                Core Based Trees (CBT version 2) Multicast Routing

                        -- Protocol Specification --

Status of this Memo

   This memo defines an Experimental Protocol for the Internet
   community. It does not specify an Internet standard of any kind.
   Discussion and suggestions for improvement are requested.
   Distribution of this memo is unlimited.

Abstract

   This document describes the Core Based Tree (CBT version 2) network
   layer multicast routing protocol. CBT builds a shared multicast
   distribution tree per group, and is suited to inter- and intra-domain
   multicast routing.

   CBT may use a separate multicast routing table, or it may use that of
   underlying unicast routing, to establish paths between senders and
   receivers. The CBT architecture is described in [1].

   This document is progressing through the IDMR working group of the
   IETF. CBT related documents include [1, 5, 6]. For all IDMR-related
   documents, see http://www.cs.ucl.ac.uk/ietf/idmr.

TABLE OF CONTENTS

1. Changes from CBT version 1

    This version of the CBT protocol specification differs significantly
    from the previous version. Consequently, this version represents
    version 2 of the CBT protocol. CBT version 2 is not, and was not,
    intended to be backwards compatible with version 1; we do not expect
    this to cause extensive compatibility problems because we do not
    believe CBT is at all widely deployed at this stage. However, any
    future versions of CBT can be expected to be backwards compatible
    with this version.

   The most significant changes to version 2 compared to version 1
   include:

   o new LAN mechanisms, including the incorporation of an HELLO
     protocol.

   o new simplified packet formats, with the definition of a common CBT
     control packet header.

   o each group shared tree has only one active core router.

     This specification revision is a complete re-write of the previous
     revision.

2. Introduction & Terminology

   In CBT, a "core router" (or just "core") is a router which acts as a
   "meeting point" between a sender and group receivers. The term
   "rendezvous point (RP)" is used equivalently in some contexts [2]. A
   core router need not be configured to know it is a core router.

   A router that is part of a CBT distribution tree is known as an "on-
   tree" router. An on-tree router maintains active state for the group.

   We refer to a broadcast interface as any interface that supports
   multicast transmission.

   An "upstream" interface (or router) is one which is on the path
   towards the group's core router with respect to this interface (or
   router). A "downstream" interface (or router) is one which is on the
   path away from the group's core router with respect to this interface
   (or router).

   Other terminology is introduced in its context throughout the text.

3. CBT Functional Overview

   The CBT protocol is designed to build and maintain a shared multicast
   distribution tree that spans only those networks and links leading to
   interested receivers.

   To achieve this, a host first expresses its interest in joining a
   group by multicasting an IGMP host membership report [3] across its
   attached link. On receiving this report, a local CBT aware router
   invokes the tree joining process (unless it has already) by
   generating a JOIN_REQUEST message, which is sent to the next hop on
   the path towards the group's core router (how the local router
   discovers which core to join is discussed in section 8). This join

message must be explicitly acknowledged (JOIN_ACK) either by the core
router itself, or by another router that is on the path between the
sending router and the core, which itself has already successfully
joined the tree.

The join message sets up transient join state in the routers it
traverses, and this state consists of <group, incoming interface,
outgoing interface>. "Incoming interface" and "outgoing interface"
may be "previous hop" and "next hop", respectively, if the
corresponding links do not support multicast transmission. "Previous
hop" is taken from the incoming control packet's IP source address,
and "next hop" is gleaned from the routing table - the next hop to
the specified core address. This transient state eventually times out
unless it is "confirmed" with a join acknowledgement (JOIN_ACK) from
upstream. The JOIN_ACK traverses the reverse path of the
corresponding join message, which is possible due to the presence of
the transient join state. Once the acknowledgement reaches the router
that originated the join message, the new receiver can receive
traffic sent to the group.

Loops cannot be created in a CBT tree because a) there is only one
active core per group, and b) tree building/maintenance scenarios
which may lead to the creation of tree loops are avoided. For
example, if a router's upstream neighbour becomes unreachable, the
router immediately "flushes" all of its downstream branches, allowing
them to individually rejoin if necessary. Transient unicast loops do
not pose a threat because a new join message that loops back on
itself will never get acknowledged, and thus eventually times out.

The state created in routers by the sending or receiving of a
JOIN_ACK is bi-directional - data can flow either way along a tree
"branch", and the state is group specific - it consists of the group
address and a list of local interfaces over which join messages for
the group have previously been acknowledged. There is no concept of
"incoming" or "outgoing" interfaces, though it is necessary to be
able to distinguish the upstream interface from any downstream
interfaces. In CBT, these interfaces are known as the "parent" and
"child" interfaces, respectively. A router is not considered "on-
tree" until it has received a JOIN_ACK for a previously sent
JOIN_REQUEST.

With regards to the information contained in the multicast forwarding
cache, on link types not supporting native multicast transmission an
on-tree router must store the address of a parent and any children.
On links supporting multicast however, parent and any child
information is represented with local interface addresses (or similar
identifying information, such as an interface "index") over which the
parent or child is reachable.

Data from non-member senders must be encapsulated (IP-in-IP) by the
first-hop router, and is unicast to the group's core router.
Consequently, no group state is required in the network between the
first hop router and the group's core. On arriving at the core
router, the data packet's outer encapsulating header is removed and
the packet is disemminated over the group shared tree as described
below.

When a multicast data packet arrives at a router, the router uses the
group address as an index into the multicast forwarding cache. A copy
of the incoming multicast data packet is forwarded over each
interface (or to each address) listed in the entry except the
incoming interface.

