size of the address space, which was our goal, but that there are tuning parameters (such as the inter-band gap size) that can make significant (constant multiplier) changes to the performance. Without tuning these parameters AIPRMA is still robust to changes in clustering, and so rather than speculate on future session clustering properties, we shall explore other issues that influence scaling.

3 Detecting an allocation clash

This far, we have attempted to design an address allocation mechanism that avoids allocation clashes. Given the decentralised mechanisms used, we cannot guarantee that clashes will not occur, but we can detect those that do occur and provide a mechanism to cause an announcement to be modified under such circumstances.

If a session directory instance that is announcing a session hears an announcement of another session using the same address, it may retract its own announcement or tell the other announcer to perform the retraction, or both. If a session directory has only made a single announcement then the clash is likely to be because of propagation delay, and so simply retracting the announcement is possible. However, it may be that the site cannot respond because it did not hear the new announcement due to some temporary failure, so un-

[4] 67% was chosen from figure 6 as approximately the proportion of the address space that can be allocated for a band of 10000 addresses before propagation delay and loss alone increase the clash probability to 0.5

[5] To simulate a single data point for AIPR-3 for 1600 addresses (3000 allocations) takes approximately 24 hours and 36 MBytes of memory on a 200MHz DEC alpha using the $O(n^3)$ time algorithm

der such circumstances we would like other sites to be able to report the clash. Thus we end up with a three phase approach:

1. A site that has had a session announced for some time discovers a clash with that session and re-sends its announcement message immediately. This will typically not occur unless a network partition has been resolved recently.

2. A site that just announced a session (whether new or pre-existing) sees another session announced with the same address within a small time window. Such a clash may occur due to propagation delay. It immediately sends a new announcement with a modified address.

3. A third party that has not announced this session sees a session announcement with an address that clashes with one of the sessions in its cache. It waits to see if the cached entry is re-announced by someone else, or if the new session is modified to resolve the clash. If neither of these has occurred after a certain amount of time, it re-announces the session on behalf of its originator.

This approach means that existing sessions will not be disrupted by new sessions. Existing sessions can only be disrupted by other existing sessions that had not been known due to network partitioning.

Allowing third parties to defend existing addresses helps cope with cache failures and partitionings where the two announcers are partitioned from each other but a third party can still communicate with both systems. However to avoid an implosion of responses, a distributed algorithm must be used to decide which third party should send its response at which time.

The simplest such distributed algorithm involves delaying a response by some random delay to allow other sites the possibility to respond. If no other site responds before this time interval elapses, then a site sends. To investigate how long this delay should be, we simulated such a generic multicast "request-response" protocol. Similar mechanisms are used in SRM[2] but in the address allocation case the group is likely to be larger, the delay matrix unknown, and even the size of the receivership unknown. In SRM and here, a member that receives a "request" delays its response by a value chosen randomly from the uniform interval $[D1{:}D2]$, and cancels its response if it sees another receiver respond within this delay period. In this case $D1$ is chosen so that the originator of an announcement can be expected to have had a chance to reply and suppress all other receivers. The value of $D2$, the topology, and the number of receivers will determine the expected number of responses and the delay before the first of those responses is received.

We can get an upper bound on the number of responses we obtain by simplifying the situation. If the highest round trip time between group members is R, then we can regard the interval $[D1{:}D2]$ as being d buckets of size R. The number of ways that n packets can be placed into d buckets (numbered from 1 to d) is d^n, and each of these is of equal probability. The number of ways k out of these n packets can be placed in bucket b and the remainder placed in the other buckets is given by

$$\frac{n!}{k!(n-k)!}(d-1)^{n-k}$$

The probability of k packets being in bucket b, $p_{k,b}$, is given by:

