Core Based Tree (CBT) Multicast

- Architectural Overview and Specification -

by

Tony Ballardie

University College London

Status of this Draft

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 [5, 11].

The primary advantages of the CBT approach are that it typically offers more favourable scaling characteristics than do existing multicast algorithms. The definition of a new network layer multicast protocol has also meant that it has been possible to integrate an enriched functionality into multicast that is not possible under other IP multicast schemes, for example, the incorporation of security features. Besides this functionality providing the ability to authenticate tree-joining host's and routers, optional in-built protocol mechanisms provide a scalable solution to the multicast key distribution problem [2].

CBT is backwards compatible with traditional IP-style multicast. Host changes are not required, and a local CBT-capable router is mandatory if CBT-style multicasts are to be forwarded beyond the local subnetwork.

CBT Expires October 20th, 1995.

Contents

1 Background

Centre based forwarding was first described in the early 1980s by Wall [14] 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 [4], 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.

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 [4], replicated database updating and querying, software update distribution, stock market information services, and more recently, resource discovery [9]. Multimedia [15, 13] 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. We measure scalability in terms of network state maintenance, bandwidth- and processing costs.

3 Document Layout

The remainder of this document is divided into four parts: Part A offers a general architectural overview and discussion on the CBT architecture. This section also includes a description of CBT "anycasting" [10].

Parts B and C comprise the protocol specification. Part B describes 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].

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

Part D looks briefly at some other related issues.

Part A

1 CBT - The New Architecture

1.1 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 forward 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.

The CBT protocol builds a delivery tree reflecting the CBT architecture, just described. This architecture allows for the enhancement of the *scalability* of the multicast algorithm, with regards to group-specific state maintained in the network, 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 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. Effectively, this corresponds to an *anycast* service [10] (see section 3).

A diagram showing a single-core CBT tree is shown in figure 1. Only one core is shown to demonstrate the principle.

[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.

