A. J. Ballardie
University College London
S. Reeve
Bay Networks, Inc.
N. Jain
Bay Networks, Inc.

April, 1996

Core Based Trees (CBT) Multicast

-- Protocol Specification --

<draft-ietf-idmr-cbt-spec-05.txt>

Status of this Memo

Abstract

This document describes the Core Based Tree (CBT) network layer multicast protocol. 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 an optimization whereby unencapsulated (native) IP-style multicasts are forwarded by CBT, resulting in very good forwarding performance. This mode of operation is called CBT "native mode". Native mode can only be used in CBT-only domains or "clouds".

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-03.txt. Other related documents include [4, 5]. For all IDMR-related documents, see http://www.cs.ucl.ac.uk/ietf/idmr.

1. Changes since Previous Revision (04)

This note summarizes the changes to this document since the previous revision (revision 04).

+ inclusion of a "group mask" field for aggregated joins/join-acks (sections 10.2, 8.1, and Appendix A).

+ removal of the term "Group DR (G-DR)", which was only a "token" identity.

+ more complete explanation of the use of CBT's IP protocol and UDP port numbers (section 11).

+ more complete explanation of non-member sender case (section 6).

+ the term FIB (forwarding information base) has been replaced throughout with the term "forwarding database (db)".

+ editorial changes throughout for extra clarity.

Finally, in keeping with CBT's tradition of simplicity, this revision is 1 page less than the previous revision :-) .

2. Some Terminology

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

The "core tree" is the part of a tree linking all core routers of a particular group together.

3. Protocol Specification

3.1. Tree Joining Process -- Overview

A CBT router is notified of a local host's desire to join a group via IGMP [6]. We refer to a CBT router with directly attached hosts as a "leaf CBT router", or just "leaf" router.

The following CBT control messages come into play subequent to a subnet's CBT leaf router receiving an IGMP membership report (also termed "IGMP join"):

+ JOIN_REQUEST

+ JOIN_ACK

If the CBT leaf router is the subnet's default designated router (see next section), it generates a CBT join-request in response to receiving an IGMP group membership report from a directly connected host. The CBT join is sent to the next-hop on the unicast path to a target core, specified in the join packet; a router elects a "target core" based on a static configuration. If, on receipt of an IGMP-join, the locally-elected DR has already joined the corresponding tree, then it need do nothing more with respect to joining.

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.

3.2. DR Election

Multiple CBT routers may be connected to a multi-access subnetwork. In such cases it is necessary to elect a subnetwork designated router (D-DR) that is responsible for generating and sending CBT joins upstream, on behalf of the subnetwork.

CBT DR election happens "on the back" of IGMP [6]; on a subnet with multiple multicast routers, an IGMP "querier" is elected as part of IGMP; at start-up, a multicast router assumes no other multicast routers are present on its subnetwork, and so begins by believing it is the subnet's IGMP querier. It sends a small number IGMP-HOST-MEMBERSHIP-QUERYs in short succession in order to quickly learn about any group memberships on the subnet. If other multicast routers are present on the same subnet, they will receive these IGMP queries; a multicast router yields querier duty as soon as it hears an IGMP query from a lower-addressed router on the same subnetwork.

The CBT default DR (D-DR) is always (footnote 1) the subnet's IGMP-querier. As a result, there is no protocol overhead whatsoever associated with electing a CBT D-DR.

3.3. 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; 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 2), the join is not propogated further, but ACK'd downstream from that point.

JOIN-REQUESTs carry the identity of all the cores associated with 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

---

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.

2 "on-tree" refers to whether a router has a forwarding db entry for the corresponding group.

JOIN-ACK, then sends a JOIN-REQUEST, subcode REJOIN-ACTIVE, to the primary core router.

If the rejoin-active reaches the primary core, it responds by sending a JOIN-ACK, subcode PRIMARY-REJOIN-ACK, which traverses the reverse-path of the join. The primary-rejoin-ack serves to confirm no loop is present without requiring explicit loop detection.

