

Receiver-Based Multicast Scoping: A new cost-conscious join/leave paradigm*

George F. Riley Mostafa H. Ammar Lenitra M. Clay
Networking and Telecommunications Group
College of Computing Georgia Institute of Technology Atlanta, GA 30332
{riley,ammar,lenitra}@cc.gatech.edu

Abstract

In Internet multicast, the set of receivers can be dynamic with receivers joining and leaving a group asynchronously and without the knowledge of the source(s). The Internet today uses source-based scoping by which the source determines how far its multicast transmissions will propagate. This mechanism, however, does not allow a receiver to bound the number of hops added to a multicast routing tree as a result of its join request or its continued membership. In this paper we explore the use of receiver-based scoping, in which a receiver's join request is augmented with one or more scope values. By use of these values, the receiver specifies its wish to join and remain in the multicast group as long as the number of additional hops its membership incurs is within its declared scope. In environments where a multicast receiver incurs a cost in relation to its incremental resource usage, this scoping technique can be used to give the receiver direct control over this cost. The paper is concerned with discussion of the design of the receiver-based scoping mechanism and its possible use within multicast applications. We also consider the implementation of the mechanism as an extension to the existing Internet multicast infrastructure in a manner that is independent of the multicast routing protocol in use.

1. Introduction

In Internet multicast, the set of receivers can be dynamic with receivers joining and leaving a group asynchronously and without the knowledge of the source(s). Multicast packets are addressed to a *group address*. Receivers wishing to join the multicast group simply “listen” to the group’s address and inform a local designated router of this via the IGMP protocol [6, 10, 5]. A multicast routing protocol operating within the net-

work ensures the delivery of multicast packets to joining receivers.

Multicast scoping is one of the methods by which the propagation of multicast packets is controlled in such a dynamic environment. A typical scoping technique utilized in the Internet’s multicast routing protocols uses a hop counter carried in the IP header’s Time-to-Live (TTL) field. This counter is initialized to some non-zero value by the multicast source. Routers decrement this counter and forward the packet only if the decremented counter value is larger than an administratively configured threshold. Forwarded multicast packets in this case contain the decremented value of the counter. Another more recently proposed technique relies on *administrative scoping* rather than TTL scoping [14] with essentially the same effect of limiting the propagation of multicast packets to within a defined region around the source. Because the source cannot control which receivers join a multicast group, scoping is the only multicast propagation (and hence multicast session cost) control method available to the source.

The multicast scoping used in the Internet today is *source-based* with the burden being placed on the source to determine the appropriate scope to use for the multicast. In this paper we explore the use of *receiver-based scoping*. In our scheme a receiver’s join request is augmented with one or more scope values. By use of these values, the receiver specifies its wish to join and remain in the multicast group as long as the number of additional hops its membership incurs is within its declared scope. We target environments where a multicast receiver incurs a cost in relation to its incremental resource usage. Receiver-based scoping can be used to give the receiver direct control over this cost.

The paper is concerned with discussion of the design of the receiver-based scoping mechanism and its possible use within multicast applications. We also consider the implementation of the mechanism as an extension to the existing Internet multicast infrastructure.

Multicast scoping in the Internet works in conjunction with multicast routing protocols that route the multicast packet over a tree [3, 15, 7]. In modern multicast routing protocols, the tree grows and shrinks to span the dynamic set of receivers. A path to a new receiver is *grafted* onto the tree whereas an existing path in the tree is *pruned* once it is no longer used to reach receivers. There are two basic flavors of multicast

*This work is supported in part by research grants from Intel, IBM and NSF under contract number NCR96-28379

routing protocols, ones that use a shared tree to deliver multicast packets from all sources [3], and ones which use a separate routing tree per multicast source [15, 8]. Some routing protocols allow a hybrid approach [7]. All such protocols use some form of grafting and pruning of paths in order to accommodate dynamic group membership. Receiver-based scoping can be implemented by appropriate modifications to the various routing algorithms. An earlier version of this paper considered this approach, which has the disadvantage of requiring different receiver-based scoping mechanisms be implemented for different multicast routing protocols. Instead, we propose in this paper a multicast routing protocol-independent approach that derives some of its inspiration from work on RSVP. [18, 4]

The rest of the paper is organized as follows. Section 2 presents our proposed receiver-based scoping approach and discusses many of the issues associated with the approach. Section 3 discusses a multicast cost allocation strategy that can be used in conjunction with receiver-based scoping, and a set of other example applications where receiver-based scoping will prove beneficial. Section 4 describes our routing protocol-independent proposal for the implementation of receiver-based scoping within the Internet multicast infrastructure. Section 5 uses receiver join/leave behavior collected over the Mbone to illustrate the effect of receiver-based scoping on a real-life multicast session. The paper is concluded in Section 6.

