Core Based Tree (CBT) Multicast

An Analysis of Multicast Routing Architectures

Tony Ballardie
e-mail: A.Ballardie@cs.ucl.ac.uk

Department of Computer Science,
University College London,
Gower Street, London WC1E 6BT,
England, UK.

Jon Crowcroft
e-mail: J.Crowcroft@cs.ucl.ac.uk

Department of Computer Science,
University College London,
Gower Street, London WC1E 6BT,
England, UK.

Abstract

Multicasting is a technique that enables a single packet transmission to reach one or more destinations or *group*. The primary benefits of a packet reaching multiple destinations from a single transmission are threefold: *bandwidth minimization*; *the exploitation of parallelism in the network*; the *optimization of transmitter costs*.

In this paper we investigate and analyse each of the different network layer multicast algorithms and protocols, looking in particular at their *scalability*, since multicast scalability was the primary motivator for this work. We measure *scalability* in terms of *network state, processing*, and *bandwidth costs*.

We proceed to present a new multicast architecture, designed for best-effort, connectionless datagram networks such as the IP Internet. This new architecture typically offers considerably more favourable scaling characteristics than do existing multicast schemes.

The quintessential question this work poses is: how can multicast be best achieved? Our conclusion is that there is no *best* way, but there are trade-offs to consider for each of the different methods, and each method has its place in the range of multicast solutions, just as each of the *unicast* routing protocols has its place in the Internet[1].

1 Introduction

Multicast 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 [20] and ATM [10] specify multicast as part of their service.

[1]Examples include: RIP, OSPF, IS-IS, IGRP, EGP, BGP, etc.

The multicast backbone, or MBONE [15], is a "virtual" network overlay of the IP Internet comprising hosts (acting as routers), and networks, with multicast capability. The benefits of multicast are becoming more apparent and are being realised by a wider community, and the MBONE is now becoming less "virtual" as multicast capability is becoming more integrated into the internetwork infrastructure, i.e. IP routers are being given multicast capability.

The diversity of multicast applications includes those for audio and video conferencing [5], replicated database updating and querying, software update distribution, stock market information services, and more recently, resource discovery [23, 22]. In general, multimedia communication [35, 33] is an area for which multicast offers an invaluable service. It has therefore been necessary of late to address all aspects of scalability with regards to multicast routing algorithms (e.g. bandwidth, memory requirements), since, if they do not scale to an internetwork size that is expected in the foreseeable future (given the growth rate of the last several years), they can not be of longlasting benefit to the internetwork user community. This motivates the need for new multicast algorithms to be investigated.

This work is the culmination of one such investigation, the end product being a new multicast architecture and protocol for datagram networks.

2 Paper Overview

In section 3 which follows, we provide a background to our work, which focusses primarily on that of Wall [34] and Deering [12]. We concentrate on their work for two reasons: Wall pioneered centre based forwarding for broadcasting, and adapted it for selective broadcasting (which is analogous to multicasting). Deering developed much of the multicast capability that we see in evidence today in our networks. He invented several multicast algorithms, two of which we will look at in detail.

As part of the background we also present an overview of *Protocol Independent Multicast (PIM)* [21] – the multicast architecture developed shortly after the emergence of the *Core Based Tree (CBT)* multicast architecture, which is the centre-piece of this paper.

Section 4 introduces the Core Based Tree (CBT) *architecture*. We identify some of the shortcomings of the existing multicast architecture, which have, in part, motivated CBT's design. This section also provides a justification for the CBT architecture, emphasizing why it can be particularly advantageous. Section 5 looks at the implications of utilizing shared trees for multicast, identifying its potential disadvantages. Section 6 provides summaries of a variety of simulation results, carried out to compare and contrast CBTs with shortest-path trees (SPTs) based on certain criteria such as delay, cost, and link utilization. Section 7 discusses core placement and management, and section 8 summarizes how Wall's centre trees differ from our CBT trees.

Finally, section 9 ddresses multicast *scalability* in detail. We analyse the scalability of four network layer multicast algorithms (DVMRP, M-OSPF, PIM, and CBT), based on the following criteria: group-state information, bandwidth consumed, and processing costs.

3 Background

The background work we discuss here concentrates on *network layer* multicast. We do not discuss transport layer (reliable) multicasting, which is a different problem space involving end-to-end delivery. Insights into transport layer multicast can be gained from [16, 32], and elsewhere in the literature.

We focus on three areas of related work. Two of these involve relatively recent contributions to network layer multicast, namely Deering's work [12], and an ongoing collaborative effort (Estrin, Deering, et al. [21]) known as *Protocol Independent Multicast (PIM)* [21]. Deering's work can be considered a culmination of the analysis of much earlier work done on multi-destination delivery by Delal and Metcalfe [9], and others [2, 34]. Deering's work represented a considerable advancement in multicast technology.

We also look at earlier work done by Wall [34]. This is so that our work can be effectively compared and contrasted with his, seeing as our work uses the same *shared tree* paradigm for multi-destination packet delivery. Looking at Wall's algorithms and ours, we see that there are few similarities between Wall's methods of tree building and maintenance, and those of our own shared trees – indeed the technical similarity largely ends there.

PIM was motivated by our own work. Both our own protocol, known as the *Core Based Tree (CBT)* protocol, and PIM, can be considered new approaches to multicasting in a datagram internetwork. We will see that PIM has taken up the "good parts" of CBT, and has augmented CBT's features in order to allay CBT's disadvantageous properties, the most prominent of which is the potential for sub-optimal paths (which usually equates to delay) between two receivers. However, as PIM currently stands, it is debatable whether "its ends justifies its means", i.e. are its advantages, in terms of performance and delay, considerable enough to justify PIM's protocol complexity.

3.1 Wall's Work on Centre Based Forwarding

Wall set out to show that it is possible to build a single, "centre" rooted *broadcast* delivery tree, shared by a set of network nodes, any of which may send and/or receive, with the properties that it had low, but not minimum, cost, and incurred low, but not minimum, average delay between random senders on the tree. He noted: "we can't hope to minimize the delay for each broadcast if we use just one tree, but we may be able to do fairly well, and the simplicity of the scheme may well make up for the fact that it is no longer optimal". Wall proved that the *maximum* delay bound of an optimal centre based tree is twice that of a shortest-path tree. Minimal cost *and* minimal delay can not be achieved using any one type of distribution tree [34]. He also adapted his broadcast algorithm for *selective* broadcast, and showed that, for a centre-rooted selective broadcast tree, the maximum delay bound remains the same.

The method he devised for finding the "centre" involved each node using its local information to calculate two values: the *maximum delay* to send a packet to its *most distant* destination; the *average delay* to send a packet to a *random* destination. Each node's values were then pooled, and by means of a simple algorithm (see [1]) the "centre" chosen as the node either with the smallest maximum delay, or the node with the smallest average delay. Either of these *criterion* was considered a suitable candidate for the " centre". For a more comprehensive discription of Wall's tree-building algorithms, see [34] and [1].

There are various points we need to make to make clear how Wall's early work on multicast diverges from the multicast model of today. Deering enumerated various properties that contribute to multicast's flexibility and generality. Collectively, these properties comprise the *Host Group Model* [12]. Two of these: *senders need not be members*, and *membership should be dynamic and autonomous*, conflict with Wall's operational environment we have just described. Also, Wall's algorithms have not made any provisions for network failures that lead to tree breakages, i.e. his algorithms are not robust. Therefore, Wall's algorithms could not be used in todays multicast environment without considerable adaptation.

3.2 Deering's Work

Deering's work [12] involved extending the three basic routing algorithms: *spanning tree, distance-vector*, and *link-state*, to achieve truncated-broadcast, and multicast, packet delivery. The basic algorithms were designed to operate in a single-level, or *flat*, internetwork (as they still do as of writing this paper), but Deering also described how a combination of his algorithms could be used in a very large internetwork that is structured as a *hierarchy*.

A *truncated-broadcast* tree is similar to a *broadcast* tree, but is pruned of *leaf* subnetworks[2]. The advantage of such a tree over a broadcast tree is that it usually results in fewer packet copies being generated whilst incurring little extra overhead to establish.

We therefore limit our discussion of Deering's work to those algorithms and associated protocols that are in use in the various portions of today's Internet that have multicast capability – the so-called MBONE [15].

