The HIP Protocol for Hierarchical Multicast Routing

Clay Shields J.J. Garcia-Luna-Aceves

{clay,jj}@cse.ucsc.edu
Computer Engineering Department
School of Engineering
University of California — Santa Cruz
Santa Cruz, CA 95064

Abstract

This paper presents a new, flexible approach to inter-domain multicast routing. The HIP protocol introduces the idea of "virtual routers" as a method of organizing the control of an entire domain so as to appear as a single router on a higher-level shared tree. HIP then routes between domains using the Ordered Core Based Tree (OCBT) protocol, and allows recursive application of virtual routers to create a multi-leveled hierarchy while providing efficient methods of distributing the location of the center point for the tree. The advantages of this approach include improved robustness, the ability to route between heterogeneous domains, and a significant reduction in the amount of bandwidth consumed in control traffic and in the amount of state stored at each router over existing hierarchical multicast protocols.

1 Introduction

There are two compelling reasons for developing hierarchical multicast routing protocols. The first is to provide a means of routing between heterogeneous domains that might use any multicast protocol internally. The second, and perhaps more pressing, is that the protocol currently being used for multicast routing on the Internet, the Distance Vector Multicast Routing Protocol (DVMRP) [8], is unable to scale with the exponential growth of the Internet. DVMRP computes and maintains its own unicast routing table between DVMRP capable routers; as the number of multicast capable subnets has grown exponentially, so has the size of the table maintained at each router. This table is rapidly growing too large to be easily stored and, unless some solution is found, it will become increasingly difficult for any single host to maintain the routing information it needs to function properly. A well known solution to this problem lies in reducing the amount of information that needs to be stored by routing *hierarchically*. The goal of a hierarchical multicast protocol is to build a single distribution tree across all receivers, but to do it in such a way that the control information regarding the tree is limited to a particular domain or level. A hierarchical scheme can also allow for heterogeneity of multicast protocols at different levels.

There have been two general ways proposed to build a hierarchical multicast tree: a tree of trees, in which the leaf nodes of a higher-level tree can each be the root node of a lower-level tree; or trees within trees, in which a node of a higher-level tree actually contains a lower-level tree. The first approach was one of the motivations for creating the Ordered Core based Tree protocol (OCBT) [10]. OCBT is a *hard-state* protocol, so once a branch of the tree is constructed, it is not removed until a link failure occurs on the branch or all receivers lower on a branch quit from the tree; this reduces the control traffic required to construct and maintain the tree. Additionally, OCBT uses the existing unicast routing table to make its routing decisions; routers do not need to maintain a separate table to locate other multicast routers. OCBT functions similarly to the Core Based Tree protocol (CBT) [1] in that a router wishing to join the tree sends a join request towards the closest core. When the join request reaches that core or an on-tree router, the receiving router generates a join acknowledgment that traverses the reverse path of the join request and instantiates that branch of the tree. In OCBT, every core and on-tree router also maintains an integer *logical level*. The level of each core is fixed and the level of the router is set by the returned join acknowledgment, which is marked with the level of the core or on-tree router that was reached. When a lower-level core receives a join request and it is not already part of the multicast tree, it must join to a higher-level core, and does so in the same way by sending a join request towards the next highest core. The join request is marked with the level of the core for which it is intended; if it reaches a branch of the tree of that level or higher, then the join acknowledgment is marked with that level and builds a branch of that level back to the sender. If the request reaches a lower-level branch, that branch breaks to allow formation of the higher-level branch. It is this labeling and breaking mechanism that ensures that OCBT remains loop free.

If it were possible to define each logical level of core as an actual level of a hierarchy, it would be an simple task to create such a tree. However, both OCBT and HPIM [7], a similar construction for PIM [2, 3], suffer in hierarchical application from the fact that it is very difficult to come up with a hierarchical placement of cores or RPs without extensive knowledge of the network topography and the receiver set. This type of hierarchy also suffers from the fact that it is homogeneous; it does not easily provide for multicast protocols or domains of other than OCBT or PIM, except as leaf nodes of the tree.

Most hierarchical proposals therefore use the second

method and construct a tree from domains that contain trees themselves. Under this type of hierarchical scheme, a single flat routing *region* is divided into several non-overlapping *domains*, each of which runs its own internal routing protocol. These domains are connected and packets are routed between them by another routing protocol at the higher level. Each member now routes packets only to subnets in its domain, and to reach subnets in other domains each member sends packets to the interface of the higher level of the hierarchy, which forwards the packets to the appropriate domains. The end result is that each member only has to keep routes to the subnets in its domain, resulting in smaller routing tables and lower overhead. This method still creates a single spanning tree; at the highest level it appears as a tree of domains.

Hierarchical DVMRP (HDVMRP) [12] divides the current flat routing region into disparate parts and assigns each a unique name. DVMRP is then used within each routing domain for *intra-domain* routing and between each region for *inter-domain* routing. Data packets travel normally within a domain, but are encapsulated for routing between domains to prevent the packet from being processed as a native data packet, which could result in duplicate packets being delivered to hosts. However, the encapsulated packets might traverse the same links as unencapsulated packets; this is a weakness because every multicast protocol tries to prevent data packet from being sent multiple times across the same link. In fact, in the worst case, it is possible to have a duplicate packet for every level of the hierarchy in this scheme. HDVMRP also introduces additional network traffic by duplicating the unicast routing. As many more routers become multicast capable, the unicast and multicast topographies in the Internet converge, and the separate routing performed by DVMRP becomes an unnecessary burden on network bandwidth.

Just as using shared trees allowed reduction in router state and in control traffic for a flat routing domain, using shared trees reduces router state and control traffic in a hierarchical scheme. There are other compelling reasons to use shared trees at a higher hierarchical level. Domains are relatively static, and running a separate unicast routing protocol between them can consume bandwidth unnecessarily. Domains are also generally connected by fewer higher-speed links, which reduces the inherent shared-tree problem of link congestion. Recognizing these benefits, the IDMR working group of the IETF has been developing a shared-tree, hierarchical protocol currently called the Border Gateway Multicast Protocol (BGMP) [11]. Along with the Multicast Address Set Claim protocol (MASC) [5], BGMP proposes a mechanism for building associating groups of multicast addresses with particular domains and then building a single level tree of domains to the domain associated with that multicast address. As of this writing, this proposal exists only as a draft document, and no proof of correctness or proof of loop freedom is available.