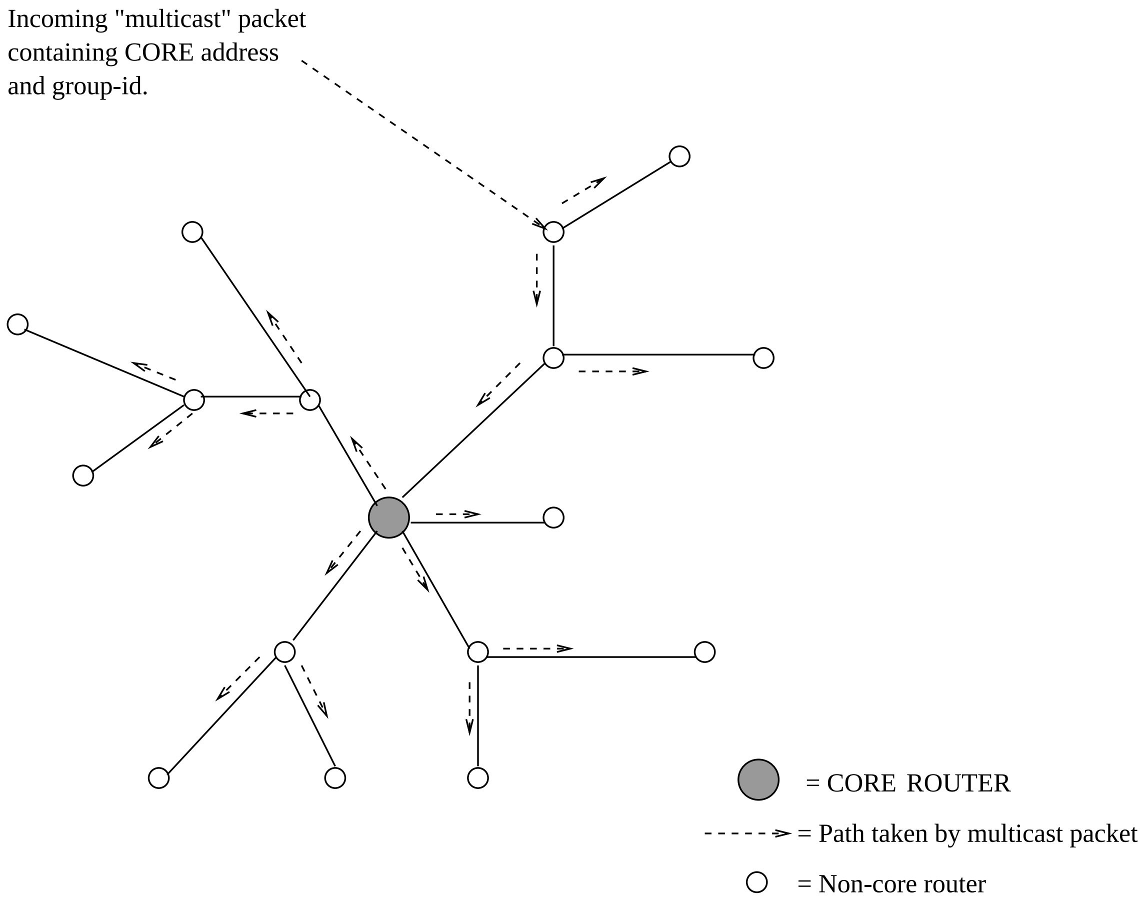

Figure 1: A single-core CBT tree

1.2 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 [14] in his investigation into low-delay approaches to broadcast and selective broadcast. Wall concluded that delay will not be minimal, as with shortest-path trees, but the delay can be kept within bounds that may be acceptable. Simulations have recently been carried out to compare the maximum and average delays of centre-based and shortest-path trees. These results are summarized in [3].

In the context of multicast, the extent to which the delay characteristics of a shared tree are less optimal than SPTs, is questionable. The simulation results described in section [3] state that CBTs incur, on average, a 10% increase in delay over SPTs. 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.

An important question raised in the SPT vs. CBT debate is: how effectively can *load sharing* be achieved by the different schemes? It would seem that SPT schemes cannot achieve load balancing because of the nature of their forwarding: nodes on a SPT do not have the option to forward incoming packets over different links (i.e. load balance) because of the danger of loops forming in the multicast tree topology.

With shared tree schemes however, each receiver can choose which of the small selection of cores it wishes to join. Cores and on-tree nodes can be configured to accept only a certain number of joins, forcing a receiver to join via a different path. This flexibility gives shared tree schemes the ability to achieve load balancing.

In general, spread over all groups, CBT has the ability to randomize the group set over different trees (spanning different links around the centre of the network), something that would not seem possible under SPT schemes.

Finally, the CBT protocol requires each receiver to explicitly join the delivery tree, resulting in a tree spanning only a group's receivers. As a result, data flows only over those links that lead to receivers, and thus there is no requirement for off-tree routers to maintain prune state, which prevents data flow where it is not needed.

2 The Implications of Shared Trees

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 potential increased delay imposed by a shared delivery tree.

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, source-based architectures scale best when there are no senders to a multicast group. This is because multicast routers in the network eventually time out all information pertaining to an inactive group. Source-based trees are said to be built "on-demand", and are "data-driven".

A consequence of the "hard-state" approach is that multicast tree branches do not automatically adapt to underlying multicast route changes[2]. This is in contrast to the "soft-state", data-driven approach – data always follows the path as specified in the routing table. Provided reachability is not lost, it is advantageous, from the perspective of uninterrupted packet flow, that a multicast route is kept constant, but the two disadvantages are: a route

[2]If multicast were part of the global internetwork infrastructure, multicast routes are gleaned exclusively from *unicast* routes.

may not be optimal for its entire duration, and, "hard-state" requires the incorporation of *control messages* that monitor reachability between adjacent routers on the multicast tree. This control message overhead can be quite considerable unless some form of message aggregation is employed.

In terms of the effectiveness of the CBT approach to multicasting, the increased delay factor imposed by a shared delivery tree 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 with numerous active senders, can potentially become traffic "hot-spots" or "bottlenecks". This has been referred to as the *traffic concentration* effect in [8].

The branches of a CBT tree are made up of a collection of branches rooted at the tree node that originated a join-request, and terminating at the tree node that acknowledged the same join. This has implications where asymmetric routes are concerned (similar to source-based schemes based on RPF) – whilst the same CBT branch is used for data packet flow in *both* directions, the child-to-parent direction constitutes a valid route reflecting the underlying unicast route (at least at the time the branch was created). However, in the parent-to-child direction, the path does not necessarily reflect underlying unicast routing at any instant, and therefore, in a policy-oriented environment, this *might* have disadvantageous side-effects.