The MBONE consists primarily of routers[3] running an instance of the distance-vector multicast algorithm known as the *Distance-Vector Multicast Routing Protocol (DVMRP)* [8, 12].

Whilst both the distance-vector and link-state multicast *algorithms* were invented by Deering, M-OSPF was developed by Moy [27].

3.2.1 The Distance-Vector Multicast Algorithm

DVMRP [8] is a protocol implementation of the distance-vector multicast *algorithm* proposed by Deering [12]. The distance-vector multicast *algorithm* uses (destination, distance) vectors to advertise multicast-capable subnetworks to participating routers. These vectors are exchanged between neighbouring multicast-capable routers.

DVMRP is based primarily on *Reverse Path Forwarding* (RPF) - an algorithm devised by Dalal and Metcalfe [9] for internetwork broadcasting. Deering modified the RPF algorithm slightly to eliminate the possibility of multicast duplicates being sent across multi-access links [11].

The RPF principle is quite simple: if a packet arrives via a link that is the shortest-path back to the source of the packet, then forward the packet on all outgoing links (so-called *child* links). Otherwise, discard the packet. DVMRP restricts the number of *outgoing links* to those which are not leaf subnetworks, unless a leaf subnetwork has group member presence, in which case a copy of the multicast packet will be forwarded over it.

[2]A "leaf" subnetwork is one that is not used by any router to reach a particular source subnetwork. Subnetworks with only one router attached also constitute leaf subnetworks.

[3]IP routers that route internetwork unicast traffic are only just beginning to be manufactured with in-built multicast capability. Currently, however, UNIX hosts running multicast protocol code make up the "routers" on the MBONE.

DVMRP uses the RPF strategy which results in a shortest-path, sender-rooted[4], delivery tree being formed between a sender and the corresponding group members.

A number of packets from a *new* sender will actually span the *truncated broadcast* tree rooted at the source. Only subsequently is the truncated-broadcast tree *pruned* back to become a true multicast delivery tree. For this to happen, a multicast packet for a particular (source, group) pair must first reach all *leaves* of the truncated-broadcast tree. A router connected to a leaf subnetwork may generate a *prune message* for the corresponding (source, group) pair, provided none of its leaf subnetworks have any members on them. Prune messages are always sent one-hop back towards the source, provided the criteria just specified can be satisfied. Prune state in routers prevents traffic for the corresponding (source, group) pair being forwarded on the links over which corresponding prunes have been received.

In summary, prune messages prevent multicast streams from reaching those parts of the internetwork that are not interested in receiving them. Prune state is "soft state", and is refreshed at fixed intervals.

After a number of timeout periods, a router connected to a leaf subnetwork will cease sending prunes upstream, since it does not know whether the source is still sending to the group. If the source *is* still sending, multicast traffic for the (source, group) will traverse the previously pruned branch, and new prunes will be generated provided there are no downstream receivers.

Should a group member appear on a previously pruned branch of a multicast tree, a mechanism was designed for quickly "grafting" back such a branch onto the tree. It requires that a router that previously sent a prune message for some (source, group) pair, send a *graft* message to the previous-hop router for that same (source, group) pair. If the receiving router in turn has sent a prune uptree, it would also be required to send a graft one-hop back towards the source, via the same path its prune was sent on. Thus, graft messages result in the removal of "prune state", thereby restoring a branch as part of a multicast delivery tree.

3.2.2 The Link-State Multicast Algorithm

M-OSPF is a protocol implementation of the link-state multicast *algorithm* proposed by Deering [12]. Link-state routing requires that participating routers periodically monitor the state of all (or a subset) of their incident links. This status information is then transmitted to all other participating routers by means of a special-purpose flooding protocol.

In order to provide link-state multicast routing, the link-state routing algorithm was extended to allow the presence of a multicast group on a link to become part of the "state" of that link. Thus, whenever a group appears or disappears, the state of that link changes, resulting in the designated router for that link flooding the new state to all other routers in the network.

Link-state multicast routers therefore, have complete knowledge of which groups are present on which links, throughout the domain of operation. Using this information, a router can use Dijkstra's algorithm to compute the shortest-path tree from any source to any group. Routers receiving multicast packets use this computation

[4]Throughout this paper, whenever we mention sender-rooted (or source-based) delivery tree, *source* should be interpreted as *source subnetwork*.

to establish whether they fall within the computed delivery tree with respect to the packet's source, and if so, to which next-hop(s) a packet must be forwarded.

The M-OSPF protocol is an interior gateway (IGP) designed to operate within an Autonomous System (AS). However, it can also be used to route multicasts hierarchically when an AS is divided into *areas.* Inter-area links thus form a backbone. Certain M-OSPF routers at the boundary of each area are responsible for routing multicasts between areas. These boundary routers are called *wild-card receivers.*

Routing multicasts between areas, and between AS's, is much more complex and less efficient than intra-area multicasting, since group membership information is sent to the backbone area in the form of summary link-state advertisements (LSAs), but the backbone area does not distribute this information to other areas. Hence, non-backbone areas are ignorant of other areas' group memberships. All multicasts generated *within* an area are delivered to the area's wild-card receiver(s), where they are discarded if group membership is exclusive to that area.

M-OSPF is the only multicast routing protocol to date that offers explicit support for multiple *types of service (TOS).* IP datagrams can be labeled with any one of five types of service (TOS), namely: *minimum delay, maximum throughput, maximum reliability, minimum monetary cost,* and *normal service.* M-OSPF calculates a separate path for each {source[5], destination, TOS} tuple, using Dijkstra's algorithm. Paths are calculated *on-demand* and cached, thereby reducing the burden imposed on routers by spreading particular route calculations over time.

Finally, because each M-OSPF router calculates its own multicast delivery tree from the perspective of the source subnetwork, it knows the distance to each downstream subnetwork where group members are located. The router thus has the ability to immediately discard received multicasts that will never reach a particular receiver(s), based on the IP TTL in the received packet [26].

3.3 Protocol Independent Multicast (PIM)

The PIM architecture was designed to establish efficient distribution trees for the cases where groups may be *sparsely* or *densely* distributed, i.e. PIM can be configured to adapt to different group and network characteristics. As a result, there are two PIM modes: *sparse mode* and *dense mode.*

As its name suggests, PIM is independent of whichever underlying unicast routing protocol is operating, unlike DVMRP or M-OSPF, which rely on particular features of their corresponding unicast routing protocols for their correct operation.

Dense-mode PIM is quite similar to DVMRP, but without the unicast protocol dependencies. For example, DVMRP uses the "poison reverse" technique [6] for leaf router detection. This involves advertising "infinity" for a source to the previous-hop router on the path to that source. The absence of such advertisements is an indication that a downstream subnetwork is a leaf. Furthermore, PIM routers do not calculate their set of outgoing *child* interfaces for each active source, but forward on all outgoing interfaces until such time as prune messages are received from a downstream router(s). Dense-mode PIM is thus said to be *data driven.*

[5] "Source" should be interpreted as "source subnetwork".

For the case where group members appear on a pruned branch of the distribution tree, PIM dense-mode, like DVMRP, makes use of *graft messages* to re-establish the previously pruned branch on the delivery tree.

PIM dense-mode is most likely to be the preferred mode of use in resource-rich environments, such as a campus LAN, where a group is likely to be uniformly dense. Only in such an environment is data driven flooding of multicasts, and subsequent pruning (and associated storage), acceptable.

Sparse-mode PIM, on the other hand, is the most likely mode for inter-domain (wide-area) multicasting. Sparse-mode PIM allows group members to receive multicast data either over a *shared tree*, which receivers must explitly join first, or over a *shortest-path tree*, which a receiver can create subsequently, in an attempt to improve delay characteristics between some active source, and itself. When a receiver creates a shortest-path to a particular source, it *prunes* itself off the shared tree for that (source, group) pair, but will continue to receive data packets for the group over the shared tree from all other sources.

The shared tree is built around so-called *rendezvous points (RPs)*, of which there may be several for robustness purposes. A new receiver need ever only join one RP if there are several, but a sender must send data packets to each of the RP's, so that all receivers for the group receiving over the RP "see" the multicasts [21].

