A New Approach
to Multicast Communication
in a Datagram Internetwork

Anthony J. Ballardie

May, 1995

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY COLLEGE LONDON
UNIVERSITY OF LONDON

A thesis submitted to the University of London
for the degree of
Doctor of Philosophy (Ph.D.)

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 thesis 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.

Our first and most significant contribution involves the presentation of a new multicast architecture and protocol, 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.

Our other most significant contribution is the security architecture that is *integral* in our new multicast proposal. It provides *authentication* of routers that make up a multicast delivery tree, and end-system subscribers. It also doubles in providing a solution to the well-known *multicast key distribution* problem.

We also provide an analysis of the security threats to multicast communication, and propose various methods to counter those threats.

One other contribution is presented: a proposal for *ATM multicasting*. This proposal is not a contribution of the author, but is based heavily on the author's multicast architecture and protocol. We therefore consider it appropriate to include an overview of this work here. As part of our discussion on ATM multicast, we also provide an explanation of why traditional IP multicast schemes are not suited to the ATM paradigm. We proceed to show how our new multicast architecture complements the ATM model.

The quintessential question this thesis 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]Examples include: RIP, OSPF, IS-IS, IGRP, EGP, BGP, etc.

Acknowledgements

In writing this page I can at last say I have come to the end of a long road I set out on, not just 3 years ago when I started this PhD, but many years before that. But that's a long story.

During the past 3 years I have been privileged to work closely with some of the best and most reputable people in the field of networking. They always set the highest standards to which I had to become accustomed, and for which I am extremely grateful.

Most especially, I would like to thank Jon Crowcroft, my supervisor, and Paul Francis, currently at NTT Japan. Jon not only provided me with direction and focus, but also gave me the opportunity to be his student. In retrospect, I would not have chosen any other. Paul also played a major role in this PhD. Without the many brainstorming sessions we had early on, this work would not have come about – the idea of multicasting using a shared tree was his. I think I have partly repaid him in that he is now a better tennis player, but this is a debt that will be long outstanding.

Paul and Jon also secured funding for the duration of my PhD from Bell Communications Research (Bellcore), U.S.A. I would like to thank Bellcore, not only for their financial support, which made student life a bit easier, but also for allowing me to visit and become part of their networking team on several occasions. Special thanks to Ramesh Govindan who assisted, and provided many insights, in my initial CBT implementation, and Liang Wu and Bruce Davie for all their work behind the scenes which made my visits possible.

Steve Deering (Xerox PARC) should not go without mention here. Without him realising it, his knowledge and expertise have, in part, motivated me during my PhD. This work has also benefitted from his insights and constructive criticism as it has progressed through the IDMR working group of the IETF.

My time at UCL wouldn't have been half as enjoyable if not for my PhD student friends and colleagues here. Thanks for all the good times in the pub and the Nepalese restaurant. I hope we continue the tradition long hereafter.

Special thanks too to Alex Coddington for proof-reading all of this – it makes a lot more sense now!

And I mustn't forget to mention my dear friends, Sarah and Dave. I can't thank you enough for letting me share your home, for all the good laughs, and much more.

Last, but by no means least, thanks to Mum and Dad for all your moral support, and for believing in me. This thesis is dedicated to you.

To

Mum and Dad,

Thank you.

Author's Notes

- **Statement of Work.**

  In light of numerous co-authored publications related to the work presented in this thesis it is maybe necessary to make clear the work attributed by the author.

  All of the work presented in this thesis is the sole contribution of the author, Tony Ballardie.

  Jon Crowcroft and Paul Francis appear either together or separately as secondary co-authors on some of the related publications. Paul Francis appeared on some of the early documents since the *idea* of using a shared tree for multicasting was his. He also provided valuable review feedback with regards to those works, and therefore it was considered appropriate that he should appear as a secondary co-author.

  Jon Crowcroft appears on most publications due to UCL CS departmental policy that a student must include the supervisor on *all* conference and journal publications.

  The term "we", used throughout, is simply a writing convention.

- **Glossary**

  A glossary of important and commonly-used terms is provided near the back of this thesis due to the large number of acronyms used in the field of networking. However, for *most* of the acronyms used we provide its definition the first time it appears.

- **Reference List**

  The reference list provided at the end of this thesis makes numerous references to "work in progress". These are *Internet-drafts* – working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. These documents are valid for a maximum of six months, after or during which they must be updated, replaced, or obsoleted. It is not permitted to cite an Internet-draft other than as a "working draft" or "work in progress".

  If you wish to retrieve any such document listed in the reference list, it is recommended you send e-mail to:

  internet-drafts@cnri.reston.va.us

  supplying the information (author(s), title, date etc.) provided in the reference list.

Contents

List of Figures

List of Tables

Chapter 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 [62] and ATM [23] specify multicast as part of their service.

The multicast backbone, or MBONE [32], 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 [16], replicated database updating and querying, software update distribution, stock market information services, and more recently, resource discovery [11, 88]. In general, multimedia communication [101, 57] 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 thesis focusses on one such investigation, and the subsequent development of a new multicast architecture and protocol for datagram networks. This introductory chapter provides a background to this new architecture, and follows on with an overview of this thesis.

1.1 Background

The IP Internet is a "network of networks", and essentially forms a tree topology that is hierarchically structured. There are four hierarchical elements to the Internet, which, ordered from the top-level down, are: *provider network*, *subscriber network*, *subnetwork*, and *end-system (host)* [36].

The top-level provider networks are often referred to as *backbone networks*, since they are used to interconnect networks lower down in the hierarchy, for example, campus or corporate networks.

Like a protocol stack, each level of hierarchy provides a service to the level (or levels) above it. In the IP Internet, the basic underlying network communication service offered is a connectionless, best-effort, datagram delivery service. This is the underlying service used for communication within and between the different levels of the Internet hierarchy.

It may be seen as disadvantageous to use an underlying communications service that offers no guarantees of reliability. On the contrary, a connectionless service best suits many applications (for example, name service [67], TFTP [91], NTP [21], and applications using RPC [66]). In essence, the approach taken by the IP protocol designers was "if the basic service isn't good enough for certain applications, then those applications can implement an end-to-end transport protocol to augment the underlying service". In IP, this enhanced service is provided by the TCP protocol [79]. If the IP protocol designers had thought otherwise, many of the applications that *require* a connectionless service, such as IP multicast applications, would never have emerged. Furthermore, it is much harder (often impossible) to de-enhance a service than to enhance it.

As we have said, IP multicast applications require a connectionless underlying delivery service. Typically, a connectionless service offers no guarantees of reliability in terms of packet delivery. This is not to say that applications can not employ their own techniques to "upgrade" the underlying service to a reliable one – this is what multicast transport protocols [2, 82] have been designed to achieve (or rather, have *attempted* to achieve[1] [47]). However, providing *data* reliability in the form of acknowledgements and re-transmissions simply does not fit in with the multicast model. Firstly, it makes no sense for some multicast applications to employ reliability features. For example, real-time conference applications that transmit voice simply could not take advantage of reliability features – if a voice packet(s) gets lost, it makes no sense to re-transmit the lost packet since it only has relevance if preceeded and succeeded by the other packets generated in the first instance.

Secondly, acknowledgements converging on a data source from any number of receivers (so-called "concast" [81]) are a considerable burden to a sender. For large groups this could also cause serious congestion problems at or near a sender.

A heterogeneity of network *types* may pervade any layer of the Internet hierarchy, for example, X.25 [17], Frame Relay [13], SMDS [62], ATM [61]. Of these, only SMDS offers a "connectionless" service, and only X.25 is a network layer protocol. Unlike TCP/IP networks, the routing function of Frame Relay, SMDS, and ATM is at the link layer.

So, the question is: how do "connection-oriented" networks support multicast? In actual fact, *most*[2] of them do not, the reasons being the same as those described above – reliability features (which are inherent in "connection-oriented" technologies) are not suited to multicast.

In the late 1980's it was recognised that the current IP protocol (IP version 4) [80] with its 32-bit address space, could not sustain the continued growth of the Internet – it currently comprises well over 20,000 registered networks [19]. This led to a concerted effort by the Internet community to come up with a replacement for IPv4, that could sustain expected Internet growth for at least the next 20 years. Recently, a proposal was approved by the Internet Architecture Board (IAB) to replace IPv4, which is called IPv6 [37].

---

[1] Typically, multicast transport protocols are complex and inefficient.

[2] A recent proposal for ATM multicast, based on our work, is the discussed in Chapter 8.

Unlike IPv4, IPv6 has made explicit provision for *flows* – a *flow* is a data stream that places certain requirements on network resources over the path the data traverses. IPv6 has a variable length header to fully accommodate variable length options, security fields, strict source routes etc., as well as a *flow-identifier*. In "connection-oriented" networks such as ATM, a flow is created as part of the call set-up phase of the communication, with flow parameters, such as desired throughput, required bandwidth, required reliability etc., being specified in the form of a *flow specification*. The flow specification tells routers how subsequent data packets should be handled and routed.

In "connectionless" networks a separate protocol is needed to create either hard state (e.g. ST-II) or soft state (e.g. RSVP) to reserve resources, or *flows*, thereby emulating virtual circuits of some "connection-oriented" networks. RSVP [102] was designed to support underlying multicast (unlike ST-II), as well as unicast, routes. Whilst this means that multicast reservations can be established, IPv4 has no place for the flow-identifier, which potentially can be used alone to route a packet. However, our proposed multicast protocol has made an allowance for the presence of a flow-id.

The multicast architecture we propose is based on a hybrid approach, utilizing both the "connection-oriented" and "connectionless" paradigms: our multicast delivery tree requires explicit set up, maintenance, and teardown, whilst data flow across a multicast delivery tree conforms to the unreliable, best-effort delivery service. We describe the architecture and protocol in chapters 3 and 5, respectively.

Our new multicast architecture is based on a technique called *centre based forwarding*, first described by Wall in the early 1980's in his PhD dissertation on broadcast and selective broadcast [100]. At this time, multicast was in its very earliest stages of development, and researchers were only just beginning to realise the benefits that could be gained from it, and some of the uses it could be put to. It was only later that the class D IP multicast address space was defined, and later again that intrinsic multicast support was taken advantage of for broadcast media, such as Ethernet.

Now that we have several years practical experience with multicast in the Internet [16], a diversity of multicast applications, and an internetwork infrastructure that wants to support it to an ever-increasing degree, we re-visit the centre-based forwarding paradigm introduced by Wall, and mould and adapt it specifically for today's multicast environment.

We will indeed see that an old idea can go a long way.

Centre based forwarding uses a *single spanning tree* that links a set, or group, of receivers in a communication. There are significant differences in the *construction* techniques of Wall's centre based trees and those of our own, which we will elaborate on in chapter 3. We will provide a summary of Wall's work, and his findings, in chapter 2.

1.2 Thesis Overview

Chapter 2 discusses related work, primarily that of Wall [100] and Deering [26]. 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). We provide a summary of Wall's tree construction techniques and the algorithms he used for tree building.

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 in this thesis – we provide an overview of these algorithms in chapter 2, and discuss their *scalability* in detail in chapter 4.

We did not consider it appropriate to provide an extensive overview of much earlier work that assisted in the evolution of the multicast capability we have today, since this is covered elsewhere [26]. However, we will briefly mention early work where we consider it relevant.

Chapter 2 also presents an overview of *Protocol Independent Multicast (PIM)* [24] – the multicast architecture developed shortly after the emergence of the *Core Based Tree (CBT)* multicast architecture, presented in chapter 3.

Chapter 3 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.

Chapter 3 also looks at the implications of utilizing core based trees for multicast, and identifies its potential disadvantages.

We also provide a summary of simulation results carried out to compare and contrast CBTs with shortest-path trees (SPTs) based on certain criteria such as delay and link utilization.

Finally, we discuss the essential differences between Wall's centre trees and our CBT trees. For our comparison, the discussion of Wall's algorithms in chapter 2 provides us with much of the necessary material.

Chapter 4 addresses 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.

Chapter 5 presents the CBT *protocol* in detail. We discuss the protocol intrinsics of CBT tree set-up, maintenance, and teardown. We also discuss data packet forwarding, which differs, depending on whether a packet is being forwarded over CBT *tree links*, or multicast-capable subnetworks.

Chapter 5 also presents our modification to the *Internet Group Management Protocol (IGMP)* which has reduced *leave latency* from around four and a half minutes to just twelve seconds[3].

In this chapter we also present CBT packet formats and message types, providing detailed descriptions thereof.

We go on to discuss *interoperability* in detail. We show how IGMP needs to be modified to reduce unnecessary bandwidth consumption and processing overhead of IGMP messages when a CBT router co-exists with a multicast router of another scheme on the same subnetwork. Further, we present some open and, as yet, unresolved interoperability issues.

Chapter 6 covers *multicast security*. The security issues discussed in this chapter are generic to multicast, and are therefore not specific to CBT multicast. We explain why multicast communication is at an increased risk from certain threats and we explain what those threats are.

---

[3] An approximation. Lower leave latency is also configurable.

We go on to present two contributions to general multicast security: *multicast group access control*, and *multicast transit traffic control*. We describe these in detail, and follow on with an analysis of our new mechanisms, together with an analysis of specific threats given the presence of our new mechanisms.

Chapter 7 presents the CBT *security architecture*. We explain why the CBT architecture is well-suited to the integration of security, and show how the CBT architecture can be used to provide a solution to the well-known *multicast key distribution problem*.

We follow with a detailed presentation of the CBT security architecture, demonstrating how it can be used to provide for the authentication of new group members and tree nodes, as well as provide low-cost, scalable multicast key distribution.

Chapter 8 discusses multicast in an ATM environment. We discuss proposals for running IP over ATM networks, and go on to explain why source-based ATM multicast trees are not suited to an ATM environment.

We follow this with an overview of a proposal made recently to use CBT for ATM multicasting. We explain how CBT is well-suited to the ATM paradigm and provide examples of CBT multicast in an ATM network.

Chapter 9 offers our conclusions, and identifies some issues outside the scope of this thesis that would benefit from further work. Finally, we present a summary of our main contributions.

Chapter 2

Related Work

This thesis, and the related work that 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 [2, 82], and elsewhere in the literature.

We primarily focus on three areas of related work. Two of these involve relatively recent contributions to network layer multicast, namely Deering's work [26], and an ongoing collaborative effort (Estrin, Deering, et al.) called *Protocol Independent Multicast (PIM)* [24]. Deering's work can be considered a culmination of the analysis of much earlier work done on multi-destination delivery by Dalal and Metcalfe [22], and others [10, 100]. His work represented a considerable advancement in multicast technology. Deering's work forms the core of multicast capability today, and several of the algorithms that he devised are in widespread use. Indeed, the current MBONE runs one of the protocols he invented.

Deering's thesis also presents an overview of much earlier related work, and so we will not discuss the very early contributions here. However, we will study the work done by Wall [100], so that our work can be effectively compared and contrasted with it, seeing as our work uses the same *shared tree* paradigm for multi-destination packet delivery. We will 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.

As we have said, we will look at *Protocol Independent Multicast (PIM)* which is an ongoing research effort. This work was motivated by our own work, and both our own work and PIM are currently undergoing review in a working group, set up by the author, of the "Internet Engineering Task Force". It is the goal of each working group of the IETF to analyse proposals put forward, with the ultimate aim of producing a standardized protocol.

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.

Let us first take a look at Wall's work on centre based forwarding, describe what he achieved, and some of the methods he used.

2.1 Wall's Work on Centre Based Forwarding

Wall's thesis provides an in-depth comparison of *low-cost* and *low-delay* tree approaches to broadcast and selective broadcast (i.e. multicast).

Cost and *delay* are very different quantities. Using the simple diagrams of figure 2.1(a) and 2.1(b), we illustrate the difference between "delay" and "cost".

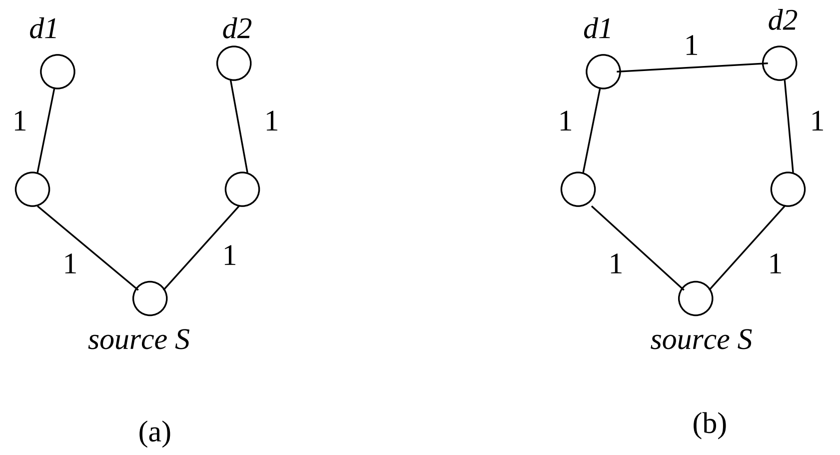

Figure 2.1: Delay vs. Cost

In figure 2.1(a), the maximum *delay* in reaching destinations d_1 and d_2, with respect to source S, is 2 hops, and the total *cost* in reaching destinations d_1 and d_2 is 4 hops from S. In figure 2.1(b), destinations d_1 and d_2 can be reached using just a single path from S, with the maximum delay being in reaching d_2, at 3 hops. The total cost, however, in reaching both destinations is 3 hops.

As is evident, building a *minimum-delay* delivery tree involves building a separate *shortest-path* tree (SPT) from a source to each destination. However, a *minimum cost* tree can be built using a *single* spanning tree.

Shortest-path trees (SPTs), whilst achieving minimal delay between a source and a destination(s), do not attempt to minimize the total cost of distribution, but are relatively easy to compute in a distributed manner, as we describe in the next section on Deering's work.

It is important to put both "cost" and "delay" in a multicast context. "Cost" has implications regarding the *scalability* of multicast, and should be divided between the following three measures:

- the greater the cost of the delivery tree, the greater will be the overall transmitter costs. In our thesis abstract we stated that optimizing transmitter costs was one of the primary benefits, or should we say goals, of multicast.
- cost of bandwidth consumed by a multicast delivery tree.

- cost associated with tree state information.

"Delay", on the other hand, has relevance only to certain multicast applications, or, to be more precise, to *users* of those applications. Delay is not always an issue, for example, replicated database updating or querying, and resource discovery do *not* impose delay constraints. However, for real-time applications, such as voice conferencing [16], the delay experienced between a sender and the group's receivers is critical, and it must not exceed certain bounds (on the order of a few hundred milliseconds for voice), otherwise the users of these applications become frustrated and disorientated.

It is worth mentioning, with respect to *cost*, the difference in complexity in building a minimum-cost tree that spans a subset of the vertices of a connected graph, i.e. a *minimum steiner tree*, and a minimum-cost tree that spans all of the vertices of the connected graph, i.e. a *minimum spanning tree*. It was shown in [53] that building a minimum-cost steiner tree is NP-complete, whereas building a minimum spanning tree is not [75]. The point we are trying to make here is that building a single, minimum cost tree, spanning a group of receivers within an internetwork, is NP-complete.

Kou, Markowsky, and Berman [58] designed an algorithm, known as the KMB algorithm, which offers a good approximation to a minimum Steiner tree. The cost of the resulting KMB tree averages 5% more than the cost of a minimum Steiner tree. However, the KMB algorithm requires complete network topology information, and therefore is of no interest where large internetworks are concerned [60]. More recently, a routing algorithm was proposed for connection-oriented networks, such as ATM, which under certain circumstances, is nearly as efficient as the KMB algorithm [51].

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 cannot be achieved using any one type of distribution tree [100]. 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 section 2.2) the "centre" chosen as the node either with the smallest maximum delay, or the node with the smallest average delay. Either of these *criteria* was considered a suitable candidate for the "centre". We describe the election of the "centre" of Wall's trees, and their construction in section 2.2.

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* [26]. 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 today's multicast environment without considerable adaptation.

Also, it may *not* always be desirable for all multicast applications that use a shared tree to have it centre-rooted. Take, for example, a lecture or seminar that is to be "audiocast" [16] to multiple recipients all over the world, as is often done on the MBONE. For such cases, where the principle sender is at one site, it makes sense for the "centre" to be moved to, or near to, that site so that the centre-rooted tree emulates a shortest-path tree. Hence, time-critical applications such as audio- and video "broadcasts" can take advantage of the more desirable delay characteristics of a shortest-path tree, whilst actually being part of a shared delivery tree.

Additional "centre nodes", or *cores* as we will call them, may be placed in the same locality[1] or alternatively, where the larger pockets of members are located. With regards

[1] By this we mean not necessarily directly adjacent, but perhaps within several hops.

to this latter strategy, in the absence of a dynamic core placement algorithm (the pros and cons we will discuss later) and associated protocol, it is often difficult to know at group-initiation time, where the larger pockets of members will be located in the internetwork. The purpose of additional cores in our model is for greater robustness – a shared delivery tree centred around a single router offers very poor fault tolerance, since the stability of the tree is completely dependent on a single router. Wall does not consider additional "centres" in his trees. We add additional cores to make a tree more robust against failure, as well as protocol mechanisms to detect and recover from failures. These will be discussed in due course.

2.2 The Construction of Wall's "Centre-Trees"

Wall's *minimization algorithm* for constructing "centre-trees" actually involves all nodes in a graph attempting to build a tree, with trees "killed off" when some better tree is seen. The resulting *single* spanning tree is a *broadcast* tree rooted at some elected "centre" node. After our description of the broadcast tree construction, we explain how this technique was adapted slightly to accommodate selective-broadcast (i.e. multicast).

Wall's algorithm involves every node in a graph (network) broadcasting, by means of a *multiply-addressed* message, its *centre criterion value*. Each receiving node notes which edges (interfaces) it uses to forward the received message.

To reduce communication overhead, each node only forwards these so-called tree "construction" messages if the message contains a lower criterion value than the lowest already recorded by that node. The primary candidate for the "centre" is the node with the lowest criterion value. Similarly, a node need only remember the edges (interfaces) over which it forwards a lowest-value "construction" message. Other "construction" messages received are simply discarded.

The nodes on the periphery of the network (leaves) are required to bounce the "construction" messages back to the sender. This constitutes a "convergecast" or "concast", and these messages are interpreted as construction message acknowledgements. Irrelevant acknowledgements (i.e. ones not corresponding to the lowest-value "construction" message) are discarded similarly by receiving nodes, thereby reducing network traffic. In this way, a single spanning tree results, identified by the paths followed by the convergecast (acknowledgments), resulting from the lowest-value "construction" message.

Finally, so the network nodes know when the algorithm is complete, on receipt of an acknowledgement, the node that generated the corresponding "construction" message sends a final message announcing that tree construction is complete, which follows the branches of the new tree [100].

