Inter-Domain Multicast Routing (IDMR) A. J. Ballardie
INTERNET-DRAFT University College London
S. Reeve
Bay Networks, Inc.
N. Jain
Bay Networks, Inc.

February 9th, 1996

Core Based Trees (CBT) Multicast

-- Protocol Specification --

<draft-ietf-idmr-cbt-spec-04.txt>

Status of this Memo

Abstract

This document describes the Core Based Tree (CBT) network layer multicast protocol specification. CBT is a next-generation multicast protocol that makes use of a shared delivery tree rather than separate per-sender trees utilized by most other multicast schemes [1, 2, 3].

This specification includes a description of an optimization whereby native IP-style multicasts are forwarded over tree branches as well as subnetworks with group member presence. This mode of operation will be called CBT "native mode" and obviates the need to encapsulate

data packets before forwarding over CBT tree interfaces. Native mode is only relevant to CBT-only domains or ‘‘clouds’’. Also included are some new "data-driven" features.

A special authors’ note is included explaining the latest updates to the CBT specification, together with some nomenclature, and miscellaneous items.

This document is progressing through the IDMR working group of the IETF. The CBT architecture is described in an accompanying document: ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-arch-00.txt. Other related documents include [4, 5]. For all IDMR-related documents, see http://www.cs.ucl.ac.uk/ietf/idmr.

1. Authors’ Note

The purpose of this note is to explain how the CBT protocol has evolved since the previous version (November 1995).

Since the previous release, CBT has been assigned official IP protocol and UDP port numbers (section 8).

The CBT designers have constantly been seeking to streamline the protocol and seek new mechanisms to simplify the group initiation procedure. Especially, it has been a high priority to ensure that join latency be kept to an absolute minimum. The November ’95 draft introduced the re-invented subnet designated router (DR) election procedure, described here in section 2.3.

The concept of proxy-ACKs was introduced in the November ’95 draft, but these have been removed since the extra message overhead does not warrant the negligible gain they provide.

The CBT loop detection mechanism (comprising rejoin-active and rejoin-nactive) has been slightly modified, and is now simpler and more straighforward. The revised mechanism incorporates a new join ack subcode, and is explained in section 5.3.

Core selection, placement, and management, which have prevented simple group initiation/joining, apparent in data-driven schemes (like DVMRP), have been separated out from the protocol itself. Core management is not a problem unique to CBT, but also PIM-Sparse Mode. Separate, protocol-independent core management mechanisms are

currently being proposed/developed [8, 9]. In the absence of core management/distribution protocol, the task could be manually handled by network management facilities.

In CBT, the core routers for a particular group are categorised into PRIMARY CORE, and NON-PRIMARY (secondary) CORES.

The core tree, the part of a tree linking all core routers together, is built on-demand (section 2.4). That is, the core tree is only built subsequent to a non-primary core receiving a join-request (non-primary core routers join the primary core router -- the primary need do nothing). Join-requests carry an ordered list of core routers (and the identity of the primary core in its own separate field), making it possible for the non-primary cores to know where to join. On-demand core tree building is explained as part of section 2.4.

CBT now supports the aggregation of neighbour keepalives, which previously were sent on a per group basis. Any two adjacent CBT routers need only send a single keepalive between each other, rather than one per group. Additional aggregation strategies are currently being worked on, and we present some ideas on aggregated rejoins in Appendix A. An updated draft fully specifying CBT aggregation strategy should appear soon.

The end result of these developments is that the CBT protocol is much simplified and more efficient.

2. Protocol Specification

2.1. CBT Group Initiation

The requirement of hosts to discover the identity of candidate core routers (or RPs) differentiates the role of hosts in shared tree multicast protocols and shortest-path tree multicast protocols; the latter need only announce their desire to join a group by means of an IGMP membership report. It is highly desirable that hosts wishing to join a shared tree need only do the same, leaving local multicast routers to discover <core, group> mappings, or have local routers configured with the identity of core(s) in the next level of a hierarchy, as suggested by Hierarchical PIM [8].

If the latter approach is eventually adopted by the IETF, then host operations need not differ due to the type of multicast tree being joined, and indeed, the type of tree being joined for a particular group can remain transparent to the host.

If the latter approach is not adopted, then hosts need to inform their local multicast router of a <core, group> mapping for each group joined. This requires hosts to discover <core, group> mappings, which in turn requires the existence of a (global) core advertisement protocol. Hosts subsequently need a means of advertising <core, group> mappings to the local multicast router so it can initiate a join. This requires an extension to IGMP, for example, the presence of IGMP RP/Core Reports, as suggested in IGMP version 3 [7], or the protocol itself must provide a means (message) for advertising cores to the local router. In the absence of H-PIM, some similar mechanism, or IGMPv3, CBT implementors may wish to extend CBT to include a core reporting message for group initiators/joiners (for example, whenever a group is initiated/joined, a configuration file is read which holds <core, group> mappings).

Alternatively, <core, group> mappings can be downloaded to local multicast routers by means of network management tools.

2.2. Tree Joining Process -- Overview

A local CBT router is notified, by IGMP, of a host's desire to join a group. If more than one CBT router is present on the subnetwork, each will receive the IGMP membership report. However, only one, the default subnet designated router (DEFAULT DR) will act upon the receipt of a report by initiating a CBT join. Note, a CBT join is only initiated if the subnetwork is not yet part of the delivery tree. Also, we assume that the local CBT default DR discovers <core, group> mappings by one of the mechanisms described in the previous section. DR election is described in section 2.3.

The following CBT control messages come into play subequent to the host sending an IGMP join (host membership report):

+ JOIN_REQUEST

+ JOIN_ACK

A join-request is generated by a locally-elected DR (see next

section) in response to receiving an IGMP group membership report from a directly connected host. The join is sent to the next-hop on the path to the target core, as specified in the join packet. The join is processed by each such hop on the path to the core, until either the join reaches the target core itself, or hits a router that is already part of the corresponding distribution tree (as identified by the group address). In both cases, the router concerned terminates the join, and responds with a join-ack, which traverses the reverse-path of the corresponding join. This is possible due to the transient path state created by a join traversing a CBT router. The ack fixes that state.

2.3. DR Election

Multiple CBT routers may be connected to a multi-access subnetwork. In such cases it is necessary to elect a (sub)network designated router (DR) that is responsible for sending IGMP host membership queries, and generating join-requests in response to receiving IGMP group membership reports. Such joins are forwarded upstream by the DR.

The IGMP querier election is as follows (note, here we talk about "CBT routers", but the described mechanism also applies to the general case). At start-up, a CBT router assumes it is the only CBT-capable router on its subnetwork. It therefore sends two IGMP-HOST-MEMBERSHIP-QUERYs in short succession (within 5 secs) (for robustness) in order to quickly learn about any group memberships on the subnet. If other CBT routers are present on the same subnet, they will receive these IGMP queries, and depending on which router was already the elected querier, yield querier duty to the new router iff the new router is lower-addressed. If it is not, then the newly-started CBT router will yield when it hears a query from the already established querier.

