Hierarchical Protocol Independent Multicast (HPIM)

Mark Handley, Jon Crowcroft
University College London

Ian Wakeman
University of Sussex

11th Nov 1995

Note:

This paper assumes a degree of familiarity with the concept and terminology of Protocol Independent Multicast.

```
TODO

address
1/ compexity criticism
2/ simulation
3/ analysis
```

1 Introduction

In this paper we propose a modification to sparse mode PIM called Hierarchical PIM (HPIM). HPIM builds upon the work of PIM, but differs from PIM in a few ways. This most important way in which HPIM differs from PIM is that HPIM does not require advertisement of rendezvous points to the senders and receivers of a group. Instead of requiring group-to-RP mapping to be done at each border router, HPIM does this mapping in a series of steps within the HPIM router hierarchy.

2 Related Work

Multicast routing is a mature area of research in the Internet. The system that we now call the "Mbone"[?] has its roots in the research by Cheriton and Deering where they developed the Internet multicast model, including the idea of host groups[1] and the basic service model for IP multicast[?].

Two specific dynamic distributed routing algorithms were developed and implemented based on distance vector, truncated reverse path broadcast, later extended to include pruning of subtrees without receivers, namely the Distance Vector Multicast Routing Protocol (DVMRP)[3], and a link-state protocol, namely the Multicast extensions to the Open Shortest Path First (MOSPF) protocol[4].

Both of these algorithms result in a traffic distribution that has a high routing message overhead when groups are sparsely distributed or a high state overhead in the routers in manageing to limit traffic from

visiting unpopulated subtrees. They both build distribution trees from sender to recipients based on the simple, but elegant idea that the path from recipient to sender can simply be reversed. Thus for a multicast group with more than one sender, multiple trees are built, each (assuming symmetric delay or capacity/load links) with the shortest path (or possibly lowest delay) possible.

These trees may result in a low delay, but more links may be used than necessary. Some multicast applications only send from a single sender at any one time. For these applications, it may be more sensible to build a single tree for each group that has a single center (the "Steiner" center or "core" is the ideal).

Core Based Trees (CBT)[5] is a protocol that does exactly this. It has the advantage that it has far less routing state and far fewer routing messages than MOSPF and DVMRP respectively for the equivalent tree. It is highly suitable to sparsely distributed groups. CBT maintains its own topology information. One reason for this is to support multiple cores for fault tolerance to core failures.

However, this duplication of functionality was noted, and the Protocol Independent Multicast (PIM)[6] scheme was devised in response. PIM operates in two modes: dense and sparse. Dense-mode PIM is like DVMRP, except that it uses the underlying unicast routing of the Internet to provide it with the reverse paths[1] Sparse-mode PIM employs "rendezvous points" or RPs in the same manner as CBT employs "cores".

A key problem with Sparse-mode PIM and CBT is that the core or RP placement problem is the steiner tree problem[18], which is well known to be NP hard. A number of heuristics have been developed aimed at providing reasonable approximations to steiner trees. Some of these have been shown to take reasonable time[14][15] [16]. However, these, and the naive routing scheme devised by Doar[17] are aimed at virtual circuit networks, where the underlying model is one of a point-multipoint call fro ma source, and the receivers are added incrementally to the tree. There are two problems with such approaches in the Internet environment: distributed versions of such algorithms need to be devised; the cost of such algorithms may simply be too high to field on a network where routes may be calculated and recalculated *per packet*.

In a related area, the scaling problems for messages and router state present in DVMRP (and MOSPF) have been recognized and both have attempted to solve these problems by introducing a hierarchy of areas and designating which level (inter or intra area) a given router is[7]. This improves the scaling for dense group multicast, and is readily achievable using dense-mode PIM since the underlying unicast routing protocols may already implement a multi-level hierachy .

Finally, one group has applied similar ideas to these and introduced a hierarchy of cores into CBT, in the Core Group based Trees (CGBT) scheme[13]. This prompted the authors to attempt to improve the automatic configuration and scaling properties for Sparse Mode PIM, and the rest of this paper addresses our approach.

2.1 Assumptions and Intuition

In using an hierarchical approach to routing in a network, we reveal two linked assumptions about the tree-like meshes that form most modern network topologies: firstly, that most traffic is local; secondly, that the links which connect sites at a lower level togerther at a higher level of the tree typically have higher propagation delay since they go further.

Our intuition (hypothesis) is that sparse multicast groups will follow this traditional pattern of sender/receiver grouping, and that a hierarchy of cores or RPs allocated recursively to groups is likely to form a good match to traffic requirements in terms of compromising between link reutilisation and delay. We also hypothesise that this match will be relatively insensitive to poor placement of RPs in terms of location and level of the hierarchy.

[1] DVMRP runs its own versio nof the ROuting Information Protocol to caclulate forward paths - there are two reasons for this: firstly it meant that DVMRP could be deployed without having an agreed underlying unicast routing protocol;secondly, and closely related is the fac that DVMRP was deployed to build the Mbone as a virtual set of routers overlayed on non-multicast-speaking routers, using "tunnels" or IP in IP encapsulation to route multrivcaxt destined packets between so-called mrouteds that were not direct neighbours.

The goal of future work on simulation and analysis is to support these hypotheses. For now, suffice it to say that our intuition is based on an extension of the idea behind Doar's[17] work on naive multicast for the case of point-multipoint. By extension, the *any-to-many* service that IP multicast provides might well be routed by a hierarchy of rendezvous points that superficially are a poor approximation to steiner centers, but given typical underlying transmission line topologies, and user demography, may well be close to optimal.