This algorithm becomes less optimal if nodes' routing tables are inconsistent, if nodes fail, or if nodes re-start. Node failure is a problem, since, if a "construction" message includes a failed node in its address list, the neighbouring nodes will shunt it around indefinitely. Wall has not made any provision for node failures in his algorithm [100].

Re-started nodes are an even harder problem. If a node has only recently re-started, a "construction" message originating in some remote part of the network is unlikely to have included it in its address list, even though it probably should have. Again, Wall has not made any explicit provision for such a scenario in his algorithm, other than to make some not-very-elegant suggestions as to how the problem might be remedied.

2.2.1 Adaptation for Selective-Broadcast

Selective-broadcast was only a secondary goal of Wall's thesis, and so his thesis is less extensive with regards to selective-broadcast (i.e. multicast). Nevertheless, Wall made suggestions for extending his broadcast algorithms for the general case of selective-broadcast.

Wall attempts to continually adapt a centre tree, so that, in spite of addition or removal of nodes (corresponding to new group members, and group members leaving, respectively), the "centre" remains optimally placed. This implies that an iteration of the minimization algorithm is required each time a node comes or goes. This further implies that highly dynamic group membership is likely to incur an excessive overhead in the form of network traffic and node processing.

The minimization algorithm, described above, has some problems in its current form in coping with added nodes (analogous to new members of a group). Currently, the algorithm states that if a node receives a message about a tree that is non-optimal, it discards that message. For selective-broadcast, the algorithm requires additional communication: on receipt of a non-optimal message, the receiving node must reply to the sending (immediate neighbour) node with a "kill" message, which is propagated back as far as necessary, until it reaches a node that knows the corresponding tree is sub-optimal. At this point, the "kill" message can be discarded. Similarly, a "kill" message is required when a "construction" message is received about a better tree – the "kill" is then forwarded over the branches of the old tree as far as necessary [100].

Finally, Wall suggests that, for the case of selective-broadcast, having the "centre" node outside the group of original nodes may result in more optimal trees. However, this would require identifying such candidate nodes early on, so as to include them in the minimization algorithm [100].

2.3 Deering's Work

Deering's work [26] involved extending the three basic routing algorithms: *single 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 thesis), but Deering also described how a combination of his algorithms could be used in a very large internetwork that is structured as a *hierarchy*. Deering's link-state and distance-vector algorithms build *source-rooted* delivery trees, i.e. delivery trees emanating from the subnetwork directly connected to the source.

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.

In the context of multicast, a truncated-broadcast tree results in packets reaching subnetworks where no members exist. A *multicast tree* however, is one which spans only those subnetworks with group members, and routers and subnetworks on the path to

[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.

subnetworks with group members. However, the establishment of a multicast tree requires that routers either maintain additional state, as is the case with *distance-vector* multicast routing, or engage in the exchange of group membership information with all other routers, as is the case with link-state multicast routing, as we will see below.

In terms of practical implementations, truncated-broadcast algorithms for multicast packet delivery have become virtually obsolete because of the superfluous packet delivery they incur. Therefore, we will not describe Deering's truncated broadcast algorithms. We will also not discuss his extensions to *spanning-tree* routing for multicast, since we are concerned with network layer multicasting – spanning-tree routing occurs between data-link layer bridges. These algorithms are described in [26].

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 [32]. 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)* [98, 26]. There are also a number of Autonomous Systems (AS's) – primarily those whose unicast interior gateway protocol (IGP) is OSPF [68], which are running the *Multicast Open Shortest-Path First (M-OSPF)* protocol [69], based on the link-state multicast algorithm. M-OSPF *regions* interface to the MBONE by means of *border router(s)* specially configured to run an instance of each protocol.

Whilst both the distance-vector and link-state multicast *algorithms* were invented by Deering, M-OSPF was developed by Moy [70].

2.3.1 The Distance-Vector Multicast Algorithm

DVMRP [98] is a protocol implementation of the distance-vector multicast *algorithm* proposed by Deering [26]. This is a refinement of Dalal and Metcalfe's Reverse-Path Forwarding algorithm [22], proposed previously. The distance-vector multicast *algorithm* uses (destination, distance) vectors to advertise multicast-capable subnetworks to participating

[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.

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 [22] for internetwork broadcasting. Deering modified the RPF algorithm slightly to eliminate the possibility of multicast duplicates being sent across multi-access links [25].

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.

DVMRP uses the RPF strategy which results in a shortest-path, sender-rooted, 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.

2.3.2 The Link-State Multicast Algorithm

M-OSPF is a protocol implementation of the link-state multicast *algorithm* proposed by Deering [26]. 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 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 protocol (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[4], 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 [69].

2.4 Protocol Independent Multicast (PIM)

The PIM multicast protocol is one of two multicast protocols (the other being the CBT Multicast protocol) currently undergoing review in the *Inter-Domain Multicast Routing (IDMR)* working group of the Internet Engineering Task Force (IETF). The IDMR working group was formed by the author as a result of his work identifying scalability problems in existing multicast routing protocols. The ultimate goal of the IDMR working group is to propose a multicast routing protocol *standard* that can be used to achieve *scalable* inter-domain multicast routing.

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

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 [20] 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*.

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 explicitly 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 [24].

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 later as a significant contribution to this thesis. 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. We will see later that there are pros and cons to each methodology.

Chapter 3

The Core Based Tree (CBT) Multicast Architecture

3.1 Introduction

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.

A Class D address identifies a single host group. The class D address space is separate portion of the IP address space, defined to be between 224.0.0.0 and 239.255.255.255 inclusive. IP address classes are defined in [80]. A thorough taxonomy of Internet addressing is provided in [36]. A small number of class D, or group, addresses are reserved for use by various routing protocols, such as 224.0.0.1 – the *all-systems* address, to which all multicast-capable UNIX end-systems are permanently subscribed.

Whenever a host wishes to subscribe to a particular group, it sets its network interface so as to receive all packets whose destination address corresponds to a particular class D address. All end-systems wishing to participate in multicast must have a directly connected multicast-capable router. This is true both for new IP multicast schemes like CBT and PIM, and older schemes like DVMRP.

IGMP is the protocol implemented in hosts and routers on LANs to monitor multicast group presence on a subnetwork. One router per LAN is elected as *membership interrogator*. This election is implicit in the IGMP protocol and happens at start-up time. At fixed intervals the elected interrogator sends a non-group-specific *membership query* to the *all-systems* multicast group. Hosts receiving this query do not respond immediately, but rather randomise their response over a ten second interval. On expiry of this interval a host sends a *group membership report*, once for each group it is affiliated to, and addressed to the corresponding group. All local multicast routers receive this report. If a group report arrives at a host before its own response interval expires, the corresponding membership report is cancelled at the receiving host. In this way, the membership interrogating router learns of subnetwork group presence, and subnetwork bandwidth consumption due to membership reporting is minimized [26].

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. The details of our new IGMP version, and the motivations behind it, are given in section 5.1.11.

3.2 Extending Multicast's Existing Properties

Deering, in [25], 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 [19] 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. A loose-coupled dependency on underlying unicast routing is a necessity, since multicast routes are, or should be[1], ultimately gleaned from the unicast routing table.

  Such independence is highly desirable for two reasons: it simplifies multicast routing across heterogeneous domain boundaries, and it allows for independent evolution of both unicast and multicast algorithms.

3.3 Existing Multicast Architecture

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.

When the existing multicast algorithms were designed, it could not be foreseen to what extent multicast would grow in popularity. Today, if one looks at the multicast tools available, such as "sd", "vat", and "wb" (all developed by Van Jacobson et al., LBL), and the protocols that have been developed to support real-time multicast traffic, such as RTP [87], it is clear that multicast capability has become a highly-desired feature in the Internet.

Below, we identify some of the shortcomings of the existing multicast architecture.

3.3.1 Some Shortcomings of the Existing Architecture

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 the next chapter,

[1] Until multicast capability is installed in the internetwork infrastructure, multicast routing protocols must implement their own *routing daemon*, thereby establishing a virtual network, which is an overlay of the underlying network.

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 [26], 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 propagated 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 [68]. 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.

3.4 CBT - The New Architecture

3.4.1 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 [100]. However, the similarity in his work and ours ends in the utilization of a shared tree. We provided an overview of the primary differences between Wall's work and ours in chapter 2. We elaborate on these differences in section 3.6.

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 emanate. 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[2] 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 3.1. 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.

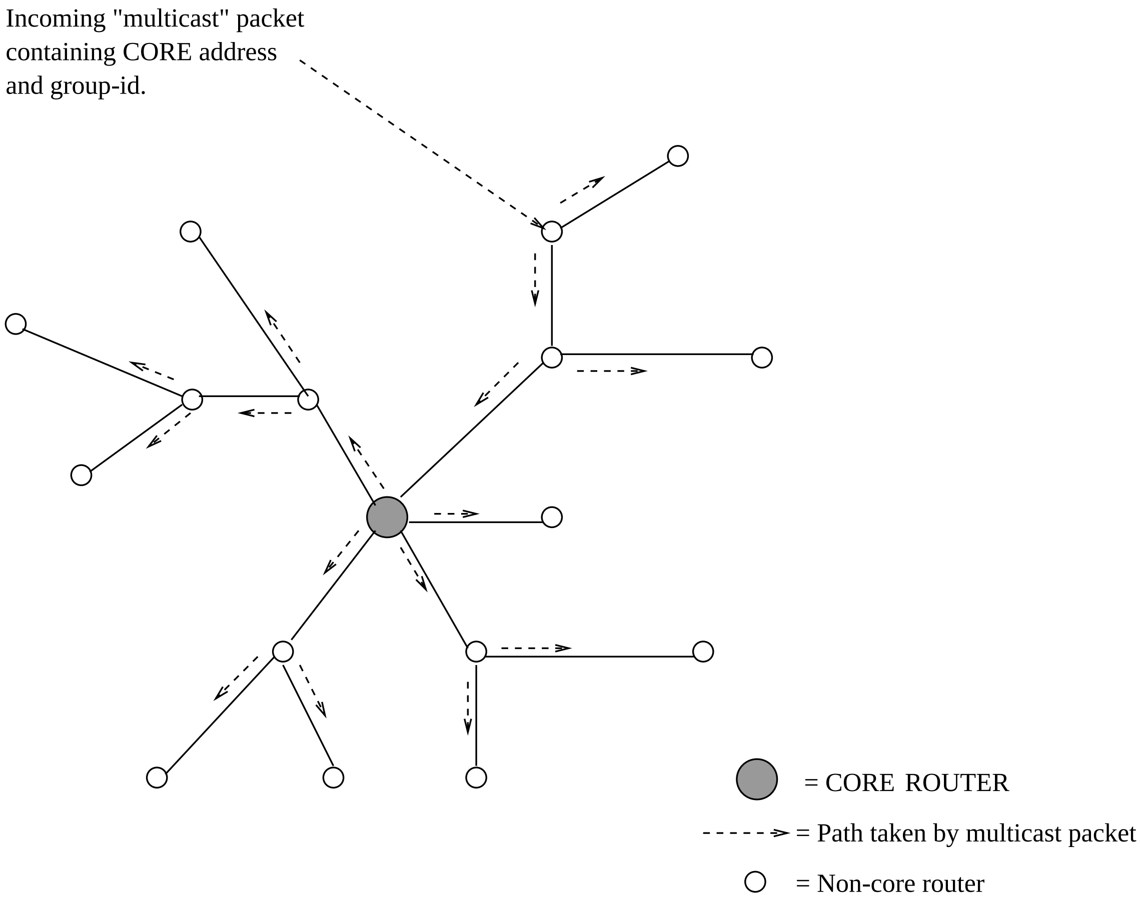

Figure 3.1: A CBT multicast delivery tree

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

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

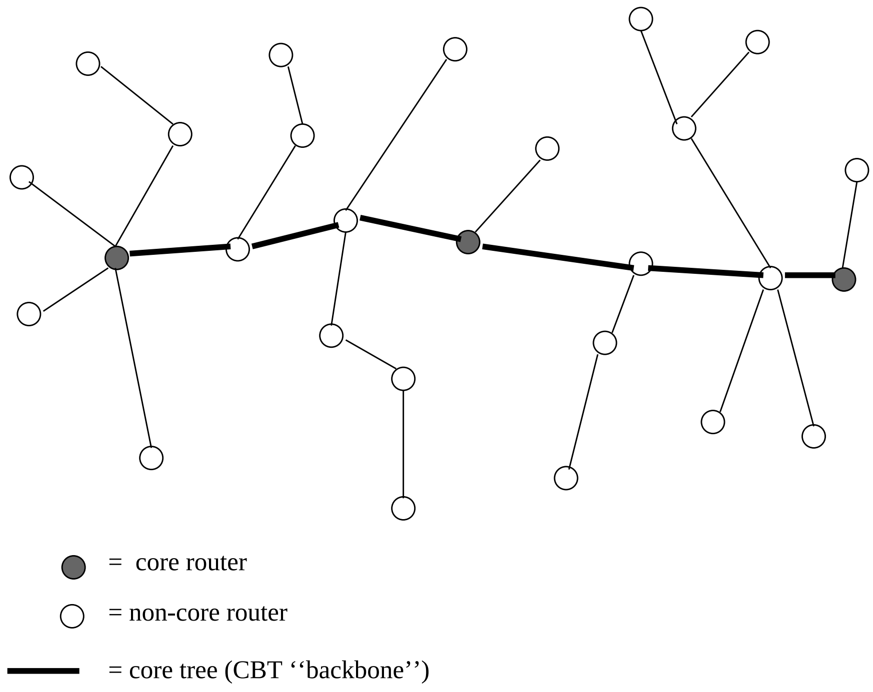

Figure 3.2: A Multi-Core CBT tree

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, in terms of group-specific state maintained in the network, particularly for the case where there are many active senders in a particular group. The CBT architecture offers an improvement in scalability over existing *source-rooted* 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. Routers that comprise a *source-rooted* tree (as built by DVMRP) maintain source-specific group information. Thus, if, within some timeframe[3], all members of a group become active senders or send simultaneously, a mesh of trees results, with network routers having to keep state for each source tree. Figure 3.3 should help illustrate this.

[3]Data-driven schemes like DVMRP cause routers to hold state information pertaining to a particular source-based tree for a period after which data flow has ceased. This is often called the *timeout* period, or *refresh* period.

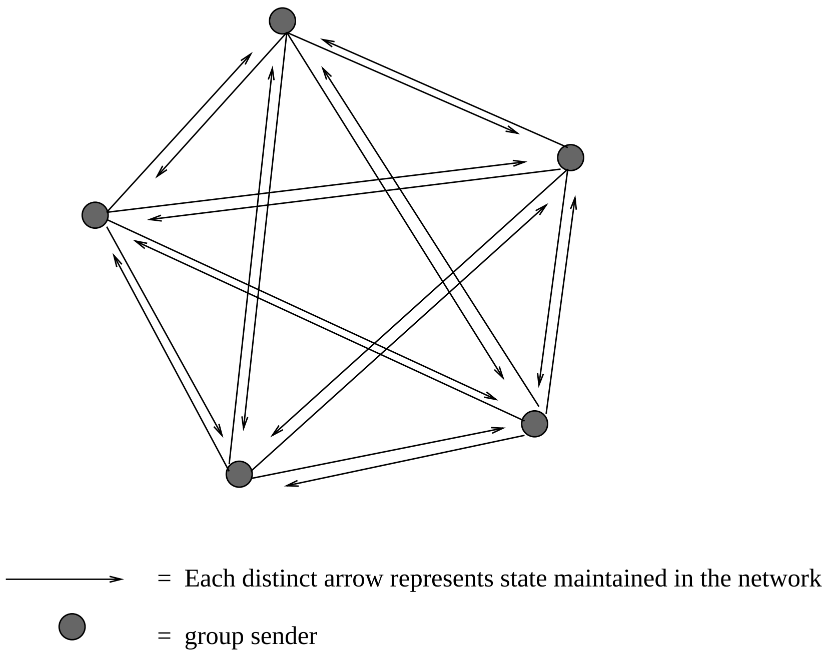

Figure 3.3: Mesh of multicast delivery trees

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 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* [73]. We believe that CBT offers a good solution to anycasting, which we discuss in more detail in section 3.6.1. Also, we provide a practical example of the use of *anycasting* in section 6.6.2.

The other main premise driving the CBT approach is that it is unicast routing protocol independent, meaning that the correct operation of the multicast protocol is not inextricably linked to the existence of a particular underlying unicast protocol, or, more precisely, specific features of the unicast protocol. As we have said, DVMRP [98] is a multicast protocol that relies on such features.

3.4.2 Architectural Justification

As we discussed in chapter 2, 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 [60]. We summarize these results in section 3.4.4.

In the context of multicast, the non-optimal delay characteristic of centre-based trees by no means makes them redundant, since there are a whole host of factors that additionally need to be taken into account – one needs to look at the membership and traffic flow dynamics of each individual group. For example, a video "broadcast" involving only one sender would enable the group's core(s) to be placed at or near the point of source, resulting in a shortest-path delivery tree. Slight discrepancies in delay may not be a critical factor for many multicast applications, such as resource discovery or database updating/querying. Even for real-time applications such as voice and video conferencing, a core based tree may indeed be acceptable, especially if the majority of branches of that tree span high-bandwidth links, such as optical fibre. In several years' time it is easy to envisage the Internet being host to thousands of active multicast groups, and similarly, the bandwidth capacity on many of the Internet links may well far exceed those of today.

The CBT architecture imposes a single, shared delivery tree that is used for sending and receiving multicasts. The protocol requires each receiver to explicitly join the delivery tree, resulting in a tree spanning *only* a group's receivers. As a result, CBT has no requirement to *scope* multicast packets, i.e. restrict them from spanning internetwork links that do *not* lead to group receivers. This is a serious drawback of certain source-based multicast algorithms.

3.4.3 The Implications of Shared Trees

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

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

A consequence of the "hard-state" approach is that multicast tree branches do not automatically adapt to underlying multicast route changes[4]. 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.

Another consequence of one shared delivery tree is that the cores for a particular group,

---

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

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 [60].

The branches of a CBT tree are made up of a collection of *shortest reverse paths*, 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 described some very simple heuristics to address some of these questions in section 3.5.1, but these may not be appropriate for large-scale implementation of CBT. However, given *a priori* knowledge of a group's participants, fast and scalable algorithms have already been devised for finding data "distribution centres" [90]. We conclude that most other aspects of core management are topics of further research.

3.4.4 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 [60] 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 [60].

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 minimal 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* [90]. 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. It is also implied that the centre-location mechanism assists in the generation of low cost centre-specific trees. 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" [90].

3. A PIM-CBT comparison has also been carried out very recently [15], 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.

3.4.5 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 [16] 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 [28] showed that dynamically changing the form of a multicast tree is usually not worthwhile for dynamic multicast groups in terms of efficiency[5] 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[6].

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.

3.5 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, presented in 2.2 and 2.3, and the CBT protocol we present in chapter 5. We summarize the essential differences between Wall's trees and CBT trees below:

---

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

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

- 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.

  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:

	conn-oriented	**tree state**	**explicit tree maintenance**	**no. of cores**	**tree adapts to route changes**	**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 3.1: Comparison of Tree Types

3.5.1 CBT and "Anycasting"

Overview of "Anycasting"

Anycasting [73] is a proposed best-effort, stateless, datagram delivery service which is used by hosts primarily to locate particular services on an internetwork. The goal of anycast is for a client to transmit *one* request to a resource "anycast address", and for a single, preferably nearest, server to receive the request and respond to it.

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

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

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

Using existing IP multicast has problems associated with it. Firstly, using *expanding ring search* [26] to locate a network resource is inefficient for two reasons: it requires potentially many re-transmissions of the request from the client, each iteration requiring a larger TTL[7] value . This continues until a response is received.

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

Defining a special class of "anycast" addresses has several problems associated with it. For example, routing must be adapted to support yet another class of IP address, and routing tables would be required to support anycast routes. Furthermore, segmenting the IP address space yet further not only involves significant administrative burden, but also assumes that existing applications will recognise particular addresses as being anycast [73].

---

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

The CBT "Anycast" Solution

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

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

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

For typical multicast applications, the receiving on-tree router disseminates the received packet(s) to adjacent outgoing on-tree neighbours, and neighbours proceed similarly on receipt of a packet. This is how multicast data packets span a CBT tree.

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

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

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

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

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

3.6 Chapter Summary

In this chapter we have identified the need to extend the existing properties of multicast, originally proposed by Deering [26], to include the property of *scalability*, and underlying routing algorithm *independence*. We also identified the disadvantages of the source-based multicast architecture, upon which all existing multicast protocols are based. Essentially, the source-based architecture involves group information, either in the form of data, or group membership information, being sent to parts of the network that do not require it. The effect of this is that, in the "uninterested" parts of the network, valuable resources are unnecessarily consumed.

We proceeded to describe a new multicast architecture based on a *shared* delivery tree approach. These *core based trees (CBTs)* span only a group's receivers, with the effect that multicast packets only flow to those parts of the network interested in receiving them.

The CBT architecture has several advantages compared with existing schemes, which include:

- the CBT architecture eliminates the source scaling factor of the source-based architecture (in terms of group-specific state maintained in the network).
- no state is maintained by the network between a non-member sender and the delivery tree.

- CBT's two-phase routing approach means that it is a candidate for information-discovery applications. "Anycasting" is the ideal solution to information-discovery in that it makes most efficient use of network resources, and finds the topologically nearest information source. Whilst CBT cannot guarantee the latter, it nevertheless offers an efficient solution to "anycasting".

We also looked at the implications of shared multicast delivery trees, and concluded that they also offer several disadvantages, including:

- shared delivery trees do not optimize delay, which is critical to some multicast applications.
- CBTs can result in so-called *traffic concentration* [60] around core routers.
- CBTs require the selection of a small number of core routers by a group's *initiator*. Core selection and placement, i.e. core management, are functions not required in existing IP multicast schemes.

We also summarized simulation results from experiments conducted independently by Wei [60] and Shukla et al. [90]. The subject material of these simulations differed slightly, but essentially they compared source-based and centre-based delivery trees, using various random network topologies and group configurations. These results conclude that CBTs are appropriate in certain situations, for example, in network topologies with low connectivity, or where delay is not a critical factor.

Finally, we presented the CBT solution to "anycasting" [73]. The CBT non-member sending mode of operation, or "anycast" mode, satisfies many of the goals of "anycast", as specified in [73].

Chapter 4

Multicast Scalability

4.1 Introduction

The fundamental premise that motivated this thesis is multicast scalability. We have already emphasized how multicasting has grown over the last couple of years – it was July 1992 that saw the Boston IETF[1] meeting audio- and videocast various meetings over a relatively sparse MBONE, as it was then [16].

Now, the MBONE comprises well over one thousand networks[2], and its surprisingly fast growth has imposed expectations on what started out as, and officially still is, an experimental service between a small number of sites. A graph illustrating MBONE growth is given in figure 4.1.