If some other on-tree router is encountered before the rejoin-active reaches the primary, that router responds with a JOIN-ACK, subcode NORMAL. On receipt of the ack, subcode normal, the router sends a join, subcode REJOIN-NACTIVE, which acts as a loop detection packet (see section 8.3). Note that loop detection is not necessary subsequent to receiving a join-ack with subcode PRIMARY-REJOIN-ACK.

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.

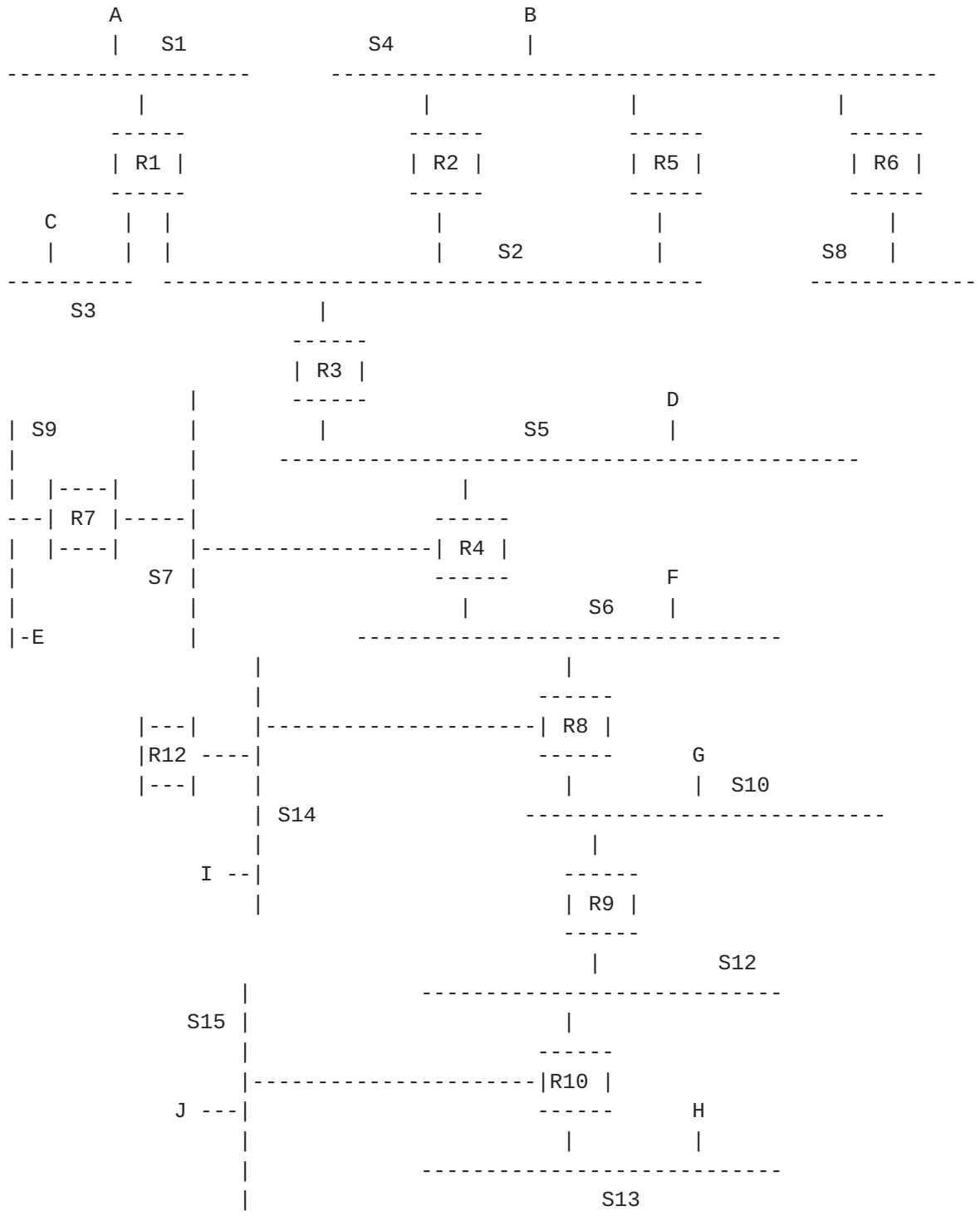


Figure 1. Example Network Topology

Taking the example topology in figure 1, host A is the group initiator, and has configured core routers R4 (primary core) and R9 (secondary core).