$$p_{k,b} = \frac{n!}{k!(n-k)!}\frac{(d-1)^{n-k}}{d^n}, 1 \leq b \leq d$$

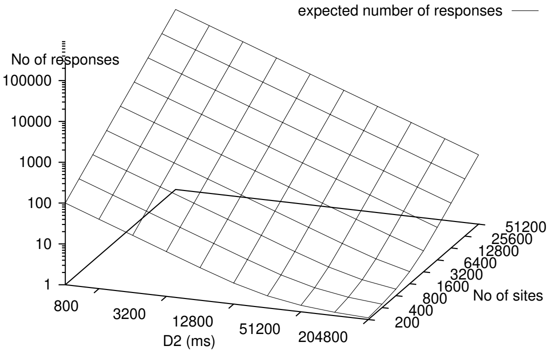

Figure 14: Upper bound on number of responders with discrete uniform delay interval

Given that k packets are in bucket b, the probability, z_b, of no packets being in buckets 1 to $b-1$ is given by:

$$z_b = \left(\frac{d-b}{d-1}\right)^{n-k}$$

As k responses can be obtained by k packets being in bucket 1, or by k packets being in bucket b and no packets being in buckets 1 to $b-1$, the expected number of packets, E, is given by:

$$E = \sum_{k=1}^{n} k \sum_{b=1}^{d} p_{k,b} z_b \qquad (2)$$

This isn't the simplest derivation of this value, but we will wish to experiment with non-uniform bucket probabilities, and so its general form suits our purposes.

Figure 14 graphs equation 2 for a range of values of d and n for $R = 200ms$. This gives an upper bound on the number of responses, as it ignores shorter round trip times than R and suppression *within* a bucket, because discrete buckets are used.

To investigate the effects of variable delay, realistic topologies, and suppression within a bucket we must resort to simulation. To generate realistic topologies, and to be able to investigate the dependence of the scheme on the multicast routing scheme, we generated topologies as follows:

- The "space" is a square grid. Nodes are allocated coordinates on this grid.
- A new node is connected to its nearest neighbour on the grid. Thus the first few nodes create the "backbone" long links and later nodes provide more local clustering. This creates a tree similar to shared trees created by CBT and sparse-mode PIM.
- Optionally, nodes a through b are additionally connected to another pre-existing node at random. a and b are $n/30$ and $n/20$ respectively where n is the total number of nodes. This provides redundant backbone links which can be exploited to form source-based shortest path trees to simulate DVMRP and sparse and dense mode PIM.

The resulting graph is very similar to those generated by Doar[1] and has a hierarchical structure, with a variable number of multiple paths that together make this class of simulations a reasonable