2. Receiver-Based Scoping

Marginal Hop Count: In the Internet’s multicast environment each join request will have an associated *marginal hop count (MHC)*. We define this as the number of additional hops that would need to be grafted onto the multicast routing tree in order for multicast packets to reach the joining receiver. We also define the MHC for a receiver that is *currently a member of the group* as the number of hops on the path that would be pruned from the tree if the receiver were to leave. Note that the MHC will vary from host to host and will also vary over time. We count the connection from the receiver to its subnet’s designated router as one hop. Therefore, a joining receiver’s MHC will be zero if there already exists another receiver on the same subnet as the receiver issuing the join request.

A receiver’s MHC is the number of hops in the multicast tree that are *unshared*, i.e., they are being used exclusively for the delivery of multicast packets to this one receiver. (Of course, they may be shared by other data flows.)

Basic Description: In receiver-based scoping, each joining host declares a *join threshold, J* , and a *leave threshold, L* , along with the join request it issues, with $L \geq J$. These thresholds indicate that the receiver wishes to:

1. join the group as long as its MHC at the time of joining is less than its declared join threshold, and
2. remain a member of the group (and continue to receive multicast packets) as long as its MHC is less than or equal to its declared leave threshold.

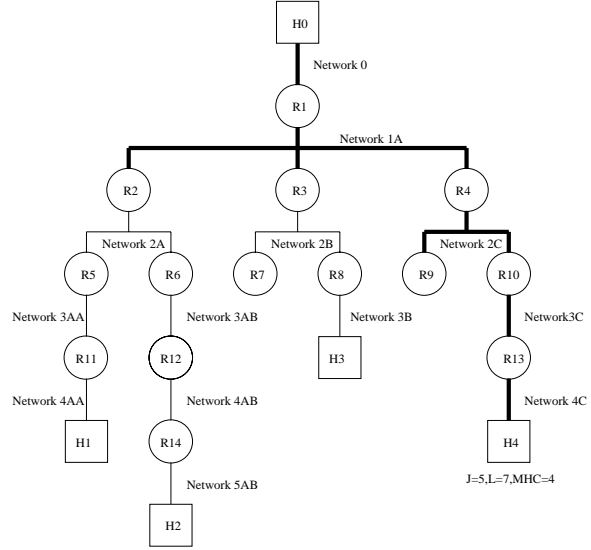


Figure 1. Sample Multicast Routing Tree

This action is known as a *Conditional Join*. Note that L needs to be larger than or equal to J to get the appropriate *hysteresis*. If $L < J$ then a receiver’s MHC can be less than its leave threshold immediately upon joining which will cause the undesired effect of requiring the receiver to leave the group immediately after a successful join.

Example: The following example illustrates the desired semantics of Receiver-Based Scoping. We discuss how it may be implemented in section 4. Consider the sample multicast routing tree shown in Figure 1 which assumes source-based multicast routing. We assume that we have a multicast source at host $H0$ and an existing receiver at host $H4$, who specified $J = 5$, $L = 7$ at the time of its initial join. Thus the multicast data is initially transmitted on Links 1A, 2C, and 4C, as indicated by the bold lines in the figure. Host $H4$ ’s MHC at this point in time is 4, and since it specified $L = 7$, it will not be forced to leave with the current routing configuration. Note also that we assume that the leaf network for the source (Network 0 in our example) will always contain the multicast data, and thus is not counted in the MHC calculations.

Now assume that host $H1$ initiates a join specifying $J = L = 2$. This join will not be successful, since the number of links on which the multicast data would be added as a result of this join is three, specifically links 2A, 3AA, and 4AA. Next, assume host $H2$ initiates a join specifying $J = L = 4$. This join will be successful, since we can indeed get the multicast data from the existing tree to host $H2$ by grafting a path of length 4, specifically links 2A, 3AB, 4AB, and 5AB. If host $H1$ now retries the original join request specifying $J = L = 2$, it will be successful since the length of the path to be grafted is now 2, specifically networks 3AA and 4AA. Let’s finally assume that Host $H3$ joins with $J = L = 7$. Of course this join will also be successful. Note that after any of host $H1$, $H2$, or $H3$ has successfully joined the group, the MHC for host $H4$ will change

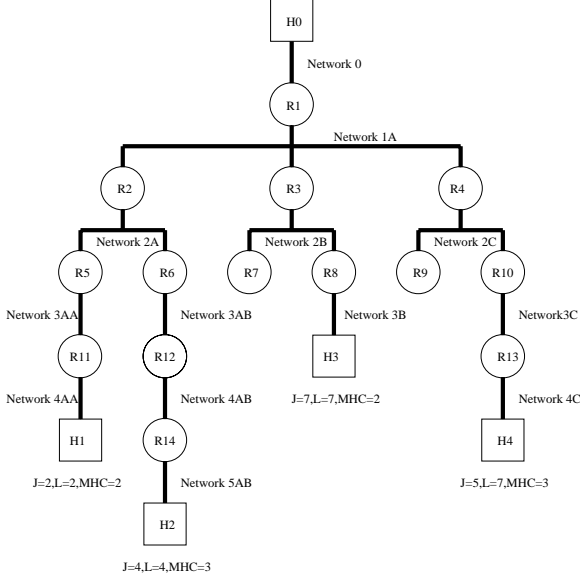


Figure 2. Sample Multicast Routing Tree after Joins

from 4 to 3, since Network 1A now becomes shared with other hosts and thus would not be pruned if H4 were to leave.

