On the Aggregatability of Multicast Forwarding State

David G. Thaler
Microsoft
dthaler@microsoft.com

Mark Handley
AT&T Center for Internet Research
mjh@aciri.org

Abstract— **It has been claimed that multicast state cannot be aggregated. In this paper, we will debunk this myth and present a simple technique that can be used to aggregate multicast forwarding state. In particular, we present an interface-centric data structure model which allows aggregation of ranges of multicast addresses in the forwarding table.**

Understanding the limits of possible aggregation is critical to our knowledge of how IP multicast will scale as it becomes widely deployed. We show through analysis and simulation that some aggregation is possible, even under purely random address allocation and purely random group membership distribution. We further show how other methods of allocation can significantly improve the ability to aggregate, and how non-random distributions of membership can affect aggregation both positively and negatively.

I. Introduction

With the advent of second and third generation multicast routing protocols[1], [2], [3], [4] and widespread host implementations, IP Multicast is finally starting to be widely deployed in the Internet. There are still relatively few multicast applications in use, but it is expected that this will start to change rapidly as application writers start to be able to assume that many hosts will have access to multicast.

To date, most work on multicast scalability has centered on the applications and on multicast routing. However, in the long run, the biggest issue facing multicast deployment is likely to be the scalability of multicast forwarding state as the number of multicast groups increases. It has been claimed [5], [6] that multicast forwarding state cannot be aggregated, but this is incorrect. In this paper we examine this issue to derive both limits and expectations for our ability to aggregate multicast forwarding state in Internet routers.

It is important to understand that our results apply to the *forwarding tables* in a router that are used to forward packets. Since a lookup among O(N) state can be done in log(N) time, reducing the state requirement is the most significant benefit of aggregation, although the speed of multicast packet forwarding may also benefit as a result. We do not concern ourselves with state the router may need to keep in order to maintain these forwarding tables, nor with routing protocols needed to maintain them. The memory to hold such ancillary state does not need to be either fast or directly located on interface processors, so it is not a significant limitation on router performance. Similarly, join messages involved in multicast routing need not be aggregated[1] so long as their rate is scaled back to prevent the routing traffic dominating, and the multicast routing protocols use appropriate scalable timers[7].

With respect to multicast forwarding state, we will argue:

- Shared trees used by second and third generation multicast routing protocols (such as CBT[2], BGMP[4], and to some extent PIM-SM[1]) mean that the number and location of sources does not affect aggregatability.
- If we consider forwarding state on a per-interface basis, then aggregation can be performed independent of the number of interfaces on a router.
- For incoming interface state on point-to-point links, and outgoing interface state on all links, we only need to consider the groups traversing a particular router and can completely aggregate all groups that do not traverse that router.
- For bidirectional trees such as those built by CBT and BGMP, no incoming interface state is required at all on point-to-point links between routers.
- Some aggregation of both incoming and outgoing forwarding state is possible, even with completely random multicast address allocation and random group membership.
- Allocating multicast addresses in a hierarchical fashion can increase the ability to aggregate forwarding state.
- The clustering of groups by locality (well known for telephony traffic) can increase the aggregatability of forwarding state when hierarchical multicast address allocation is also performed.

We will only consider *perfect* forwarding state aggregation in this paper; that is aggregation that does not change the distribution of multicast traffic. *Imperfect* or *leaky* ag-

[1] It may be possible to do some aggregation of these messages, but even if this is not done, it will not normally cause problems.

gregation is also possible, in which some low rate groups are allowed to traverse links that would not normally need to carry the traffic. Leaky aggregation is an extension of perfect aggregation that results in a further compression of the forwarding state at the expense of using additional link bandwidth. We leave the study of leaky aggregation for future work.

We also do not consider methods of reducing forwarding state that rely on encapsulation, although these methods also show significant promise. In such mechanisms (e.g., Dynamic Tunnel Multicast [8]), sparse groups may be unicast encapsulated across sections of the backbone where no multicast fanout occurs, relieving the intervening routers of the need to hold forwarding state.

The remainder of this paper is organized as follows: Section II gives some background on multicast forwarding, and section III presents an interface-centric state model. Section IV provides an analysis of aggregatability under independence assumptions, and section V presents simulations of aggregatability after removing these assumptions. Finally, section VI discusses related work, and section VII covers conclusions and future work.

II. BACKGROUND

We start by describing the actions that must be performed by a multicast forwarder. First, a packet arrives on some interface, and the forwarder must decide whether to accept or drop the packet. For example, the decision might be based on whether it arrived on the interface towards the source (known as a "Reverse Path Forwarding (RPF) check"), such as is done when DVMRP [9] or PIM-SM is running on the arrival interface. It might instead be based on some other criteria (as in the case of MOSPF [10] or CBT). In general, this decision can be viewed as applying an input packet filter $IF_i(group, source)$ per interface i. The acceptance decision can then be viewed independently of the multicast protocol in use on the interface; the protocol merely supplies the filter $IF_i : (group, source) \rightarrow \{0,1\}$.

In general, a filter F is an implementation of a function

$$F : (group, source) \rightarrow \{0,1\}$$

We now concatenate $group$ and $source$ into one double-length address a, and the filter becomes:

$$F : a \rightarrow \{0,1\}$$

This representation will allow our results to apply both to architectures which require (group,source) state, as well as those (like CBT) which only require group state and for whom a would simply be the group address. Since we desire to explore the scalability of multicast state, we will hereafter concern ourselves primarily with protocols which use group-shared trees and hence a will be synonymous with the group address. In other words, using group-shared trees makes the number and location of sources irrelevant, since per-source state is not kept.

If the packet passes the input filter, the forwarder must then determine which of its other interfaces are considered "outgoing" interfaces for the packet, and send the packet out of each outgoing interface. This operation can be logically viewed as an output packet filter per interface ($OF_i : a \rightarrow \{0,1\}$). A packet received by the router is sent out of a given interface if it passes both the input filter on the incoming interface, as well as the output filter on the outgoing interface.