Each router that comprises a CBT multicast tree, except the core
router, is responsible for maintaining its upstream link, provided it
has interested downstream receivers, i.e. the child interface list is
not NULL. A child interface is one over which a member host is
directly attached, or one over which a downstream on-tree router is
attached. This "tree maintenance" is achieved by each downstream
router periodically sending a CBT "keepalive" message (ECHO_REQUEST)
to its upstream neighbour, i.e. its parent router on the tree. One
keepalive message is sent to represent entries with the same parent,
thereby improving scalability on links which are shared by many
groups. On multicast capable links, a keepalive is multicast to the
"all-cbt-routers" group (IANA assigned as 224.0.0.15); this has a
suppressing effect on any other router for which the link is its
parent link. If a parent link does not support multicast
transmission, keepalives are unicast.

The receipt of a keepalive message over a valid child interface
prompts a response (ECHO_REPLY), which is either unicast or
multicast, as appropriate. The ECHO_REPLY message carries a list of
groups for which the corresponding interface is a child interface.

It cannot be assumed all of the routers on a multi-access link have a
uniform view of unicast routing; this is particularly the case when a
multi-access link spans two or more unicast routing domains. This
could lead to multiple upstream tree branches being formed (an error
condition) unless steps are taken to ensure all routers on the link
agree which is the upstream router for a particular group. CBT
routers attached to a multi-access link participate in an explicit
election mechanism that elects a single router, the designated router
(DR), as the link's upstream router for all groups. Since the DR
might not be the link's best next-hop for a particular core router,
this may result in join messages being re-directed back across a
multi-access link. If this happens, the re-directed join message is
unicast across the link by the DR to the best next-hop, thereby

preventing a looping scenario. This re-direction only ever applies to
join messages. Whilst this is suboptimal for join messages, which
are generated infrequently, multicast data never traverses a link
more than once (either natively, or encapsulated).

In all but the exception case described above, all CBT control
messages are multicast over multicast supporting links to the "all-
cbt- routers" group, with IP TTL 1. The IP source address of CBT
control messages is the outgoing interface of the sending router. The
IP destination address of CBT control messages is either the "all-
cbt- routers" group address, or a unicast address, as appropriate.
All the necessary addressing information is obtained by on-tree
routers as part of tree set up.

If CBT is implemented over a tunnelled topology, when sending a CBT
control packet over a tunnel interface, the sending router uses as
the packet's IP source address the local tunnel end point address,
and the remote tunnel end point address as the packet's IP
destination address.

4. Protocol Specification Details

   Details of the CBT protocol are presented in the context of a single
   router implementation.

4.1. CBT HELLO Protocol

   The HELLO protocol is used to elect a designated router (DR) on
   broadcast-type links. It is also used to elect a designated border
   router (BR) when interconnecting a CBT domain with other domains (see
   [5]). Alternatively, the designated BR may be elected as a matter of
   local policy.

   A router represents its status as a link's DR by setting the DR-flag
   on that interface; a DR flag is associated with each of a router's
   broadcast interfaces. This flag can only assume one of two values:
   TRUE or FALSE. By default, this flag is FALSE.

   A network manager can preference a router's DR eligibility by
   optionally configuring an HELLO preference, which is included in the
   router's HELLO messages. Valid configuration values range from 1 to
   254 (decimal), 1 representing the "most eligible" value. In the
   absence of explicit configuration, a router assumes the default HELLO
   preference value of 255. The elected DR uses HELLO preference zero
   (0) in HELLO advertisements, irrespective of any configured
   preference. The DR continues to use preference zero for as long as
   it is running.

HELLO messages are multicast periodically to the all-cbt-routers
group, 224.0.0.15, using IP TTL 1. The advertisement period is
[HELLO_INTERVAL] seconds.

HELLO messages have a suppressing effect on those routers which would
advertise a "lesser preference" in their HELLO messages; a router
resets its [HELLO_INTERVAL] if the received HELLO is "better" than
its own. Thus, in steady state, the HELLO protocol incurs very little
traffic overhead.

The DR election winner is that which advertises the lowest HELLO
preference, or the lowest-addressed in the event of a tie.

The situation where two or more routers attached to the same
broadcast link areadvertising HELLO preference 0 should never arise.
However, should this situation arise, all but the lowest addressed
zero advertising router relinquishes its claim as DR immediately by
unsetting the DR flag on the corresponding interface. The
relinquishing router(s) subsequently advertise their previously used
preference value in HELLO advertisements.

4.1.1. Sending HELLOs

When a router starts up, it multicasts two HELLO messages over each
of its broadcast interfaces in successsion. The DR flag is initially
unset (FALSE) on each broadcast interface. This avoids the situation
in which each router on a multi-access subnet believes it is the DR,
thus preventing the multiple forwarding of join-requests should they
arrive during this start up period. If no "better" HELLO message is
received after HOLDTIME seconds, the router assumes the role of DR on
the corresponding interface.

A router sends an HELLO message whenever its [HELLO_INTERVAL]
expires. Whenever a router sends an HELLO message, it resets its
hello timer.

4.1.2. Receiving HELLOs

A router does not respond to an HELLO message if the received HELLO
is "better" than its own, or equally preferenced but lower addressed.

A router must respond to an HELLO message if that received is lesser
preferenced (or equally preferenced but higher addressed) than would
be sent by this router over the same interface. This response is sent
on expiry of an interval timer which is set between zero (0) and
[HOLDTIME] seconds when the lesser preferenced HELLO message is
received.

4.2. JOIN_REQUEST Processing

   A JOIN_REQUEST is the CBT control message used to register a member
   host's interest in joining the distribution tree for the group.