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 (subcode NORMAL) 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.

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 asymmetric routes in the underlying unicast routing, does not affect 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.

3.4. Forwarding Joins on Multi-Access Subnets

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 subnetwork, which actually forwards off-subnet.

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. R6's configured information suggests R4 as the target core for this group. R6 thus 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 the ack back to R2. R2 forwards it to R6, the origin of the join-request.

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.

3.5. On-Demand "Core Tree" Building

The "core tree", the part of a CBT tree linking all of its cores together, is built on-demand. That is, the core tree is only built subsequent to a non-primary (secondary) core receiving a join-request. This triggers the secondary core to join the primary core; the primary need never join anything.

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 secondary cores to know where to join when they themselves receive a join. Hence, the primary core must be uniquely identified as such across a whole group. A secondary joins the primary subsequent to sending an ack for the join just received.

3.6. 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 receipt of a quit-request triggers the receiving parent router to immediately query its forwarding database, and establish whether there remains any directly connected group membership, or any children, for the said group. If not, the router itself sends a quit-request upstream.

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.

4. Data Packet Forwarding Rules

4.1. Native Mode

In native mode, when a router receives a data packet, the packet's TTL is decremented, and, provided the packet's TTL remains greater than/equal to 1, forwards the data packet over all outgoing interfaces that are part of the corresponding CBT tree.

4.2. CBT Mode

In CBT mode, routers ignore all non-locally originated native mode multicast data packets. Locally-originated multicast data is only processed by a subnet's D-DR; in this case, the D-DR forwards the native multicast data packet, TTL 1, over any outgoing member subnets for which that router is D-DR. Additionally, the D-DR encapsulates the locally-originated multicast and forwards it, CBT mode, over all tree interfaces, as dictated by the CBT forwarding database.

When a router, operating in CBT mode, receives an encapsulated multicast data packet, it decapsulates one copy to send, native mode and TTL 1, over any directly attached member subnets for which it is D-DR. Additionally, an encapsulated copy is forwarded over all outgoing tree interfaces, as dictated by the CBT forwarding database.

Like the outer encapsulating IP header, the TTL value of the encapsulating CBT header is decremented each time it is processed by a CBT router.

An example of CBT mode forwarding is provided towards the end of the

next section.

5. CBT Mode -- Encapsulation Details

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:

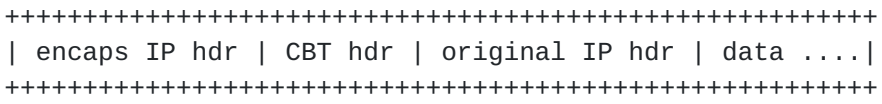


Figure 2. Encapsulation for CBT mode

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.

It is worth pointing out here the distinction between subnetworks and tree branches (especially apparent in CBT mode), 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 sends an unaltered (unencapsulated) data packet across a directly-connected subnetwork with group member presence. Any host originating multicast data, does so in this form.

+ 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 database (db) entries whenever they send or receive a JOIN_ACK. The forwarding database describes the parent-child relationships on a per-group basis. A forwarding database entry dictates over which tree interfaces, and how (unicast or multicast) a data packet is to be sent. A forwarding db entry is shown below:

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

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

Using our example topology in figure 1, let's assume the CBT routers are operating in CBT mode.

