November 21st, 1995

Core Based Trees (CBT) Multicast

-- Protocol Specification --

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts).

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

Abstract

This document describes the Core Based Tree (CBT) 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 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 primary

differences between this latest specification and the previous release (June 1995).

The CBT architecture is described in an accompanying document: 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. _A_u_t_h_o_r_s' _N_o_t_e

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

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 the group joining process is as transparent as possible for new receivers; ideally, from a user perspective, only a minimum of information should be required in order to join a CBT group -- the knowledge/input of two group parameters, group address and TTL value, is a reasonable expectation. At the same time, we strive to keep join latency to an absolute minimum.

The factor most affecting join latency in CBT is the mechanism by which each group on a LAN elects a so-called designated router (DR). This mechanism has now been re-invented, being simpler, and keeps join latency to a minimum. This new DR election process is explained in section 2.3.

Core selection, placement, and management have prevented a simple group initiation/joining process, inherent in data-driven schemes (like DVMRP); some network entity needs to elect a group's cores, and a mechanism is needed to distribute this information throughout the network so it is available to potential new receivers.

CBT separates out most aspects of core management from the protocol itself. This has been made easier due to the fact that 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. 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, making it possible for the non-primary cores to know where to join.

CBT now supports the aggregation of certain types of control message on distribution trees, provided aggregation is at all possible. This depends on coordinated multicast address assignment.

Also catalytic in the simplification of the CBT protocol are the "multi-protocol support" aspects of the latest proposal of IGMP (IGMPv3 [6]), in particular, the introduction of the RP/Core-Report message (see Appendix and [6]).

The end result of these developments is that the CBT protocol is further simplified and more efficient; six message types have been eliminated from the previous version of the protocol, thereby reducing protocol overhead. Furthermore, the new DR election mechanism ensures group join latency is kept to a minimum.

Throughout this draft, we assume IGMPv3 is operating between hosts and routers on a LAN.

2. Protocol Specification

2.1. CBT Group Initiation

A group's initiator elects a small number of candidate cores (which may be advertised by "some means"). Subsequently, the core distribution engine (if available) is notified of the new group now associated with the elected cores. Subsequent network advertisements provide the <core,group> mapping information for potential new senders and/or receivers.

_2._2. _T_r_e_e _J_o_i_n_i_n_g _P_r_o_c_e_s_s -- _O_v_e_r_v_i_e_w

It is assumed that hosts receive <core,group> mapping advertisements via some protocol external to CBT. Given this assumption, the following steps are involved in a host joining a CBT tree:

o+ the joining host learns of the candidate cores for the group.

o+ subsequently, an IGMP RP/Core-Report is issued on the subnetwork, addressed to the corresponding multicast group.

  All IGMP messages are received by all operational CBT multicast routers on the subnetwork. One CBT-capable router per subnetwork is initially elected as the default LAN CBT DR (DEFAULT DR) for all groups. This election happens automatically when CBT routers are initialised. If the subnetwork has multiple CBT routers present, a (possibly different) group-specific DR (GROUP DR) may subsequently be elected. This is fully explained in section 2.3.

o+ on receiving an IGMP RP/Core-Report, the local DR takes care of establishing the subnet as part of the corresponding CBT delivery tree.

The following CBT control messages come into play during the host joining process:

o+ JOIN_REQUEST

o+ 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 simply fixes that state.

_2._3. _D_R _E_l_e_c_t_i_o_n

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 for generating join-requests in response to receiving IGMP group membership reports. Such joins are forwarded upstream by the DR.

At start-up, a CBT router assumes it is the only CBT-capable router on its subnetwork. It therefore sends two or three IGMP-HOST-MEMBERSHIP-QUERYs in short succession (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 (exception, next para) the subnet's IGMP-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.

On multi-access LANs where different routers may be running different multicast routing protocols, there may be times when a LAN's (subnet's) elected querier is a non-CBT router. CBT routers keep track of their immediate CBT neighbouring routers, and can therefore easily establish if the source of an IGMP query is CBT-capable or not. If an elected querier is not CBT-capable, the DR is (implicitly) elected to be the lowest-addressed neighbour on the same link; if a CBT router on such a link knows of a lower-addressed neighbour on the same link, it either does not attempt to claim DR status, or relinquishes its DR status if it was previously elected DR.