4.2.1. Sending JOIN_REQUESTs

   A JOIN_REQUEST can only ever be originated by a leaf router, i.e. a
   router with directly attached member hosts. This join message is sent
   hop-by-hop towards the core router for the group (see section 8).
   The originating router caches <group, NULL, upstream interface> state
   for each join it originates. This state is known as "transient join
   state". The absence of a "downstream interface" (NULL) indicates
   that this router is the join message originator, and is therefore
   responsible for any retransmissions of this message if a response is
   not received within [RTX_INTERVAL]. It is an error if no response is
   received after [JOIN_TIMEOUT] seconds. If this error condition
   occurs, the joining process may be re-invoked by the receipt of the
   next IGMP host membership report from a locally attached member host.

   Note that if the interface over which a JOIN_REQUEST is to be sent
   supports multicast, the JOIN_REQUEST is multicast to the all-cbt-
   routers group, using IP TTL 1. If the link does not support
   multicast, the JOIN_REQUEST is unicast to the next hop on the unicast
   path to the group's core.

4.2.2. Receiving JOIN_REQUESTs

   On broadcast links, JOIN_REQUESTs which are multicast may only be
   forwarded by the link's DR. Other routers attached to the link may
   process the join (see below). JOIN_REQUESTs which are multicast over
   a point-to-point link are only processed by the router on the link
   which does not have a local interface corresponding to the join's
   network layer (IP) source address. Unicast JOIN_REQUESTs may only be
   processed by the router which has a local interface corresponding to
   the join's network layer (IP) destination address.

   With regard to forwarding a received JOIN_REQUEST, if the receiving
   router is not on-tree for the group, and is not the group's core
   router, and has not already forwarded a join for the same group, the
   join is forwarded to the next hop on the path towards the core. The
   join is multicast, or unicast, according to whether the outgoing
   interface supports multicast. The router caches the following
   information with respect to the forwarded join: <group, downstream
   interface, upstream interface>. Subsequent JOIN_REQUESTs received for
   the same group are cached until this router has received a JOIN_ACK
   for the previously sent join, at which time any cached joins can also
   be acknowledged.

If this transient join state is not "confirmed" with a join
acknowledgement (JOIN_ACK) message from upstream, the state is timed
out after [TRANSIENT_TIMEOUT] seconds.

If the receiving router is the group's core router, the join is
"terminated" and acknowledged by means of a JOIN_ACK. Similarly, if
the router is on-tree and the JOIN_REQUEST arrives over an interface
that is not the upstream interface for the group, the join is
acknowledged.

If a JOIN_REQUEST for the same group is scheduled to be sent over the
corresponding interface (i.e. awaiting a timer expiry), the
JOIN_REQUEST is unscheduled.

If this router has a cache-deletion-timer [CACHE_DEL_TIMER] running
on the arrival interface for the group specified in a multicast join,
the timer is cancelled.

4.3. JOIN_ACK Processing

A JOIN_ACK is the mechanism by which an interface is added to a
router's multicast forwarding cache; thus, the interface becomes part
of the group distribution tree.

4.3.1. Sending JOIN_ACKs

The JOIN_ACK is sent over the same interface as the corresponding
JOIN_REQUEST was received. The sending of the acknowledgement causes
the router to add the interface to its child interface list in its
forwarding cache for the group, if it is not already.

A JOIN_ACK is multicast or unicast, according to whether the outgoing
interface supports multicast transmission or not.

4.3.2. Receiving JOIN_ACKs

The group and arrival interface must be matched to a <group, ...,
upstream interface> from the router's cached transient state. If no
match is found, the JOIN_ACK is discarded. If a match is found, a
CBT forwarding cache entry for the group is created, with "upstream
interface" marked as the group's parent interface.

If "downstream interface" in the cached transient state is NULL, the
JOIN_ACK has reached the originator of the corresponding
JOIN_REQUEST; the JOIN_ACK is not forwarded downstream. If
"downstream interface" is non-NULL, a JOIN_ACK for the group is sent

over the "downstream interface" (multicast or unicast, accordingly).
This interface is installed in the child interface list of the
group's forwarding cache entry.

Once transient state has been confirmed by transferring it to the
forwarding cache, the transient state is deleted.

4.4. QUIT_NOTIFICATION Processing

A CBT tree is "pruned" in the direction downstream-to-upstream
whenever a CBT router's child interface list for a group becomes
NULL.

4.4.1. Sending QUIT_NOTIFICATIONs

A QUIT_NOTIFICATION is sent to a router's parent router on the tree
whenever the router's child interface list becomes NULL. If the link
over which the quit is to be sent supports multicast transmission, if
the sending router is the link's DR the quit is unicast, otherwise it
is multicast.

A QUIT_NOTIFICATION is not acknowledged; once sent, all information
pertaining to the group it represents is deleted from the forwarding
cache immediately.

To help ensure consistency between a child and parent router given
the potential for loss of a QUIT_NOTIFICATION, a total of [MAX_RTX]
QUIT_NOTIFICATIONs are sent, each HOLDTIME seconds after the previous
one.

The sending of a quit (the first) also invokes the sending of a
FLUSH_TREE message over each downstream interface for the
corresponding group.

4.4.2. Receiving QUIT_NOTIFICATIONs

The group reported in the QUIT_NOTIFICATION must be matched with a
forwarding cache entry. If no match is found, the QUIT_NOTIFICATION
is ignored and discarded. If a match is found, if the arrival
interface is a valid child interface in the group entry, how the
router proceeds depends on whether the QUIT_NOTIFICATION was
multicast or unicast.

If the QUIT_NOTIFICATION was unicast, the corresponding child
interface is deleted from the group's forwarding cache entry, and no
further processing is required.