Leave Options: With receiver-based scoping receivers leave a multicast group in one of two ways:

1. A *voluntary leave* occurs when the application receiving multicast data at a host quits at the user's request.
2. A *forced leave* occurs when the receiver's membership in the group needs to be dropped because its MHC exceeded its declared leave threshold.

Continuing with the example above, and observing Figure 2, let us now assume that host $H4$ voluntarily leaves the multicast group. This will not cause any forced leaves, nor will it in fact change the MHC at all for $H1$, $H2$ or $H3$, since they all still share Network 1A in the routing tree. Similarly, if $H3$ were to voluntarily leave, the MHC values for $H1$ and $H2$ will still remain the same as they are still sharing Network 1A. Notice that neither $H1$ or $H2$ individually have sufficient leave thresholds to reach Network 1A, but jointly they are able to remain members since together they are sharing the cost of Network 1A.

Finally, if $H2$ were to leave voluntarily, this would cause the MHC for $H1$ to increase from 2 to 4, since the formerly shared links 1A and 2A would now become unshared, and thus would be pruned should $H1$ leave. In this scenario, $H1$ would receive notification that its leave threshold has been exceeded and it should be forced to leave the group. Clearly, allowing forced leaves is not suitable for all applications. We can, however, conceive of applications that can tolerate a forced leave as will be discussed in Section 3.2. Note that forced leaves can be disabled by declaring a value of $L = \infty$.

3. Using Receiver-Based Scoping

This section presents some areas where the application of Receiver-Based Scoping may provide some value for multicast users. Section 3.1 discusses the application of Receiver-Based Scoping in an environment where a receiver would like to have an upper bound on the costs incurred as a result of membership in a multicast group. Section 3.2 discusses the use of Receiver-Based Scoping in other applications.

3.1. Limiting Cost Allocation

One of the main purposes of receiver-based scoping is to give a receiver an assurance of some upper bound on the cost per unit time of membership in a given group. In order for this to occur we must provide two mechanisms, namely a specific cost allocation scheme, and a way for receivers to determine the effect of that scheme on the cost allocated to their join request.

Receiver-Based Scoping and Multicast Cost Allocation: A detailed analysis of how to allocate total costs of network services in a multicast environment is given by Herzog, Shenker, and Estrin in [11]. In this work, two basic cost allocation methodologies are proposed, Equal Tree Split (*ETS*) and Equal Link Split Downstream (*ELSD*). We propose a new methodology which is a modification of *ELSD* that we call Scoped Equal Link Split Downstream (*SELSD*). In *SELSD*, the typical case is that receivers share equally link costs only within their specified scope, either the join scope J , or the leave scope L , depending on the presence or absence of other receivers and their specified scopes. In unusual circumstances, a given receiver may be allocated a portion of a link cost outside of his specified scope.

The *SELSD* cost allocation policy is specified as follows. Let

- ℓ_i = Each link i in the routing tree from a source to a receiver.
- C_i = Cost of each ℓ_i
- RT_i = The set of all receivers downstream from ℓ_i (relative to the source)
- RL_i = The set of all receivers downstream from ℓ_i with sufficient leave thresholds L to reach ℓ_i
- RJ_i = The set of all receivers downstream from ℓ_i with sufficient join thresholds to reach ℓ_i

Clearly, $RJ_i \subseteq RL_i \subseteq RT_i$. With these definitions, there are three possibilities for allocating the cost of each link ℓ_i .

1. $|RJ_i| > 0$. In this case at least one receiver has a sufficient join threshold to reach this link, and thus the link costs are shared equally among all receivers with sufficient join thresholds, as follows: $\forall M_k \in RJ_i, \text{cost for } M_k = C_i/|RJ_i|$
2. $|RJ_i| = 0$ and $|RL_i| > 0$. In this case, no receiver has a sufficient join threshold to reach this link, but at least one receiver has a sufficient leave threshold to reach this link. Thus the link costs are shared equally among all receivers with sufficient leave thresholds to reach this link, as follows: $\forall M_k \in RL_i, \text{cost for } M_k = C_i/|RL_i|$

3. $|RJ_i| = 0$ and $|RL_i| = 0$. In this case, no receiver has either sufficient join or leave thresholds to reach this link. Thus we simply share the link costs equally among all downstream receivers, as follows: $\forall M_k \in RT_i$, cost for $M_k = C_i/|RT_i|$. This is exactly the ELSD method proposed in [11]. This case can only occur when two or more receivers have joined with smaller L thresholds than some third receiver, and that third receiver subsequently left the group, as described in the example given in the previous section. The two remaining receivers are not given a forced leave notification, since together they are sharing the links outside their L scope.