_2._4. _B_a_c_k_w_a_r_d_s _C_o_m_p_a_t_i_b_i_l_i_t_y _w_i_t_h _I_G_M_P_v_1 & _v_2 _H_o_s_t_s

To comply with this specification, CBT routers are expected to run IGMP version 3 [7]. However, it cannot be assumed that all hosts on a subnetwork will be running IGMPv3; there may be instances of IGMP versions 1 and/or 2.

IGMPv1 & v2 hosts will not be able to issue RP/Core Reports,

available with IGMPv3. The implications of this primarily mean that such hosts must inform a D-DR of <core, group> mappings by means of network management. Alternatively, hosts may implement minimal user-level code to emulate IGMPv3-specific messages, and send them as CBT auxiliary control messages to the specified group address.

NOTE: one recent core distribution proposal [8] does not require hosts to participate in core election at all. Rather, a local DR is configured to know a set of core addresses in the lowest level of a core hierarchy, and a function is used to map a group address onto a particular core in the hierarchy.

2.5. 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 specified core. Core information is gleaned either by means of an IGMP RP/Core Report, also sent in response to an IGMP host membership query, but prior to an IGMP host membership report, or by some other means.

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, the join is not propogated further, but ACK'd 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 (the first listed in the core listing, contained within the join) 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. Any such router other than the primary core proceeds by transforming the rejoin into a REJOIN-NACTIVE for loop detection. This is described in section 6.3.

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 are prefixed with S.

```
           A                               B
           |   S1              S4          |
   -------------------      ------------------------------------------------
             |                     |               |               |
           ------                 ------           ------           ------
           | R1 |                 | R2 |           | R5 |           | R6 |
           ------                 ------           ------           ------
      C     |  |                    |                |                 |
      |     |  |                    |    S2          |            S8   |
   ----------  ------------------------------------------         ------------
        S3                 |
                         ------
                         | R3 |
                 |       ------                       D
   | S9          |         |               S5         |
   |             |      ---------------------------------------------
   |  |----|     |                    |
   ---| R7 |-----|                  ------
   |  |----|     |------------------| R4 |
   |          S7 |                  ------            F
   |             |                    |         S6    |
   |-E           |            ---------------------------------
                      |                       |
                      |                     ------
             |---|    |---------------------| R8 |
             |R12 -----|                    ------      G
             |---|    |                       |         |  S10
                      | S14                 ----------------------------
                      |                         |
                  I --|                       ------
                      |                       | R9 |
                                              ------
                                                |         S12
                     |             ----------------------------
                 S15 |                        |
                     |                      ------
                     |----------------------|R10 |
                J ---|                      ------      H
                     |                        |         |
                     |             ----------------------------
                     |                           S13
```

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. The <core,group> mapping is subsequently advertised by some (possibly same) protocol.

Host A generates an IGMP RP/Core-Report and an IGMP group membership report when the multicast application is invoked on host A. Both reports are multicast to the corresponding group address. All multicast routers receive all multicast-addressed messages by default. The only CBT router on A's subnet (S1) is R1, which is, by default, the D-DR.

Router R1, receives the RP/Core-Report and the group 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 in the RP/Core Report. 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 JOIN-REQUESTs, 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. R3 sends a join-ack to R2, which sends a joinj-ack to R1. A new CBT branch has been created, attaching subnet S1 to the CBT delivery tree for the corresponding group.

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].

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.

2.6. D-DRs, G-DRs, and Proxy-acks

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-LAN. It is not necessary or desirable to have a tree branch rooted anywhere other than at a router that is the interface to and from the LAN; only this router need keep group state information, the join originator (D-DR) need not since the first hop is on the same LAN. Because of this, CBT incorporates a simple mechanism that prevents the D-DR in such scenarios from keeping group state.