This paper presents a new protocol called the HIP protocol (HIP) that uses OCBT as the inter-domain routing protocol in a hierarchy that can include any multicast routing protocol at the lowest level. The goal of HIP is to provide hierarchical multicast routing at a highly reduced cost in terms of network traffic and storage requirements at routers. The HIP protocol introduces the idea of a *virtual router* (VR) that is formed by all border routers of a domain operating in concert to appear as a single router in the higher-level tree. Virtual routers provide a convenient level of abstract on and enhanced robustness, while providing a data delivery service that does not duplicate packets over any link. A domain can use any multicast protocol internally while acting as a virtual router on the higher-level tree. OCBT is used to route between actual and virtual routers at a higher level in the multicast routing hierarchy, and if more hierarchical levels are required, an OCBT domain containing virtual routers can itself be made into a virtual router for a higher-level tree. Additional functionality gives HIP the option to allow a convenient and efficient method of center point location[1]. HIP's functionality comes primarily through the operation of a domain's border routers, and is therefore amenable to easy deployment by aligning the multicast domains with existing domains. HIP always attempts to maintain short paths by choosing the border router with the shortest path to the center point to provide connectivity for the entire domain. Furthermore, HIP also provides additional robustness at the highest level by replacing a single center point of the shared tree with several routers that operate together in a distributed fashion and can tolerate failure. HIP also provides a possible framework for center-point location distribution at a domain level.

The next section describes the construction and operation of virtual routers and how they are used to build a single level of hierarchy. Section 3 details HIP's recursive application of OCBT to create a multi-level hierarchical tree and mechanisms for center-location distribution. Section 4 presents an example of how HIP distributes center-point information and builds a hierarchical tree. Section 5 offers a proof of HIP's correctness.

2 The HIP Protocol

The correct operation of OCBT is not dependent on the order in which messages are received at the router, though the resulting multicast tree may differ in shape from a tree produced by a different ordering of messages. This property allows us to operate an entire domain such that it can accept the same input messages as an OCBT router and produce correct, if not always identical, output. By carefully organizing the *border routers*(BR — those routers that define the edge of the domain and "speak" the protocol of both the internal and external domains), the input arriving on the external interfaces can be processed collectively in such a way that the output of the virtual router can simulate the output of a single router in place of the domain. This organization turns out to be very simple. For each multicast group, one border router is selected to be the *controller* for the group. This selection can be dynamic, with all border routers agreeing by communicating over an All-Border-Routers (ABR) internal multicast address, or it can be pre-configured. All other border routers, called *subordinate routers*, forward all virtual router control traffic to the controller, which keeps all state for the virtual router and sends instructions back to the border routers. The controller can communicate with the border routers over individual TCP connections to insure that control messages are not lost, or over the ABR address using some reliable multicast protocol. A simple analogy can be drawn between an OCBT router and an OCBT virtual router: a normal router keeps all the routing state and sends messages to its neighbors via given interfaces; a virtual

[1] Note that we designate the router that forms the root of the tree as the center point (CP) rather than the core to avoid confusion with cores that may be local to a single domain and do not effect the entire tree.

router controller keeps almost all the routing state and communicates with it neighbors via given border routers. The only state that subordinate routers maintain is their own list of on-tree interfaces so that they can forward data as it arrives without it going through the controller. The subordinate routers do not change any state on their own. Only the controller makes state changes and notifies the subordinates when to change their state.

When a virtual router is part of the tree, the controller selects which subordinate will serve as the *exit router* to connect the domain to its parent on the higher-level shared tree, and always chooses for this role the border router with the shortest path to the given center point. Selecting the exit router in this way helps maintain short paths from lower to higher levels. The controller chooses the exit router by sending a message asking each BR to query its routing table for the distance to a specified address, and then chooses the BR with the shortest reported distance as the exit router. Border routers that have on-tree links for a particular group outside the virtual router must join the same multicast address internally. This way, when data is received from outside the VR, it is forwarded across the internal multicast domain to other border routers and then onto the other links attached to the virtual router. Data flows just as it does on a single-level shared tree. The actual internal operation of the virtual router will differ depending on the internal multicast protocol used. Source-routed protocols, such as DVMRP and PIM-DM, use reverse path forwarding checks to accept or drop a packet. Data may arrive and be distributed from different BRs than those that pass the reverse path forwarding (RPF) checks. In these cases, some data packets might need to be encapsulated and forwarded to the correct border router for distribution internally so that the RPF checks are passed. Shared tree protocols, such as PIM-SM and CBT can use whatever core dissemination mechanism they choose internally, and the border routers can connect or send directly to core or RP as needed. For OCBT, the border routers themselves can be used as cores, with the exit router serving as a level two core and all other BRs as level one cores. This way, the internal routers can be pre-configured to contact any BR as a core, and the tree formed from there as required. This is illustrated clearly in Section 2.2.

There are advantages and disadvantages to the virtual router approach. Clearly, a message arriving at a border router needs to travel to the controller and the reply returned before the subordinate can reply for the VR. This increases the response latency of the virtual router. For an OCBT virtual router, the maximum incurred delay will the twice the round trip time from the receiving BR to the controller (as the control message is forwarded and an answer received), plus the maximum round trip time from the controller to the furthest border router (if the controller needs to query the subordinates to receive routing distance information). While this latency grows as the domain size increases, there is a simple trade-off between bandwidth and latency that might help. Each time a border router receives a message likely to cause the controller to issue a request for routing information, that border router can transmit the address to be looked up on the ABR address as it forwards the control message. When the controller receives the control message, it then needs only to wait for the replies and does not have to send out a request itself; unnecessary replies can just be dropped. This will lower the latency to twice the round trip time from the receiving BR to the controller, plus a time less than or equal to the maximum time from the receiving BR to any other BR plus the maximum time from any BR to the controller.