If the QUIT_NOTIFICATION was multicast, and the arrival interface is
a valid child interface for the specified group, the router sets a
cache-deletion-timer [CACHE_DEL_TIMER].

Because this router might be acting as a parent router for multiple
downstream routers attached to the arrival link, [CACHE_DEL_TIMER]
interval gives those routers that did not send the QUIT_NOTIFICA-
TION, but received it over their parent interface, the opportunity to
ensure that the parent router does not remove the link from its child
interface list. Therefore, on receipt of a multicast
QUIT_NOTIFICATION over a parent interface, a receiving router
schedules a JOIN_REQUEST for the group for sending at a random
interval between 0 (zero) and HOLDTIME seconds. If a multicast
JOIN_REQUEST is received over the corresponding interface (parent)
for the same group before this router sends its own scheduled
JOIN_REQUEST, it unschedules the multicasting of its own
JOIN_REQUEST.

4.5. ECHO_REQUEST Processing

The ECHO_REQUEST message allows a child to monitor reachability to
its parent router for a group (or range of groups if the parent
router is the parent for multiple groups). Group information is not
carried in ECHO_REQUEST messages.

4.5.1. Sending ECHO_REQUESTs

Whenever a router creates a forwarding cache entry due to the receipt
of a JOIN_ACK, the router begins the periodic sending of ECHO_REQUEST
messages over its parent interface. The ECHO_REQUEST is multicast to
the "all-cbt-routers" group over multicast-capable interfaces, unless
the sending router is the DR on the interface over which the
ECHO_REQUEST is being sent, in which case it is unicast (as is the
corresponding ECHO_REPLY).

ECHO_REQUEST messages are sent at [ECHO_INTERVAL] second intervals.

Whenever an ECHO_REQUEST is sent, [ECHO_INTERVAL] is reset.

If no response is forthcoming, any groups present on the parent
interface will eventually expire [GROUP_EXPIRE_TIME]. This results in
the sending of a QUIT_NOTIFICATION upstream, and sends a FLUSH_TREE
message downstream for each group for which the upstream interface
was the parent interface.

4.5.2. Receiving ECHO_REQUESTs

   If an ECHO_REQUEST is received over any valid child interface, the
   receiving router schedules an ECHO_REPLY message for sending over the
   same interface; the scheduled interval is between 0 (zero) and
   HOLDTIME seconds. This message is multicast to the "all-cbt-routers"
   group over multicast-capable interfaces, and unicast otherwise.

   If a multicast ECHO_REQUEST message arrives via any valid parent
   interface, the router resets its [ECHO_INTERVAL] timer for that
   upstream interface, thereby suppressing the sending of its own
   ECHO_REQUEST over that upstream interface.

4.6.   ECHO_REPLY Processing

   ECHO_REPLY messages allow a child to monitor the reachability of its
   parent, and help ensure the group state information is consistent
   between them.

4.6.1. Sending ECHO_REPLY messages

   An ECHO_REPLY message is sent in response to receiving an
   ECHO_REQUEST message, provided the ECHO_REQUEST is received over any
   one of this router's valid child interfaces. An ECHO_REPLY reports
   all groups for which the link is its child.

   ECHO_REPLY messages are unicast or multicast, as appropriate.

4.6.2. Receiving ECHO_REPLY messages

   An ECHO_REPLY message must be received via a valid parent interface.

   For each group reported in an ECHO_REPLY, the downstream router
   attempts to match the group with one in its forwarding cache for
   which the arrival interface is the group's parent interface. For each
   successful match, the entry is "refreshed". If however, after
   [GROUP_EXPIRE_TIME] seconds a group has not been "refreshed", a
   QUIT_NOTIFICATION is sent upstream, and a FLUSH_TREE message is sent
   downstream, for the group.

   If this router has directly attached members for any of the flushed
   groups, the receipt of an IGMP host membership report for any of
   those groups will prompt this router to rejoin the corresponding
   tree(s).

4.7. FLUSH_TREE Processing

   The FLUSH_TREE (flush) message is the mechanism by which a router
   invokes the tearing down of all its downstream branches for a
   particular group. The flush message is multicast to the "all-cbt-
   routers" group when sent over multicast-capable interfaces, and
   unicast otherwise.

4.7.1. Sending FLUSH_TREE messages

   A FLUSH_TREE message is sent over each downstream (child) interface
   when a router has lost reachability with its parent router for the
   group (detected via ECHO_REQUEST and ECHO_REPLY messages). All group
   state is removed from an interface over which a flush message is
   sent. A flush can specify a single group, or all groups
   (INADDR_ANY).

4.7.2. Receiving FLUSH_TREE messages

   A FLUSH_TREE message must be received over the parent interface for
   the specified group, otherwise the message is discarded.

   The flush message must be forwarded over each child interface for the
   specified group.

   Once the flush message has been forwarded, all state for the group is
   removed from the router's forwarding cache.

5. Non-Member Sending

   Data can be sent to a CBT tree by a sender not attached to the group
   tree. The sending host originates native multicast data, which is
   promiscuously received by a local router, which must be CBT capable.
   It is assumed the local CBT router knows about the relevant <core,
   group> mapping, and thus can encapsulate (IP-in-IP) the data packet
   and unicast it to the corresponding core router. On arriving at the
   core router, the data packet is decapsulated and disemminated over
   the group tree in the manner already described.