3 General Idea and Problems

In Protocol Independent Multicast, a rendezvous-point (RP) is defined for each sparse mode group (along with backup RPs in case of failure of the primary). When an end-system wishes to join a multicast group, it issues an IGMP group-join. A PIM router known as the designated router (DR) on the same LAN as the end-system then sends a join message for that multicast group towards the RP for the group, and this instantiates state from the RP to the receiver along which traffic for that group can flow. If a sender wishes to send, the sender simply starts sending to the relevant group address. The DR on its LAN then sends a register message directly to the RP for the group, along with traffic destined for that group. This combination of sender registering and receiver joining creates a shared or RP tree along which data can flow from senders to receivers. At some later time, it is possible for routers on this shared tree to initiate a transfer to a shortest path tree, but that is outside the scope of this paper, which only deals with the formation of shared trees.

One problem with SM-PIM's construction of the RP tree is that the DR on a LAN must be able to lookup the address of the RP associated with the multicast group in order to be able to send register or join messages to the RP on behalf of its end-systems. This RP lookup mechanism must be able to work at gateways between multicast routing protocols in addition to just at end-systems.

Another potential problem with SM-PIM concerns how to get the traffic from a sender to the RP. Initially, a SM-PIM DR encapsulates a local source's data traffic and unicasts it directly to the RP for that group, and then the data is distributed along the shared tree from the RP. This means that, if the RP is not close to a sender, receivers close to that sender may suffer a much longer path (via the RP) than necessary. If SM-PIM detects a significant amount of traffic arriving from a source encapsulated, then the RP can make a source-specific join back to the source. This means that traffic no longer needs to flow all the way to the RP and back to reach receivers close to the source - the PIM router at the convergence of the source path to the RP and the receiver path to the RP can perform local fanout of this source's traffic to the close receivers and prune the source from the shared tree. However, this results in source-specific state being distributed between the RP and the source's DR. For many uses, the number of senders is small, so this overhead is not great. However, there are applications such as distributed simulation where the number of senders is large and those senders are transient, so keeping state in the network for each sender is undesirable.

Finally, a desirable property of shared trees would be to be able to dynamically migrate the RP for an active group depending on the traffic in that group in order to achieve a shorter shared tree when the original RP tree becomes significantly sub-optimal.

Hierarchical PIM should allow us to address all of these issues to an extent.

4 Hierarchical PIM

A candidate-RP router (C-RP) is a PIM router that is capable of being a rendez-vous point (RP) and is configured as being available to be an RP.

In SM-PIM, a candidate-RP router does not know that it is a candidate-RP router until it receives a register or join message from a DR.

In HPIM, we change this significantly - candidate-RP routers not only know they are candidate-RP routers, but they also are given a "level" in a global hierarchy. Every candidate RP router advertises its availability as a candidate-RP router to all the other candidate RP routers at its level in the hierarchy. By using a simple convergent protocol the candidate-RP routers at a particular level in a particular scope region form a consistent candidate RP list for their level. This candidate RP list is then communicated to the candidate RPs in the level below them in the hierarchy.

Typically there will be a small number of levels in the hierarchy (for example: UCL-CS, UCL, UK, Eu-north, Eu, west-hemisphere, world) and a relatively small number of candidate RP routers at each level in each scope area. Not all sites will have the same number of levels between their LAN and the world-level - two or more levels can be concatenated into one level within a layer of candidate RP routers.

Multicast addresses are allocated in bands which determine the scope of the session in a similar way to administrative scoping in DVMRP. Candidate RPs know that they are the border for one or more particular administrative scope address ranges. These address ranges can be autoconfigured as shown below.

Joining a group

When a DR receives a group join, it computes a hash of the multicast address, which gives an index into the candidate-RP list for level 1. It then sends a join message to the L1 RP, which acknowledges the join. If the multicast group is not in a border group that the L1 router knows about, then the L1 RP computes a hash of the multicast group address into the L2 C-RP list, and sends a join message to the resulting L2 RP which is acknowledged. Each RP in sequence joins to the RP in the level above chosen by hashing the multicast address into the relevant C-RP list until an RP is reached which is the border for that multicast group, at which point the join is complete.

If each RP is up and the join is not lost, there is no delay in joining other than propogation delay - the message is passed directly from one level to another until it reaches the maximum level for the group, or until it reaches a router that has already joined the group. If a join is lost, the join is retransmitted after a short timeout. Traffic will not flow from *all* sources until a join is successful in being relayed to the top level RP for that group. However traffic can start flowing from intermediate (more local) sources as soon as the join reaches an RP they are registered to.

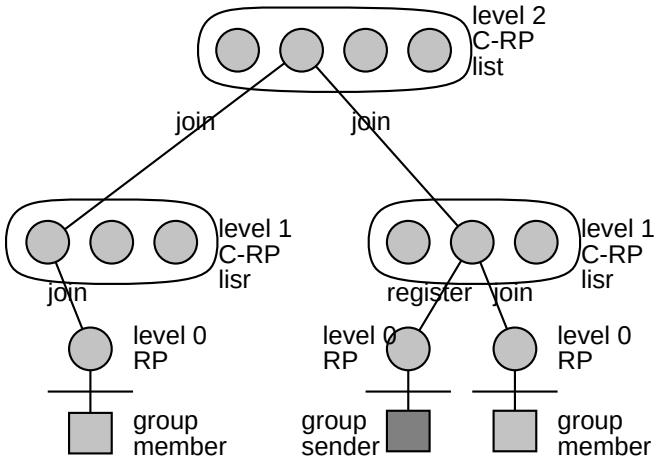


Figure 1: Formation of a Hierarchical PIM shared tree

If a join is not acked after a number of retransmissions, a secondary hash-function is tried and a secondary RP is chosen at the level above. Note that even if the primary RP for that group at that level subsequently recovers, the group is not segmented unless the level of the failure was at the maximum level for the group as the tree will re-join at the level above. If the failure was at the maximum level for the group, the

secondary RP knows about the failure because it can compute the same hash as the RPs in the level below, and therefore knows it is not the first choice primary RP for this group. As long as it has receivers, it can periodically send join messages to the dead primary RP. If one of these succeeds, the secondary RP can then register with the primary RP and start sending data to it. At this point (or slightly later to reduce routing flaps), it can also send RP-redirect messages to the RPs joined to it in the level below, and these RPs can now initiate a transfer of RP to the recovered primary RP by first sending a join to the primary RP, and if this is successful, then sending a leave to the secondary RP.

