Core Based Tree (CBT) Multicast

- Architectural Overview and Specification -

by

Tony Ballardie

University College London

Status of this Draft

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts. Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Abstract

CBT is a new architecture for local and wide-area IP multicasting, being unique in its utilization of just one shared delivery tree, as opposed to the source-based delivery tree approach of existing IP multicast schemes, such as DVMRP [5] and MOSPF [11].

The primary advantages of the CBT approach are that it offers more favourable scaling characteristics than do existing multicast algorithms, and is independent of whichever underlying unicast routing protocol is operating.

This draft describes the CBT protocol in detail, as well as the CBT architecture. The definition of a new network layer multicast protocol has also meant that it has been possible to integrate a much enriched functionality into multicast that is not possible under other IP multicast schemes, for example, the integration of security features and resource reservation.

CBT has been designed to interoperate with existing IP multicast techniques, as well as other new IP multicast proposals, such as PIM [14]. Interoperation will be described in detail. We also introduce a solution to the *multicast key distribution* problem. Some open and, as yet, unresolved issues are also discussed.

CBT Expires March 10th, 1995.

Contents

1 Background **1**

2 Introduction **1**

3 Document Layout **1**

Part A **2**

1 Protocol Overview **2**
1.1 CBT Group Initiation 2
1.2 Tree Joining Process 2
1.3 Tree Leaving Process 4
1.4 Tree Maintenance Issues 4
1.4.1 Node Failure 4
1.4.2 Core Failure 5
1.4.3 Unicast Transient Loops 5
1.4.4 Loop Detection 6
1.5 Core Placement 7
1.6 LAN Designated Router 7
1.7 Non-Member Sending 8
1.8 Data Packet Forwarding 8
1.9 Some Interoperability Specifics 10
1.10 Data Packet Encapsulation and Translation 11
1.10.1 Originating Case 11
1.10.2 Receiving Case 13
1.11 Lower Group Leave Latency 15

Part B **16**

1 CBT Packet Formats and Message Types **16**
1.1 CBT Header Format 17
1.2 Control Packet Header Format 18
1.3 Primary Maintenance Message Types 19
1.4 Auxilliary Maintenance Message Types 20

21

Part C **21**

1 Interoperability Issues **21**
1.1 Isolation of CBT Routes 21
1.2 Backwards Compatibility of Data Packets 22
1.3 Tree Overlap of CBT with Other Schemes 23
1.4 IGMP in the Presence of Multiple Protocols 24

2 Resource Reservation **25**

3 CBT Security Architecture **26**
3.1 Security Example – CBT for Multicast Key Distribution 27

Part D **29**

1 Suggested Additional Properties of a Multicast Architecture **29**

2 CBT - The New Architecture **30**
2.1 Architectural Overview 30
2.2 Architectural Justification 33

3 Disadvantages of the CBT Architecture **34**

1 Background

Centre based forwarding was first described in the early 1980s by Wall [17] in his PhD thesis on broadcast and selective broadcast. At this time, multicast was in its very earliest stages of development, and researchers were only just beginning to realise the benefits that could be gained from it, and some of the uses it could be put to. It was only later that the class-D multicast address space was defined, and later again that intrinsic multicast support was taken advantage of for broadcast media, such as Ethernet.

Now that we have several years practical experience with multicast [3], a diversity of multicast applications, and an internetwork infrastructure that wants to support it to an ever-increasing degree, we re-visit the centre-based forwarding paradigm introduced by Wall, and mould and adapt it specifically for today's multicast environment. We will indeed see that an old idea can go a long way.

2 Introduction

Multicast group communication is an increasingly important capability in many of today's data networks. Most LANs and more recent wide-area network technologies such as SMDS [12] and ATM [6] specify multicast as part of their service.

Since the wide-area introduction of multicasting there has been a large increase in the number and diversity of multicast applications, examples of which include audio and video conferencing [3], replicated database updating and querying, software update distribution, stock market information services, and more recently, resource discovery [10]. Multimedia [18, 16] is another fast expanding area for which multicast offers an invaluable service. It has therefore been necessary of late to address the topic of scalability with regards to multicast algorithms, since, if they do not scale to an internetwork size that is expected (given the growth rate of the last several years), they can not be of longlasting benefit. This motivates the need for new multicasting techniques to be investigated.

This draft describes a new multicast routing architecture which is applicable to a datagram network. The CBT architecture has attractive scaling characteristics, and is unicast routing protocol independent. CBT is motivated in [4]. We also present a specification of the CBT multicast protocol for IP networks based on this new architecture.

3 Document Layout

The remainder of this document is divided into four parts: Part A embarks on a protocol overview, discussing protocol engineering design features, such as CBT group initiation, the tree joining process, tree maintenance issues, the tree leaving process, LAN issues, data packet forwarding, and data packet encapsulation and translation[1]. We also introduce

[1]We will refer to the copying (and sometimes manipulation) of various fields of the IP header to a CBT header as *translation* throughout. This may not be in total agreement with how the term is used elsewhere.

a new, backwards compatible version of IGMP that significantly improves group leave latency.

Part B illustrates and describes in detail, individual CBT packet formats and message types.

Part C discusses *interoperability*, how lightweight *resource reservation* is possible as part of the CBT protocol, CBT *security* features, and a proposed solution to the multicast key distribution problem.

Finally, Part D offers a general architectural overview and discussion on the CBT architecture.

Part A

1 Protocol Overview

1.1 CBT Group Initiation

Like any of the other multicast schemes, one user, the *group initiator*, initiates a CBT multicast group. The procedures involved in initiating and joining a CBT group involves a little more user interaction than current IP multicast schemes, for example, it is necessary to supply information such as desired group *scope*, as well as select the *primary* core from a selection of pre-configured core routers. Explicit *core rankings* help prevent loops when the core tree is initially set up. It also assists in the tree maintenance process should the tree become partitioned.

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*.

1.2 Tree Joining Process

Once the cores have been enumerated by a group's initiator, and the application, port number etc. have been selected, the group-initiating host sends a special CORE-NOTIFICATION message to each of them, which is acknowledged. The purpose of this message 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. 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. Routers on the core backbone will usually include not only the cores themselves, but intervening CBT-capable routers on the unicast

path between them. Once this set up is complete, any pending joins for the same group can be acknowledged.

All the CBT-capable routers traversed by a JOIN-ACKnowlegement change their status to *CBT-non-core routers* for the group identified by group-id. It is the JOIN-ACK that actually creates a tree branch. The JOIN-ACK carries the complete core list for the group, which is stored by each of the routers it traverses. Between sending a JOIN-REQUEST and receiving a JOIN-ACK, a router is in a state of *pending* membership. A router that is in the *join pending* state can not send join acknowledgements in response to other join requests received for the same group, but rather caches them for acknowledgement subsequent to its own join being acknowledged. Furthermore, if a router in the pending state gets a better route to the core to which its join was sent, it sends a new join on the better route after cancelling its previous join (this is required to deal with unicast transient loops).

Non-member senders, and new group receivers, are expected to know the address of at least one of the corresponding group's cores in order to send to/join a group. The current specification does not state how this information is gleaned, but it might be obtainable from a directory such as "sd" (the multicast session directory[2]) or from the Domain Name System (DNS)[3].

In accordance with existing IP multicast schemes, CBT multicasting requires the presence of at least one CBT-capable router per subnetwork for hosts on that subnetwork to utilize CBT multicasting. Only one local router, the *designated router*, is allowed to send to/receive from uptree (i.e. the branch leading to/from the core) for a particular group. We therefore make a clear distinction between a *group membership interrogator* – the router responsible for sending IGMP host-membership queries onto the local subnet, and *designated router*. However, they may or may not be one and the same. LAN specifics are discussed in sections 1.6, 1.7 and 1.8.

Once the most appropriate designated router (DR) has been established, i.e. the router that is on the shortest-path to the corresponding core, the new receiver (host) sends a special CBT report to it, requesting that it join the corresponding delivery tree if it has not already. If it has, then the DR multicasts to the group a notification to that effect back across the subnet. Information included in this notification include whether the DR was successful in joining the corresponding tree, and *actual* core affiliation.

> NOTE: the actual core affiliation of a tree router may differ from the core specified in the join request, if that join is terminated by an on-tree router whose affiliation is to a different core.

If the local DR has not joined the tree, then it proceeds to send a JOIN-REQUEST and awaits an acknowledgement, at which time the notification, as described above, is multicast across the subnetwork.

---

[2] By Van Jacobson et al., LBL.

[3] We considered disseminating core identities by including them in link-state routing updates. However, this does not provide scalability since it involves global group information distribution. Further, it involves a dependency on link-state routing

1.3 Tree Leaving Process

A QUIT-REQUEST is a request by a CBT router to leave a group. A QUIT-REQUEST may be sent by a router to detach itself from a tree if and only if it has no members for that group on any directly attached subnets, AND it has received a QUIT-REQUEST on each of its child interfaces for that group (if it has any). The QUIT-REQUEST can only be sent to the parent router. The parent immediately acknowledges the QUIT-REQUEST with a QUIT-ACK and removes that child interface from the tree. Any CBT router that sends a QUIT-ACK in response to receiving a QUIT-REQUEST should itself send a QUIT-REQUEST upstream if the criteria described above are satisfied.

Failure to receive a QUIT-ACK despite several re-transmissions gives the sending router the right to remove the relevant parent interface information, and by doing so, removes itself from the CBT tree for that group.

1.4 Tree Maintenance Issues

Robustness features/mechanisms have been built into the CBT protocol as has been deemed appropriate to ensure timely tree re-configuration in the event of a node or core failure. These mechanisms are implemented in the form of *request-response* messages. Their frequency is configurable, with the trade-off being between protocol overhead and timeliness in detecting a node failure, and recovering from that failure.