Finally, there are questions concerning the *cores* of a group tree: how are they selected, where are they placed, how are they managed, and how do new group members get to know about them? We have attempted to describe some very simple heuristics to address some of these questions in [1], but these may not be appropriate for large-scale implementation of CBT. Work is currently underway in the development of a core placement/location protocol.

3 CBT and "Anycasting"

3.1 Overview of "Anycasting"

Anycasting [10] is a proposed best-effort, stateless, datagram delivery service which is used by hosts primarily to locate particular services on an internetwork. The goal of anycast is for a client to transmit *one* request to a resource "anycast address", and for a single, preferably nearest, server to receive the request and respond to it.

The motivation for anycasting is that it simplifies the task of finding the appropriate server in a network, and obviates the need to configure applications with particular server address(es), for example, as in DNS resolvers.

Questions that, as yet, remain unanswered regarding anycasting, include: how best can anycasting be achieved, and should anycast addresses be a special class of IP address?

As for how best to achieve anycast, there are two possible approaches: use existing IP multicast, or, answering our second question, define a special class of IP *anycast* address within the IP address space, and have servers additionally bind an anycast address on which they listen for client requests.

Using existing IP multicast has problems associated with it. Firstly, using *expanding ring search* [7] to locate a network resource is inefficient for two reasons: it requires potentially many re-transmissions of the request from the client, each iteration requiring a larger TTL[3] value . This continues until a response is received.

The other problem with using IP multicast is that, for any multicast transmission, potentially more than one response may be received. To summarize, using existing IP multicast for anycast is inefficient in its use of network resources, and does not necessarily achieve the desired goal of anycast, namely that only one server respond to a client request. Also, anycasting should *not* require managing the IP TTL value of client request packets – the goal of anycast is to send a single packet, which follows a *single* path, in order to locate a single, preferably nearest, server.

Defining a special class of "anycast" addresses has several problems associated with it. For example, routing must be adapted to support yet another class of IP address, and routing tables would be required to support anycast routes. Furthermore, segmenting the IP address space yet further not only involves significant administrative burden, but also assumes that existing applications will recognise particular addresses as being anycast [10].

3.2 The CBT "Anycast" Solution

It so happens that the CBT multicast architecture provides an effective solution to the anycasting problem, without requiring the definition of special anycast addresses.

The CBT architecture was explained in section 3.4. CBT is especially attractive for *resource discovery* applications, where it is assumed that different network resources for distinct CBT groups. The reason CBT is particularly suited to resource discovery, as described, is because it typically involves many *senders*, whereby a sender is *not* a group member. As we have already explained, CBT multicast, unlike other IP multicast schemes, involves maintaining group-specific state in the network that is *independent* of the number of *active* sources. Moreover, this state is constrained to the tree links that span only a group's receivers.

In CBT multicast, non-member senders actually utilize *unicast* to route multicast data to the CBT delivery tree. This is known as CBT's *2-phase routing*. These packets are unicast addressed to a single core router (of which there may be several), and will first encounter the delivery tree either at the addressed core, or at an *on-tree* (non-core) router that is on the unicast path between the sender and the addressed core.

For typical multicast applications, the receiving on-tree router disseminates the received packet(s) to adjacent outgoing on-tree neighbours, and neighbours proceed similarly on

[3]This is a field of the IP header which is decremented each time the corresponding packet traverses a router. If the TTL field reaches zero, a router will discard the packet.

receipt of a packet. This is how multicast data packets span a CBT tree.

For *anycast* (and resource discovery applications) however, the first on-tree node encountered does not disseminate the packet further, but responds to the received request.

Thus, we believe that CBT offers an effective solution to “anycasting” and resource discovery in general. However, some questions remain: what level of fault tolerance does the CBT solution offer, by what means does a sender establish the unicast address of a CBT core router, and finally, is there a guarantee that a client request will hit the CBT tree, i.e. reach a server, at the nearest point to the sender?

The question of fault tolerance is indirectly related to the question of establishing a core address. A CBT tree should never comprise only one core router for reasons of robustness. We envisage there should be at least two cores for local groups, and possibly up to five for wide-area groups. By whatever means a client establishes the identity of a core, it will always simultaneously establish the identities of *all* cores for a particular tree.