The other disadvantage of a virtual router set up is that the overall paths traversed by a link can be made much longer by the need to traverse a virtual router. If a connection to a center point encounters a virtual router, then the path needs to traverse the VR, even if a short path is available. As an example of this, look ahead to Figure 3(b) and assume that some undrawn router one hop from router D and outside the VR wants to join the multicast group at the center point shown. Because the VR has chosen the border router E to contact the parent, the path for our imaginary router will go one hop to D, three hops internally to E, and two more hops to the center point. This will occur even though there is a shorter path to the center point for our imaginary receiver that avoids traversing the virtual router. This seems to be a problem with any shared tree hierarchical protocol; as a particular domain may have only one parent on the shared tree, some receivers within the domain or connecting through the domain will have worse paths than they might have in a flat routing scheme. This is the design tradeoff – lower state and lower control overhead in exchange for longer paths, just as might be intuitively expected by changing from a source tree to shared tree. The actual delay incurred by the longer paths is dependent on the network topography, though it appears that the difference between a source tree and shared tree might not be as significant as intuitively expected.

While using virtual routers can increase join latency and transmission delays, these are known design trade-offs to gain the benefits that virtual routers can bring. Virtual router provide a convenient level of abstraction in building hierarchical protocols. Many protocols exist that provide flat multicast routing and there is significant experience with these protocols. Using virtual routers allows these protocols to be used in an inter-domain role without significant modification. Additionally, virtual routers can be configured as desired on a policy basis; the domains that are routed between are configured manually rather than automatically. While this imposes a small burden in configuration it should not be excessive, as most organizations are fully aware of their domain boundaries and where the border routers are. This can be a major benefit as the virtual router can be made to use policy-based unicast routing scheme, such as BGP [9], to route to other domains. The scheme is also flexible enough to allow for virtual routers within virtual routers if necessary. This can be used for as many levels of hierarchy as needed or desired, as demonstrated in Section 3. Finally, use of virtual routers can significantly reduce the amount of state maintained by routers and the amount of control traffic between them, depending on what routing protocol is used between domains. To this end we propose using OCBT as the inter-domain routing protocol.

2.1 OCBT as an Inter-domain routing protocol

OCBT has several properties that make it a good candidate for use as an inter-domain routing protocol. Besides being proven correct and loop free at all times, OCBT relies directly on the unicast routing protocol used to route between domains, so that policy-based routing such as BGP can be used. OCBT is also a very simple protocol, and can be easily represented by a finite state machine as shown in Figures 1 and 2. In these figures, the possible transitions are marked with the type of message that causes the transition on top and the type of messages the router emits before changing

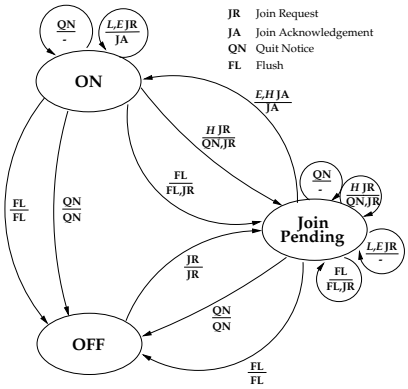


Figure 1: OCBT state transitions for a non-core router.

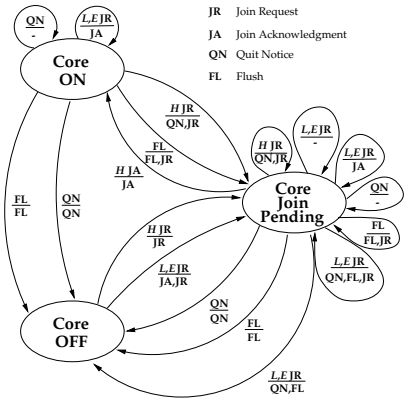


Figure 2: OCBT state transitions for a core router.

state on the bottom of the transition label. The italic *L,E,* and *H* indicate whether the level of the received message is of logical level lower, equal to or higher than that of the receiver; a comma means any of the listed levels. Under normal operation, by which we mean in the absence of network partitions, each OCBT router is in one of only six states, and there are only four types of messages related to the construction and destruction of the tree. Each router also sends periodic keep-alive messages to its parent to ensure that the link is alive and that the state between routers is consistent. The messages themselves are very small, the largest containing two integers representing the message type and message level and three IP addresses representing the message source, the message originator (if the source is forwarding a message) and the core for which the message is destined. This is 14 bytes of information, and most messages are smaller; many consist of only six bytes of control information.

Internally, each OCBT router needs to keep very little state for each multicast group – two integers representing the routers state and level; the address and interface over which to reach the parent router and, if a core, an integer representing the level of the parent; a list of addresses of cores which will vary in size depending on the scheme used to disseminate core information and a single address of the current core; a list of the level, address and interface for each child directly attached to the router and an integer for each child to track keep-alive messages; and only two timers for generating retransmissions and to track keep-alive messages.

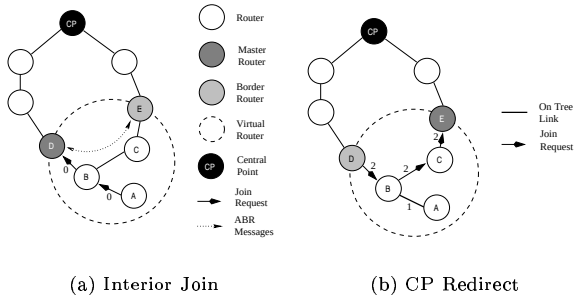


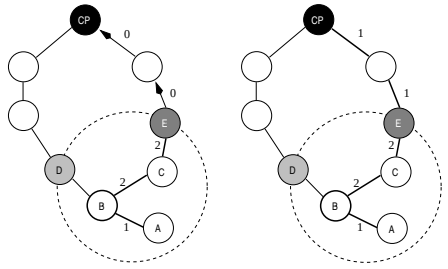
Figure 3: Interior OCBT Virtual Router Operation

Assuming four byte IP addresses and single byte integers for interface labels and for tracking levels, a core with five children and a list of five possible cores (a reasonable value under the core scheme in which each border router is an OCBT core) will therefore need only 66 bytes of storage and two timers for that group, not counting such things as pointers for the storage data structures. The state that the controller of an OCBT virtual router would need to maintain is similarly small. Instead of maintaining children based on their ID and interface, the controller would also need to maintain which border router was parent to the child. This would add only one byte, given a list of border router addresses at four bytes each but usable by all groups, that the single byte could be used to index. In addition, the parent interface needs the same index. The controller also needs to use the keep-alive timer to send and receive keep-alive messages over the ABR address, and needs to keep one byte for each border router to track missing keep-alive messages. Therefore, if there were five border routers (the same five that the lower-level routers used as a core list) and there were still five cores for the VR, this would still total less than 100 bytes of storage for the multicast group. This state can be further reduced by maintaining child lists and the parent identity by interface only and omitting the child ID; this is certainly very effective if there is only one child per link, but requires some modification to the protocol to handle broadcast links with multiple group members. This modification would remove the child ID list and reduce the OCBT state for the example given to only 39 bytes and the VR state to 50 bytes, given the same number of VR children and border routers.

