<draft-ietf-idmr-cbt-spec-02.txt>

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

June 20th, 1995

Core Based Trees (CBT) Multicast

-- Protocol Specification --

Status of this Memo

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].

The 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 insert a CBT header into data packets before forwarding over CBT interfaces. Native mode is only relevant to CBT-only domains or ``clouds''.

The CBT architecture is described in an accompanying document: draft-ietf-idmr-arch-00.txt. Other related documents include [4, 5].

_1. _D_o_c_u_m_e_n_t _L_a_y_o_u_t

We describe the protocol details by means of example using the topology shown in figure 1. Examples show how a host joins a group and leaves a group, and we also show various tree maintenance scenarios.

In this figure member hosts are shown as capital letters, routers are prefixed with R, and subnets are prefixed with S.

Figure 1 is shown over...

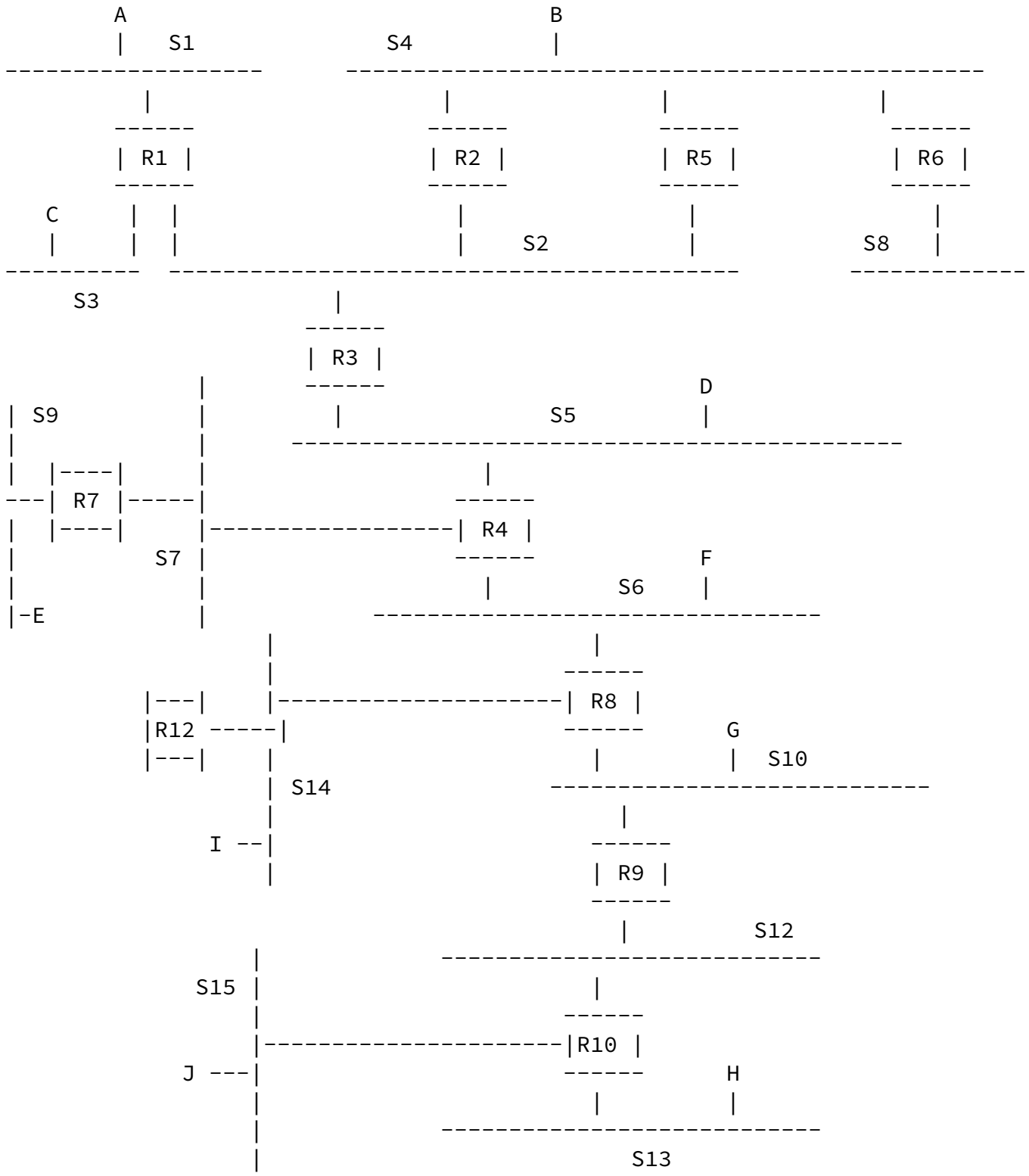
A
B
S1
S4
R1
R2
R5
R6
C
S2
S8
S3
R3
D
S9
S5
R7
R4
S7
F
S6
E
R8
R12
G
S10
S14
I
R9
S12
S15
R10
J
H
S13