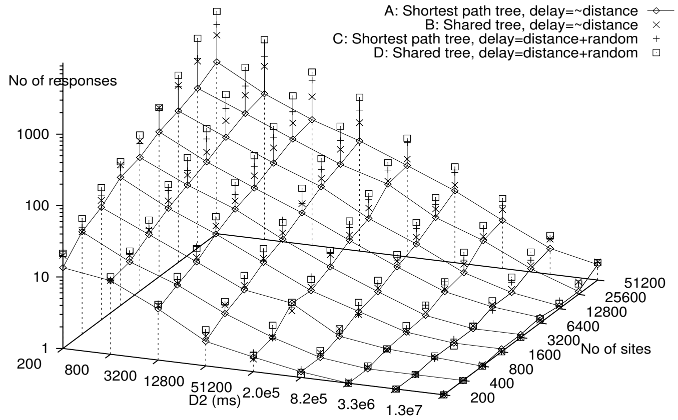

Figure 15: Simulations of a Multicast Request-Response Protocol

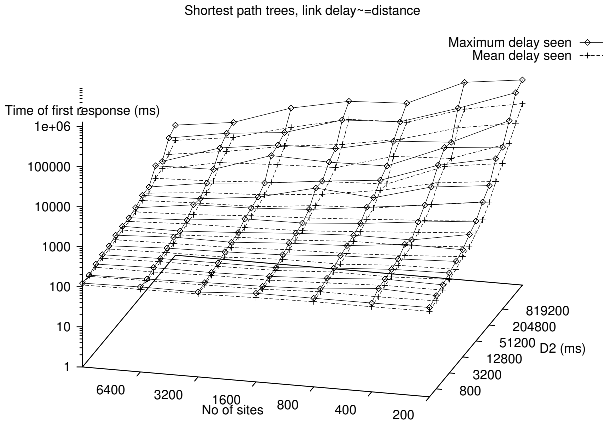

Figure 16: Delay in a Multicast Request- Response Protocol

model of the real internet. Link delays were primarily based on distance between the nodes forming the link, plus optionally a random per-hop amount on a per-packet basis to simulate queuing.

Figure 15 shows the results of these simulations for varying sizes of networks and values of $D2$. Results are shown for a source-based shortest-path tree multicast routing algorithm and for a shared tree algorithm. Additionally, we simulated random jitter as would be caused by variable queuing delays. Figure 16 shows the delay before the first response corresponding to the equivalent simulation infigure 15.

These results indicate that suppression is insufficient to reduce the number of respondees to close to one for large group sizes without incurring significant delays when the number of potential respondees is small. They indicate a small difference between shortest-path trees and shared trees in terms of the suppression process[6], but not one that greatly affects the choice of mechanism.

Thus we need to modify the basic algorithm for use in the circumstances we are considering. SRM modifies the algorithm by making the delay dependent on the previously measured round-trip delay from the data source, but we clearly cannot do this. However there are a number of things we can do to help.

[6] the number of respondees is smaller with shortest-path trees than with shared trees

Firstly, we can reduce the number of possible responders by initially only allowing the sites that are actually announcing sessions to respond. This has the advantage that we know the number of announcing sites, and so we can use this as a parameter in any solution. These sites should be distributed throughout the network, so they should approximate the distribution of all sites. Sites that are not session announcers can always be allowed to respond later by setting their $D1$ value to the value of $D2$ of the announcing sites.

Secondly, we can change the uniform random interval to be a non-uniform random interval. Lastly, we can arbitrarily rank the sites using any additional information that we have.

3.1 Non-uniform random interval

If instead of choosing a delay uniformly from the interval $[D1, D2]$, we choose a delay from an interval with a different distribution, we can reduce the value of $D2$ and hence reduce the worst-case delay without increasing the expected number of responses.

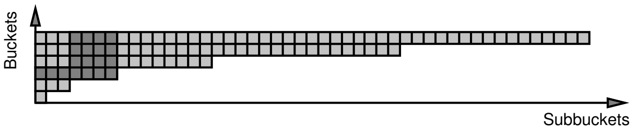

Figure 17: Bucket-to-subbucket mapping

An exponential distribution has desirable properties to use in such a scenario. Consider again the upper bound given by equation 2, but instead of having d buckets of equal probability, we have d buckets such that the probability of bucket b (where $1 < b \leq d$) is double that of bucket $b-1$. Numbering the buckets from 1 to d, this is equivalent to choosing uniformly from $2^d - 1$ sub-buckets, where the bucket b contains 2^{b-1} sub-buckets (see figure 17). Thus the number of ways n packets can be placed in $2^d - 1$ sub-buckets is $(2^d - 1)^n$, and each of these has equal probability. The number of ways k out of n packets can end up in bucket b is now given by:

$$\frac{n!}{k!(n-k)!} 2^{(b-1)k} (2^d - 2^{b-1} - 1)^{n-k}$$

The probability, $p_{k,b}$ of k out of n packets being sent in bucket b is:

$$p_{k,b} = \frac{n! 2^{(b-1)k}}{k!(n-k)!} \frac{(2^d - 2^{b-1} - 1)^{n-k}}{(2^d - 1)^n}, 1 \leq b \leq d \qquad (3)$$

Given that k packets are in bucket b, the probability, z_b, of no packets being in buckets 1 to $b-1$ is given by:

$$z_b = \left(\frac{2^d - 2^b}{2^d - 2^{b-1} - 1} \right)^{n-k}$$

Again, the number of expected packets is given by:

$$E = \sum_{k=1}^{n} k \sum_{b=1}^{d} p_{k,b} z_b \qquad (4)$$

Figure 18 shows the expected number of responses from equation 4 for a range of values of d and n for $R = 200ms$. This gives appropriate numbers of responses for relatively small values of $D2$. Note that unlike in figure 14, the curve does not tend to one response in the extreme[7], and this is the small price we pay for using an exponential in this formula.

