Core Based Trees (CBT)

An Architecture for Scalable Inter-Domain Multicast Routing

Tony Ballardie*(University College London)
e-mail: A.Ballardie@cs.ucl.ac.uk

Paul Francis†(Bellcore, N.J., U.S.A.)
e-mail: francis@thumper.bellcore.com

Jon Crowcroft (University College London)
e-mail: J.Crowcroft@cs.ucl.ac.uk

Abstract

One of the central problems in one-to-many wide-area communications is forming the delivery tree - the collection of nodes and links that a multicast packet traverses. Significant problems remain to be solved in the area of multicast tree formation, the problem of scaling being paramount among these.

In this paper we show how the current IP multicast architecture scales poorly (by scale poorly, we mean consume too much memory, bandwidth, or too many processing resources), and subsequently present a multicast protocol based on a new scalable architecture that is low-cost, relatively simple, and efficient. We also show how this architecture is decoupled from (though dependent on) unicast routing, and is therefore easy to install in an internet that comprises multiple heterogeneous unicast routing algorithms.

1 Introduction

Multicast group communcation 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 [7] specify multicast as part of their service, but perhaps the most apparent and widespread growth in multicast applications is being experienced in the IP Internet. We can see evidence of this growth in the MBONE, the set of routers and networks with multicast capapility.

In order to cater to a very large number of internetwork-wide multicast applications, examples of which include audio and video conferencing [15], replicated database updating and querying, software update distribution, stock market information services, and more recently, resource discovery [11], it is important that the multicast routing protocol used be first and foremost scalable with respect to a network of very large size, and low-cost in terms of computational overhead and storage requirements - properties lacking in current IP multicasting techniques. The protocol should also be designed to operate "invisibly" across domain boundaries, i.e. independent of the underlying unicast routing algorithm, so that it can evolve independently.

This paper describes a new multicast routing architecture which is applicable to any datagram network whose switches have multicast forwarding capability. We will present a multicast routing protocol (CBT) for IP networks based on this new architecture that not only satisfies the above criteria, but is also relatively simple in design.

In the following section we discuss the existing multicast architecture. Section 3 describes the current IP multicast environment and goes on to briefly describe two IP multicast routing protocols. Section 4 presents a comprehensive critique of the existing architecture showing how it is inherently non-scalable and bound to particular underlying unicast routing algorithms. This leads us to the new architecture in section 5 followed by a description of a protocol built on this new architecture in section 6. Sections 7 and 8 offer some thoughts on future work and an overall summary, respectively.

2 Existing Multicast Architecture

The existing multicast architecture is not restricted to IP networks, but is being accepted as the solution to multicasting in many different kinds of networks and environments.

For each multicast group, the current architecture builds a shortest-path source-based delivery tree be-

*Principal author
†Previously published under the name Paul Tsuchiya

SIGCOMM'93 - Ithaca, N.Y., USA /9/93
© 1993 ACM 0-89791-619-0/93/0009/0085...$1.50

tween each sender and the corresponding multicast recipients. The multicast tree-building algorithms are tightly-coupled to particular unicast algorithms. At domain boundaries, where differing multicast algorithms may interface, various ad hoc means are used to establish the tree. This is further discussed in section 4.2. Routers on a multicast tree store (source, group) pair information.

2.1 Existing Properties

Several properties, originally conceived for the LAN multicast environment, and later extended as desirable properties for internetwork multicasting, include:

- *Host Group Model conformance.* The Host Group Model is a multicast service model for datagram internetworks, developed in the mid-1980s by Deering [9]. It defines what the multicast service looks like to users of the internetwork service interface within a host; it does not define how that service is implemented. Further, it lists a set of properties a multicast routing protocol should exhibit, that contribute to its flexibility and generality; for example, a sender to a group need not know the location or identities of any of the group members, and the sender itself need not be a member of the group.

- *High probability of delivery.* The probability of successful delivery of multicast packets decreases when sending those packets over the wide-area. However, the successful delivery rate should remain high enough to allow for the recovery of lost/damaged packets by end-to-end protocols [8].

- *Low delay.* Low delay is an important property for many multicast applications, for example, audio conferencing. LANs impose very little delay on the delivery of multicast packets, but the delays over the wide-area are, inevitably, higher due to the greater geographic extent, and the greater number of links and switches packets must traverse. Therefore, optimizing multicast routes can be an important factor in minimizing delay exacerbation.