Some misconceptions of state *requirements* are based on the popular Unix kernel implementation. This implementation keeps multicast forwarding state per (group,source) pair, consisting of an incoming interface, and an outgoing interface list[2]. When a packet arrives, a matching (group,source) entry is found or created, the incoming interface check is done to see whether to drop it on input, and if not dropped, then the packet is sent out of each interface in the outgoing interface list. Since this implementation is common, it is easy to incorrectly assume that this is the *only* possible implementation model. In this model, the input filter is implemented by comparing the interface i on which the packet arrived with the incoming interface stored in the (group,source) entry. Hence, $IF_i(group, source)$ is implemented as "$i == iif(group, source)$". The output filter is implemented by testing to see whether a given interface is in the outgoing interface list stored with the (group,source) entry. Hence, $OF_i(group, source)$ is "$i \in oiflist(group, source)$".

In the next section, we present an alternative implementation model which illustrates more clearly how forwarding state aggregation may be done without loss of information.

III. IMPLEMENTATION MODEL

While the Unix kernel has a (group,source)-centric state implementation model, we describe in this section an *interface*-centric implementation model.

Let each interface be associated with its own copy of an input filter and an output filter. Each filter is again such that it yields a pass-or-fail answer for a given group and source. Each interface's state is independent of the state for all other interfaces. When a packet arrives on an interface, it

[2] The list is stored in the Unix kernel as a bitmap of 32 interfaces. To efficiently support an arbitrary number, an actual list would be needed.

must first pass that interface's input filter. If it passes, then the output filter of every other interface is independently checked to see whether the packet should be sent out of that interface. In comparing the interface-centric implementation model to the (group,source)-centric implementation model, we note that the two models have equivalent functionality.

A. Router/Switch Architectures

Router technology is moving more towards having multiple parallel processors, with the extreme being a separate processor per interface (e.g., [11], [12]). Hence, the interface-centric model need not entail any processing time disadvantage over the (group,source)-centric model, since output filters may be tested in parallel. Hence, we will concentrate on the ability to aggregate state.

A survey of a number of multicast switch architectures can be found in [13]. Such architectures typically distribute multicast packets in one of two ways.

Enumerate and unicast: When a multicast packet is to be forwarded, a lookup is first done on the header to find the list of outgoing interfaces. Copies are then made, with one copy being sent to each outgoing interface.

In this method, the lookups may or may not be done in a centralized location on the router for all interfaces. Our interface-centric state model could be applied, for example, if lookups were done in a centralized location, and that module did in parallel a lookup in each outgoing interface filter.

Broadcast and filter: The multicast packet is sent to all interfaces (e.g. across a shared bus), and a "fast filter" drops those packets which should not be sent out the interface. According to [13], such switches yield the best performance, but are more expensive.

In this method, the "fast filter" corresponds exactly to an output filter in the interface-centric state model.

Recent work (e.g. [14], [15], [16], [17]) has shown that even route lookups can be done at gigabit or higher speeds.

B. Locally-Active Groups

The next step is to take into account the fact that many, if not most, addresses are not in use at any given time, and of those that are, a given router will not be on the tree for most of them. Sparse-mode protocols, such as PIM-SM, CBT, and BGMP, require no state for groups for which a router is not on the distribution tree. Dense-mode protocols, such as PIM-DM [18], DVMRP, and MOSPF, create state in routers regardless of whether the router is on the tree. Since dense-mode protocols are less scalable than sparse-mode protocols, deployment is moving more towards using sparse-mode protocols. Hence, we will hereafter assume that a router has state for a given address only if it has downstream receivers, and therefore is on the distribution tree[3].

We define a group to be "locally-active" at a given router if state is required. Hence, "locally-active" means that the local router is receiving requests for data matching the given address.

Let G_{rtr} be the number of locally-active groups. Let A be the size of the address space for a. Let $u = G/A$ be the address space utilization (or density). If locally-active groups are randomly distributed throughout the address space, then u is also the probability that any given address is in use by a locally-active group.

For all addresses which are not locally-active, let $IF_i(a) = 0 \quad \forall i$ and $OF_j(a) = d$ ("don't care") $\quad \forall j$. This invariant provides perfect filtering.

Let $\mathbf{G_{glob}}$ be the number of groups which are active anywhere in the network. Let $E[n]$ be the expected number of routers on a given multicast distribution tree. Then $\mathbf{G_{glob} * E[n] = E[G_{rtr}] * N}$, where N is the number of routers in the network. Hence $E[G_{rtr}] = \mathbf{G_{glob} * E[n]/N}$.

To investigate the aggregatability of multicast forwarding state, we are now interested in the amount of state necessary to implement a filter F as usage (in terms of number of groups, sources, and receivers) increases.

We first give existence proofs showing how a filter F can be implemented in state-efficient ways. This will give upper bounds on the forwarding state requirement. We will not show that there does not exist an even more efficient implementation model; we will however show that multicast forwarding state is inherently at *least* as aggregatable as our bounds.

IV. Aggregating Filter State

Our goal will be to construct a filter F' such that F' and F are equivalent for all addresses which matter (as defined below). We will see that in some cases, such as for addresses not in use, $F(a) =$ "don't care", denoted d. Hence, we desire an efficient $F'(a)$ where either $F'(a) = F(a)$ or $F(a) = d$.

The first step is to realize that adjacent values of a which yield the same result can be aggregated. That is, if $F(a + k) = F(a)\ \forall k : 1 \leq k \leq n$, then $F(a)$ through $F(a+n)$ are aggregatable.

Thus, a filter F can be implemented by storing a set $\mathcal{S}$ of ranges of addresses which yield a result of 1 ($F(a) = 1 \Leftrightarrow \exists r \in \mathcal{S}, \dashv \in \nabla$). If there are $R = |\mathcal{S}|$ ranges, then a binary search can be done in $O(\log R)$ time. Our objective

[3] For unidirectional trees, this means that at least one interface is an outgoing interface. For bidirectional trees, at least two interfaces are outgoing interfaces.

will be to find the relationship between R and the number of groups active at a router G_{rtr}.

We will initially assume that successive values of $F(a)$ are independent. We will then relax this assumption in Section V.