Sending to a group

When a new sender starts sending, it's local level-0 RP (SM-PIM DR) receives its traffic.

If a local receiver has already successfully joined, there will be shared tree state in place, and the source's data traffic is simply sent to the next hop router "for that state". Note: the state set up by join messages is bi-directional, and can be used equally well for sending or receiving - a router with more that two interfaces with shared tree state simply sends an incoming data packet out of all shared tree interfaces except the one the traffic arrived on.

If there is no local receiver the L0 RP computes the same hash from the group address as it would do for a join to calculate the location of the L1 RP. It then encapsulates the data packet in a HPIM register packet containing both the L0 and the L1 RP addresses and sends it to the first hop HPIM router towards the L1 RP. This router then decapsulates the register packet and re-encapsulates it again to send it to the next hop router towards the L1 RP. In addition, the router sends a hop-register-ack back to the previous PIM router, which then no longer needs to encapsulate the data. If a PIM router on the hop-by-hop path has active shared tree join state, then the register has reached the shared tree, and a register ACK can be sent back down the tree towards the previous RP. If an encapsulated register packet reaches the L1 RP, a register ACK is also sent back down the tree to the L0 RP. If there is no active shared tree join state at the L1 RP, and the L1 RP is not a scope boundary for this group, then the L1 RP computes a hash to calculate the L2 RP from the L2 candidate RP list, and begins encapsulating the data in an HPIM register packet to register to the L2 RP, as so on up the hierarchy.

Thus, if there are no joined receivers and no existing senders along the path from a new sender to the top level RP for the group, the first data will be hop-by-hop encapsulated all along the route. Except for encapsulation time, there is no additional delay as this data travels up the tree to the top level RP. As hop-register-acks travel back single hops, subsequent data packets for that group are no longer encapsulated. The register state instantiated along the route is per group, not per sender so can be used by any subsequent new senders. Unlike join state, shared tree register state is unidirectional shared tree state - it can be used be other senders to reach the next RP, but cannot be used by traffic proceeding down the tree *from* the next RP.

Register/Join loops

If RP placement is not optimal along a top-level RP rooted tree to all the senders and receivers, it is possible for register or join messages to be directed along a path that causes a loop. If this loop is allowed to remain, subsequent data traffic will loop continuously, which is extremely undesirable. An example of loop creation is given in figure 2.

Such a forwarding loop is easily broken, but to do so requires that join and register messages carry the level in the hierarchy of the RP generating the join or register message. If a router has active forwarding state for a group at a particular level on an interface it forwarded the join or register out of, and it receives a join or register for the same group at a higher level which should be forwarded out of a different interface, it must inactive its existing forwarding state before creating the new higher level state and passing the new join or register message on up the hierarchy.

Figure 3 gives such an example. A loop is potentially created as a register message from RP0 is propogated

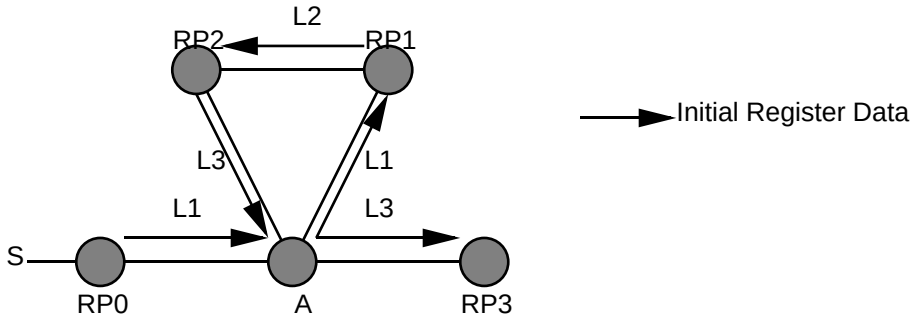


Figure 2: Potential creation of a forwarding loop in HPIM

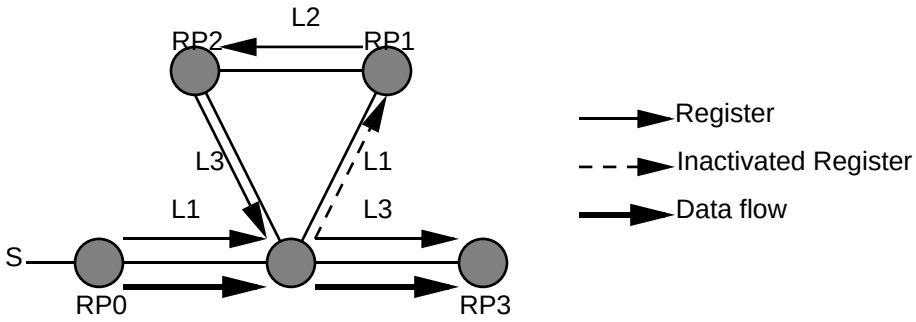


Figure 3: Prevention of a forwarding loop in HPIM

via A to RP1 and RP2, and then the register again traverses A on its way to RP3. Thus there is active forwarding state for G on A's RP1 interface when a register for G arrives at A from RP2 destined for RP3. Router A must never be in the situation where it has *created* active forwarding state for the same group on two different interfaces, so it invalidates its forwarding state on the interface created at the lower level (A's RP1 interface) and instantiates forwarding state on its RP3 interface instead. Register messages from S continue to be forwarded to RP1, but they are marked as "inactive". These continue to flow so that RP1 and RP2 continue to refresh the register state to RP3, although data traffic no longer takes this suboptimal route.