The benefits of multicast have quickly become apparent to network managers, partly due to the ever-increasing competition for Internet resources (such as bandwidth, and buffer space [102, 89, 99]) and the monetary cost of those resources. Application developers are, and have been for some time, encouraged to write conference applications that utilize multicast, since it is conference applications that are the most demanding of network resources. Example conference applications include "vat", "nv", "nevot", "vic", and "ivs"

[1] Internet Engineering Task Force.

[2] Actually, the MBONE distinguishes between *subnetworks* rather than networks.

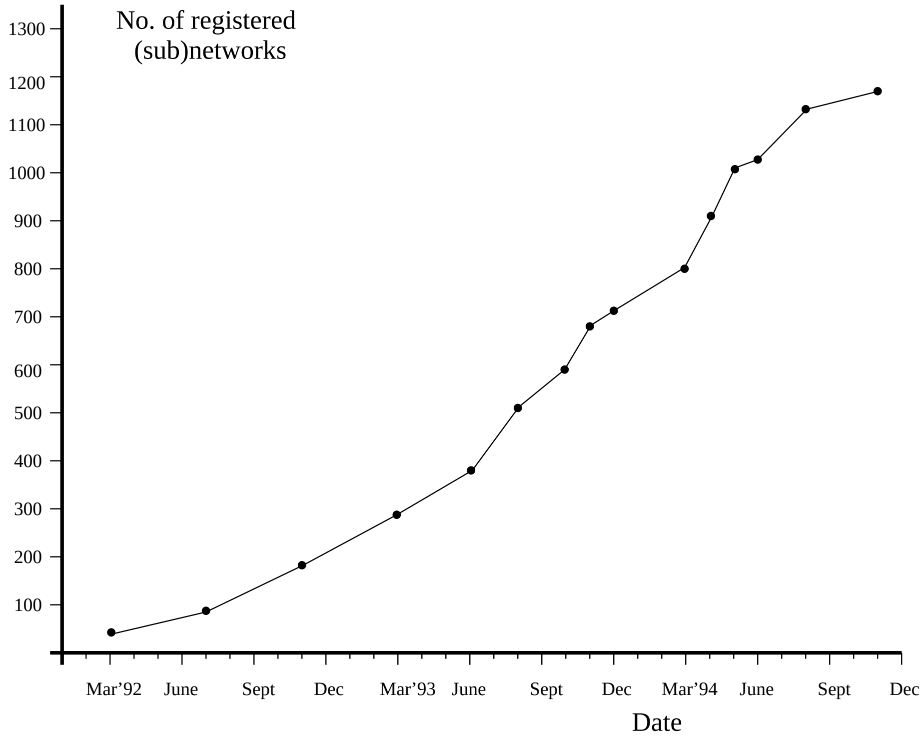

Figure 4.1: The Growth of the MBONE between March 1992 and November 1994

are all described in [65] and [32].

As a consequence of multicast's success, as evidenced by the rapid growth of the MBONE, the traditional IP multicast protocol, DVMRP (see section 2.3.1), is being stretched to its limits in its current form. That is, it was designed for a *flat* internetwork, where all participating routers in the network maintain the same level of information [26]. In other words, it operates as an *interior gateway protocol (IGP)*. Deering did suggest in [26] how his algorithms could be implemented in a *hierarchical* fashion, whereby, at a particular level of hierarchy, one algorithm operates (for example, M-OSPF operates within "areas"), and possibly a different algorithm, but not necessarily a different algorithm, routes multicasts between different levels of the hierarchy. Whilst this could be done, and would contribute to traditional algorithms' scalability, it would only do so with respect to the resources consumed *within* individual multicast routers. However, as we will see below, there are several criteria affecting an algorithm's scalability, and the *cost* to a transmitter is only one of them. Besides this, hierarchical routing tends to make routing a lot less optimal, depending on how many levels of hierarchy are introduced.

In this chapter 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.

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

4.2 The Scalability of the Distance-Vector Multicast Algorithm

DVMRP [98] is the only implementation of the distance-vector multicast algorithm [26] to date, and so we will use it as an example for our analysis.

4.2.1 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" [26].

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 [26].

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, and N is the number of groups present in the internetwork.

More precisely, the amount of prune state off-tree routers must maintain is bounded by the number of subordinate routers they have [26], and is 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?'.

4.2.2 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)[3], 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. 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 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-*

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

tree bandwidth consumed, other than to say that it is dependent on the membership dynamics and membership distribution of *active* groups.

4.2.3 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 [26].
- 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 [26]. 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 [27].

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 depend 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.

4.3 The Scalability of the Link-State Multicast Algorithm

M-OSPF [69] 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" [68]. 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.

4.3.1 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 [76]. 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 [24].

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 [69]. 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 [69].

4.3.2 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 disseminating group link-state updates is dependent on the frequency of the appearance and disappearance of group members on a subnetwork [26]. 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 few seconds [26].

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 [77] 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 [69].

4.3.3 Processing Costs

The most substantial cost to routers running the link-state multicast algorithm is the cost associated with Dijkstra computations. This cost scales $O(n \times log\ n)$, where n is the number of nodes in the network. This is the other major factor limiting the link-state algorithm's applicability to wide-area multicasting [24].

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 [69], 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 incur considerable processing costs.

4.4 The Scalability of the PIM Architecture

The PIM architecture was designed "to efficiently establish distribution trees across wide-area internets" [24], 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 environments, 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[4]. 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.

4.4.1 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.

[4] 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...

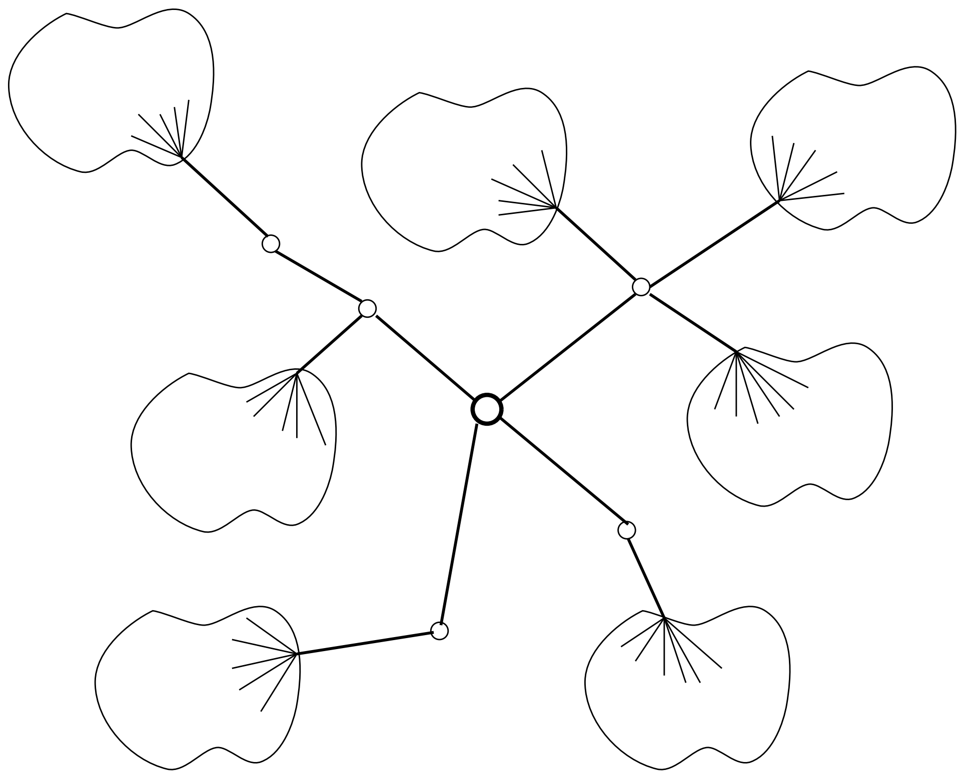

Figure 4.2: How PIM is used for Inter-Domain Multicasting

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 [48]. 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 [48].

4.4.2 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 in section 4.4.1 above.

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 [48] 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 disseminated 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" [24] are unavoidable.

4.4.3 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) [48].

Additionally, in sparse mode there is join traffic to consider. Using the inter-domain case (see figure 4.2) 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 [48].

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 $O(S_{local})$ [48].

4.5 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 disadvantages of the CBT architecture, in chapter 3.

4.5.1 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 disseminated 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* (minimal group information representing a range of groups), 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.

4.5.2 Bandwidth Consumption

No bandwidth is wasted due to pruning, and its associated inefficiencies, or grafting, since the CBT architecture does not 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[5], 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 overlap less than around the centre. Therefore, the

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

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.

With regards to data flow, because CBT imposes a shared delivery tree, parts of the tree can become traffic "hot spots" – this is especially true for core routers of wide-area groups that are shared between different groups. This problem is directly related to *link utilization* – CBT does not take advantage of additional links in a connectivity-rich environment as source-based schemes tend to do. However, as network connectivity decreases, link utilization of shortest-path tree schemes is very similar to that of CBTs [60].

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 section 5.1.8). 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 increase 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 [49].

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 4.1: 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.

4.5.3 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 daemon impose an upper bound on the number of routes that can be successfully handled [27].

4.6 CBT vs. DVMRP – A Quantitive Analysis

Before closing this chapter we thought it a good idea to present some sample figures to represent our qualititive analysis. Therefore, in the table below we compare the amount of state stored in CBT and DVMRP routers, given a variety of group dynamics.

It is extremely difficult without long-term practical experimental analysis to provide a quantitive comparison of the two protocols, and therefore the figures given are only approximations, but we expect them to be reasonably accurate.

The figures provided are based on the following protocol state characteristics:

- DVMRP requires a minimum of 10 bytes per source per group in routers forming part of a delivery tree, i.e. *active* state. A minimum of 10 bytes per source per group are also required in routers not part of a delivery tree, i.e. *prune* state.

- CBT requires a minimum of 6 bytes per group, irrespective of source, plus 1 byte per child per group.

Note: the above statistics allow a router to have a maximum of 16 directly attached subnetworks. DVMRP uses a fixed-size 2-byte statically allocated bitmap to register children, whereas CBT dynamically allocates 1 byte per child as necessary. We assume each CBT router has 2 children.

All groups are considered *wide-area* groups, i.e. traffic emanating from a group can potentially span the diameter of the MBONE. The MBONE currently comprises nearly 2000 subnetworks, with one DVMRP router per subnetwork on average. Also note that the sample storage figures are **per router**. For DVMRP routers, active and prune state for any (source, group) is not maintained in the same router.

No. of groups	10			100			1000		
Average group size (no. of members)	20			40			60		
No. of sources (per group)	10%	50%	100%	10%	50%	100%	10%	50%	100%
DVMRP									
active state	200	1000	2000	4000	20,000	40,000	60,000	300,000	600,000
prune state	200	1000	2000	4000	20,000	40,000	60,000	300,000	600,000
CBT	80			800			8000		

Table 4.2: Bytes of Storage Comparison of DVMRP and CBT Routers

4.7 Chapter Summary

In this chapter we have analysed the *scalability* of four multicast protocols – DVMRP, M-OSPF, PIM, and CBT, in terms of group state information, bandwidth consumption, and processing costs.

Multicast algorithms that build *source-rooted*, shortest-path delivery trees require most state to be maintained in the network, and scale at best $O(S \times G)$.

Multicast algorithms that build *shared* delivery trees eliminate the source scaling factor of source-based schemes, and scale $O(G)$.

The PIM multicast protocol was designed to establish efficient distribution trees that reflect both application requirements and group membership dynamics. PIM gives a receiver the option to either receive multicasts via a *shared* tree, which is the default initially, or switch to a *shortest-path* tree subsequent to pruning itself from the shared tree. For this reason, PIM is unusual in that, at best, it scales $O(G)$, and at worst, it scales $O(S \times G)$.

Scalability alone is not the only factor in choosing a multicast algorithm. One must take other factors such as application requirements, and domain (or scope) of operation into account. As we have said, within a resource-rich environment such as an organisational LAN, the characteristics of a multicast algorithm will be far more important than scalability.

Finally, we closed this chapter with a quantitive comparison of the amount of state required by each of a DVMRP and CBT router, given a variety of group dynamics. From this it is somewhat easier to extrapolate the overall state maintained in the network by each of the protocols.

Chapter 5

CBT – The Protocol

5.1 Introduction

This chapter describes the CBT protocol in detail. We describe the process of setting up a core based delivery tree, as is typically instigated by one group member – the so-called *group initiator*. We also describe how a new group member joins an existing CBT delivery tree, and how branches of a tree are removed when receivers (directly connected to such branches by a CBT router) leave the group.

We follow with an extensive description of tree maintenance issues, showing how a CBT delivery tree adapts to router, and core, failure. We also show how a loop-free tree is formed despite the possible presence of transient loops in a CBT router's database.

We go on to present some simple heuristics for core placement, and so-called *designated router (DR)* election on a multi-access subnetwork[1]. Finally, before moving on to describe some features which are less directly related to the protocol, we explain CBT's data packet forwarding rules both for member- and non-member senders.

[1]See glossary

5.2 Protocol Overview

5.2.1 CBT Group Initiation

Like any of the other multicast schemes, one user, the *group initiator*, or *group leader*, initiates a CBT multicast group. The procedures involved in initiating and joining a CBT group involves a little more user interaction than current IP multicast schemes, for example, it is necessary to supply information such as desired group *scope* so that the group's cores can be selected appropriately within the desired region. The current implementation makes use of three configuration files which correspond to the three currently-defined group scopes: *local*, *national*, and *international*. Each of these files contains an ordered list of cores which are used for all groups selecting that group scope.

Explicit *core rankings* help prevent loops when the core tree is initially set up. It also assists in the tree maintenance process should the tree become partitioned.

Group initiation could be carried out by a network management centre, or by some other external means, rather than have a user act as group initiator. However, in the author's implementation, this flexibility has been afforded the user, and a CBT group is invoked by means of a graphical user interface (GUI), known as the *CBT User Group Management Interface*. The CBT User Group Management Interface is shown below.

5.2.2 Tree Joining Process

Once the cores have been selected by a group's initiator, and the application, port number etc. have been entered at the GUI, the group-initiating host sends a special CORE-NOTIFICATION message to each of them, which is acknowledged. The purpose of this message is twofold: firstly, to communicate the identities of all of the cores, together with their rankings, to each of them individually; secondly, to invoke the building of the core backbone. These two procedures follow on one to the other in the order just described. New receivers attempting to join whilst the building of the core backbone is still in progress have their explicit JOIN-REQUEST messages stored by whichever CBT-capable router, involved in the core joining process, is encountered first. Routers on the core backbone will usually include not only the cores themselves, but intervening CBT-capable routers

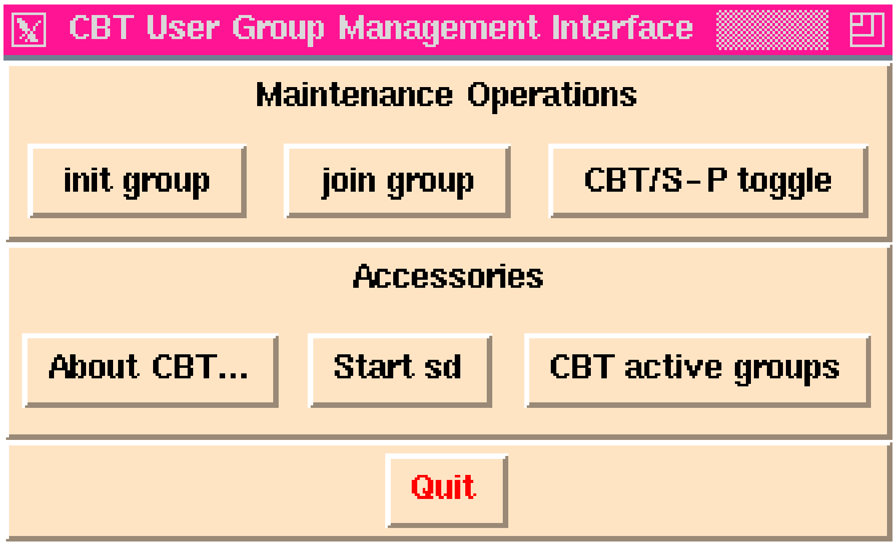

Figure 5.1: The CBT User Group Management Interface

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

All the CBT-capable routers traversed by a JOIN-ACKnowledgement change their status to *CBT-non-core routers* for the group identified by group-id. The JOIN-ACK follows the *reverse* of the path traced out by the corresponding JOIN-REQUEST. It is the JOIN-ACK that actually creates a tree branch. The tree branch is a *reverse-shortest path* rooted at the node where the corresponding JOIN-REQUEST originated.

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

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

In accordance with existing IP multicast schemes, CBT multicasting requires the presence of at least one CBT-capable router per subnetwork for hosts on that subnetwork to utilize CBT multicasting. Only one local router, the *designated router*, is allowed to send to/receive from uptree (i.e. the branch leading to/from the core) for a particular group. We therefore make a clear distinction between a *group membership interrogator* – the router responsible for sending IGMP host-membership queries onto the local subnet, and the *designated router*. However, they may or may not be one and the same. LAN specifics are discussed in sections 5.1.8, 5.1.9. and 5.1.10.

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

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

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

---

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

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

5.2.3 Tree Leaving Process

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

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

5.2.4 Tree Maintenance Issues

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

Node Failure

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

Each child node on a CBT tree monitors the status of its parent/parent link at fixed intervals by means of a “keepalive” mechanism operating between them. The “keepalive” mechanism is implemented by means of two CBT control messages: CBT-ECHO-REQUEST and CBT-ECHO-REPLY.

For any non-core router, if its parent router, or path to the parent, fails, that non-core router is initially responsible for re-attaching itself, and therefore all routers subordinate to it on the same branch, to the tree (Note: re-joining is not necessary just because unicast calculates a new next-hop to the core).

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

The significance of sending a RE-JOIN-REQUEST (as opposed to a JOIN-REQUEST) is because of the presence of subordinate routers, i.e. there exists a downstream branch connected to the re-joining router. Care must be taken in this case to avoid loops forming on the tree. If the joining router did not have downstream routers connected to it, it would not be necessary to take precautions to avoid loops since they could not occur (this is explained in more detail below).

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

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

Core Failure

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

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

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

> NOTE: It had been considered to just send a JOIN-REQUEST, rather than the apparent overhead of sending a CBT-CORE-PING first. The reason this design option was chosen was because a JOIN-REQUEST instantiates state along the path from the sending router all the way to the core (or an on-tree router on the way to it). If the target core was down, a mechanism would have to be introduced to explicitly remove that state – a disadvantage of not using the "soft state" approach. However, the *unicast* CBT-CORE-PING instantiates no such state.

Unicast Transient Loops

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

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

- a join is sent over a transient loop, but no part of the corresponding CBT tree forms part of that loop. In this case, the join will never get acknowledged and will therefore timeout. Subsequent re-tries will succeed after the transient loop has disappeared.
- a join is sent over a transient loop, and the loop consists either partly or entirely of routers on the corresponding CBT tree. If the loop consists only partly of routers on the tree and the join originated at a router that is not attempting to re-join the tree, then the JOIN-REQUEST will be acknowledged. No further action is necessary since a loop-free path exists from the originating router to the tree.

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

Loop Detection

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

We distinguish between three types of JOIN-REQUEST: active; active re-join; and non-active re-join.

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

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

A *non-active re-join* is one that originally started out as an active re-join, but has reached an on-tree router for the corresponding group. At this point, the router changes the join status to *non-active* re-join and forwards it on its parent branch, as does each CBT router that receives it. Should the router that originated the active re-join subsequently receive the non-active re-join, a loop is obviously present in the tree. The router must therefore immediately send a QUIT-REQUEST to its parent router, and attempt to re-join again. In this way the re-join acts as a *loop-detection* packet.

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

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

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

5.2.5 Core Placement

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

Careful placement of core(s) no doubt assists in optimizing the routes between any sender and group members on the tree. Depending on particular group dynamics, such as sender/receiver population, and traffic patterns, it may well be counter-productive to place a core(s) near or at the centre of a group. In any event, there exists no polynomial time algorithm that can find the centre of a dynamic multicast spanning tree [100].

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

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

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

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

Very recently, a fast and scalable algorithm for locating "distribution centres" was used in simulations [90]. This algorithm takes into account network load, and participants' resource requirements. However, it requires *a priori* knowledge of participants' locations.

5.2.6 LAN Designated Router

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

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

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

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

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

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

5.2.7 Non-Member Sending

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

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

Packets sent from a non-member sender will first encounter the corresponding delivery tree either at the addressed core, or hit an on-tree router that is on the shortest-path between the sender and the core. What happens when a CBT packet hits the corresponding delivery tree is dealt with in section 5.1.8 below.

CBT's 2-phase routing is the catalyst behind the CBT solution to "anycasting" (see section 3.6.1).

The question is begged: do non-member sending hosts need to have some CBT functionality?

The answer is that hosts should *not* be required to understand the CBT protocol, in fact, distinguishing between CBT and other "traditional" multicast protocols should be transparent to end-systems.

So, how should multicast packets reach the correct distribution tree if they are simply sent destined to a multicast address?

The answer is that the first-hop multicast router should be incorporated with functionality that distinguishes between CBT multicasts and "other" multicasts. There are two obvious possible ways to implement this functionality in routers: firstly, a separate part of the multicast address space could be set aside for CBT multicast. Whenever a multicast-capable router receives a multicast packet, if the group address in the destination field of the IP header falls within a particular range, the router could instigate a lookup in an advertising service such as "sd", which should hold (core, group) mappings. If a group address corresponds to a mapping, then the unicast core address specified in the mapping is placed in the IP header, and the router appends a CBT header behind the IP header, inserting the necessary information into the given CBT header fields.

The second approach does not involve segmenting the group address space, but would instead involve a multicast router performing a lookup for *every* distinct multicast address in an attempt to find a mapping entry in the session advertising service. If none is found, it is assumed the multicast is to be forwarded using the default multicast protocol operating in the router.

Both schemes have their advantages and disadvantages – segmenting the multicast address space is an administrative burden, whilst having multicast routers perform a lookup for every distinct multicast address has a performance impact. Both schemes require all multicast routers to have a *minimum CBT functionality*. This enables CBT multicast to remain invisible to the end-systems.

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

5.2.8 Data Packet Forwarding

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

> CBT uses the *Internet Group Management Protocol (IGMP)* in much the same way as traditional IP schemes, namely to establish group presence on directly-connected subnets, and to exchange CBT routing information. A new IGMP message type has been created for exchanging CBT routing messages. Some slight modifications have been made to IGMP specifically for CBT in order to significantly reduce *leave latency* (although the new version of IGMP can be easily adopted by other multicast protocols). This new version of IGMP is described in section 5.1.11.