Lemma 1: **(Range Span with Independence)** Let successive values of a given filter $F(a)$ be independent. Let p be the probability that $F(a) = 1$. Let q be the probability that $F(a) = 0$. Let $r = 1 - p - q$ be the probability that $F(a) = d$ ("don't care"). Then a filter F' can be constructed with expected state

$$E[R] = A\frac{pq}{p+q} = A\frac{pq}{1-r}$$

Proof: One range, plus the inter-range gap following it, represents a series of "successes" (1's and "don't cares"), followed by a series of "failures" (0's and don't cares). The number of successes until failure follows a geometric distribution with parameter q. Hence, the expected number of successes until failure is $1/q$. Likewise, the expected number of failures until success is $1/p$.

The expected number of addresses which can be aggregated into a single range is therefore given by:

$$\frac{1}{q} + \frac{1}{p} = \frac{p+q}{pq}$$

So $E[R] = Apq/(p+q) = Apq/(1-r)$. □

Lemma 1 can be applied to input filter state when addresses are allocated randomly from a uniform distribution, and addresses have no topological significance (meaning that a router is on a random set of distribution trees), as shown in the following two theorems.

For the remainder of this paper, we will use the term R_{IF} to mean the number of ranges in an input filter, and R_{OF} to mean the number of ranges in an output filter.

Theorem 1: **(Input Filter State for Bidirectional Trees under Random Allocation)** Let addresses be allocated randomly from a uniform distribution, and have no topological significance. For bidirectional trees (which have no incoming interface check):

$$E[R_{IF}] = G_{rtr}\left(1 - \frac{G_{rtr}}{A}\right)$$

Proof: For input filter state, $r = 0$. Since filter output values are independent, $q = 1 - p$. Since no incoming interface check is done, $p = u = (G_{rtr}/A)$. Lemma 1 then gives $E[R_{IF}] = G_{rtr}(1 - \frac{G_{rtr}}{A})$. □

This function has a maximum at $G_{rtr} = A/2$. When $G_{rtr} \ll A$ (which is indeed true in today's MBone), then $E[R_{IF}] \approx G_{rtr}$, but this drops off for large values of G_{rtr}, as shown in Figure 1(a). For example, if addresses are allocated from a space of 16 K addresses (as `sdr` does today), the state reaches a maximum (of 4096 ranges) once 8192 addresses have been allocated, and state declines thereafter.

We also observe from Theorem 1 that the expected state used ($E[R]$) does not depend on the number of interfaces. Thus, the number of interfaces does not affect the aggregatability of the input filter state.

Corollary 1: **(Input Filter State for Point-to-Point Links on Bidirectional Trees under Random Allocation)** For point-to-point links between routers on bidirectional trees:

$$E[R_{IF}] = 0$$

On point-to-point interfaces between routers, where traffic is not sourced by the routers on either end, a sparse-mode protocol can use "don't cares" for addresses without locally-active groups. This is because traffic only arrives on an interface if a group was explicitly requested, or if local sources are present. Since neither is the case, $q = 0$ and hence $E[R_{IF}] = 0$. □

Backbone routers in the core of today's Internet, where minimizing state is most important, typically have at most one LAN interface, but may have many point-to-point interfaces to other routers. Hence, the corollary is potentially much more significant than the theorem, since an input filter may only be needed for at most a single interface.

The reason input filters are still required for LAN interfaces is that some *other* router or host on the LAN may have requested data for some groups.

Theorem 2: **(Input Filter State for Unidirectional Trees under Random Allocation)** Let addresses be allocated randomly from a uniform distribution. Let f be the number of interfaces on the router. For unidirectional trees (ones employing an incoming interface check):

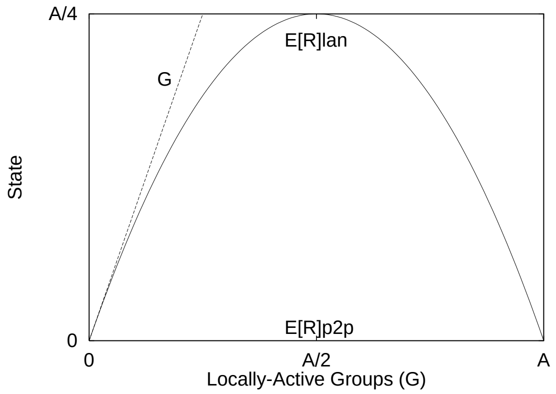
$$E[R_{IF}] = \left(\frac{G_{rtr}}{f}\right)\left(1 - \frac{G_{rtr}}{fA}\right)$$

Proof: For input filter state, $r = 0$. Since filter output values are independent, $q = 1 - p$. When $p = (G_{rtr}/Af)$, meaning that the incoming interface is randomly chosen among f interfaces, then $E[R_{IF}] = (\frac{AG_{rtr}}{Af})(1 - \frac{G_{rtr}}{Af})$. □

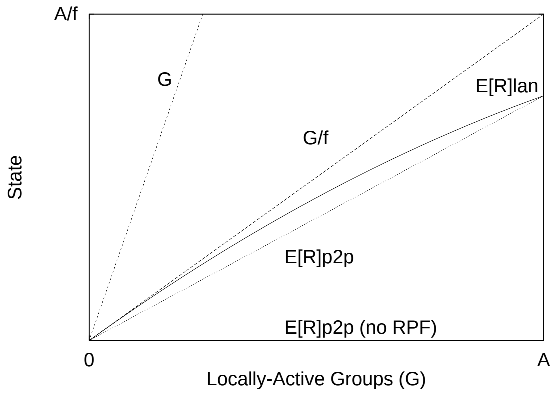
Corollary 2: **(Input Filter State for Point-to-Point Links on Unidirectional Trees under Random Allocation)** On point-to-point interfaces between routers, the expected input filter state for unidirectional trees with randomly allocated addresses is:

$$E[R_{IF}] = \left(\frac{G_{rtr}}{f}\right)\left(1 - \frac{1}{f}\right)$$

Furthermore, if the RPF check is completely removed, then $E[R_{IF}] = 0$.



(a) Bidirectional Trees



(b) Unidirectional Trees

Fig. 1. Input Filter State under Random Allocation

On point-to-point interfaces between routers, where traffic is not sourced by the routers on either end, a sparse-mode protocol can use "don't cares" for addresses without locally-active groups.