2.2 An Example of an OCBT Virtual Router

We now describe a virtual router as it might operate for a single OCBT domain on a higher-level tree. Although HIP can support any multicast routing protocol running in a lowest-level domain, HIP uses OCBT at the higher levels so its operation is worth understanding. In Figure 3(a) the example first shows a receiver initiating the join for its domain and the controller determining the master router for the multicast group. Figure 3(b) shows how OCBT can create a tree between border routers to "redirect" a join request from a subordinate to an exit router. The sequence in Figures 4(a) and 4(b) shows how the virtual router acts as a normal OCBT router to join the multicast tree at the higher level.

In the example topography, routers *D* and *E* are the border routers of the OCBT domain and the virtual router con-



(a) Ext. Join (b) Ext. Reply

Figure 4: Exterior OCBT Virtual Router Operation

sisting of the border routers and the other single-lettered routers. At some point in time, as shown in Figure 3(a), router A determines that it has members on some subnet that wish to receive a particular multicast group that has a given CP address associated with it. Router A initiates a normal level-zero OCBT join request for the designated CP. This request travels to router B, which consults its unicast routing table for the best next hop towards the center point. Though there are two equal paths, B chooses the next hop according to its routing table and happens to select the border router, D, as the next hop. In this case the border router is not yet part of the multicast tree. So far, this is all normal OCBT operation, but when the request reaches the border router, HIP causes a slightly different behavior. Because any router on a shared tree can have only one parent, and because HIP attempts to build shortest-path trees, the virtual router controller must determine which will be the exit router for that multicast group based on proximity to the groups designated center point. The controller sends a message requesting both the border router D and E's unicast distance to the given center point. The subordinate routers reply with that distance (if and only if its next hop to the center point is outside of the domain, otherwise it returns a message that it has no external path), at which time the controller determines that E is closer to the center point, and is thus chosen to be the exit router for the domain to that CP.

The border routers are now configured to build the shortest-path tree for the domain; however, the join request from A that started the process has reached the wrong router to reach the CP through the chosen exit router. This is not a problem because we use OCBT. In every OCBT virtual router, the subordinate routers act as a level-one cores while the exit router acts as a level-two core. Figure 3(b) shows how the router D simply sends a level-one join acknowledgment back to router A, instantiating a level-one branch from the subordinate router to A, then sends a level-two join request towards the exit router E to form a branch from itself to E so that it can deliver data on the level-one branch. As it happens, the route from the master router to the subordinate traverses part of the level-one branch. Because OCBT breaks existing lover-level branches to allow formation of higher-level ones, the link between D and B becomes part of the level-two branch that is forming along the path from B to C to the exit router. Router A still receives data from B over the level-one link once the level-two branch is in place, which occurs when the exit router receives the request and sends the appropriate join acknowledgment back to B. At this point, D realizes that it has no children and does not want to receive this multicast group, so it quits from the tree resulting in the internal tree structures shown in Figure 4(a).

The virtual router still must join the external tree. This is straight forward; the exit router sends a normal OCBT level-zero join request to the CP; as shown in Figure 4(b), the CP replies with a level-one join acknowledgment that forms the branch back to the virtual router. If in the future, the subordinate router D receives a join request on an external interface, it need only resend the internal level-two join request towards the exit router to rejoin the internal tree.

3 Hierarchical Routing and Center Point Dissemination

The previous section shows that HIP can be used to build a single level hierarchy, first by constructing virtual routers from domains and next building a shared tree, without discussing how to provide multiple levels of hierarchy or the details of how the center point location is distributed or even what the center point is. In HIP, it is a simple matter to provide multiple levels of hierarchy if desired or needed. Assume that a large organization runs it own network. Smaller parts of the organization each have their own subnetworks which run different multicast protocols internally. In order to route between the subnetworks, the organization has made each subnetwork its own virtual router and is routing between them internally. Now the organization wants to route multicast traffic to other organizations, requiring another level of hierarchy. Recursive application of the virtual router protocol solves this problem. The organization places its entire network into a single virtual router and places that virtual router on the shared tree between organizations. A three-level hierarchy is now in place without making any modifications other than at the border routers of the organization. This recursive application can be applied as many times as necessary or convenient. Distributing center points for higher-level shared trees requires some additional work. The following sections describe center points and their dissemination in detail and give an example of the recursive application of HIP and its CP mechanisms.

3.1 Virtual Center Points

The hierarchical nature of HIP dictates that a domain that contains a CP within itself must become a virtual CP for the shared tree at its level. The shared tree at each level will then have a CP with which to connect. To see that this is necessary, consider the arrival of a join message on the external interface of a border router destined for a particular internal router that is designated as the physical CP for a multicast group. Forwarding the request to join inward across the domain border and onto the inner shared tree would be a violation of the hierarchy. Therefore, the border routers must reply to the join message as if they were the CP. The ability to serve as a virtual CP is already a necessary part of the encapsulation protocol that completely simulates a single router, as being a core is one of the possible router states. After acknowledging the external request, the border routers then can send a join message inwards towards the real CP, which may connect directly or may be received by some interior virtual CP. Border routers must be able to determine which center points are within their domain in order to recognize requests destined for an internal center point.

HIP's use of virtual center points is a major advantage when router failures are considered because each virtual router can be made up of several physical routers that can provide redundancy. A single CP is a likely point of failure in a shared-tree protocol, and though such protocols take this eventuality into consideration, the solutions to the problem can be expensive in terms of overhead and network traffic. Because the virtual router is made up of several border routers that work in concert to provide the same service, the failure of one or more of the border routers does not constitute failure of the virtual CP, and remaining border routers can maintain the functionality.