If a join-ack has returned to the originating subnet of the corresponding join, but has not yet reached the originating router of the corresponding join, obviously the join-request's first hop is on the same subnet as the originating router (the D-DR). A router knows when it is in this situation by extracting the origin router's subnet address using its own subnet mask, then comparing the result with its own address (using address and mask of the subnet that is about to be forwarded over). If one further hop is required for the join-ack to reach the originator of the corresponding join-request, the router does not send a normal join-ack, but rather sends a JOIN-ACK with subcode PROXY-ACK. Proxy-acks, like normal join-acks, are unicast.

A router receiving a proxy-ack cancels any transient state it has created for the corresponding group. The sender of a proxy-ack becomes the group-specific DR (G-DR) for the group - a token (implicit) identity. In the normal case where there is no LAN extra hop, the receipt of a JOIN-ACK means that the D-DR becomes the G-DR for the specified group.

Control packets may continue to be incurred an extra-hop if they are generated by the D-DR, but data packets will not; since only the sender of the proxy-ack keeps a FIB entry for the group, it is the only router on the LAN that has an upstream forwarding entry.

Now let's see an illustration of this; a host joins a CBT group (the first to do so on the subnet), but more than one router is present on its subnet. B's subnet, S4, has 3 CBT routers attached. Assume also that R6 has been elected IGMP-querier and CBT D-DR.

The invoking of a multicast application on B causes an IGMP RP/Core-Report and an IGMP group membership report to be multicast to the corresponding group. The target core and ordered core list are contained within the RP/Core report. 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 would be forwarded to the origin, R6. R2 unicasts the join-ack on its final hop, but sets the ack subcode to PROXY-ACK. This results in the D-DR (R6) removing its pending join information for the specified group. Another consequence of receiving a proxy-ack is that the D-DR need not create a FIB entry for the specified group.

If an IGMP RP/Core-Report is received by a D-DR with a join for the same group already pending, it takes no action.

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 asymmetricity is detected by the CBT protocol.

2.7. 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.

Let's see, using the example topology of figure 1, 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 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 R2 has no CBT children, and no other directly attached subnets with group G presence, it immediately follows on by sending a QUIT_REQUEST to R3, its parent on the tree for group G. R3 responds 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 has been removed from the tree.

_3. _C_B_T _P_r_o_t_o_c_o_l _P_o_r_t_s

CBT routers implement user-level code for tree building, maintenance, and teardown. This results in a group-specific forwarding information base (FIB) being built in user-space. This FIB is downloaded into kernel-space for fast and efficient data packet forwarding. Any changes in FIB entries are communicated to the kernel as they occur, so that the kernel FIB always reflects the current state of any particular group's tree.

CBT primary and auxiliary control packets then travel inside UDP datagrams, as the following diagram illustrates:

```
++++++++++++++++++++++++++++++++++++++++++++
| IP header | UDP header | CBT control pkt |
++++++++++++++++++++++++++++++++++++++++++++
```

Figure 2. Encapsulation for CBT control messages

The following UDP port numbers are currently being used (their use at this stage is unofficial, and pending official approval):

o+   CBT Primary control messages - UDP port 7777

o+   CBT Auxiliary control messages - UDP port 7778

_4. _D_a_t_a _P_a_c_k_e_t _F_o_r_w_a_r_d_i_n_g (_n_a_t_i_v_e _m_o_d_e)

In CBT "native mode" only one forwarding method is used, namely 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". If this latter constraint cannot be satisfied it is necessary to encapsulate IP-over-IP before forwarding to a child or parent reachable via non-CBT-capable router(s).

The rules for native mode forwarding are altogether simpler than those for CBT-mode forwarding (see next section); data packets are sent over child/parent interfaces as specified in the corresponding FIB entry, as native IP multicasts. This applies to point-to-point links as well as broadcast-type subnetworks such as Ethernets.

_5. _D_a_t_a _P_a_c_k_e_t _F_o_r_w_a_r_d_i_n_g (_C_B_T _m_o_d_e)

"CBT mode" as opposed to "native mode" describes the forwarding of data packets over CBT tree interfaces containing a CBT header encapsulation. For efficiency, this encapsulation is as follows:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| encaps IP hdr | CBT hdr | original IP hdr | data ....|
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 3. Encapsulation for CBT mode

By using the encapsulations above there is no necessity to modify a packet's original IP header until it is forwarded over subnets with group member presence in native mode. When this happens, the TTL value of the original IP header is set to one before forwarding.