Member G originates an IP multicast (native mode) packet. R8 is the DR for subnet S10. R8 therefore sends a (native mode) copy over any member subnets for which it is DR - S14 and S10 (the copy over S10 is not sent, since the packet was originally received from S10). The multicast packet is CBT mode encapsulated by R8, and unicast to each of its children, R9 and R12; these children are not reachable over the same interface, otherwise R8 could have sent a CBT mode multicast. R9, the DR for S12, need not IP multicast (native mode) onto

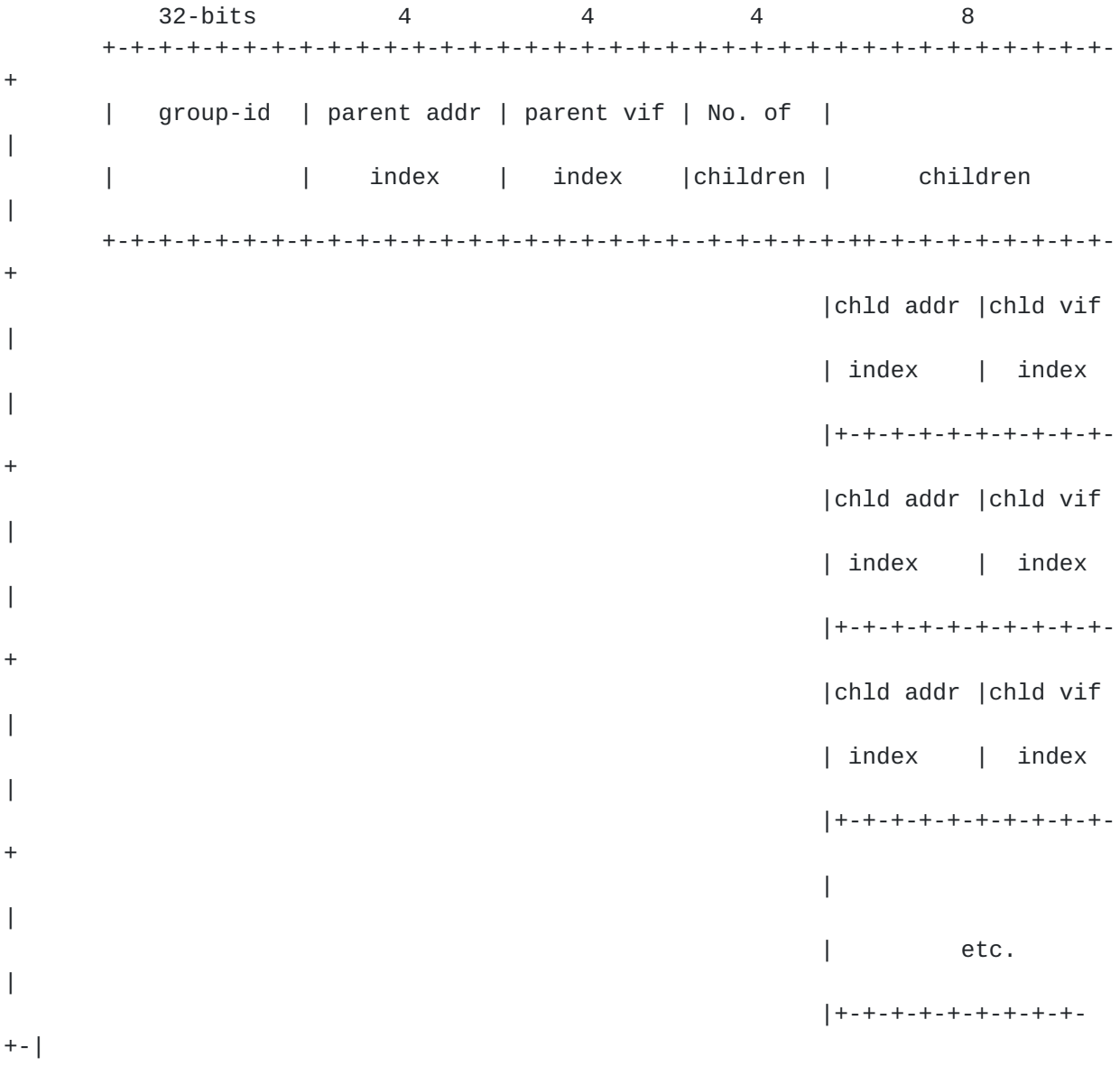


Figure 3. CBT forwarding database entry

S12 since there are no members present there. R9, in CBT mode, unicasts the packet to R10, which is the DR for S13 and S15. R10 decapsulates the CBT mode packet and IP multicasts (native mode) to each of S13 and S15.