2.2 Proposed Properties

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:

- *Scalability.* With the internet growing at its current rate, we can expect to see a large increase in the number of wide-area multicasts. These can vary considerably in their characteristics. 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.

- *Robustness.* Any multicast routing algorithm should include features that provide robustness in terms of maintaining/repairing connectivity between group members.

- *Information hiding.* Information hiding is an important aspect of scaling. Routers/bridges, whose subnetwork(s) have no members with respect to a particular group, should not have to know any information as to the existence of that group, even if they need to forward multicast packets.

- *Routing Algorithm independence.* It is highly desirable that a multicast routing algorithm be designed independent of any unicast routing algorithm, resulting in much simplified multicast tree formation across domain boundaries.

- *Multicast tree flexibility.* Multicast applications vary considerably in nature, according to sender population, receiver population, traffic characteristics, and membership dynamics. Three example multicast applications with such differing characteristics are video-broadcasting, audio/video conferencing, and resource discovery. Therefore, a multicast delivery tree should be built so as to reflect the nature of the application.

3 Existing IP Multicast Algorithms

The essential aim of wide-area multicast routing is establishing a reasonably optimal path between a multicast source and the other members of the group. A multicast packet should only ever need to be replicated when a shared path diverges into disjoint paths. This has the result of incurring the least packet processing and forwarding overhead per multicast router in the path, and the least bandwidth consumption by multicast packets between the source and destination(s). To summarise, multicasting optimizes bandwidth consumption and transmitter costs.

In the following subsection we briefly describe some features of the current IP multicast environment. Subsequent subsections outline current IP multicast algorithms, namely the Distance-Vector Multicast Routing Protocol (DVMRP) and the Link-State Multicast Routing Protocol, repectively. A more comprehensive description of these algorithms can be found in [9] and [8].

3.1 IP Multicast Environment

Most broadcast LANs, such as Ethernet, FDDI, and ATM, intrinsically support multicast addressing. That is, most end-systems and routers on these types of LAN are able to distinguish multicast packets from other types of traffic by means of address type; IP supports *class D* addressing. A class D address is an address taken from a portion of the IP address space set aside for multicasting. Each class D address uniquely identifies a single host group.

Routers normally set their network interfaces to promiscuously receive all multicast packets, but a host only does so after a higher-layer application explicitly requests it to do so. A single router, called the ***membership interrogator***, or ***designated router***, polls the LAN for host memberships at intervals, to which only group member hosts reply, once for each group. This group-query/group-reporting mechanism is implemented on LANs in hosts and routers by means of the *Internet Group Management Protocol (IGMP)*.

3.2 DVMRP

DVMRP [5] is based primarily on ***Reverse Path Forwarding*** (RPF) - an algorithm devised by Dalal and Metcalfe [6] for internetwork broadcasting. DVMRP uses a modified RPF algorithm to allow members (and non-member senders) of a group to build a shortest-path sender-based multicast delivery tree. The first few[1] multicast packets transmitted from a source are ***truncated broadcast***[2] throughout the internetwork.

Once the first packet has reached those routers that have neither child subnets nor leaves with members on them, those routers are each responsible for sending a special message called a "prune" back one hop on the reverse-path tree. If the one-hop-back router receives prune messages from all of its subordinate routers, AND if its child subnets also have no members of the destination group, it in turn sends a prune message back to its predecessor.

In this way, information about the absence of group members propogates back up the tree towards the source along all branches that do not lead to group members. Subsequent packets from the same source to the same group are blocked from travelling down the unnecessary branches by the routers at the heads of those branches.

A mechanism was also designed for quickly "grafting" a pruned branch back onto a multicast tree; a router can cancel a previously sent "prune" message by sending a "graft" message to the same router. The "graft" message is propogated as far as necessary to rejoin the sending router to the multicast tree.

[1] How many depends on how long it takes the special "prune" message to reach the source.

[2] A broadcast to all subnetworks throughout the internet except those which are "leaf" subnets with respect to the source. A "leaf" subnet of the reverse-path tree for a particular source, S, is a child subnet that no other router uses to reach S. In the case of DVMRP multicast packets only reach "leaf" subnetworks that have at least one member.

3.3 Link-State Multicast Routing Protocol

The link-state routing algorithm was extended in [9] to support shortest-path multicast routing by simply having routers include, as part of the "state" of a link, a list of groups that have members on that link. Whenever a new group appears or an old group disappears from a link, the designated router on that link floods the new state to all other routers in the internetwork. Given full knowledge of which groups have members on which links, any router can compute the shortest-path multicast tree from any source to any group using Dijkstra's algorithm [1]. If the router doing the computation falls within the tree computed, it can determine which links it must use to forward copies of multicast packets from the given source to the given group.