If the RPF check is completely removed, the result is the same as for bidirectional trees. However, the downside is that loops can occur while route changes are in the process of converging. Such loop prevention is one of the motivations for using unidirectional trees in the first place.

When RPF checks are done for active groups, $p = (G_{rtr}/Af)$, $r = 1 - u$, and $q = 1 - p - r$. Using these values, we obtain

$$E[R_{IF}] = \frac{A}{u}\left(\frac{G_{rtr}}{Af}\right)\left(u - \frac{G_{rtr}}{Af}\right)$$

$$= \left(\frac{G_{rtr}}{f}\right)\left(1 - \frac{1}{f}\right) \quad \Box$$

Both these functions have maximums at $G_{rtr} = A$ (since $A \leq Af/2$), as shown in Figure 1(b), where $f = 4$ was used. Furthermore, we observe that for point-to-point links, $E[R_{IF}]$ is independent of A. Thus, the amount of address space between the G_{rtr} locally-active groups is irrelevant and can be ignored.

Finally, by comparing Figures 1(a) and 1(b), we observe that for LAN interfaces, unidirectional trees require less state than bidirectional trees. However, unidirectional trees use input filter state on point-to-point interfaces while bidirectional trees do not. Since point-to-point interfaces are the norm in the network backbone, bidirectional trees can provide a significant state benefit.

Lemma 1 can also be applied to *output* filter state when addresses are allocated randomly from a uniform distribution, as shown in the next theorem. This theorem uses the fact that, for some values of a, the value of an output filter $OF'_j(a)$ doesn't matter, as they will not pass any input filter, That is, if $IF_i(a) = 0 \quad \forall i$, then $OF_j(a) = d \quad \forall j$. This provides greater aggregatability of an output filter, since a through $a + N$ can all be aggregated if $OF_j(a + k) = OF_j(a)$ or $OF_j(a + k) = d$, for $1 \leq k \leq N$, rather than only if $OF_j(a + k) = OF_j(a)$.

Other potential optimizations, which we will ignore, include:

- If $OF_j(a) = 0$ for all interfaces $j \neq i$, then $IF_i(a) = d$.
- If $IF_i(a) = 0$ for all interfaces $i \neq j$, then $OF_j(a) = d$.

Theorem 3: **(Output Filter State under Random Allocation)** Let addresses be allocated randomly from a uniform distribution. Let f be the number of potential outgoing interfaces. If, for each group, M downstream members are randomly distributed among f interfaces, then:

$$E[R_{OF}] = G_{rtr}C(1 - C)$$

$$where\ \ C = \left(\frac{f-1}{f}\right)^M$$

Proof: Since all unused addresses are "don't cares", $r = 1 - u = 1 - G_{rtr}/A$. The probability q that $F(a) = 0$ is the probability that the address is locally-active, but all downstream members have joined on other interfaces. Hence:

$$q = u\left(\frac{f-1}{f}\right)^M$$

Lemma 1 then gives:

$$E[R_{OF}] = Apq/(1 - r) = G_{rtr}C(1 - C)$$

$$where\ \ C = \left(\frac{f-1}{f}\right)^M \quad \Box$$

Again, this is $O(G_{rtr})$ state when f and M are held constant. If we vary M, then a maximum exists at $M = \frac{\log 0.5}{\log(1-1/f)}$ (e.g. $M \approx 2.4$ for $f = 4$). Another way of looking at the theorem is by saying that $E[R_{OF}] \leq G_{rtr}C$, so minimizing C is desirable; thus less state is needed as M increases, and as f decreases. For example, for $f = 2$ (such as a router with 3 interfaces using unidirectional trees), $E[R_{OF}] \leq 2^{-M}G_{rtr}$.

Hence, if the average number of downstream members M of a group grows as IP Multicast usage increases, then aggregatability could improve dramatically. This might be the case, for example, if audio/video broadcasts reached a large number of receivers, such as is the case with other broadcast mediums such as television, cable, and satellite.

A subtle point associated with decreasing f is worth mentioning. One cannot simply decrease f by removing redundant links, since this also has the side effect of increasing the number of global trees which go through the local router. Hence, doing so would also increase G_{rtr}, bringing it closer to $\mathbf{G_{glob}}$.

Finally, Theorem 3 says that aggregatability of outgoing interface state is independent of A. That is, the amount of address space between the G_{rtr} locally-active groups is irrelevant and can be ignored.

A. *Implications for the Internet*

Assuming, as we do above, that multicast group addresses are randomly allocated and that group members are randomly distributed throughout the network, makes for a worst-case scenario from the point of view of multicast forwarding state aggregation because they minimize correlation between groups.

However, even in such a scenario, we can perform a degree of multicast forwarding state aggregation. How much aggregation we achieve depends on several factors:

- Whether we're looking at the input filter or the output filter.
- Whether the multicast routing protocol is unidirectional or bidirectional.
- Whether we are looking at a point-to-point link or a LAN.

The total forwarding state is comprised of both the input and output filters, and so we need to minimize both.

An interesting special case is when we consider a bidirectional protocol such as CBT or BGMP operating over point-to-point links - such a scenario might be commonplace in the Internet backbone. In this case, no input filter state is needed at all. Figure 2(a) graphs the amount of output filter state aggregation possible (Number of local groups/expected number of ranges) as a function of the number of interfaces and the mean number of downstream group members. Thus each interface requires a number of aggregated ranges less than 25% of the number of groups traversing the router. Figure 2(b) shows the same data, but graphs the output filter state *aggregation ratio*, which we define as the number of groups traversing an interface, divided by the number of ranges on that interface. This graph assumes group members are equally distributed among interfaces. Thus each interface needs a number of aggregated ranges that is less than the number of groups with traffic exiting that interface, and often much less for large M or small f. Taking input and output filters together compared to a non-aggregated router, such an average backbone router might require less than 25% of the state to be held compared to an unaggregated router.