[7] the limit in this case is a mean of 1.442698 responses

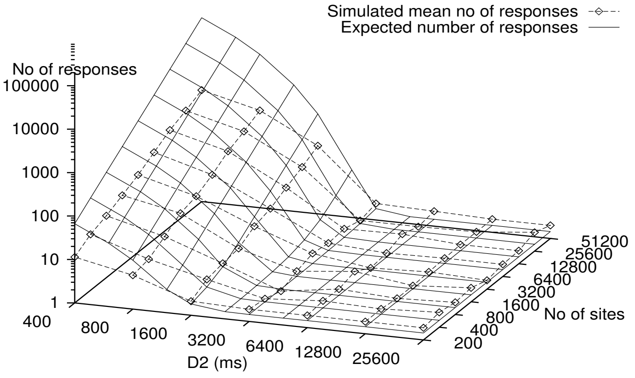

Figure 18: Number of responders using exponential delay interval

Again, this curve approximates an upper bound on the number of responses, and we must simulate the behaviour to take account of differing round-trip times and more natural suppression of responses. Figure 18 also shows the results of simulating this exponential function in a continuous form to randomly choose a delay before responding. The simulator used is identical to that used for figure 15 except for the change in delay function. The precise delay, D, for a group member is generated using:

$$D = r.log_2((2^d - 1)x + 1)$$

where $d = \frac{D2-D1}{r}$, r is the maximum RTT, and x is a random number chosen uniformly from [0:1]. In practice, a dependence on an *accurate* estimate of RTT is unnecessary, and is introduced here to ensure that the curves in figure 18 have the same time axis. The value of d is of primary importance in determining the number of responses.

We can conclude from comparing the simulation with the simplified theoretical prediction that suppression occurring within one RTT is significant only when the number of responses is large, and that this is a regime in which we do not wish to operate.

Using a delay chosen from an exponential random distribution results in a sharp distinction below which the number of responses is large and above which it is small. This cut-off point only increases slowly with the size of the multicast group.

The number of responses is not the only important value; equally important is that the delay before the first response is not excessive. A few seconds delay is acceptable for this application, but hundreds of seconds is probably not so as this will not permit sufficiently rapid retraction of a announcement with an address clash. Figure 19 shows the mean number of responses against the mean delay before the first response for the simulations in figure 15C and figure 18. A curve is shown for each value of $D2$; each curve shows eight points corresponding to the number of receivers ranging from 200 to 25600. Thus although both uniform and exponential random delays provide acceptable behaviour (around two responses and one second delay), the uniform random delay is very dependent on the size of the potential receiver set, whereas the exponential random delay allows us to choose a value of $D2$ that suits a wide range of receiver sets. As we do not know a-priori how many receivers might know about a particular clash, it is clear that the exponential random delay is much simpler to deploy to achieve acceptable behaviour.

Instead of adjusting the distribution from which members choose a random delay, we can also use other natural differences to avoid multiple responses. Additional examination of one such approach can be found in [8], but for this application, the approach above yields the best results.