4 A Critique of the Existing Multicast Architecture

In this section we present a critique of the existing source-based multicast architecture in light of the fact that the internet is a fast-growing, enormously complex, heterogeneous structure. In the not too distant future we expect to see huge numbers of multicast groups in existence at any one time.

4.1 Scaling Characteristics of Source-Based Trees

Poor scaling properties are inherent in multicast routing algorithms that build source-based delivery trees. The multicast algorithms we have discussed store ***per source*** information, which, if S is the number of active sources per multicast group, and N is the number of multicast groups present, results in a scaling factor of $S \times N$. This has serious consequences for routers in terms of storage and packet forwarding overheads.

DVMRP exhibits another scaling characteristic that is both interesting and alarming, namely, that routers not on a multicast tree are "charged" for staying off it, i.e. those routers not interested in sending/receiving multicast packets to/from a group are involved in the reception, generation, and interpretation of prune and graft messages, and additionally the storage of prunes, per (source, group) pair.

4.2 Unicast Routing Algorithm Dependence

The multicast routing algorithms we have so far presented are based on a flat internetwork consisting of one large autonomous system in which all routers are running the same multicast/unicast algorithms. In reality the internet is a complex, heterogeneous environment with ASs running internal routing protocols of their choice. Tight coupling between multicast and unicast algorithms complicates the development of unicast algorithms, since they must be modified to take multicast into consideration. This coupling also requires specialised solutions for multicasting between domains running different multicast algorithms[3]. Indeed, such solutions have yet to be developed for IP [3]. The MBONE encompasses only those networks and routers that have multicast forwarding capability and which are running the same multicast algorithm.

4.3 More Algorithm Specifics

A DVMRP router must invest a modest amount of processing power to determine which of its attached subnets are child and leaf subnets relative to a given source. This overhead is incurred whenever the router's distance or next-hop subnet for a given source changes, or whenever the distance to a given source reported by a router on an attached subnet changes. Therefore, the total overhead involved in determining child and leaf subnets depends on the stability or dynamicity of the internet.

There are two implications involved in disseminating group information in link-state packets: firstly, link-state packets are flooded throughout the internetwork by a LAN's *designated router* both as the result of normal topology changes, and group membership changes on any of its directly attached subnets; secondly, and more seriously, global group membership information is maintained by all routers, whether they form part of a multicast tree(s) or not. Whilst the bandwidth overhead due to more frequent generation of link state packets could be deemed as less significant as we see media bandwidths continually increasing, we consider more serious the overhead of having all routers in the internet store global group membership information as unacceptable.

5 CBT - The New Architecture

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]. He used a single centre-based forwarding 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".

A core-based tree then, 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[4] between the nodes on the tree, since multicasts vary in nature, and correspondingly, so can the form of a core-based tree.

Why then, is a core-based tree (CBT) architecture so attractive compared with the source-based architecture? The key architectural features which drive the CBT approach are listed below:

- *Scaling.* This is the fundamental premise driving CBT. 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. Each router on the tree need only store incident link information per group (i.e. per tree) as opposed to incident link information per (source, group) pair. This represents the minimum possible any router need store with respect to its membership of a particular group. Routers not on the tree require no knowledge of the tree whatsoever.

- *Tree creation.* The formation of core-based trees is *receiver-based*, i.e. no router is involved in becoming part of a tree for a particular group unless that router is intent on becoming a member of that group (or is on the path between a potential member and the tree, in which case that router must[5] become part of the tree). This implies that a tree is not built from a sender - only one tree is ever created per group. This is of significant benefit to all routers on the shortest-path between a non-receiver sender and the multicast tree, since they are incurred no tree-building overhead.

- *Unicast routing separation.* Core-based tree formation and multicast packet flow are decoupled from,

[3] "Tunnelling" has been defined as a technique for transporting multicast packets between multicast-capable routers. A "tunnel" then, is a sequence of routers that do not have multicast capability. A "tunnel" is created using a technique based on loose source routing, encapsulation, or a combination of both.

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

[5] A router has the option to refuse a request to become part of a multicast tree.

but take full advantage of, underlying unicast routing, irrespective of which underlying unicast algorithm is operating. All of the multicast tree information can be derived solely from a router's existing unicast forwarding tables, with no additional processing necessary. These factors result in the CBT architecture being as robust as the underlying unicast routing algorithm - most of which are designed with robustness as a high priority.