The CBT DEFAULT DR (D-DR) is always (footnote 1) the subnet's IGMP-

---

1 This document does not address the case where some routers on a multi-access subnet may be running multicast routing protocols other than CBT. In such cases, IGMP querier may be a non-CBT router, in which case the CBT DR election breaks. This will be discussed in a CBT interoperability document, to appear shortly.

querier; in CBT these two roles go hand-in-hand. As a result, there is no protocol overhead whatsoever associated with electing the CBT D-DR.

2.4. Tree Joining Process -- Details

The receipt of an IGMP group membership report by a CBT D-DR for a CBT group not previously heard from triggers the tree joining process.

Immediately subsequent to receiving an IGMP group membership report for a CBT group not previously heard from, the D-DR unicasts a JOIN-REQUEST to the first hop on the (unicast) path to the target core specified in the CBT join packet.

Each CBT-capable router traversed on the path between the sending DR and the core processes the join. However, if a join hits a CBT router that is already on-tree (footnote), the join is not propogated further, but ACK'd downstream from that point.

JOIN-REQUESTs carry the identity of all cores for the group. Assuming there are no on-tree routers in between, once the join (subcode ACTIVE_JOIN) reaches the target core, if the target core is not the primary core (as indicated in a separate field of the join packet) it first acknowledges the received join by means of a JOIN-ACK, then sends a JOIN-REQUEST, subcode REJOIN-ACTIVE, to the primary core router. Either the primary core, or the first on-tree router encountered, acknowledges the received rejoin by means of a JOIN-ACK. In the former case, the primary core responds by sending a join-ack, subcode PRIMARY-REJOIN-ACK, which traverses the reverse-path of the join. In the latter case, the join-ack is returned with subcode NORMAL; the receiving router responds to this with a rejoin-Nactive, for loop detection. Note that loop detection is not necessary subsequent to receiving a join-ack with subcode PRIMARY-REJOIN-ACK. Loop detection is described further in section 5.3.

To facilitate detailed protocol description, we use a sample topology, illustrated in Figure 1 (shown over). Member hosts are shown as individual capital letters, routers are prefixed with R, and subnets

---

"on-tree" describes whether a router has a FIB entry for the corresponding group.

are prefixed with S.

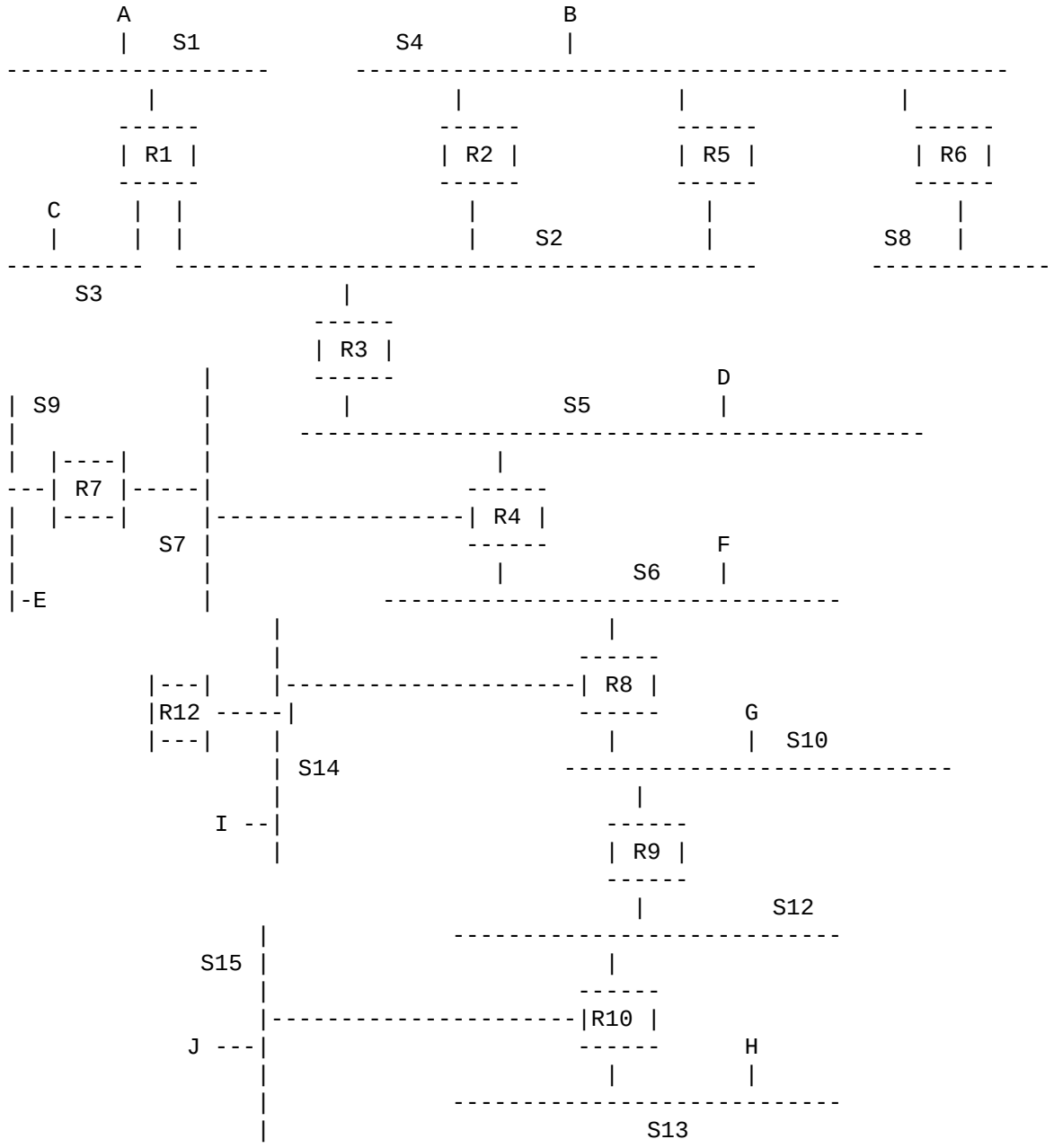

Figure 1. Example Network Topology

Taking the example topology in figure 1, host A is the group initiator, and has elected core routers R4 (primary core) and R9 (secondary core) by some external protocol. We assume the local CBT DR discovers <core,group> mappings by "some means", possible one of the mechanisms described in section 2.1.

Router R1 receives an IGMP host membership report, and proceeds to unicast a JOIN-REQUEST, subcode ACTIVE-JOIN to the next-hop on the path to R4 (R3), the target core. R3 receives the join, caches the necessary group information, and forwards it to R4 -- the target of the join.