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

It has been an important engineering design goal for CBT to be backwards compatible with IP-style multicasts. Until the interface with other multicast protocols is clearly defined, CBT routing information is not exchanged with that of any other schemes.

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

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

The child(ren) or parent of a CBT router may be reachable over a multi-access LAN. This is the case where a subnetwork and a tree branch are one and the same. In this case, the forwarding of the CBT-style packets is achieved with multicast as opposed to unicast. End-systems subscribed to the same group may receive these packets, but they will be discarded, since end-systems will not recognise the upper-layer protocol identifier, i.e. CBT.

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

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

The primary forwarding rules for CBT-capable routers are simple, and are as follows:

- a data packet is only forwarded if a CBT header is present in the packet.
- a data packet is *only* forwarded by a CBT-capable router if there is a forwarding information base (FIB) entry for the group specified in the CBT header of the data packet, i.e. the CBT router must be on-tree for the group. CBT does *not* forward data packets if they do not have a CBT header. The forwarding router *must* be the designated router (DR) for the group on the subnetwork.

  Exception: if a data packet *originates* on a directly-connected subnetwork, the local multicast-capable router may be required to append a CBT header, if that router has established that the specified group is a CBT group (as discussed in section 5.1.7). In order to forward such a packet, a CBT-capable router need not be on-tree for the specified group.

A FIB entry is shown below.

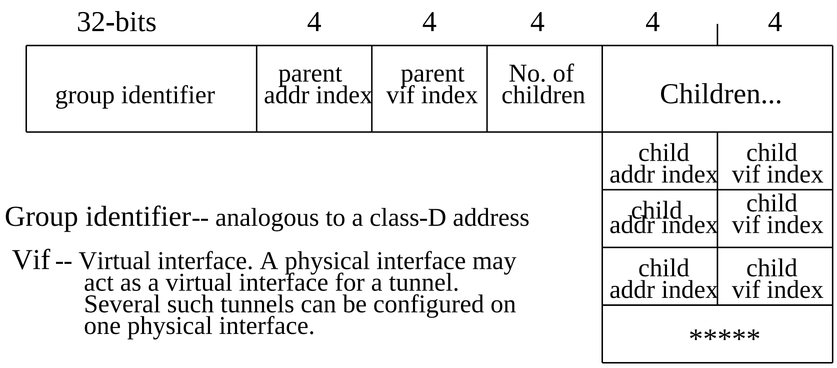

Figure 5.2: A CBT FIB entry in the author's implementation

The CBT DR for the specified group fills in the CBT and IP headers as follows: (NOTE: the fields shown in bold are CBT header fields, and are explained in detail in section 5.2.1).

- the multicast group address (group-id) is inserted into the **group-id** field of the CBT header.
- the unicast address of a core router for the corresponding group is placed in the core address field of the CBT header.
- the IP address of the originating host is inserted into the **origin** field of the CBT header.
- the **proto** field of the CBT header is set to identify the upper-layer (transport) protocol.
- the **ttl** field of the CBT header is set to the value reflected in the packet's IP header (if the packet originated locally).
- the **on-tree** field of the CBT header is set (provided this CBT router is on-tree for the specified group). It is left unset otherwise.
- the source address field of the IP header is set to the unicast address of the originating host (the IP source address changes as the CBT-style packet is passed router-to-router on a CBT tree).

- the destination field of the IP header is set to the unicast address of the on-tree neighbour (set to group address if more than one neighbour is reachable over the same interface).

- the protocol field of the IP header is set to the CBT protocol value.

- the TTL value of the IP header is set to the value specified in the packet from the original source.

  The packet is now ready for sending. Once this packet arrives at a CBT router, the packet is "reverse-engineered" (using the information carried in the CBT header) to produce an IP-style multicast for sending on directly-connected subnets with group presence. For forwarding CBT-style over on-tree interfaces, only the IP header need be manipulated, as would be expected (i.e. source address, destination address, TTL value).

What happens to a multicast packet *originated* on a subnetwork is illustrated in figure 5.3.

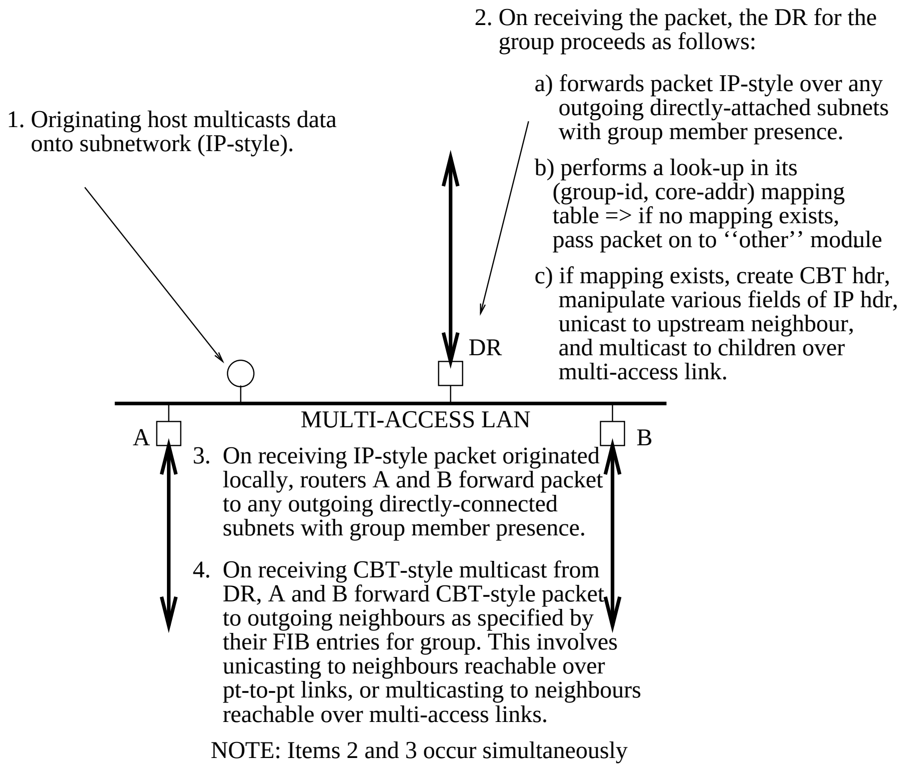

Figure 5.3: CBT Data Packet Forwarding on a LAN (originating case)

Figure 5.4 illustrates how a CBT router handles an incoming CBT data packet, and generates an IP-style multicast.

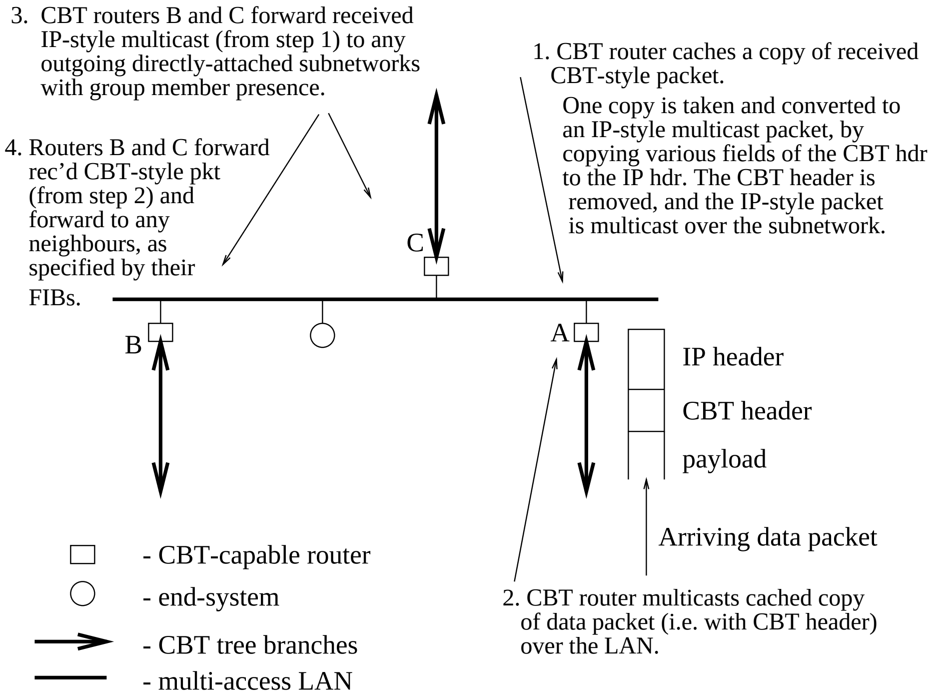

Figure 5.4: CBT Data Packet Forwarding on a LAN (receiving case)

5.2.9 Lower Group Leave Latency

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

CBT introduces an explicit *IGMP group leave* message to drastically reduce leave latency[5]. It was considered an important design goal since, over the last few years, multicast has been adopted as the preferred transport mechanism for many high-bandwidth

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

applications, including multimedia applications. Now, even comparatively resource-rich LANs are not immune to the congestion problems usually only witnessed on slower, wide-area links. It is therefore essential that, once there are no longer any receivers on a subnet, the corresponding traffic flow should cease as soon as possible thereafter. RSVP [102] uses its filter mechanism to achieve a similar effect, simply by switching its filters on or off.

The new version of IGMP is interoperable, and backwards compatible with the older version. The interoperation of new IGMP operating in CBT routers with IGMP operating in non-CBT routers on the *same* subnet, is discussed in section 5.3.3.

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

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

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

5.3 CBT Packet Formats and Message Types

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

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

5.3.1 CBT Header Format

The CBT header is illustrated below:

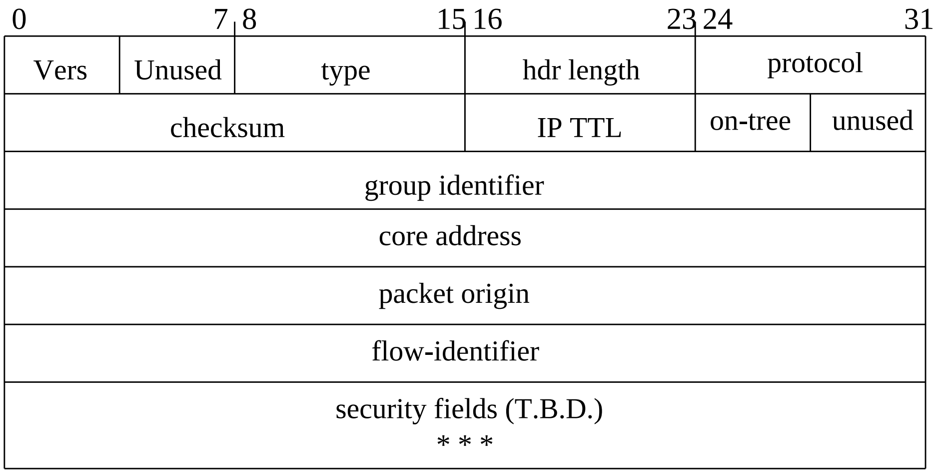

Figure 5.5: CBT Header

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

- **Vers**: Version number – this release specifies version 1.
- **type**: indicates whether the payload is data or control information.
- **hdr length**: length of the header, for purpose of checksum calculation.
- **protocol**: upper-layer protocol number.
- **checksum**: the 16-bit one's complement of the one's complement of the CBT header, calculated across all fields.
- **IP TTL**: TTL value gleaned from the IP header where the packet originated. It is decremented each time it traverses a CBT router.
- **on-tree**: indicates whether the packet is on- or off-tree. Once this field is set (i.e. on-tree), it is non-changing.
- **group identifier**: multicast group address.
- **core address**: the unicast address of a core for the group. A core address is always inserted into the CBT header by an originating host, since at any instant, it does not know if the local DR for the group is on-tree. If it is not, the local DR must unicast the packet to the specified core.
- **packet origin**: source address of the originating end-system.
- **flow-identifier**: value uniquely identifying a previously set up data stream.
- **security fields**: these fields (T.B.D.) will ensure the authenticity and integrity of the received packet.

5.3.2 Control Packet Header Format

The illustration over shows a CBT control packet header.

The individual fields are described below. It should be noted that the contents of the fields beyond "group identifier" are empty in some control messages:

- **Vers**: Version number – this release specifies version 1.
- **type**: indicates control message type.

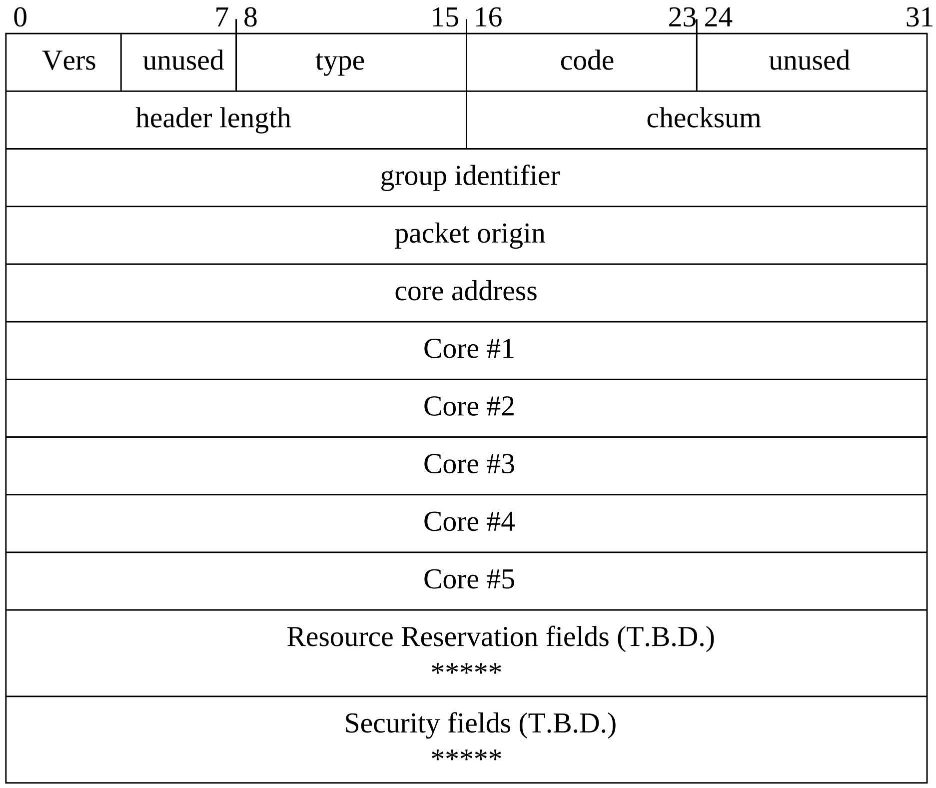

Figure 5.6: CBT Control Packet Header

- **code**: indicates sub-code of control message type.
- **header length**: length of the header, for purpose of checksum calculation.
- **checksum**: the 16-bit one's complement of the one's complement of the CBT control header, calculated across all fields.
- **group identifier**: multicast group address.
- **packet origin**: source address of the originating end-system.
- **core address**: desired/actual core affiliation of control message.
- **Core #Z**: Maximum of 5 core addresses may be specified for any one group. An implementation is not expected to utilize more than, say, 3.

> NOTE: It was an engineering design decision to have a fixed maximum number of core addresses, to avoid a variable-sized packet.

- **Resource Reservation fields**: these fields (T.B.D.) are used to reserve resources as part of the CBT tree set up procedure.

- **Security fields**: these fields (T.B.D.) ensure the authenticity and integrity of the received packet.

5.3.3 Primary Maintenance Message Types

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

- **JOIN-REQUEST**: invoked by an end-system, generated and sent (unicast) by a CBT router to the specified core address. Its purpose is to establish the sending CBT router as part of the corresponding delivery tree.

- **JOIN-ACK**: an acknowledgement to the above. The full list of core addresses is carried in a JOIN-ACK, together with the actual core affiliation (the join may have been terminated by an on-tree router on its journey to the specified core, and the terminating router may or may not be affiliated to the core specified in the original join). A JOIN-ACK traverses the same path as the corresponding JOIN-REQUEST, and it is the receipt of a JOIN-ACK that actually creates a tree branch.

- **JOIN-NACK**: a negative acknowledgement, indicating that the tree join process has not been successful.

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

- **QUIT-ACK**: acknowledgement to the above. If the parent, or the path to it is down, no acknowledgement will be received within the timeout period. This results in the child nevertheless removing its parent information.

- **FLUSH-TREE**: a message sent from parent to all children, which traverses a complete branch. This message results in all tree interface information being removed from each router on the branch, possibly because of a re-configuration scenario.

The JOIN-REQUEST has three valid sub-codes, namely JOIN-ACTIVE, RE-JOIN-ACTIVE, and RE-JOIN-NACTIVE.

A JOIN-ACTIVE is sent from a CBT router that has no children for the specified group.

A RE-JOIN-ACTIVE is sent from a CBT router that has at least one child for the specified group.

A RE-JOIN-NACTIVE originally started out as an active re-join, but has reached an on-tree router for the corresponding group. At this point, the router changes the join status to *non-active* re-join and forwards it on its parent branch, as does each CBT router that receives it. Should the router that originated the active re-join subsequently receive the non-active re-join, it must immediately send a QUIT-REQUEST to its parent router. It then attempts to re-join again. In this way the re-join acts as a *loop-detection* packet.

5.3.4 Auxiliary Maintenance Message Types

There are eleven CBT auxiliary maintenance message types:

- **CBT-DR-SOLICITATION**: a request sent from a host to the CBT "all-routers" multicast address, for the address of the best next-hop CBT router on the LAN to the core as specified in the solicitation.
- **CBT-DR-ADVERTISEMENT**: a reply to the above. Advertisements are addressed to the "all-systems" multicast group.
- **CBT-CORE-NOTIFICATION**: unicast from a group initiating host to each core selected for the group, this message notifies each core of the identities of each of the other core(s) for the group, together with their core ranking. The receipt of this message invokes the building of the *core tree* by all cores other than the highest-ranked (primary core).
- **CBT-CORE-NOTIFICATION-REPLY**: a notification of acceptance to becoming a core for a group, to the corresponding end-system.
- **CBT-ECHO-REQUEST**: once a tree branch is established, this messsage acts as a "keepalive", and is unicast from child to parent.
- **CBT-ECHO-REPLY**: positive reply to the above.

- **CBT-CORE-PING**: unicast from a CBT router to a core when a tree router's parent has failed. The purpose of this message is to establish core reachability before sending a JOIN-REQUEST to it.

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

- **CBT-TAG-REPORT**: unicast from an end-system to the designated router for the corresponding group, subsequent to the end-system receiving a designated router advertisement (as well as a core notification reply if group-initiating host). This message invokes the sending of a JOIN-REQUEST if the receiving router is not already part of the corresponding tree.

- **CBT-CORE-CHANGE**: group-specific multicast by a CBT router that originated a JOIN-REQUEST on behalf of some end-system on the same LAN (subnet). The purpose of this message is to notify end-systems on the LAN belonging to the specified group of such things as: success in joining the delivery tree; actual core affiliation.

- **CBT-DR-ADV-NOTIFICATION**: multicast to the CBT "all-routers" address, this message is sent subsequent to receiving a CBT-DR-SOLICITATION, but prior to any CBT-DR-ADVERTISEMENT being sent. It acts as a tie-breaking mechanism should more than one router on the subnet think itself the best next-hop to the addressed core. It also prompts an already established DR to announce itself as such if it has not already done so in response to a CBT-DR-SOLICITATION.

5.4 Interoperability Issues

It was a primary design goal that CBT interoperate with existing IP multicast schemes. We have already discussed in detail how CBT multicast data packets are backwards compatible with existing IP multicast schemes (see section 5.1.8). In this section we summarize this, and address other interoperability issues.

5.4.1 Isolation of CBT Routes

Multicast capability is not yet fully integrated into the internetwork infrastructure, although its use is becoming more and more widespread. As a result, each multicast scheme must establish a topology map of the multicast-capable subnetworks and distances to those subnetworks, by means of a unicast-like *routing daemon*. CBT is no different.

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

5.4.2 Tree Overlap of CBT with Other Schemes

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

Segmenting the multicast address space so that CBT uses addresses only within a certain range would not necessarily solve the problem, since, for example, `mrouted` (router implementation of DVMRP) promiscuously forwards multicast traffic until *prunes* are received for that traffic. If `mrouted`'s were specifically configured *not* to forward multicasts in a particular address range, then the problem could be solved, but this imposes an administrative burden: it requires the official partitioning of the multicast address space, and it requires users of *all* schemes to know which addresses they can and cannot use.

The problem is illustrated in figure 5.7 below.

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

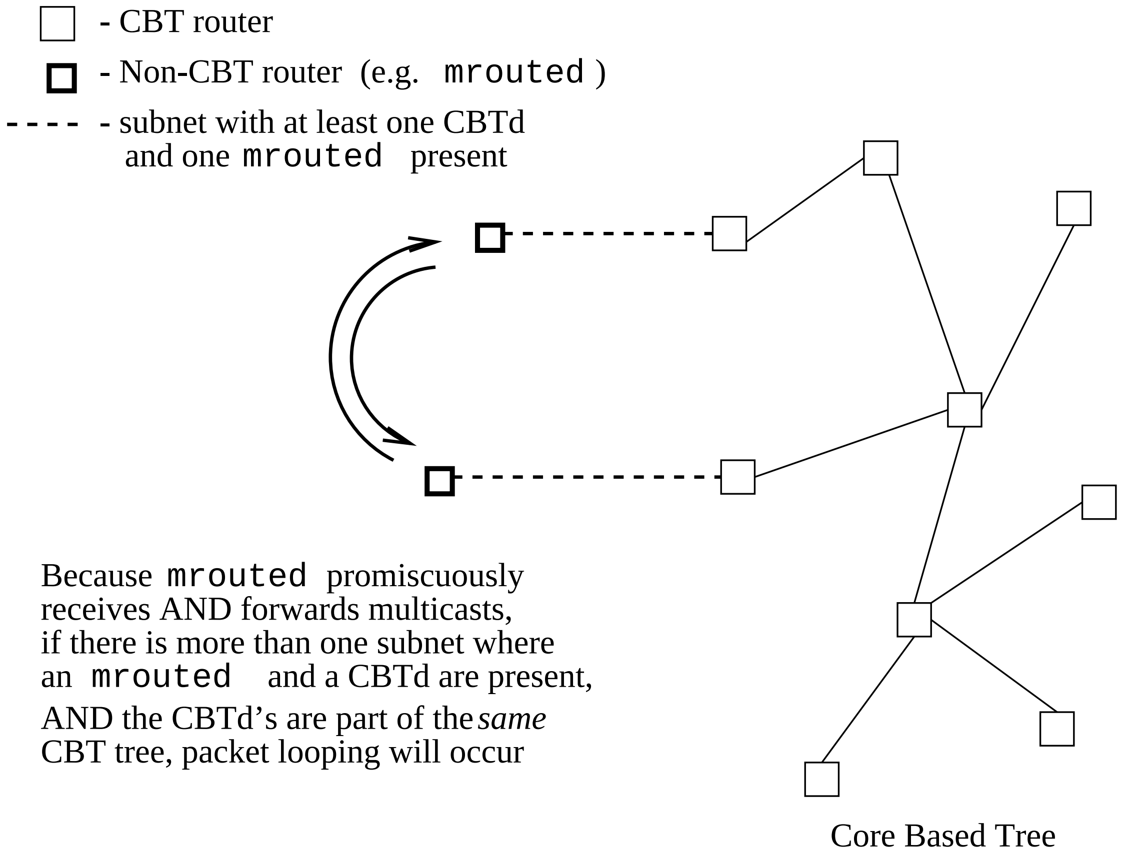

Figure 5.7: Overlapping delivery trees

5.4.3 IGMP in the Presence of Multiple Protocols

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

Multicast routers on the same subnet running the same protocol, or different protocols AND are exchanging route information (for example, DVMRP and PIM [24]), regard themselves as *neighbours*. The lowest-addressed neighbour on a subnet is implicitly elected as that subnet's *membership interrogator*, and is responsible for sending host membership queries periodically to the "all-systems" multicast group.