In this architecture we can identify two distinct routing phases which provide the architecture with its scalability: firstly, unicast routing is used to route multicast packets *to* a multicast tree, allowing multicast groups and multicast packets to remain "invisible" to routers not on the tree. This is achieved by using the unicast address of the centre (core) of the multicast spanning tree in the destination field of multicast packets originating off-tree; secondly, once on the corresponding tree, multicast packets ***span*** the tree based on the packet's group identifier, or group-id[6] (similar to a class D IP address). We consider this two-phase routing approach an important advancement in multicasting. It has only been possible as a result of recognising the need for having just one tree per group.

With respect to IP networks, CBT requires no partition of the unicast address space.

A diagram showing a single-core CBT tree is shown in Figure 1.

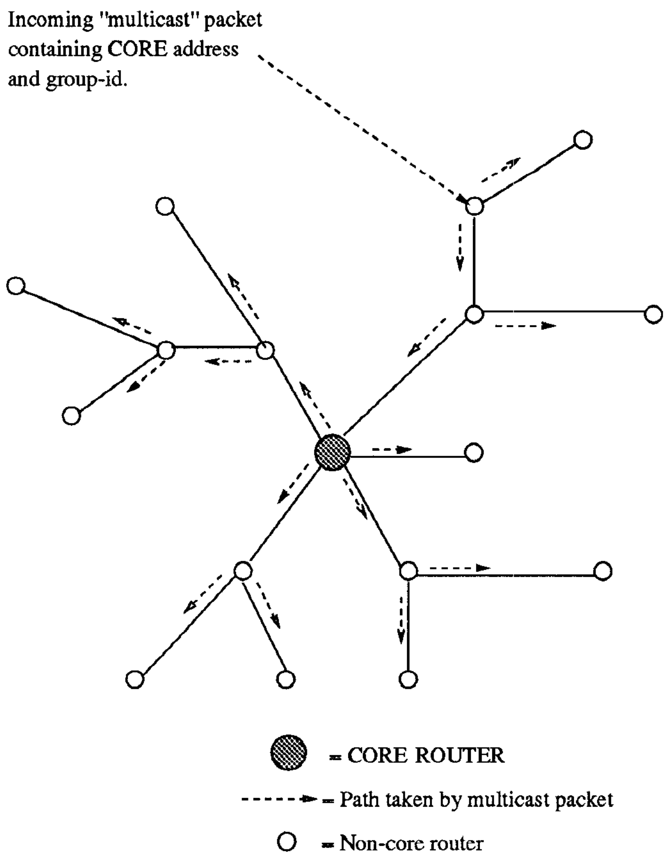
Incoming "multicast" packet containing CORE address and group-id.
= CORE ROUTER
= Path taken by multicast packet
= Non-core router

Figure 1: A single-core CBT tree

5.1 Disadvantages of the CBT Architecture

The following weaknesses can be identified through having one core-based multicast tree per group, namely:

- *Core placement and shortest-path trees.* Core-based trees *may* not provide the most optimal paths between members of a group. This is especially true for small, localised groups that have a non-local core. A dynamic core placement mechanism (see section 7) should prevent this from happening. In general, however, we feel that manual "best guess" placement will be aceptable for most situations.

- *The Core as a Point of Failure.* The most obvious point of vulnerability of a core-based tree is its core, whose failure *can* result in a tree becoming partitioned. Having multiple cores associated with each tree solves this problem (though at the cost of increased complexity).

[6]The core (unicast) address and the (multicast) group-id could be one and the same, i.e. there could be no separate group-id space, but this constrains assignment of addresses.

6 CBT - The Protocol

This section describes the CBT protocol.

6.1 CBT Addresses, Identifiers, and Group Names

In CBT there are one or more core addresses and a group identifier associated with every group. The core addresses are the normal unicast addresses of the core routers. These addresses are used to get packets *to* the tree. Once on the tree, the packet is multicast based on a globally unique group identifier, or group-id.

The ***group-id*** is a flat, 32-bit identifier chosen independently by each core router from a subset of group-ids with which it is configured. Each potential core router then, has a unique subset of 32-bit group-ids, each of which can be assigned to identify a single group.

The ***group name*** is a human-readable string whose structure is based on the "dotted" notation of the Domain Name System (DNS). We present our discussion in the context of DNS, but similar principles can be applied to X.500 [10]. A proposal is made in [10] to incorporate DNS infomation into the X.500 directory information tree.