Figure 1. Example Network Topology

_2. _P_r_o_t_o_c_o_l _S_p_e_c_i_f_i_c_a_t_i_o_n

_2._1. _C_B_T _G_r_o_u_p _I_n_i_t_i_a_t_i_o_n

Like any of the other multicast schemes, one user, the group initiator, initiates a CBT multicast group. Group initiation could be carried out by a network management centre, or by some other external means, rather than have a user act as group initiator. However, in the author's implementation, this flexibility has been afforded the user, and a CBT group is invoked by means of a graphical user interface (GUI), known as the CBT User Group Management Interface.

NOTE: Work is currently in progress to address the issue of core placement.

_2._2. _T_r_e_e _J_o_i_n_i_n_g _P_r_o_c_e_s_s

The following steps are involved in a host establishing itself as part of a CBT multicast tree:

o+ the joining host must inform all routers on its subnet that it requires a Designated Router (DR) for the group it wishes to join (it is a requirement that only one router, the DR, forward to and from upstream to avoid loops).

o+ the establishment of a DR for the group.

o+ once established, the DR must proceed to join the distribution tree.

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

NOTE: all CBT message types are described in section 8 irrespective

of some of the comments included with certain message types below.

- CORE_NOTIFICATION (sent only by a group initiating host to inform each core for the group that it has been elected as a core for the group).

- CORE_NOTIFICATION_ACK

- DR_SOLICITATION

- DR_ADVERTISEMENT_NOTIFICATION (sent only by a local CBT-capable router when that router is unaware of a DR for the group on the same subnet, and believes it is candidate for the best next-hop router off the LAN to the core address as specified in the DR_SOLICITATION. This message acts as a tie-breaker in the case where there are two or more such routers on a subnet).

- DR_ADVERTISEMENT

- TAG_REPORT (sent by a joining host to the DR subsequent to receiving a DR_ADVERTISEMENT. This message serves to invoke the DR to become part of the distribution tree, if not already, by sending a JOIN_REQUEST).

- JOIN_REQUEST (sent only by the group's DR iff it is not yet part of, or in the process of, joining the corresponding CBT tree).

- JOIN_ACK

- HOST_JOIN_ACK (multicast across the subnet by the local DR as an indication that the DR is part of the distribution tree. This message may be sent in immediate response to receiving a TAG_REPORT, depending on whether the DR is already part of the CBT tree or not. If not it is sent subsequent to the DR receiving a JOIN_ACK).

A group-initiating host sends a CORE-NOTIFICATION message to each of the elected cores for the group. This message is acknowledged (CORE_NOTIFICATION_ACK) by each core individually. Provided at least one ACK is received a host will not be prevented from joining the tree.

The purpose of the CORE_NOTIFICATION is twofold: firstly, to communicate the identities of all of the cores, together with their rankings, to each of them individually; secondly, to invoke the building of the core backbone or core tree. These two procedures follow on one to the other in the order just described. New receivers attempting to join whilst the building of the core backbone is still in progress have their explicit JOIN-REQUEST messages stored by whichever CBT-capable router involved in the core joining process is encountered first.

Taking our example topology in figure 1, host A is the group initiator. The elected cores are router R4 (primary core) and R9 (secondary core). Host A first sends a CORE_NOTIFICATION to each of R4 and R9, and each responds positively with a CORE_NOTIFICATION_ACK. CORE_NOTIFICATION messages are always unicast.

Subsequent to sending a CORE_NOTIFICATION_ACK, each secondary core router (in this case there is only one secondary, R9) proceeds to join the primary core, and thus forms the core tree, or backbone; R9 unicasts a JOIN_REQUEST (subcode CORE_JOIN) to R8, its best next-hop to the primary core, R4. JOIN_REQUESTs (and corresponding ACKs) are processed by all intervening CBT-capable routers, and forwarded if necessary. R8 forwards the JOIN_REQUEST to R4, remembering the incoming and outgoing interfaces of the JOIN_REQUEST.

R4 receives the JOIN_REQUEST (subcode CORE_JOIN), realises it is the target of the join, and therefore sends a JOIN_ACK back out of the receiving interface to the previous-hop sender of the join. R8 receives the JOIN_ACK and forwards it to R9 over the interface the join was received from R9. On receipt of the JOIN_ACK, R9 need take no further action. Core tree set up is complete.