So, how could core addresses be found out about? One obvious solution would be to advertise core addresses, together with their associated network resource, in an application such as, or similar to, “sd”.

With regards to our final question, the choice of core will determine if a packet reaches a nearest server. Since users cannot be expected to know about network topology, it is assumed that the choice of core will be fairly random. Hence, our scheme makes no guarantees that a client request will reach the nearest server.

Part B

1 Protocol Overview

1.1 CBT Group Initiation

Like any of the other multicast schemes, one user, the *group initiator*, or *group leader*, 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*. The CBT User Group Management Interface is shown below.

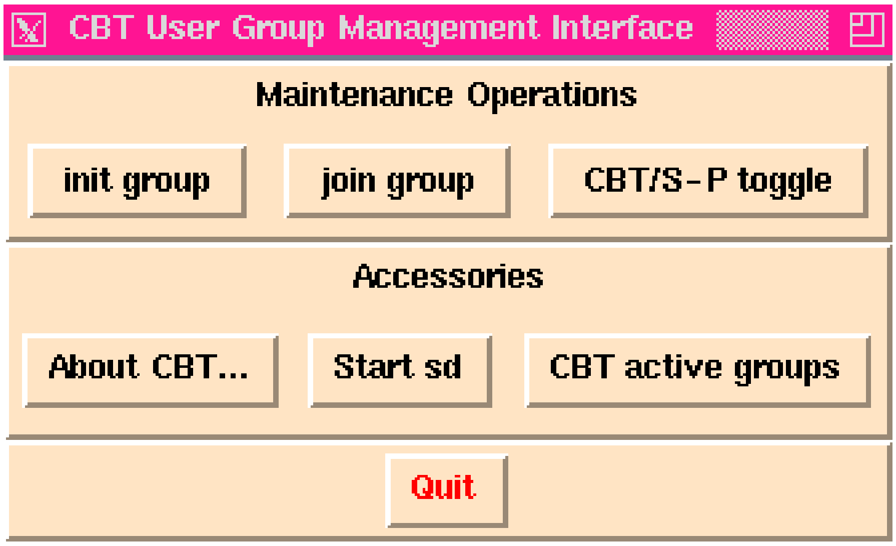

Figure 2: The CBT User Group Management Interface

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. The JOIN-ACK follows the *reverse* of the path traced out by the corresponding JOIN-REQUEST. It is the JOIN-ACK that actually creates a tree branch. The tree branch is a *reverse-shortest path* rooted at the node where the corresponding JOIN-REQUEST originated.

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[4]) or from the Domain Name System (DNS)[5].

In accordance with existing IP multicast schemes, if the scope of multicasts is to extend beyond the local area, at least one CBT-capable router must be present on the local subnetwork for hosts on that subnetwork to utilize CBT multicast delivery. 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 the *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.

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

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

[5]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

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 C, 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 [14].

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[6], and the addresses of these cores would be "well-known".

Work is currently in progress to develop a core location/placement mechanism.

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.

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.

[6] 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.

1.7 Non-Member Sending

For non-member senders For non-member senders wishing to send multicasts beyond the scope of the local subnetwork, 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 a host unicasting the packet from the originating host to one of the group's cores (the destination field of the IP header carries the unicast address of the core). The second phase is the disemmination of the the packet by the receiving router to neighbouring (adjacent) routers on the corresponding tree. Similarly, when an on-tree neighbour receives the packet, it distributes it in the same fashion.

Before the multicast leaves the originating subnetwork, it is necessary for the local CBT DR to append a CBT header to the packet (behind the IP header), and change the IP destination address field from a multicast address to the unicast address of a core for the group. How does the CBT DR know that this multicast address is associated with a CBT group? The answer is that there must be some form of mapping mechanism, which has information about which group address correspond to CBT multicast groups. This mechanism maps an IP multicast address to a unicast core address.

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.

> NOTE: No host changes are required for CBT. CBT hosts are simply required to run the CBT application-level software that provides the CBT user group management interface.

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.

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 for CBT to be backwards compatible with IP-style multicasts. Until the interface with other multicast protocols is clearly defined, CBT routing information is not exchanged with that of any other schemes.