An interesting issue arises on local area networks (such as DMZ's at some network interconnect points) with such bidirectional trees. Here an input filter must be held, as shown in figure 1(a), and the number of ranges inversely depends on the size of the address space the groups are allocated from. For example, if a router holds state for 10000 groups randomly allocated from an address range of 16K addresses, it needs to have 3897 ranges in the filter. However, if it holds state for 10000 groups from the entire IPv4 address range of 2^{28} addresses, it will need to hold 9999 ranges. A similar effect holds for unidirectional trees such as those built by Sparse-Mode PIM. In both cases, the difference is not huge, but it argues in favor of densely allocating parts of the multicast address space before freeing up additional parts of the address space to be used. Such an address allocation mechanism is currently being proposed in the demand-driven hierarchy of MASC[4] and AAP[19].

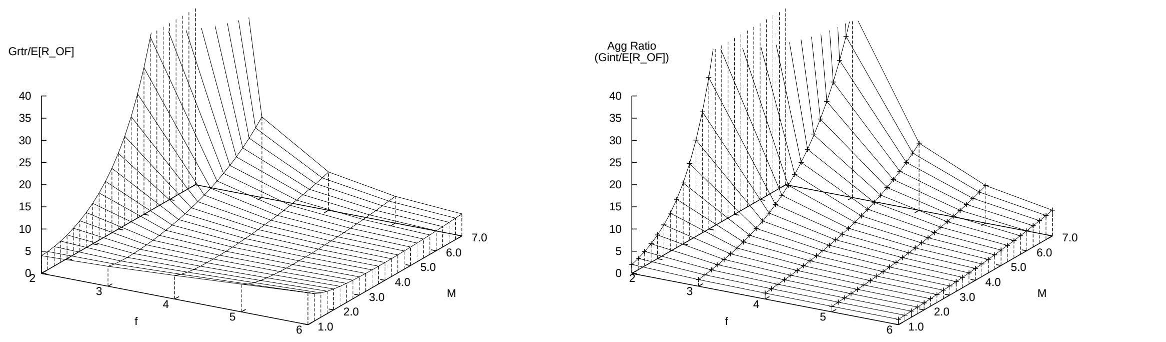
V. Removing the Independence Assumption

When successive values of a filter are independent, minimizing $E[R]$ means minimizing p and q. However, minimization is limited by the constraint $p + q + r = 1$ (e.g., $p + q = 1$ when $r = 0$).

When the independence assumption is removed, this constraint will no longer hold, providing even greater opportunities for aggregation, as we will see. That is, if we can improve the chances that 1's and 0's will be clustered, then aggregation will improve.

In this section, we will investigate the effects on aggregatability of various factors in the real Internet which break the independence assumption, including:

- Non-random address allocation
- Hierarchical address allocation
- Clustering of receivers
- Multi-group sessions



(a) Per-local-group state

(b) Per-interface group state

Fig. 2. Output Filter State under Random Allocation as a function of number of interfaces and downstream members per group

A. Non-Random Allocation

When independent groups are allocated addresses which are not distributed over A in a uniformly random fashion, then $Prob[F(a) = 1]$ is not a constant.

For example, if $\mathbf{G_{glob}}$ groups are assigned addresses sequentially starting at a_0, then $Prob[F(a) = 1 | a > a_0 + \mathbf{G_{glob}}] = \mathbf{0}$, while $Prob[F(a) = 1 | a \leq a_0 + \mathbf{G_{glob}}] > \mathbf{0}$.

However, if the groups are still joined randomly, then all results in Section IV which are independent of A still hold. This is because we have only changed the *location* of unused addresses in the range, but not the independence of the groups used.

Hence the aggregatability of output filter state, as well as that of input filter state for point-to-point interfaces, is unaffected if addresses were allocated (say) sequentially rather than randomly.

B. Hierarchical Address Allocation

In the BGMP/MASC architecture [4], address prefixes are assigned to domains so that groups in the same prefix are rooted at the same place. Thus, the interface towards the tree root is the same for large blocks of adjacent addresses. This can have a very beneficial effect on input filter state for unidirectional trees, since the incoming interface check can be aggregated.

For example, let us look at input filter state on point-to-point interfaces. For bidirectional trees, input filters are not needed, as explained earlier. For unidirectional trees using RPF checks, recall that Corollary 2 gave $E[R_{IF}] = (G_{rtr}/f)(1 - 1/f)$ for random allocation. When all G_{rtr} groups within a range of A addresses have the same incoming interface, then for that interface, $q = 0$ since all locally-active groups have $F(a) = 1$, and all unused groups have "don't cares". Since $q = 0$, the entire filter for that range is just $IF'(a) = 1$. For interfaces other than the incoming interface, $p = 0$, since no packets should be accepted, and hence the entire filter for that range is just $IF'(a) = 0$.

As a result, the amount of input filter state on a given interface is at most the number of routes pointing out that interface. When prefixes are assigned hierarchically, this scales as the *log* of the network size, regardless of the number of multicast groups.

Output Filter Simulations

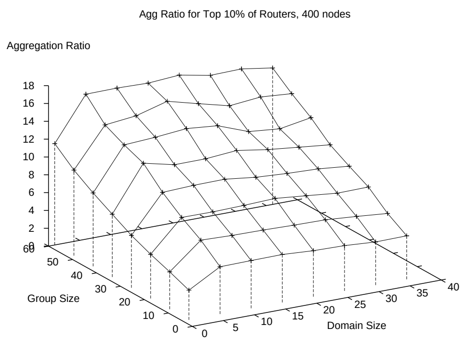


Fig. 3. Aggregation Ratio as a Function of Group and MASC Domain Size

To examine the effect of various aspects of non-random group membership on output filter state for bidirectional trees, we have written a special purpose simulator. We construct random network topologies which have a transit-stub like arrangement by allocating nodes a location on a rectangular grid, and linking each new node to the closest existing node. Additional long distance links are then added to turn this tree into a graph, with properties similar to those described in [20].

To examine the effect of MASC-like address allocation, we recursively subdivide the network into domains of connected routers, and allocate each domain a range of addresses. Multicast groups are then allocated randomly from the range of addresses given to the domain of the "group creator".