Note that in all three cases the complete cost of link ℓ_i is recovered, since in each case $1/n^{th}$ of the cost is allocated to each of n receivers. The method still provides strong incentive for receivers to specify smaller thresholds, since they are more likely to be given a “free ride” on one or more links.

The actual implementation of the *SELSD* cost allocation mechanism requires substantial knowledge about group membership by each router on the path from the source to the individual receivers. Specifically, each router needs not only the magnitude of the sets RT_i , RL_i , and RJ_i , but also the individual members M_k of each set, in order to perform the cost allocations in steps 1 through 3 above. In order to keep the implementation of our protocol as simple as possible and potentially scalable to large membership sets, we chose not to attempt to provide this detailed information to the routers with our protocol. However, the following describes a method for approximating this allocation, which allows multicast service providers to statistically recover all costs, and which allows receivers to still control their allocated share of these costs.

Bounding Cost by Choosing Appropriate J and L Values: Using the cost allocation method outlined in the previous section, we can now provide a method for a receiver to insure that its cost allocation for group membership does not exceed some bound. We start by assuming that a receiver R_k has some budget B_k that it does not want to exceed for a given multicast session, and that there will be some actual cost allocation A_k for receiver R_k associated with the session. For simplicity, we will assume all link costs are equal to one unit of cost.

Define PHC to be the physical hop count from the source to the receiver. Our protocol provides this information to a receiver as we shall describe later. By the cost allocation method previously given, the absolute worst case allocation is $A_k = L + (PHC - L)/2$. In other words, in the worst case the receiver will bear all of the cost of the links within L scope, and will bear half of the costs of the links outside L scope. Thus, we can calculate the largest L value that satisfies $A_k \leq B_k$ as $L \leq 2B_k - PHC$. If the calculated L is negative, then the receiver cannot safely join this group without potentially exceeding its budget. If the calculated L is greater than PHC , then the receiver can set L to PHC and thus will never receive a forced leave.

As shown above the L value (specified by the receiver) and the PHC (determined by network topol-

ogy) combine to provide the upper bound on allocated cost over time for a multicast session. Similarly, the J value contributes to a lower bound on the cost allocation. To insure a minimum cost allocation, a receiver should simply set J to the MHC value reported by our protocol (described later).

If routers were able to perform the cost allocations as described in the previous section, then all costs would be precisely allocated, and the provider would recover exactly all costs incurred. However, as previously mentioned, the actual implementation of this allocation method is difficult. From the perspective of the multicast service provider, one possibility is to simply allocate costs to all members as if it were the worst case, and allocate all members exactly $A_k = L + (PHC - L)/2$. Clearly, in the typical case with a large number of group members, this would result in total allocations greatly exceeding the actual costs incurred. The advantage of this method is that the allocation of costs can be determined with information known precisely at the time of the join, and a receiver could be informed at that time of the allocated costs.

Since allocating costs according to the formula $A_k = L + (PHC - L)/2$ will normally result in large over-allocation of costs, providers may wish to scale down the terms in the allocation formula by two constant factors, k_l and k_p . The resultant cost allocation to each receiver would then be $A_k = L/k_l + (PHC - L)/k_p$. In other words, providers could apply a priori knowledge of the expected size and distribution of group members, and allocate costs accordingly. With this allocation method there is a chance that not all costs will be recovered. By periodic adjustment of the k_l and k_p values a provider can insure that long term cost recovery matches incurred cost. This method is similar to that used by long distance telephone service providers in determining the cost per unit time billed to customers. Many customers share a single communication medium, and the costs are allocated based on the expected number of customers sharing at any point in time.

It would be simple to enhance our protocol to provide to the receivers the k_l and k_p values as determined by the provider. With this revised formula for A_k , and the known values for k_l and k_p , a receiver can compute $L \leq (k_l k_p B_k - k_l PHC)/(k_p - k_l)$ and again determine the correct L value to prevent a session membership cost from exceeding a budget.

3.2. Other Applications

In our previous research in multicast communication we have encountered many situations where the availability of receiver-based scoping would be helpful. These include:

The Cost-Controlled Monitor: As multicast gains popularity, monitoring the usage of the network by the multicast groups becomes an important management function. Several monitoring/management tools exist [17, 1] today. In some cases, a network manager is interested in monitoring multicast packets received on groups that already have members within a certain locality. Receiver-based scoping would allow this to

happen easily. This is an example where a forced leave may be desirable. The monitor may only be interested in being part of the group as long as there are other members within a certain administrative region. The monitor will then configure its join/leave thresholds to result in a forced leave once there are no group members within the region it is monitoring.