R4, being the target of the join, sends a JOIN_ACK back out of the receiving interface to the previous-hop sender of the join, R3. A JOIN-ACK, like a JOIN-REQUEST, is processed hop-by-hop by each router on the reverse-path of the corresponding join. The receipt of a join-ack establishes the receiving router on the corresponding CBT tree, i.e. the router becomes part of a branch on the delivery tree. Finally, R3 sends a join-ack to R1. A new CBT branch has been created, attaching subnet S1 to the CBT delivery tree for the corresponding group (footnote 2).

For the period between any CBT-capable router forwarding (or originating) a JOIN_REQUEST and receiving a JOIN_ACK the corresponding router is not permitted to acknowledge any subsequent joins received for the same group; rather, the router caches such joins till such time as it has itself received a JOIN_ACK for the original join. Only then can it acknowledge any cached joins. A router is said to be in a pending-join state if it is awaiting a JOIN_ACK itself.

Note that the presence of underlying transient asymmetric routes is irrelevant to the tree-building process; CBT tree branches are symmetric by the nature in which they are built. Joins set up transient state (incoming and outgoing interface state) in all routers along a path to a particular core. The corresponding join-ack traverses the reverse-path of the join as dictated by the transient state, and not the path that underlying routing would dictate. Whilst permanent asymmetric routes could pose a problem for CBT, transient

---

2 At this point, it is proposed that IGMP (v3) group multicasts a notification across the subnet indicating to member hosts that the delivery tree has been joined successfully. Such a message would greatly benefit multicast protocols requiring explicit joins [5, 10].

asymmetricity is detected by the CBT protocol.

2.5. Default DRs and Group DRs

The DR election mechanism does not guarantee that the DR will be the router that actually forwards a join off a multi-access network; the first hop on the path to a particular core might be via another router on the same (sub)network, which actually forwards off-subnet.

The CBT router that becomes the interface between the subnet and the rest of the CBT tree, i.e. the CBT router at which a join-ack arrives on the subnet, becomes the CBT GROUP DR. This group-specific DR (G-DR) is a token (implicit) identity. In the normal case where there is no subnet extra hop, the receipt of a JOIN-ACK means that the D-DR becomes the G-DR for the specified group.

Although very much the same, let's see another example using our example topology of figure 1 of a host joining a CBT tree for the case where more than one CBT router exists on the host subnetwork.

B's subnet, S4, has 3 CBT routers attached. Assume also that R6 has been elected IGMP-querier and CBT D-DR.

R6 (S4's D-DR) receives an IGMP group membership report. By some means, R6 discovers the <core, group> mapping for the group specified in the report; R4 is the target core for the group. R6 generates a join-request for target core R4, subcode ACTIVE_JOIN. R6's routing table says the next-hop on the path to R4 is R2, which is on the same subnet as R6. This is irrelevant to R6, which unicasts it to R2. R2 unicasts it to R3, which happens to be already on-tree for the specified group (from R1's join). R3 therefore can acknowledge the arrived join and unicast it back to R2. R2 realises it is not the origin of the corresponding join-request, but sees that the origin (R6) is on the same subnet as itself, and that over which the join-ack should be forwarded to the origin, R6. R2 unicasts the join-ack on its final hop. R2 has thus become the group's G-DR, with R6 remaining the D-DR for all groups.

If an IGMP membership report is received by a D-DR with a join for the same group already pending, or if the D-DR is already on-tree for the group, it takes no action.

2.6. Tree Teardown

There are two scenarios whereby a tree branch may be torn down:

+ During a re-configuration. If a router's best next-hop to the specified core is one of its existing children, then before sending the join it must tear down that particular downstream branch. It does so by sending a FLUSH_TREE message which is processed hop-by-hop down the branch. All routers receiving this message must process it and forward it to all their children. Routers that have received a flush message will re-establish themselves on the delivery tree if they have directly connected subnets with group presence.

+ If a CBT router has no children it periodically checks all its directly connected subnets for group member presence. If no member presence is ascertained on any of its subnets it sends a QUIT_REQUEST upstream to remove itself from the tree.

The following example, using the example topology of figure 1, shows how a tree branch is gracefully torn down using a QUIT_REQUEST.

Assume group member B leaves group G on subnet S4. B issues an IGMP HOST-MEMBERSHIP-LEAVE (relevant only to IGMPv2 and later versions) message which is multicast to the "all-routers" group (224.0.0.2). R6, the subnet's D-DR and IGMP-querier, responds with a group-specific-QUERY. No hosts respond within the required response interval, so D-DR assumes group G traffic is no longer wanted on subnet S4.

Since R6 has no CBT children, and no other directly attached subnets with group G presence, it immediately follows on by sending a QUIT_REQUEST to R2, its parent on the tree for group G. R2 responds with a QUIT-ACK, unicast to R6; R2 removes the corresponding child information. R2 in turn sends a QUIT upstream to R3 (since it has no other children or subnet(s) with group presence).

> NOTE: immediately subsequent to sending a QUIT-REQUEST, the sender removes the corresponding parent information, i.e. it does not wait for the receipt of a QUIT-ACK.

R3 responds to the QUIT by unicasting a QUIT-ACK to R2. R3 subsequently checks whether it in turn can send a quit by checking group G presence on its directly attached subnets, and any group G children. It has the latter (R1 is its child on the group G tree), and so R3

cannot itself send a quit. However, the branch R3-R2-R6 has been removed from the tree.

3. Data Packet Forwarding Rules

When a router receives (non-locally originated) data packets for forwarding over directly attached member subnets, it only does so over the set of outgoing member subnets (interfaces) for which that router is DR, irrespective of whether group membership is registered on other local interfaces. In addition, in native mode, packets are forwarded over any remaining interfaces specified by the FIB entry for the group that are not in the above set (excluding the incoming interface). In CBT mode, encapsulated data packets are forwarded over the full set of interfaces specified by the FIB entry, except the incoming interface.

A router only forwards data packets originated by directly attached hosts iff the router is the DR on the interface over which those packets were received.

4. Data Packet Forwarding -- Encapsulation Details

In "native mode" all data packets are forwarded over CBT tree interfaces as native IP multicasts, i.e. there are no encapsulations required. This assumes that CBT is the multicast routing protocol in operation within the domain (or "cloud") in question, and that all routers within the domain of operation are CBT-capable, i.e. there are no "tunnels".

In a multi-protocol environment, whose infrastructure may include non-multicast-capable routers, it is necessary to tunnel data packets between CBT-capable routers. This is called "CBT mode". Data packets are de-capsulated by CBT routers (such that they become native mode data packets) before being forwarded over subnets with member hosts. When multicasting (native mode) to member hosts, the TTL value of the original IP header is set to one. CBT mode encapsulation is as follows:

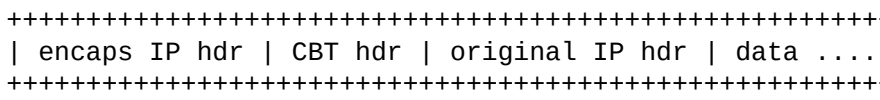

Figure 2. Encapsulation for CBT mode

The TTL value of the CBT header is set by the encapsulating CBT router directly attached to the origin of a data packet. This value is decremented each time it is processed by a CBT router. An encapsulated data packet is discarded when the CBT header TTL value reaches zero.