The effect of this aggregation can be seen in Figure 3, which shows the mean aggregation ratio as a function of the number of members in a group and the domain size. The network size is 400 nodes. Domain size is measured in routers, and is an upper bound - the subdivision algorithm attempts to allocate domains of between 33% and 100% of this. We show results only for the busiest 10% of the routers, as these "backbone" routers are the ones where we care most about aggregation. We allocate sufficient groups so that the mean number of groups *per router* is kept constant to avoid this weighting the results as the group size changes. In the graph, a domain size of 1 indicates a special case where we do not perform hierarchical allocation, but instead perform random allocation throughout the entire network.

This result shows that, as predicted, the aggregation ratio improves with group size (closely related to number of downstream members in Theorem 3) and that hierarchical allocation improves aggregation by between 50% and 100%. The size of a domain is not critical to aggregation.

C. Clustering of Receivers

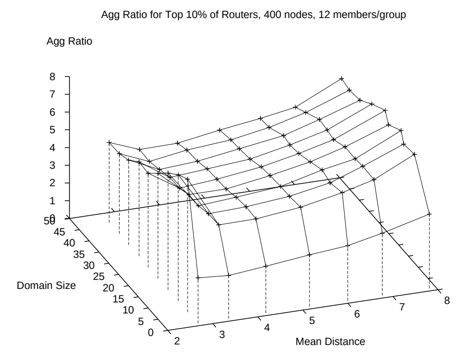
In the real Internet, receivers are not randomly distributed throughout the network, but instead they are clustered to some extent. This may be due to many issues ranging from timezones and language to belonging to a community of common interest that is correlated with network topology. In some cases, clustering is enforced by administrative scope boundaries[21], but in many cases it simply emerges.

To model clustering, we can separately examine two parts of the issue:

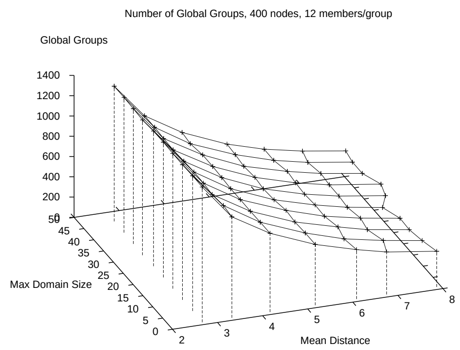
- Receivers clustering around the group creator.
- Receiver clusters scattered throughout the network, independent of the location of the group creator.

The former is the sort of clustering that timezone issues create, whereas the latter is more like the clustering that happens with communities of common interest.

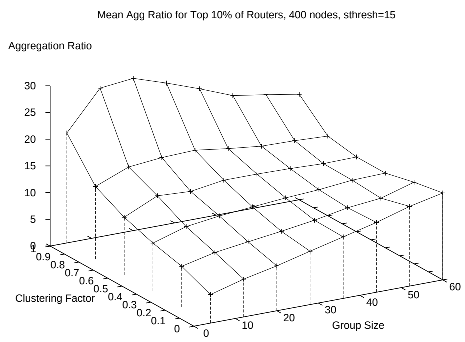
To examine receiver-creator clustering, we use the same simulator as before, but weight the random allocation of members so that nodes closer to the group creator are more likely to become group members. This is shown in figure 4(a) with 12 members per group. Larger group memberships increase the aggregation ratio but do not affect the shape of the curve.



(a) As a function of distance and domain size (backbone routers)



(b) Total number of groups in fig. 4(a)



(c) As a function of receiver clustering and group size

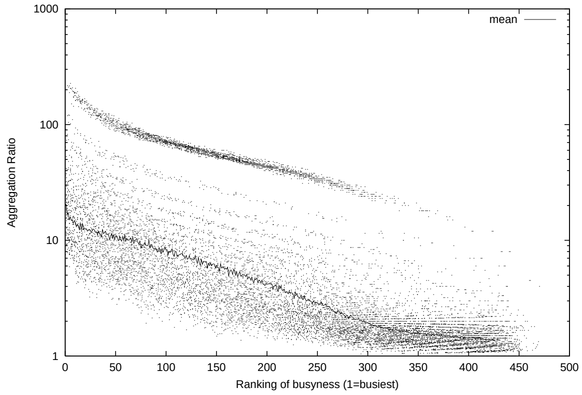
Fig. 4. Aggregation Ratio

This graph shows aggregation ratio as a function of mean distance from the sender and domain size, with the mean number of groups per interface kept constant at 80. The benefit of MASC-style allocation is still clear, but these simulations also indicate that clustering about the creator *decreases* aggregatability except when the members are clustered very closely around the group creator. It should be stressed that as receivers cluster around the creator, the number of groups the network can support for the same amount of state increases greatly. The total number of groups for the simulation in figure 4(a) is shown in figure 4(b). Thus although there's a slight decrease in aggregation ratio for the backbone routers, the network as a whole can support many more groups for the same amount of state when members are clustered in this way.

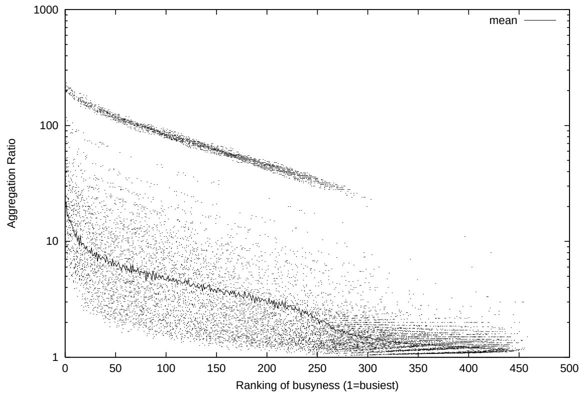
To examine scattered receiver clusters we must find a way to replicate the clustering that might happen in real networks. To do this we generate a two-dimensional array of randomly allocated affinities, such that *affinity*$[i][j]$ is used to weight the choice of the members domain, i, based on the group creator's domain, j. We then adjust a clustering parameter over a range of values from zero, meaning affinity has no effect on the random member allocation, to one, where affinity completely determines the receiver's domain.