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 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 at this point 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 forwarding mode there are three forwarding methods used by CBT routers:

o+   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.

o+   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.

o+   CBT multicasting. This method sends 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 (with the exception of a proxy-ack, as explained in section 2.5). 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:

```
     32-bits          4            4            4         4     |    4
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   group-id   | parent addr | parent vif | No. of  |                    |
  |              |    index    |    index   |children |      children      |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                      |chld addr |chld vif |
                                                      | index    |  index  |
                                                      |+-+-+-+-+-+-+-+-+-+-+
                                                      |chld addr |chld vif |
                                                      | index    |  index  |
                                                      |+-+-+-+-+-+-+-+-+-+-+
                                                      |chld addr |chld vif |
                                                      | index    |  index  |
                                                      |+-+-+-+-+-+-+-+-+-+-+
                                                      |                    |
                                                      |         etc.       |
                                                      |+-+-+-+-+-+-+-+-+-+-+
```

Figure 4. 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:

o+ 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, the packet was sourced locally). If the packet is not of local origin, it is discarded.

o+ 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.

o+ the packet is encapsulated for CBT forwarding (see figure 3) 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.

5.1. Non-Member Sending (CBT mode)

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 IP-style packet and unicast it to a core for the group. This requires CBT routers to have access to a mapping mechanism between group addresses and core routers. This mechanism is currently beyond the scope of this document.

Alternatively, hosts could perform the CBT encapsulation themselves, but this would require hosts to run a core discovery protocol. Host modifications required for such a protocol, and the subsequent data packet encapsulation, are considered extremely undesirable, and are therefore not considered further.

_5._2. _E_l_i_m_i_n_a_t_i_n_g _t_h_e _T_o_p_o_l_o_g_y-_D_i_s_c_o_v_e_r_y _P_r_o_t_o_c_o_l _i_n _t_h_e _P_r_e_s_e_n_c_e _o_f _T_u_n_n_e_l_s

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.

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.

5.3. Non-Member Sending (native mode)

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) on the subnetwork must encapsulate (IP-over-IP) the IP-style packet and unicast it to a core for the group. This requires CBT routers to have access to a mapping mechanism between group addresses and core routers. This mechanism is currently beyond the scope of this document.

Again, host changes could obviate the need for a local router to perform a <core, group> mapping and an encapsulation, but this is not considered a desirable option.

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. Immediately subsequent to a parent/child relationship being established, a child unicasts a CBT-ECHO-REQUEST to its parent, which unicasts a CBT-ECHO-REPLY in response.

CBT echo requests and replies may be aggregated to conserve bandwidth on links over which tree branches overlap. However, this is only possible if group address assignment has been coordinated to facilitate aggregation. (see section 8.4).

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 a child 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. The router on the tree immediately subordinate to the failed router arbitrarily elects a core from its list of cores for this group. The rejoining router then 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 the specified number of retransmissions, an alternate core is arbitarily elected from the core list. The process is repeated until a JOIN-ACK is received for a maximum of RECONNECT-TIMEOUT seconds (90 secs is the recommended default).

- if a parent stops receiving CBT_ECHO_REQUESTs from a child. In this case the parent simply removes the child interface from its FIB entry for the particular group.

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 RP/Core 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 the core. This join is forwarded as normal until it reaches either the core, or a non-core router that is already part of the tree. If the join reaches the specified core, the join terminates there and is ACKd as normal. If however, the join is terminated by non-core router, the ACTIVE_REJOIN is converted to a NON_ACTIVE_REJOIN, keeping the origin as that specified in the ACTIVE_REJOIN, and forwarded upstream. A JOIN_ACK is also sent downstream to acknowledge the received join.