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, at which time it can acknowledge any cached joins. A router is said to be in a pending-join state if it is awaiting a JOIN_ACK itself.

Returning to host A which has just received both

CORE_NOTIFICATION_ACKs, it must now establish which local CBT router is DR for the group. Since A is the group initiator it is highly unlikely that a DR for the group will already exist. If A was joining an existing group a DR may already be present.

Host A sends a DR_SOLICITATION (IP TTL 1) to the "all-CBT-routers" address (224.0.0.7). The solicitation contains one of core addresses as elected by the host, to which it wishes a join to be sent. Any routers on the same subnet receiving the solicitation establish whether they are the best next-hop to the specified core or not. If a router does consider itself a candidate and has no record for a DR for the group, it multicasts a DR_ADV_NOTIFICATION to the "all-CBT-routers" group (224.0.0.7). This message acts as a tie-breaker in the case where there is more than one CBT router on the subnet which

thinks it is the best next-hop to the core. The lowest-addressed source of a DR_ADV_NOTIFICATION wins the election and subsequently advertises itself as DR by means of a DR_ADVERTISEMENT, multicast to the "all-systems group (224.0.0.1). As R1 is the only router on A's subnet, it responds with a DR_ADV_NOTIFICATION followed by a DR_ADVERTISEMENT.

The time between sending a DR_ADV_NOTIFICATION and a DR_ADVERTISEMENT should be configurable and ideally less than one second so as to keep join latency to a minimum.

The DR election for subnet S4 is more complex. When host B sends a DR_SOLICITATION routers R2, R5 and R6 receive it. Assuming R2 and R5 both believe they are the best next-hop to R4 (the specified core) both send a DR_ADV_NOTIFICATION. R2 (the lower addressed) wins the tie-breaker and subsequently multicasts a DR_ADVERTISEMENT to S4. All subnets with joining hosts proceed similarly.

A DR candidate is a router whose outgoing interface, as specified in its routing table entry for the destination, is different than the interface over which the DR_SOLICITATION arrived.

On receiving a DR_ADVERTISEMENT host A sends a TAG_REPORT to the DR, R1. R1 responds by unicasting a JOIN_REQUEST (subcode ACTIVE_JOIN) to R3 -- the best next-hop to R4, the desired target of the join. R3 forwards (unicast) the received join to R4, remembering incoming and outgoing interfaces. R4, now already established on tree for the

group responds to the JOIN_REQUEST with a JOIN_ACK, and sends it to R3, which in turn sends it to R1. The branch R1-R3-R4 is now complete and part of the distribution tree.

On receipt of the JOIN_ACK, R1 multicasts to the "all-systems" address (224.0.0.1) a HOST_JOIN_ACK which is a notification to the joining end-system that the DR has been successful in joining the tree. The multicast application running on host A can now send data.

Host B proceeds to join the group in a similar fashion, but there are some subtle differences. Host B is not the group initiator and it need not send CORE_NOTIFICATIONs. Host B's first step is to elect a DR, as described above. On receipt of a DR_ADVERTISEMENT from router R2 in this case, B unicasts a TAG_REPORT to R2. The core specified in the TAG_REPORT is R4. In response the the TAG_REPORT, R2 unicasts a JOIN_REQUEST (subcode ACTIVE_JOIN) to R3, the best next-hop to R4. R3 however, has just joined the tree and so can acknowledge the received join, i.e. it need not travel all the way to R4. R3 unicasts a

JOIN_ACK to R2, which results in R2 multicasting a HOST_JOIN_ACK across subnet S4.

_3. _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/sending of data packets over CBT tree interfaces containing a CBT header encapsulation. For efficiency, this encapsulation is as follows:

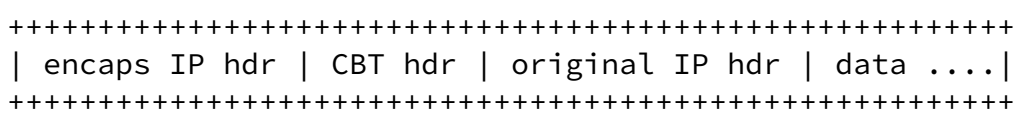

Figure 2. Encapsulation for CBT mode

By using the encapsulations above there is virtually no necessity to modify a packet's original IP header, and decapsulation is relatively

efficient.

It is worth pointing out at this point the distinction between sub-networks 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 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. Thus, 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.

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 will not be recognized.