1.4.1 Node Failure

The CBT protocol treats core- and non-core failure in the same way, using the same mechanisms to re-establish tree connectivity.

Each child node on a CBT tree monitors 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 (Note: re-joining is not necessary just because unicast calculates a new next-hop to the core).

Subsequent to sending a QUIT-REQUEST on the parent link, a non-core router initially attempts to re-join the tree by sending a RE-JOIN-REQUEST (see section 1.4.4) on an alternate path (the alternate path is derived from unicast routing) to an arbitrary alternate core selected from the core list. The corresponding core is tested for reachability before the re-join is sent, by means of the control message: CBT-CORE-PING. Failure to receive a response from the selected core will result in another being selected, and the process continues to repeat itself until a reachable core is found.

The significance of sending a RE-JOIN-REQUEST (as opposed to a JOIN-REQUEST) is because of the presence of subordinate routers, i.e. there exists a downstream branch

connected to the re-joining router. Care must be taken in this case to avoid loops forming on the tree. If the joining router did not have downstream routers connected to it, it would not be necessary to take precautions to avoid loops since they could not occur (this is explained in more detail in section 1.4.3).

> NOTE: It was an engineering design decision not to flush the complete (downstream) branch when some (upstream) router detects a failure. Whilst each router would join via *its* shortest-path to the corresponding core, it would result in an overall longer *re-connectivity latency.*

A FLUSH-TREE control message is however sent if the best next-hop of the re-join is a child on the same tree.

1.4.2 Core Failure

Once the *core tree* has been established as the initial step of group initiation, core router failure thereafter is handled no differently than non-core router failure, with a core attempting to re-connect itself to the corresponding tree by means of either a join or re-join.

When a core router re-starts subsequent to failure, it will have no knowledge of the tree for which it is supposed to be currently a core. The only means by which it can find out, and therefore re-establish itself on the corresponding tree is if some other on-tree router sends it a CBT-CORE-PING message. This message, by default, always contains the identities of all the cores for a group, together with the group-id.

On receipt of a CBT-CORE-PING, a recently re-started core will re-join the tree by means of a JOIN-REQUEST.

1.4.3 Unicast Transient Loops

Routers rely on underlying unicast routing to carry JOIN-REQUESTs towards the core of a core-based tree. However, subsequent to a topology change, transient routing loops, so called because of their short-lived nature, can form in routing tables whilst the routing algorithm is in the process of converging or stabilizing.

There are two cases to consider with respect to CBT and unicast transient loops, namely:

- a join is sent over a transient loop, but no part of the corresponding CBT tree forms part of that loop. In this case, the join will never get acknowledged and will therefore timeout. Subsequent re-tries will succeed after the transient loop has disappeared.

- a join is sent over a transient loop, and the loop consists either partly or entirely of routers on the corresponding CBT tree. If the loop consists only partly of routers on the tree and the join originated at a router that is not attempting to re-join the tree, then the JOIN-REQUEST will be acknowledged. No further action is necessary since a loop-free path exists from the originating router to the tree.

If the loop consists entirely of routers on the tree, then the router originating the join is attempting to re-join the tree. In this case also, the join could be acknowledged which would result in a loop forming on the tree, so we have designed a loop-detection mechanism which is described below.

1.4.4 Loop Detection

The CBT protocol incorporates an explicit *loop-detection* mechanism. Loop detection is only necessary when a router, with at least one child, is attempting to re-connect itself to the corresponding tree.

We distinguish between three types of JOIN-REQUEST: active; active re-join; and non-active re-join (see Part B, section 1.3).

An *active* JOIN-REQUEST for group A is one which originates from a router which has no chilren belonging to group A.

An *active re-join* for group A is one which originates from a router that has children belonging to group A.

A *non-active re-join* is one that 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, a loop is obviously present in the tree. The router must therefore immediately send a QUIT-REQUEST to its parent router, and attempt to re-join again. In this way the re-join acts as a *loop-detection* packet.

Another scenario that requires consideration is when there is a break in the path (tunnel) between a child and its parent. Although the parent is active, the child believes that the parent is down – the child cannot distinguish between the parent being down and the path to it being down. If the path failure is short-lived, whilst the child will have chosen a new route to the core, the parent will be unaware of this, and will continue forwarding over its child interfaces, the potential risk being apparent.

We guard against this using a *parent assert* mechanism, which is implicit, i.e. involves no control message overhead, in the reception of CBT-ECHO-REQUESTs from a child. If no CBT-ECHO-REQUEST is heard, after a certain interval the corresponding child interface is removed by the parent.

As an additional precaution against packet looping, multicast data packets that are in the process of spanning a CBT's delivery tree branches (remember, we distinguish between actual tree branches and attached subnetworks, although there are cases when they are one and the same) carry an *on-tree* indicator in the CBT header of the packet. Provided a data packet arrives via a valid tree interface, all routers are obliged to check that the on-tree indicator is set accordingly. A data packet arriving at the tree for the first time from a non-member sender will have the *on-tree* indicator bits set by the receiving router. These bits should never subsquently be modified by any router. Should a packet be erroneously forwarded by an on-tree router over an off-tree interface, should that packet somehow work

its way back on tree, it can be immediately recognised and discarded, since it will have arrived via a non-tree interface, but will have its on-tree bits set.

1.5 Core Placement

As it stands, the current implementation of CBT uses trivial heuristics for core placement.

Careful placement of core(s) no doubt assists in optimizing the routes between any sender and group members on the tree. Depending on particular group dynamics, such as sender/receiver population, and traffic patterns, it may well be counter-productive to place a core(s) near or at the centre of a group. In any event, there exists no polynomial time algorithm that can find the centre of a dynamic multicast spanning tree [17].

One suggestion might be that cores be statically configured throughout the Internet - there need only be some relatively small number of cores per backbone network[4], and the addresses of these cores would be "well-known".

Alternatively, and possibly more appropriately, any router could become a core when a host on one of its attached subnets wishes to initiate a group. This is particularly attractive for a one-to-many "broadcast" where the sender remains constant, since, if the sender is the core, the multicast tree formed will be a shortest-path spanning tree rooted at the sender.

We have stressed that the placement of a group's core should positively reflect that group's characteristics. In the absence of any better mechanism, CBT adopts the "hand-selection" approach to selecting a group's cores, based on a judgement of what is known about the network topology between the current members.

1.6 LAN Designated Router

As we have said, there must only ever exist one DR for any particular group that is responsible for uptree forwarding/reception of data packets.

A group's DR is elected by means of an explicit mechanism. Whenever a host initiates/joins a group, part of the process is for it to send a CBT-DR-SOLICITATION message, addressed to the CBT "all-routers" address, which is a request for the best next-hop router to a specified core.

If the group is being initiated, a DR will almost certainly not be present on the local subnet for the group, whereas if a group is being joined, the DR may or may not be present, depending on whether there exist other group members on the LAN (subnet).

If a DR is present for the specified group, it responds to the solicitation with a CBT-DR-ADVERTISEMENT, which is addressed to the group.

[4]The storage and switching overhead incurred by these core routers increases linearly with the number of groups traversing them. A threshold value could be introduced indicating the maximum number of groups permitted to traverse a core router. Once exceeded, additional core routers would need to be assigned to the backbone.

If no DR is present, each CBT router inspects its unicast routing table to establish whether it is the next best-hop to the specified core.

A router which considers itself the best next-hop does not respond immediately with an advertisement, but rather sends a CBT-DR-ADV-NOTIFICATION to the CBT "all-routers" address. This is a precautionary measure to prevent more than one router advertising itself as the DR for the group (it is conceivable that more than one router might think itself as the best next-hop to the core). If this scenario does indeed occur, the advertisement notification acts as a *tie-breaker*, the router with the lowest address winning the election. The lowest addressed router subsequently advertises itself as DR for the group.

1.7 Non-Member Sending

For non-member senders, the presence of a local CBT-capable router is mandatory. The sending of multicast packets from a non-member host to a particular group is two-phase: the first phase involves *multicasting* the packet from the originating host to the local CBT designated router (DR), which will need to have been elected beforehand if not already present for the group. The ultimate destination of the packet is carried in the **core address** field of the CBT header (see Part B, section 1.1.); the second phase is the *unicasting* of the packet from the local DR to its ultimate destination. The DR replaces the destination address in the IP header with the core address contained within the CBT header, and forwards the packet.

These actions will require some host modifications to the IP module, which could be done in parallel with the host modifications required for IP next-generation. The details of host modifications are not given in this document, but will appear in an accompanying document shortly.

Packets sent from a non-member sender will first encounter the corresponding delivery tree either at the addressed core, or hit an on-tree router that is on the shortest-path between the sender and the core. What happens when a CBT packet hits the corresponding delivery tree is dealt with under "Data Packet Forwarding" in section 1.8 below.

1.8 Data Packet Forwarding

In this section we describe how multicast data packets span a CBT tree.

> It is important to note that CBT uses the *Internet Group Management Protocol (IGMP)* in much the same way as traditional IP schemes, namely to establish group presence on directly-connected subnets, and to exchange CBT routing information. A new IGMP message type has been created for exchanging CBT routing messages. Some slight modifications have been made to IGMP specifically for CBT in order to significantly reduce *leave latency* (although the new version of IGMP can be easily adopted by other multicast protocols). This new version of IGMP is described in section 1.11.

We must again bring to the reader's attention the distinction between *tree branches* and *subnets*, although there are cases where they are one and the same.

It has been an important engineering design goal to have CBT interoperate seamlessly with other multicast protocols. It is important to note, however, that CBT routing information is *not* exchanged with that of any other schemes.