Short cut forwarding

An important point to note about HPIM is that a group's RP hierarchy tree is needed for the flow of control traffic (registers and joins) but it is not necessary for the actual data traffic to take this tree when the RP placement would lead to a very suboptimal topology.

An extreme example of very suboptimal RP placement is given in figure 4. In this example, encapsulated register traffic can only figure out where the next hop RP is when it reaches the previous RP, and this results in the initial encapsulated register data being routed across the same link several times. However, as figure 5 shows, using the "higest level register" rule discussed above, an improved tree is formed, and subsequent data from S is not routed to RP1 or RP3 unless a receiver joins via either of them.

Note that in this extreme case, during initial registering of each RP in turn, RP0 undergoes four reversals of which one of its interfaces (RP1 or RP2) has the active forwarding state. There are potential windows here during which a state reversal causes a data packet to be arriving on the interface it should now be forwarded out of. If this data packet is still register-encapsulated, this causes no problem as RP0 knows the next hop RP from the register data. If however, the data has ceased to be encapsulated, then RP0 has no choice but to drop the packet. This is a transient effect, and can be avoided at the expense of transient

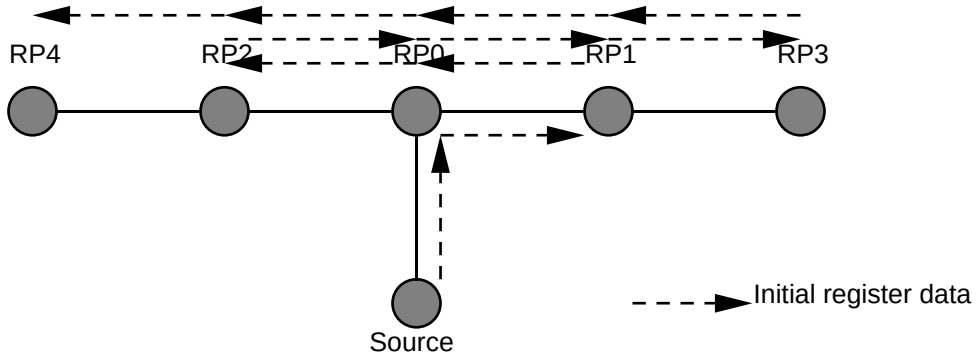


Figure 4: An extreme example of bad RP placement

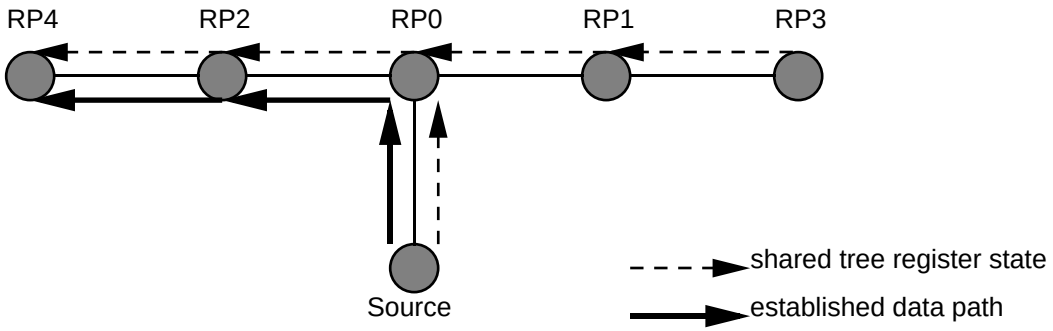


Figure 5: An extreme example of bad RP placement

traffic concentration and extra router load (performing encapsulation) if data traffic remains encapsulated for a short time after register acknowledgement.

Pruning of RPs

If a sender is sending data up a shared-tree register-path to the top level RP, there may be no receivers joined to that RP because there are no receivers in the group at all, or all the lower level RPs are joined on the same interface that all the data is arriving on (i.e., all receivers are receiving the data by a shared path from a lower point in the tree). In such a case, an RP can issue a shared-tree prune message following the register state back towards the source(s).

If there we no RPs joined to the top level RP, this prune message is conveyed all the way back to the source(s) L0 RPs and instead of sending data traffic up the tree, they merely ackowledge the prune message by sending periodic register messages back up the tree so the higher level RPs know there are senders available.

If there were RPs joined to the top level RP, but only on the same interface as all the incoming traffic, the RP can also send a shared-tree prune message down the tree towards the lower level RPs. Somewhere along the tree, the prune message will reach a router receiving joins on one interface and traffic on another - this is the lowest router that can be pruned, and on receipt of the prune, it will stop forwarding data traffic to the RP that issued that prune. Whilst the RP keeps issuing prunes, this pruned router keeps acknowledging them with register messages to keep the RP state alive.

There is also another case where RPs may prune themselves from the shared tree traffic - this is where all an RP's join messages from the level below arrive on the same interface, and its join messages to the RP in the level above also leave this interface. In this case the RP is actually a leaf in the shared tree, and thus it is not necessary for the flow of data traffic - only for the flow of control traffic, and it can be pruned off