6. Timers and Default Values

   This section provides a summary of the timers described above,
   together with their recommended default values. Other values may be
   configured; if so, the values used should be consistent across all
   CBT routers attached to the same network.

   o   [HELLO_INTERVAL]: the interval between sending an HELLO message.
        Default: 60 seconds.

     o   [HELLO_PREFERENCE]: Default: 255.

     o   [HOLDTIME]: generic response interval. Default: 3 seconds.

     o   [MAX_RTX]: default maximum number of retransmissions. Default 3.

     o   [RTX_INTERVAL]: message retransmission time. Default: 5 seconds.

     o   [JOIN_TIMEOUT]: raise exception due to tree join failure.
            Default: 3.5 times [RTX_INTERVAL].

     o   [TRANSIENT_TIMEOUT]: delete (unconfirmed) transient state.
            Default: (1.5*RTX_INTERVAL) seconds.

     o   [CACHE_DEL_TIMER]: remove child interface from forwarding cache.
            Default: (1.5*HOLDTIME) seconds.

     o   [GROUP_EXPIRE_TIME]: time to send a QUIT_NOTIFICATION to our
            non-responding parent. Default: (1.5*ECHO_INTERVAL).

     o   [ECHO_INTERVAL]: interval between sending ECHO_REQUEST to parent
            routers. Default: 60 seconds.

     o   [EXPECTED_REPLY_TIME]: consider parent unreachable. Default: 70
            seconds.

7. CBT Packet Formats and Message Types

   CBT control packets are encapsulated in IP. CBT has been assigned IP
   protocol number 7 by IANA [4].

7.1. CBT Common Control Packet Header

   All CBT control messages have a common fixed length header.

```
      0                   1                   2                   3
      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 | type  |  addr len     |           checksum            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
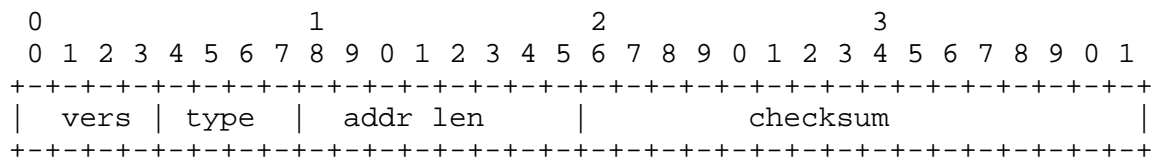

               Figure 1. CBT Common Control Packet Header

   This CBT specification is version 2.

CBT packet types are:

o   type 0: HELLO

o   type 1: JOIN_REQUEST

o   type 2: JOIN_ACK

o   type 3: QUIT_NOTIFICATION

o   type 4: ECHO_REQUEST

o   type 5: ECHO_REPLY

o   type 6: FLUSH_TREE

o   type 7: Bootstrap Message (optional)

o   type 8: Candidate Core Advertisement (optional)

o   Addr Length: address length in bytes of unicast or multicast
       addresses carried in the control packet.

o   Checksum: the 16-bit one's complement of the one's complement
       sum of the entire CBT control packet.

7.2. HELLO Packet Format

```
     0                   1                   2                   3
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                   CBT Control Packet Header                   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Preference  |  option type  |  option len   | option value |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
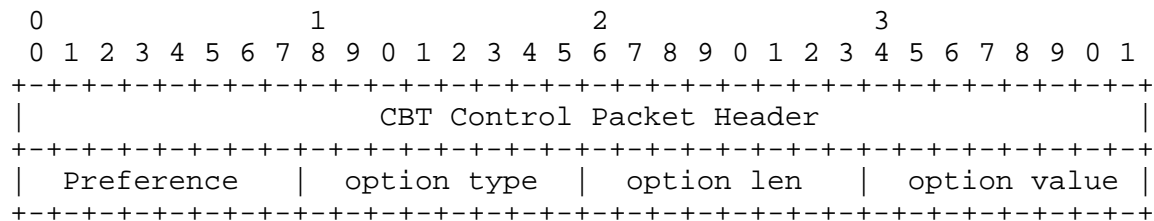
Figure 2. HELLO Packet Format

HELLO Packet Field Definitions:

o   preference: sender's HELLO preference.

o   option type: the type of option present in the "option value"
       field. One option type is currently defined: option type 0
       (zero) = BR_HELLO; option value 0 (zero); option length 0
       (zero). This option type is used with HELLO messages sent by a

        border router (BR) as part of designated BR election (see [5]).

   o   option len: length of the "option value" field in bytes.

   o   option value: variable length field carrying the option value.