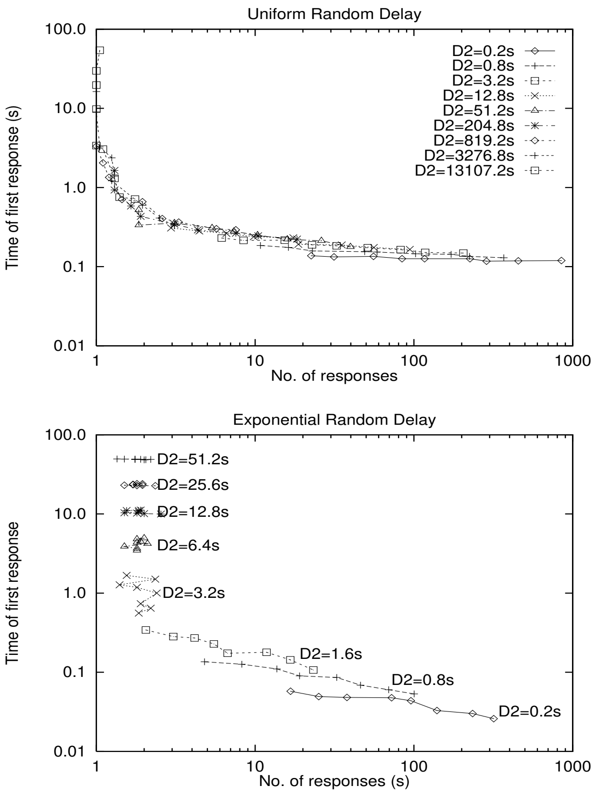

Figure 19: Simulation of Multicast Request-Response Protocol performance for both uniform and exponential random delay

4 Conclusions

The announcement mechanism in sdr is extremely robust, and can be very timely (compared to alternative mechanisms). Using the same mechanism for session announcement and multicast address allocation is elegant, but this analysis shows that this approach has some limitations.

Using the same mechanism means that if multicast address allocation is to scale reasonably, the following requirements are placed on the session announcement mechanism:

The session announcement rate must be non-uniform. To get multicast address allocation to scale reasonably, figure 6 shows that the mean propagation delay must be low. This indicates that the announcement rate should be non-uniform. Optimally, it should start from a high announcement rate (say a 5 second interval) and exponentially back off the rate until a low background rate is reached. This uses the bandwidth effectively from the point of view of reducing mean propagation delay due to packet loss, but as the background rate will now be lower for the same total bandwidth, missing sessions will take longer to be discovered.

The same announcement channel must be used by all announcements of the same scope. This is necessary for the session directory to be able to build up a complete list of sessions in the scope range so that it can perform multicast address allocation. However, this means that all sites must receive all appropriately scoped session announcements, which may be desirable whilst the MBone is relatively small, but ceases to be so as the MBone scales and distinct user groups emerge. As this happens, the amount of bandwidth dedicated to announcements would have to increase significantly or the inter-announcement interval would become too long to give any kind of assurance of reliability.

To support these distinct groups we would like to dynamically allocate new announcement addresses for certain categories of announcement, and only announce the existence of the category on the base session directory address. This would function in a similar way to dynamic adaptive naming in CCCP[7], and would allow receivers to decide the categories for which they receive announcements, and hence the bandwidth used by the session directory.

Whilst a session directory is using the same mechanism for announcements and address allocation, this is not possible[8]

A mechanism must be provided to detect and correct address allocation clashes. Thus a global session announcement may be made, and then be "corrected" as the allocation clash is discovered. We have demonstrated an approach that scales appropriately.

4.1 Beyond sdr: Further Improving Scalability

Despite the simplicity of the session directory model, if we are to achieve scalable session directories and scalable multicast address allocation, this study implies that session announcement and multicast address allocation should be separated from each other.

From the point of view of session announcement, this would mean that multiple groups can be used employing either a manually configured hierarchy of announcement groups or, more ideally, a dynamic arrangement of session categories across announcement groups. This would reduce the state that a session directory needs to keep to be only that in which the user is interested, and would reduce session announcement bandwidth at the edges of the network.

For multicast address allocation, figure 6 which indicates the effects of announcement packet loss, gives most cause for concern. The conclusion to be drawn is that for global sessions, even a good session announcement mechanism with a perfect version of IPRMA cannot expect to allocate an address space of 270 million addresses effectively. It could probably allocate an address space of 65,536 addresses (the current size of the IANA range for dynamically-allocated addresses), but we should be aiming for higher goals.