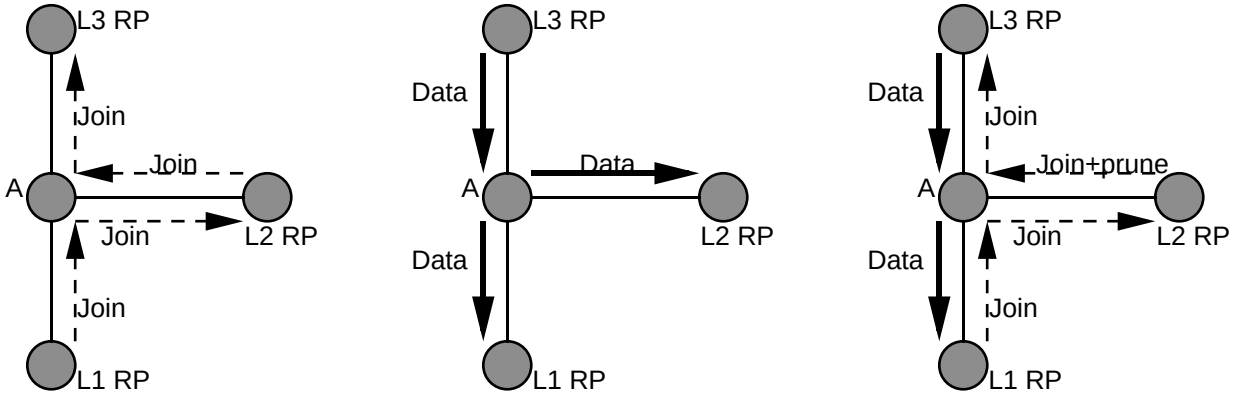


Figure 6: Pruning of a leaf intermediate RP

the distribution tree (see 6 for an example).

In both these cases, prune messages sent along links with join state effectively convert that join state into register state *from* the RP to the pruned shared tree. If a new sender registers to the pruned RP (from a different interface) there will therefore already be register state for its traffic to reach the pruned shared tree.

Note - pruning as discussed here is shared-tree prunes - as opposed to source-specific prunes used in SM-PIM.

4.1 Candidate-RP Lists

Candidate-RPs at level n receive each others' candidate-RP announcements, and build a candidate RP list, which they then distribute to the level n-1 routers. This candidate-RP-list is time stamped with a future time at which the C-RP list will be come active, and C-RP lists are only changed relatively infrequently (every 24 hours or so) although they are advertised by candidate-RPs at level n to candidate-RPs at level n-1 relatively frequently (say every few minutes).

Building C-RP Lists

Each C-RP router in a scope area at a particular level periodically multicasts its presence to the other C-RPs on a well known dense-mode address. On hearing another C-RP, a C-RP router adds that C-RP its potential candidate-RP list. This potential candidate-RP list is also periodically multicast to the level's DM address. Each list entry has a field indicating the status of each router. This status field can take one of tree values:

- **direct** - the candidate RP sending this list has heard directly from a C-RP marked direct.
- **indirect** - the candidate RP sending this list has not heard directly from a C-RP marked indirect.
- **poison** - there is a problem with the candidate RP concerned, and all C-RPs should poison its entry in their C-RP lists.

Each router's C-RP list itself also has a status field, with potential values of:

- **unstable** - the C-RP list is still changing.
- **stable** - the C-RP list has not changed for a significant time.

- **locked** - the C-RP list is now locked and can be used.

When a C-RP hears directly from another C-RP, it adds this other C-RP ti its potential C-RP list and marks it **direct**. When a C-RP receives a potential C-RP list with an entry marked **direct** or **indirect** status, it adds the entry to its own potential C-RP list and marks it **indirect**.

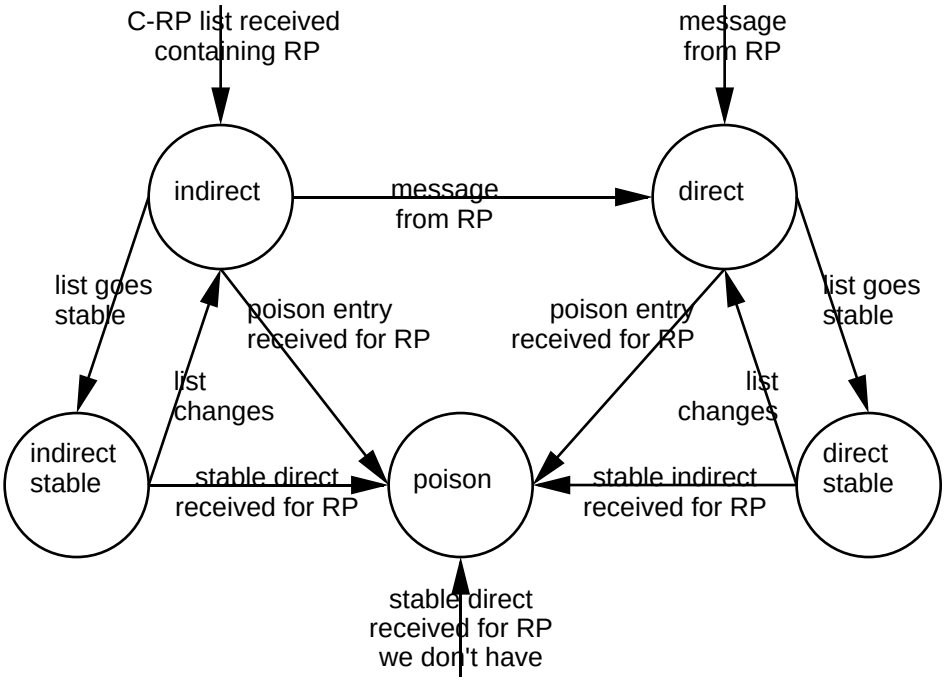


Figure 7: State Transistion Diagram for a potential C-RP list entry

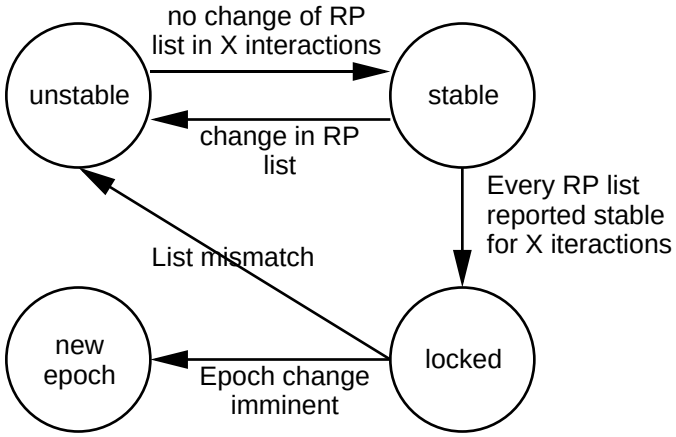


Figure 8: State Transistion Diagram for a potential C-RP list entry

After a number of iterations a C-RP will be receiving the same data from the other C-RPs every time. Once this situation occurs, an iteration counter is started, and after a fixed number of iterations during which the C-RP always gets the same data from the same C-RPs, it switches the status of its C-RP list from **unstable** to **stable**. Note that this doesn't indicate that all the C-RP lists it has seen are identical - just that they have not caused the C-RP's own potential C-RP list to change.