7.3. JOIN_REQUEST Packet Format


```
        0                   1                   2                   3
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                   CBT Control Packet Header                  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                        group address                        |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                        target router                        |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                      originating router                     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  option type  |  option len   |        option value         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
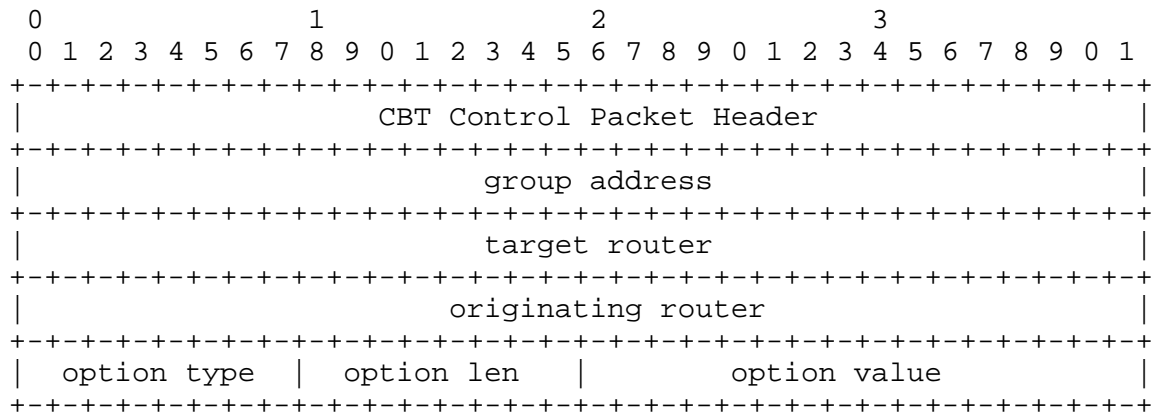
                  Figure 3. JOIN_REQUEST Packet Format

       JOIN_REQUEST Field Definitions

   o   group address: multicast group address of the group being joined.
        For a "wildcard" join (see [5]), this field contains the value of
        INADDR_ANY.

   o   target router: target (core) router for the group.

   o   originating router: router that originated this JOIN_REQUEST.

   o   option type, option len, option value: see HELLO packet format,
        section 7.2.

7.4. JOIN_ACK Packet Format

       JOIN_ACK Field Definitions

   o   group address: multicast group address of the group being joined.

   o   target router: router (DR) that originated the corresponding
        JOIN_REQUEST.

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   CBT Control Packet Header                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       group address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       target router                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| option type   | option len    |         option value         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
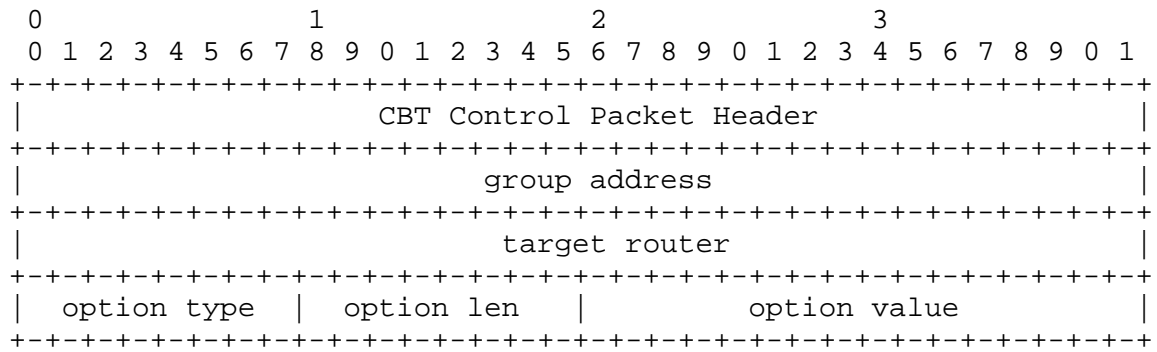
                  Figure 4. JOIN_ACK Packet Format

   o   option type, option len, option value: see HELLO packet format,
       section 7.2.

7.5. QUIT_NOTIFICATION Packet Format

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   CBT Control Packet Header                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       group address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   originating child router                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

              Figure 5. QUIT_NOTIFICATION Packet Format

       QUIT_NOTIFICATION Field Definitions

   o   group address: multicast group address of the group being joined.

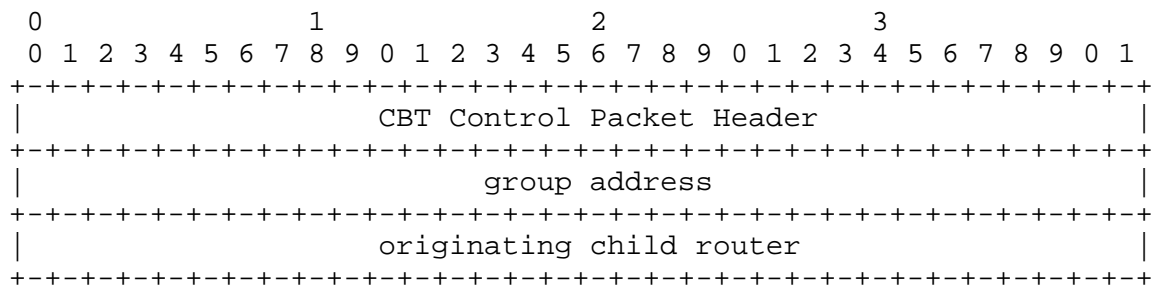   o   originating child router: address of the router that
       originates the QUIT_NOTIFICATION.

7.6.   ECHO_REQUEST Packet Format
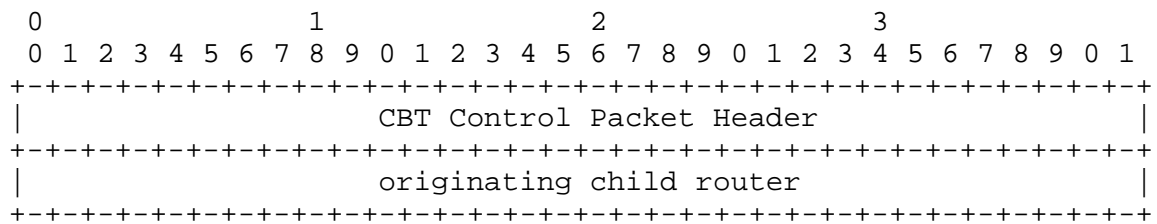
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   CBT Control Packet Header                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    originating child router                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                 Figure 6. ECHO_REQUEST Packet Format

       ECHO_REQUEST Field Definitions

   o   originating child router: address of the router that
        originates the ECHO_REQUEST.

7.7.   ECHO_REPLY Packet Format
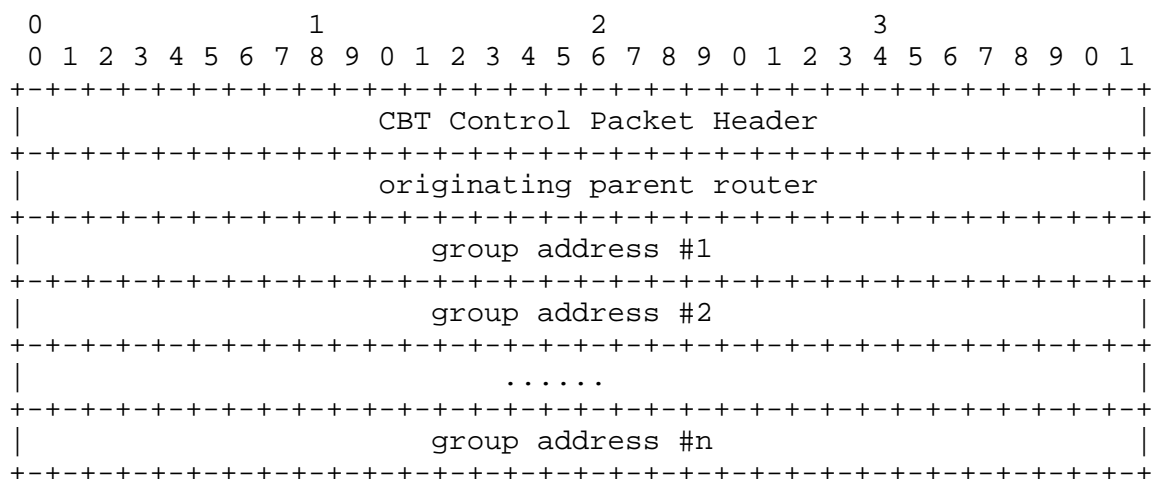

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   CBT Control Packet Header                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    originating parent router                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       group address #1                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       group address #2                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          ......                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       group address #n                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                  Figure 7. ECHO_REPLY Packet Format

       ECHO_REPLY Field Definitions

   o   orininating parent router: address of the router originating
        this ECHO_REPLY.

   o   group address: a list of multicast group addresses for which

this router considers itself a parent router w.r.t. the link
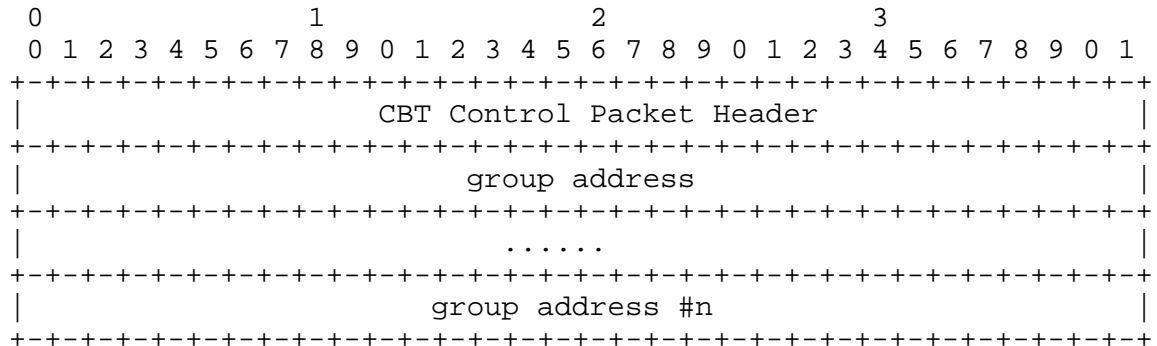over which this message is sent.

7.8. FLUSH_TREE Packet Format