When data packets containing a CBT header arrive at a CBT router, either from a subnet (because they originated there), or via a tree branch, they are handled as follows: for all directly-connected subnets with group member presence, various fields are copied from the CBT header to the IP header (some fields being altered in the process[5]), the CBT header is removed, and the packet is forwarded as an IP-style multicast to each such subnet. The IP-style packet is also multicast back across the originating subnet if there is group member presence there. The header translation process is more fully explained in section 1.10.

On a multi-access LAN, the interface to a parent and/or children can be the same as the interface to the subnet itself, containing end-systems. Irrespective of this, data packets sent to children from a parent, or vice-versa, include a CBT header.

On a multi-access LAN over which a CBT router's parent or children is/are reachable, a data packet containing the CBT header is *multicast* over the corresponding LAN interface, and will be received by the corresponding child routers (or parent router). End-systems subscribed to the same group may receive these packets, but they will not be processed, since end-systems will not recognise the upper-layer protocol identifier, i.e. CBT.

> NOTE: it was an engineering design decision to *multicast* data packets with a CBT header on multi-access links – the case of unicasting separately from parent to n children is clearly more costly. Multicasting in the direction of child to parent also reduces traffic, since when the parent receives the packet, it does not need to re-send the packet to any of its other children that may be present on the multi-access link, since they will have received a copy from the child's multicast.

Data arriving at a CBT router is always multicast *first* IP-style onto any directly-connected subnets with group member presence, and only subsequently unicast (multicast on multi-access links) to parent/children with a CBT header.

A CBT router will not forward data packets unless that router has a forwarding information base (FIB) entry for the specified group, unless the packet's CBT header indicates that the packet originated at a directly-connected host (the corresponding field of the CBT header being set to indicate this – see Part B), and the **core address** field of the CBT header is not NULL (in this case, when there is no FIB entry, the local originating host must be a non-member sender). Furthermore, the router will only forward such a packet if it has been elected designated router (DR) for the corresponding group. A FIB entry is shown below.

[5] Throughout, we will refer to the copying/alteration of fields from one header to another, as *translation*.

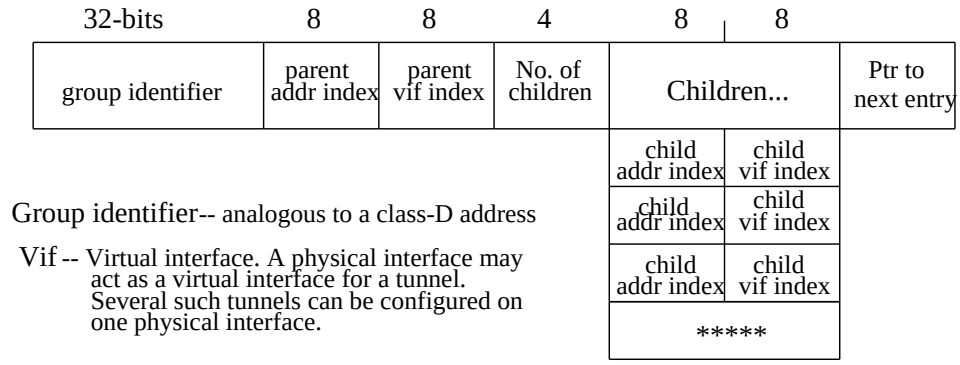

Figure 1: A CBT FIB entry in the author's implementation

If a CBT router receives a data packet not encapsulated in a CBT header, the packet is discarded. However, this may change due to the interoperability issues discussed in the next section.

1.9 Some Interoperability Specifics

Multicast data packets *originating* on a host not running the CBT protocol code will be normal IP-style multicast packets. For interoperability reasons, multicast routers of other schemes should be able to forward multicasts that originate at CBT-capable hosts, and vice-versa. For this interoperability goal to be satisfied, we suggest that hosts satisfy the following requirements:

- multicast data packets originating at hosts that do not have full CBT multicast capability should be marked in some way so as to be recognizable by CBT routers. This could be achieved by having these hosts include in their packets an IP option. This option can be removed as soon as the packet is received by any multicast router, since it is only relevant on the originating subnet. The option will be ignored by receivers (hosts) on the same subnet. We consider this to be the *minimal requirement* of any host.

- hosts with CBT capability means that such hosts can originate multicasts with a CBT header (behind the IP header). The IP header of the packet should contain the corresponding multicast address. All CBT routers on the same subnet will receive the CBT-style multicast, but it is the responsibility of the DR for the group to *multicast* it back across the same subnet as a traditional IP-style multicast (provided there exists group member presence on that subnet). The IP-style multicast will be received and processed by all member hosts on that subnet, whether CBT-capable or not.

The latter item conforms with non-member sending hosts' requirements for them to be able to originate multicasts. Non-member sending was discussed in section 1.7.

1.10 Data Packet Encapsulation and Translation

As we have said, data packets are encapsulated[6] by CBT routers for forwarding along CBT tree branches. Where tree branches overlap a multi-access LAN, data packets are multicast with a CBT header from parent to children, and vice-versa, as well as being multicast IP-style for the benefit of group receivers on the multi-access subnet (this is only done if there exists at least one member of the group on the subnet).

There are two cases where CBT routers need to manipulate packet headers to achieve CBT multicast that is interoperable with traditional IP multicast schemes:

- when receiving a multicast packet that originated from a host on a directly-connected subnet.
- when receiving a data packet via a CBT tree branch.

1.10.1 Originating Case

For reasons of interoperability, CBT routers *promiscuously* receive multicast packets, similar to multicast routers of other schemes.

The receiving router checks for the presence of a CBT header. The local DR must also be able to distinguish if the packet originated on the LAN it was received from so as to know whether to send it back across the LAN as an IP-style multicast. For CBT style data packets originating on a LAN, re-multicasting them IP-style back over the subnet is the responsibility of the DR for the group.

A packet is distinguishable as having originated at a host on a subnet through the presence of a field in the CBT header that is set by originating hosts, or by the presence of an IP option if the packet originated at a non-CBT-capable host.

If a router has no FIB entry for the specified group, the packet is multicast IP-style back across the subnet by the DR, provided there is member presence on that subnet. If any receiving router has any other directly-connected subnets with group member presence, an IP-style multicast is sent over each. How a CBT router generates an IP-style multicast from a CBT data packet is described in the next section.

The CBT header is filled in by the originating host, as follows:

- ◇ the multicast group address (group-id) is inserted into the **group-id** field of the CBT header.
- ◇ the unicast address of a core router for the corresponding group is placed in the **core address** field.
- ◇ the **org** field of the CBT header is set to 0xFF. This indicates that the packet originated at this host.

[6] The encapsulation we refer to here is CBT header encapsulation, and takes place *behind* the IP header.

- ◇ the IP address of the originating host is inserted into the **origin** field of the CBT header.
- ◇ the **proto** field of the CBT header is set to identify the upper-layer (transport) protocol.
- ◇ the **ttl** field of the CBT header is set to IP TTL's maximum value.
- ◇ the **on-tree** field of the CBT header is left unset (it can only be set by a CBT router that is on-tree for the corresponding group).
- ◇ the destination field of the IP header is filled in with the multicast group address.
- ◇ the protocol field of the IP header is set to reflect the presence of a CBT header behind the IP header.
- ◇ the TTL value of the IP header is set to 1.

The packet is now *multicast* from the host onto its local subnet. Group member hosts will receive the multicast, but discard the packet because hosts never process multicasts containing a CBT header.

However, CBT routers promiscuously receive multicasts, CBT-style and IP-style. CBT routers on the same subnet as the originating host will receive the packet. CBT routers check the *protocol field* of the IP header on all multicasts received, so as to know whether they are handling IP-style or CBT-style multicasts (how CBT routers handle IP-style multicasts will be discussed in Part C). If a receiving router has a FIB entry for the group specified in the destination field of the packet's IP header, and the packet is a CBT-style multicast, the packet is forwarded over each outgoing tree branch for the group. Irrespective of the presence of a FIB entry for the group, the packet is also multicast IP-style (after removing the CBT header and copying various fields to the IP header, as described in the next section) onto any outgoing subnets with group member presence.

The DR for the group processes the received packet similarly, except it additionally multicasts the packet IP-style back across the originating subnet, provided there is group member presence on the subnet. Only the DR can do this, for obvious reasons. Furthermore, only a group's DR can forward (unicast) a packet that originated on a directly-connected subnet despite not having a FIB entry for the corresponding group. This is so that non-members can send to the group.

The process is illustrated in figure 2.