If a C-RP receives a C-RP list marked stable which differs in any way from its own stable C-RP list (the status of an entry is different, or it there is an entry missing from one of the lists) then the C-RP changes the status of the relevant entry in its own list to **poison**, and the status of its own C-RP list to unstable.

Any C-RP receiving a **poison** entry must set the equivalent entry in its own to **poison**. If that C-RP's potential C-RP list status was **stable** and the relevant entry was not already marked **poison**, then the status is reset to **unstable**.

Eventually all mismatches due to assymetric routing, black holes, etc will have been resolved unless there is a symmetric partitioning of the C-RP DM group. At the point where all C-RP's lists have been **stable** for a set number of interations and all list entries have either **direct** or **poison** status then a C-RP can set its list status to **locked**, and it is ready for communication to the level below's C-RP DM group.

If there is a symetric partitioning of the C-RP group, then a different C-RP list will be generated and locked in each partition. C-RP lists are changed for the RP level below on a clocked epoch boundary. If, at the time of the resolution of a partitioning, an epoch change is immiment or has passed, C-RPs simply form the union of the two (or more) C-RP lists without a renegotiation phase. If there is sufficient time before the epoch change, the C-RPs should switch to **unstable** state, and re-negotiate.

If there was an assymetric partitioning of the C-RP group, then one partition (the master partition) was being heard in the other partition (the slave partition). In this case, the standard negotiation process results in all the potential C-RPs in the slave partition being poisoned, and no inconsistency can result.

C-RP list modification

If a candidate RP stops advertising its C-RP list to the level below, it is marked by the C-RPs in its level and the level below as "down" but the C-RP list itself is not changed until the next C-RP epoch. The reason for marking a C-RP as down is to ensure that traffic sent to a new group (encapsulated register traffic) that would normally transit the down RP is sent to an alternative RP, and thus is not lost. If the alternative RP is the top level RP for the group and the primary RP has just recovered, this will be discovered and the encapsulated register will reach all the intended recipients as the alternative RP will attempt to register to the primary RP.

Similarly if a C-RP starts advertising itself, it is not added to the C-RP list until the next C-RP epoch. If a C-RP router discovers there are no C-RP routers in the level above (it has received no new C-RP list and the change of epoch is imminent) then the C-RP router can "upgrade" its level and join the dense mode C-RP group for the level above.

Automatic Configuration of Scope Boundaries

Scope address configuration is configured by an IGMP scope band definition sent from an (authenticated) end-system defining the multicast address range along with a defined "maximum level". Each candidate RP at levels below the "maximum level" then unicasts the scope band definition to a C-RP randomly chosen from the level above, which relays it to a C-RP from the level above it and so forth. When a C-RP at the maximum level receives the scope band definition, it multicasts it to the C-RP group for its level. The scope band definition is refreshed periodically from the relevant end-system, although each time it may take a different path.

4.2 RPs are not advertised

The most important difference between hierarchical PIM and PIM is that HPIM does not need RPs to be advertised. Each router in the tree makes a local decision about where the next hop RP is based on the multicast address and a candidate RP list which is synchronised across all RPs in the same scope at

the same level. This means that end-systems and application programs do not need to be involved in distribution of RP information to DRs and gateway routers.

This is of particular relevance for gateway systems between different routing paradigms which need to be able to discover the next level RP with no end-system assistance.

Note that the amount of state that must be held anywhere to maintain the candidate RP lists is independent of the number of multicast groups or the number of senders and receivers.

4.3 Choice of Top Level RP

Hierarchical PIM will create an RP based shared tree between all the senders and receivers in a group given a multicast address in the appropriate scope band for the group. However, by itself it does not guarantee that the top-level RP is in an appropriate location for the group. The higher the top level RP is in the hierarchy, the further it can be from its senders and receivers. Now it is possible to address this to a degree by adding more levels to the hierarchy, but there is an alternative solution.

HPIM RP "routing protocol"

When each candidate RP in level n multicasts its availability, it does so at a fixed TTL. The C-RPs in level n-1 all check the TTL field and store the distance to the sending C-RP. Alternatively the distance can come from the existing unicast routing tables (depending on the routing prorotol) so long as there is no routing border between the levels. When they advertise their availability to level n-2 they also present the C-RP list from level n along with the distances. The level n-2 routers pass average the level n-1 to level n paths for each level n router, and pass this on to the level n-3 routers along with their own distances from level n-1 routers. Thus a level 0 C-RP router will accumulate a distance table to all the C-RP routers in all the levels above. Typically their might be 5 levels with 10-20 C-RPs in each level, so this table is not too large.

When a session directory address allocator wishes to allocate an appropriate address, it requests the distance table for the maximum level for the scope it wishes to allocate. This information is supplied by its local router. If sd's address allocator knows the hash function in use, knows the C-RP list at the groups maximum level and the distances of those C-RPs, then it can choose a multicast address that selects a close RP. In addition, as sd knows all the current sessions advertised, it also knows which top level RP they will use, so if it looks like an RP might be overloaded, it can choose an alternative close by top level RP. If important, it can choose a multicast address with a close-by secondary RP in case of failure with the primary RP.