A group name is decided upon externally. It should be chosen so that some information as to the nature of the group can be derived from it. The name itself should be taken from the namespace unique to the group initiator. In this way, name clashes are only possible within that namespace. However, such clashes are easily avoided by simply having each name authorised by the local system administrator. For example, the name of an audio conference initiated at University College London could be called cbttalk.cs.ucl.ac.uk. Once a group's name, core address(es), and group-id have been established, directory service must be updated (in the case of DNS, which cannot be dynamically updated by hosts, the information must be manually entered[7]).

Whenever a host wishes to join a group for which it knows only the name, it must query directory service for both the group-id and core address(es) corresponding to that group name. Subsequently, the host can generate a group membership report as part of IGMP containing the information necessary for the local router to send a request to join the group.

6.2 Cores and Core Placement

CBT involves having a single-core tree per group, with additional cores to add an element of robustness to the model. The cores associated with a particular tree are collectively known as a *core list.* Each router in the core list is assigned a *priority* or *ranking* by the group initiator, and therefore is an explicitly ordered list. The highest-ranked in the list is known as the *primary core*, the next-highest being the *secondary core*, followed by the *tertiary core*, and so on.

We now outline several trivial heuristics for core placement[8]. It is considered likely[9] that the placement of a core for a multicast tree will assist in optimizing the routes between any sender and group members on the tree. Multicast path-finding algorithms from a known source have been devised [4] for networks of varying multicast capability, but there exists no polynomial time algorithm that can find the centre of a dynamic multicast spanning tree. We consider this a topic requiring further research.

Cores could be statically configured throughout the internet - there need only be some relatively small number of cores per backbone network[10], 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 groups characteristics. In the absence of a dynamic core placement mechanism, a core should be "hand-picked", i.e. selected by external agreement based on a judgement of what is "known"[11] about the network topology between the current members.

6.3 CBT Forwarding Algorithm

We distinguish between CBT*control packets* and multicast *data packets.* CBT control packets do not carry user data and are primarily concerned with tree building, re-configuration, or tree teardown. Multicast data packets on the other hand, carry only user data to/on a tree once the tree has been established. CBT control packets and multicast data packets travel in IP datagrams. CBT control packets are forwarded as per unicast and are processed at each hop. Therefore the IP destination of control packets is always set to the next-hop on the path to the corresponding core.

- *CBT data packet forwarding algorithm.* Multicast data travels in an IP packet. Therefore, CBT-capable routers must have a way of recognising when mulitcast data packets are being carried in an IP packet. To make this possible, data packets destined for a particular group tree carry the group core address in the "destination field" and the group-id in the "option" field of the IP packet's header. This implies the need for a new IP "options number" to be defined for CBT.

  On arriving at an on-tree router, the core address in the destination address field of the IP header is discarded, and the group-id in the option field is placed in the destination address field. The reasons for this are twofold: firstly, it allows for faster on-tree switching, since there is no option processing necessary; secondly, since multicast-capable (non-CBT) routers process all multicast packets by default, CBT multicast packets are prevented from

[7]There is a time-delay between updating a DNS server and when the information becomes available, as well as the potentially much longer delay in getting DNS updated by a system administrator. To overcome these shortcomings, each site could have a certain number of permanent groups, i.e. cores addresses, group-id, and group name would remain fixed. Whenever a host wishes to initiate a group, it simply chooses one from a selection of these tuples which is not in use.

[8]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

[9]Yet to be proven by experiments.

[10]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.

[11]alleged to be known

being forwarded (unicast) by these routers, to the core[12].

CBT routers forward arriving multicast data packets based on information contained in their CBT Forwarding Information Base (FIB). The FIB contains a list of interfaces for each group with which it is associated. A multicast data packet is forwarded on the corresponding outgoing interfaces.

6.4 Protocol Overview

CBT routers will continue to use IGMP to monitor group membership on their directly attached subnetworks. CBT will also continue to use both the "all hosts" and the "all routers" addresses, and therefore these two addresses must remain reserved. At the link level, CBT multicasts will appear identical to existing IP multicasts. Taking Ethernet as an example, currently the low-order 23-bits of the destination address field are mapped directly onto the low-order 23-bits of the special Ethernet multicast hardware address. CBT will continue to do this, except the destination address will not contain a class D address. However, this should not affect link-level multicasting at all.