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

As we have said, CBT does *not* interoperate with multicast routers of other schemes, and therefore cannot partake in the election procedure in the same way. Despite this, we have devised a scheme that continues to elect the lowest-addressed router, including CBT routers. Without such a scheme, routers and hosts would suffer twice as much processing overhead of IGMP messages, and twice as much bandwidth would be consumed, through the presence of two membership interrogators.

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

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

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

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

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

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

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

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

5.5 Resource Reservation

The Internet has seen the evolution of two resource reservation protocols to date: the Internet Stream Protocol, version 2 (ST-II) [95], and more recently the Resource Reservation Protocol (RSVP) [102].

In this section we provide an overview of each of the two protocols. We proceed to discuss the pros and cons of each protocol, and finally show how CBT multicast has made provision for resource reservation and how the CBT approach provides a complementary vehicle for establishing multicast reservations.

5.5.1 The Internet Stream Protocol – version 2 (ST-II): Overview

The Internet Stream Protocol was developed in the late 1970's to support the transmission of voice over wide-area networks (WANs). After much revision in the late 1980's, a new specification was issued in 1990, and this revised version is commonly known as ST-II.

ST-II is a network layer protocol which establishes reservations by means of its own explicit control protocol (SCMP). To establish a stream, a logical connection is set up on a hop-by-hop basis *from* a sender (origin) *to* a recipient(s) (target(s)). A sender is also responsible for explicitly tearing down a flow.

Each hop of a flow has associated with it a *hop-id* – essentially a connection identifier, similar to those used in X.25 virtual circuits [17]. The origin is notified of the success or failure by any ST-II router (agent) that cannot satisfy the origin's *flow specification*. ST-II does not utilize underlying IP multicast to efficiently reserve resources between an origin and multiple targets – rather a connection is set up for each target individually. The ST-II specification does not specify *how* multiple targets are addressed – multiply-addressed messages or the use of loose source routing do not scale well. Furthermore, the ST-II specification [95] is overly complex.

5.5.2 The Resource Reservation Protocol (RSVP): Overview

Like ST-II, RSVP [102] is embodied as an upper sublayer of the IP network protocol [78]. In contrast to ST-II, RSVP adopts the "soft-state" philosophy to maintaining reservation state in RSVP-capable routers. As a result, reservation messages must be sent periodically to "refresh" state, otherwise it eventually times out. Reservations, whether for unicast or multicast communication, are made by individual *receiver(s)*, and reservations are merged so that the total resources reserved on any link are simply the maximum of individual reservations. Reservation messages create transient state as they travel on the *reverse path*, as specified by underlying uni/multicast routing, *from* a receiver *to* a sender. So that reservation messages can find a sender (as indicated by the sender's IP address contained within a reservation message), a sender transmits so-called *path* messages periodically which follow the path to intended recipient(s), as indicated by underlying uni/multicast routing. As a result of the presence of *path state*, reservation messages are routed correctly (backwards) to a sender.

5.5.3 The Pros and Cons of ST-II vs. RSVP

There are three primary disadvantages of ST-II: poor fault tolerance, sender-oriented flow reservations, and its lack of multicast support.

With regards to fault tolerance, ST data packets carry a small header (8 bytes) which is processed at each hop along a flow. The only information in the header that is relevant to the corresponding flow is the *hop-id*. It is this lack of information which makes it difficult for ST agents to recover from failures, in particular after a crash or re-boot. In such cases, the recovered node does not know which upstream agents it needs to contact, so that the source can eventually re-establish a "connection". An implementation analysis [74] concluded that this lack of ability to recover from failures is a direct result of "too much" separation between data and control information.

It is often thought that poor fault tolerance is automatically a direct result of the "hard state" (i.e. connection-oriented) approach to protocol design, but this is not always necessarily true. We think it depends both on the overall design philosophy of the protocol involved, and whether recovery mechanisms implemented as part of the control protocol fit in with this design philosophy. As far as ST-II is concerned, the sender-oriented flow reservation approach combined with the lack of ST header information makes it very difficult for ST-II to recover from failures.

ST-II's sender- and connection-oriented approach may well complement ATM as far as resource reservation is concerned, since ATM requires the setting up of a uni-directional virtual circuit prior to communication taking place.

RSVP implements minimal control information (*resv, path*, and *error* messages) which instantiates (or refreshes existing) reservation state in RSVP-capable routers. RSVP's "soft-state" design philosophy means that it is considerably more robust than ST-II as it currently stands.

There are still several problems associated with RSVP, one of them being the following: RSVP reservations between a sender and receiver(s) are made based upon the routes as specified by underlying uni/multicast. As we have said, both *resv* and *path* states require refreshing at fixed intervals, and the corresponding control messages follow the routes specified by underlying routing at the time they are sent. The implications of this are that, if underlying routes change, then so will the path(s) taken by *resv* messages. As a result, it cannot be guaranteed that the new path will have the resources available to accommodate reservations made previously on other path(s). Furthermore, admission control [43] procedures will have to be re-applied.

The RSVP designers have recognised this flaw [35], which is a direct result of the "soft-state" design approach, and are working towards a solution which involves "pinning" underlying routes. We discuss this further in section 5.4.4 below.

Whilst we have emphasized the lack of fault tolerance of ST-II, partly due to the minimal information carried in the ST header of data packets, IP version 4 (IPv4) makes no provision for carrying the RSVP *flow-identifier (flow-id)*, which uniquely identifies an RSVP reservation. As a result, RSVP implementations that sit on top of IPv4 must inspect transport layer information, such as source port and destination port, combined with information available in the network layer header, such as source and destination addresses, in order to be able to identify particular flows. This means that such implementation are likely to be much less efficient that those implementations that sit on top of IP version 6 (IPv6), which has made provision for the *flow-id* in the network layer header.

The debate whether sender-oriented or receiver-oriented reservations are best is an interesting one [74]. On the one hand, it may be reasonable for a sender to expect that the recipient(s) of its data receive it with the quality of service (QoS) the sender desires, or, either not at all, or with the next-best QoS that is available. On the other hand, particularly in the multicast case (where competition for resources will often be great, for example, due to real-time conference applications), receivers should be able to choose the quality of service they receive, especially if resource-demanding data is being received from a variety of sources. In such cases, a receiver may, for example, wish to divide the available inbound bandwidth equally between all senders.

Furthermore, receivers downstream of low-capacity links will not be capable of receiving data sent with a particular QoS, and so it is unreasonable for a sender to attempt to reserve resources when those resource requirements cannot and will not be satisfied.

A serious drawback of ST-II as compared with RSVP is its lack of specification with respect to *multi-destination* reservations. RSVP has benefited in many ways from being designed as a successor to ST-II, not least because it was designed with the ability to establish multi-destination reservations by taking advantage of underlying multicast routes.

For a receiver, the process of reserving resources for a multicast communication is no different to that for a unicast communication. The only difference in the reservation message (*resv*) is the presence of a multicast address in the **destination address** field rather than a unicast one. This *resv* message travels from the receiver to the sender, aided by the presence of *path* state, initiated by the sender, which is present along the multicast distribution tree. Hence, it is as a result of *path* messages that *resv* messages are able to find their way to a sender. *Path* messages simply take advantage of underlying uni/multicast routing to create state on a hop-by-hop basis to receiver(s). RSVP therefore, utilizes underlying routing, but remains independent of it [102].

Furthermore, RSVP has a rich functionality that has been achieved without incurring excessive protocol complexity. Unfortunately, this cannot be said of ST-II.

5.5.4 Resource Reservation: How CBT Fits In

In light of our comparison between RSVP and ST-II, it is clear that RSVP has the flexibility to satisfy the needs of today's Internet users. Furthermore, RSVP is scalable. We therefore adopt RSVP as the accepted resource reservation strategy for our subsequent discussion.

As we have said, RSVP is protocol independent, i.e. it can run over any underlying unicast or multicast protocol, or combination thereof. However, we now proceed to explain why CBT, when present as the underlying *multicast* protocol, provides a more complementary vehicle for providing multicast resource reservations.

RSVP utilizes three types of receiver-initiated reservation styles: *dynamic filter, fixed filter*, and *wildcard filter* [102]. Of relevance here is the wildcard filter, which effectively establishes a single resource "pipe" along a link. This "pipe" is shared by data packets from all sources for a session. The size of the "pipe" is simply the maximum of all the resource requests.

If RSVP is implemented over a soft-state multicast routing protocol such as DVMRP, wildcard filters, which do not specify reservations for any particular sender set (rather the reservation is for all senders), can result in *reservation loops* being formed. This is because, with soft-state multicast schemes the underlying redundant multicast topology only becomes loop-free after a router makes a forwarding decision based on an arriving packet's source address, i.e. the outgoing interfaces a multicast packet are sent over depend on a packet's source. RSVP therefore, must implement explicit loop prevention mechanisms, which add to RSVP's complexity. If CBT is the underlying multicast routing protocol however, because CBT is loop-free, RSVP need not implement such mechanisms.

Using the diagram below to explain this, assume routers *A*, *B*, and *C* are attached to hosts *h1*, *h2*, and *h3* respectively, which are participating in an audio conference.

If router *A* receives a wildcard filter reservation from *h1*, *A* forwards it to routers *B* and *C*. When *B* receives it, *B* does not know that the reservation should stop, and therefore *B* may forward it to *C*, which already has it [103].

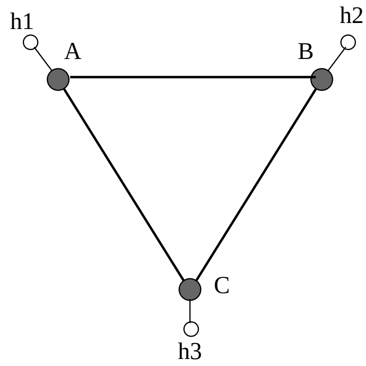

Figure 5.8: Reservation Loops in a Redundant Topology

Similar problems can occur with RSVP's dynamic filters.

In the previous section we explained two other problems associated with RSVP, namely the necessity to "pin" underlying routes, and the lack of provision in IPv4 for a *flow-id*, respectively. It may be argued that the route "pinning" problem is one of routing itself, and that the so-called route "flapping" problem should be corrected in the routing protocols.

Because CBT is a "hard state" multicast protocol, it eliminates the necessity to "pin" underlying multicast routes – CBT routes do not change unless there is an on-tree failure. An established CBT route does not change simply because underlying unicast routing says there is a better route, as do "soft-state" schemes. Therefore, resource reservation and admission control procedures need only be repeated if there is a failure.

The CBT header has made provision for a *flow-id*. This provision for reserved flows in multicast is supported by the NIMROD proposal [83] for IP next generation (IPv6), which emphasizes that the establishment of multicast routes should be closely linked with reserving resources on those routes [83]. Like RSVP, CBT is a network layer protocol, and therefore the information contained in the CBT header is readily available to routers. We conclude therefore, that the presence of CBT as the underlying multicast routing protocol enhances the "protocol independent" nature of RSVP.

Figure 5.9 illustrates the relationship between the different network layer protocols.

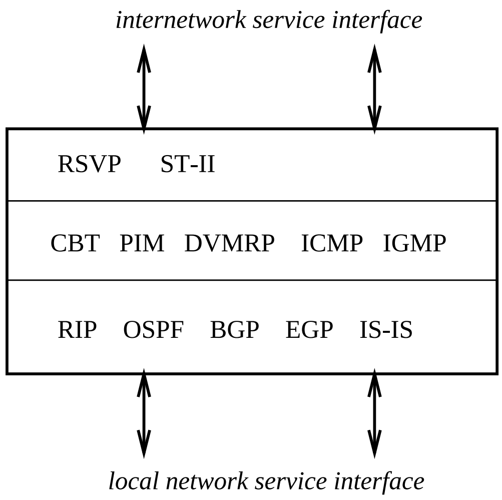

Figure 5.9: Network Layer Protocol Relationships

5.6 Chapter Summary

In this chapter we introduced the CBT multicast protocol, and described in detail the specifics of its operation, such as CBT tree-building, tree maintenance, tree teardown, LAN designated router election, and data packet forwarding and reception. Packet formats and message types were also presented.

We also discussed interoperability with existing multicast schemes, and emphasized the importance of CBT being interoperable with existing schemes. More precisely, we provided a solution to the IGMP protocol to alleviate increased IGMP message processing and bandwidth consumption on subnetworks with at least one CBT-capable router present, as well as a multicast router(s) of another scheme. This was a requirement since CBT routers do not peer with multicast routers of other schemes.

We also modified IGMP to reduce *leave latency* from around four and a half minutes (which it is currently), to around 12 seconds. This is a result of incorporating an explicit "leave" message into IGMP, which a host multicasts to the "all-CBT-routers" multicast address. The membership interrogating router on the subnetwork reacts by multicasting a group membership query to the "all-systems" multicast address, and, if no response is received within 12 seconds, the group specified in the "leave" is removed from the router's interface list, thereby preventing traffic destined for the same group from further traversing that subnetwork.

Finally, we looked at Resource Reservation, and discussed the two most prominent resource reservation protocols in existence in the Internet today: *ST-II* and *RSVP*.

Each of these protocols differs considerably in architectural design from the other. ST-II builds virtual-circuit-type, sender-initiated reservations, and lacks specification with respect to multi-destination reservations. RSVP, having benefited from "design with hindsight", adopts the soft-state approach to reserving network resources. RSVP is receiver-initiated, and supports underlying unicast and multicast routes.

Both protocols have their "domain of applicability" – ST-II complements connection-oriented networks well, such as ATM for example, whereas RSVP is well-suited to connectionless datagram networks such as the Internet.

However, one drawback of RSVP's soft-state approach to multi-destination reservations, i.e. those built upon underlying multicast routes, is the potential for an underlying multicast route to become disjoint from the corresponding reservation in the event an underlying multicast route change. In this case it would be advantageous if multicast paths changed only in response to a network failure affecting the said distribution tree. Since the CBT architecture reflects the "fixed" path approach, we concluded that if RSVP were running over CBT, the problem described above would not occur. Hence, whilst RSVP is underlying protocol independent, we conclude that CBT complements RSVP particularly well.

Chapter 6

Multicast Security

6.1 The Goals of Chapters 6 and 7

This chapter and the next both deal with multicast security, and therefore it is appropriate to state up front what each chapter is attempting to achieve.

This chapter provides what we term a *generic* security solution (in that it is totally multicast protocol independent). We present an effective, relatively low-cost solution for networks where certain perceived threats are low, whilst others are high. More precisely, in networks where the threat of *source-spoofing* and *unauthorized sending* are greater than the threat of *eavesdropping*. We purposfully avoided the use of application-level encryption to provide a low-cost solution.

The mechanisms proposed in chapter 7 are based on a different set of security requirements, namely the requirement that the routers and end-systems that comprise a delivery tree are individually authenticated as part of delivery tree set-up. The use of CBT allows for the scalable distribution of a group (or session) key, which can be subsequently used by each group member to securely and scalably distribute a sender-specific key to other group members for the purpose of application-level encryption.

Whereas chapter 6 is based on minimal use of cryptographic techniques, chapter 7 uses cryptographic techniques extensively. It is also assumed in chapter 7 that end-systems desire the use of application-level encryption for increased security, which therefore incurs increased “cost”.

In summary, each of the two chapters offers a different security solution based on a different set of security requirements.

6.2 Overview

Security issues in multicast communication have rarely been touched upon to date. We believe that wide-area multicast communication is at a substantially increased risk from specific security threats, compared with the same threats in unicast. This arises both from the lack of any form of effective *group access control*, and from the fact that multicast traffic traverses potentially many more communication links than does a single unicast communication, thereby creating more opportunity for a link attack.

From the growth in interest and usage of wide-area multicast over the last few years, we expect multicast traffic to form a significant proportion of internetwork traffic in the not too distant future. Therefore, in order to preserve the integrity of the Internet as a whole, it is critical that safeguards are employed to avert the increased threat to multicast traffic from both malicious and non-malicious attacks.

In this chapter we discuss specific threats that are relevant to multicast, and explain why they are so. We propose security mechanisms specifically for multicast groups requiring safeguards that afford protection against these threats, malicious or otherwise. More precisely, we propose a version of the IGMP protocol that can reliably employ subnet-level *group access control*. We also describe a scalable mechanism to control multicast traffic in transit that can, for example, prevent a misbehaving source from causing undue congestion over the wide-area.

6.3 Introduction

Signs are emerging that the IP community is slowly becoming more aware of the need to implement security measures, for example, through the introduction of Privacy Enhanced Mail (PEM) [54]. Also, IPv6 has been designed taking security into account [5]. More recently the Internet Architecture Board (IAB) held a workshop to clarify the requirements of security in the Internet architecture. Their suggestions and recommendations are discussed in [12]. Other signs are also emerging that the research community is beginning to see *multicast* security as an important issue that needs addressing. For example, [94] emphasises the growing demand for the integration of security services into multimedia conference applications. Also, we have recently witnessed new proposals for multicast key management and multicast key distribution [8, 42]. This overall awareness comes in light of the continuing expansion of global interconnectivity and use of the Internet by a diversity of commercial organisations which impose security requirements on their traffic because of its sensitivity.

The multicast architecture makes it inherently more susceptible, providing avenues for attack which have no counterpart in unicast. The types of security threat present in unicast are also present in multicast, but the potential risk of particular attacks are considerably greater in multicast than in unicast.

In this chapter we focus on security mechanisms that should be available to groups that specifically request it, and therefore are willing to pay the extra cost of implementing those mechanisms. We concentrate on three issues: specific threats that are amplified by multicast communication by both malicious and non-malicious attacks; how a group's susceptibility is likely to be reduced by means of *multicast group access control*; and finally, we show how *unauthorized* multicast traffic can be detected, and subsequently prevented, from causing wide-area disruption by employing controls on multicast traffic in transit[1].

6.4 Why Multicast Accentuates Security Risks

The participants of a multicast communication may, at times, wish to restrict their corresponding group membership to specific users/hosts/subnets (for example, audio and

---

[1]It should be noted that it is not our objective to define the various security services and security mechanisms currently available. To do so would only detract from our discussion, although we will do so on occasions where we consider it relevant and necessary, and for reasons of brevity. For a comprehensive guide to security mechanisms and services, and a discussion of network security, the reader is referred to [45], [97], [50], [54], [71], [46], and [40].

video conferencing), but there currently exist no policing mechanisms that can implement group restrictions, other than employing end-to-end encryption. Full packet encryption imposes a burden on end systems, but a communication may not be considered sensitive enough to warrant its use. However, it may be the only way of preventing a user from randomly joining a multicast group and "listening in". The above is a scenario where end systems will be able to benefit from multicast *group access control*, since it helps constrain a group's accessibility without necessarily requiring end-to-end encryption. Group access control does not, however, provide any protection against link attacks.

Without group access control, any user anywhere in the internetwork can receive data from any multicast group at any time by simply becoming a group member. More serious is the ease with which any user can mount a denial of service attack, through malicious intent or otherwise[2], that can affect not only a group's receivers, but a potentially large proportion of the internetwork. This danger arises both from group members' ability, and non-group members' ability, to randomly send data to any group address at any time. Moreover, the use of UDP as IP multicast's transport protocol means that there are no in-built protocol mechanisms to detect/prevent congestion [99] caused unintentionally by a sender.

Furthermore, the number of communication links traversed by wide-area multicasts can potentially be far greater compared with a single unicast, where the communication path is a collection of links and nodes between just one source and one destination. Therefore, multicast intrinsically creates more opportunity for traffic interception.

We therefore conclude that wide-area multicast communication is at a substantially increased risk from certain threats, namely the unauthorized *observation* of multicast traffic (i.e. *listening*), the subsequent threats that can come about as a result of being able to snoop traffic, and *denial of service* attacks, which arise from unauthorized or "unintentional" *sending* to a group.

We summarize our conclusions below:

- the participants of a multicast communication currently have no mechanisms at

[2]Unintentional disruptions are often caused by users untrained in the use of applications that are new to them.

their disposal by which to impose group membership restrictions, thereby rendering groups easily accessible. Consequently, multicast provides a vehicle for *attackers* to pose as legitimate group members.

- the IP multicast address space is a well-known contiguous part of the unicast IP address space, making it easy for an *attacker* to locate, and become part of, a multicast group at random.
- there exist no mechanisms that can prevent either group members, or non-group members, from sending (possibly spurious) multicast data to a group, which can result in denial of service (usually due to congestion) over the wide-area. This is of particular concern to multicast because of the "multiplier effect" of multicast packet distribution, not present in unicast.
- the use of UDP as IP multicast's transport protocol means that there are no in-built protocol mechanisms to prevent congestion caused unintentionally by a multicast sender.
- multicast intrinsically creates more opportunity for unauthorized interception of multicast traffic.

With regards to this latter point, in order to look more closely at the potential for traffic interception, it is worth looking at the type of delivery tree built by the multicast algorithm. These trees are either *source-based*, whereby a source tree is built from an active sender, or *shared trees*, which disseminate multicast traffic using one delivery tree spanning all group members.

For source-based trees, if all receivers are active senders at some point over the lifetime of a particular group, then a fully interconnected mesh of communication paths will have resulted over that group's duration. As a result, an attacker has a potentially very large selection of communication paths over which to mount an attack.

On the other hand, multicast algorithms that build just one *shared* delivery tree per group [7], offer in comparison, less opportunity for such an attack[3]. Thus, an attacker is more constrained in the number of communication paths over which to mount an attack.

[3] We are certainly not implying shared trees are not a problem – they still constitute a much larger threat than does a single unicast communication.