```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   CBT Control Packet Header                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         group address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           ......                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       group address #n                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8. FLUSH_TREE Packet Format

FLUSH_TREE Field Definitions

o   group address(es): multicast group address(es) of the group(s)
     being "flushed".

8. Core Router Discovery

There are two available options for CBTv2 core discovery; the
"bootstrap" mechanism (as currently specified with the PIM sparse
mode protocol [2]) is applicable only to intra-domain core discovery,
and allows for a "plug & play" type operation with minimal
configuration. The disadvantage of the bootstrap mechanism is that
it is much more difficult to affect the shape, and thus optimality,
of the resulting distribution tree. Also, to be applicable, all CBT
routers within a domain must implement the bootstrap mechanism.

The other option is to manually configure leaf routers with <core,
group> mappings (note: leaf routers only); this imposes a degree of
administrative burden - the mapping for a particular group must be
coordinated across all leaf routers to ensure consistency. Hence,
this method does not scale particularly well. However, it is likely
that "better" trees will result from this method, and it is also the
only available option for inter-domain core discovery currently
available.

8.1. "Bootstrap" Mechanism Overview

   It is unlikely that the bootstrap mechanism will be appended to a
   well-known network layer protocol, such as IGMP [3], though this
   would facilitate its ubiquitous (intra-domain) deployment. Therefore,
   each multicast routing protocol requiring the bootstrap mechanism
   must implement it as part of the multicast routing protocol itself.

   A summary of the operation of the bootstrap mechanism follows
   (details are provided in [7]). It is assumed that all routers within
   the domain implement the "bootstrap" protocol, or at least forward
   bootstrap protocol messages.

   A subset of the domain's routers are configured to be CBT candidate
   core routers. Each candidate core router periodically (default every
   60 secs) advertises itself to the domain's Bootstrap Router (BSR),
   using "Core Advertisement" messages. The BSR is itself elected
   dynamically from all (or participating) routers in the domain. The
   domain's elected BSR collects "Core Advertisement" messages from
   candidate core routers and periodically advertises a candidate core
   set (CC-set) to each other router in the domain, using traditional
   hop- by-hop unicast forwarding. The BSR uses "Bootstrap Messages" to
   advertise the CC-set. Together, "Core Advertisements" and "Bootstrap
   Messages" comprise the "bootstrap" protocol.