Layer Addition in Layered Video Multicast: With layered video multicast [13, 12] video is transmitted from a source to a set of heterogeneous receivers over multiple multicast streams. Each stream is sent to a different multicast address. One stream carries a must-receive base video layer and others carry one enhancement layer each. Receivers join the group on which the base layer is transmitted. Receivers then add layers if they have the processing and network capabilities and drop layers as congestion is experienced. Often adding a layer in such a scheme can have some adverse effects on other receivers or network paths. Receiver-based scoping can be beneficial in this case in that it will only allow a receiver to add a layer if it is already being received within a user-specified region.

Delivery Options with a Push/Pull Server: Push information delivery is gaining in appeal as an approach to scale information delivery in today’s Internet. One possible scenario would allow a combination of push and pull delivery at a server. The server would multicast frequently requested information repetitively and would in addition accept requests for information in the normal transaction-oriented way [2]. A client will then have the option of receiving information via multicast or making an explicit request from the server for unicast delivery. The choice of which option to use would depend to a large degree on the MHC for the receiver with respect to the multicast tree on which the desired information is being multicast. If the MHC is larger than a particular threshold (equivalent to the unicast delivery resource requirement) then the receiver should request the information via unicast. Our receiver-based scoping approach can be used as a mechanism for the client to test if its MHC is within the desired threshold.

4. Implementing Receiver-Based Scoping

In this section, we present the outline of an implementation of Receiver-Based Scoping. Complete details of this protocol can be found in [16]. We start with the simplest possible case, where a particular receiver only cares about resource requirements at the time of the join, and then build on that solution to give an implementation for the more complex case.

4.1. A Forced-Leave Disabled Implementation

Consider a receiver that wants to join a multicast session G_d , but only wants to do this if the current MHC for that receiver is less than or equal to some threshold J .¹ Furthermore, the receiver wants to

¹We do not define an exact method for a receiver to specify the J value, but expect some enhancement to the existing sockets API for `IP_ADD_MEMBERSHIP` would be adequate.

remain a member of group G_d after a successful join, regardless of subsequent changes in the value of his MHC , so it specifies $L = \infty$.

Overview: The idea behind our protocol is that each source of multicast data group G_d will periodically send an *advertisement* message, but will send this message addressed to a separate *Control Group* G_c . Any simple algorithm for determining the address of G_c from the address of G_d (perhaps simply adding or subtracting one) will suffice. Advertisement messages sent on group G_c will contain the same sender scope (TTL) as the actual data message sent on group G_d . Receivers who wish to conditionally join group G_d will first unconditionally join group G_c in order to find out the state of the routing tree for group G_d . At first glance, this method seems to defeat the purpose of conditional joins, since joining group G_c will produce a potentially large growth in the routing tree for group G_c . However, the total data rate on group G_c is extremely small, and thus the overall network load incurred by the large routing tree for G_c is minimal. These advertisement messages are conceptually similar to the *Path* messages as defined by *RSVP*[18] a path identical to the actual data of interest, but our advertisement messages themselves are smaller and less frequent than the data desired. Our advertisement messages are also similar in concept to the *advertisement* messages described in [9], in that they traverse the routing tree from source to destination gathering information about the state of the routing tree. There is obviously the additional router state overhead for group G_c , which is part of the overhead needed to allow multicast groups to use conditional joins.

The advertisement messages contain fields which track the total *physical hop count* (PHC) from a source to a receiver, the marginal hop count (MHC), and the parent node address (PNA), which identifies the parent node in the routing tree. At each node in the routing tree for G_c , the router will determine if it is presently forwarding data for group G_d on each interface, and report that information to each child node for that interface in the routing tree for group G_c . If data for G_d is being forwarded, there is no increase in the MHC , otherwise the MHC for all downstream hosts is increased by 1. The routers will also manage and report the PHC which is simply increased by 1 at each hop. The PHC value is not used in the simplest scenario, but we reference it later in the more complex case of forced-leaves enabled. As long as the routing tree for group G_d is a subtree of the tree for group G_c then each leaf in the routing tree for G_c will have accurate information regarding the state of the routing tree for G_d .

While the host H is trying a conditional join, it will watch for the advertisement messages on group G_c . When an advertisement message arrives, the host can simply compare the value of the join threshold J with the reported value of MHC , and if $J \geq MHC$ then H will proceed to join G_d (using standard IGMP message exchanges), otherwise a conditional join failure will be reported to the application. If no advertisement messages are received within a reasonable time period, the host can conclude that there are no senders for group G_d , or that it is out of the sender specified scope for

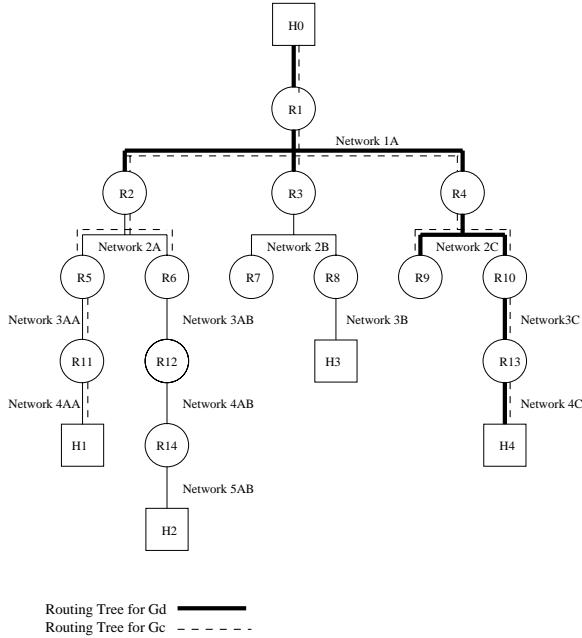


Figure 3. Simple Tree with Advertisements

group G_d , and in either case it can report a join failure.