6.5 Specific Types of Threat

Security attacks are classified as *active* or *passive*. Attacks that result in information release are *passive*, and those that involve message modification or denial of resources are *active* [97]. Specifically, multicast is at an increased risk from the following types of attack (we explained *why* multicast is particularly susceptible to attack in the previous section):

- **Traffic Observation**[4] (passive).

  Often called *eavesdropping*, traffic observation concerns the interception of information between communicating parties, thereby resulting in the disclosure of information such as traffic type, content, frequency, presence/absence, amount.

- **Denial of Service** (active).

  Certain multicast applications impose demands on network resources, such as bandwidth, which is often in short supply.

  It should be noted that any unauthorized sending of multicast data can be construed as a denial of service attack.

These attacks pose a serious threat to all Internet communications, and arise partly as a result of the openness of the multicast paradigm, i.e. the ability to join a multicast group without being subject to any form of membership policing, as well as the ability of any member or non-group member to unreservedly multicast data to a group. As a result of traffic observation from unauthorized group membership, multicast is also at risk from other attacks, some of which we mention below.

- **Masquerading** (active).

  Often called *spoofing*, masquerading concerns the issuance of information, the receipt of information, or the acquiral of access rights, by a *principal*[5] using an identity other than its own.

[4] We use the term *traffic observation* to combine the terms *traffic analysis* and *data interception*. These are often classified as separate attacks. However, for the purpose of our discussion their difference is subtle enough to treat them jointly.

[5] The term *principal* is often used when discussing security. A principal is any entity that is the subject of a particular security procedure, i.e. it can be computers, users, applications.

It remains relatively easy for a user to insert a bogus source address in the network-layer header of an IP packet. Hence, an intruder has the ability to pose as some other legitimate multicast group member.

Multicast algorithms based on *reverse-path forwarding* form an implicit safeguard against masquerading, since multicast packets that arrive on an interface other than that which is used to reach the source, are discarded. Therefore, an attacker must be on the shortest reverse-path for the authentic source, from the perspective of a multicast router, for the attack to succeed.

- **Malicious Replay** (active).

  A *replay* attack is the result of an intruder having intercepted information, and replaying it at a later time. This, therefore, can result in denial of service. Denial of service affects a greater proportion of the internetwork than it does in unicast because of the "multiplier effect" of multicast packet distribution.

- **Repudiation**.

  Repudiation is the denial by a principal of having participated in some or all of a particular communication. This can arise as a direct result of masquerading. In unicast only one recipient is affected. In multicast, the number of recipients affected is proportional to the number of group receivers.

To summarize, it is essential to have the ability to control group access, and for a group's receivers to be sure of the *integrity, authenticity*, and *freshness* of the data they receive. At the same time, it is often critical to certain applications that no unnecessary delays are imposed on traffic forwarding, for example, real-time traffic such as voice.

6.6 Security Framework

In order to be able to successfully implement security mechanisms and services, it is necessary to have a security infrastructure or framework which includes provision for key management, upon which security services are based. The X.509 Authentication Framework is one such example [46].

From a global perspective, *asymmetric*, as opposed to *symmetric*, cryptotechniques are more practical because public keys are far easier to manage, and there is no requirement for secure key distribution [55]. We provide a more detailed motivation for using a *public-key* management infrastructure in section 6.7.

Currently, the Internet does not have a key management infrastructure, although it has been proposed to make public keys (created off-line) available through directory service, such as DNS [31] and X.500 [56].

A *Certification Hierarchy* comprising multiple *Certification Authorities (CAs)* was proposed as the certificate management architecture for the OSI security framework, which is based on the CCITT's security recommendation X.509 [46]. This is a dedicated hierarchy whose elements (CAs) are assumed trusted and physically secure. The purpose of a CA is to make public keys available in the form of *certificates*, which are digitally signed by a CA. Certificates are made available to subscribers, who may choose to publish them in a directory service. As the certification hierarchy is trusted, the recipient of a certificate can be assured of the validity, integrity, and authenticity of the key contained within it.

There are two other ways to generate key pairs: a user can generate its own, having the advantage that a user's key is never divulged to another entity; a third party could generate the key pair, requiring the secret key to be transmitted securely to the user.

6.7 Approaches to Multicast Security

6.7.1 Overview

There are two approaches to implementing security measures, namely *prevention-based*, and *detection-based*. We assume these terms to be self explanatory.

With regards to policing multicast group membership, we adopt the prevention-based approach. For controlling multicast transit traffic, we adopt a detection-based approach, with subsequent prevention. By combining these two approaches, we can enforce restrictions on multicast group memberships and traffic at a relatively low cost.

From this point on, we assume the presence of a *certification hierarchy* which is responsible for the issuance and maintenance of public key certificates. We assume that these public key certificates are available through directory services.

6.7.2 Authorization Infrastructure

We propose the use of *Authorization Servers (ASs)* which are logically "linked" together by being part of a single, well-known CBT multicast group.

The reason ASs belong to a CBT multicast group tree is because the network nodes (routers) that contact an AS are able to take advantage of CBT's *2-phase routing* available to CBT non-member senders: non-members *unicast* an authorization query to a single core address. This packet should hit the tree at the nearest point to the sending router. The on-tree node can then respond to the query – it need not multicast queries to other nodes on the CBT tree. In this way, CBT provides an effective solution to *anycasting* [73]. Our example demonstrates how CBT "anycasting" is used for *resource discovery.* A more complete discussion of CBT anycasting is given in chapter 3.

As the security of our scheme is based on the correct operation of all authorization servers in the AS group, we assume they are maintained and monitored sufficiently so as not to be prone to malfunction or misbehaviour.

6.7.3 Cost-Reducing Mechanisms

Security always comes with some additional cost to the systems that implement it. With regards to end-systems, we advocate constraining this cost to only those end-systems that belong to restricted groups.

Secure IGMP, which we propose in section 6.8.3, requires interaction with an Authorization Server (AS). However, this is constrained to just *one* interaction per group per interface – no interaction with the AS is required pertaining to a received IGMP group membership report on a particular interface so long as there is group member presence on the same interface.

With regards to *multicast transit traffic control*, this cost manifests itself in the form of an *authorization stamp* (described below) included in each packet, and, if confidentiality is required, packet encryption. However, the sensitivity of each communication should determine whether the cost of incorporating security on a packet-wise basis outweighs the threat. End-systems belonging to non-restricted groups should have no additional processing burden placed upon them.

Routers are not involved in any security processing since all security checking is carried out by an Authorization Server. With regards to multicast transit traffic control, routers merely *decide* which packets should be checked. Constraining trust to the AS group, and requiring routers to store only the public keys of the local Certification Authority (CA)[6], and the public key of a local Authorization Server (AS), has obvious advantages.

Furthermore, security checking has no effect on packet forwarding delay, since all security checking is done *post facto*, i.e. after forwarding. Such after-the-fact security control has been termed *optimistic authorization* by Cheriton in [18]. This approach is sometimes used in high-speed networks where forwarding delay is a critical factor.

Our proposal for implementing security controls on transit traffic (see section 6.10) in order to detect, and subsequently prevent, attacks such as denial of service, involves appending an *authorization stamp*[7] to each data packet. This stamp is subject to security checking, and should be available to routers as network-layer information. Therefore, so as not to compromise layering principles with respect to routing, we decided that the stamp should travel either as an IP (version 6) option in a packet's network layer header, or in a separate header immediately behind the IP header, but which is accessible to routers.

IP version 4 (IPv4) routers are not usually optimized to handle IP options, and therefore they are usually additional overhead. The IP version 6 (IPv6) proposal [37] has been designed so that different *payloads* can be optionally inserted into a packet. An example of this is the IPv6 *authentication header* [4]. When an IPv6 router parses a datagram, a pointer in the packet indicates the payload type to be parsed next. Therefore, in IPv6, the presence of the security stamp should not be problematic.

As we have said, multicast transit traffic control is instigated by routers. Controlling traffic on a per-packet basis is usually neither necessary nor desirable, and there exists a spectrum of packet checking possibilities, as outlined in [34]. Routers are not involved in explicitly performing security controls – they merely *decide* which packets should be checked. All security checking is carried out by an AS, which is dedicated to this purpose.

[6] So as to be able to verify users' public keys.

[7] A stamp is similar to a *visa* as used in the Visa protocol [96].

However, routers incur the cost of the generation/reception and processing of a small number of messages that result subsequent to being notified of a traffic violation, but this is insignificant. Furthermore, traffic controls are carried out on a post-facto basis, thereby incurring no extra delay on packet forwarding in routers.

We now look at multicast group access control, and multicast transit traffic control, in more detail.

6.8 Motivation for Proposed Key Management Architecture

It is well-known that asymmetric cryptographic algorithms, such as RSA, cannot be implemented in software as computationally efficiently as symmetric algorithms such as DES[8].

However, the advantages that can be gained from using asymmetric algorithms are significant: they tend to be much more resilient to attack; public keys are either well-known or easily accessible, obviating the requirement that routers and hosts be entrusted with secret key information; the public-private key paradigm provides a natural vehicle for the provision of digital signatures which in turn provides a means for message origin authentication and message integrity.

The use of symmetric cryptographic algorithms is more difficult with multicast in terms of key management and distribution. The author presents an effective solution to multicast key distribution in chapter 7.

In multicast, a security association may be created for a group, in which a particular *security association identifier (SAID)*[9] is shared for all communications within the group, i.e. all group communications are authenticated using the same security configuration

[8]There are additional political issues, as documented in [12] that preclude public key cryptosystems from being widely used, but we consider these orthogonal to their technical merits.

[9]The concept of SAIDs was introduced in the development of security for IPv6 [5]. A SAID is an identifier shared between communicating parties, and is carried in the packets sent between those parties. Its purpose is to uniquely identify a security association, and is used by routers to index into a *security association table*, which holds parameters, keys etc., belonging to a particular association.

parameters, such as algorithm and key etc. In this case, a receiver cannot authenticate a particular sender since there is no information present that can reliably distinguish a particular sender. The only assurance possible with a shared SAID is that the multicast sender is indeed a group member.

It is possible to use a separate SAID for each sender of multicast traffic, whereby each sender could be authenticated, but this has serious scaling drawbacks where symmetric algorithms are concerned. The problem lies in the fact that symmetric algorithms rely on *shared secrets*. In multicast, either some *same* secret has to be shared between a group's senders and receivers, or there must exist a different shared secret between each sender and receiver.

As for the former case of sharing a group secret, the problem here lies in securely distributing a group key to all receivers in a scalable fashion. This is known as the *multicast key distribution problem*.

The latter case of sharing a secret between each sender and receiver has a complexity of $O(n^2)$, and has serious scaling implications in two areas: firstly, the SAID parameters need to be encrypted n times, once for each recipient, by whichever network entity is responsible for key distribution, for example, a *Key Distribution Centre (KDC)*; secondly, the authenticator included in a data packet is proportional in size to the number of receivers – there is a separate authenticator for each receiver. This has implications both on packet processing time, and bandwidth consumption.

It should be obvious from the above discussion that our proposals for multicast group access control and multicast transit traffic control are only practically implementable in an internetwork with a public-key management infrastructure, such as that proposed in [31].

6.9 Multicast Group Access Control

6.9.1 Overview

Multicast group access control is the ability of a user, namely the *group initiator*[10] to specify restrictions on group membership as deemed appropriate. Mechanisms, such as keying strategies and authentication algorithm(s), must be in place that can enforce those restrictions. Group access control reduces significantly the degree of threat to multicast by constraining group *accessibility*. However, group access control cannot help prevent link attacks. We are proposing that group access control be at the granularity of *subnet-level*.

We have already assumed the presence of a hierarchy of Certification Authorities (CAs), and Authorization Servers (ASs) logically linked in a CBT multicast group, in sections 6.5 and 6.6.2, respectively. For simplicity, we also assume that an Authorization Server *creates* and *maintains* multicast certificates. Multicast certificates are created subsequent to an AS receiving a group *access control list*. After creation, a certificate is signed by the AS using the AS group key, and multicast to the AS group.

Multicast group access control requires modification of the *Internet Group Management Protocol (IGMP)* [26].

Our multicast group access control mechanism is *stateless*, and multicast *algorithm independent*.

Whether multicast group access control is used is the responsibility of a group's initiator. An initiator is responsible for communicating a list of authorized (or unauthorized) subnets, and the multicast group address, to an AS prior to a group being advertised[11]. For our discussion, let's use group identifier G. Note that the information sent contains additional information that allows an AS to perform *one-way strong authentication*[12]. This information is called the *authentication token* [46].

The AS authenticates the initiator. The AS subsequently composes a *multicast certifi-*

[10] It is important that only one user has the ability to specify group restrictions.

[11] And thereby prior to it becoming operational.

[12] *Strong authentication* is "authentication by means of cryptographically derived credentials." [46]

cate for group G, provided no multicast certificate already exists for G, and multicasts the certificate to the rest of the AS group. If a certificate does exist, only the group initiator, as specified in the certificate, may modify it. The contents of a multicast certificate are shown below. Many of the fields are similar to those of public-key certificates as discussed in PEM [54] and X.509 [46].

- **Version Number**
- **Serial Number**
- **Issuer (AS) name**
- **Validity period**
- **Multicast Group Identifier**
- **Multicast group initiator Distinguished Name (DN)**
- **Group Inclusion (or Exclusion) List**
- **Sender List**
- **Digital signature** (created using the AS group key)

A multicast certificate, in the form shown above, is never disclosed outside the AS group.

Below we define only those attributes of the multicast certificate that are relevant to our discussion (the remaining attributes are largely self explanatory).

- **Multicast Group Identifier.**
  An IP multicast group address.

- **Multicast group initiator Distinguished Name (DN).**
  A DN is an X.500 directory system concept, and is used to uniquely identify each user. In IP, the equivalent of a DN is a Domain Name System (DNS) name.

- **Group Inclusion (Exclusion) List.**
  The inclusion list includes those subnets, together with their associated subnet masks (due to CIDR [84]) on which end-systems are authorized to become group members.

An exclusion list includes those subnets on which end-systems are *not* authorized to become group members.

- **Sender List**

  A sender list is a list of group members authorized to send to a group. This may not be relevant for some groups, but it should be optionally available.

  NOTE: As a result of constraining who can *send* to a group, the group is no longer classed as an *open* group [26], but rather a *closed* group.

An example of a scenario requiring subnet-level group access control is as follows: a seminar series is to be audio- and video-cast to participants at selected universities around the world. Each university wishing to participate is required to pay a registration fee of £100. After submitting a list of campus subnets with the registration fee, any hosts on the specified subnets may send/receive to/from the corresponding group when it becomes operational.

6.9.2 Group Access Control – Details

Potential members of restricted groups are likely to know that a group is restricted, and therefore adopt the correct joining procedures. We expect there to be a "secure" command line option when joining a group, or, if the group joining procedure is via a graphical user interface, a corresponding selector. End-systems bypassing the correct procedures will not succeed in joining a group, unless that group is unrestricted, due to the protocol exchange between the membership interrogating router and an AS. Whenever a client wishes to join a restricted group, it must first send a *user-request* to an AS, to become authorized. The following occurs:

- ◇ The authenticity and integrity of the client's request is verified by the AS by means of one-way strong authentication.
- ◇ If the AS has a multicast certificate for group G, as specified in the request, it attempts to match the subnet of the request packet's source IP address with one in the *subnet* inclusion/exclusion list for the group. This is done by masking the packet's source address with the mask of each subnet in the list.

◇ If a match is found in the corresponding *inclusion* list, then a multicast *authorization stamp* is created by the AS, and returned to the client. An authorization stamp is comprised of the following:

1. the DN of the requesting client.
2. the IP source address of the requesting client.
3. a *timestamp* (together with a lifetime over which it is valid), for use in data packets. Timestamps are not directly relevant to group access control, and are discussed in section 6.10.1.
4. the *digital signature* of the AS (Note: the digital signature is a signed message digest, computed using the MD5 algorithm [85]).

The authorization stamp is encrypted using the public key of the client, and is returned to the client.

◇ If a match is found in the AS's *exclusion* list for the group, it returns a *rejection message* to the client, indicating to the client that it does not have authorization to join the named group. This message is digitally signed by the AS.

◇ If no multicast certificate is found for G, the group is assumed to be unrestricted. The AS returns a message to the client, indicating that the group is unrestricted. It is digitally signed by the AS, but the message is *not* encrypted, since secrecy is not seen as being necessary where unrestricted groups are concerned. The data packets of unrestricted groups need not carry an authorization stamp.

6.9.3 Secure IGMP

If a client is successful in obtaining an authorization stamp for group G, or if the group is unrestricted, then it may proceed to join group G. The next step in the joining process involves the host sending an *IGMP report* to the client's local designated router (DR). If the group being reported has restricted access, the authorization stamp must be included in the membership report. The report is verified by the DR, as follows, *before* adding group G to its corresponding interface list:

◇ the DR sends a *router-request* to the AS for group G, which includes the following:

1. the group address included in the membership report, and, if present, the authorization stamp.
2. the IP interface address over which the group report was received.

The router-request is digitally signed by the sending router.

- ◇ the AS verifies the router-request using one-way strong authentication. If no authorization stamp is included in the router request, the AS need merely confirm that there exists no multicast certificate for the specified group.

  If an authorization stamp is included in the router request, the AS also verifies its own signature on the authorization stamp. Using the multicast certificate for G, the AS also attempts to match the router's interface address included in the request with a subnet in the inclusion (exclusion) list for G.

  If a match is found, or if the group is unrestricted, the AS returns an *IGMP-accept* (*IGMP-reject*) message, which is digitally signed by the AS. This message contains a *nonce* (timestamp) to ensure freshness and prevent replay.

- ◇ on receipt and successful verification of an IGMP-accept message from the AS, provided the router successfully verifies the signature, the router adds group G to its corresponding interface list, thereby allowing multicast traffic for G to flow onto that subnet.

Note that only *one* interaction with the AS is necessary per group per network interface (or virtual interface), since once a group has been added to a designated router's interface group list, it makes no sense to query the AS again regarding a received group membership report for the same group. This is because traffic for the corresponding group has already been authorized to flow over the subnetwork. Thus, a subnetwork's designated router is only required to send an IGMP router-request to an AS if it has no registered group presence for the corresponding group on the interface over which a group report was received.

The diagram below illustrates the secure IGMP process:

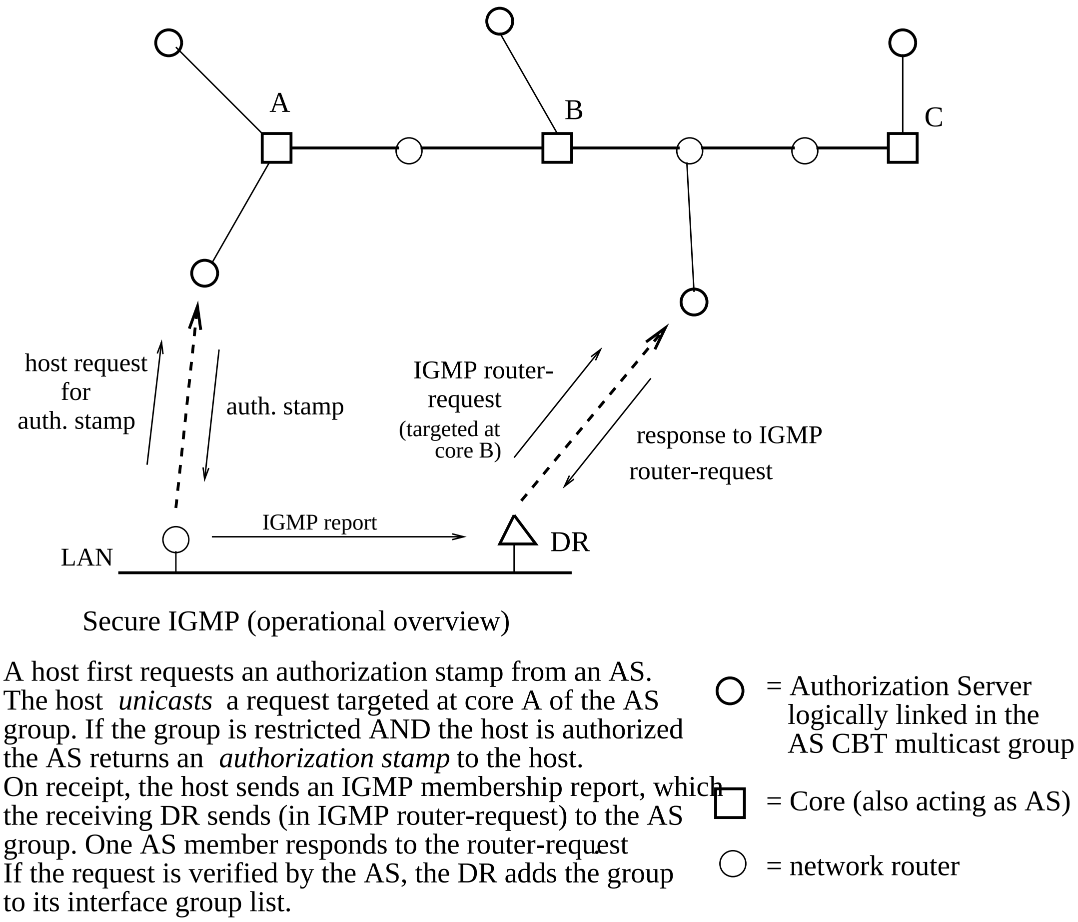

Figure 6.1: Secure IGMP - Model of Interactions

6.9.4 Authorization Stamp Creation – Message Contents

The following **terminology** will be used throughout:

- PK_A indicates the *public key* of entity A.
- SK_A indicates the *secret key* of entity A. The secret key can be used by a sender to *digitally sign* a digest of the message, computed using a strong, one-way hash function, such as MD5 [85]. It is assumed that the original message contains the identity of its originator so that a recipient can establish the correct key with which to verify the digital signature.

- Unencrypted messages will appear enclosed within square brackets, e.g. *[X, Y, Z]*. If a message is *digitally signed*, a superscript will appear outside the right hand bracket, indicating the secret key of the message signer. Encrypted messages will appear enclosed within curly braces, with a superscript on the top right hand side outside the closing curly brace indicating the encryption key, e.g. $\{X, Y, Z\}^{PK\_A}$.

- a *token* is information sent as part of a strong authentication exchange, which aids a receiver in the message verification process. It consists of a *timestamp, t* (to demonstrate message freshness), a *random, non-repeating number, r* (to demonstrate message originality), and the unique *name* of the message recipient (to demonstrate that the message is indeed intended for the recipient). A *digital signature* is appended to the token by the sender (which allows the recipient to authenticate the sender). The token is as follows:

  $[t_A, r_A, B]^{SK\_A}$ – token sent from A to B.

- $A \longrightarrow B$: – denotes a message sent from A to B.

- *grpAddr* – denotes a multicast group address.

- $srcAddr^A$ – denotes the IP (host) source address of entity A.