Going upstream from R8, R8 CBT mode unicasts to R4. It is DR for all directly connected subnets and therefore IP multicasts (native mode) the data packet onto S5, S6 and S7, all of which have member presence. R4 unicasts, CBT mode, 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 (native mode) onto S9. R3 CBT mode unicasts to R1 and R2, its children. Finally, R1 IP multicasts (native mode) onto S1 and S3, and R2 IP multicasts (native mode) onto S4.

6. 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. The encapsulation required is shown in figure 2; CBT mode encapsulation is necessary so the receiving CBT router can demultiplex the packet accordingly.

If the encapsulated packet hits the tree at a non-core router, the packet is forwarded according to the forwarding rules of section 4.2.

If the first on-tree router encountered is the target core, various scenarios define what happens next:

+ if the target core is not the primary, and the target core has not yet joined the tree (because it has not yet itself received any join-requests), the target core simply forwards the encapsulated packet to the primary core.

  if the target core is not the primary, but has children, the target core forwards the data according to the rules of section 4.2.

+ if the target core is the primary, the primary forwards the data according to the rules of section 4.2.

7. 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 [1]. 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 forwarding database 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 forwarding db 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.

8. 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 (or 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 transit network, or backbone network, links.

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.

CBT echo requests and replies can be aggregated and sent on a per link basis, rather than individually for each group; the CBT control packet header (section 10.2) accommodates such aggregation.

8.1. 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. For all groups sharing a common core set (corelist), provided those groups can be specified as a CIDR-like aggregate, an aggregated join can be sent representing a range of groups. Aggregated joins are made possible by the presence of a "group mask" field in the CBT control packet header. Aggregated joins are also discussed in Appendix A.

  If a range of groups cannot be represented by a mask, then each group must be re-joined individually.

  CBT's re-join strategy is as follows: the rejoining router 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), the next-highest priority 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, if the parent has not received an expected keepalive after CHILD_ASSERT_EXPIRE_TIME, all children reachable across that link are removed from the parent's forwarding database.

8.2. Router Re-Starts

There are two cases to consider here:

+ Core re-start. All JOIN-REQUESTs (all types) carry the identities (i.e. IP 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.

8.3. 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 because the

primary never has a parent. 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; this ack creates tree state, just like a normal join-ack.

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.