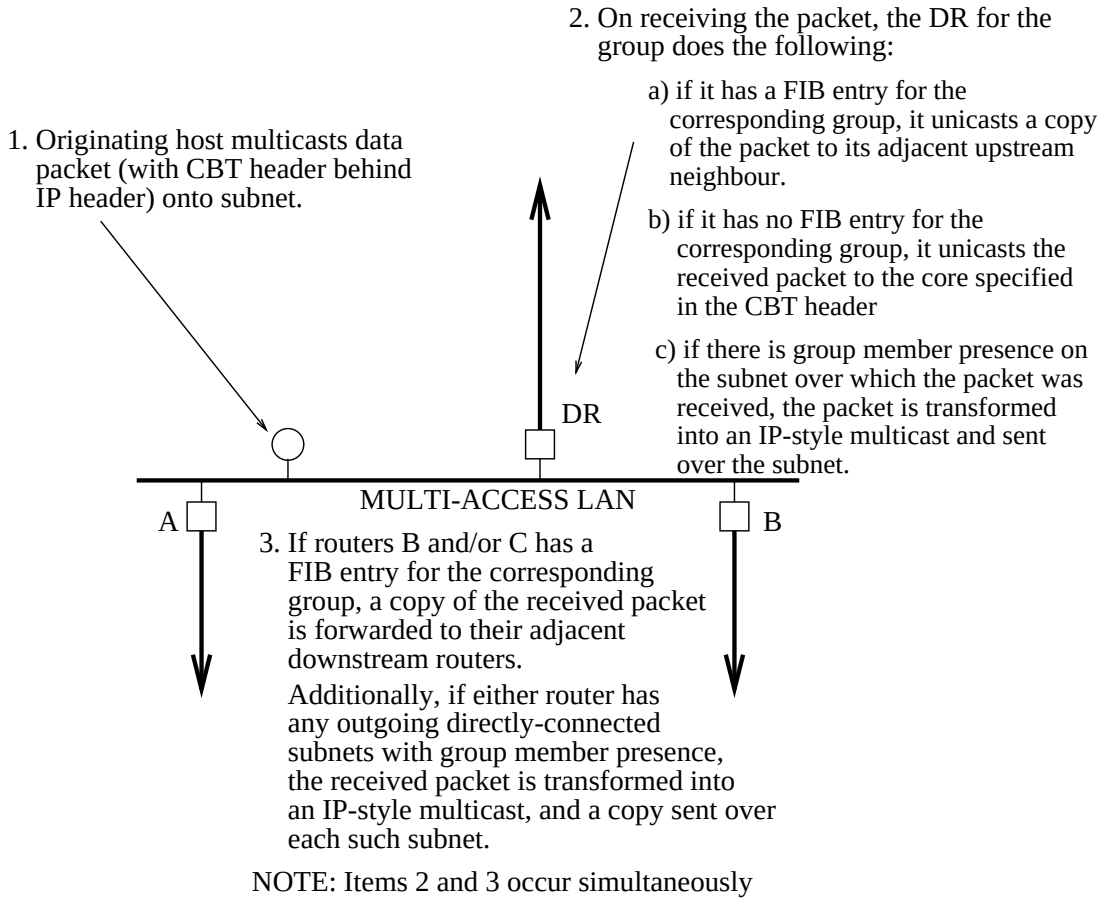

Figure 2: CBT Data Packet Forwarding on a LAN (originating case)

The next section looks in more detail at how a CBT router handles an incoming packet, and how it generates an IP-style multicast.

1.10.2 Receiving Case

CBT routers de-multiplex incoming multicast-addressed packets based on the contents of the protocol field of the IP header. A CBT multicast data packet, arriving at a CBT router, is handled as follows:

- ◇ the FIB entry is checked for the presence of the corresponding group entry. If no FIB entry exists for the group, AND the packet did not originate at a directly-connected host, the packet is discarded.

  If there is no FIB entry AND the packet did originate at a directly-connected host, it is multicast back across the subnet as an IP-style multicast by the group's DR, provided there is group member presence on that subnet. Furthermore, if the packet's **core**

address field is not NULL, the core address is copied to the IP header's destination address field, the **org** field is unset (see below), and the packet is forwarded (unicast) by the DR.

◇ if the **org** field is set (0xFF indicating the packet originated at a directly-connected host), it is unset (0x00).

◇ if the packet arrived via an interface that is not on-tree for the group, the **on-tree** field is set in the CBT header.

◇ for directly-connected subnets that have group member presence, de-capsulation and translation of some values from the CBT header to the IP header needs to take place, so that an IP-style multicast can be sent over those subnets. More precisely, the **group-id** of the CBT header is copied to the destination field of the IP header, the **origin** field is copied to the source address field, the **proto** field is copied to the protocol field, and finally the **ttl** field is copied after being decremented.

◇ on completion of the above, the multicast data packet is ready for sending across each subnet with member presence. This packet is compatible with all existing IP multicast schemes.

The process is illustrated in figure 3.

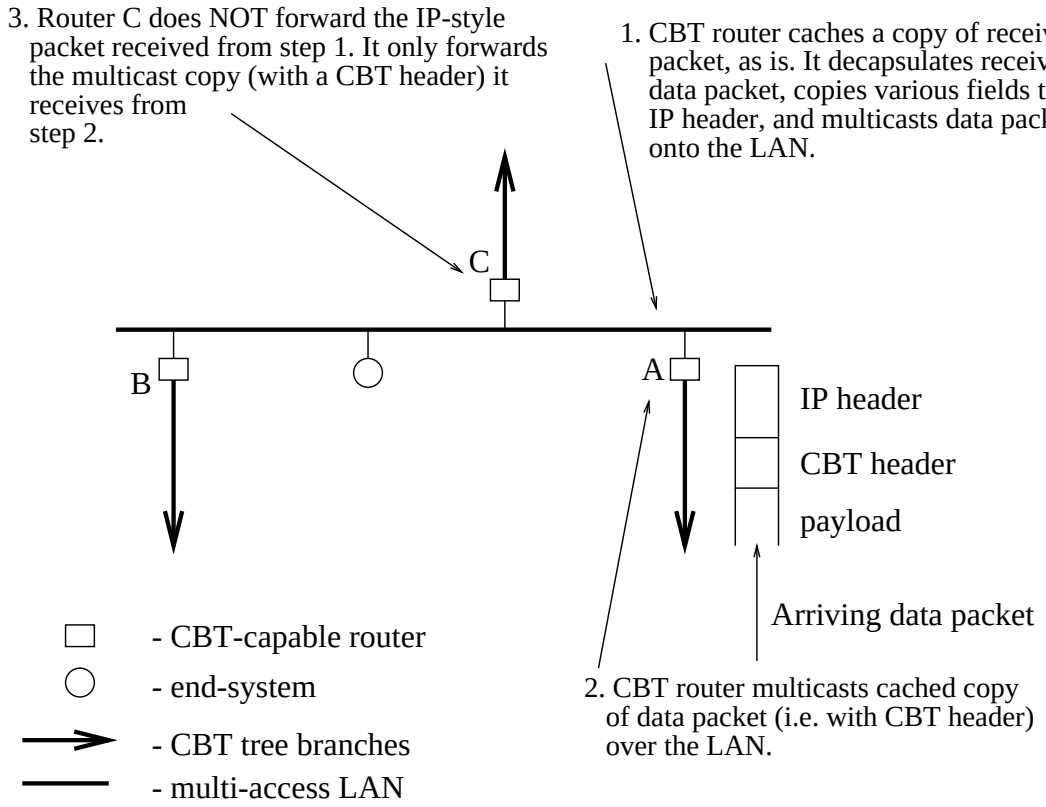

Figure 3: CBT Data Packet Forwarding on a LAN (receiving case)

1.11 Lower Group Leave Latency

One of the design goals of CBT was to modify the *Internet Group Management Protocol (IGMP)* to reduce group leave latency, i.e. the time between the last claim to a group on a particular subnet being relinquished, and the time group traffic is no longer forwarded onto that subnet. Using DVMRP as an example, this takes around four and a half minutes. The reason leave latency is currently so long is because this is the shortest time considered reasonable for multicast routers to implicitly deduce, from the absence of group membership report messages, that there are no longer any claims to a particular group on a subnet.

CBT introduces an explicit *IGMP group leave* message to drastically reduce leave latency[7]. It was considered an important design goal since, over the last few years, multicast has been adopted as the preferred transport mechanism for many high-bandwidth applications, including multimedia applications. Now, even comparitively resource-rich LANs are not immune to the congestion problems usually only witnessed on slower, wide-area links. It is therefore essential that, once there are no longer any receivers on a subnet,

[7]This need not be restricted to CBT alone, but may be adopted by other schemes such as DVMRP.

the corresponding traffic flow should cease as soon as possible thereafter. RSVP [8] uses its filter mechanism to achieve a similar effect, simply by switching its filters on or off.

The new version of IGMP is interoperable, and backwards compatible with the older version. The interoperation of new IGMP operating in CBT routers with IGMP operating in non-CBT routers on the *same* subnet, is discussed in Part C.

As we have said, lower leave latency is possible through the introduction of a new IGMP message type: IGMP-HOST-MEMBERSHIP-LEAVE. When a host relinquishes its last claim to a particular group membership, it multicasts a IGMP-HOST-MEMBERSHIP-LEAVE message to the "all-CBT-routers" address. This messsage contains the multicast address of the group being relinquished. Irrespective of whether any of the receiving CBT routers is the subnet's membership interrogator (a non-CBT router on the same subnet may have been elected – see section C, 1.4), since the LEAVE message is only interpretable by CBT routers, they respond by sending an IGMP-HOST-MEMBERSHIP-QUERY to the "all-systems" multicast address. Only one CBT router actually responds with a query, since the responses are randomized over an interval of five seconds, and the receipt of a query cancels out a CBT router's pending query.

From the moment the LEAVE message arrives at a CBT router, a timer starts running for the group being relinquished, and is only cancelled if, subsequent to the query, a report arrives before the timeout period, which currently is around 12 seconds. This is comprised of the 10 seconds randomized response interval of hosts after hearing a query, plus a 2 second safety margin.

Leave latency *could* be further reduced if a host's randomized response interval were shortened to, say, 5 seconds. The trade-off then is between increased protocol overhead/bandwidth consumption of more frequent IGMP messages, and a shorter group leaving time.

Part B

1 CBT Packet Formats and Message Types

CBT packets travel in IP datagrams. For clarity, we distinguish between three types of CBT packet: those directly concerned with tree building, and re-configuration – so called *primary maintenance messages*; those concerned with general tree maintenance – so called *auxilliary maintenance messages*; those carrying multicast *data*.

All of the above message types are encapsulated in a *CBT header*. Primary and auxilliary maintenance messages are additionally encapsulated in a *CBT control header*. All packets then, data and control, carry the CBT header, but control packets only require the parsing of four of the fields of the CBT header. The reason a CBT header is present even partially in control packets is partly administrative – it requires the definition of just one protocol number. We propose this protocol number be 7. Control packets therefore, travel inside (a portion of) a CBT header, and are identifiable as such by the contents of

the TYPE field in the CBT header (TYPE can only be "control" or "data").