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

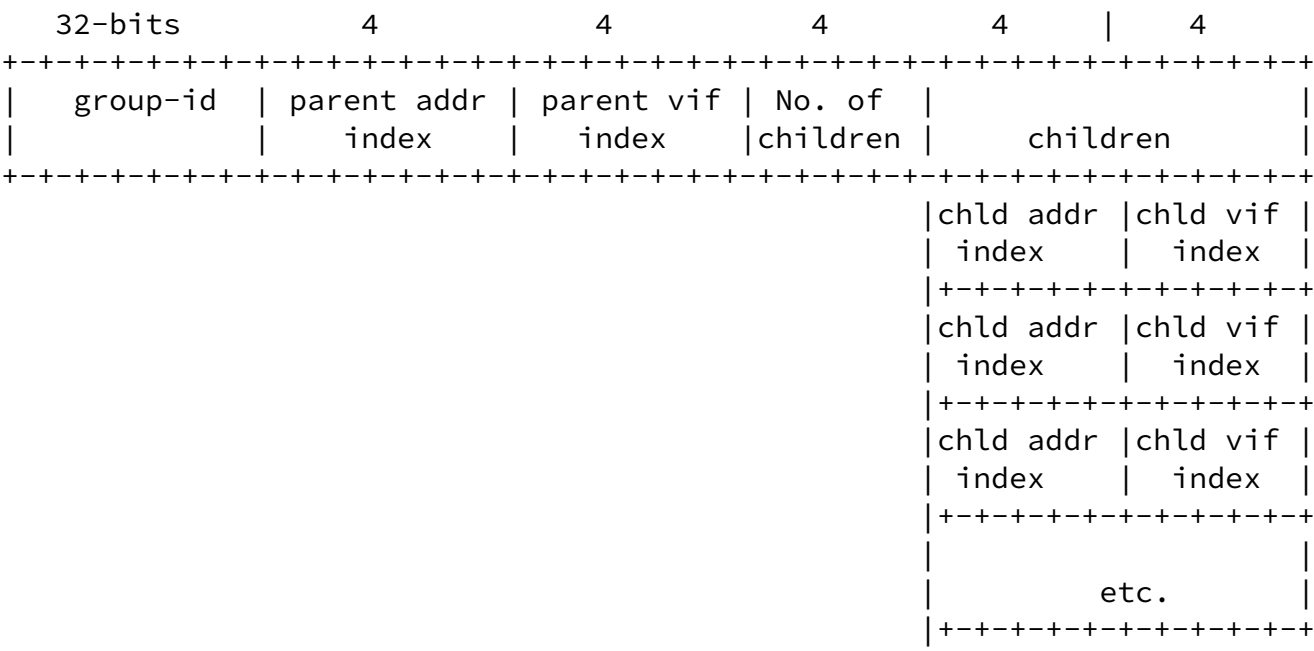

Figure 3. CBT FIB entry

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 ANDed with the packet's source IP address equals the arrival interface's subnet number, the packet was sourced locally). If it does not the packet 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 2) and unicast to parent and children. However, if more than one child is reachable over the same interface the packet will be CBT multicast. Therefore, it is possible that an IP-style multicast and a CBT multicast will be forwarded over a particular subnetwork.

Using our example topology in figure 1, let's assume member G originates an IP multicast packet. R8 is the DR for subnet S10 (R4 is DR for all its attached subnets). 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.

_3._1. _N_o_n-_M_e_m_b_e_r _S_e_n_d_i_n_g

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 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.

_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. It also assumes 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).

Besides the structural characteristics of "native mode" data packets, described above, the data packet forwarding rules are identical to those described in section 3.

_4._1. _N_o_n-_M_e_m_b_e_r _S_e_n_d_i_n_g (_n_a_t_i_v_e _m_o_d_e)

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 DR for the group 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.

_5. _T_r_e_e _M_a_i_n_t_e_n_a_n_c_e

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.

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

_5._1. _R_o_u_t_e_r _F_a_i_l_u_r_e

A non-core router can detect a failure from the following two cases:

- o+ 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. It does so by arbitrarily choosing an alternate core from its list of cores for this group. It establishes a chosen core's reachability by unicasting a CBT_CORE_PING message to it, to which the core responds with a CBT_PING_REPLY. On receipt of the latter, the re-joining router sends a JOIN_REQUEST (subcode ACTIVE_REJOIN) to the best next-hop router on the path to the core. A router will continue arbitrarily choosing an alternate core until a CBT_PING_REPLY is received.

- o+ 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.

_5._2. _R_o_u_t_e_r _R_e-_S_t_a_r_t_s

There are two cases to consider here:

- o+ Core re-start. In this case, the core router relies on receiving a CBT_CORE_PING message, which contains the list of cores for the specified group. Obviously, one of the core addresses will

be its own. If a core realises its core status for a group in this way, if it is not the primary it sends a JOIN_REQUEST (sub-code ACTIVE_JOIN) to the primary core. If the router in ques-

tion is the primary it need not send a join, but rather awaits joins and considers itself part of the tree again.