IP-style multicast data packets arriving at a CBT router are checked to see if they originated locally. If not, they are discarded. Otherwise, the local CBT DR for the group first sends a copy of the IP-style packet over any directly-connected subnetworks with group member presence (provided the TTL allows), then appends a CBT header to the packet for forwarding over outgoing tree interfaces.

CBT-style packets arriving at a CBT router are forwarded over tree interfaces for the group, and sent IP-style over any directly-connected subnetworks with group member presence. The conversion from a CBT-style packet to an IP-style packet requires the copying of various fields of the CBT header to the IP header.

The child(ren) or parent of a CBT router may be reachable over a multi-access LAN. This is the case where a subnetwork and a tree branch are one and the same. In this case, the forwarding of the CBT-style packets is achieved with multicast as opposed to unicast. 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 also reduces traffic – when a parent receives a 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 IP-style multicsat data packets unless that router has a forwarding information base (FIB) entry for the specified group, The exception to this is if a multicast originates on a local subnetwork. In this case, the local CBT DR for the group needs to insert a CBT header in the packet (behind the IP hdr) and unicast it to one of the cores for the group.

A CBT FIB entry is shown below:

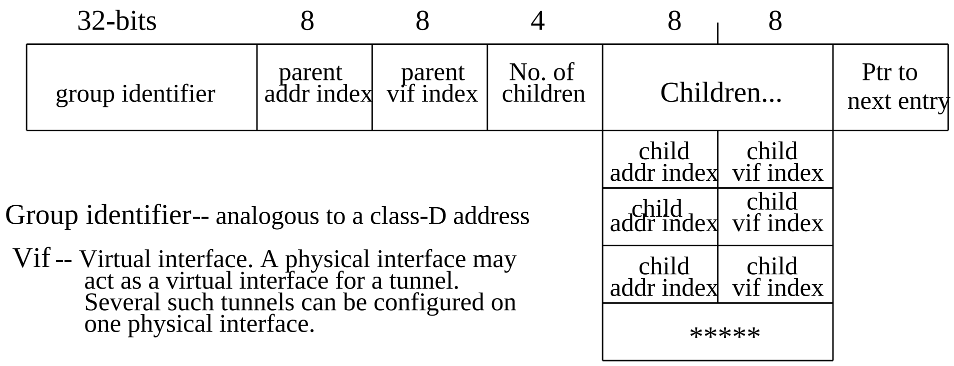

Figure 3: A CBT FIB entry in the author's implementation

The CBT DR for the specified group fills in the CBT and IP headers as follows (the CBT header is shown over):

- 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 of the CBT hdr.
- 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 either decremented (if CBT-style packet was received) or it is set to the value reflected in the packet's IP hdr (if the pkt originated locally).
- the **on-tree** field of the CBT header is set (provided this CBT router is on-tree for the specified group). It is left unset otherwise.
- the source address field of the IP header is set to the unicast address of the originating host (the IP src addr changes as the CBT-style packet is passed router-to-router on a CBT tree).
- the destination field of the IP header is set to the unicast address of the on-tree neighbour (set to group address if more than one neighbour is reachable over the same interface).
- the protocol field of the IP header is set to the CBT protocol value.

- the TTL value of the IP header is set to MAX_TTL.

  The packet is now ready for sending. Once this packet arrives at a CBT router, the packet is "reverse-engineered" (using the information carried in the CBT hdr) to produce an IP-style multicast for sending on directly-connected subnets with group presence.

Part C

1 CBT Packet Formats and Message Types

CBT packets travel in IP datagrams. We distinguish between two types of CBT packet: CBT data packets, and CBT control packets.

CBT data packets carry a CBT header when these packets are traversing CBT tree branches. The CBT header is positioned immediately behind the IP header.

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

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

1.1 CBT Header Format

The CBT header is illustrated below:

We proceed to describe each of the fields of the CBT header:

- **Vers**: Version number – this release specifies version 1.
- **type**: indicates whether the payload is data or control information.
- **hdr length**: length of the header, for purpose of checksum calculation.
- **protocol**: upper-layer protocol number.
- **checksum**: the 16-bit one's complement of the one's complement of the CBT header, calculated across all fields.
- **IP TTL**: TTL value gleaned from the IP header where the packet originated. It is decremented each time it traverses a CBT router.
- **on-tree**: indicates whether the packet is on- or off-tree. Once this field is set (i.e. on-tree), it is non-changing.