3.2 Center Point Location Distribution

A disadvantage of receiver initiated multicast protocols is that a receiver must be able to determine the location of the center point associated with the group address in order to join that group. There are four approaches to distributing this information [4]: advertisement at the application level; advertisement at the router level; a distributed directory supporting center point lookup; and algorithmic mapping. Each has advantages and disadvantages; application level advertisements can be protocol dependent, distributed lookups can introduce startup delay for receivers, router level advertisements require a significant change to the multicast model and algorithmic mapping can require significant network traffic to implement and can create very poorly shaped trees over many levels of hierarchy. While there is no best answer to CP location dissemination, HIP uses a combination of several approaches that minimizes network traffic, reduces the information that any individual router must have, and limits the delay before a member of the multicast group can start transmitting. HIP does not require that this particular method of CP location be used if other methods are made popular. For example, the Multicast Address Set Claim [5] provides a method of distributing a mapping of multicast addresses to particular domain which act as center points and could be used with HIP without significant modification; however our methods allow for easy mixing of sender and receiver initiated protocols.

To help distribute center point locations effectively, HIP defines an additional multicast address within each higher-level domain. This address, known as the all-virtual-router (AVR) address, can be subscribed to by any virtual router that encapsulates a sender-initiated domain at the lowest level. Any domain that has internal AVR receivers joins the AVR address in the higher-level domain. Using the ABR and AVR addresses across the hierarchical domains, we can create an efficient means of distributing information about available multicast groups and their CP locations. To minimize the amount of traffic required to distribute the group address to center point location mapping, we combine two approaches. In the first and most conservative approach, an advertisement with the mapping is transmitted by the new CP (or by the creator of the multicast group) to the ABR address. One of the border routers receiving the advertisement caches it and then forwards it on its external ABR address. This process continues until the advertisement reaches the highest level, where it is either flooded across the entire level or delivered to some directory service available to the highest level routers and virtual routers. Each router is configured with the addresses of its border routers; when a receiver wishes to receive the multicast group, its router sends a join request towards any border router. The border router that receives the request acknowledges it and then commences locating the proper center point of the group. If the receiving group of border routers does not contain the CP information in their cache, the controller for that domain creates and forwards a CP location information request towards the higher levels on its external ABR address. As the border routers of each higher level receive the lookup request, they check their distributed cache and forward the information back to the requester or forward the request up. This process continues until the request reaches the top level or a level where some border router has the information cached. The border router at the level containing the information sends it down along the reverse path of the request. The reply is cached by each successive set of border routers and forwarded along the reverse path until it arrives at the requesting virtual router. That virtual router is then free to join the tree using a process discussed in Section 3.3.

This approach limits the amount of network traffic needed, and only receivers requesting the information actually receive it. The information is also cached at each level, as it is reasonable to expect other nearby receivers may try to join the group, and having the information cached will reduce the number of requests traveling upwards in the hierarchy. There are two drawbacks to this approach, however. First, each virtual router must wait for CP information before joining the tree, increasing join latency for the domain and requiring that the border router cache any data being sent from internal members that are sending data. Second, a receiver must initiate the join, which is fine for the shared-tree protocol used at the higher level, it will not work for sender initiated domains (e.g. DVMRP or PIM-DM) that may be operating as a virtual router at the lowest level. These domains must receive an announcement that the group is available. Thus, a more widespread distribution of group information is required to accommodate these protocols.

In the second approach, information about the availability of a group and its center point is multicast to all border routers in the region. The CP does this by announcing it self on both the ABR and AVR addresses within its domain. One border router for the VR receiving such an advertisement on an internal interface also forwards it to both the external ABR and AVR addresses; if the advertisement is received on an external interface (as a result of being sent to the AVR address) it is forwarded to the internal AVR address, with the same elected border router address described above added to aid the joining process. The information travels up to the highest level via ABR transmissions; it travels across the highest level and to and into each domain as a result of the AVR transmissions and eventually reaches all domains in the region. A domain containing a sender initiated protocol will have the advertisement (which can simply consist of the first data of the session) flooded within it from one border router, and if the group is not pruned back entirely, a join request will be generated by the virtual router for the group. A border router from a receiver-initiated group wishing to join can request the CP information via the ABR address and then commence the join process. If the request for the multicast group information results in a cache miss, the border routers can revert to the first method and forward the request upwards until the information is discovered. While this protocol is robust and addresses the need for sender-initiated protocols to be notified about group availability, it can be very wasteful in terms of excessive control traffic if there are few sender initiated domains.

In HIP, the method of CP information combines both

approaches; receiver initiated lookup for areas with few receivers and without sender initiated domains, and multicast distribution for areas needing robust delivery. Only virtual routers containing a sender initiated group or a set of border routers that process too many requests for CP information need join the AVR address on its external interface. Any domain with an internal receiver on this group must become a receiver on the same address on the higher-level shared tree. This continues up to the highest level; advertisements for service arrive at and get sent across the highest level and down to each AVR receiver. Domains not subscribing to this group use the first request mechanism instead.

Limiting the area of a multicast via administrative *scoping* is very simple in HIP. The original advertisement announcing the availability of the group can also contain the name of the highest domain that can carry the multicast. When the advertisement reaches this level it is not forwarded to the higher level. Receivers outside the scoped domain will not be able to receive information about the group and hence will not be able to join. The information about a scoped group must be maintained at the highest scoped level until the group expires however; if the information were allowed to drop out of the cache there would not be any information about the group at a higher level to answer requests.

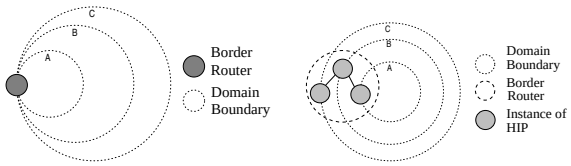
3.3 Building the Tree

With a mechanism for CP information distribution in place, the process of building the tree can begin. For lowest-level domains using a source initiated internal protocol, this occurs when the group advertisement is flooded throughout the domain from one elected border router and is not pruned completely back, indicating that some internal member wishes to receive the multicast. In a receiver initiated protocol, it starts when some internal router receives an IGMP [6] message from a subnet requesting the multicast. This physical router sends a join request to some border router of its domain and connects to it. If the border routers do not have the correct CP information, one retrieves it via the method described in Section 3.2. If the contacted border router turns out not to be the exit router for that group for the domain, it can form a connection to the exit router as shown in Section 2.2. The exit router then sends a join request for the group towards the appropriate center point.