- $recvAddr^A$ – denotes an IP interface address of entity A.

- $authStamp^A$ – denotes the authorization stamp of entity A.

The message contents involved in the creation of a multicast authorization stamp for host A are as follows:

- $A \longrightarrow AS : [token^A, grpAddr, srcAddr^A]^{SK\_A}$

  srcAddr must match the IP source address of the message itself.

- $AS \longrightarrow A :$
  $[A, srcAddr^A, t^{data}, lifetime]^{SK\_AS}$

  t^{data} and *lifetime* are the authorization stamp's timestamp and lifetime, respectively.

Secure IGMP

- $A \longrightarrow DR:$
  $[group\ membership\ report + authStamp^A]$
- $DR \longrightarrow AS:$
  $[token^{DR}, grpAddr, authStamp^A, recvAddr^{DR}]^{SK\_DR}$
  (IGMP router-request)
- $AS \longrightarrow DR: [IGMP\_ACCEPT, t^{data}]^{SK\_AS}$
  (on successful verification of IGMP router-request)
- $AS \longrightarrow DR: [IGMP\_REJECT, t^{data}]^{SK\_AS}$
  (on unsuccessful verification of IGMP router-request)

6.10 Multi-Access LANs

Multicast group access control at the granularity of subnetwork works fine for *leaf subnets*, i.e. those that have no downstream routers attached to them. The preclusion of multicast packet flow across these subnets has no effect on downstream portions of the delivery tree.

However, in any delivery tree, it might be that there exist multi-access LANs, which act as transit networks on the delivery tree. It is essential, therefore, that unauthorized transit networks which form part of a delivery tree nevertheless allow multicast traffic to flow over them.

If multicast traffic flows across these subnets as is, then obviously group access enforcement becomes problematic with regards to these subnetworks. We therefore propose that multicast traffic is *tunnelled* across unauthorized multi-access LANs that form part of the delivery tree.

This may not be as straightforward as it sounds: tunnels need to be manually configured at present; group access control should also have the property that a group access control list can be modified over the lifetime of a group. Thus, a particular subnet may become authorized or unauthorized at some point, and the implications of this are that tunnels may need to be re-configured "on-the-fly". If the tunnel were a point-to-point encrypted tunnel, it *might* be less of a burden to do tunnel management if the encapsulating security protocol had appropriate features [3].

At present, we cannot otherwise offer an effective solution to the above problem. However, it may be worth noting that, for the most part, multicast delivery trees are made up of tunnels and point-to-point links, and so our multicast group access control scheme should hold, without complication, for the majority case.

We consider refined group access control techniques, and finer granularity group access control, for example, at the level of an individual user, to be topics of ongoing/future research.

6.11 Multicast Transit Traffic Control

6.11.1 Overview

We now focus on how an unauthorized *sender*[13], sending multicast data to a *restricted* group, can be detected, and subsequently prevented, from causing disruption over the wide-area.

It should be noted that only *multicast data* packets are subject to transit traffic controls.

Multicast data packets of restricted groups each carry a *timestamp*. Timestamps are used to prevent replay attacks. They are not deemed relevant to group access control – its concern is with *who* or *what* is permitted group access, rather then *when* a group is considered active.

Timestamps are issued by the entity that creates authorization stamps. They are issued to members of restricted groups, and inserted into an authorization stamp, together

[13]An example of what we mean by this is – a sender maybe transmitting multicast data at such a high rate that other multicast transmissions are "locked out". This is, of course, a result of multicast capability not yet being available as part of the Internet infrastructure, and therefore the overall available transmission capacity of the existing multicast infrastructure is somewhat reduced.

with a *granularity value* – effectively, the lifetime of the timestamp. Since authorization stamps are verified only by the AS, there is no necessity for transit routers to maintain any synchronisation regarding current time. Hosts should request a new authorization stamp shortly before their current stamp expires. This should not be so frequent so as to become a performance burden, but the re-issuing interval should be chosen carefully so as to balance performance and protection. If replay is considered a serious threat however, then a host may request a new timestamp more frequently from an AS.

We adopt the *soft-state* approach to multicast transit traffic control, i.e. state is transient, but this state is limited to just one router per "misbehaving" source. This router is that which is directly connected to the source.

Therefore, our proposal for multicast transit traffic control is low-cost in terms of processing and storage, and does not infringe on the scalability of any multicast algorithm.

If an unauthorized packet is detected, the AS informs the router that invoked the packet check so that it can take preventative action with regards to traffic with the misbehaving (source, group) combination. This involves reliably unicasting a choke-like packet, called an *alert packet*, one hop back towards the specified source. Each router receiving an alert packet reliably unicasts it one hop back towards the source, until it finally reaches the router that is directly connected to the subnet of the source. This router obviously need forward it no further. However, it must store the alert packet for some specified period of time. This serves to *prevent* further multicast packets from the source to the corresponding group emanating beyond the source's subnet, thereby preventing further disruption over the wide-area. Once the problem is contained to the source's subnet in this way, it can be dealt with as a local matter by the local system administrator.

Failure to receive an acknowledgement of an alert packet could be an indication that the previous-hop router is the source of the problem, or that the router itself is misbehaving. If no acknowledgement is received for an alert packet, the sending router discards all subsequent packets from that router destined for the corresponding group. If indeed, a previous-hop router *is* misbehaving, a subsequent check by any router on the source's side of the misbehaving router will result in another alert packet eventually being generated and forwarded towards the source.

6.11.2 Multicast Transit Traffic Control – Details

We proceed to describe the details of multicast transit traffic control:

◇ All multicast data packets sent within restricted groups must carry an authorization stamp. A multicast sender digitally signs each packet, using MD5 and its private key. The 128-bit message digest computed by MD5 covers the entire network layer and upper-layer data, except fields which change in transit.

◇ Any multicast packet that is subject to transit control is forwarded immediately, and a copy cached. The controlling router encapsulates the cached packet inside a *transit-router request*, and sends it to the AS to be verified.

◇ The AS extracts the multicast packet encapsulated in the router-request. If an authorization stamp is present in the request, it first verifies the digital signature covering the authorization stamp (included in the packet's network layer header), and verifies the *freshness* of the packet by inspecting the timestamp and timestamp granularity fields. The AS then verifies the signature covering the entire packet, using the public key of the user specified in the authorization stamp. Provided all of the above checks are successful, or the group is found to be unrestricted (in which case no authorization stamp will be present and no multicast certificate will exist for G), the AS returns a *control-verified* packet to the requesting router. If any single check fails, the AS immediately returns a *control-failure* packet to the requesting router.

◇ On receipt of a *control-verified* packet, a router need take no action.

◇ If a *control-failure* packet is received from the AS, the receiving router proceeds as follows:

NOTE: both *control-verified* and *control-failure* packets carry a *nonce* (timestamp) to prevent replay attacks.

1. unicasts reliably, an *alert* packet one hop back towards the misbehaving source, specifying the corresponding (source, group) pair. An alert packet has a TTL of 1. Since alert packets are transmitted reliably, no long-term state is incurred by transit routers *except* for the router that is directly connected to the subnet of the source.

2. routers that receive an alert packet, are obliged to forward it to the next hop towards the specified source. The acknowledgement of receipt of an alert packet from a one-hop-back router is assumed to be an acknowledgement that it has been forwarded further.

3. when the router directly connected to the specified source receives an alert packet, it stores the information contained in it in a relevant database so as to prevent further traffic emanating from the specified source, destined for the corresponding group. It need not forward the alert packet any further.

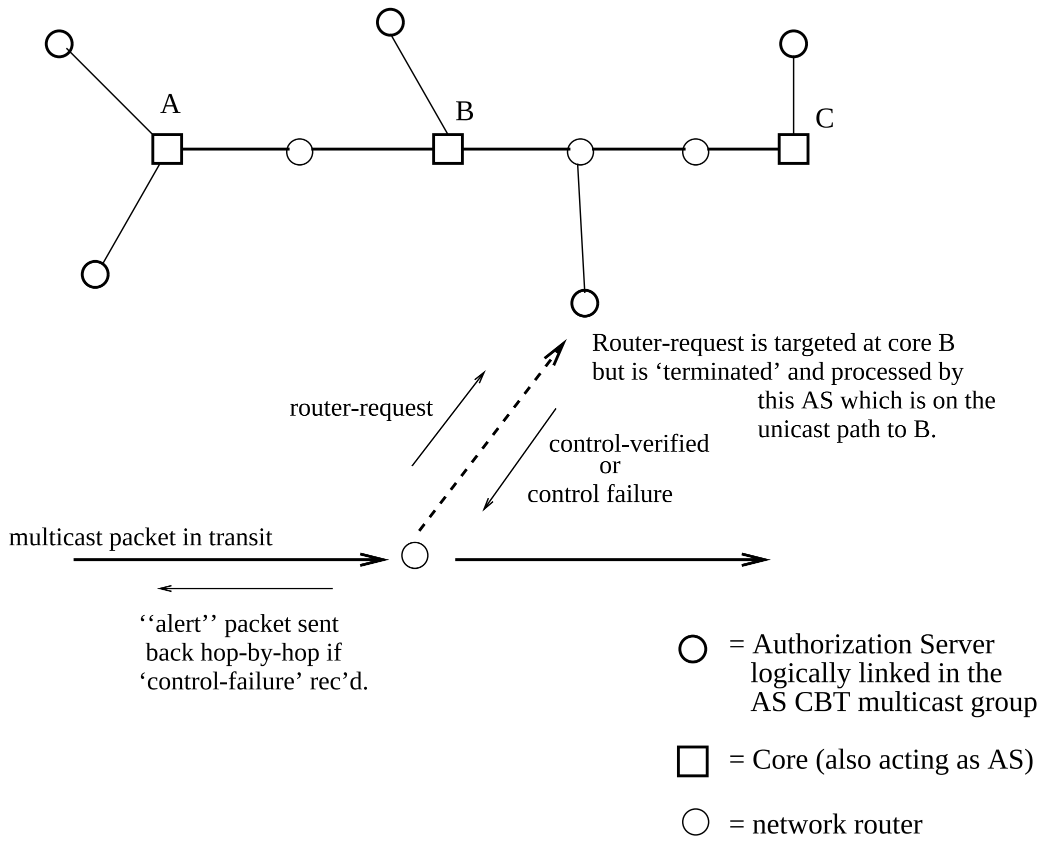

Figure 6.2: Multicast Transit Traffic Control – Model of Interactions

6.11.3 Multicast Transit Traffic Control – Message Contents

The *terminology* used here adheres to the conventions used in section 6.8.4.

Multicast transit traffic control comprises the following message exchanges:

(NOTE: R denotes a controlling router, and AS is an Authorization Server).

- $R \longrightarrow AS : [datapacket]$

  Although not shown, a *data packet* may include an authorization stamp, and a packet signature covering the entire data and invariable portions of the network layer header. The diagram below illustrates the security-relevant contents of the packet, present either in the packet's IP network layer *options* field (IPv6), or in a separate header immediately behind the IP header (IPv4).

authorization stamp	authorization stamp digital signature	packet digital signature

Figure 6.3: Security-relevant information contain within a packet

  The AS verifies its own digital signature included in the authorization stamp. It also verifies the freshness of the packet by inspecting the timestamp included in the authorization stamp. Finally, AS verifies the data packet's signature using the sender's public key.

- $AS \longrightarrow R : [CONTROL\_VERIFIED, t^{data}]^{SK\_AS}$

  (if all checks verified)

- $AS \longrightarrow R : [CONTROL\_FAILURE, t^{data}]^{SK\_AS}$

  (if any check fails)

6.12 Multicast Certificate Modification

Over the lifetime of a group, receivers can appear and disappear at random. Furthermore, misbehaving sources cannot be known to be so until they begin misbehaving. For this, and other reasons, it is important for a group's initiator to be able to create, delete, or modify an *existing* multicast certificate's inclusion or exclusion list, at any time.

Using a misbehaving sender as an example, we now describe how a group's initiator modifies the corresponding multicast certificate, maintained by the AS, so that packets from that sender no longer cause disruption over the wide-area.

- ◇ A multicast group's initiator, say for group G, unicasts reliably a *cert-modify* message to the AS, which is digitally signed by the initiator. The message includes the group identifier, the list which is to be modified (inclusion or exclusion), and how it is to be modified. In this case, the exclusion list for G is to be appended with the identity of sender S.
- ◇ The AS authenticates the sender (group initiator) for group G, with the aid of G's multicast certificate. It proceeds to append user S to the certificate's exclusion list.

From this point on, packets sent from source S to group G, are liable to detection, should a router submit a packet with an (S, G) combination to the AS for checking. That router will be notified by the AS that packets with (S, G) combination are unauthorized. As a result, the router sends an alert packet, which travels hop-by-hop to the router directly connected to the offending sender, thereby preventing further traffic from S destined for G from causing disruption over the wide-area.

6.13 Security Analysis

6.13.1 Minimizing *Assumed* Trust

We make no assumptions about hosts being trusted. With regards to our proposals for *group access control* and *multicast transit traffic control*, it is critical that all ASs belonging to the AS group behave correctly, i.e. adopt the correct procedures when requested to perform an authorization.

We assume routers in the network behave correctly *most* of the time, although, if this does not hold for some routers, provided all routers do not simultaneously misbehave, unauthorized traffic should not flow for very long before being detected, and subsequently prevented, from causing disruption over the wide-area.

6.13.2 Security Analysis of Group Access Control

Multicast group access control substantially reduces the overall risk of attack by restricting a group's accessibility. It does not, however, reduce the threat of attack on communications links. Group access control depends on a local subnet's membership-interrogating router correctly participating in an exchange with the AS as part of secure IGMP. If this assumption does not hold then unauthorized listening is possible, unless all of the corresponding group's data packets are encrypted. Unauthorized sending however, is liable to be detected (and subsequently prevented) by random checking by transit routers, as discussed in section 6.6.3.

Also, the scheme we have described cannot enforce group access control at the granularity of a user or host: if multicast is to be taken advantage of on broadcast media, it is not possible to prevent hosts on an authorized subnetwork from receiving multicast traffic already authorized to flow over that subnetwork without altering the internetwork service interface [26] of *hosts*.

6.13.3 Security Analysis of Multicast Transit Traffic Control

The threats we present in section 6.4 can only be effectively countered by multicast transit traffic control if the corresponding group is restricted.

Multicast transit traffic control serves to detect, and subsequently prevent, unauthorized or misbehaving sources from sending multicast *data* to a restricted group. This removes the threat of a successful denial of service attack resulting from multicast traffic.

All security checking is carried out by an Authorization Server. As the AS's are assumed to function correctly, we can be assured of the integrity of the result of a packet check.

The frequency with which routers are expected to check a multicast packet should be such that multicast packets from a particular source do not go unnoticed for more than a few seconds, amortized over the entire delivery tree. However, if some routers are misbehaving, then an alert packet may experience difficulty in reaching a misbehaving source. Misbehaving *routers* can be quickly identified by a neighbouring router through failure to receive an acknowledgement for an alert packet. That neighbouring router can then advise the network administrator accordingly. Furthermore, a subsequent packet check by a router upstream of a misbehaving router will eventually result in the quelling of the data source.

Other than the assumptions we place on routers as stated above, the security of multicast transit traffic control is dependent on the *mechanisms* we use to counter specific threats. We discuss specific threats below.

6.13.4 Security Analysis of Specific Threats

The multicast-specific threats we have discussed throughout this chapter can only be countered if group access restrictions are imposed. We are not proposing the use of our schemes for so-called *open groups* because of the "freedom" afforded to both senders and receivers.

In the context of restricted groups, masquerading and repudiation are not possible under the scheme we have described. This is because the authorization stamp carried by packets of restricted groups is unforgeable – it is created and digitally signed by the AS. It contains the identity of the requestor, as well as the requestor's source IP address, thereby *binding* a requestor to a particular host. Authorization stamps cannot be "stolen" whilst being returned to the requestor, since they are encrypted using the public key of the requestor. Furthermore, the combined presence of a stamp and the sender's digital signature across the network-layer header and packet data, means that both header and data are *bound* together, thereby preventing stamps from being "stolen" from data packets in transit.

The presence of an authorization stamp and sender's digital signature on each packet means that a receiver of a packet can be sure that it originated from the sender specified in the stamp, and is from the host as specified by the packet's IP source address.

The presence of a timestamp together with a lifetime in the authorization stamp, means that replay is only possible *within* the granularity, i.e. lifetime, of the timestamp. If replay is considered a serious threat, then the lifetime granularity can be increased, at the cost of an increased number of exchanges with the AS.

If replay is caused by a *stuttering* source rather than by malicious interception, such that it causes undue congestion, it can be prevented using transit traffic control, as described in section 6.10.

6.14 Chapter Summary

In this chapter we motivated the need for the integration of security controls and mechanisms in multicast communication. We have proposed two schemes that should be available to multicast groups that desire precautionary measures to be taken to avert the threats we have discussed. Our proposals eliminate many security threats to so-called *restricted* multicast groups.

Multicast group access control provides a mechanism for imposing group membership restrictions, and significantly reduces the overall security threat by constraining a group's *accessibility*.

Multicast transit traffic control provides an effective and flexible means for reliably ensuring the *authenticity, integrity*, and *freshness* of multicast data packets sent to restricted groups, without incurring any packet forwarding delay. It also provides a means for preventing misbehaving multicast sources from causing undue disruption over the wide-area.

Whilst security measures usually incur additional processing burden, we have introduced mechanisms, as described in section 6.6.3, which help reduce this cost significantly with minimal impact to the security assurances offered. Our schemes incur no additional burden to end-systems of unrestricted groups.

Both schemes rely on the existence of a security framework such as that already proposed by the Internet community. Our schemes are scalable (since they require extremely little or no state), and are multicast algorithm independent.

As no implementation yet exists for our schemes, we do not yet have a more detailed analysis of the *cost* of these schemes. Our schemes incur very little processing overhead on routers, since the AS performs all packet security checking. The major cost then, is to an AS.

There is also the consequential cost of the *delay* of a message exchange with an AS, which depends on the time it takes an AS to process a packet. The processing cost to the AS can be extrapolated from similar analyses done in [96] and [34]. The resulting cost to group access control is a somewhat longer *join latency* than there would be if no controls were employed. The cost to transit traffic control is a longer delay in a router detecting unauthorized traffic.

In neither case do we expect these delays to be of more than a few seconds at the very most, which is largely insignificant in the context of our schemes. The reason we consider this to be insignificant is, taking an audio- or video conference as an example, data flows typically for extended periods (*O(minutes)* between any one sender and the receivers, and therefore one or two seconds would appear as no more than a "blip" in transmission.

Chapter 7

CBT Security Architecture

7.1 Introduction

In introducing the previous chapter we explained how and why the Internet is becoming much more security-aware. In this chapter we describe the CBT security architecture which allows for the *secure* joining of a CBT group tree. The CBT multicast protocol has made explicit provision for security features, unlike existing IP multicast schemes [98, 70], and more recently proposed schemes [24].

This chapter also presents a solution to the *multicast key distribution* problem. These security-related services are *integral* in the CBT protocol. Their use is optional, and is dependent on each individual group's requirement for security. Furthermore, the use of the CBT multicast protocol for multicast key distribution does *not* preclude the use of other multicast protocols for the actual multicast communication itself.

Secure joining implies the provision for *authentication*, *integrity*, and optionally, *confidentiality*, of CBT join messages. In short, our scheme serves to *authenticate* tree nodes (routers) and receivers (end-systems) as part of the tree joining process.

Existing network layer multicast protocols, such as DVMRP [98] and M-OSPF [69], have no provisions for security features – indeed, any security implemented for these protocols would need to be implicit in the network layer protocol itself, i.e. IP [78]. This is because existing multicast protocols are integral to the IP protocol, unlike CBT which is its own network layer protocol[1] with its own header. Furthermore, the "connection-oriented" approach adopted by CBT complements security well for reasons we explain below (see section 7.3).

The security architecture we propose is independent of any particular cryptotechniques, although many security services, such as authentication, are easier if public-key cryptotechniques are employed. A more detailed discussion of why public-key cryptotechniques are more suited to multicast is given in chapter 6.

7.2 The Need for Network Layer Security

A pertinent question is: do we *need* to implement security functions in the network layer, or should security be left to multicast applications?

Security functions can be implemented at virtually any layer of a communications protocol stack, from the data link layer up to the application layer, and there are valid reasons, in terms of protection, efficiency, and application requirement, why implementing security functions in the different layers makes sense [86].

It is only relatively recently that serious consideration has been given to implementing security at the network layer [64]. A group of Internet researchers is already studying various network layer security proposals, for example, SP3 [29], I-NLSP [93], PIPS [30], and swIPe [44]. These considerations followed on from the growth and commercialization of the Internet [52] which resulted in the division of the Internet into administrative domains whose boundaries became subject to access policy restrictions [14]. As a result, the traditional end-to-end (transport layer and above) argument supporting the protection of end-system resources expanded downwards in the protocol stack to encompass the network layer, since network resources, e.g. routers, are end-points in a communications sense, and therefore require protection in their own right [33].

The importance of integrating security into routing protocols is recognised in [59]. It

[1] As we have already explained earlier, CBT is an *upper* network layer protocol, similar to ICMP, which actually runs over IP.

is possible for an intruder who has either compromised a router, or a communications link, to add, modify or delete information in routing traffic, with far-reaching implications for end-users. This applies equally to multicast routing as it does for unicast routing. Therefore, it is necessary to provide authenticity and integrity of routing updates, and this can be achieved by appending a digital signature (or message integrity check (MIC)[2]) to routing messages [59].

In our discussion on CBT security, we do not discuss security regarding *multicast route updates*[3], since this is sufficiently orthogonal to secure *multicast tree construction*, which is also a network layer function.

7.3 How the CBT Architecture Complements Security

Soft state source based multicast schemes, such as DVMRP and M-OSPF, do not lend themselves well to security implementations. The reason for this can be extracted from two independent properties of existing IP multicast schemes: a multicast delivery tree only exists as long as there is data flow, i.e. there is no explicit multicast tree set up; a router receiving a multicast packet does not necessarily know the previous-hop router that forwarded it, and therefore, the forwarding router (previous-hop) cannot be authenticated. All a receiving router can be sure of is the interface over which a multicast arrived.

The CBT architecture, on the other hand, does not suffer from this problem. CBT's "connection-oriented" approach means that all routers that make up a delivery tree know who their on-tree neighbours are, and these neighbours are authenticated as part of tree set-up. As part of secure tree set-up, neighbours could exchange a secret *packet handle* for inclusion in the CBT header of data packets exchanged between those neighbours, allowing for the simple and efficient hop-by-hop authentication of data packets (on-tree).

The presence of tree focal points (i.e. cores) gives a CBT tree a natural *authorization*

---

[2] A message integrity code (MIC) is similar to a digital signature, except a shared secret key is used to encrypt a message digest.