Example: To illustrate the workings of the advertisement messages, consider the routing tree shown in Figure 3. In this example, the source for group G_d is host $H0$, and host $H4$ is presently a member of groups G_d and G_c . Host $H1$ is attempting a conditional join for group G_d , and thus has unconditionally joined group G_c , as shown. Host $H1$ has specified $J = 4$, $L = 4$, so we expect the conditional join to ultimately be successful, since the join can be satisfied by grafting only three branches to the routing tree, namely networks 2A, 3AA and 4AA.

First the sender $H0$ prepares an advertisement message with source address $H0$, data group address G_d , physical hop count (PHC) 0 and marginal hop count (MHC) 0, and multicasts that message to group G_c . Router $R1$ sees the advertisement and unconditionally increments PHC to 1. Since it is presently forwarding group G_d on Network 1A, it does *NOT* increment MHC . Then it routes the message on group G_c normally, which results in the message being forwarded on Network 1A, and received by routers $R2$, $R3$, and $R4$.

Router $R2$ will unconditionally increment PHC to 2. Since it is not forwarding group G_d it increments MHC to 1. Then it routes the message for group G_c normally, which results in the message being forwarded on Network 2A to routers $R5$ and $R6$. $R5$ will increment PHC to 3, increment MHC to 2 (since it is not forwarding G_d), and forward the message on Network 3AA. $R11$ does the same, sending $PHC = 4$, $MHC = 3$ on Network 4AA. Finally host $H1$ sees the message, indicating that the physical hop count (PHC) is 4 and the marginal hop count (MHC) is 3. Since host $H1$ is joining with $J \geq 3$, the conditional join will be successful. Router $R3$ will discard the advertisement message

since it is not forwarding data for group G_c .

Router $R4$ will unconditionally increment PHC to 2. Since it is forwarding group G_d it does not increment MHC (it remains at 0). Then it routes the message normally, which results in the message being forwarded on Network 2C to $R9$ and $R10$. Router $R10$ likewise increments PHC , leaves MHC alone, and forwards the message, as does $R13$. When the advertisement message gets to host $H4$, PHC is 4 and MHC is 0. If there were other hosts on Network 4C (with $H4$), they could satisfy a conditional join with $J = 0$, meaning they could receive the data for G_d for free since it is already present on that network.

4.2. A Forced-Leave Enabled Implementation

We now consider the implementation of a protocol to support conditional join requests that are augmented with the stipulation that the receiver also wants to leave the group should the MHC ever exceed some *leave threshold* L . We allow for the specification of $L = \infty$, which degenerates to the simpler case previously discussed.

At first glance, this would seem to be as simple as watching for advertisement messages and comparing the reported MHC value with the specified threshold L . Unfortunately, once a given host H has joined group G_d the reported MHC value will always be 0, since the routing tree for group G_d will of course include host H . The next sections outlines a protocol for advising hosts when the leave threshold has been exceeded, followed by an example.

Overview: Conceptually, what we need is for each router to determine two pieces of information:

1. The largest L value for any downstream host which is a member of group G_d , minus the number of hops from the host with that L value to this router.
2. Whether or not data forwarded on each output interface for group G_d is shared between at least two downstream hosts.

The existence of an unshared link in the routing tree (i.e. a link with only one downstream host) which is more than L hops from that host indicates that the leave threshold for that host has been exceeded. The way the routers determine this is by sending *Leave Threshold Report (LTR)* messages flowing back up the multicast tree towards the source, which specify the largest L value of any downstream host, minus the number of hops from that host to each router, and a flag (called the *shared link flag*) stating whether they have a single, or multiple, hosts downstream. The largest L value is simply the largest L value reported from children (minus 1). The shared link flag is set to *shared* if either (a) one or more child routers report a *shared* state link, or (b) more than one child reports a link with a single downstream host. The shared link flag is set to *single* if exactly one child router reports a link in *single* state, and no child routers report a *shared* link. This seems like it may result in a large number of messages, but as we shall demonstrate in the example, the messages can be aggregated at each router, such that