This join message reaching the next higher set of border routers triggers that virtual router to join the forming tree itself. In this case the virtual router already has the information needed as to where to target its join message; it was forwarded and cached previously as a result of the internal router joining. If a virtual center point receives the request, it responds as a real CP would. The receiving border router then sends a join inward towards the actual CP. This join could pass through several other virtual CPs before reaching the actual CP; each responds as if it were the actual and then joins the group internally as well. No border router joins the internal multicast group unless it is necessary.

3.4 Border Routers for Multiple Domains

All of our explanation and examples of HIP have assumed that each domain has a single border router that operates for that domain only; that is, no single border router serves multiple domains on either side. In practice this need not be true. While HIP does require that a lower-level domain be enclosed by the higher-level domain, should a higher-level



(a) Physical Configuration

(b) Software Configuration

Figure 5: Multiple domains sharing a common border router.

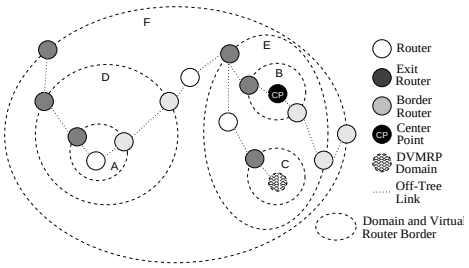


Figure 6: Example Topography

and some lower-level domains share a border router, a few special steps need to be taken to ensure that the protocol operates properly. Figure 5(a) shows a single border router that serves three separate domains, A, B and C.

The border router runs multiple instances of the virtual router protocol, one for each domain for which it is a border. This is shown in Figure 5(b), in which each instance of the protocol serves one domain and all instances are organized into a stack, with the instance serving the lowest level domain on the bottom and each successive instance serving the next higher domain. This simulates each domain having its own border router that is connected directly and only to the next higher domain's border router. When traffic arrives at the bottom-most instance of the protocol that would normally be forwarded along to the next higher domain's border router it is instead simply passed up the protocol stack as appropriate. This allows as many domain boundaries to sit together as necessary. This method does require that messages from border routers be marked with which domain the border router is in, so when it arrives at another border router serving multiple domains, the message can be sent to the appropriate instance of the protocol.

4 An Example of HIP in Operation

While the actual ideas behind HIP are fairly simple, its recursive organization can make it difficult to envision it in action. This section presents an example showing, through a series of figures, how the protocol distributes CP information and constructs a shared tree. Figure 6 shows the example topography used throughout the example, and is followed by Figure 7, which shows how the center point location is disseminated throughout the network. Once the center-point is known, individual domains are able to join the multicast tree. A sender-initiated domain's joining process is presented in Figure 8 and a receiver-initiated domains process of locating the center point and joining the description in Figure 9

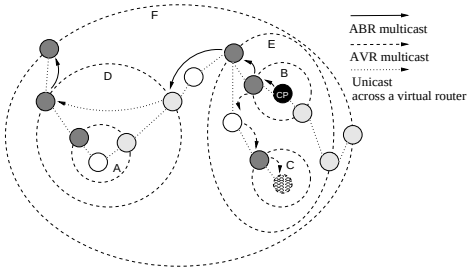


Figure 7: CP Location Dissemination

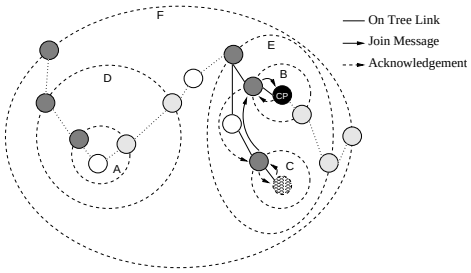


Figure 8: The DVMRP Domain Joins

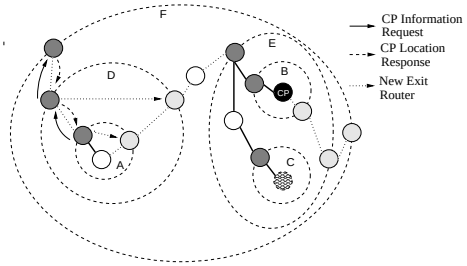


Figure 9: Receiver Initiated CP Information Request

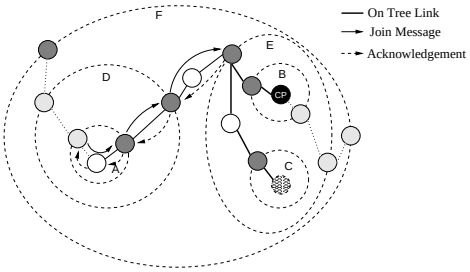


Figure 10: Receiver Initiated Join Process

and Figure 10.

Figure 6 shows the simple topography used. Virtual router A is an OCBT domain that does not subscribe to the all-virtual-routers address to receive CP information, while C contains a DVMRP domain that must be notified when a multicast group becomes available and therefore does subscribe to the AVR address. Virtual router B contains the physical CP for the group, making both E and F virtual center points for their levels. The exit routers are not aligned in any particular configuration initially; once they receive appropriate CP information they will switch to lie on the shortest path. In practice they need not even be identified until needed, however for illustration purposes they are selected and made visible.

The setup of the multicast group begins with the distribution of center point information. The new CP announces its availability on both the all-border-router and all-virtual-router addresses. The all-virtual-router announcement travels only within domain E to C as there are no subscribers to the AVR address in the higher-level domain F. The advertisement on ABR travels out of domain E to domain D, where it is unicast across the domain and forwards it to the exit router for the overall domain labeled F so that it can be forwarded towards the highest level its scoping permits. The route that the CP information follows is shown in Figure 7. Notice that domains A and D only forward and do not cache the CP information as they are not ABR receivers, but only hops on the shared tree at that level.