The motivation behind sparse-mode PIM is the desire to accommodate sparsely distributed groups, i.e. the type of group that is most prevalent in wide-area internetworks. Whilst shared trees, for the most part, scale more favourably than source-rooted trees, the designers of PIM wanted a receiver to have the choice to receive data over a shortest-path tree, thereby optimizing delay. The trade-off in doing so is between routers keeping less state on a shared tree, or more state on a shortest-path tree. Also, as the number of shortest-path trees grow to a particular source, more overall bandwidth is consumed by the sum of the shortest-path trees than for a single shared tree.

As of writing, PIM is 'ongoing work', and there remain issues which still need to be resolved, for example, what are the criteria for switching between a shared tree and a shortest-path tree? How is such a switch instrumented? Furthermore, the PIM protocol is considerably more complex than existing IP multicast protocols.

The shared tree concept of multicasting in PIM is based on the *Core Based Tree (CBT)* multicast [7] approach, which we present in the next section. However, there is one outstanding difference between PIM and CBT: the CBT protocol adopts the "hard-state" approach to tree building and maintenance, whereas PIM sparse-mode adopts the "soft-state" approach.

4 The Core Based Tree (CBT) Multicast Architecture

The *Core Based Tree (CBT)* multicast architecture differs quite considerably from the existing IP multicast architecture in that it utilizes a single, shared delivery tree that spans a group's receivers. CBT takes full advantage of various aspects of the existing multicast infrastructure, such as *class-D* IP addresses, used for identifying multicast groups, and the *Internet Group Management Protocol (IGMP)*, which is used by multicast routers to establish group member presence of directly-connected subnetworks.

We have extended IGMP to reduce *leave latency* – the time between the last claim to a group on a particular subnet being relinquished, and the time group traffic is no longer forwarded onto that subnet. Our improvements to IGMP prompted the recent development and release of IGMP version 2 (IGMPv2).

4.1 Extending Multicast's Existing Properties

Deering, in [11], suggested that several important properties, originally conceived for the LAN multicast environment, be extended as desirable properties for internetwork multicasting. These include: *host group model conformance, high probability of delivery of multicasts*, and *low delay.* We propose extending these properties further, given that the Internet is ever increasing in size [4] and heterogeneity, and given the fact that multicast is becoming increasingly popular on a global scale:

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

- *Routing algorithm independence.* What we mean by routing algorithm independence is an independence from unicast protocol-specific features.

  Routing algorithm independence is a "double-edged sword" – the extent to which it is advantageous depends on the context from which it is viewed: on the one hand it simplifies multicast routing across heterogeneous domain boundaries, and it allows for the independent evolution of both unicast and multicast algorithms. On the other hand, it is unclear whether multicast routing should follow the policies specified by unicast routing (i.e. should there be separate routing policies for multicast?). Furthermore, implementation is made considerably more complex [29].

4.2 Existing Multicast Architecture and Some of its Shortcomings

DVMRP and M-OSPF comprise the protocols that are based on the existing multicast architecture. This architecture builds *source based*, shortest-path multicast delivery trees between a sender's source *subnetwork* and the corresponding group receivers.

The implication of a multicast architecture based on source-rooted trees is one of *scalability.* We will discuss the scalability of each of four multicast algorithms in section 9, but it is worth mentioning briefly here, why the source-based architecture does not offer very favourable scaling characteristics for wide-area multicasting.

DVMRP, based on the distance-vector algorithm [12], periodically broadcasts packets everywhere when pruned branches of the multicast tree timeout. If there are no receivers at the leaves of the tree, new prunes will be propogated up-tree. Thus, this incurs overhead, i.e. the storage of prune state, on routers that are *not* on the multicast tree.

M-OSPF can only be used in domains running OSPF [25]. M-OSPF broadcasts changes in group membership on a particular link, throughout the domain of operation, since all M-OSPF routers have a complete topology map of the location of group members. This is necessary for the Dijkstra computations, performed by each router, which compute the multicast tree from the perspective of the active source. It should be obvious that the storage of global membership information, as well as the overhead of Dijkstra computations, does not scale to *internetwork-wide* multicasting.

Whilst source-based, data driven schemes like those above do not have very good scaling characteristics in terms of state maintenance or processing costs, they do exhibit good robustness properties. Conversely, shared tree schemes rely on a small set of core routers to maintain connectivity between all senders and receivers.

4.3 CBT - The New Architecture

4.3.1 Architectural Overview

The *Core Based Tree* multicast architecture involves constructing a *single* delivery tree that is shared by a group's members. Multicast data is *sent* and *received* over the same delivery tree, irrespective of the source.

The idea of core based trees for multicasting was derived from Wall's work on broadcasting and selective-broadcasting [34]. However, the similarity in his work and ours ends in the utilization of a shared tree. We provide on overview of the primary differences between Wall's work and ours in section 8.

A core-based tree involves having a single node, or router, which acts as a *core* of the tree (with additional cores for robustness), 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[6] between the nodes on the tree, since multicasts vary in nature, and correspondingly, so should the form of a group's delivery tree. CBT is unique in that it allows the multicast tree to be built to reflect the nature of the application.

A core based tree is shown in figure 1. Only one core is shown for demonstration purposes. The dotted arrowed lines indicate how a CBT multicast packet, sent from a non-member sender, spans a CBT tree.

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

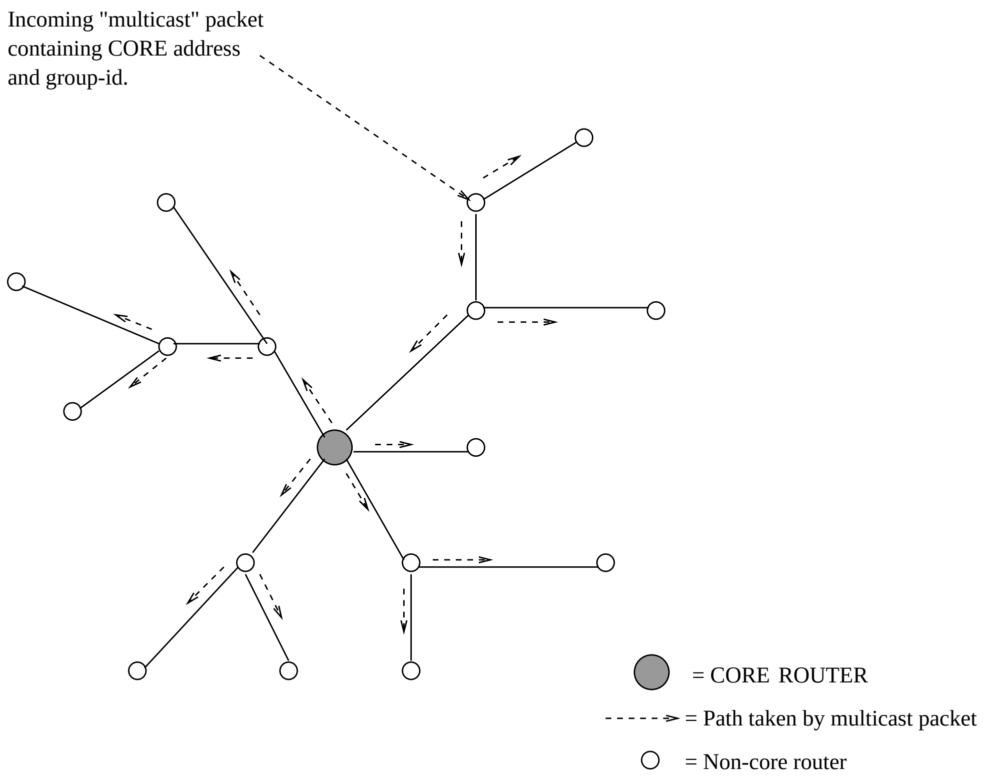
Incoming "multicast" packet containing CORE address and group-id.
= CORE ROUTER
= Path taken by multicast packet
= Non-core router

Figure 1: A CBT multicast delivery tree

Figure 2 illustrates a multi-core CBT tree. The *core tree*, i.e. tree backbone, is bold line connecting each of the core routers. Note that both core and non-core routers typically make up the core tree.

The CBT protocol is built so as to reflect the CBT architecture, just described. This architecture allows for the enhancement of the *scalability* of the multicast algorithm, particularly with respect to group-specific state maintained in the network. This is of particular interest when 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).