- 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.

5.3. 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 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. If the originator of the corresponding ACTIVE_REJOIN should receive the NON_ACTIVE_REJOIN it immediately sends a QUIT_REQUEST to its recently 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 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 need 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 of re-tries.

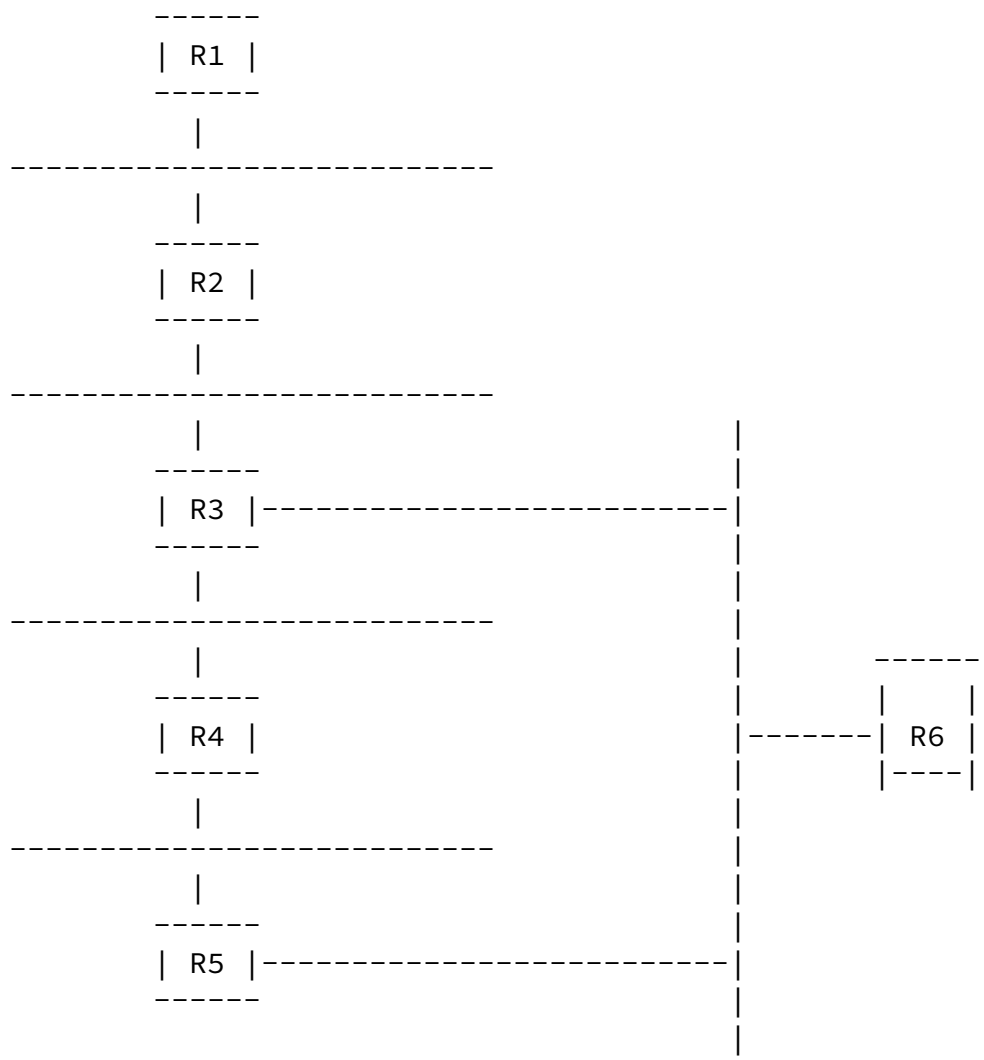

Figure 4: Example Loop Topology

_6. _D_a_t_a _P_a_c_k_e_t _L_o_o_p_s

NOTE: this is only applicable when CBT header encapsulation is in use.

When a data packet hits its first on-tree router, that router is responsible for setting the on-tree bits in the CBT header. This indicates to all subsequent routers on the tree that the packet is in the process of spanning the tree for the group. However, it might be that a misbehaving router forwards an on-tree packet over a non-tree interface, and such a packet might work its way back onto the tree, potentially forming a data packet loop. Therefore, the on-tree bits in the CBT header serve to identify such packets -- should a router receive a data packet with its on-tree bits set over a non-tree interface the packet is immediately discarded.