Note that even though the shared tree will not continue to be used for delay-sensitive traffic (which will switch to the shortest path tree) it is worthwhile trying to ensure that an appropriate top level RP is chosen.

4.4 Register Message Aggregation

Because register messages are passed and acknowledged level-by-level, there is no need for the top-level RP to receive all the register messages from all DRs. If an RP is already registered to the RP for that group in the level above, it does not need to send any additional register message for a new source. This limits any register message implosion that might occur in various applications such as distributed simulation which have large numbers of relatively inactive sources which may be synchronised in their change of multicast group.

4.5 RP migration

One desirable feature of a shared tree algorithm is the ability to change top level RP dependent on the traffic flows in the group. As the number of routers joined or registered to a top level RP is small with HPIM, it becomes reasonable to send them RP redirect messages to ask them to change to another better placed RP.

Deciding that the current RP at the top level is not optimal can be performed by checking the TTL fields in the join messages from the level below (these can be sent with a fixed TTL, so the remaining TTL indicates the distance). Alternatively, this distance may be available from the unicast routing tables. If the median distance is greater than a pre-set threshold and the group is significantly active, a search for a better RP can be begun.

Choosing the best top-level RP can be done in a number of ways. The optimal RP can be found by the current RP polling the RPs that are joined to it in the level below, requesting their distance from all the top-level RPs. The optimal RP can then be chosen using one of a number of criteria (note this much simpler than the Steiner tree problem).

Once a better RP is chosen, the original primary RP simply joins and registers to it. The new RP knows what is happening as it can see (from the hash function) that the primary RP for a group is attempting to join to it, so it won't attempt to join back to the primary RP as it would in the backup RP scenario. Once this join and register has occurred, the primary RP issues RP-direct messages to all the RPs in the level below that have registered to it asking them to change RPs. Any new RPs from the level below attempting to join the original RP are redirected to the new RP so long as the new RP keeps acking the original RP's join messages. Traffic from a new RP in the level below that is attempting to register is forwarded to the new top-level RP although its register message will be replied to with an RP-redirect message. Note that an RP that is registered to the original RP first attempts to register to the new RP before sending its traffic there. It does not ever send its traffic to both top level RPs. Being joined to both top level RPs merely means that the traffic will reach an RP in the level below via both paths and one set of traffic should be discarded until a leave message has reached the original top-level RP.

5 Simulation or Analysis

In this section, we follow the analysis in Wei's work[19]. HOwever, we are interested in Shared Path Trees, and Member rooted shared path trees, and hierarchical shared path trees only, and not in source rooted shortest path trees.

We note that all Internet Multicast sparse/centered tree placements are a different problem from the steiner tree pt-mpt problem, since

1. the group and sender locations are unknown to the network, and to each other(and change)
2. the sender(s) may not be in the receiver group!

We have two types of trees that data traverses from sender to receviers:

```
up-trees from sources to RPs
```

```
down-trees from RPs to group
```

We have two distributions that are input into the analysis:

```
clustering function for groups (which is a function of the scope
```

and size of the group).

sender distribution (scope w.r.t group)

we want to determine two costs as a function of number of levels of the hierarchy (where the base case is 1 level, and is simple single core CBT or single RP PIM-SM):

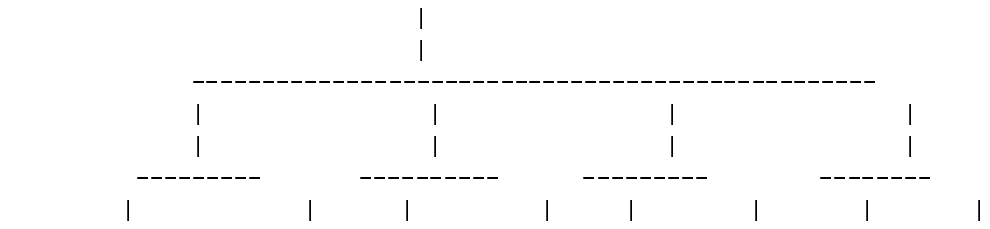
average expected delay cost of a HPIM tree

average expected link cost of an HPIM tree

the delay and link costs are functions of the uptree and down tree.

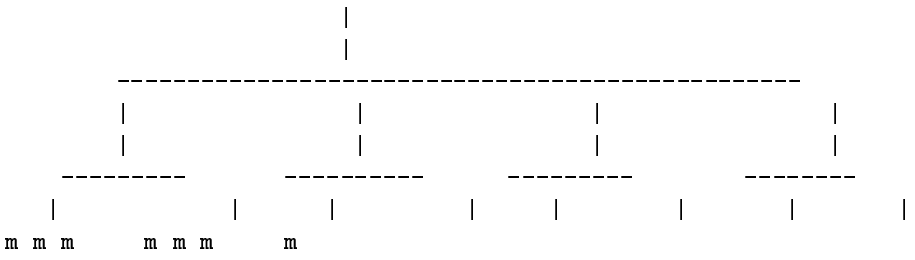
Our approach is to assume a perfect placement of a set of RPs and vary the group and sender placement (rather than trying to find a perfect tree for an arbitrary graph and sender/recipient locations!)

We have a tree of RPs placed in some sense optimally, if the groups were perfectly located with respect to an hierarchical unicast network.



we place a sender and a set of recipients on this tree - for a given scope and size, we have a simple probablilty function that says whether a member is in the right place w.r.t to the relevant RP tree.

Intuitively, if a member is plaecd on the wrong branch, then the delay and link costs are increased



Thus here, the errant member (whether sender or receive, incurs an additional cost of 1 level up and 1 level own the tree to deliver them data. This is tantamount to assuming that the link costs and delays, as you approach the backbone (of the backbone (of the backbone)) increase exponentially This is a reasonable assumption right now in the mbone (to be determiend from mtract and current measured topology!)