The purpose of the (outer) encapsulating IP header is to "tunnel" data packets between CBT-capable routers (or "islands"). The outer IP header's TTL value is set to the "length" of the corresponding tunnel, or MAX_TTL (255)if this is not known, or subject to change.

For native mode IP multicasts, i.e. those without any extra encapsulation, the TTL value of the IP header is decremented each time the packet is received by a multicast router.

It is worth pointing out here the distinction between subnetworks and tree branches, although they can be one and the same. For example, a multi-access subnetwork containing routers and end-systems could potentially be both a CBT tree branch and a subnetwork with group member presence. A tree branch which is not simultaneously a subnetwork is either a "tunnel" or a point-to-point link.

In CBT mode there are three forwarding methods used by CBT routers:

+ IP multicasting. This method is used to send a data packet across a directly-connected subnetwork with group member presence. System host changes are not required for CBT. Similarly, end-systems originating multicast data do so in traditional IP-style.

+ CBT unicasting. This method is used for sending data packets encapsulated (as illustrated above) across a tunnel or point-to-point link. En/de-capsulation takes place in CBT routers.

+ CBT multicasting. Routers on multi-access links use this method to send data packets encapsulated (as illustrated above) but the

outer encapsulating IP header contains a multicast address. This method is used when a parent or multiple children are reachable over a single physical interface, as could be the case on a multi-access Ethernet. The IP module of end-systems subscribed to the same group will discard these multicasts since the CBT payload type (protocol id) of the outer IP header is not recognizable by hosts.

CBT routers create Forwarding Information Base (FIB) entries whenever they send or receive a JOIN_ACK. The FIB describes the parent-child relationships on a per-group basis. A FIB entry dictates over which tree interfaces, and how (unicast or multicast) a data packet is to be sent. Additionally, a data packet is IP multicast over any directly-connected subnetworks with group member presence. Such interfaces are kept in a separate table relating to IGMP. A FIB entry is shown below:

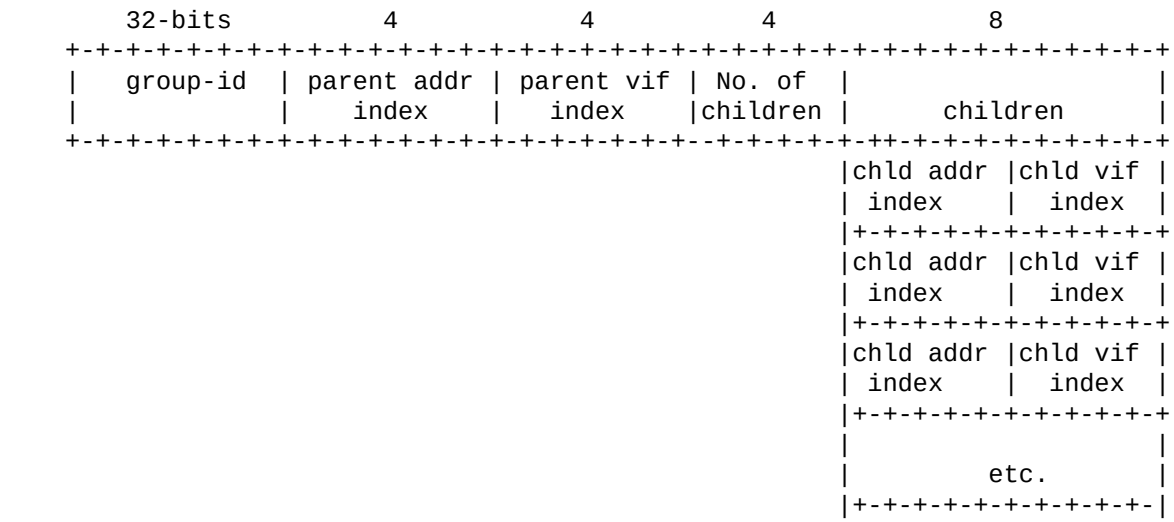

Figure 3. CBT FIB entry

Note that a CBT FIB is required for both CBT-mode and native-mode multicasting.

The field lengths shown above assume a maximum of 16 directly connected neighbouring routers.

When a data packet arrives at a CBT router, the following rules apply:

+ if the packet is an IP-style multicast, it is checked to see if it originated locally (i.e. if the arrival interface subnetmask bitwise ANDed with the packet's source IP address equals the arrival interface's subnet number, then the packet was sourced locally). If the packet is not of local origin, it is discarded.

+ the packet is IP multicast to all directly connected subnets with group member presence. The packet is sent with an IP TTL value of 1 in this case.

+ the packet is encapsulated for CBT forwarding (see figure 2) and unicast to parent and children. However, if more than one child is reachable over the same interface the packet will be CBT multicast. Therefore, it is possible that an IP-style multicast and a CBT multicast will be forwarded over a particular subnetwork.

NOTE: the TTL value of encapsulated data packets is manipulated as described at the beginning of this section.

Using our example topology in figure 1, let's assume member G originates an IP multicast packet. R8 is the DR for subnet S10. R8 CBT unicasts the packet to each of its children, R9 and R12. These children are not reachable over the same interface. R8, being the DR for subnets S14 and S10 also IP multicasts the packet to S14 (S10 received the IP style packet already from the originator). R9, the DR for S12, need not IP multicast onto S12 since there are no members present there. R9 CBT unicasts the packet to R10, which is the DR for S13 and S15. It IP multicasts to both S13 and S15.

Going upstream from R8, R8 CBT unicasts to R4. It is DR for all directly connected subnets and therefore IP multicasts the data packet onto S5, S6 and S7, all of which have member presence. R4 unicasts the packet to all outgoing children, R3 and R7 (NOTE: R4 does not have a parent since it is the primary core router for the group). R7 IP multicasts onto S9. R3 CBT unicasts to R1 and R2, its children. Finally, R1 IP multicasts onto S1 and S3, and R2 IP multicasts onto S4.

4.1. Non-Member Sending

For a multicast data packet to span beyond the scope of the originating subnetwork at least one CBT-capable router must be present on that subnetwork. The default DR (D-DR) for the group on the subnetwork must encapsulate the (native) IP-style packet and unicast it to a core for the group. In native mode this encapsualation constitutes IP-in-IP. In CBT mode, the encapsulation required is shown in figure 2. In both cases, CBT routers are required to know <core, group> mappings. The alternatives for discovering these are discussed in section 2.1. Beyond this, this topic is beyond the scope of this document.

5. Eliminating the Topology-Discovery Protocol in the Presence of Tunnels

Traditionally, multicast protocols operating within a virtual topology, i.e. an overlay of the physical topology, have required the assistance of a multicast topology discovery protocol, such as that present in DVMRP. However, it is possible to have a multicast protocol operate within a virtual topology without the need for a multicast topology discovery protocol. One way to achieve this is by having a router configure all its tunnels to its virtual neighbours in advance. A tunnel is identified by a local interface address and a remote interface address. Routing is replaced by "ranking" each such tunnel interface associated with a particular core address; if the highest-ranked route is unavailable (tunnel end-points are required to run an Hello-like protocol between themselves) then the next-highest ranked available route is selected, and so on. The exact specification of the Hello protocol is outside the scope of this document.