The reason it is possible to make such an improvement in the scaling factor of shared trees is because the forwarding of multicasts over a shared tree does not depend on the source of those multicasts, unlike source-based schemes. Hence their requirement to maintain source-specific group information. We contrast the different types of multicast tree, based on various criteria, in section 9.

Therefore, 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 (i.e. group) whatsoever in order to forward CBT multicasts, since packets from non-member senders are *unicast* towards the core. This *two-phase* routing approach is unique to the CBT

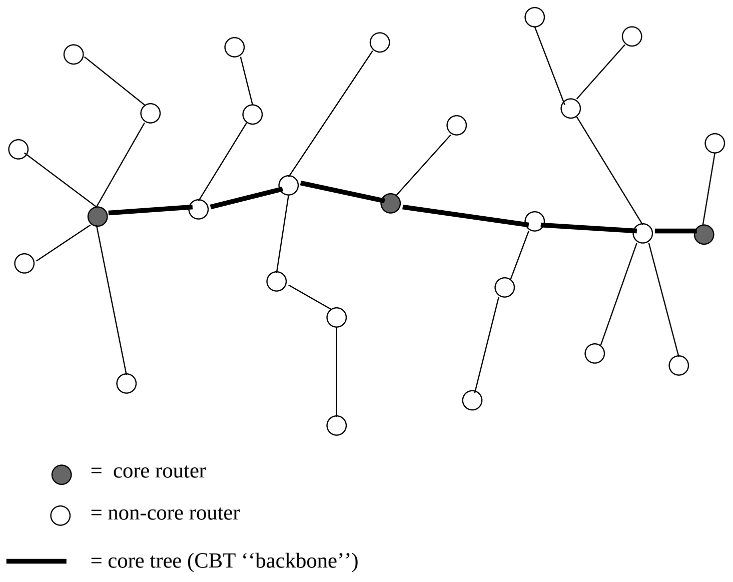

Figure 2: A Multi-Core CBT tree

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. This is an example of *anycasting* [24]. We believe that CBT offers a good solution to anycasting, which the author discusses in [1].

4.3.2 Architectural Justification

As we discussed in section 3.1, Wall's work involved investigating low-delay approaches to broadcast and selective broadcast. With regards to his *centre-based forwarding* trees, 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 [19]. We summarize these results in section 6.

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 6 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 (the strategic placement of cores facilitates the spreading of the load) – something that would not seem possible with 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.

5 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 significant 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[7]. 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 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.

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

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 [19].

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 implement some very simple heuristics to address some of these questions, but these may not be appropriate for large-scale implementation of CBT.

We conclude in section 7 that most aspects of core management are topics of further research.

6 Simulation Results

Three independent simulations have been carried out recently comparing and contrasting shared trees with shortest-path trees. We discuss each simulation separately:

1. Wei [19] recently investigated the trade-offs between shared tree types and shortest-path trees. Their performance was evaluated according to the following criteria: path length, link cost, and traffic concentration.

   Wei enumerated the parameters that can affect the performance of a distribution tree, namely: proportion of long links and short links in a graph (thus influencing delay and cost), graph node degree, group size, number of active senders, distribution of senders and receivers, and graph size [19].

   We will confine our discussion to the experiments and results observed for shortest-path trees (SPTs), and *delay-optimal* core based trees (CBTs), i.e. CBTs whose centre node is chosen so the resulting tree has mimimal maximum delay or minimum average delay among all members/senders (in the experiments, senders are assumed to be members also).

   Most of Wei's simulations were run over random graphs of two different sizes: 50 node graphs, and 200 node graphs. In each, the average node degree was approximately 4. A randomly selected multicast group was inserted into graph, and the corresponding trees (CBT and SPT) computed. The effect of group size on *delay* and *cost* was measured for each type of tree, with group size ranging between 5 and 25.

   In the 50-node graph, the ratio of *maximum delay* of a CBT to an SPT was shown to be around 1.2. This remained fairly constant for group sizes between 5 and 25. The *cost* ratio for the same group size range was shown to remain fairly constant at around 0.9. In the 200-node graph simulations, group size ranged from 20 to 80. The results were found to be similar to the 50-node graph.

To measure traffic concentration, 50-node graphs were used, with 300 fixed-sized randomly placed multicast groups, each group with a small, fixed sender population, also randomly selected. The graphs' average node degree varied between 3 and 8.

The results showed that, as the node degree increases, the maximum link load of a SPT decreases. This is as a direct result of the presence of a more redundant topology, giving rise to a greater number of paths between any pair of nodes. CBT's maximum link loads were shown to have remained fairly stable as the average degree node increased.

The conclusion to be drawn from the traffic concentration experiment is that in networks with low connectivity (such as the MBONE), CBTs will exhibit similar link utilization properties to SPTs. Wei concluded overall that "CBTs are not optimal for everyone, and overall, they are also not bad for everyone either."

2. The following simulation results take into account the presence of asymmetric links (this required slight modification of the CBT tree-building algorithm), and implement a simple *centre-location mechanism* [3]. The other fundamental difference in these simulations is that *a priori* knowledge of participants' locations is required for the purpose of the core placement mechanism.

   The main conclusion drawn from these simulations is that CBTs are lower cost than source-specific trees for many concurrent senders, with only a slight increase in the average path length. The results also show that source-specific trees "are more economical then centre-specific trees when the number of concurrent senders is large and when the network is highly interconnected" [3]. It is also implied that the centre-location mechanism assists in the generation of low cost centre-specific trees.

3. A PIM-CBT comparison has also been carried out very recently [28], partly to address the question of whether PIM is a superset of CBT. A summary of these results follows:

   - These results showed that PIM SPTs improved the delay of CBTs by 5-20%.
   - PIM SM (shared tree) consistently shows much longer delays ($>$ 50%) than CBT.
   - PIM SM (with S,G state to RP) is identical in delay to CBT.
   - Control packet overhead:
     - ◇ PIM SM with SPTs incurs about double the overhead of CBTs.
     - ◇ PIM SM shared tree mode incurs about the same overhead as CBT.
   - PIM and CBT join times were shown to be about equal.
   - PIM SM with SPTs can involve up to a 50% increase in join latency.

   Regarding the question: is PIM a superset of CBT? It would seem that PIM is a superset of CBT, but results show that there are distinct advantages to using CBT.

   The conclusions drawn from these results are as follows:

- the end-to-end delay was on average 10% lower with PIM SPTs than the CBT delay.
- in shared-tree mode, there is no advantage to using PIM over CBT.
- in terms of bandwidth utilization (overhead), both protocols were shown to be about equal.

7 Core Placement and Management

Core placement and management are other issues that may be seen as disadvantageous to the CBT approach. Whilst the author has adopted only simple heuristics for core placement, i.e. placement of a group's cores is "by hand" based on the topological distribution of group membership at group initiation time, our simple core placement strategy is not always unfavourable, the prime example being for single-sender "broadcasts". IETF audio/video-casts [5] fall under this category, and such "broadcasts" are not uncommon. A beneficial side-effect of our simple heuristic is that it involves no protocol overhead whilst being a satisfactory mechanism for certain applications.

Core management, which encompasses all aspects of core selection and *dynamic* placement strategies subsequent to group set-up, is a much more complex problem which will require the development of new algorithms dedicated to these purposes. Indeed, Doar [14] showed that dynamically changing the form of a multicast tree is usually not worthwhile for dynamic multicast groups in terms of efficiency[8] benefits, since any benefit is likely to be short-lived. His simulations showed that, by using a very simple heuristic for "patching" new receivers onto a distribution tree, created initially between a known set of receivers, the inefficiency of the resulting tree differed little from that of the initial tree[9].

In summary, core management is likely to be complex, and the benefits that can be gained from it are often relatively minor. The integration of core management techniques would also involve considerably more protocol overhead to CBT itself. For these reasons, we will not discuss core management further. However, core placement and management strategies are important topics of research.

8 How Wall's Trees Differ from CBT Trees

Wall's approach to tree construction is considerably different to our approach to building core based trees. This can be seen by comparing Wall's algorithms in [34], and the CBT protocol presented in [1]. We summarize the essential differences between Wall's trees and CBT trees below:

- Wall's tree building is inefficient for two reasons. First, it requires that *all* nodes in a network participate in tree building. The process of tree building requires each node to run two algorithms separately – one algorithm involves a node calculating its *criterion value*, used for establishing a node's eligibility for the *centre* node. The second algorithm is the *minimization protocol*, used for *electing* the centre node and delivery tree.