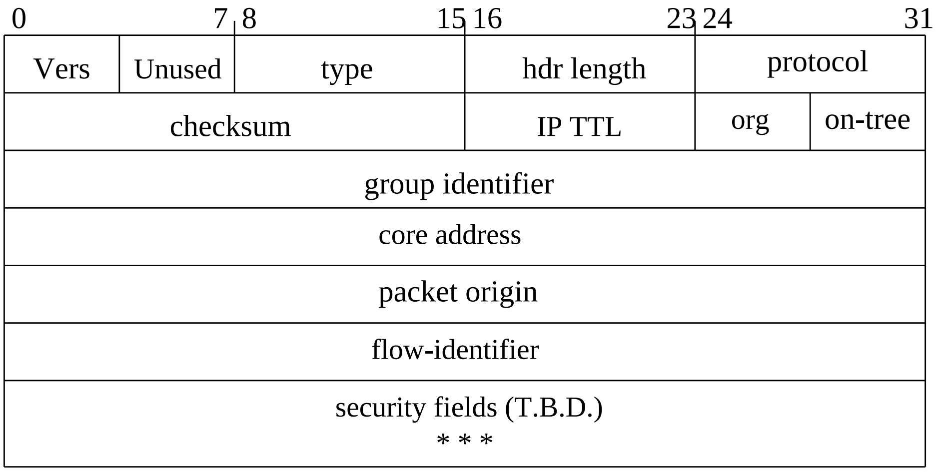

Figure 4: CBT Header

- **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.

<table>
<tr><td>0</td><td>7</td><td>8 15</td><td>16 23</td><td>24 31</td></tr>
<tr><td>Vers</td><td>unused</td><td>type</td><td>code</td><td>unused</td></tr>
<tr><td colspan="3">header length</td><td colspan="2">checksum</td></tr>
<tr><td colspan="5">group identifier</td></tr>
<tr><td colspan="5">packet origin</td></tr>
<tr><td colspan="5">core address</td></tr>
<tr><td colspan="5">Core #1</td></tr>
<tr><td colspan="5">Core #2</td></tr>
<tr><td colspan="5">Core #3</td></tr>
<tr><td colspan="5">Core #4</td></tr>
<tr><td colspan="5">Core #5</td></tr>
<tr><td colspan="5">Resource Reservation fields (T.B.D.)
*****</td></tr>
<tr><td colspan="5">Security fields (T.B.D.)
*****</td></tr>
</table>

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.
- **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 D

1 Related Issues

1.1 Interoperability Issues

One of the design goals of CBT is for it to fully interwork with other IP multicast schemes. We have already described how CBT-style packets are transformed into IP-style multicasts, and vice-versa.

In order for CBT to fully interwork with other schemes, it is necessary to define the interface(s) between a "CBT cloud" and the cloud of another scheme. The CBT authors are currently working out the details of the "CBT-other" interface, and therefore we omit further discussion of this topic at the present time.

1.2 A Router Optimization

In a CBT-only environment it is possible to optimize the performance of CBT with respect to data packet forwarding in CBT-capable routers. In such an environment the presence of a CBT header is not necessary, and its absence is likely to improve switching times by around 50 per cent. However, the downside is that the functionality the CBT header provides, such as CBT security, is lost.

1.3 CBT Security Architecture

See current I-D [2].

Acknowledgements

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

Steve Ostrowitz (Bay Networks Inc.) for his suggestions and comments on making a CBT router implemention as optimal as possible.

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

Author's Address

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

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

References

[1] A. J. Ballardie. *A New Approach to Multicast Communication in a Datagram Internetwork*. PhD thesis, University College London, University of London, 1994 (to appear).

[2] A. J. Ballardie. Scalable Multicast Key Distribution. *Working draft*, April 1995.

[3] J. Bibb Cain. CBT vs. PIM Comparison. *Presentation at IDMR wg, IETF, April 1995.*

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

[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 a Datagram Internetwork.* PhD thesis, Stanford University, California, U.S.A., 1991.

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

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

[10] C. Partridge T. Mendez and W. Milliken. RFC 1546, Host Anycasting Service. *SRI Network Information Center*, November 1993.

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

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

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