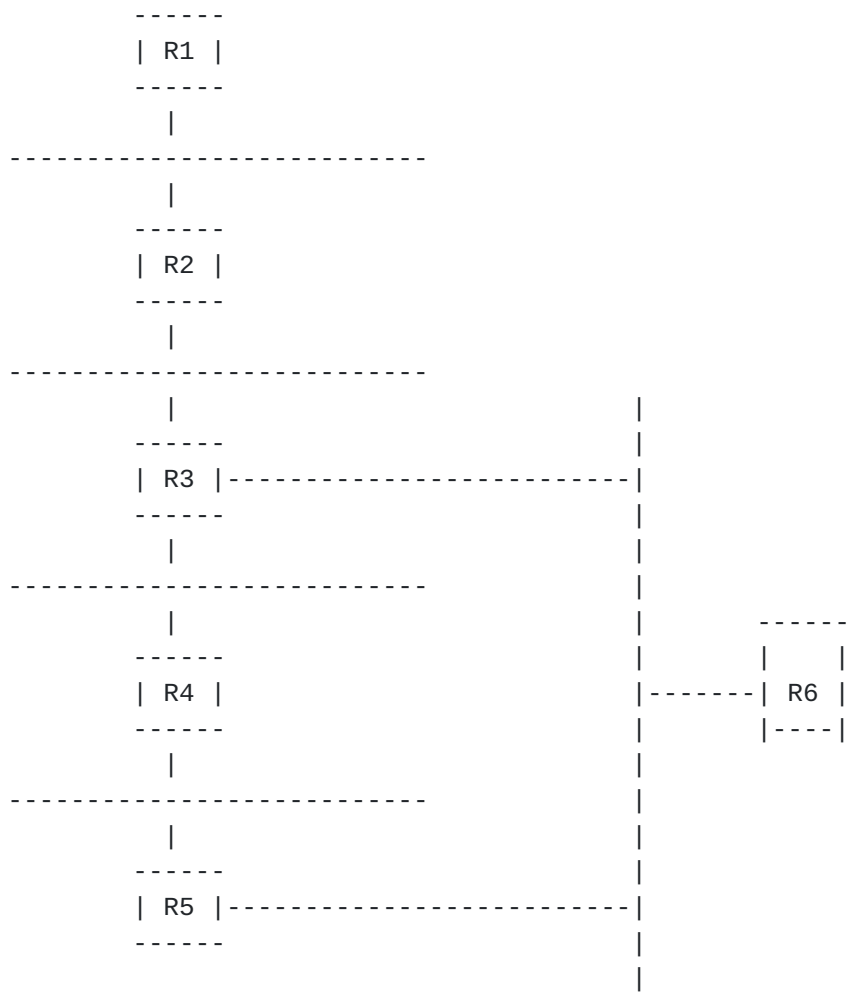


Figure 4: Example Loop Topology

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.

9. Data Packet Loops

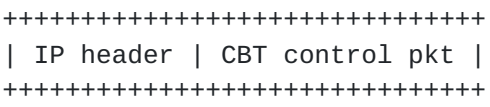
The CBT protocol builds a loop-free distribution tree. If all routers that comprise a particular tree function correctly, data packets should never traverse a tree branch more than once.

CBT mode data packets from a non-member sender must arrive on a tree via an "off-tree" interface. The CBT mode data packet's header includes an "on-tree" field, which contains the value 0x00 until the data packet reaches an on-tree router. The first on-tree router must convert this value to 0xff. 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, an 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.

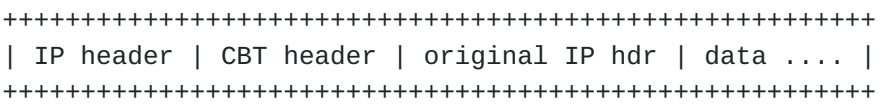
10. CBT Packet Formats and Message Types

We distinguish between two types of CBT packet: CBT mode data packets, and CBT control packets. CBT control packets carry a CBT control packet header.

For "conventional router" implementations, it is recommended CBT control packets be encapsulated in IP, as illustrated below:

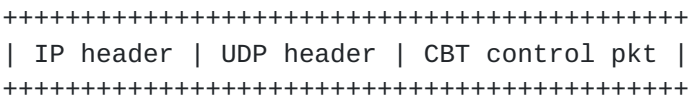


In CBT mode, the original data packet is encapsulated in a CBT header and an IP header, as illustrated below:



The IP protocol field of the IP header is used to demultiplex a packet correctly; CBT has been assigned IP protocol number 7. The CBT module then demultiplexes based on the encapsulating CBT header's "type" field, thereby distinguishing between CBT control packets and CBT mode data packets (the first 16 bits of both the CBT control and CBT data packet headers are identical).

Some implementations of CBT encapsulate CBT control packets in UDP (like the workstation router version). In these implementations, the encapsulation of CBT contol packets is as follows:



CBT has been assigned UDP port number 7777 for this purpose.

It is recommended for performance reasons that conventional router implementations implement the IP encapsulation for control packets, not the UDP encapsulation.