[8] Doar measured *efficiency* as "the ratio of the cost of the resulting spanning tree to the cost of the minimal Steiner tree."

[9] Doar's work was done in the context of connection-oriented networks, such as ATM.

	conn-oriented	**tree state**	**explicit tree maintenance**	**no. of cores**	**tree adapts to route change s**	**tree adapts to failures**
Source trees	*N*	$O(S \times G)$	*N*	*n/a*	*Y*	*Y*
Wall trees	*N*	$O(G)$	*N*	*1*	*N*	*N*
CBT trees	*Y*	$O(G)$	*Y*	> 1	*configurable*	*Y*

Table 1: Comparison of Tree Types

Second, as a consequence of all nodes participating in the minimization protocol, all possible trees are built initially, and all but one (the best) are "killed off". Potentially, this results in large volumes of traffic being generated.

- Once a tree is built, there are no explicit tree maintenance mechanisms in Wall's trees, to detect such things as node failure.
- Node failure and re-start are not handled gracefully by Wall trees.
- Wall's trees specify only *one* centre node, resulting in very poor fault tolerance.
- For selective-broadcasting (analogous to multicasting), Wall trees assume that a sender is a group member (analogous to a *closed* multicast group).

The table below offers a comparison of the different tree types:

9 Multicast Scalability

The fundamental premise that motivated this work was multicast *scalability.* We have already emphasised how multicasting has grown over the last two years, since the first IETF[10] meeting in Boston audio- and videocast various meetings over a relatively sparse MBONE, as it was then [5]. A graph illustrating MBONE growth over the last two years is given in figure 3.

In this section we evaluate four multicast algorithms, *distance-vector*, *link-state*, *CBT*, and *PIM*, in terms of various factors. These factors most influence a multicast algorithm's scalability:

- group state information maintained in the network.
- bandwidth consumption/link utilization.
- processing costs.

[10] Internet Engineering Task Force.

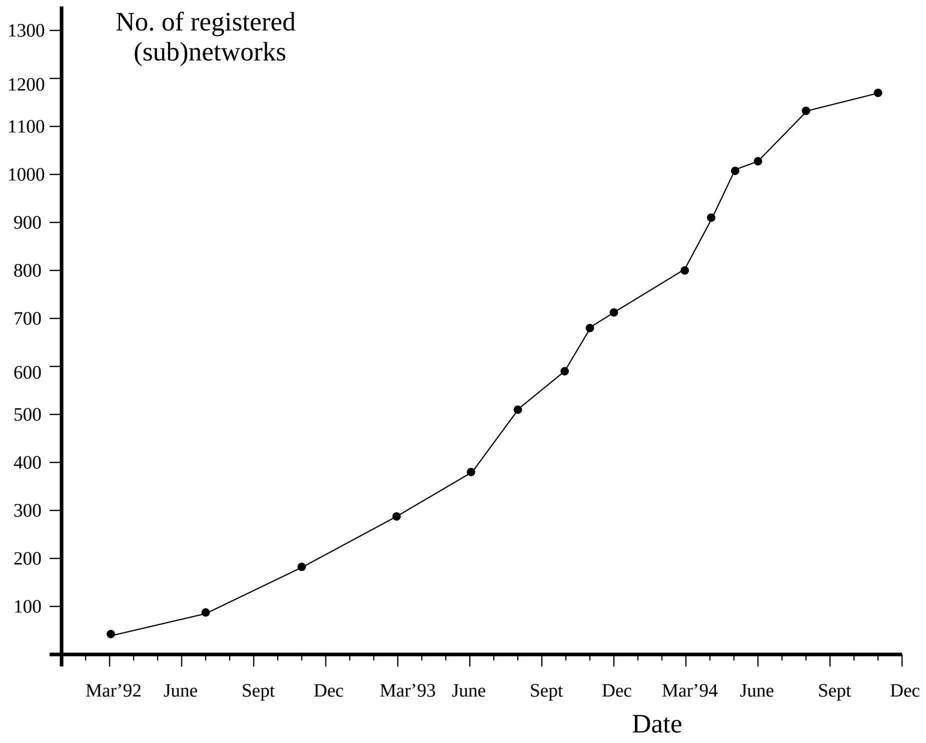

Figure 3: The Growth of the MBONE between March 1992 and November 1994

We now analyse each of the above-mentioned algorithms based on each of these scalability criteria.

9.1 The Scalability of the Distance-Vector Multicast Algorithm

DVMRP [8] is the only implementation of the distance-vector multicast algorithm [12] to date, and so we will use it as an example for our analysis.

9.1.1 Group State Information

Group state maintained in the network by the Distance-Vector algorithm falls into two categories:

- (source, group) state for *active* sources for routers that fall *within* an active distribution tree. Let's call this state "tree state".
- (source, group) state for routers that *do not* fall within an active distribution tree. This is called "prune state" [12].

Prune messages need to be recorded both by the transmitting station, as well as the receiving station(s). Each prune message consumes approximately 20 bytes of storage, if we assume 4-byte IP addresses [12].

The actual amount of "tree state" and "prune state" stored by routers on- and off-tree respectively, is dependent on the number of active multicast sources, and the distribution of receivers in the network. State is only cached by routers for *active* trees. After a period of inactivity, cached state times out. For this reason, it is called the *soft-state* approach to storage.

The amount of prune state stored very much depends on how group members are distributed throughout the (inter)network. For instance, a group may be quite sparse, but evenly distributed, the result being that overall fewer prunes would be generated. Alternatively, group membership may be sparse and concentrated in "pockets" dotted around the (inter)network, in which case overall more prunes will be generated as a result of the numerous "pockets" not interested in multicast reception. If however, a group is densely populated and evenly distributed in the (inter)network, very few prunes will be generated at all.

Hence, the amount of tree state and prune state is very much dependent on group membership dynamics, such as group density, group dynamicity (i.e. how frequently members of a group appear and disappear), group distribution, and the number of active sources per group. With respect to group-specific state maintained in the network, the distance-vector multicast algorithm scales $O(S \times N)$, where S is the number of *active* sources.

More precisely, the amount of prune state off-tree routers must maintain is bounded by the number of subordinate routers it has [12], and further dependent on the lifetime of prune messages. If prune lifetime is short, prune state may time out even though there are still no group members downstream. This will result in subsequent data packets being unnecessarily forwarded over an "empty" branch, the consequence being that valuable resources are consumed unnecessarily.

If prune lifetime is long, prune state may be maintained long after a source has ceased transmitting, and so this prune state is consuming off-tree routers' resources unnecessarily.

Thus, the trade-off, dependent on prune message lifetime, is between storage consumed by off-tree routers, and bandwidth consumed along an "empty" branch. Perhaps "trade-off" is the wrong term to use here, since there is nothing being gained. We should probably instead ask, 'which is the lesser of the two evils?'.

9.1.2 Bandwidth Consumption

The Distance-Vector algorithm results in bandwidth being consumed unnecessarily in those parts of the network that are *not* part of a multicast delivery tree.

As we mentioned in the previous section, unnecessary bandwidth may be consumed if the lifetime of prune messages is short. The total bandwidth wasted in the (inter)network, again, is dependent on group membership dynamics, such as its density and distribution. If we assume that most wide-area multicast groups are quite sparsely distributed, and, for the most part, consist of O(tens of members)[11], the amount of bandwidth wasted due to a short prune lifetime would be quite substantial.

The Distance-Vector algorithm uses a mechanism for quickly "grafting" back a previously pruned branch of a group's tree. This is done by means of a "graft" message, which is acknowledged by the receiving upstream router (i.e. the next-hop router towards the active source). Any router receiving a graft for some active (source, group) pair must itself send a graft if it had sent a prune upstream for the same (source, group) pair.

Graft messages are not stored, but serve to cancel out a previously-sent prune for some (source, group) pair.

[11] This is the size of *most* groups based on the past few years' experience of wide-area multicasting.