Once the CP advertisement reaches the DVMRP domain, the exit router of that domain (not illustrated) floods the advertisement through the domain. When the tree is not pruned back, the controller has the exit router send a join message towards the primary CP address received in the CP advertisement. This request travels first to the exit router of domain C, where it is acknowledged and forces the issuance of a join request for the domain. This higher-level request goes directly to the exit router for domain B. It does not need to be acknowledged by the border routers of domain E, as B is the virtual CP within E and E sees the request pass by on an internal interface. The join message is acknowledged back to C, and the exit router for B then joins the internal physical CP. This process is illustrated in Figure 8.

When the single router in the A domain receives an IGMP message from an attached subnet that desires the multicast, it selects a border router and joins to it for that multicast group. The controller for A must then locate the correct CP for the group as even though it traversed the domain earlier, it did so as an encapsulated control packet and was not read or cached in A. Figure 9, shows how A sends a CP information request to the ABR group of its domain, D, which gets received at that domain's exit router. Because the original core advertisement passed through the D domain without getting read or cached, the controller for D must also forward the request to a higher level. The information is found when the request arrives at the F controller and is sent back down to D. There the controller consults the border routers confer and realize that the exit router must be switched to provide a shortest path to the indicated CP. The CP location is forwarded to A and the exit router for that group is switched.

At A, again, the controller realizes that the other border router must be the exit to provide the best exit point for the domain, and the role is transferred. The former exit router then redirects the initial join from the internal router by sending a level-2 request to the new exit router. This triggers the controller to have the new exit router to send a join request for domain A as shown in Figure 10. As in the join from the DVMRP domain, it gets acknowledged at the border router on the way out of domain D, then travels to domain E, where it encounters a border router that is already part of the tree that sends the final acknowledgment back to the D master router, completing the branch.

5 Correctness of HIP

Proving that HIP is correct requires us to show two things; first, that OCBT always builds a multicast tree in bounded time without loops; and second, that the virtual router protocol correctly emulates the function of a single OCBT router. Recursive application of these proofs will then show HIP to be correct at all levels. First, OCBT has been previously proven to operate correctly; it always produces a multicast tree in finite time even under unstable unicast routing and link and router failure [10]. The proof relies on the fact that timers are used to generate retransmissions, so OCBT is always live, and that the use of logical-level labels guarantees that branches will form while preventing loops in the construction of the tree. It therefore remains to show that the virtual router protocol is never deadlocks and that it always correctly emulates the function of an OCBT router, even under internal link or router failure.

Theorem 1: Given a connected network, the virtual router protocol remains safe and live and always correctly simulates an OCBT router.

Proof: It is easy to show that the virtual router protocol does not reach deadlock as each message that requires a response has associated with it a timer. If the timer expires without a reply being received, the message is either resent or, after several failures, it is assumed that the intended recipient is unreachable and appropriate action taken. In either case, deadlock is avoided. To prove that a virtual router has the same functionality as a OCBT router only is slightly more complicated and requires that the responses, in terms of changes of internal state and transmission of control or data messages, be identical for both a virtual router and an OCBT router. This is also very simple, as the virtual router runs the same protocol and keeps the almost the same state as an actual router. Instead of tracking children by interface, the virtual router tracks them by border router and interface, and the subordinate routers keep state about data forwarding as directed by the controller. In addition, the routing table lookup of a router is replaced by the distributed lookup of the virtual router; since these transmissions and their replies go over reliable unicast or multicast channels the reply is assured as long as the routers are up. Therefore, this lookup will succeed and the virtual router can forward messages based on the distributed routing tables. When the pseudo code for OCBT and an OCBT virtual router is compared and state transition diagrams are prepared, they are identical. A virtual router therefore behaves the same as an actual router would in its place. □

Given that a virtual router behaves the same as an OCBT router under normal conditions, we now need to examine its performance under failure. There are three ways a link or router can fail in a domain. The failure can occur such that either the ABR multicast tree or the data multicast tree is disconnected but paths exist between all border routers; a border router can fail; or a partition can separate the border routers into separate connected components. The following three lemmas show that HIP functions correctly in each of the three cases.

First we make some simple assumptions about the nature of the internal multicast routing protocol. First, we assume that if there exists some route between all border routers, the internal multicast routing protocol will rebuild the multicast tree in some finite time following a link or router failure. Second, if the domain is partitioned and there is no possible route between all border routers, we assume that the internal multicast protocol will build an ABR and data multicast tree within each partition, so that any border routers within the partition will be able to communicate, and that the tree will be rebuilt when the partitions merge. These are correct assumptions for all OCBT domains, as OCBT correctly builds a tree in finite time in any connected component following a failure or partition [10]. If, in fact, the internal multicast cannot support this, the virtual router protocol encapsulating that domain will need additional functionality to fulfill those requirements.

Lemma 1: In the event of an internal link or router failure such that all border routers have paths to each other, the virtual router protocol will continue to operate correctly.

Proof: In fact, in this case HIP needs to do nothing. The internal multicast protocol is able to reconstruct either or both the ABR and data tree as a path exists between all border routers, and the controller will be able to communicate and share data and control messages with all border routers. There may be internal routers that are not able to receive or send on those addresses, but that does not effect the function of the virtual router as a whole; it is still able to make appropriate state changes and to carry data across the virtual router for transmission on the higher-level tree. □

Lemma 2: In the event of a border router failure, the virtual router protocol will continue to operate correctly.

Proof: In this case, a single border router of a virtual router fails. This failure will be detected by the other border routers. If the controller is the router that failed, this will be detected when the subordinates cease receiving keep-alive messages; if it is a subordinate router that fails, then the controller will detect the failure when the keep-alive messages go unanswered. If it is a subordinate router that goes down, the master router simply removes state about any children of the virtual router that were being served by the failed router, as if the external link of an OCBT router had failed. This is safe as the outgoing links are unreachable through the failed router and it is as if the domain simply has a different topology. If the controller is the router that has failed, it as if the domain has been partitioned in such a way that the controller is unreachable and the subordinate routers follow the same procedure as a network partition and the case is covered by Lemma 3. □

Lemma 3: In the event of a network partition, each partition will form its own virtual router, resulting in correct performance but a topology change.