To further improve the scalability of multicast address allocation, we believe a hierarchy needs to be introduced.

At the lower level of the hierarchy, an address allocation scheme similar to the one described here can be used to allocate addresses from a space of up to 10,000 addresses - this work in this paper implies that this is a reasonable bound on flat address space allocation.

[8] In theory, we could partition the address space by category. However, as many categories will only exist in local scopes, this introduces further problems. Such a category-partition-based solution could probably be made to work along similar lines to AIPRMA given a total ordering of categories sorted using scope as a primary index, with an additional announcement address for category address usage summaries, but this introduces more complexity, and is open to denial of service attacks on the summary address. In addition, a locality-based solution is more likely to help with making sparse-mode multicast routing scale than a category-based solution is.

At the higher level, a dynamic "prefix" allocation scheme should be used based on locality. At a particular location, the lower-level scheme assigns addresses from the prefix allocated to the region encompassing that location. To get good address space packing, the prefixes themselves need to be dynamically allocated too, based on how many addresses are in use from the prefix by the lower level address allocation scheme. This paper indicates that this would greatly help scaling because the timescales used to allocate prefixes can be much longer than those used for individual addresses in order to negate the effects of packet loss and so achieve low probabilities of prefix collision. This is acceptable because the timeliness requirements for prefix allocation are much more relaxed than for individual addresses. In addition, the lower-level scheme would only need to announce the addresses in use within the local region, and this improved locality means that more address-usage announcement messages can be sent increasing the timeliness significantly.

Such a hierarchical scheme would dynamically associate multicast prefixes with regions of the network. The routing tables so derived can be used to discover the location of shared-tree cores, and we plan to use them as a part of Border Gateway Multicast Protocol (BGMP)[11], a new upper-level multicast routing protocol intended to introduce hierarchy to multicast routing. Because we want to use the prefix allocations for multicast routing, we cannot use multicast to perform prefix allocation, and so the prefix allocation mechanism will use BGP routing exchanges as a form of "multicast" to implement an announce/listen model similar to that of session directories.

We are reluctant to lose the simplicity and elegance of the current model, but this analysis indicates that an approach along such lines will be necessary if IP multicast is ever to become ubiquitous.

References

[1] M. Doar, "A Better Model for Generating Test Networks", IEEE Global Telecommunications Conference/GLOBECOM'96, London, November 1996.

[2] S. Floyd, V. Jacobson, S. McCanne, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Proc ACM SIGCOMM 95, Aug 1995 pp. 342-356.

[3] A. Ghosh, "mcollect - collect data from the mserv server (mwatch)", Unix manual page, University College London.

[4] A. Ghosh, "mserv - multicast map server (mwatch)", Unix manual page, University College London.

[5] M. Handley, V. Jacobson, "sdr - A Multicast Session Directory", Unix manual page, University College London.

[6] M. Handley, "SAP - Session Announcement Protocol", Internet Draft, Work in progress.

[7] M. Handley, I. Wakeman, J. Crowcroft, "The Conference Control Channel Protocol (CCCP): A Scalable Base for Building Conference Control Applications", Proc. ACM Sigcomm '95, Cambridge, MA., USA, ACM, 1995.

[8] M. Handley, "On Scalable Internet Multimedia Conferencing Systems", PhD Thesis, University College London, 1997.

[9] V. Jacobson, "Multimedia Conferencing on the Internet", Tutorial Notes - ACM Sigcomm 94, London, Sept 1994.

[10] P. Tsuchiya, "Efficient and Flexible Hierarchical Address Assignment", TM-ARH-018495, Bellcore, February 1991.

[11] D. Thaler, D. Estrin, D. Meyer (editors), "Border Gateway Multicast Protocol (BGMP): Protocol Specification", Internet-draft, Oct 1997.