1.1 CBT Header Format

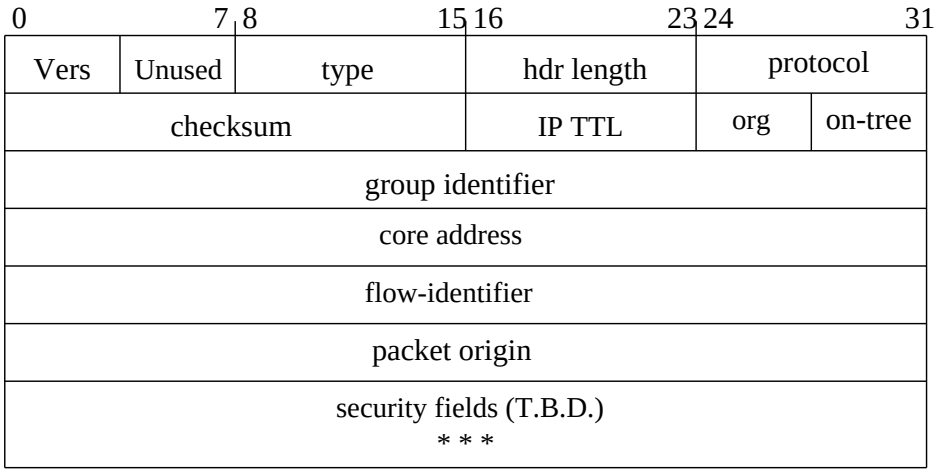

Figure 4: CBT Header

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.
- **org**: indicates whether the packet originated on a directly-connected subnet. It is unset by a receiving CBT router, since it has only local significance.
- **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.

1.2 Control Packet Header Format

The illustration below shows a CBT control packet header.

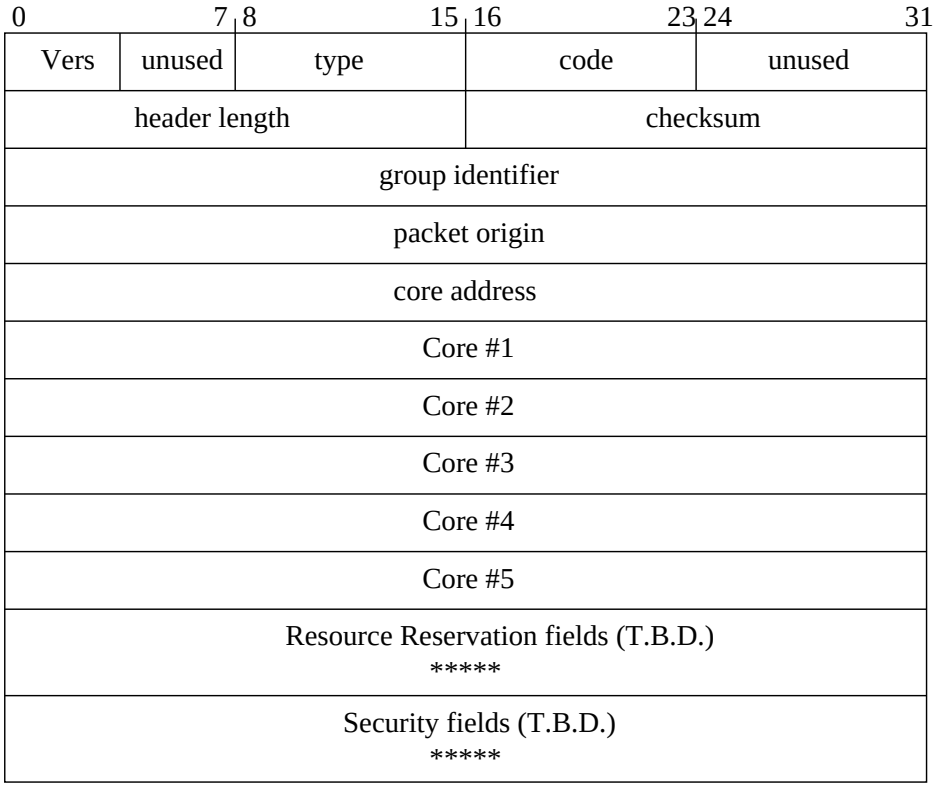

Figure 5: CBT Control Packet Header

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:

- **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.

1.3 Primary Maintenance Message Types

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

- **JOIN-REQUEST**: invoked by an end-system, generated and sent (unicast) by a CBT router to the specified core address. Its purpose is to establish the sending CBT router as part of the corresponding delivery tree.
- **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.
- **JOIN-NACK**: a negative acknowledgement, indicating that the tree join process has not been successful.

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

1.4 Auxilliary Maintenance Message Types

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-REPLY**: 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.
- **CBT-ECHO-REPLY**: positive reply to the above.
- **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.
- **CBT-PING-REPLY**: positive reply to the above.
- **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.
- **CBT-CORE-CHANGE**: 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.
- **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.

Part C

1 Interoperability Issues

It was a primary design goal that CBT interoperate with existing IP multicast schemes. We have already discussed in detail how CBT multicast data packets are backwards compatible with existing IP multicast schemes (see section A, 1.9). In this section we summarize this, and address other interoperability issues.

1.1 Isolation of CBT Routes

Multicast capability is not yet fully integrated into the internetwork infrastructure, although its use is becoming more and more widespread. As a result, each multicast scheme

must establish a topology map of the multicast-capable subnetworks and distances to those subnetworks, by means of a unicast-like *routing deamon*. CBT is no different.

The current specification states that CBT routes shall remain separate from those of other schemes, and therefore no route exchange takes place between CBT and other schemes. This may be revised in the future once interoperability of CBT with other schemes has been more fully investigated.

1.2 Backwards Compatibility of Data Packets

We explained in section A, 1.8. that data packets are multicast IP-style onto subnets with group member presence, by CBT routers. These multicast data packets are *no different* to those of existing IP multicast schemes, and therefore may be received and processed both by all end-systems and all multicast-capable routers (all schemes) on the same subnet.

Furthermore, we stated that multicast data packets originating from a non-CBT-capable host on a subnet should be *marked in some way* (we proposed using a CBT IP option) to indicate to receiving CBT routers on the same subnet that these may be forwarded – under normal circumstances CBT routers only forward data packets received which contain a CBT header behind the IP header. The use of an IP option should not have an effect on receiving routers of other schemes – they will merely not recognise the CBT IP option and forward as appropriate.

In Part A, section 1.10, we did not explain how IP-style multicasts originating at a non-CBT-capable host are handled by receiving CBT routers on the same LAN (subnet). As we have said above, such multicasts should be identifiable to CBT routers in some way, for example, by the presence of an IP option. Such an option only has relevance to local CBT router(s), and so the overhead of its processing is localized to multicast routers on the same subnet. CBT routers receiving IP-style multicasts that are not marked by an option, are discarded.

It is obviously desirable, for interoperability reasons, that CBT routers which receive IP-style multicasts *with* the IP option (i.e. those that originated on the local subnet at a non-CBT-capable host), forward them. They will *only* do so provided the receiving router has a FIB entry for the corresponding group. Otherwise, such packets are discarded.

Assuming the receiving router has a FIB entry for the group, a copy of the packet is sent to each outgoing directly-connected subnet that has group member presence (the IP option being deleted beforehand).

In order for a CBT router to forward the multicast along tree branches, it must append a CBT header, as well as modify various fields of the IP header before forwarding to any adjacent on-tree routers. More precisely, the following occurs:

- for each adjacent on-tree router, the multicast address in the destination address field of the IP header copied to the **group-id** field of the CBT header, and the IP destination address field is replaced with the unicast address of the corresponding

adjacent router[8]. If any adjacent router is reachable over a multi-access link, the multicast address in the IP destination field is *not* overwritten since the packet will be *multicast* over the link.

- the **on-tree** field of the CBT header is set to 0xFF.
- the packet's origin is copied from the source address field of the IP header to the **org** (origin) field of the CBT header. The IP source address is overwritten with the address of the sending CBT router.
- the TTL field of the IP header is copied to the **TTL** field of the CBT header.

The packet is now ready for forwarding to adjacent CBT on-tree routers.

1.3 Tree Overlap of CBT with Other Schemes

Where a CBT router co-exists with a multicast router of another scheme on the same subnet, a potential problem arises: if both routers are forwarding traffic for the *same* group, AND there is more than one such subnet (i.e. the delivery trees of the different schemes overlap in more than one place for the same group), then data packets will be unnecessarily duplicated, with serious consequences of packet looping and proliferation. This problem would not occur if a CBT router was not permitted to be active on a subnet containing multicast routers of other schemes. We consider this constraint too restrictive and inflexible to be given any further consideration.

The problem is illustrated in figure 6 below.

As of writing, no effective solution has been found to the problem of different schemes' overlapping delivery trees. This is an area of ongoing work.

[8]This is for the cases where the adjacent on-tree router is either one hop away, or is reachable through a tunnel. Hence, in CBT there is never a requirement to tunnel IP over IP.