The NON_ACTIVE_REJOIN is a loop detection packet. All routers receiving this must forward it over their parent interface. This process continues until the NON_ACTIVE_REJOIN is received by the primary core for the group, or the NON_ACTIVE_REJOIN is received by the originator of the corresponding ACTIVE_REJOIN. A router will know this since the "origin" field remains unchanged when a join is converted from an ACTIVE_REJOIN to a NON_ACTIVE_REJOIN. In the former case, the primary core acknowledges the NON_ACTIVE_REJOIN with JOIN-ACK, sub-code NACTIVE_REJOIN. This message is unicast directly to the REJOIN_ACTIVE originator. In the latter case, the ACTIVE_REJOIN originator immediately sends a QUIT_REQUEST to its newly-established parent and the loop is broken.

o+ Using figure 5 (over) to demonstrate this, if R3 is attempting to re-join the tree (R1 is the core in figure 5) 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 changes the join subcode to NON_ACTIVE_REJOIN, and forwards this to its parent, R4. R4 forwards the NON_ACTIVE_REJOIN to R3, its parent. R3 originated the corresponding ACTIVE_REJOIN, and so 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, but there might be cases where, due to failure, the parent cannot respond. In this case the child nevertheless removes the parent information after some small number (typically 3) of re-tries.

```
                   ------
                   | R1 |
                   ------
                     |
           ---------------------------
                     |
                   ------
                   | R2 |
                   ------
                     |
           ---------------------------
                     |                             |
                   ------                          |
                   | R3 |--------------------------|
                   ------                          |
                     |                             |
           ---------------------------             |
                     |                             |       ------
                   ------                          |       |    |
                   | R4 |                          |-------| R6 |
                   ------                          |       |----|
                     |                             |
           ---------------------------             |
                     |                             |
                   ------                          |
                   | R5 |--------------------------|
                   ------                          |
                                                   |
```

Figure 5: Example Loop Topology

In the other scenario where no loop is actually formed, router 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 unicasts a JOIN-ACK to the originator of the REJOIN_ACTIVE, i.e. the join-ack remains invisible to R2.

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 data packets carry a CBT header when these packets are traversing CBT tree branches. The enscapsulation (for "CBT mode") is shown below:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| encaps IP hdr | CBT hdr | original IP hdr | data ....|
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 6. Encapsulation for CBT mode

CBT control packets carry a CBT control header. All CBT control messages are implemented over UDP. This makes sense for several reasons: firstly, all the information required to build a CBT delivery tree is kept in user space. Secondly, implementation is made considerably easier.

CBT control messages fall into two categories: primary maintenance messages, which are concerned with tree-building, re-configuration, and teardown, and auxiliary maintenance messsages, which are mainly concerned with general tree maintenance.

8.1. CBT Header Format

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  vers |unused |      type     |   hdr length  | on-tree|unused|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          checksum             |      IP TTL   |     unused    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         group identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           core address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           packet origin                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          flow identifier                      |
|                              (T.B.D)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          security fields                      |
|                              (T.B.D)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7. CBT Header

Each of the fields is described below:

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

- type: indicates whether the payload is data or control information.

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

- o+ 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.

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

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

- o+ group identifier: multicast group address.

- o+ core address: the unicast address of a core for the group. A core address is always inserted into the CBT header by an originating host, since at any instant, it does not know if the local DR for the group is on-tree. If it is not, the local DR must unicast the packet to the specified core.

- o+ packet origin: source address of the originating end-system.

- o+ flow-identifier: (T.B.D) value uniquely identifying a previously set up data stream.

- o+ security fields: these fields (T.B.D.) will ensure the authenticity and integrity of the received packet.

8.2. Control Packet Header Format

See over...