The CBT data packet header is illustrated below:

10.1. CBT Header Format (for CBT Mode data)

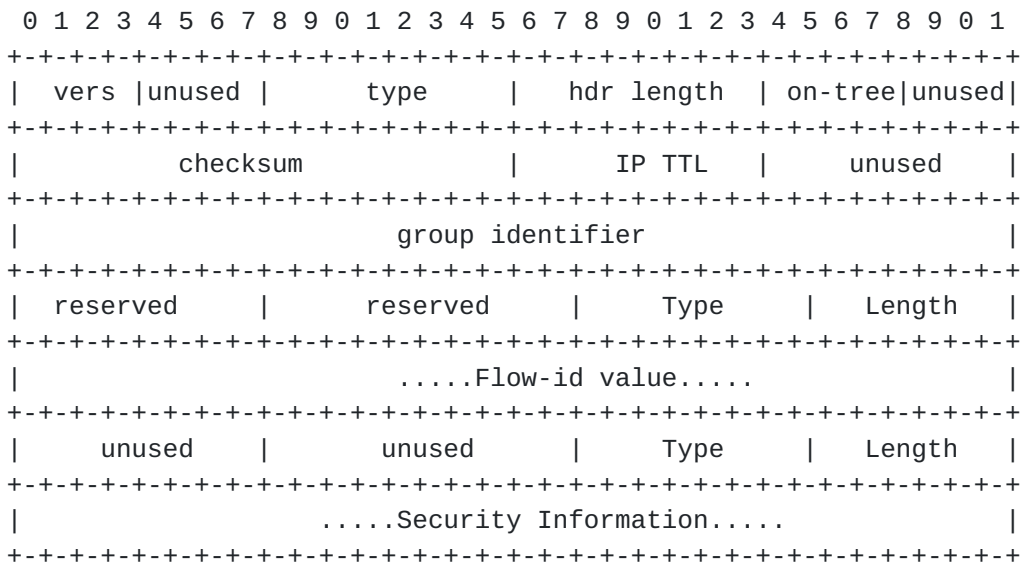


Figure 5. CBT Header

Each of the fields is described below:

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

+ type: indicates CBT payload; values are defined for control (0x00), and data (0xff). For the value 0x00 (control), a CBT control header is assumed present rather than a CBT header.

+ 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).

+ 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.

+ 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.

10.2. Control Packet Header Format

The individual fields are described below.