CBT trees are built using the same join/join-ack mechanisms as before, only now some branches of a delivery tree run in native mode, whilst others (tunnels) run in CBT mode. Underlying unicast routing dictates which interface a packet should be forwarded over. Each interface is configured as either native mode or CBT mode, so a packet can be encapsulated (decapsulated) accordingly.

As an example, router R’s configuration would be as follows:

intf	type	mode	remote addr
#1	phys	native	-
#2	tunnel	cbt	128.16.8.117
#3	phys	native	-
#4	tunnel	cbt	128.16.6.8
#5	tunnel	cbt	128.96.41.1

core	backup-intfs
A	#5, #2
B	#3, #5
C	#2, #4

The CBT FIB needs to be slightly modified to accommodate an extra field, "backup-intfs" (backup interfaces). The entry in this field specifies a backup interface whenever a tunnel interface specified in the FIB is down. Additional backups (should the first-listed backup be down) are specified for each core in the core backup table. For example, if interface (tunnel) #2 were down, and the target core of a CBT control packet were core A, the core backup table suggests using interface #5 as a replacement. If interface #5 happened to be down also, then the same table recommends interface #2 as a backup for core A.

6. Tree Maintenance

Once a tree branch has been created, i.e. a CBT router has received a JOIN_ACK for a JOIN_REQUEST previously sent (forwarded), a child router is required to monitor the status of its parent/parent link at fixed intervals by means of a ‘‘keepalive’’ mechanism operating between them. The ‘‘keepalive’’ mechanism is implemented by means of two CBT control messages: CBT_ECHO_REQUEST and CBT_ECHO_REPLY. Adjacent CBT routers only need to send one keepalive per link, regardless of how many groups are present on that link. This aggregation strategy is expected to conserve considerable bandwidth on "busy" links, such as those nearer the "centre" of the network.

The keepalive protocol is simple, as follows: a child unicasts a CBT-ECHO-REQUEST to its parent, which unicasts a CBT-ECHO-REPLY in response.

For any CBT router, if its parent router, or path to the parent, fails, the child is initially responsible for re-attaching itself, and therefore all routers subordinate to it on the same branch, to the tree.

6.1. Router Failure

An on-tree router can detect a failure from the following two cases:

+ if the child responsible for sending keepalives across a particular link stops receiving CBT_ECHO_REPLY messages. In this case the child realises that its parent has become unreachable and must therefore try and re-connect to the tree for all groups represented on the parent/child link. Until an aggregation strategy is fully worked out, a (re)join must be sent for each group individually. (We present some ideas on rejoin aggregation in Appendix A).

  The rejoining router (that which is immediately subordinate to the failure) sends a JOIN_REQUEST (subcode ACTIVE_JOIN if it has no children attached, and subcode ACTIVE_REJOIN if at least one child is attached) to the best next-hop router on the path to the elected core. If no JOIN-ACK is received after three retransmissions, each transmission being at PEND-JOIN-INTERVAL (10 secs), an alternate core is elected from the core list, and the process repeated. If all cores have been tried unsuccessfully, the D-DR has no option but to give up.

+ if a parent stops receiving CBT_ECHO_REQUESTs from a child. In this case the parent simply removes the child interface from FIB entries that are represented by that parent/child link.

6.2. Router Re-Starts

There are two cases to consider here:

+ Core re-start. All JOIN-REQUESTs (all types) carry the identities (i.e. addresses) of each of the cores for a group. If a router is a core for a group, but has only recently re-started, it will not be aware that it is a core for any group(s). In such circumstances, a core only becomes aware that it is such by

receiving a JOIN-REQUEST. Subsequent to a core learning its status in this way, if it is not the primary core it acknowledges the received join, then sends a JOIN_REQUEST (subcode ACTIVE_REJOIN) to the primary core. If the re-started router is the primary core, it need take no action, i.e. in all circumstances, the primary core simply waits to be joined by other routers.

+ Non-core re-start. In this case, the router can only join the tree again if a downstream router sends a JOIN_REQUEST through it, or it is elected DR for one of its directly attached subnets, and subsequently receives an IGMP membership report.

6.3. Route Loops

Routing loops are only a concern when a router with at least one child is attempting to re-join a CBT tree. In this case the re-joining router sends a JOIN_REQUEST (subcode ACTIVE REJOIN) to the best next-hop on the path to an elected core. This join is forwarded as normal until it reaches either the specified core, another core, or a non-core router that is already part of the tree. If the rejoin reaches the primary core, loop detection is not necessary. The primary core acks an active-rejoin by means of a JOIN-ACK, subcode PRIMARY-REJOIN-ACK. This ack must be processed by each router on the reverse-path of the active-rejoin. If an active-rejoin is terminated by any router on the tree other than the primary core, loop detection must take place, as we now describe.

If, in response to an active-rejoin, a JOIN-ACK is returned, subcode NORMAL (as opposed to an ack with subcode PRIMARY-REJOIN-ACK), the router receiving the ack subsequently generates a JOIN-REQUEST, subcode NACTIVE-REJOIN (non-active rejoin). This packet serves only to detect loops; it does not create any transient state in the routers it traverses, other than the originating router. Any on-tree router receiving a non-active rejoin is required to forward it over its parent interface for the specified group. In this way, it will either reach the primary core, which returns, directly to the sender, a join ack with subcode PRIMARY-NACTIVE-ACK (so the sender knows no loop is present), or the sender receives the non-active rejoin it sent, via one of its child interfaces, in which case the rejoin obviously formed a loop.

If a loop is present, the non-active join originator immediately

sends a QUIT_REQUEST to its newly-established parent and the loop is broken.

Using figure 4 (over) to demonstrate this, if R3 is attempting to re-join the tree (R1 is the core in figure 4) and R3 believes its best next-hop to R1 is R6, and R6 believes R5 is its best next-hop to R1, which sees R4 as its best next-hop to R1 -- a loop is formed. R3 begins by sending a JOIN_REQUEST (subcode ACTIVE_REJOIN, since R4 is its child) to R6. R6 forwards the join to R5. R5 is on-tree for the group, so responds to the active-rejoin with a JOIN-ACK, subcode NORMAL (the ack traverses R6 on its way to R3). R3 now generates a JOIN-REQUEST, subcode NACTIVE-REJOIN, and forwards this to its parent, R6. R6 forwards the non-active rejoin to R5, its parent. R5 does similarly, as does R4. Now, the non-active rejoin has reached R3, which originated it, so R3 concludes a loop is present on the parent interface for the specified group. It immediately sends a QUIT_REQUEST to R6, which in turn sends a quit if it has not received an ACK from R5 already AND has itself a child or subnets with member presence. If so it does not send a quit -- the loop has been broken by R3 sending the first quit.