The total bandwidth consumed by graft messages, and their corresponding acknowledgements, for some active source, is very much dependent on a group's distribution throughout the (inter)network and a group's dynamicity. For example, if a group is widely and evenly distributed across the network, there will be fewer pruned branches, and so grafts will be unnecessary. On the other hand, if a group is only sparsely distributed, and a group is highly dynamic, many grafts will be generated.

From the practical experience gained over the last few years, for the most part, groups are dynamic, but the rate with which new receivers join a group tends to be gradual rather than rapid. However, there are may well be random periods of join burstiness, for example, just after group initiation.

These interpretations are extrapolated from practical experience over a long period, but there exist no statistics to substantiate our claims about groups' dynamicity, or indeed any other group characteristics. Hence, we cannot offer more precise figures on total *off-tree* bandwidth consumed, other than to say that it is dependent on the membership dynamics and membership distribution of *active* groups.

9.1.3 Processing Costs

The processing costs of the distance-vector algorithm fall into two categories:

- the processing cost involved in the generation, reception, and interpretation of prune and graft messages [12].
- the processing cost of the distance-vector routing messages.

With regards to the former item, these costs are proportional to the number of prunes and grafts that are sent/received [12]. It should be clear from the previous section that this cost is dependent on various aspects of a group's characteristics.

As for the processing cost of distance-vector routing messages, it has been shown in experiments that routers running the distance-vector multicast algorithm (DVMRP) become highly unstable when the number of routes (subnetworks) handled reaches about 9000 [13].

This demonstrates a scaling drawback of multicast algorithms that rely on distance-vector routing messages explicitly for multicast. Clearly, this motivates the need for multicast capability to be deployed throughout the global internetwork infrastructure. Provided a multicast algorithm does not dependent on a particular underlying routing protocol type, such deployment would obviate the need for explicit multicast route exchange, and also allow both the multicast algorithm and underlying routing protocol to evolve independently, which is advantageous. Unfortunately, DVMRP has such underlying routing protocol dependencies.

9.2 The Scalability of the Link-State Multicast Algorithm

M-OSPF [26] is the only implementation of the Link-State Multicast algorithm to date, and we will therefore discuss the scalability of the link-state algorithm in the context of M-OSPF.

M-OSPF is primarily designed to operate within OSPF "areas" [25]. However, it can be used hierarchically to route multicasts between areas, as well as between different autonomous systems (ASs), albeit less efficiently than within a single area.

9.2.1 Group State Information

M-OSPF routers include group membership information as part of the *status* of each of their links, and, in accordance with the link-state paradigm, advertises this information to all other routers in the same domain by means of a special flooding protocol [30]. This "global" distribution and storage of group membership information is the primary scaling drawback of the link-state multicast algorithm, and the reason why it is not used for inter-domain multicasting [21].

When used hierarchically, M-OSPF routers that route between different levels of the hierarchy are termed *wild card receivers*, since they receive all multicast datagrams, irrespective of their destination [26]. In order to keep the size of the link-state database within bounds in the hierarchical case, border routers advertise their area's group membership information to inter-area routers (so-called *backbone routers*) in the form of *summary link-state advertisements (LSAs)*. This information, however, is not re-distributed by the backbone routers as is done in OSPF. Rather, they use the information received by default from non-backbone area border routers, to route multicasts between the areas they know have group membership [26].

9.2.2 Bandwidth Consumption

As we stated in the previous section, M-OSPF wild-card receivers, which are located at hierarchical boundaries, receive all multicast datagrams. The implication of this is that, when a particular group is confined to a single area, it is nevertheless forwarded to the inter-area wild-card receivers. Even though the receiving router(s) will discard such datagrams, they consume bandwidth unnecessarily. The amount of bandwidth wasted due to this inefficiency of M-OSPF would be quite considerable if a conference application was generating the data. Hence, in terms of bandwidth consumption, M-OSPF becomes quite inefficient when it is used hierarchically.

The bandwidth cost of disemminating group link-state updates is dependent on the frequency of the appearance and disappearance of group members on a subnetwork [12]. However, Deering observed that the cost of multicast link-state update traffic is likely to be insignificant because, for example, memberships tend to be long-lived. He nevertheless showed how routers can bound the overhead of LSAs by delaying updates for group disappearance and "piggybacking" them on other updates, as well as rate-limiting updates, for example, by restricting group appearance updates to once every N seconds [12].

When used within a single domain, i.e. non-hierarchically, the link-state multicast algorithm provides a multicast distribution tree without requiring explicit pruning, unlike DVMRP. This is the result of each router having complete topological group membership information. Each router uses Dijkstra's algorithm [31] to compute the multicast tree with respect to an active source. A side-effect of this is that the link-state algorithm can minimize network bandwidth consumed by packets that will not reach a particular receiver(s) because the packet's IP TTL value is too small. Such packets will not be forwarded by the router concerned [26].

9.2.3 Processing Costs

The most substantial cost to routers running the link-state multicast algorithm is the cost associated with Dijkstra computations. This cost scales $O(n \times log\ n)$. This is the other major factor limiting the link-state algorithm's applicability to wide-area multicasting [21].

M-OSPF calculates a separate path for each (source, group, Type of Service) tuple. Potentially, this can result in a significant burden to M-OSPF routers that fall within the corresponding tree(s) – routers not falling within a tree are not burdened with any calculations for that tree.

To reduce the burden on M-OSPF routers, calculations are performed "on demand", i.e. on receipt of the first multicast datagram for some new (source, group, Type of Service) tuple. The result of this calculation is then cached. This "on demand" strategy has the benefit of spreading calculations over time [26], resulting in a lesser "impact" for participating routers. However, in the presence of large numbers of active groups, M-OSPF routers connected to transit networks will be incurred considerable processing costs.

9.3 The Scalability of the PIM Architecture

The PIM architecture was designed "to efficiently establish distribution trees across wide-area internets" [21], given that many groups will be sparsely represented in the context of the wide-area. PIM's two modes of operation, *dense mode* and *sparse mode*, have very different scaling properties and characteristics. Dense mode involves data driven "flooding" which assumes that all downstream systems want to receive multicast data, with subsequent pruning by downstream sites not interested in the multicast traffic. This mode of operation is only acceptable in resource-rich envoronments, or where a group(s) is widely distributed, as is likely to be the case within a campus network. Sparse mode, on the other hand, tries to constrain the distribution tree to only those parts of the network interested in receiving the multicast traffic[12]. It involves receivers explictly joining a *shared* multicast distribution tree.

In our analysis of PIM's scaling characteristics we compare and contrast the scaling properties of each mode, as well as its overall scalability when the two modes interoperate to achieve multicasting across domain boundaries.

Before venturing further, it is interesting to note that scalability was not mentioned as one of the design objectives of the PIM architecture [21].

9.3.1 Group State Information

With regards to state maintenance, the fundamental difference between PIM's dense and sparse modes are as follows:

- in dense mode, a router's default behaviour is to forward a received multicast packet on all outgoing interfaces if the packet arrived on the reverse-path interface to the source, until such time as explicit prunes are triggered downstream, which "trim" the receiving router's outgoing interface list. This is the data driven "flooding" approach, referred to above.

[12]This of course, also includes routers and networks on the path to interested receivers.

- in sparse mode, routers connected to interested receivers must explicitly join the distribution tree, which initially, is a *shared* tree. Subsequently, such routers may switch to a shortest-path tree by sending a join towards an active source, and a prune for the same source over its shared tree interface.

The characteristics of group state creation are thus different in the two PIM modes: in dense mode, multicast traffic flow instantiates (source, group) pair information in routers, whereas in sparse mode, (source, group) state is created explicitly by interested receivers. In both cases, the state maintained is *soft state*, i.e. it times out in routers unless refreshed periodically.

More precisely, in terms of network state, dense mode scales $O(S \times G)$, where G is the number of *active* groups, and S is the number of *active* sources.

Sparse mode scales between $O(G)$ and $O(S \times G)$, again, where S and G are active. The reason sparse mode scales between this range is because receivers can either choose to remain on the shared, so-called *RP (rendezvous point)* tree, or prune themselves from the RP tree and switch to a shortest-path distribution tree. Shared trees, or group trees, scale $O(G)$. So, sparse mode starts out by scaling $O(G)$, and tends to $O(G \times S)$ as receivers move to shortest-path trees.