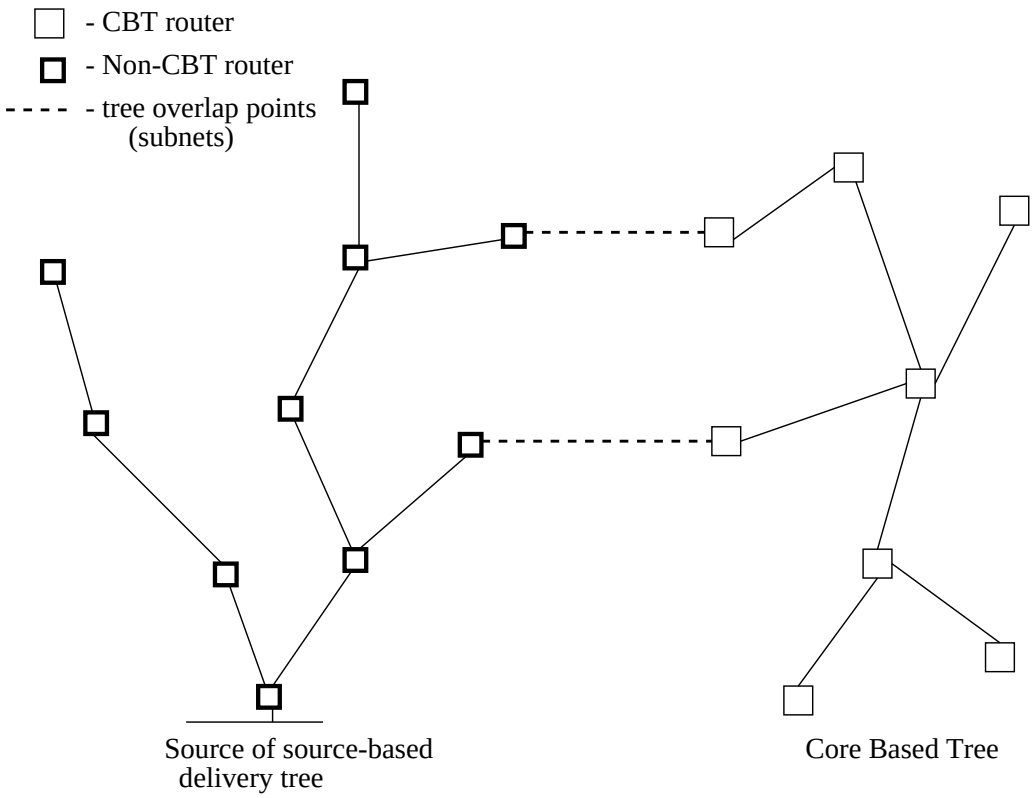

Figure 6: Overlapping delivery trees

1.4 IGMP in the Presence of Multiple Protocols

The *Internet Group Management Protocol (IGMP)* is a query-response protocol operating on multicast links (subnets) between hosts and multicast-capable routers.

Multicast routers on the same subnet running the same protocol, or different protocols AND are exchanging route information (for example, DVMRP and PIM [14]), regard themselves as *neighbours*. The lowest-addressed neighbour on a subnet is implicitly elected as that subnet's *membership interrogator*, and is responsible for sending host membership queries periodically to the "all-systems" multicast group.

If a neighbour is not heard from after some pre-specified time, a peer elects itself as membership interrogator, unless it knows of another lower-addressed neighbour, in which case it will elect itself (all multicast routers have a uniform view of their neighbours on a particular subnet).

As we have said, CBT does *not* peer with multicast routers of other schemes, and therefore can not partake in the election procedure in the same way. Despite this, we have devised a scheme that continues to elect the lowest-addressed router, including CBT routers. Without such a scheme, routers and hosts would suffer twice as much processing

overhead of IGMP messages, and twice as much bandwidth would be consumed, through the presence of two membership interrogators.

At start-up, a query is sent from each router of each scheme, so the election can proceed. CBT routers receiving queries from a particular (directly-attached) subnet know if each originated at a CBT peer (neighbour) or not, since, like other schemes, CBT routers keep a list of their neighbouring CBT routers. If a query arrives from a lower-addressed source that is *not* a CBT neighbour, the receiving router relinquishes its querier duties, and stores the address of the source in a table. There is one table entry for each attached subnetwork.

The question now is: what if the non-CBT membership interrogator 'goes away'?

Each table entry is stored on a timer. The timer expires 30 seconds after the next expected query, if that query fails to arrive. At this point, the current CBT router re-assumes querier duties. Each time a query arrives before the timeout interval, the corresponding entry time is reset.

If a query arrives from a different (non-CBT) source, but via the same subnet, the source address is replaced in the corresponding entry of the non-CBT querier table.

What we have just described is a mechanism that CBT has adopted in the presence of non-peering multicast protocols. It has assumed that the querier duty has fallen on a non-CBT router. The next question is: what if the querier duty falls on a CBT router? A mechanism is needed to prevent a non-CBT router from continuing to send IGMP queries. We therefore propose simple and minor modifications to the protocols of other schemes, so that querying is discontinued if a query is heard from a non-neighbour (non-peer).

What if the quierier duty falls on a CBT router and that querier subsequently 'goes away'? The answer is: the lowest-addressed CBT neighbour will assume querier duties, just as the routers on the subnet were exclusively CBT-capable.

However, there may not be a CBT neighbour on a particular subnet, for example, if the subnet is a *leaf*, but there may be a multicast router of another scheme present on the same leaf. So, a non-CBT router has relinquished querier duties to a CBT router, which has 'gone away'. How can the non-CBT router re-establish itself as querier for the subnet?

The answer to our question is, that other multicast schemes should adopt the same simple mechanism as CBT as part of their election procedure, i.e. non-CBT routers should store the non-neighbour querier address in a table, and re-assume querier duties if a query is not heard after some timeout period.

2 Resource Reservation

RSVP (Resource Reservation Protocol) [8] has been designed to be capable of reserving resources independent of whichever routing protocol is operating, unicast or multicast.

We believe that it should be possible to incorporate a *lightweight* version of the RSVP [8] resource reservation protocol into the CBT protocol. This could be achieved by incorporating the necessary fields into the CBT control header, and resources would be reserved as an integral part of tree set-up. Several of the CBT auxilliary control messages would

take on a *dual rôle*, serving their original purpose, and functioning simultaneously as RSVP messages.

Because of the explicit nature of CBT tree set-up and teardown, RSVP *path* messages no longer need to glean route change information. Also, the RSVP mechanisms to avoid message loops are obviated because of CBT's own mechanism of preventing loops. The danger of resource reservation duplication also disappears because the CBT join process explicitly prevents loops forming.

Although this is still considered 'future work', it is conceivable that lightweight RSVP could be incorporated into the CBT protocol without violating any of RSVP's design principles and goals [8].

The NIMROD proposal for IP next generation has emphasised that the establishment of multicast routes should be closely linked with reserving resources on those routes [13].

3 CBT Security Architecture

Soft state source based multicast schemes do not lend themselves well to security implementations. Transient state can result in security services, such as authentication, being unnecessarily repeated if tree state times out due to the lack of data packet flow.

The CBT architecture does not suffer from this problem – indeed, the explicit join mechanism, and the presence of tree focal points (i.e. cores) which can act as *authorization points* in a security sense, means that the CBT architecture provides a complementary platform for security implementations.

"Hard-state" protocol mechanisms are often thought of as being less fault tolerant than soft-state schemes. However, in the case of CBT we have built a high level of fault tolerance into the protocol, for example, by using multiple cores per tree to increase robustness, and by having various tree re-building/re-configuration mechanisms.

The security architecture we propose may well also provide a solution to the problem of *multicast key distribution* []. The essential problem here lies in the fact that a *key distribution centre (KDC)* must authenticate each of a group's receivers, as well as securely distribute a *session key* to each of them. This involves encrypting the relevant message n times before multicasting it to the group[9], where n may be very large. In short, existing multicast key distribution methods do not scale.

The security architecture we propose is independent of any particular cryptotechniques, although many security services, such as authentication, are easier if public-key cryptotechniques are employed. A more detailed discussion of why public-key cryptotechniques are more suited to multicast is given in [1]. They are also endorsed in [2].

We describe the CBT security architecture in the context of a *multicast key distribution* scheme, where the core(s) of the tree is a KDC. Equally, the scheme applies to a tree whose core is not a KDC, but a normal unicast router with CBT capability.

[9] Alternatively, the KDC could send an encrypted message to each of the receivers individually, but this does not scale either.

If a hierarchy of KDCs were present in the internetwork infrastructure, dedicated to multicast, as part of the actual group set up phase, receivers would first *securely* join the KDC group for the purposes of *scalable, secure, multicast key distribution*. Subsequently, the same receivers would join another group, whose purpose is the actual multicast communication itself. The joining of this group need not be secure. Communication between the group's receivers could remain confidential, or multicast data could include a corresponding authenticator, as a result of having joined the secure tree to obtain the necessary key(s)/parameters, etc.

Our scheme serves to *authenticate* tree nodes (routers) and receivers (end-systems) as part of the tree joining process.

The catalyst behind the scalability of our proposal is the distributed manner in which tree nodes are authenticated as part of the group joining process. This corresponds with the way in which JOIN-REQUESTs are processed and forwarded on a hop-by-hop basis, until they reach an on-tree router.

3.1 Security Example – CBT for Multicast Key Distribution

The CBT architecture assumes the presence of a security fields (T.B.D.) in the CBT header and CBT control header.

The example we provide below specifies the cores of the CBT tree as KDCs. As a direct result of this, the example given doubles as a proposed solution to the multicast key distribution problem, as stated above.

In the diagram below, host h is joining the multicast group G. The topologically "nearest" core is C. We assume that h's local DR, router A, has not yet joined the CBT tree for G.

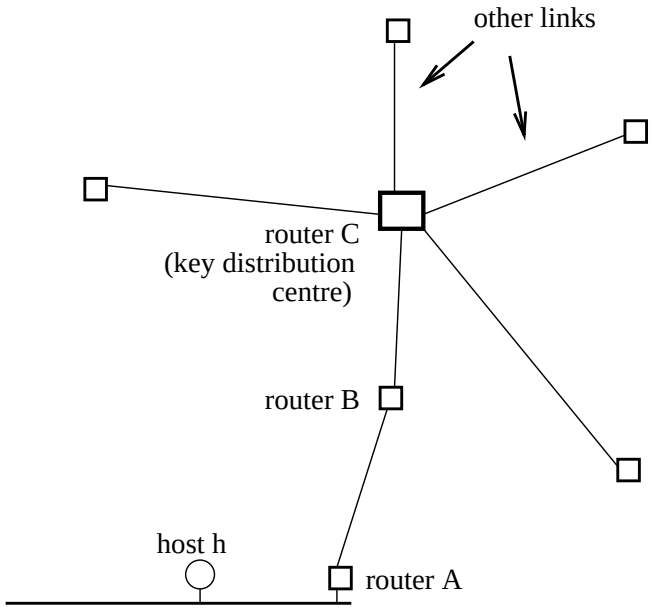

Figure 7: CBT Authenticated Join process (hosts and routers) and Key Distribution

A branch is created as part of the CBT secure tree joining process, as follows:

NOTE: All control messages described here carry a *Message Integrity Check (MIC)* to ensure the integrity of each message. On receipt of a message, a node always first verifies the integrity of the received message before processing further.