Proof: In this final case, the virtual router is partitioned in such a way that some border routers are separated from each other, with one or more border routers part of each partition. Because the internal multicast protocol rebuilds the ABR tree within each partition and because the only exits out of the partition are through the border routers, each separate partition meets the criteria of and acts as its own virtual router. Therefore, each partition can appear as an OCBT router on the higher-level tree. The partition that does contain the controller simply removes state about the border routers it cannot reach and continues as in the case above. The partitions that do not contain the controller must elect a controller for that partition once the ABR multicast tree is constructed within the partition. This mechanism is very simple: each border router within the partitions sends keep-alive messages to the ABR address; as all BRs within a partition listen to this address, after several keep-alives have been sent each border router will have the addresses of all other BRs within the partition. One particular BR (the one with the lowest or highest IP address) will then take over

as the controller and send an announcement of its status to the others within the partition; this announcement causes them to immediately drop all state about children that they have, after which they modify their state as directed by the controller. To the network outside the VR, it will appear as if the VR has restarted, and new connections will be formed to the VR. When the partitions merge again, the ABR multicast tree between the two partitions reforms and each former partition receives the keep-alive messages of the other; at this point an election is held again for controller, with one particular router selected again by IP number or by pre-configuration if the set controller is within the partition. The losing controller sends no message but drops all routing state; the elected controller again announces its status to all other border routers. In this case, only the former subordinates of the losing controller need clear VR state and start anew. The these cases the topography of the higher-level tree is different now, with several disconnected virtual routers where there used to be a single one, but each virtual router will still function correctly. Because OCBT functions correctly regardless of the topography, HIP will still operate properly.

If a virtual center point is partitioned, it forms a new group of virtual routers as described above, but with one difference; only the partition containing the actual center point can respond to build the higher-level tree. Each part of the partition must therefore check via its unicast routing tables that the physical center point is reachable before accepting connections for it on the higher-level tree. The reason for this is clear – if there were several virtual center points accepting connections, a single tree would not be built at the higher level; instead, a forest of disconnected trees would be built. This may mean some partitions will be rejecting connection requests until the unicast routing has updated routes to the physical center point, but in time (if the partition lasts long enough) the join requests will start going to the correct partition. □

Theorem 2: Given failures inside a domain, the virtual router protocol will still simulate one (or more) OCBT routers.

Proof: Since the virtual router protocol functions correctly in each of the three possible failure conditions, HIP always correctly simulates an OCBT router, and since OCBT is correct, HIP will always be correct. □

6 Conclusions

We have described a new protocol for hierarchical multicast routing called HIP that joins heterogeneous routing domains using the Ordered Core Based Tree protocol. The method makes domains appear as a single virtual router on a higher-level shared tree using a simple protocol that organizes the border routers of the domain to simulate a single OCBT router. Any routing domain can be made a virtual router; all that is required is the ability to send multicast packets internally. Recursive creation of virtual routers yields a flexible hierarchy that is robust; has no strict ordering of levels; aligns easily with existing unicast domain boundaries; and always attempts to make the path to the center point of the tree as short as possible. The mechanisms for center-point location distribution are simple and effective in minimizing the amount of control traffic needed.

While maintaining the same functionality of other hierarchical multicast schemes, HIP approaches the problem in a very different manner. Routing between the domains is based solely on the unicast routing tables; the lower-level domains do not need to be explicitly named and no separate routing needs to occur to build paths to domains. This allows great flexibility in the addition or modification of domains in the region. Data traffic flows over a single tree, and while higher-level control messages may be encapsulated for transmission across a region, data packets are never encapsulated or duplicated. The tree itself is always built with an attempt at forming the average shortest path to the center point for the group. The ordering of the hierarchy is also extremely general; except for the highest level domain, no domain needs to know what its place is in the hierarchy. While the bulk of the control traffic in HIP is for the distribution of proper core and best-path information, the amount of traffic is minimized as much as possible by only sending the information where it is requested or where it needs to be delivered for the proper operation of encapsulated protocols. The HIP protocol takes best advantage of the convergence of the unicast and multicast topographies and the development of high performance shared-tree protocols, while retaining the ability to inter-operate with existing domains and protocols.

References

[1] Ballardie, A. Core Based Trees (CBT version 2) Multicast Routing Protocol Specification. Internet draft, July 1997. Work in progress.

[2] Deering, S., Estrin, D., Farinacci, D., Jacobson, V., Helmy, A., and Wei, L. Protocol Independent Multicast Version 2, Dense Mode Specification. Internet draft, May 1997. Work in progess.

[3] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and Wei, L. Protocol Independent Multicast-Sparse Mode (PIM-SM):Protocol Specification. Internet draft, September 1997. Work in progress.

[4] Estrin, D., Handley, M., Helmy, A., Huang, P., and Thaler, D. A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing. Tech. rep., University of Southern California, 1997. http://netweb.usc.edu/pim/.

[5] Estrin, D., Handley, M., Kumar, S., and Thaler, D. The Multicast Address Set Claim (MASC) Protocol. Internet-draft, Novermber 1997. Work in Progress.

[6] Fenner, W. Internet Group Management Protocol, Version 2. Internet draft, Xerox PARC, January 1997. Work in progress.

[7] Handley, M., Crowcroft, J., and Wakeman, I. Hierarchical Protocol Indepenent Multicast (HPIM). University College London, November 1995.

[8] Pusateri, T. Distance Vector Multicast Routing Protocol. Internet-Draft, March 1998. Work in Progress.

[9] Rekhter, Y., and Li, T. A Border Gateway Protocol 4 (BGP-4). Tech. rep., RFC 1771, March 1995.

[10] Shields, C., and Garcia-Luna-Aceves, J.J. The Ordered Core Based Tree Protocol, Proceedings of the IEEE INFOCOM, Kobe, Japan, April 1997.

[11] Thaler, D., Estrin, D., and Meyer, D. Border Gateway Multicast Protocol (BGMP). Internet-draft, October 1997. Work in Progress.

[12] Thyagarajan, A., and Deering, S. Hierarchical distance-vector multicast routing for the MBone, Proceedings of the ACM SIGCOM, Cambridge, Massachusetts, Sept. 1995.