Figure 4(c) shows the aggregation ratio as a function of group size and our clustering parameter, with domain size of 15. The effect of receiver clustering can be quite large, and produces a significant increase in aggregation ratio over purely random receiver placement. In general, the ability to aggregate improves as groups get denser, and receiver clustering results in relatively dense clusters for ranges of addresses, thus improving aggregation.

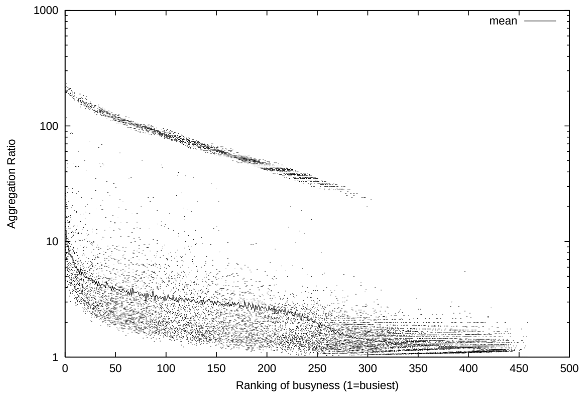
It is interesting to look at a scatter plot of aggregation ratio broken down by router interface. Figure 5(a) shows aggregation ratio against busyness ranking. In this ranking, 1 is the interface with the most groups joined through it, and higher values indicate successively less busy interfaces. The parameters here are a maximum domain size of 15 routers, no distance weighting, but an inter-member clustering factor of 0.5, 20 members per group, and a mean of 80 groups per interface. The mean aggregation ratio for each value of the busyness ranking is also shown. Leaf routers with only one interface are ignored, as always. The busiest "backbone" routers have by far the best aggregation ratio, which is what we would hope for. The dark band at the top consists of about one-sixth of the interfaces in this simulation that managed to aggregate all their groups into one range. Most routers are not so lucky, but a significant number of the busier routers manage to have fairly high aggregation ratios. Contrast this with the same graph generated with random address allocation and no clustering in figure 5(c), and hierarchical allocation without clustering in figure 5(b). The difference is small for leaf routers (high ranking), but significant for backbone routers (low ranking).



(a) Hierarchical Allocation with Clustering



(b) Hierarchical Allocation, No Clustering



(c) Random Allocation

Fig. 5. Aggregation Ratio vs Busyness Ranking

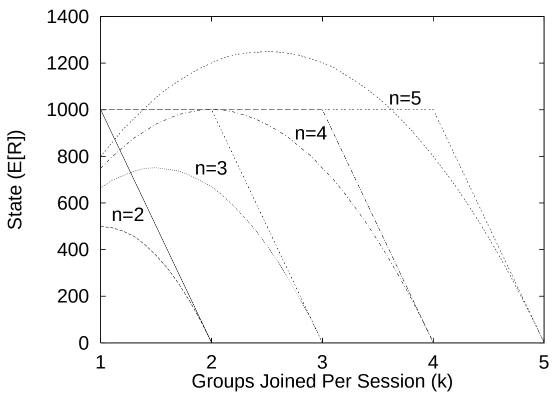


Fig. 6. Multi-Group Session Allocation

D. Multi-Group Sessions

We define a "layered" multi-group session as a set of related groups whose members join a non-empty subset of the groups, in a particular order. Thus a member typically does not join the $(i+1)$th group unless it is also a member of the first through ith groups. Some examples of such sessions include:

- multiple-media presentations where low-bandwidth receivers might only want audio while high bandwidth receivers might want both audio and video,
- layered codec schemes such as that employed by RLM [22], and
- reliable multicast mechanisms which use layered repair groups, such as to add additional levels of Forward Error Correction (FEC).

When groups in such a session use sequential addresses, a cluster of 1's followed by 0's may appear in filters. We ran a set of simulations to observe the effect of such clusters on interface state. Figure 6 shows the results, using 1000 sessions, and averaged over 100 trials.

The results of four simulations are shown, for session sizes (n) of 2 through 5 groups each. The vertical axis shows the number of aggregated ranges used, while the horizontal axis shows the average number of groups (k) per session with downstream members. Finally, this simulation assumes that any unused address space can be ignored, as explained in previous sections. Hence, these results apply to input filter state for point-to-point interfaces on a unidirectional tree, as well as to worst case output filter state for a given interface (where in the worst case, all groups have a downstream member on *some* interface).

For each cluster size, the straight lines indicate the state resulting from using sequential addresses within a multi-group session. The parabolic lines indicate the state resulting from using random addresses.

The key result of this simulation is that for layered sessions with four or fewer groups, less state is required if addresses are allocated randomly. For sessions with more than four groups, less state is required if addresses are allocated sequentially, since all groups in a session can be represented with one range.

The reason for the change at $n = 4$ can be easily explained mathematically. For $1 \leq k \leq n$, sequential addresses yield $E[R] = C$, where C is the number of sessions. For random addresses, on the other hand, Lemma 1 can be applied using $A = Cn$, $r = 0$, $q = 1 - p$. We obtain $E[R] = Cnp(1-p)$ which has a maximum of $Cn/4$ at $p = 1/2$. Hence, for $n \leq 4$, random is better. For higher values of n, sequential becomes better.

VI. Related Work

Tian and Neufeld [8] describe a scheme which avoids keeping any state with a single interface in the outgoing interface list, by encapsulating packets inside unicast packets between the upstream and downstream branching points (or terminii). The penalty paid is the CPU overhead of encapsulation and decapsulation, the extra bits on the wire, and the extra state for more logical interfaces at the tunnel endpoints.

Briscoe and Tatham [6] proposed a scheme for aggregation of multicast addresses, but would require changes to all hosts, routers, and applications, which may not be feasible. Since aggregation is done end-to-end, routing state is aggregated, and hence smaller forwarding state can be obtained. They allow addresses to be aggregated into ranges, but a range is not limited to covering changes in the least significant bits of an address. In this sense, their scheme is analogous to the use of non-contiguous masks in Kampai [23], with the same problem: a much greater burden in understanding and debugging problems. They also suggest that aggregation could be improved by having the session initiator specify in some way information about the likely receiver locations. While this is likely true, there is no proposed way to specify this, and it makes address allocation quite complicated.