QUIT_REQUESTs are typically acknowledged by means of a QUIT_ACK. A child removes its parent information immediately subsequent to sending its first QUIT-REQUEST. The ack here serves to notify the (old) child that it (the parent) has in fact removed its child information. However, there might be cases where, due to failure, the parent cannot respond. The child sends a QUIT-REQUEST a maximum of three times, at PEND-QUIT-INTERVAL (10 sec) intervals.

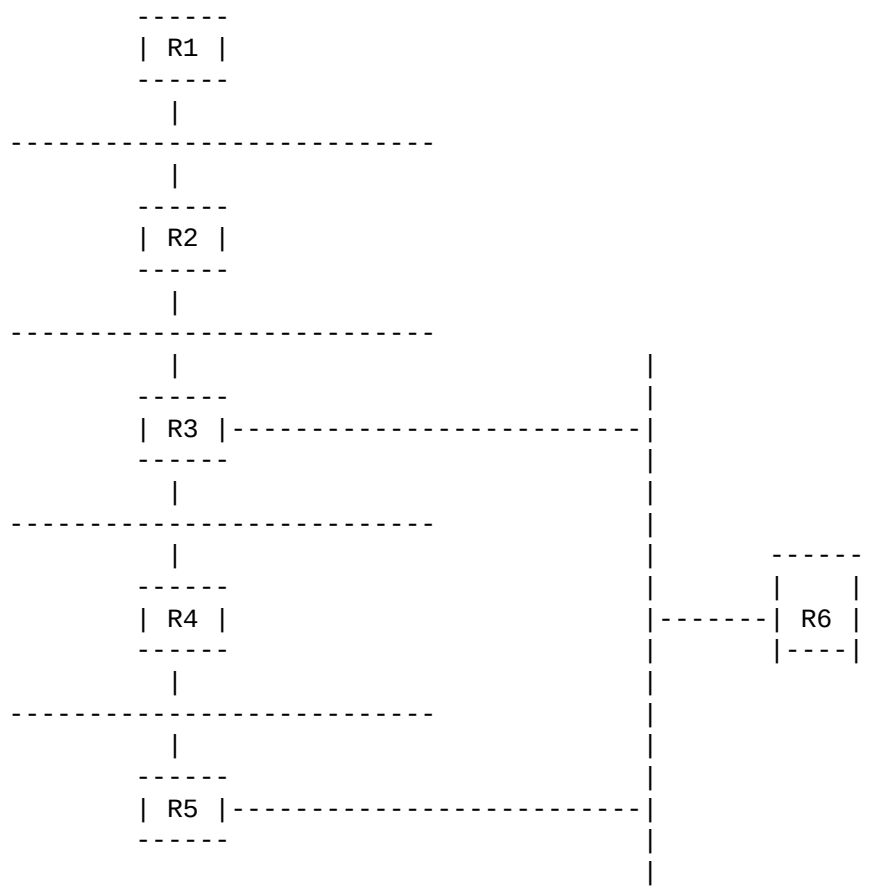

Figure 4: Example Loop Topology

In another scenario the rejoin travels over a loop-free path, and the first on-tree router encountered is the primary core, R1. In figure 4, R3 sends a join, subcode REJOIN_ACTIVE to R2, the next-hop on the path to core R1. R2 forwards the re-join to R1, the primary core, which returns a JOIN-ACK, subcode PRIMARY-REJOIN-ACK, over the reverse-path of the rejoin-active. Whenever a router receives a PRIMARY-REJOIN-ACK no loop detection is necessary.

If we assume R2 is on tree for the corresponding group, R3 sends a join, subcode REJOIN_ACTIVE to R2, which replies with a join ack,

subcode NORMAL. R3 must then generate a loop detection packet (join request, subcode REJOIN-NACTIVE) which is forwarded to its parent, R2, which does similarly. On receipt of the rejoin-Nactive, the primary core unicasts a join ack back directly to R3, with subcode PRIMARY-NACTIVE-ACK. This confirms to R3 that its rejoin does not form a loop.

7. Data Packet Loops

The CBT protocol builds a loop-free distribution tree. If all routers that comprise a particular tree function correctly, data packets should never traverse a tree branch more than once.

CBT routers will only forward native-style data packets if they are received over a valid on-tree interface. A native-style data packet that is not received over such an interface is discarded.

Encapsulated CBT data packets from a non-member sender can arrive via an "off-tree" interface (this is how CBT-mode sends data across tunnels, and how data from non-member senders in native-mode or CBT-mode reaches a tree). The encapsulating CBT data packet header includes an "on-tree" field, which contains the value 0x00 until the data packet reaches an on-tree router. At this point, the router must convert this value to 0xff to indicate the data packet is now on-tree. This value remains unchanged, and from here on the packet should traverse only on-tree interfaces. If an encapsulated packet happens to "wander" off-tree and back on again, the latter on-tree router will receive the CBT encapsulated packet via an off-tree interface. However, this router will recognise that the "on-tree" field of the encapsulating CBT header is set to 0xff, and so immediately discards the packet.

8. CBT Packet Formats and Message Types

CBT packets travel in IP datagrams. We distinguish between two types of CBT packet: CBT data packets, and CBT control packets. CBT control packets carry a CBT control header. All CBT control messages are implemented over UDP. CBT mode data (figure 2) requires a CBT data packet header.

8.1. CBT Header Format (for CBT Mode data)

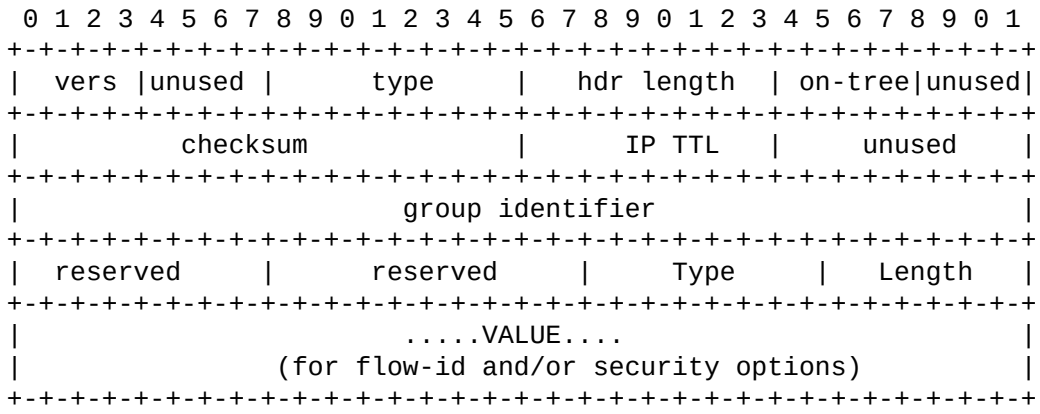

Figure 5. CBT Header

Each of the fields is described below:

+ Vers: Version number -- this release specifies version 1.