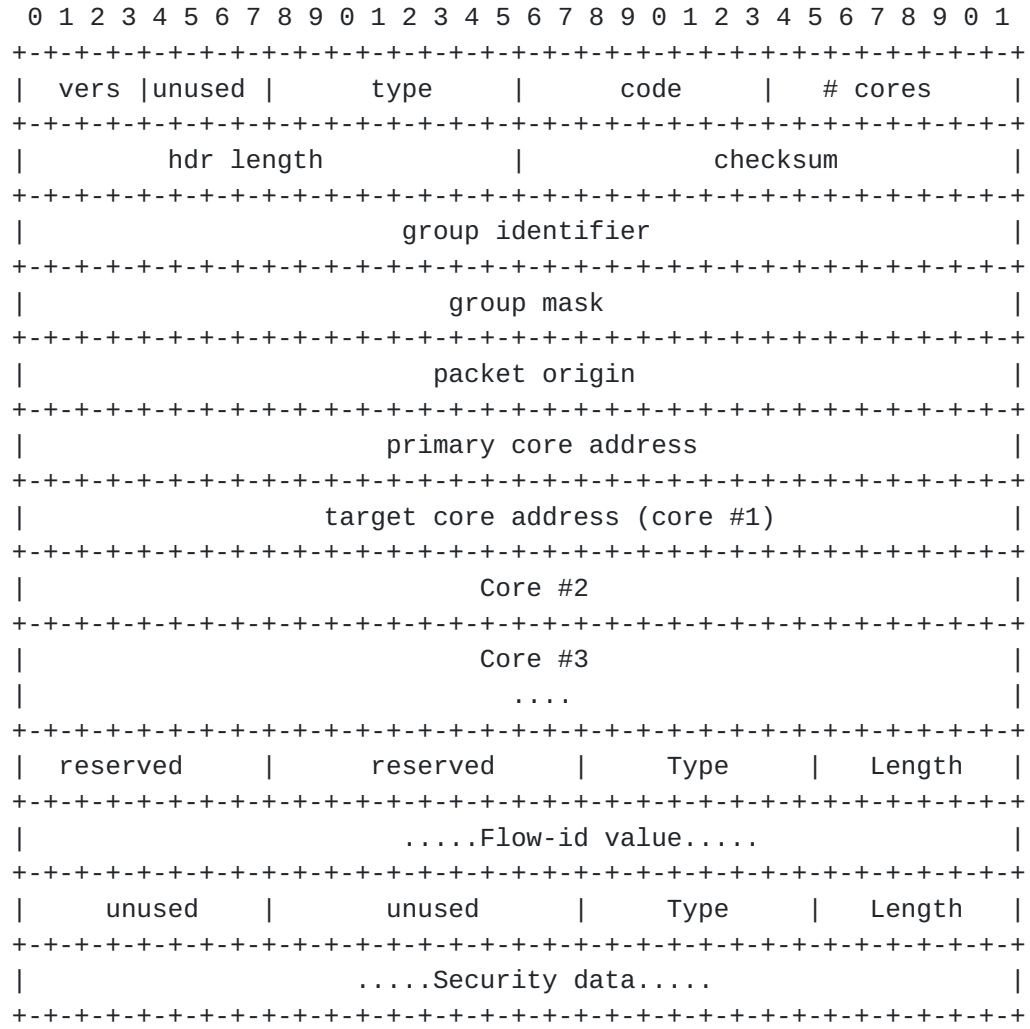


Figure 6. CBT Control Packet Header

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

+ type: indicates control message type (see sections 10.3).

+ code: indicates subcode of control message type.

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

+ 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.

+ group mask: mask value for aggregated CBT joins/join-acks. Zero for non-aggregated joins/join-acks.

+ 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 #1, #2, #3 etc.: IP address for each of a group's cores.

+ 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.

10.3. CBT Control Message Types

There are ten 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 originating 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 reverse 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 "fixes" tree state.

+ 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 (can be aggregated from one per group to one

per link).

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

+ CBT-BR-KEEPALIVE (type 9): applicable to border routers only, when attaching a CBT domain to some other domain. See [11] for more information.

+ CBT-BR-KEEPALIVE-ACK (type 10): acknowledgement to the above.

10.3.1. 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.

11. CBT Protocol and Port Numbers

CBT has been assigned IP protocol number 7, and UDP port number 7777. The UDP port number is only required for certain CBT implementations, as described at the beginning of section 10.

12. 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)

+ BR-KEEPALIVE-INTERVAL 200 (backup designated BR to designated BR keepalive interval)

+ BR-KEEPALIVE-RETRY-INTERVAL 30 (keepalive interval if BR fails to respond)

13. Interoperability Issues

Interoperability between CBT and DVMRP has recently been defined in ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-cbt-dvmrp-00.txt.

Interoperability with other multicast protocols will be fully specified shortly.

14. CBT Security Architecture

see [4].

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

There are situations where it is advantageous to send a single join-request that represents potentially many groups. One such example is provided in [11], whereby a designated border router is required to join all groups inside a CBT domain.

Such aggregated joining is only possible if each of the groups the join represents shares a common corelist. Furthermore, aggregation is only efficient over contiguous ranges of group addresses; the "group mask" field in the CBT control packet header is used to specify a CIDR-like group address mask.

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. NOTE: DVMRP version 3 is specified as a working draft.

[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; RFC XXXX, SRI Network Information Center, 1996.

[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).

[11] A. Ballardie. CBT Multicast Interoperability - Stage 1; Working draft, April 1996. Also available from: ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-cbt-dvmrp-00.txt