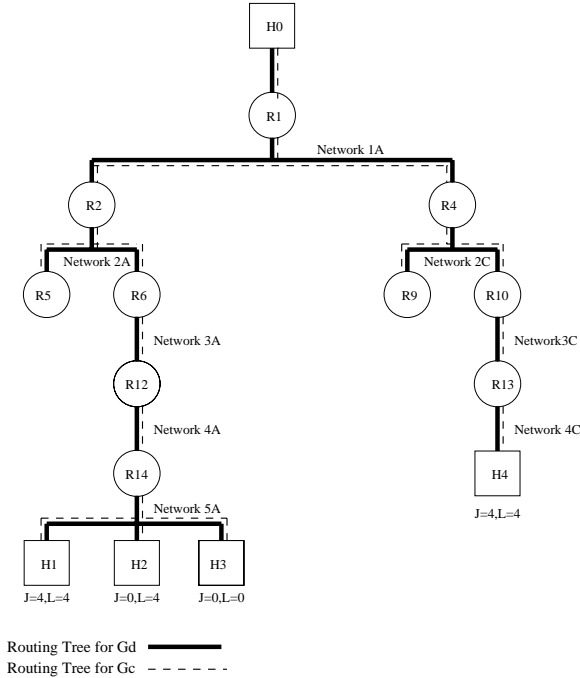


Figure 4. Simple Tree with Leaf Thresholds

any router at level k in the routing tree will only process n messages, where n is the number of routers at level $k + 1$ that are nodes in the routing tree for group G_d .

Once a router determines that there is an unshared link that is farther away from a given host than its L value will allow, it will send a *Forced Leave Notification (FLN)* message back down the multicast tree to inform the host that its leave threshold has been exceeded. Since the link is unshared, this implies that there is only a single host downstream on this link, so the router does not need to know the identity of the single host. The message is simply addressed to group G_c and forwarded by normal multicast routing for group G_c . The *FLN* message will ultimately be received and processed by that single host. A host receiving the *FLN* message will then leave group G_d using the usual IGMP mechanisms.

Example: Consider the sample multicast routing tree shown in Figure 4. In this example, the source for group G_d is host H_0 . Receivers H_4 , H_1 , H_2 and H_3 have joined group G_d in that order, with the J and L values shown.

Now consider what happens when host H_4 voluntarily leaves group G_d . The routing tree prunes off networks 2C, 3C, and 4C, but still contains networks 1A, 2A, 3A, 4A, and 5A. Each host H_1 , H_2 , and H_3 will respond to the advertisement message by indicating their respective L values (4, 4, and 0), and the shared link flag as *single*. Router R_{14} sees more than one *single* flag, and thus forwards the link state as *shared*, and also reports the largest L value minus 1 (3 in this case). Router R_{12} sees a single report message with $L = 3$ and link state *shared*, and forwards the message with

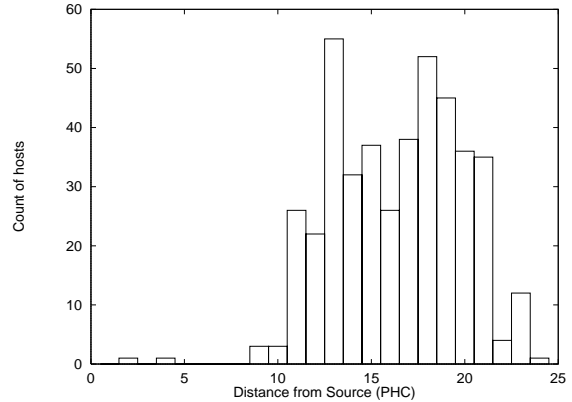


Figure 5. Physical Hop Count Distribution

$L = 2$ and link state *shared*. Router R_6 sees the single message, decrements L and reports $L = 1$ and link state *shared*. Router R_2 forwards $L = 0$ and link state *shared*. When router R_1 sees the $L = 0$, it can then determine if any downstream hosts are out of leave scope by simply examining the link state flag. If it is *shared*, then all downstream hosts are still within scope and no action is necessary. If it is *single*, and it has received no other report messages from other children on this interface, then that single host must be informed that it is out of scope.

Next assume that host H_3 leaves the group voluntarily. Router R_{14} still sees more than one report (two this time instead of the previous three), and everything proceeds identically as before.

Finally assume that host H_1 voluntarily leaves the group. This time, router R_{14} will see only a single report message, and thus will forward the link state as *single*. Routers R_{12} , R_6 , and R_2 do the same. When router R_1 sees the report message with $L = 0$ and link state as *single*, then it must inform the host (host H_2 in our example) that it has exceeded its leave threshold and should leave the group.

This section has presented our protocol in an overview format, with informal discussions of the working of the protocol. Complete details for a proposed implementation can be found in [16].

5 Effect on an Mbone Session

In this section we briefly explore the effect of receiver-based scoping and its parameters on the progress of an Mbone session. For this we use data collected using our Mlisten tool [1] which records various information about multicast group joins. For the analysis in this section we used the data collected over a few days for the Mbone multicast of the Space Shuttle Mission STS-80 in November 1996. This data contains information on 1,291 distinct joins from 429 different hosts.