CBT then, is an overlay of the underlying unicast routing algorithm. Its position and relationship to other IP protocols is shown in Figure 2.

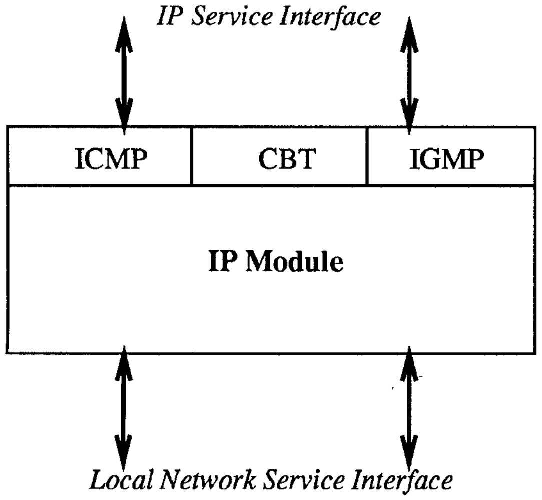

Figure 2: Protocol Relationships

CBT operation is invoked whenever a router receives a group membership report as part of IGMP from some host on a directly attached subnetwork. A group membership report contains the group-id. A group membership report *may* also have the core list "piggybacked" on it. Alternatively, the core list is transmitted separately to the local router[13]. Once the local router has received the core list, it can proceed to send a JOIN-REQUEST.

A JOIN-REQUEST includes, as part of its header, the 32-bit group-id, and an "active" bit which is set before sending a join. The relevance of the "active" bit will be explained in section 6.6.1. The JOIN-REQUEST is then forwarded to the next-hop router on the path to the core, as determined by the unicast forwarding table. Section 6.7.2 discusses what happens if there is no such next-hop.

The join continues its journey until it either reaches the addressed core, or reaches a CBT-capable router that is already part of the tree, as identified by the group-id. At this point, the join's journey is terminated by the receiving router, which then decides whether to acknowledge the join request. A JOIN-REQUEST is normally acknowledged by means of a JOIN-ACK[14].

All the CBT-capable routers traversed by a JOIN-ACK 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. 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 acknowledgement. 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).

Each router on a CBT tree records its *parent* and *child* interfaces with respect to a particular tree, i.e. group. The *parent* interface is that over which a JOIN-ACK was received. A non-core router can have only one parent interface per CBT tree. A *child* interface is one over which a JOIN-ACK has been forwarded with respect to a particular group[15]. A non-core router may have multiple child interfaces per CBT tree.

A QUIT-REQUEST is a request by a non-core 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. The QUIT-REQUEST is 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 non-core router that sends a QUIT-ACK in

[12]Interoperability has been an important design goal throughout the development of CBT. For this reason alone, we decided that the group-id should be selected from the class D IP address space.

[13]On a multi-access LAN one router will be elected as designated router

[14]However, a negative acknowledgement (JOIN-NACK) may be sent in reply to a join request for various reasons, for example, in the event of loop detection, or simply because the router does not wish to become part of a CBT tree for that group.

[15]And over which no QUIT-ACK has been sent (see below).

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 retransmissions gives the sending non-core router the right to remove the relevant parent interface information, and by doing so, removes itself from the CBT tree for that group.

6.5 Path or Node Failure

Parent/parent link failure is recognisable as a result of a "keepalive" mechanism operating between adjacent (directly attached) on-tree routers. The "keepalive" mechanism is implemented by means of ICMP echo request/reply messages sent from child to parent.

For any non-core router, if its parent router or path to the parent fails, that non-core router has one of two options (configurable) for failure recovery: it can either attempt to re-join the tree by sending a JOIN-REQUEST to the highest-priority reachable core, thus keeping the failure transparent to the rest of the downstream branch; alternatively (and as a result of the above mechanism failing) the router subordinate to the failure can send a FLUSH-TREE message downstream, thus allowing each router to independently attempt to re-attach itself to the tree, possibly via a better route than previously. Routers must always attempt to join the highest priority reachable core.

It should be noted that on re-start, all CBT routers must discard all state pertaining to CBT trees.

6.6 In the Presence of Unicast Transient Loops

Routers rely on underlying unicast routing in order to be able to forward 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 retries 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.

6.6.1 Loop Detection

The CBT protocol incorporates an explicit *loop-detection* mechanism, depending on the nature of the join request.