6 HPIM and RSVP and ATM

Here we present a novel idea for virtual RP inside ATM Cloud. Basically, ATM supports point-multipoint, which is inherently different from multicast i nthat the topology of a tree is implicitly tied to the idea of tightly bound single 1 source and destination set. However, we can map this to IP multicast if we know what kind of multicast - i.e. dense mode of sparse mode

If dense mode, then we have a forest of point-multipoint trees in the ATM view, 1 from each source to the sink set. (i.e. the source is at the 'edge' of the ATM cloud)

If sparse mode, then we can use the center of the tree as the source of a point multipoint distribution, and have sources send unicast to the center, and the center multicast from there (i.e. the source in the 'middle' of the ATM cloud) But, we have an IP layer and an ATM layer For the dense case, this is not a problem. So, we need to reveal something of the ATM topology to the IP routers so that they can find the center of the transmission network (whether it is inside or outside the atm cloud...)

[2]

Where does HPIM come in? Consider a hierarchy of RPs, matching the hierarchy of administrative scopes: if this is extended within the ATM cloud, it would work nicely. Some new feature of PNNI to allow you to get at the internal topology, then MARS is enhanced to let you choose sparse of dense mode, and the dense mode atm pt-mpt call source points, or sparse mode pt-mpt call center switch are found.

Then, you can use a PVC like this for RSVP messages . PATH messages simply get sent to the multicast address which maps to the pt-mpt call appropriately: When UNI 3 is deployed, then a receiver doing IGMP does MARS to the right tree, and starts getting path messages and can send RESV on the call to the MARS server, though you could improve on this. When you have UNI4, the receiver doing IGMP join simply does leaf join, and things proceed similarly ....

Then if you have to enhance the MARS servers to do new calls to actually carry the data. However, if the MARS (Resource Reservation Servers) are at HPIM RP points, they will be very well placed to do this!

7 Discussion of RP discovery

There are a few other areas for future work that are also needed - the choice of hash function for mapping from group and level/scope to appropriate RP needs to be explored further. The decision when to switch from hierarchical sparse mode to dense mode needs further work. And the hierarchy maintenance protocol needs some analysis for correctness (loop-freeness etc).

8 Acknowledgements

We acknowledge the discussion with members of the IDMR working group of the IETF.

References

[1] IGMP Steve Deering Multicast Routing in Internetworks and Extended LANs, ACM SIGCOMM 88, August 1988, pp 55-64 and Host Extensions for IP Multicasting, RFC 1112

[2] MARS is pretty reasonable for small scale - for example, host to IP/ATM - i.e. a sort of IGMP to ATM pt-mpt gateway - we can use this for the IP/ATM host to IP/ATM router - we could view it also as appropriate, if modified to be aware of the sparse/dense mode split - e.g. by IP and ATM address spaces, for finding appropriate IP/ATM cloud edge routers, or RPs ....)

[2] S.Deering and D.Cheriton. "Multicast routing in datagram internetworks and extended LANs." ACM Transactions on Computer Systems, pages 85–111, May 1990.

[3] DVMRP RFC 1075 S. Deering, C. Partridge, D. Waitzman, "Distance Vector Multicast Routing Protocol", 11/01/1988.

[4] MOSPF RFC 1584 J. Moy, "Multicast Extensions to OSPF", 03/24/1994.

[5] CBT An Architecture for Scalable Inter-Domain Multicast Routing, A.Ballardie, P.Francis, J.Crowcroft ACM SIGCOMM 1993, pp 85-95

[6] PIM An Architecture for Wide Area Multicast Routing S.Deering, D.Estrin, D.Farinacci, V.Jacobson, C-G.Liu, L.Wei ACM SIGCOMM 1994, London October 1994, ACM CCR Vol 24, No. 4, 126-135

[7] HDVMRP Hierarchical Distance Vector Multicast Routign for the Mbone Ajit, S. Thyagarajan, S.Deering ACM SIGCOMM 1995, Cambridge, Mass, 1995, ACM CCR Vol 25, No 4, pp 60-67.

[8] S. Floyd, V. Jacobson, S. McCanne, C-G. Liu, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Proc ACM SIGCOMM 1995, Cambridge, Mass.

[9] J. Bolot, I.Wakeman, T.Turletti, "Scalable feedback control for multicast video distribution in the Internet", Proc ACM SIGCOMM 1994, London,UK

[10] D.D. Clark, D.L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", Proc ACM SIGCOMM 1990, Philadelphia, Pennsylvania

[11] M.R. Macedonia, , D.P. Brutzman, "Mbone Provides Audio and Video Across the Internet", IEEE Computer, Vol.27 No.4, April 1994, pp. 30-36.

[12] RTP H. Schulzrinne, S.Casner, R.Frederick and V.Jacobson RTP: A Transport Protocol for Real-Time Applications Internet Draft draft-ietf-avt-rtp-07.txt, Work In Progress, Late 1995.

[13] CGBT Multiparty Videoconferencing using IP Multicast Y.-C Chang, Z.-Y.Shae, .H.Willebeek-LeMair Multimedia COmputing and Networking 1996

[14] WATATM Multicast Provision for High Speed Networks IFIP High Performance '92 in Danthine, A., Spaniol, O

[15] WATHEU A New Heuristic for ATM Multicast Routing 2nd IFIP Conference on Performance Modelling 4-7 Jul, 1994, Bradford

[16] KATM Minimising Packet Copies in Multicast Routing bt Exploiting Geographic Spread ACM CCR, Vol 24, No. 3, July 1994, pp 47-62

[17] NAIVE How Bad is Naive Multicast Routing Proc IEEE Infocomm 1993

[18] STEI S.E.Dreyfus and R.A.Wagner The Stenier Problem in Graphs Networks 1, pp 195-207, 1972

[19] wei A Comparison of Multicast Trees and Algoriths Liming Wei and Deborah Estrin INFOCOM 94