   When a router receives an IGMP host membership report from one of its
   directly attached hosts, the local router uses a hash function on the
   reported group address, the result of which is used as an index into
   the CC-set. This is how local routers discover which core to use for
   a particular group.

   Note the hash function is specifically tailored such that a small
   number of consecutive groups always hash to the same core.
   Furthermore, bootstrap messages can carry a "group mask", potentially
   limiting a CC-set to a particular range of groups. This can help
   reduce traffic concentration at the core.

   If a BSR detects a particular core as being unreachable (it has not
   announced its availability within some period), it deletes the
   relevant core from the CC-set sent in its next bootstrap message.
   This is how a local router discovers a group's core is unreachable;
   the router must re-hash for each affected group and join the new core
   after removing the old state. The removal of the "old" state follows
   the sending of a QUIT_NOTIFICATION upstream, and a FLUSH_TREE message
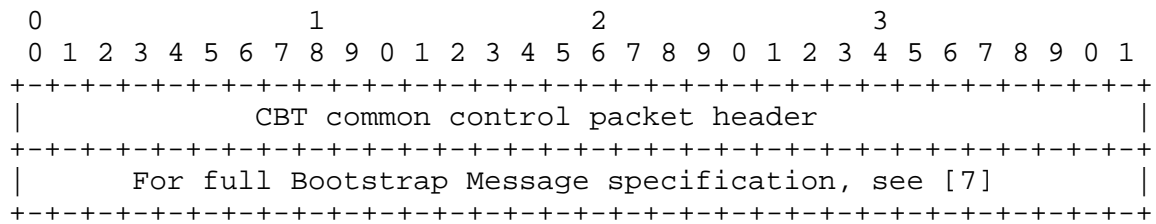   downstream.

8.2. Bootstrap Message Format

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  CBT common control packet header             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       For full Bootstrap Message specification, see [7]       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

               Figure 9. Bootstrap Message Format

8.3. Candidate Core Advertisement Message Format


```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  CBT common control packet header             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    For full Candidate Core Adv. Message specification, see [7] |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
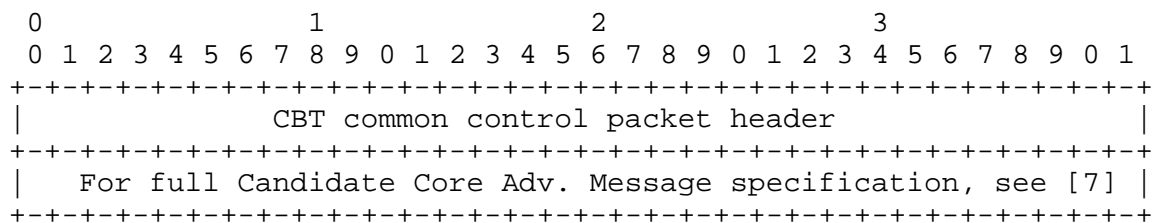
        Figure 10. Candidate Core Advertisement Message Format

9. Interoperability Issues

   Interoperability between CBT and DVMRP is specified in [5].

   Interoperability with other multicast protocols will be fully
   specified as the need arises.

10. Security Considerations

   Security considerations are not addressed in this memo.

   Whilst multicast security is a topic of ongoing research, multicast
   applications (users) nevertheless have the ability to take advantage
   of security services such as encryption or/and authentication
   provided such services are supported by the applications.

   RFCs 1949 and 2093/2094 discuss different ways of distributing
   multicast key material, which can result in the provision of network
   layer access control to a multicast distribution tree.

   [9] offers a synopsis of multicast security threats and proposes some
   possible counter measures.

Beyond these, little published work exists on the topic of multicast security.

Acknowledgements

References

[1] Core Based Trees (CBT) Multicast Routing Architecture; A. Ballardie; RFC 2201, September 1997.

[2] Protocol Independent Multicast (PIM) Sparse Mode/Dense Mode; D. Estrin et al; ftp://netweb.usc.edu/pim   Working drafts, 1996.

[3] Internet Group Management Protocol, version 2 (IGMPv2); W. Fenner; ftp://ds.internic.net/internet-drafts/draft-ietf-idmr-igmp-v2-**.txt. Working draft, 1996.

[4] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[5] CBT Border Router Specification for Interconnecting a CBT Stub Region to a DVMRP Backbone; A. Ballardie; ftp://ds.internic.net/internet-drafts/draft-ietf-idmr-cbt-dm-interop-**.txt. Working draft, March 1997.

[6] Ballardie, A., "Scalable Multicast Key Distribution", RFC 1949, July 1996.

   [7] A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast
   Routing; D. Estrin et al.; Technical Report;
   ftp://catarina.usc.edu/pim

   [8] The Ordered Core Based Tree Protocol; C. Shields and J.J. Garcia-
   Luna-Aceves; In Proceedings of IEEE Infocom'97, Kobe, Japan, April
   1997;
   http://www.cse.ucsc.edu/research/ccrg/publications/infocomm97ocbt.ps.gz

   [9] Multicast-Specific Security Threats and Counter-Measures; A.
   Ballardie and J. Crowcroft; In Proceedings "Symposium on Network and
   Distributed System Security", February 1995, pp.2-16.

Author Information:

   Tony Ballardie,
   Research Consultant

   EMail: ABallardie@acm.org