A JOIN-REQUEST has an "active" bit and a "rejoin" bit associated with it. The "active" bit, when set, indicates that the join is in the process of reaching a tree router. This bit remains set until the join hits a tree router, at which time it is unset, and the join can be acknowledged by means of a JOIN-ACK. The "rejoin" bit is set if the originating router is attempting to re-join the tree subsequent to that router's parent or parent-link having failed. A JOIN-REQUEST which contains the combination "active" bit unset and "rejoin" bit set results in the join being forwarded over each routers parent interface for that tree. In this way the join acts as a *loop-detection* packet.

Should the router that originated the active join receive the corresponding inactive join, it must immediately send a QUIT-REQUEST on the interface over which the active join was sent. After some short time the router may try again to reattach itself to the tree in the hope underlying unicast routing has converged.

6.7 Core Failure

Certain conditions may arise whereby a router (possibly a non-member sending router) cannot reach a core. If none of the cores in the core list is reachable/available, the router/host has no option but to wait some random period before retrying.

6.7.1 Approaches to Failure

For reasons of robustness, we need to consider what happens when a primary core fails. There are two approaches we can take, namely have:

- *single-core CBT trees.* Paths as well as cores themselves can fail, which may result in parts of the network being partitioned from others. In the presence of CBT trees this can mean a single tree can itself become partitioned. To cater for tree partitions, we have multiple "backup" cores to increase the probability that every network node can reach at least one of the cores of a CBT tree. At any one time, a non-core router is part of a single-core CBT tree.

- *multi-core CBT trees.* Multi-core CBT trees are most useful for groups that are topologically widespread. Each core is then strategically placed where the largest "pockets" of members are located so as to optimize the routes between those members. Each of the cores must be joined to at least one other, and a reachability/maintenance protocol must operate between them. There exists no ordering between the multiple cores, and senders send multicasts preferably to the nearest core. The essential difference between multi-core and single-core trees is that single-core trees have no explicit protocol operating between the "backup" cores (even though the cores are joined together), and, for any sender at any instant, there is one core to which it must attempt to send its multicast packets.

Although it may seem advantageous to have multi-core CBT trees as opposed to single-core trees, complex failure scenarios and the overhead of an explicit protocol operating between the cores have led us to the conclusion that management of multi-core CBT trees is too complex to justify. Our choice of having, in the worst case, multiple partitioned single-core trees per group, offers a trade-off between complexity and robustness. We feel that our design offers an acceptable level of robustness.

6.7.2 Robust Single-Core Trees

To expand on our approach, we are presenting a model based on robust single-core trees. Although our design provides each tree (group) with multiple cores, which join each other at group initiation time, it is the primary core that is considered the "central hub" of a tree, with the additional cores simply providing an element of robustness to the design. This allows senders of multicast packets more than one destination option, thus increasing the probability that every network node can "see" a particular tree.

At group initiation time, each core in a group's core list must attempt to join the primary core for the group. "Cycles" are avoided between the the cores by having each core set its "re-join" bit prior to sending a JOIN-REQUEST. A non-primary core that cannot reach the primary must attempt to join the highest-ranked core that is reachable from it.

A core that has not (yet) been successful in joining the primary core, and which receives multicast data packets, sends an ICMP cbt_redirect to the originator of those packets. An ICMP cbt_redirect prevents data packets disappearing down "black holes" by indicating to the sender to target its packets elsewhere, but giving no precise indication as to which core the sender should forward its packets.

Once the cores have successfully joined the primary core, they act essentially as non-core routers except they can be the targets of CBT control packets and multicast data packets, as well as being able to generate ICMP cbt_redirects (which non-core routers cannot do).

All senders should direct multicast packets (and control packets) at the highest-priority reachable core. Under normal, failure-free circumstances this will be the primary core router. The reason for this is to force the emphasis of the tree onto just one router.

If the primary core should fail, the recovery scenario is the same as that described in section 6.5.

7 Future Work

Work still needs to be done in the following areas:

- *Testing and analysis.* Once a prototype implementation is in place, it will be important to establish the answers to some frequently-asked questions. For example: how will the shape of a CBT tree affect performance/delay characteristics between members; what are per node switching overheads/delays on and off-tree; how robust is CBT in the presence of link failures, and how quickly can CBT adapt to such failures?

- *Dynamic core selection and placement.* Our scheme, presented in section 6.2, may involve management overhead to re-select and re-place the core(s) for those groups whose characteristics vary considerably over the lifetime of the group. Therefore, dynamic core selection and placement algorithms/heuristics and mechanisms need to be developed. These *could* contribute to the optimality of core-based trees.