_7. _T_r_e_e _T_e_a_r_d_o_w_n

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

o+ 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 re-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. Subsequent to sending a FLUSH_TREE, the router can send the re-join to its child.

o+ 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.

With regards to the latter scenario, lets see using the example topology of figure 1 how a tree branch is torn down.

Assume member E leaves the group (if IGMPv2 is in use an explicit IGMP_LEAVE message will be sent by E). If R7 registers no further group presence (by means of IGMP) then R7 sends a QUIT_REQUEST to R4. R4 responds with a QUIT_ACK to R7. R4 has children AND subnets with group presence, and so does not itself attempt to quit the tree. The branch R4-R7 has been torn down.

_8. _C_B_T _P_a_c_k_e_t _F_o_r_m_a_t_s _a_n_d _M_e_s_s_a_g_e _T_y_p_e_s

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:

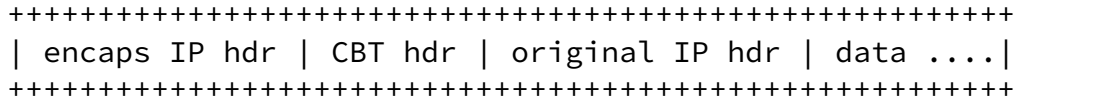

Figure 5. Encapsulation for CBT mode

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. _C_B_T _H_e_a_d_e_r _F_o_r_m_a_t

See over....

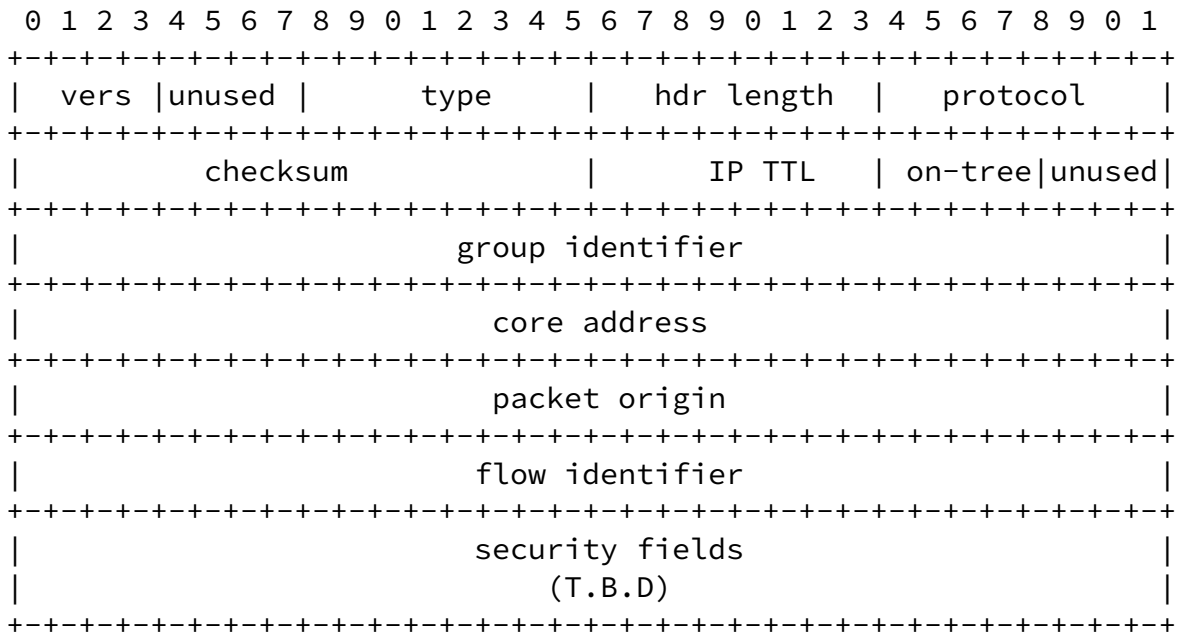

Figure 6. CBT Header

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.

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

- protocol: upper-layer protocol number.

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

- IP TTL: TTL value gleaned from the IP header where the packet

originated. It is decremented each time it traverses a CBT router.

- on-tree: indicates whether the packet is on- or off-tree. Once this field is set (i.e. on-tree), it is non-changing.

- group identifier: multicast group address.

- 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.

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

- flow-identifier: value uniquely identifying a previously set up data stream.