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

```
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  vers |unused |      type     |      code     |   # cores     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         hdr length            |           checksum            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        group identifier                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          packet origin                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       target core address                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Core #1                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Core #2                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Core #3                           |
   |                               ....                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Resource Reservation fields                  |
   |                             (T.B.D)                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         security fields                       |
   |                             (T.B.D)                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8. CBT Control Packet Header

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

o+ type: indicates control message type (see sections 1.3, 1.4).

o+ code: indicates subcode of control message type.

o+ # cores: number of core addresses carried by this control packet.

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

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

- o+ group identifier: multicast group address.

- o+ packet origin: source address of the originating end-system.

- o+ target core address: desired/actual core affiliation of control message.

- o+ Core #Z: IP address of core #Z.

- o+ Resource Reservation fields: these fields (T.B.D.) are used to reserve resources as part of the CBT tree set up procedure.

- o+ Security fields: these fields (T.B.D.) ensure the authenticity and integrity of the received packet.

8.3. Primary Maintenance Message Types

There are six types of CBT primary maintenance message. Primary message subcodes are described in the next section.

- o+ 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.

- o+ 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.

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

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

- o+ 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.

- o+ 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.

8.3.1. Primary Maintenance Message Subcodes

The JOIN-REQUEST has three valid subcodes:

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

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

- o+ REJOIN-NACTIVE (code 2) - converted from a REJOIN-ACTIVE by the first on-tree router receiving a REJOIN-ACTIVE. This message is forwarded over a router's parent interface until it either reaches the primary core, or is received by the originator of the corresponding REJOIN-ACTIVE.

A JOIN-ACK has three valid subcodes:

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

- o+ PROXY-ACK (code 1) - acknowledgement of a join-request by a router connected to the same subnet as the originator (subnet D-DR) of the corresponding join.

- o+ REJOIN-NACTIVE (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 converted the corresponding REJOIN-ACTIVE to REJOIN-NACTIVE. The CBT control packet "origin" field contains the IP address of the originator of the REJOIN-ACTIVE, so in order for the primary core to directly reach the source of the REJOIN-NACTIVE, the converting router inserts its IP address in the "core address" field of the control packet header. The primary core uses the address in this field to determine the target of the join-ack, subcode REJOIN-NACTIVE.

8.4. Auxilliary Maintenance Message Types

There are two CBT auxilliary maintenance message types. CBT auxiliary messages are encoded in a CBT control packet header, and the fields of the control packet are interpreted as illustrated below. The interpretation of certain fields further depends on whether aggregation and security are implemented.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  vers |unused |     type      |     code      |   aggregate   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         hdr length            |           checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           group identifier  (or low end of range)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      group id mask or NULL                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  NULL (if security implemented)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             security fields if implemented or NULL            |
|                             (T.B.D)                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9. CBT Echo Request/Reply

- o+ CBT-ECHO-REQUEST (type 7): once a tree branch is established, this messsage acts as a ``keepalive'', and is unicast from child to parent.

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

CBT Echo Requests/Replies can be sent as aggregates, or individually for each group if multicast address assignment is such that aggregation is not possible. If aggregation is implemented, the "aggregate" field (which replaces the "# cores" field of the standard control packet header. In this case, no cores are assumed present in the message) will contain the value 0xff, otherwise 0x00.

If aggregation is not implemented, the "group id mask" field is set to NULL, or is not present, depending on whether security is implemented or not. Masks are used according to their standard networking usage.

The "flow-id" field (to be done) of the standard control packet header is NULL if security is implemented, not present otherwise.

The security fields (to be done) are only present if security is implemented.

9. 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):

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

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

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

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

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

- o+ CHILD-ASSERT-INTERVAL 90 (check last time we rec'd an ECHO from each child)

- o+ CHILD-ASSERT-EXPIRE-TIME 180 (remove child information if no ECHO received)

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

10. 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 the ``CBT-other'' interface, and therefore we omit further discussion of this topic at the present time.

11. CBT Security Architecture

see current I-D: 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 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, Eric Crawley, Scott Reeve, and Nitin Jain. 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

IGMP version 3 has recently been proposed [6]. The authors have the following recommendations for amendments (all minor) to IGMPv3:

o+ The IGMPv3 draft [6] introduces a new IGMP message type, the PIM RP-REPORT message. Its message format is shown below:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Code      |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Group Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |   Reserved    |         # of RP's (N)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RP Address [1]                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RP Address [...]                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RP Address [N]                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10. PIM RP-REPORT.

The CBT authors propose the following minor amendments to the IGMP PIM RP-REPORT:

o+ the report to be re-named RP/CORE-REPORT

o+ RP fields re-named RP/Core fields

o+ the reserved field to be re-named the "target core" field, to contain the numeric value of the position of the target core in the RP/Core list

o+ The introduction of a new code value to distinguish PIM RP reports from CBT Core reports.

These minor amendments to IGMPv3 would satisfy CBT's operational requirements.

Author's Address:

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

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 (draft-ietf-idmr-mkd-01.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).

[9] D. Estrin et al. USC/ISI, Work in progress. (document not yet available).

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