Whenever a receiver switches from an RP tree to a shortest-path tree, (source, group) *forwarding* state is instantiated along the shortest path from the receiver to the sender. *Prune* state is also maintained along the RP path for the active (source, group) pair. Hence the scaling factor of $S \times G$ as receivers move from the RP to SP distribution trees.

The most likely way PIM will be used to achieve *inter-domain* multicasting, is if PIM dense mode is used within domains, and PIM sparse mode is used to multicast between domains, as the diagram below illustrates...

The diagram illustrates dense mode PIM operating within domains, or clouds, with a sparse mode *shared tree* linking the clouds together.

Remember that, in sparse mode, receivers send joins *towards* an RP (or a sender if an SP tree is desired). So, for the inter-domain multicast scenario illustrated, the state at the border router of each domain scales $O(G_{local})$ – the number of inter-domain groups with local members [18]. However, if receivers within domains decide to switch to SP trees, more state will need to be maintained in the backbone while a source is active, so the scalability then tends to $O(S_{local} \times G_{local})$, where S_{local} is the number of active senders within a domain [18].

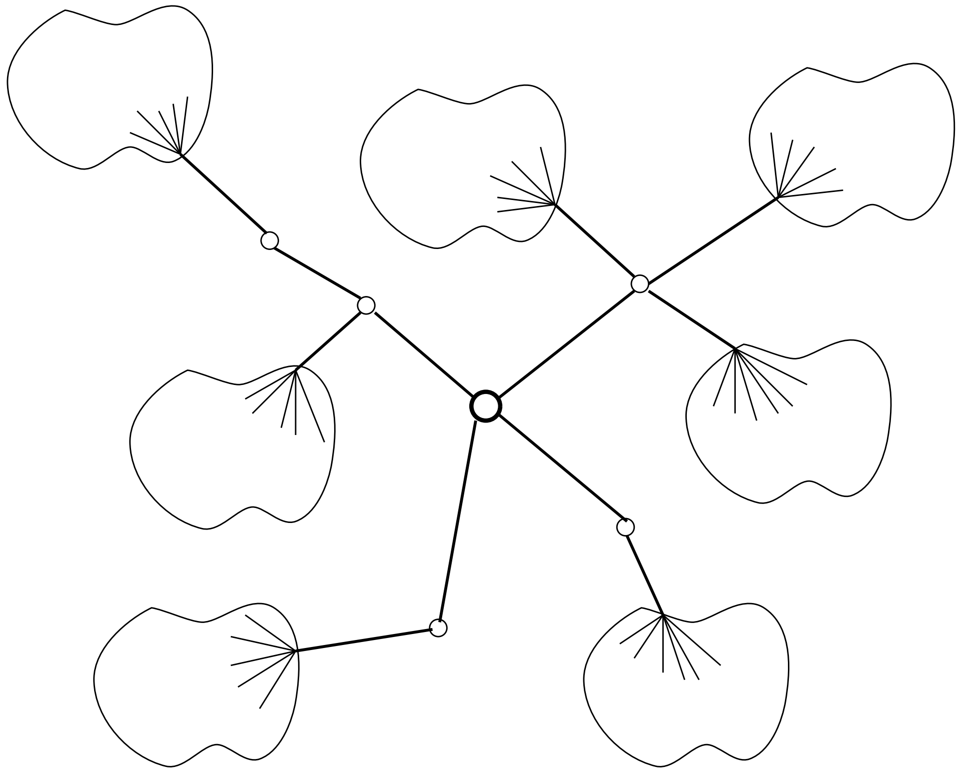

Figure 4: How PIM is used for Inter-Domain Multicasting

9.3.2 Bandwidth Consumption

In dense mode, the bandwidth consumed unnecessarily by the "flooding" of data packets is similar to that of DVMRP, and is the result of the periodic flushing of prune information. The extent to which this is significant therefore, depends on the distribution of group members, as we explained.

Besides prune and graft messages, PIM also introduces additional control message types, for example *joins*, and *RP-reachability* messages, which too require refreshing at fixed periodic intervals. This imposes additional bandwidth requirements on the architecture, which, in the presence of large numbers of active groups, will be quite substantial. For the PIM authors, ways of bounding and aggregating control traffic remain open issues. One suggestion on bounding the amount of control traffic [17] would be to administratively set a link limit on bandwidth for control traffic. However, it has been suggested that better responsiveness can be achieved by dividing so-called control bandwidth non-uniformly between active and inactive groups.

Like the CBT architecture, PIM sparse mode utilizes several *rendezvous points (RPs)* per group, for robustness purposes. Receivers join just one RP, but a sender must unicast its data to each one separately, from where it is disemminated to non-pruned RP receivers. However, for the case where there are no RP-interested receivers, data packets nevertheless get unicast to each of a group's RPs, wasting valuable bandwidth between the sender and each RP.

Finally, a consequence of the unicast protocol independence of dense-mode PIM is that there is no child-parent database, as calculated in DVMRP. This means that "some duplicate packets" [21] are unavoidable.

9.3.3 Processing Costs

The costs in PIM associated with generation, reception, and interpretation of prunes and grafts are the same as those for DVMRP (see above) [18].

Additionally, in sparse mode there is join traffic to consider. Using the inter-domain case (see figure 4) again as our example, with a sparse mode shared tree linking dense mode clouds, receivers' outbound join traffic scales $O(G_{local})$ – the number of inter-domain groups with local members [18].

The amount of inbound join traffic, generated as a result of external domain receivers wishing to switch to an SP distribution tree, is dependent on the number of local sources sending to inter-domain groups, and scales $O(S_{local})$ [18].

9.4 The Scalability of the CBT Architecture

The fundamental motivation behind the design of the CBT architecture was the ability to significantly improve multicast scalability. This improvement does not come without its cost, and in the case of CBT the primary disadvantage of the architecture is the potential for increased delay between two multicast receivers – a consequence of the absence of source-based shortest-path trees. We elaborated on this, and other potential disadvantages of the CBT architecture, in section 5.

9.4.1 Group State Information

As we have seen, traditional IP multicast schemes build *source-rooted* delivery trees. For non-member senders, as well as member senders, this means that (source, group) state information is maintained in the network from the point of source. Hence, there is no concept in traditional IP multicast schemes of *on-tree* and *off-tree* forwarding.

CBT however, uses a *two-phase* approach to routing multicasts from a non-member sender. In the first phase, a multicast data packet is *unicast* towards a core of the corresponding group tree and either encounters the tree first at that core, or encounters a router that is part of the tree on the unicast path to the core. At the point the data packet hits the tree, phase two of the multicast routing takes over, and the packet is disemminated over each *outgoing* tree interface, being *unicast* hop-by-hop to each neighbouring CBT router on the tree. The packet is only *multicast* (IP-style) by routers with directly-connected subnetworks with group member presence.

The advantage of this two-phase multicasting approach as far as scalability is concerned, is that routers in the unicast path between the non-member sender and the shared delivery tree need maintain no information whatsoever regarding the multicast group. Hence the concept of off- and on-tree multicast routing. Off-tree, no group state is required in the network.

With regards to *on-tree* routers, they must maintain a list of tree interfaces associated with each group. Unlike DVMRP and PIM dense-mode, forwarding is done irrespective of the packet's IP *source* address, and therefore

CBT routers do *not* need to maintain source-specific group state.

To summarize, we saw that source-rooted schemes scale $O(S \times G)$, where S is either the number of *active* source *subnetworks*, or an aggregate thereof, depending on whether hierarchical multicast routing is implemented. CBT eliminates the source scaling factor, and therefore scales $O(G)$ – the number of groups present in the (inter)network.

In the absence of *group aggregation*, for which there are no known mechanisms due to the lack of structure in multicast addresses, $O(G)$ scaling is the best that can be achieved.

9.4.2 Bandwidth Consumption

No bandwidth is wasted due to pruning, and its associated inefficiencies, or grafting, since the CBT architecture does send multicast packets to parts of the network not interested in them.

CBT is not data-driven, i.e. forwarding state is not instantiated in routers through the flow of multicast traffic. Rather, a CBT tree must be explicitly built, and receivers must join and leave a group tree explicitly. This implies the need for control and maintenance traffic.