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

_8._2. _C_o_n_t_r_o_l _P_a_c_k_e_t _H_e_a_d_e_r _F_o_r_m_a_t

The individual fields are described below. It should be noted that the contents of the fields beyond ``group identifier'' are empty in some control messages:

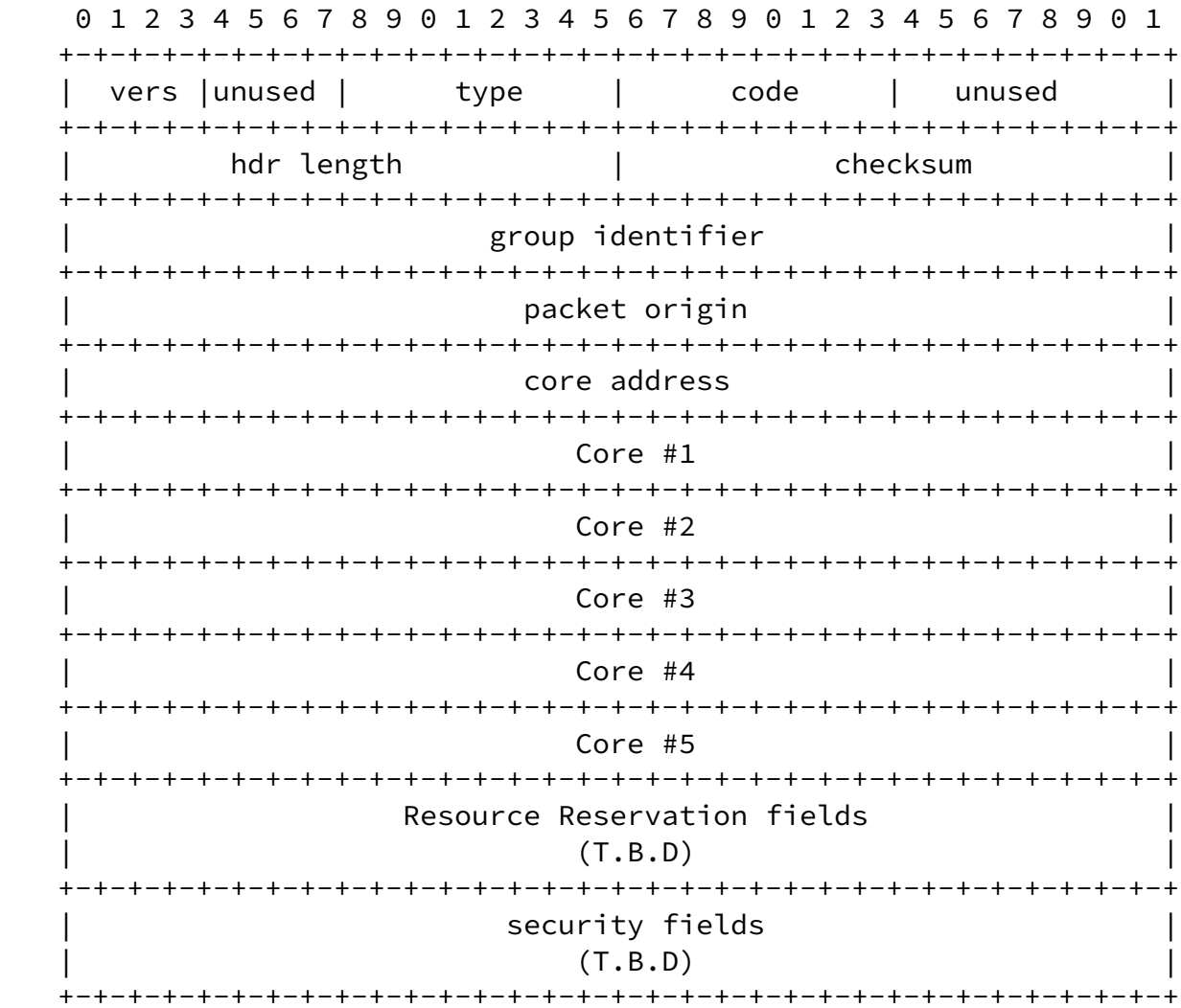
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
vers
unused
type
code
unused
hdr length
checksum
group identifier
packet origin
core address
Core #1
Core #2
Core #3
Core #4
Core #5
Resource Reservation fields
(T.B.D)
security fields
(T.B.D)

Figure 7. CBT Control Packet Header

- Vers: Version number -- this release specifies version 1.
- type: indicates control message type (see sections 1.3, 1.4).
- code: indicates sub-code of control message type.
- header length: length of the header, for purpose of checksum calculation.
- checksum: the 16-bit one's complement of the one's complement of the CBT control header, calculated across all fields.

- group identifier: multicast group address.
- packet origin: source address of the originating end-system.
- core address: desired/actual core affiliation of control message.
- Core #Z: Maximum of 5 core addresses may be specified for any one group. An implementation is not expected to utilize more than, say, 3.