A number of recent papers (e.g., [14], [15], [16], [17], [24], [25]) have explored alternative data structures for *unicast* forwarding state to provide fast lookups. Such work refutes the old belief that route lookups could not be done at gigabit and terabit speeds. Whereas most of them provide faster lookups at the expense of additional state, the Degermark, et.al. [15] method in particular improves performance by constructing very small forwarding tables, so as to provide a high memory cache hit rate.

Draves, et.al. [26] describe an algorithm which can be used to compress unicast forwarding state, and which can give an aggregation ratio of about 1.7 for unicast. Like our methods, it does not affect routing protocol state, and aggregation is performed when forwarding state is to be installed.

VII. Conclusions

In this paper, we examined the aggregatability of multicast forwarding state, and showed that significant potential for aggregation exists.

We first presented an interface-centric state model which is more amenable to aggregation than the traditional Unix model. We then analyzed its performance under purely random address allocation and purely random member placement, and showed that aggregation is possible by a factor of 4 in the worst case, and much higher in other cases.

We then simulated the effects on aggregation of various factors in the real Internet which either allocate addresses non-randomly, or which result in non-random member placement. We found that such factors can significantly reduce state, and showed their effects on aggregatability. In particular, noteworthy results include:

- MASC-style hierarchical address allocation can be used to reduce state requirements, and improve aggregatability by between 50 and 100%.
- Aggregatability is significantly higher when receivers are clustered due to interest. Aggregatability decreases when receivers are clustered around the creator. Both types of clustering still lower the amount of resulting state required, however.
- Aggregatability is greatest on the interfaces which are busiest, and hence need it the most.

Overall, we believe that state for busy interfaces could achieve an order of magnitude or more reduction using our techniques.

In this paper, we have only dealt with *perfect* aggregation; that is, aggregation which wastes no bandwidth. We leave the issue of trading a small amount bandwidth (for low-rate groups) for further aggregatability as future work.

References

[1] Estrin, Farinacci, Helmy, Thaler, Deering, Handley, Jacobson, Liu, Sharma, and Wei, "Protocol independent multicast-sparse mode (PIM-SM): Specification," June 1998, RFC-2362.

[2] Tony Ballardie, Paul Francis, and Jon Crowcroft, "An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, September 1993, pp. 85–95.

[3] Dino Farinacci, Yakov Rekhter, Peter Lothberg, Hank Kilmer, and Jeremy Hall, "Multicast source discovery protocol (MSDP)," *Internet Draft*, June 1998, `draft-farinacci-msdp-*.txt`.

[4] Satish Kumar, Pavlin Radoslavov, David Thaler, Cengiz Alaettinoglu, Deborah Estrin, and Mark Handley, "The MASC/BGMP architecture for inter-domain multicast routing," in *Proc. ACM SIGCOMM*, October 1998, pp. 93–104.

[5] Manolo Sola, Masataka Ohta, and Toshinori Maeno, "Scalability of internet multicast protocols," in *Proceedings of INET*, 1998.

[6] R. Briscoe and M. Tatham, "End to end aggregation of multicast addresses," *Internet Draft*, November 1997, `http://www.labs.bt.com/people/briscorj/projects/lsma/e2ama.html`.

[7] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson, "Scalable timers for soft state protocols," in *Proc. IEEE INFOCOM*, 1997.

[8] Jining Tian and Gerald Neufeld, "Forwarding state reduction for sparse mode multicast communication," in *Proceedings of the IEEE INFOCOM*, 1998.

[9] Stephen E. Deering and David R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 85–111, May 1990.

[10] John Moy, "Multicast routing extensions for OSPF," *Communications of the ACM*, vol. 37, no. 8, August 1994.

[11] Craig Partridge et.al., "A 50-Gb/s IP router," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, June 1998.

[12] Guru Parulkar, Douglas C. Schmidt, and Jonathan S. Turner, "IP/ATM: A strategy for integrating IP with ATM," in *Proc. ACM SIGCOMM*, August 1995.

[13] Ming-Huang Guo and Ruay-Shiung Chang, "Multicast ATM switches: Survey and performance evaluation," *Computer Communication Review*, vol. 28, no. 2, pp. 98–131, April 1998.

[14] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner, "Scalable high speed IP routing lookups," in *Proc. ACM SIGCOMM*, 1997.

[15] Mikael Degermark, Andrej Brodnik, Svante Carlsson, and Stephen Pink, "Small forwarding tables for fast routing lookups," in *Proc. ACM SIGCOMM*, September 1997, pp. 3–14.

[16] Pankaj Gupta, Steven Lin, and Nick McKeown, "Routing lookups in hardware at memory access speeds," in *Proceedings of IEEE INFOCOM*, 1998.

[17] Butler Lampson, V Srinivasan, and George Varghese, "IP lookups using multiway and multicolumn search," in *Proceedings of IEEE INFOCOM*, 1998.

[18] Steven Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ahmed Helmy, and Liming Wei, "Protocol independent multicast version 2, dense mode specification," *Internet Draft*, August 1998, `draft-ietf-pim-dm-spec-*.txt`.

[19] M. Handley, "Multicast address allocation protocol (AAP)," *Internet Draft*, June 1999, `draft-ietf-malloc-aap-*.txt`.

[20] M. Doar, "A better model for generating test networks," in *Proc. IEEE Global Telecommunications Conference/GLOBECOM'96*, November 1996.

[21] D. Meyer, "Administratively scoped IP multicast," July 1998, RFC-2365.

[22] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driver layered multicast," in *Proc. ACM SIGCOMM*, August 1996, pp. 117–130.

[23] Paul Tsuchiya, "Efficient and flexible hierarchical address assignment," in *INET '92*, June 1992, pp. 441–450.

[24] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *Proc. ACM SIGCOMM*, 1998.

[25] T.V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," in *Proc. ACM SIGCOMM*, 1998.

[26] Richard P. Draves, Christopher King, Srinivasan Venkatachary, and Brian D. Zill, "Constructing optimal IP routing tables," in *Proceedings of IEEE INFOCOM*, March 1999.