+ type: indicates CBT payload is data. The only value defined for this field is 255 (0xff).

+ hdr length: length of the header, for purpose of checksum calculation.

+ on-tree: indicates whether the packet is on-tree (0xff) or off-tree (0x00). Once this field is set (i.e. on-tree), it is non-changing. This field can only be set by a router that has a FIB entry for the corresponding group, i.e. a router that has received a join-ack for a join-request previously sent/forwarded.

+ checksum: the 16-bit one’s complement of the one’s complement of the CBT header, calculated across all fields.

+ IP TTL: TTL value gleaned from the IP header where the packet originated. It is decremented each time it traverses a CBT router.

+ group identifier: multicast group address.

+ The TLV fields at the end of the header are for a flow-identifier, and/or security options, if and when implemented. A "type" value of zero implies a "length" of zero, implying there is no "value" field.

8.2. Control Packet Header Format

The individual fields are described below. It should be noted that only certain fields beyond ''group identifier'' are processed for the different control messages.

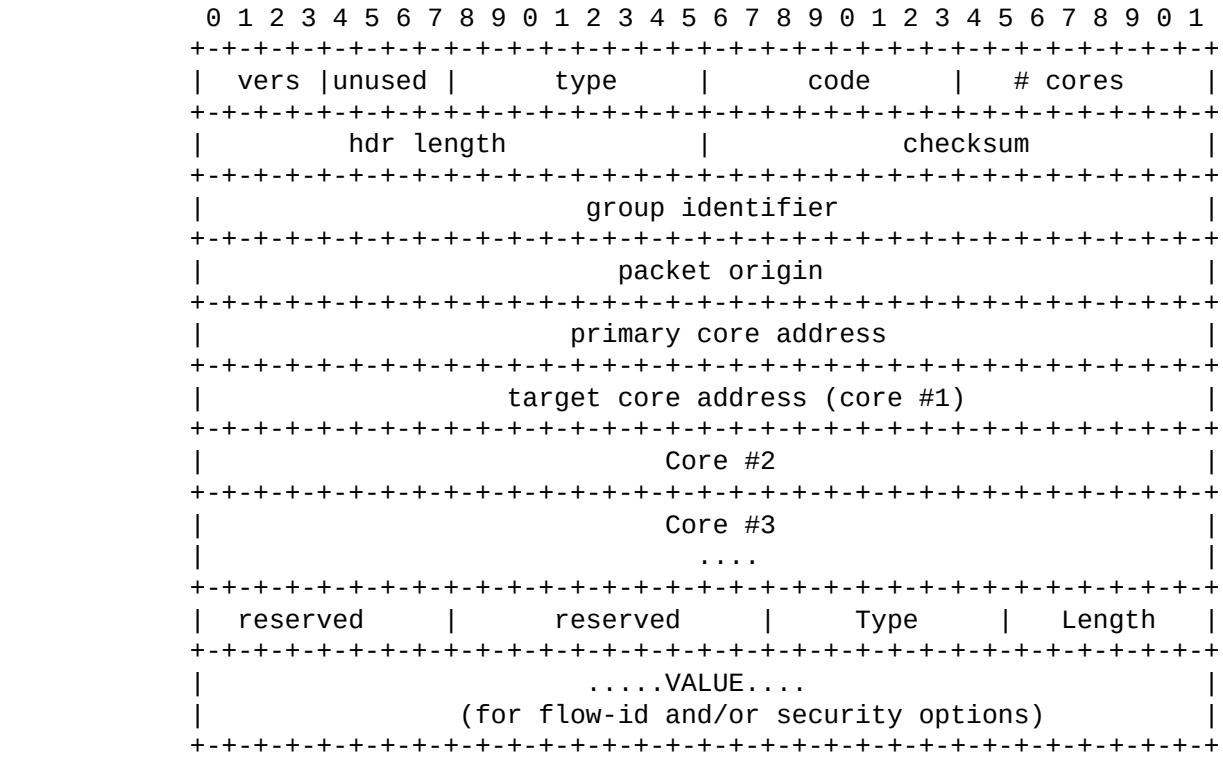

Figure 6. CBT Control Packet Header

+ Vers: Version number -- this release specifies version 1.

+ type: indicates control message type (see sections 7.3, 7.3.1).

+ code: indicates subcode of control message type.

+ # cores: number of core addresses carried by this control packet (does not include "primary core address" field).

+ header length: length of the header, for purpose of checksum calculation.

+ checksum: the 16-bit one's complement of the one's complement of the CBT control header, calculated across all fields.

+ group identifier: multicast group address.

+ packet origin: address of the CBT router that originated the control packet.

+ primary core address: the address of the primary core for the group.

+ target core address: desired core affiliation of control message.

+ Core #Z: Z refers to some arbitrary IP address representing a core.

+ The TLV fields at the end of the header are for a flow-identifier, and/or security options, if implemented. A "type" value of zero implies a "length" of zero, implying there is no "value" field.

8.3. CBT Control Message Types

There are eight types of CBT message. All are encoded in the CBT control header, shown in figure 6.

+ JOIN-REQUEST (type 1): generated by a router and unicast to the specified core address. It is processed hop-by-hop on its

way to the specified core. Its purpose is to establish the sending CBT router, and all intermediate CBT routers, as part of the corresponding delivery tree. Note that all cores are carried in join-requests.

+ JOIN-ACK (type 2): an acknowledgement to the above. The full list of core addresses is carried in a JOIN-ACK, together with the actual core affiliation (the join may have been terminated by an on-tree router on its journey to the specified core, and the terminating router may or may not be affiliated to the core specified in the original join). A JOIN-ACK traverses the same path as the corresponding JOIN-REQUEST, with each CBT router on the path processing the ack. It is the receipt of a JOIN-ACK that actually creates a tree branch.

+ JOIN-NACK (type 3): a negative acknowledgement, indicating that the tree join process has not been successful.

+ QUIT-REQUEST (type 4): a request, sent from a child to a parent, to be removed as a child to that parent.

+ QUIT-ACK (type 5): acknowledgement to the above. If the parent, or the path to it is down, no acknowledgement will be received within the timeout period. This results in the child nevertheless removing its parent information.

+ FLUSH-TREE (type 6): a message sent from parent to all children, which traverses a complete branch. This message results in all tree interface information being removed from each router on the branch, possibly because of a re-configuration scenario.

+ CBT-ECHO-REQUEST (type 7): once a tree branch is established, this messsage acts as a ‘‘keepalive’’, and is unicast from child to parent (one per link, NOT one per group).

+ CBT-ECHO-REPLY (type 8): positive reply to the above.

8.3.1. CBT Control Message Subcodes

The JOIN-REQUEST has three valid subcodes:

+ ACTIVE-JOIN (code 0) - sent from a CBT router that has no children for the specified group.

+ REJOIN-ACTIVE (code 1) - sent from a CBT router that has at least one child for the specified group.