  NOTE: It was an engineering design decision to have a fixed maximum number of core addresses, to avoid a variable-sized packet.

- Resource Reservation fields: these fields (T.B.D.) are used to reserve resources as part of the CBT tree set up procedure.
- Security fields: these fields (T.B.D.) ensure the authenticity and integrity of the received packet.

_8._3. _P_r_i_m_a_r_y _M_a_i_n_t_e_n_a_n_c_e _M_e_s_s_a_g_e _T_y_p_e_s

There are six types of CBT primary maintenance message, namely:

- o+ JOIN-REQUEST: invoked by an end-system, generated and sent (unicast) by a CBT router 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: 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, and it is

the receipt of a JOIN-ACK that actually creates a tree branch.

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

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

- o+ QUIT-ACK: 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: 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.

The JOIN-REQUEST has three valid sub-codes, namely JOIN-ACTIVE, RE-JOIN-ACTIVE, and RE-JOIN-NACTIVE.

A JOIN-ACTIVE is sent from a CBT router that has no children for the

specified group.

A RE-JOIN-ACTIVE is sent from a CBT router that has at least one child for the specified group.

A RE-JOIN-NACTIVE originally started out as an active re-join, but has reached an on-tree router for the corresponding group. At this point, the router changes the join status to non-active re-join and forwards it on its parent branch, as does each CBT router that receives it. Should the router that originated the active re-join subsequently receive the non-active re-join, it must immediately send a QUIT-REQUEST to its parent router. It then attempts to re-join again. In this way the re-join acts as a loop-detection packet.

_8._4. _A_u_x_i_l_l_i_a_r_y _M_a_i_n_t_e_n_a_n_c_e _M_e_s_s_a_g_e _T_y_p_e_s

There are eleven CBT auxilliary maintenance message types:

- CBT-DR-SOLICITATION: a request sent from a host to the CBT ``all-routers'' multicast address, for the address of the best next-hop CBT router on the LAN to the core as specified in the solicitation.

- CBT-DR-ADVERTISEMENT: a reply to the above. Advertisements are addressed to the ``all-systems'' multicast group.

- CBT-CORE-NOTIFICATION: unicast from a group initiating host to each core selected for the group, this message notifies each core of the identities of each of the other core(s) for the group, together with their core ranking. The receipt of this message invokes the building of the core tree by all cores other than the highest-ranked (primary core).

- CBT-CORE-NOTIFICATION-ACK: a notification of acceptance to becoming a core for a group, to the corresponding end-system.

- CBT-ECHO-REQUEST: once a tree branch is established, this

messsage acts as a ``keepalive'', and is unicast from child to parent.

- o+ CBT-ECHO-REPLY: positive reply to the above.

- o+ CBT-CORE-PING: unicast from a CBT router to a core when a tree router's parent has failed. The purpose of this message is to establish core reachability before sending a JOIN-REQUEST to it.

- o+ CBT-PING-REPLY: positive reply to the above.

- o+ CBT-TAG-REPORT: unicast from an end-system to the designated router for the corresponding group, subsequent to the end-system receiving a designated router advertisement (as well as a core notification reply if group-initiating host). This message invokes the sending of a JOIN-REQUEST if the receiving router is not already part of the corresponding tree.

- o+ CBT-HOST_JOIN_ACK: group-specific multicast by a CBT router that originated a JOIN-REQUEST on behalf of some end-system on the same LAN (subnet). The purpose of this message is to notify end-systems on the LAN belonging to the specified group of such things as: success in joining the delivery tree; actual core affiliation.

- o+ CBT-DR-ADV-NOTIFICATION: multicast to the CBT ``all-routers'' address, this message is sent subsequent to receiving a CBT-DR-SOLICITATION, but prior to any CBT-DR-ADVERTISEMENT being sent. It acts as a tie-breaking mechanism should more than one router on the subnet think itself the best next-hop to the addressed core. It also promts an already established DR to announce itself as such if it has not already done so in response to a CBT-DR-SOLICITATION.

_9. _I_n_t_e_r_o_p_e_r_a_b_i_l_i_t_y _I_s_s_u_e_s

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.

_1_0. _C_B_T _S_e_c_u_r_i_t_y _A_r_c_h_i_t_e_c_t_u_r_e

see current I-D: draft-ietf-idmr-mkd-02.txt

Acknowledgements

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

Thanks also to 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.

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.

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

Nitin Jain,
Bay Networks, Inc.
3 Federal Street,
Billerica, MA 01821,
USA.

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

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

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

References

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

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

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

[4] A. J. Ballardie. Scalable Multicast Key Distribution (draft-ietf-idmr-mkd-02.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.