- *Dynamic group-id assignment.* We consider this complementary to dynamic core selection and placement. As multicast capability has been developing and the number of multicast applications growing, the issue of dynamic multicast group address[16] assignment has very much been left in the background. The failure to address this problem[17] means that group address conflicts are more and more likely to occur.

  We consider the development of a dedicated *multicast directory service* which would be responsible for all aspects of group management, a long term goal of internetwork multicasting in general. A similar idea has recently been suggested in [13].

- *Scoping.* Refined scoping mechanisms need to be developed. We consider the cores of core-based trees to be focal points with respect to scoping. As

[16] For the purpose of our discussion, in this section read group address as equivalent to group identifier

[17] A distributed algorithm for distributed class D address allocation has recently been mentioned by Van Jacobson et al. in a remote conference, but has not yet been written up.

such, we envisage a core-based tree spanning multiple ASs to be made up of a hierarchy of concatenated single core trees, thus allowing for scoping within each sub-tree. A sub-tree may comprise any size from a LAN to an AS, or even multiple ASs.

- *Policy Routing and Multicasting.* Policy routing [2] and multicasting have so far evolved independently of each other. As the internet becomes more policy oriented, it has yet to be investigated what effects this will have on multicasting. If internet-wide AD policies are going to be dynamic and wide-ranging, group membership could be severely constrained resulting in a lack of *openness* - a desirable property of internetwork multicasting.

- *CBT Security Architecture.* Security services such as authentication and encryption have yet to be incorporated into the CBT architecture. The architecture permits a number of different security approaches. These will be investigated by the author in due course.

8 Summary

In this paper we have discussed the existing multicast architecture, and current IP multicast algorithms. We have shown the current architecture, based on multicast trees rooted at each sender, to scale poorly in an internet consisting on hundreds of thousands of multicast groups at any one time. Furthermore, existing multicast schemes were not designed to operate in a heterogeneous unicast routing environment and therefore do not operate "invisibly" across or inside heterogeneous domains. Subsequently we presented a new multicast routing protocol for IP based on a different architecture that builds core-based multicast trees, one per group. This implies that routers not on a multicast tree need know nothing about the existence of groups of which they are not a member, and therefore can route packets from non-member senders to the tree as per unicast. Routers on a multicast tree need only know their immediate uptree and downtree neighbour routers - a minimal amount of information with respect to a group. The architecture is thus scalable, low-cost, and efficient. It can be applied to networks and technologies other than IP.

Work is continuing to refine CBT and eliminate potential weak points where possible. A CBT implementation is planned shortly, after which we can realistically assess its performance, especially when compared with existing multicast techniques.

Acknowledgments

There are several people whom we would especially like to thank for their valuable suggestions and contributions to CBT. Ian Wakeman (UCL) made the suggestion to separate the group identifier from the IP core address. We considered this an important design issue offering several advantages over the original design. Benny Rodrig (RAD Network Devices, Ltd) suggested the loop detection mechanism of forwarding a special packet uptree. Zheng Wang (UCL) proposed swapping the group-id and destination (core) address once on-tree. Also, Joel Halpern (Network Systems Corporation), Steve Deering (Xerox Parc), and Scott Brim (Cornell University) for their general constructive comments and suggestions since the conception of CBT.

References

[1] J. E. Hopcroft A. V. Aho and J. D. Ullman. *Data Structures and Algrorithms.* Addison-Wesley, Reading, Mass, U.S.A., 1983.

[2] Lee Breslau and Deborah Estrin. Design and Evaluation of Inter-Domain Policy Routing Protocols. *Internetworking: Research and Experience*, 2:177–198, September 1991.

[3] Scott Brim and John Moy. Support for Multicast Communications Across Wide-Area Networks. *High Performance Network Research Report, Cornell Univ.*, June 1992.

[4] Ching-Hua Chow. On Multicast Path Finding Algorithms. In *Infocom, Conference on Computer Communications*, pages 1274–1283. IEEE, April 1991.

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

[6] Y. K. Dalal and R. M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of the ACM*, 21:1040–1048, December 1978.

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

[8] 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.

[9] S. E. Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, California, U.S.A., 1991.

[10] S. E. Hardcastle-Kille. RFC 1279, X.500 and Domains. *SRI Network Information Center*, September 1991.

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

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

[13] S. Zabele R. Braudes. RFC 1458, Requirements for Multicast Protocols. *SRI Network Information Center*, May 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.