+ REJOIN-NACTIVE (code 2) - generated by a router subsequent to receiving a join ack, subcode NORMAL, in response to a active-rejoin.

A JOIN-ACK has three valid subcodes:

+ NORMAL (code 0) - sent by a core router, or on-tree non-core router acknowledging joins with subcodes ACTIVE-JOIN and REJOIN-ACTIVE.

+ PRIMARY-REJOIN-ACK (code 1) - sent by a primary core to acknowledge the receipt of a join-request received with subcode REJOIN-ACTIVE. This message traverses the reverse-path of the corresponding re-join, and is processed by each router on that path.

+ PRIMARY-NACTIVE-ACK (code 2) - sent by a primary core to acknowledge the receipt of a join-request received with subcode REJOIN-NACTIVE. This ack is unicast directly to the router that generated the rejoin-Nactive, i.e. the ack it is not processed hop-by-hop.

9. CBT Protocol and Port Numbers

CBT mode (data) encapsulation (figure 2) requires an IP protocol number assignment for CBT. An official protocol number has recently been approved by the IANA; CBT has IP protocol number 7.

CBT control packets travel inside UDP datagrams, as the following diagram illustrates:

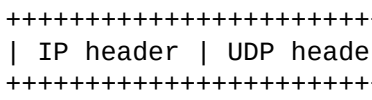

Figure 7. Encapsulation for CBT control messages

CBT therefore requires a UDP port assignment for control messages. An official UDP port number has recently been approved by the IANA; CBT control messages are received on UDP port 7777.

10. Default Timer Values

There are several CBT control messages which are transmitted at fixed intervals. These values, retransmission times, and timeout values, are given below. Note these are recommended default values only, and are configurable with each implementation (all times are in seconds):

+ CBT-ECHO-INTERVAL 30 (time between sending successive CBT-ECHO-REQUESTs to parent).

+ PEND-JOIN-INTERVAL 10 (retransmission time for join-request if no ack rec'd)

+ PEND-JOIN-TIMEOUT 30 (time to try joining a different core, or give up)

+ EXPIRE-PENDING-JOIN 90 (remove transient state for join that has not been ack'd)

+ PEND_QUIT_INTERVAL 10 (retransmission time for quit-request if no ack rec'd)

+ CBT-ECHO-TIMEOUT 90 (time to consider parent unreachable)

+ CHILD-ASSERT-INTERVAL 90 (increment child timeout if no ECHO rec'd from a child)

+ CHILD-ASSERT-EXPIRE-TIME 180 (time to consider child gone)

+ IFF-SCAN-INTERVAL 300 (scan all interfaces for group presence. If none, send QUIT)

11. Interoperability Issues

One of the design goals of CBT is for it to fully interwork with other IP multicast schemes. We have already described how CBT-style packets are transformed into IP-style multicasts, and vice-versa.

In order for CBT to fully interwork with other schemes, it is necessary to define the interface(s) between a ''CBT cloud'' and the cloud of another scheme. The CBT authors are currently working out the details of interoperability, and we expect an interoperability document to be available shortly.

12. CBT Security Architecture

see current I-D: ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-mkd-01.{ps,txt}

Acknowledgements

Special thanks goes to Paul Francis, NTT Japan, for the original brainstorming sessions that brought about this work.

Thanks too to Sue Thompson (Bellcore). Her detailed reviews led to the identification of some subtle protocol flaws, and she suggested several simplifications.

Thanks also to the networking team at Bay Networks for their comments and suggestions, in particular Steve Ostrowski for his suggestion of using "native mode" as a router optimization, and Eric Crawley.

Thanks also to Ken Carlberg (SAIC) for reviewing the text, and generally providing constructive comments throughout.

I would also like to thank the participants of the IETF IDMR working group meetings for their general constructive comments and suggestions since the inception of CBT.

APPENDIX A

A single rejoin could be sent for all the groups the keepalive represents. This constitutes an aggregated rejoin strategy; a single rejoin message can serve to rejoin multiple groups to their respective trees, provided those groups share a common core (that which is being rejoined). Therefore, it may be that several rejoins need to be sent to re-connect all groups traversing the router after a failure. Similarly, the corresponding join-ack would represent an aggregate.

NOTE: it remains to be worked out how the new parent establishes from the aggregated rejoin all those groups which the rejoin represents (so the new parent can create/modify the necessary FIB entries). A "group aggregate" field may be necessary in the control packet. Alternatively, when the ack is received in response to the rejoin, each group represented by the rejoin sends a group-specific echo until an ack is received for each.

Authors' Addresses:

Tony Ballardie,
Department of Computer Science,
University College London,
Gower Street,
London, WC1E 6BT,
ENGLAND, U.K.

Tel: ++44 (0)71 419 3462
e-mail: A.Ballardie@cs.ucl.ac.uk

Scott Reeve,
Bay Networks, Inc.
3, Federal Street,
Billerica, MA 01821,
USA.

Tel: ++1 508 670 8888
e-mail: sreeve@BayNetworks.com

Nitin Jain,
Bay Networks, Inc.
3, Federal Street,

Billerica, MA 01821,
USA.

Tel: ++1 508 670 8888
e-mail: njain@BayNetworks.com

References

[1] DVMRP. Described in "Multicast Routing in a Datagram Internetwork", S. Deering, PhD Thesis, 1990. Available via anonymous ftp from: gregorio.stanford.edu:vmtp/sd-thesis.ps.

[2] J. Moy. Multicast Routing Extensions to OSPF. Communications of the ACM, 37(8): 61-66, August 1994.

[3] D. Farinacci, S. Deering, D. Estrin, and V. Jacobson. Protocol Independent Multicast (PIM) Dense-Mode Specification (draft-ietf-idmr-pim-spec-01.ps). Working draft, 1994.

[4] A. J. Ballardie. Scalable Multicast Key Distribution (ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-mkd-01.{ps,txt}). Working draft, 1995.

[5] A. J. Ballardie. "A New Approach to Multicast Communication in a Datagram Internetwork", PhD Thesis, 1995. Available via anonymous ftp from: cs.ucl.ac.uk:darpa/IDMR/ballardie-thesis.ps.Z.

[6] W. Fenner. Internet Group Management Protocol, version 2 (IGMPv2), (draft-idmr-igmp-v2-01.txt).

[7] B. Cain, S. Deering, A. Thyagarajan. Internet Group Management Protocol Version 3 (IGMPv3) (draft-cain-igmp-00.txt).

[8] M. Handley, J. Crowcroft, I. Wakeman. Hierarchical Rendezvous Point proposal, work in progress.
(http://www.cs.ucl.ac.uk/staff/M.Handley/hpim.ps) and
(ftp://cs.ucl.ac.uk/darpa/IDMR/IETF-DEC95/hpim-slides.ps).

[9] D. Estrin et al. USC/ISI, Work in progress.
(http://netweb.usc.edu/pim/).

[10] D. Estrin et al. PIM Sparse Mode Specification. (draft-ietf-idmr-pim-sparse-spec-00.txt).