If a public-key cryptographic method is employed, each message can be *digitally signed* by the originator of the messege. This provides both message *integrity* and *origin authenticity*.

- As part of h's joining process, it must send a special CBT message to the local DR (router A), so that A invokes a JOIN-REQUEST. The source, h, is authenticated by A (we assume the message carries an authenticator).
- If h is authenticated, A generates a JOIN-REQUEST and sends it to the best next-hop on the path to the core (C). The best next-hop in our example is router B. The join carries A's authenticator, and that of h.
- If router A's join is authenticated by B, the above step is repeated, except the join is sent from B to C, the core, which happens to be the next-hop, and the authenticator is that of B. h's authenticator is copied to the new join.
- C authenticates B's join. As the tree's *primary authorization point*, C also authenticates the host that triggered the join process, i.e. h, provided h is included in the

KDC's access control list [1] for the group (remember that h's authenticator has been copied across consecutive joins). If h is not in the corresponding access control list, authentication is redundant, and a JOIN-NACK is returned from C to B.

Once B and h have been verified, C sends a copy of the group's *session* key encapsulated in a JOIN-ACK, together with C's authenticator. The session key is encrypted twice (separately), one copy being only decipherable by h, the other being only decipherable by B. The JOIN-ACK also contains the access control list for the group. The access control list may be encrypted by C so as to be only decipherable by B, if confidentiality of the list is required.

> NOTE: Such encryption may indeed be desirable, since CBT-capable routers are often connected by "tunnels", i.e. multicast delivery trees (of all schemes) consist of "islands" separated by non-multicast-capable routers.

- B authenticates the JOIN-ACK received from C, and stores the included access control list in an appropriate table. It copies the encrypted session key that is for itself, stores one copy in encrypted form, and decrypts the other copy for immediate re-encryption so as to be decipherable only by A. The encrypted session key for h is copied to the JOIN-ACK to be forwarded to A. Also included in this JOIN-ACK is the access control list (possibly encrypted by B so as to be only decipherable by A). The JOIN-ACK is forwarded to A.

- A authenticates the JOIN-ACK received from B. The encrypted session key that is for itself is stored as is. The included access control list is stored in an appropriate table. The session key encrypted for h included in the JOIN-ACK is forwarded directly to host h, which decrypts it for subsequent use.

If paths, or nodes fail, a new route to a core is gleaned as normal from the underlying unicast routing table, and the re-joining process occurs in the same secure fashion.

JOIN-REQUESTs terminated by an on-tree router that has already joined the tree securely, can be authenticated at that point, since the terminating router has all the necessary information, i.e. the access control list, and the group's session key, to carry out authentication, as well as distribute the session key.

Part D

1 Suggested Additional Properties of a Multicast Architecture

With the advent of multicasting in an Internet of ever increasing size and heterogeneity (with respect to routing and addressing) we feel that the list of desirable properties a

multicast routing algorithm should exhibit, should be extended to include[10]:

- *Scalability.* With the Internet growing at its current rate, and the global expanse of interest in multimedia applications, we can expect to see a large increase in the number of wide-area multicasts. Clearly, any routing algorithm/protocol that does not exhibit good scaling properties across the full range of applications will have both limited usefulness and a restricted lifetime in the Internet.

- *Routing algorithm independence.* It is highly desirable that a multicast routing algorithm be designed independent of any particular unicast routing algorithm. A loose-coupled dependency on underlying unicast routing is a necessity, but tight-coupling, whereby the multicast protocol depends on certain unicast algorithm-specific features, is undesirable for various reasons; for example, multicast deployment across heterogeneous routing domain boundaries would be much simplified if there was no such coupling between multicast and underlying unicast protocols and any dependence precludes the independent evolution of both the unicast and multicast algorithms.

A more comprehensive discussion on the above properties, and how they relate to existing IP multicast algorithms, is given in [4].

Needless to say, the CBT architecture exhibits the above properties.

2 CBT - The New Architecture

2.1 Architectural Overview

A core-based tree involves having a single node, in our case a router (with additional routers for robustness), known as the *core* of the tree, from which branches emmanate. These branches are made up of other routers, so-called *non-core* routers, which form a shortest path between a member-host's directly attached router, and the core. A router at the end of a branch shall be known as a *leaf* router on the tree. Unlike Wall's trees, the core need not be topologically centred[11] between the nodes on the tree, since multicasts vary in nature, and correspondingly, so should the form of a group's delivery tree. CBT is unique in that it allows the multicast tree to be built to reflect the nature of the application.

The CBT protocol is built on top of the CBT architecture, just described. This architecture allows for the enhancement of the *scalability* of the multicast algorithm, particularly for the case where there are many active senders in a particular group. The CBT architecture offers an improvement in scalability over existing techniques by a factor of the number of active sources (where a source is a subnetwork aggregate). Hence, a core-based architecture allows us to significantly improve the overall scaling factor of $S \times N$ we have in

[10] A discussion of existing multicast properties is given in [7].

[11] To find the topological centre of a dynamic network is NP-complete

the source-based tree architecture, to just *N*. This is the result of having just one multicast tree per group as opposed to one tree per (source, group) pair.

It is also interesting to note that routers between a non-member sender and the CBT delivery tree need no knowledge of the multicast tree/group whatsoever in order to forward CBT multicasts, since these are *unicast* towards the core. This *two-phase* routing approach is unique to the CBT architecture. One such application that can take advantage of this two-phase routing is resource discovery, whereby a resource, for example, a replicated database, is distributed in different locations throughout the Internet. The databases in the different locations make up a single multicast group, linked by a CBT tree. A client need only know the address of (one of) the core(s) for the group in order to send (unicast) a request to it. Such a request would not *span* the tree in this case, but would be answered by the first tree router encountered, making it quite likely that the request is answered by the "nearest" server.

The other main premise driving the CBT approach is that it is unicast routing protocol independent, meaning that the correct operation of the multicast protocol is not inextricably linked to the existence of a particular underlying unicast protocol, or, more precisely, specific features of the unicast protocol. DVMRP [5] is a multicast protocol that relies on such features.

The significant advantages to be gained from this property are: firstly, both the multicast and unicast protocols can evolve independently of each other; secondly, this feature promotes and enhances complex-free multicasting across multiple heterogeneous unicast routing domains.

Besides this, DMVRP and PIM both rely on so-called *source-destination* routing, i.e. the forwarding interfaces are chosen only if a packet arrives on the shortest-reverse path to the given source. This can lead to complications where asymmetric routes are concerned, as illustrated in figure 8.

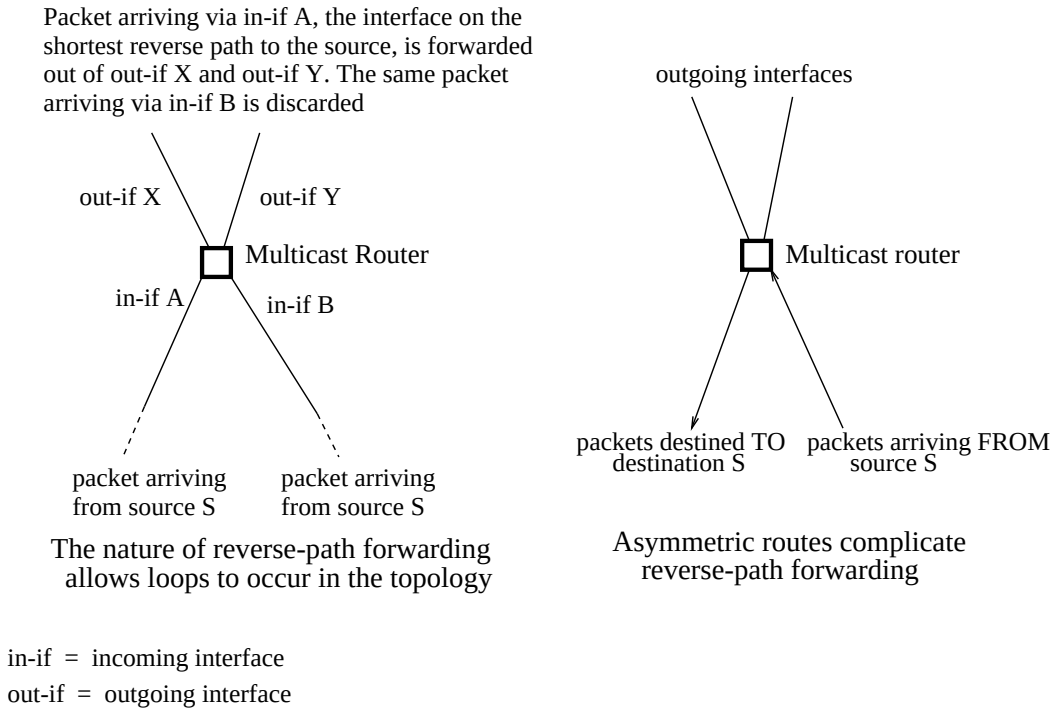

Figure 8: Reverse-Path Forwarding and Asymmetric Routes

It is unclear what criteria source-destination protocols, such as DVMRP and PIM, would use to forward packets in the presence of asymmetric routes, which may be imposed due to policy constraints. CBT forwarding interfaces are derived solely from a packet's arrival interface, irrespective of a packet's source address.