The traffic associated with tree *building* is short-lived and sporadic. The most intense bandwidth consumer in CBT, apart from data itself, are the periodic "keepalives" between adjacent CBT routers, which serve to identify tree breakages when and if they occur. Each "keepalive", and its associated reply, is 56 bytes long. This comprises a 24 byte Ethernet header[13], a 20 byte IP header, and a 12 byte CBT control message header.

It is probably fair to assume that around the periphery of the internetwork, branches of different CBT trees will tend to be shared less than around the centre. Therefore, the most significant consumption of bandwidth on any link due to CBT maintenance traffic will be around the centre of the network, where it will increase linearly with the number of wide-area groups using CBT multicast. The bandwidth resource requirement on any link can be alleviated by reducing the frequency of these control messages. Hence, their frequency is configurable.

Like the PIM authors, the CBT authors are looking into ways of aggregating maintenance traffic. At the time of writing, this remains ongoing work.

Data packets spanning CBT *tree branches*, i.e. in transit between two adjacent CBT routers for a particular group, carry a CBT header immediately behind the IP header (see [1]). For the benefit of our discussion, if we assume for the moment that data packets are not part of an explicit *flow*, and are not subject to *security* checks such as authentication, then we can omit the **flow-identifier** and **security fields**, the latter of which can vary in length depending on the implementation, from our bandwidth overhead calculation.

We focus our attention on *audio* data, since these packets are typically small. *Video* data packets can be as large as individual video frames, or multiples thereof. Packaging video frames in this way is more efficient. Using audio packets as an example enables us to see a worst-case percentage increases in packet sizes as a result of the CBT header. We also calculate the extra bandwidth consumed by packets carrying a CBT header.

Looking at the three currently available audio formats, *pcm, pcm2* and *pcm4*, these encode into 195-, 355-, and 680-byte chunks of data, respectively.

[13] Using Ethernet as an example of a typical subnetwork technology.

Tree branches may span a variety of media, from 10 Mbps Ethernet, to 64 kbps point-to-point links. Given that audio is carried as UDP data, each audio data packet comprises a link layer header (let's assume that this is 24 bytes, as it is for Ethernet), a 20-byte IP header, and an 8-byte UDP header, the table below shows the percentage increase in packet size as a result of the CBT header for each of the three audio packet formats. Also shown is the additional bandwidth consumed for the specified media:

	audio format	% increase in pkt size	% extra b/w consumed
Ethernet	pcm	8.09	0.0016
	pcm2	4.91	
	pcm4	2.73	
T1 line (1.544 Mbps)	pcm	8.09	0.01
	pcm2	4.91	
	pcm4	2.73	
64 kbps link	pcm	8.09	0.25
	pcm2	4.91	
	pcm4	2.73	

Table 2: Overhead for various media due to the presence of a CBT header.

As can be seen from the table, the extra bandwidth consumed as a result of the increase in packet size due to the presence of a CBT header is well under 1% in all cases.

If we now assume the presence of flows and packet authentication, the CBT header becomes somewhat larger. How large depends on the cryptographic technique employed, key size etc., so exact packet size for this case is implementation dependent. If we assume that each packet may become 30% or 40% larger as a result of authentication requirements, this still does not contribute significantly to bandwidth consumption.

9.4.3 Processing Costs

The processing costs in CBT only become significant as the number of groups traversing a particular router increases. We can divide these costs into two: those associated with maintenance traffic, and those associated with data packet forwarding.

Our arguments with regards to these are the same as in the previous section, but we must now look at how they affect routers themselves.

For "keepalives", processing involves looking up a record in memory, indexed on group address, matching the source address, updating a timer, and generating and sending a reply. The burden imposed on a router is dependent on the frequency of "keepalives", which is configurable, and the number of group trees traversing a particular router. In order to maintain a bound on the processing burden, as the number of groups increases, the frequency of the "keepalives" should be decreased.

The processing cost associated with data packet forwarding is dependent on the number of group trees traversing a router. For "hot spot" core routers, this may well be considerable. A way to control this problem is to put an upper bound on the number of groups for which any single router can become a core. This would improve network link utilization and reduce the processing burden on individual routers. Furthermore, since no RPF lookup is required as is the case with DVMRP and PIM dense mode, forwarding an individual packet is slightly "cheaper" with CBT.

Finally, as with DVMRP, because of the absence of multicast capability in the global network infrastructure, CBT relies on CBT-capable routers participating in neighbour-to-neighbour exchange of *distance-vector* multicast routing packets. As we mentioned for DVMRP, typical implementations of the RIP-like routing deamon impose an upper bound on the number of routes that can be successfully handled [13].

10 Summary

The focal point of this paper has been the presentation of a new multicast architecture and protocol for datagram networks.

The primary motivating factor behind the design of this new architecture was *multicast scalability*. In this paper we have closely analysed the scalability of the most prevalent multicast algorithms in the Internet today. *Core Based Trees (CBTs)* offer the most favourable scaling characteristics for the typical case where there are *some* senders within or outside a group of receivers.

Imposing a shared delivery tree between a group of multicast receivers has meant that it has been necessary to sacrifice shortest-paths between a sender and each receiver, although a shortest-path may be present between a sender and some receivers on any tree. Whilst, CBTs offer no guarantees that this will be the case, very recent simulation work has been done which showed that, with a given centre-location mechanism, CBTs are lower cost than source based trees for many concurrent senders, with "only a modest increase in the average path length" [3].

The overall conclusion we draw from this work is: CBT may not be suitable for all multicast applications, but it will be satisfactory for many. Like the variety of unicast routing protocols, CBT has its place and applicability in the Internet.

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] D. R. Boggs. Internet Broadcasting. *Xerox Parc Technical Report CSL-83-3*, October 1983.

[3] S. Shukla E. Boyer and J. Klinker. Multicast Tree Construction in Network Topologies with Asymmetric Link Loads. *Naval Postgraduate School TR: NPS-EC-94-012*, December 1994.

[4] K. Claffy H-W. Braun and G. C. Polyzos. Tracking Long-Term Growth of the NSFNET. *Communications of the ACM*, 37(8):34–45, August 1994.

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

[6] D. E. Comer. *Internetworking with TCP/IP, Volume 1.* Prentice-Hall, 1991.

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

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

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

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

[11] 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.

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

[13] Steve Deering. Private communication. September 1994.

[14] M. Doar and I. Leslie. How Bad is Naive Multicast Routing? In *Conference Proceedings of IEEE Infocom'93*, pages 82–89. IEEE, 1993.

[15] H. Eriksson. MBONE: The Multicast Backbone. *Communications of the ACM*, 37(8):54–60, August 1994.

[16] S. Armstrong A. Freier and K. Marzullo. RFC 1301, Multicast Transport Protocol. *SRI Network Information Center*, February 1992.

[17] V. Jacobson. IDMR wg presentation at IETF. July 1994.

[18] V. Jacobson. Slide Presentation. *IDMR working group meeting*, July 1994.

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

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

[21] S. Deering D. Farinacci V. Jacobson C. Lui and L. Wei. An Architecture for Wide-Area Multicast Routing. In *Conference Proceedings of ACM Sigcomm'94*, pages 126–135. ACM SIGCOMM, September 1994.

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

[23] C. Bowman P. Danzig U. Manber and M. Schwartz. Scalable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM*, 37(8):98–107, August 1994.

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

[25] J. Moy. RFC 1247, OSPF Version 2. *SRI Network Information Center*, August 1991.

[26] J. Moy. Multicast Routing Extensions for OSPF. *Communications of the ACM*, 37(8):61–66, August 1994.

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

[28] R. Voigt (Naval Postgraduate School). IDMR wg presentation at IETF. April 1995.

[29] Tom Pusateri (NetEdge Inc.). Private communication. April 1995.

[30] R. Perlman. *Interconnections: Bridges and Routers.* Addison-Wesley Professional Computing Series, 1992.

[31] R. Perlman. *Interconnections: Bridges and Routers, chapter 9 on Routing Algorithm Issues.* Addison-Wesley Professional Computing Series, 1992.

[32] B. Rajagopalan and P. McKinley. A Token-Based Protocol for Reliable, Ordered Multicast Communication. In *Eight Syposium on Reliable Distributed Systems*, pages 84–93, October 1989.

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

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

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