We first built a static routing tree containing each of the set of recorded receivers, assuming that the source of the multicast session was at Georgia Tech and using

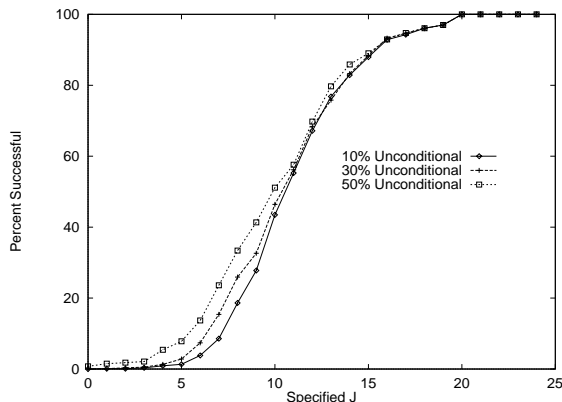


Figure 6. Percentage of Successful Joins

traceroute to determine shortest paths to each receiver in the log².

In order to put the potential J and L values into the proper perspective, we produced a histogram of the PHC values of all hosts which participated in the Mbone session during the monitoring period. Figure 5 shows the distribution of the physical hop count (PHC) values for these hosts. The PHC values range from a minimum of 2 (for a host on the local Georgia Tech subnetwork) up to a maximum of 24 (for a host in Japan).

We then set out to determine the effects of various J values on the join success rate in such an Mbone tree under the assumption that forced leaving is disabled (i.e., $L = \infty$). Clearly, for a multicast tree to exist at all, the first successful joiner must specify a J value equal to or greater than his distance to the source. Subsequent joiners may be able to successfully join with smaller J values, once a few branches of the tree exist. We assumed that some percentage of joins were made by persons willing to specify $J = \infty$ and $L = \infty$ (i.e., an *unconditional join*). We then determined the percentage of joins made by others that would have been successful for various J values. The results, shown in Figure 6, indicate that even though very few hosts are as close as 10 hops from the source, over 50% of the join requests would have been successful had they specified a J value of 10. More details of this analysis can be found in [16].

6. Concluding Remarks

The objective of this work is to introduce the concept of receiver-based scoping and discuss the issues relevant to its specification and implementation. We believe that receiver-based scoping is a powerful tool that can be used by receivers to control multicast reception cost in a direct fashion. Such control can help with the wider deployment of IP multicast as system

administrators are given control on the receiver side of multicast transmissions.

References

- [1] K. Almeroth and M. Ammar. Multicast group behavior in the internet's multicast backbone. In *IEEE Communications*, June 1997.
- [2] K. C. Almeroth, M. H. Ammar, and Z. Fei. Scalable delivery of web pages using best-effort (UDP) multicast. *Proceedings of Infocom 1998*, March 1998.
- [3] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT) an architecture for scalable multicast routing. In *Computer Communications Review, Proceedings of ACM SIGCOMM 93*, pages 85–95, September 1993.
- [4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP), September 1997. Internet RFC 2205.
- [5] B. Cain. Internet group management protocol, version 3. Internet Engineering Task Force, August 1994. Internet Draft.
- [6] S. Deering. Host extensions for IP multicasting. Network Working Group, August 1989. Internet RFC 1112.
- [7] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.G.Liu, and L. Wei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.
- [8] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, August 1994.
- [9] M. Faloutsos, A. Banerjee, and R. Rankaj. Qosmic: Quality of service sensitive multicast internet protocol. In *To appear in Proceedings of ACM SIGCOMM 98*, Sep 1998.
- [10] W. Fenner. Internet group management protocol, version 2. Internet Engineering Task Force, January 1997. Internet Draft.
- [11] S. Herzog, S. Shenker, and D. Estrin. Sharing the “cost” of multicast trees: An axiomatic analysis. In *Computer Communications Review, Proceedings of ACM SIGCOMM 95*, pages 315–327, August 1995.
- [12] X. Li, M. Ammar, S. Paul, and P. Pancha. Layered video multicast with retransmission (LVRM): Evaluation of error recovery schemes. In *Network and Operating System Support for Digital Audio and Video*, pages 171–182, May 1997.
- [13] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Computer Communications Review, Proceedings of ACM SIGCOMM 96*, pages 117–130, August 1996.
- [14] D. Meyer. Administratively scoped IP multicast. MBONED Working Group, August 1989. Internet Draft.
- [15] T. Pusateri. Distance vector multicast routing protocol. Internet Engineering Task Force, April 1997. Internet Draft.
- [16] G. Riley, M. Ammar, and L. Clay. Receiver based scoping, May 1998. Technical Report GIT-CC-97-28.
- [17] D. Thaler. Multicast debugging handbook. Internet Engineering Task Force, March 1997. Internet Draft.
- [18] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Network*, September 1993.

²Our results in [1] show that for large groups such as the one we are considering here, the placement of the source had little effect on the characteristics of the routing tree.