It is interesting to compare the characteristics of an existing multicast routing protocol such as DVMRP, the ST-II protocol [15] for reserving multicast data streams, and CBT.

DVMRP builds delivery trees which require no protocol overhead for tree maintenance (a direct result of the "soft-state" approach to tree building), and it maintains some topology state, i.e. each DVMRP router must know which of its interfaces is on the shortest-reverse path to a given source. Furthermore, traffic flow ensures that tree state does not time out.

ST-II builds virtual circuits as a way of reserving resources for particular data streams. Virtual circuits means that tree maintenance is explicit, and traffic flow and topology state are also maintained.

CBT is somewhere between DVMRP and ST-II. CBT does not build virtual circuits, does not maintain specific topology state, but tree maintenance is explicit.

A diagram showing a single-core CBT tree is shown in figure 9. Only one core is shown to demonstrate the principle.

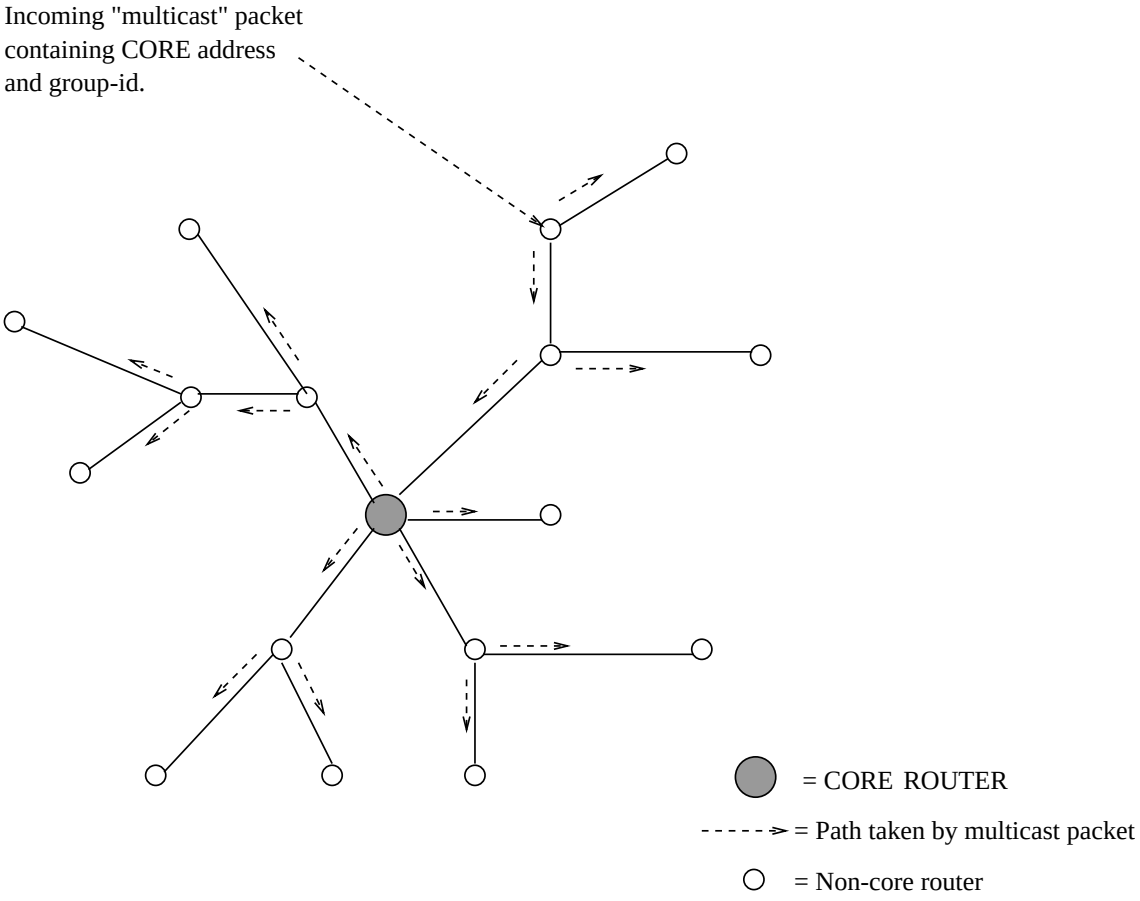

Figure 9: A single-core CBT tree

2.2 Architectural Justification

First of all, exactly *what* is a core-based tree (CBT) architecture? Core-based, or *centre-based* forwarding trees, were first described by Wall [17]. He used a single centre-based spanning tree to investigate low-delay broadcasting and selective-broadcasting. He noted: "we can't hope to minimize the delay for each broadcast if we use just one tree, but we may be able to do fairly well, and the simplicity of the scheme may well make up for the fact that it is no longer optimal". Wall proved that the maximum delay bound of an optimal core based tree is twice that of a shortest-path tree. Simulations have also been carried out [9] to compare the maximum delay of the two tree types, and Wall's prognosis was found to largely hold true.

This by no means makes core based tree multicasting redundant, since there are a whole host of factors that additionally need to be taken into account – one needs to look at the membership and traffic flow dynamics of each individual group. For example, a video

"broadcast" involving only one sender would enable the group's core(s) to be placed at or near the point of source, resulting in a shortest-path delivery tree. Slight discrepancies in delay may not be a critical factor for many multicast applications, such as resource discovery or database updating/querying. Even for real-time applications such as voice and video conferencing, a core based tree may indeed be acceptable, especially if the majority of branches of that tree span high-bandwidth links, such as optical fibre. In several years' time it is easy to envisage the Internet being host to thousands of active multicast groups, and similarly, the bandwidth capacity on many of the Internet links may well far exceed those of today.

3 Disadvantages of the CBT Architecture

The trade-offs introduced by the CBT architecture focus primarily between a reduction in the overall state the network must maintain (given that a group has a proportion of active senders), and the delay imposed by a shared delivery tree.

Whilst we have already pointed out in section 2.2, that the maximum delay bound of a centre-based tree has been proven to be twice that of a shortest-path tree, the delay factor may not always be acceptable, particularly if a portion of the delivery tree spans low bandwidth links. This is especially relevant for real-time applications, such as voice conferencing.

Another consequence of one shared delivery tree is that the cores for a particular group, especially large, widespread groups, can potentially become traffic "hot-spots" or "bottle-necks".

Core placement/management is another issue that may be seen as disadvantageous to the CBT approach. However, this can be turned aound into an advantage for some groups, for example, video "broadcasting", as discussed in section 2.2.

Finally, we have emphasized CBT's much improved scalability over existing schemes for the case where there are *active* group senders. However, because of CBT's "hard-state" approach to tree building, i.e. group tree link information does not time out after a period of inactivity, as is the case with most source-based architecutures. As a result, source-based architectures scale best when there are no senders to a multicast group, since multicast routers in the network eventually time out all information pertaining to an inactive group. Source-based trees are thus said to be built "on-demand".

Acknowledgements

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

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

Author's Address

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

e-mail: A.Ballardie@cs.ucl.ac.uk
Office Tel: ++44 (0)71 387 7050 ext. 3462

References

[1] J. Crowcroft A. Ballardie. Multicast Specific Security Threats and Counter Measures. *Work in progress*, 1994.

[2] R. Atkinson. SIPP Authentication Header. *Work in progress*, 1994.

[3] S. Casner and S. Deering. First IETF Internet Audiocast. *ConneXions*, 6, No.6:10–17, June 1992.

[4] A. Ballardie J. Crowcroft and P. Francis. Core based trees (CBT) - An Architecture for Scalable Inter-Domain Multicast Routing. In *Conference Proceedings of ACM Sigcomm'93*, pages 85–95. ACM SIGCOMM, September 1993.

[5] C. Partridge D. Waitzman and S. Deering. RFC 1075, Distance Vector Multicast Routing Protocol. *SRI Network Information Center*, November 1988.

[6] Martin de Prycker. *Asynchronous Transfer Mode*. Ellis Horwood Limited, Chichester, England, 1991.

[7] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. In *ACM Symposium on Communication Architectures and Protocols*, pages 55–64. ACM SIGCOMM, August 1988.

[8] L. Zhang R. Braden D. Estrin S. Herzog and S. Jamin. Resource Reservation Protocol (RSVP). *Work in progress*, 1994.

[9] D. Estrin L. Wei. The Trade-Offs of Multicast Trees and Algorithms. In *International Conference on Computer Communications and Networks*, September 1994.

[10] B. Kahle M. Schwartz, A. Emtage and B. Neuman. A Comparison of Internet Resource Discovery Approaches. *Computing Systems*, 5 (4):461–493, Fall 1992.

[11] John Moy. RFC 1584, multicast extensions to OSPF. *SRI Network Information Center*, March 1994.

[12] D. Piscitello. RFC 1209, The Transmission of IP Datagrams over the SMDS Service. *SRI Network Information Center*, March 1991.

[13] R. Ramanathan. Multicast Support for Nimrod: Requirements and Solution Approaches. *Work in progress*, 1994.

[14] D. Farinacci S. Deering, D. Estrin and V. Jacobson. Protocol Independent Multicasting. *Work in progress*, 1994.

[15] C. Topolcic. RFC 1190, Experimental Internet Stream Protocol v2. *SRI Network Information Center*, August 1991.

[16] J. Pasquale V. Kompella and G. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1, No.3:286–292, June 1993.

[17] David W. Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Stanford University, California, U.S.A., June, 1980.

[18] S. Wilbur and M. Handley. Multimedia Conferencing: from Prototype to National Pilot. In *INET'92, International Networking Conference*, pages 483–490, June 1992.