[3] Once multicast capability is installed throughout the internetwork infrastructure, *multicast routes* will be gleaned purely from unicast routes. Therefore, the integrity of multicast routes will depend on the integrity of unicast routes.

point in a security sense – the formation of a CBT tree requires the core to acknowledge at least one join in order for a tree branch to be formed. This authorization capability can be passed on to joining nodes that are verified.

In terms of security, the "connection-oriented" approach of CBT offers several additional advantages: once a multicast tree is established, tree state maintained in the routers that make up the tree does not time out or change necessarily to reflect underlying unicast topology. The implications of this with regards to security, are that nodes need not be subject to repeated authentication subsequent to a period of inactivity, nor must tree nodes re-authenticate themselves as a result of an underlying unicast topology change, unless of course, an on-tree node failure has occurred.

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

7.4 The Multicast Key Distribution Problem

We believe that multicast key distribution needs to be combined with *group access control.* Without group access control, there is no point in employing multicast key distribution, since, if there are no group restrictions, then it should not matter to whom multicast information is divulged. We discussed group access control in the previous chapter.

The essential problem of distributing a *session (or group) key* to a group of multicast receivers lies in the fact that some central key management entity, such as a *key distribution centre (KDC)*[4], must authenticate each of a group's receivers, as well as securely distribute the session key to each of them. This involves encrypting the relevant message n times, once with each secret key shared between the KDC and corresponding receiver, before

[4]A *Key Distribution Centre (KDC)* is a network entity, usually residing at a well-known address. It is a third party entity whose responsibility is to generate and distribute symmetric key(s) to peers, or group receivers in the case of multicast, wishing to engage in a "secure" communication. It must therefore be able to identify and reliably authenticate requestors of symmetric keys.

multicasting it to the group[5]. Potentially, n may be very large. Using public key encryption for distributing group keys is not an option, since the group key would be accessible to anyone holding the source's public key, and public keys are either well-known or readily available. In short, existing multicast key distribution methods do not scale.

There is very little published work to date on multicast key distribution. However, very recently work has been published describing various methods for securely distributing a *source-specific* symmetric key (e.g. a DES key) to members of a group without requiring a group *key-encrypting key (KEK)* [38]. Moreover, the source-specific key is distributed in encrypted form, and decipherable only by a group's receivers, by appending it (once for each receiver) to the source's enciphered message. This eliminates the need for explicit group re-keying by means of separate messages [39]. These methods have several advantages and disadvantages over traditional multicast key distribution methods. A significant advantage is that each source can use a distinct encryption key, and therefore receivers can authenticate an individual sender. Furthermore, re-keying can be done as part of sending a message – this gives the sender the ability to perform *receiver filtering*, a very desirable feature in groups where some receivers are to be excluded from certain, or all, transmissions. The most serious disadvantage is that a sender must know the group membership each time re-keying is done – the message decryption key is encrypted with the public-key of each receiver and appended to the initial message. In open dynamic groups, ideally, a sender should not be required to keep track of group membership.

Another proposal has emerged recently, called the *Group Key Management Protocol (GKMP)* [42]. This was designed for military networks, but the authors have demonstrated how the architecture could be applied to a network like the Internet, running receiver-oriented multicast applications.

GKMP goes a considerable way to addressing the problems of multicast key distribution: it does not rely on a centralised KDC, but rather places the burden of key management on a group member(s). This is the approach adopted by the CBT solution, but our solution takes this distributed approach considerably further, which makes our scheme that much more scalable. Furthermore, our scheme is relatively simple.

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

The disadvantages of GKMP include: the requirement of hosts to hold *permission certificates* – these are akin to *capability lists*. This implies certificate management, and the one-to-many distribution of certificates to hosts. In the CBT model, multicast certificates (see chapter 6), which are *access control lists*, are managed centrally by Authorization Servers, which provide similar functionality; for very large networks, GKMP requires additional GKMP management nodes that provide for key management and distribution. In CBT, this functionality is automatically "passed on" to a node when that node joins the tree; the GKMP management application must initially distribute a *key package* to each member individually, requiring each group member to engage in a secure protocol exchange with the group manager, which involves authentication. This requires the group manager to transmit securely a key package to each member. This is a serious drawback of GKMP, and, whilst it has moved key management functionality away from a centralised KDC, the burden of n secure transmissions from any network entity is unattractive and imposes certain performance requirements on such management entities.

The CBT model for multicast key distribution is unique in that it is *integrated* into the CBT multicast protocol itself. It offers a simple, low-cost, scalable solution to multicast key distribution. We describe the CBT multicast key distribution approach below. The model we propose assumes that network routers are not prone to malicious behaviour. Section 7.6 presents a slightly amended version for environments where this assumption does not hold.

7.5 The CBT Multicast Key Distribution Model

The security architecture we propose allows not only for the secure joining of a CBT tree, but also doubles in providing a solution to the *multicast key distribution* problem [41]. This is achievable as an implicit and integral part of the secure tree joining process. If a group session key is not required, its distribution may be omitted. Key distribution does *not* incur any extra message overhead, although its presence will increase message size slightly.

The use of CBT for scalable multicast key distribution does not preclude the use of other multicast protocols for the actual multicast communication. CBT could be used solely for multicast key distribution – any multicast protocol could be used for the actual multicast communication itself.

How a group key is used depends on the security requirements and the sensitivity of the traffic. Traffic between a group's receivers could be encrypted, or multicast data could include a digital signature (or MIC). The parameters for any cryptotechniques implemented are obtained in the secure CBT tree joining process.

In a large internetwork it is not practical to have a centralised KDC – such a model does not scale. In the context of multicast communication, the problem is compounded by the need for new members to explicitly request the group key from the KDC, which must authenticate each requestor with the aid of a group access control list (ACL). The group ACL is transmitted to the KDC by *one* group member (probably the group initiator). If a requestor is verified by the KDC, it securely transmits the group key to the new receiver.

The model that we propose does not rely on the presence of a centralised KDC – indeed, the KDC we propose need not be dedicated to key distribution. We are proposing that each group have its own *group key distribution centre (GKDC)*, and that the functions it provides should be able to be "passed on" to other nodes as they join the tree. Hence, our scheme involves truly distributed key distribution capability, and is therefore scalable. It does not require dedicated KDCs. We are proposing that a CBT core initially take on the rôle of a GKDC.

7.5.1 Operational Overview

Multicast sessions are typically announced by means of a session directory such as "sd". For CBT multicast sessions, the session announcements will contain the identities of the cores for a group, together with other session parameters such as multicast address, port number, etc. For secure multicast sessions, a session directory could also announce the public keys for each core router, or where such keys can be located. Hence, receivers know the identities of core routers, and are able to retrieve their public keys.

When a CBT group is initiated, it is the group initiator's responsibility to create a multicast access control list (as described in the previous chapter). The group initiator may also be called the *group leader*. It subsequently *securely* transmits the list to the *primary core* for the group by including the list in the CBT-CORE-NOTIFICATION[6] message – we are advocating that the cores of a CBT tree take on the initial rôle of GKDCs. The group initiator need not create a group session key, rather, the primary core for the group creates it subsequent to receiving, and verifying, the message containing the access control list.

The primary core also creates a *key encrypting key (KEK)* which is used for re-keying the group just prior to an old key exceeding its lifetime. This re-keying strategy means that an active key is less likely to become compromised during its lifetime.

The access control list, group key, and KEK are distributed to each of the other cores (as well as any other intermediate CBT routers) as part of the initial tree set-up (i.e. backbone set-up) phase, which occurs "securely". By this we mean that receiving CBT routers authenticate the origin of the security information, i.e. the primary core, as well as the router that passed on this information, which may or may not be the primary core. In other words, the access control list, group key, and KEK are disseminated to other tree nodes as they join the tree. How this happens should become clear in the example we give below.

Any tree node with this information can verify a joining member, and hence, secure tree joining and multicast session key distribution are truly distributed across already verified tree nodes. Since the session security parameters are originated and digitally signed by the primary core, which is well-known to all group members, each router receiving the session security parameters, as well as each member receiving a group key and KEK, can authenticate the issuer of those keys, i.e. the primary core.

7.5.2 Example – Integrated Join Verification and Multicast Key Distribution

For simplicity, in our example we assume the presence of an internetwork-wide public key management scheme, such as that proposed in [31]. However, we are not precluding the use of symmetric cryptographic techniques – all of the security services we are proposing,

[6]This message is sent as part of a group's initiation (see section 5.1.2).

i.e. integrity, authentication, and confidentiality, can all be achieved using symmetric cryptography, albeit at a greater expense: e.g. negotiation with a third party to establish pairwise secret keys, and the secure distribution of secret keys.

For these reasons, we assume that a public key management scheme is globally available, for example, through the Domain Name System (DNS) [31] or World Wide Web (WWW) [9].

Much of the **terminology** we use here, we used in the previous chapter. However, we formally define some additional terms here:

- **grpKey**: group key used for encrypting group data traffic.
- **ACL**: group access control list (for contents, see previous chapter).
- **KEK**: key encrypting key, used for re-keying a group with a new group key.
- **group access package (grpAP)**: sent from an already verified tree node to an adjacent joining node. The group access package is as follows:

$$[token^{sender}, [ACL]^{SK\_core}, \{[grpKey, KEK]^{SK\_core}\}^{PK\_origin-host},$$
$$\{[grpKey, KEK]^{SK\_core}\}^{PK\_next-hop}]^{SK\_sender}$$

As we have already stated, the elected primary core of a CBT tree takes on the initial rôle of GKDC. In our example, we assume that a group access control list has already been transmitted to the primary core, that the primary core – as a result of receiving the group's ACL, has created a group key and a key-encrypting key, and finally, that any secondary cores have joined the primary core, thereby forming the backbone, or *core tree*. Therefore, all routers along the tree's backbone have effectively become GKDCs for the group.

Rather than demonstrate how the routers along the core tree become GKDCs, due to the similarity, it should suffice to describe how a host securely joins a CBT tree (i.e. group), and how CBT routers along a particular tree branch are given GKDC capability. This is what we describe as the *integrated* CBT secure joining and multicast key distribution process.

In the diagram below, host h is joining the multicast group G. By some means, h knows that the topologically nearest core is C, and therefore requests its local designated router (DR), A, which has not yet joined the tree, to target a JOIN-REQUEST at C.

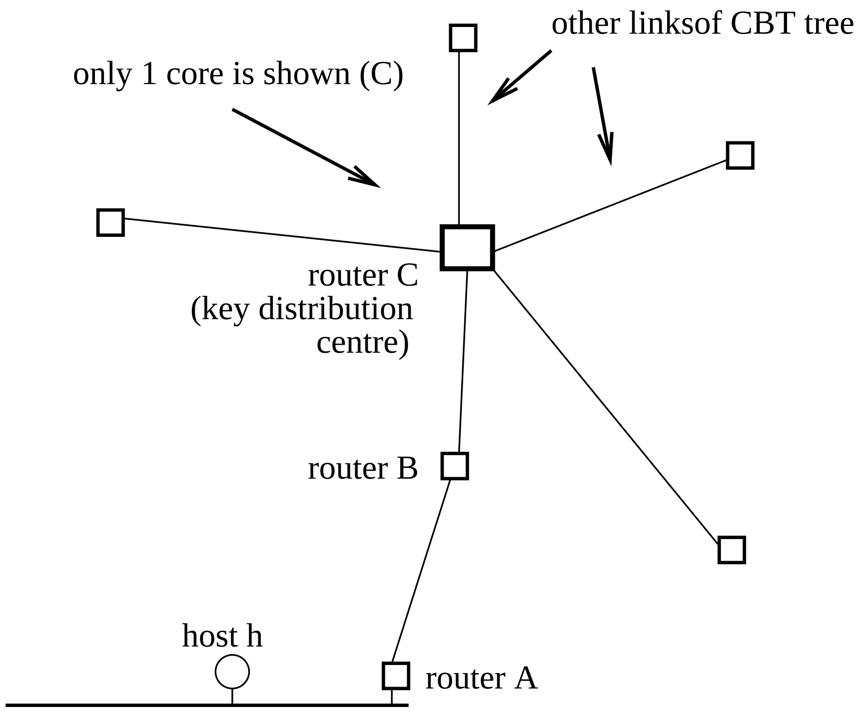

Figure 7.1: CBT Authenticated Join process (hosts and routers) and Key Distribution

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

- As part of host h's joining process, it sends a special CBT message (CBT-TAG-REPORT) to the local DR (router A), containing the address of a core router to which it wishes a tree branch to be created from the DR. This message invokes A to send a JOIN-REQUEST. Contained within the message is h's digitally signed *token*[7].

[7] A *host*'s token differs in two respects compared with the standard definition of a token [46]. To refresh, a token assists a recipient in the verification process, and *typically* contains: recipient's unique identity, a timestamp, and a pseudo-random number. A token is also *usually* digitally signed by its originator.

Firstly, a host's token does *not* contain the intended recipient's identity, since this token may need to traverse several CBT routers before reaching a GKDC. A host does not actually know which router, i.e. GKDC, will actually acknowledge the join that it invoked.

$$h \longrightarrow DR(A) : [[token^h]^{SK\_h}, CBT - TAG - REPORT]^{SK\_h}$$

- Local designated router (DR) A verifies the complete message, and the token received from h. On successful verification, A generates a JOIN-REQUEST which includes its own *unsigned* token, and the token of h, which is signed by h. This join is digitally signed by A and sent to the best next-hop on the path to the core (C). The best next-hop in our example is router B.

$$A \longrightarrow B : [token^A, [token^h]^{SK\_h}, JOIN - REQUEST]^{SK\_A}$$

- B first verifies A's JOIN-REQUEST. On successful verification, B repeats the above step, except the join is sent from B to C, the core, which happens to be the next-hop. Also, the token included in the join is that of B. h is not verified by B. h's token is copied to this new join.

$$B \longrightarrow C : [token^B, [token^h]^{SK\_h}, JOIN - REQUEST]^{SK\_B}$$

- C authenticates B's join. As the tree's *primary authorization point* (and GKDC), C also authenticates the host, h, that triggered the join process, provided h is included in the GKDC's access control list [6] for the group. If h is not in the corresponding access control list, authentication is redundant, and a JOIN-NACK is returned from C to B, which eventually reaches h's local DR, A.

  Once B and h have been verified, C forms a **group access package**, which includes C's token, and encapsulates it in a JOIN-ACK. Two key pairs (group key, KEK) are included in the group access package, one for the originating host, and one for the next-hop CBT router to which the JOIN-ACK is destined. Each key pair is digitally signed by the issuer, i.e. primary core for the group. The *host* key pair is encrypted using the public key of the originating host, so as to be only decipherable by the originating host, and the other key pair is encrypted using the public key of the next-hop router to which the ACK is destined – in this case, B. The access control list contained within the **group access package** is digitally signed by the

Secondly, the host's token *is* digitally signed – this is usual for a token. However, tokens generated by routers need not be explicitly digitally signed because the JOIN-REQUESTs and JOIN-ACKs that carry them are themselves digitally signed.

primary core. Finally, host h's signed token is included in the message to B. The token is used by the router connected to the subnet where h resides so as to be able to identify the key-requesting host, i.e. new member.

$$C \longrightarrow B : [[token^h]^{SK\_h}, grpAP, JOIN - ACK]^{SK\_C}$$

- On receipt of the JOIN-ACK from C, B first verifies C's digital signature covering the packet. B then extracts its encrypted key pair[8] from the group access package, decrypts it to verify the primary core's digital signature, then re-encrypts the key pair for storage. It also verifies the digital signature covering the access control list. It subsequently stores the verified ACL in an appropriate table. The key pair for the originating host remains enciphered.

  A copy of the router's key pair, which is digitally signed by the primary core, is taken and deciphered using its *secret key*, and immediately enciphered with the public key of next-hop to which a JOIN-ACK must be passed, i.e. router A. A **group access package** is formulated by B for A. It contains B's *token*, the group ACL (which is digitally signed by the primary core), a (group key, KEK) pair encrypted using the public key of A, and the originating host's key pair, already encrypted. The group access package is encapsulated in a JOIN-ACK and forwarded to A.

$$B \longrightarrow A : [[token^h]^{SK\_h}, grpAP, JOIN - ACK]^{SK\_B}$$

- A verifies the digital signature covering the JOIN-ACK received from B. A copy of the encrypted key pair that is for itself is extracted from the group access package and deciphered in order to verify the primary core's digital signature. On successful verification, the enciphered key pair is stored by A. The digital signature of the included access control list is also verified, and stored in an appropriate table. The key pair encrypted for h is extracted from the group access package, and is forwarded directly to host h, which is identified from the presence of h's signed token. On receipt, host h decrypts the key pair for subsequent use.

[8]Routers do not hold key pairs for any other purpose than having the ability to pass them on, as part of a *group access package*, to other routers. The key pair is stored in encrypted form so as to be less vulnerable to a router attack. As a result, the group key and KEK only ever appear in plain text for very short periods in the decryption-encryption phase when being passed on to another tree-joining router.

$$A \longrightarrow h : [[token^h]^{SK\_h}, \{grpKey, KEK\}^{PK\_h}]$$

Going back to the initial step of the tree-joining procedure, if the DR for the group being joined by host h were already established as part of the corresponding tree, it would already be a GKDC. It would therefore be able to directly pass the group key and KEK to host h after receiving a CBT-TAG-REPORT from h:

$$A \longrightarrow h : [[token^h]^{SK\_h}, \{grpKey, KEK\}^{PK\_h}]$$

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

7.6 A Question of Trust

The security architecture we have described, involving multicast key distribution, assumes that all routers on a delivery tree are trusted and do not misbehave. A pertinent question is: is it reasonable to assume that network routers do not misbehave and are adequately protected from malicious attacks?

Many would argue that this is not a reasonable assumption, and therefore the level of security should be increased to discount the threat of misbehaving routers. As we described above, routers periodically decrypt key pairs in order to verify them, and/or re-encrypt them to pass them on to joining neighbour routers.

In view of the above, we suggest that if more stringent security is required, the model we presented earlier should be slightly amended to accommodate this requirement. However, depending on the security requirement and perceived threat, the model we presented may be acceptable.

We recommend the following change to the model already presented above, to provide a higher level of security:

All join-requests must be validated and verified by a **core router**, *i.e. a join arriving at an on-tree router must be forwarded upstream to the core if it is identified as being a* *"secure" join (as indicated by the presence of a signed host token).*

The implication of this is that key distribution capability remains with the core routers and is *not* distributed to non-core routers whose joins have been verified. Whilst this makes our model somewhat less distributed than it was before, the concept of key distribution being delegated to the responsibility of individual groups remains. Our scheme therefore retains its attractiveness over centralized schemes.

7.7 Chapter Summary

This chapter has described the *secure* CBT tree joining process, which serves to authenticate tree nodes (routers) and receivers (end-systems) as an integral part of the tree joining process.

Compared with all other IP multicast protocols, CBT is unique in that it has made explicit provision for security. This has been possible because CBT is its own network layer protocol, whereas other IP multicast schemes are an integral part of the IP protocol. Thus, providing security in these other protocols requires the implementation of security into IP itself.

With no extra message overhead, the CBT "secure" joining process offers a scalable solution to the multicast key distribution problem, by distributing key distribution functionality between already verified on-tree nodes, thereby eliminating the need for a centralized key distribution centre (KDC).

Whilst our solution is based on the CBT architecture and protocol, this does not preclude the use of other multicast protocols for secure multicast communication subsequent to key distribution. Furthermore, virtually all of the functionality present in our solution is in-built in the *secure* version of the CBT protocol, and therefore multicast key distribution is (optionally) an integral part of the CBT protocol.

Chapter 8

ATM and Multicast

8.1 Introduction

Broadband-ISDN[1] is the term used to describe the next generation of public wide-area networks (WANs). B-ISDN is a high-speed networking concept, offering communication services at data rates that range from a few megabits per second (Mbps) to several gigabits per second (Gbps). It has been designed to carry virtually any kind of data, for example, voice, video, text, graphics etc., and does so in fixed-sized packets called *cells*.

The underlying transmission technology for B-ISDN is known as *Asynchronous Transfer Mode (ATM)*, whose standards are defined by CCITT[2]. ATM is a link-layer, connection-oriented transmission technology which is responsible for transferring 53 octet *cells* (5 octet header + 48 octets payload) between network subscribers.

Various major assumptions lie behind the design of ATM, for example, that ATM networks will have a hierarchical structure, similar to today's 'phone networks, and that the networks over which ATM runs are fibre optic, which contributes significantly to ATM's expected low error and loss rates [72]. It is for these reasons that ATM is of tremendous interest to telephone companies – ATM not only satisfies many of the requirements of

[1]Integrated Services Digital Network

[2]Some standards have also been defined by a user and vendor group known as the *ATM Forum*.

a telephone network, but also removes the need for the two separate networks 'phone companies currently utilize, one for voice, and one for signaling. Furthermore, ATM provides the means for the telephone network to provide a vast new range of services, e.g. video-on-demand, that were not possible with the old technology, and provides the functionality that previously required two separate networks.

For all its advantages, there are several aspects of ATM which are hindering its widespread deployment, not least its lack of specification for *dynamic routing*, and *local-* and *wide-area multicasting*. In this chapter we explain why *traditional* IP Internet multicast schemes cannot be adapted for ATM multicasting. In contrast, we describe how and why the CBT multicast architecture *can* be applied to ATM networks – indeed, the CBT multicast approach maps rather well onto the ATM paradigm.

8.2 Traditional IP Multicast and ATM

In order to understand why traditional IP multicast (i.e. DMVRP, M-OSPF) is not suited for an ATM environment, we need to look at the service provided by the data-link layer of IP and ATM networks[3].

In IP, the data-link layer provides the network layer with nothing more than a best-effort, connectionless datagram delivery service. In contrast, the data-link layer in ATM networks offers a *uni-directional*, connection-oriented, low error rate service to the network layer. Connections in ATM are either of a *permanent*, or *semi-permanent* nature. If a *virtual channel (VC)*[4] between some two end-points does not exist, then a VC has to be explicitly set up before data can be transferred between those two points. In the context of ATM *multicast*, there are 2 different approaches that can be taken for establishing point-to-multipoint connections:

---

[3]The data-link layer of *subscribers'* equipment, i.e. end-systems, is called the *User-Network Interface (UNI)* in ATM. The data-link layer in ATM *switches* is called the *Network-Network Interface (NNI)*. In IP, the former is analagous with a host-subnetwork interface, and the latter with a router-to-router interface or point-to-point link.

[4]ATM's terminology for a virtual circuit connection.

- a uni-directional VC can be set up between a sender and each of a group's receivers, potentially resulting in a *mesh* of ATM connections between group members. A sender then becomes the root of an ATM multicast tree.
- a *server* can be elected, to which a single, uni-directional VC is established by each of a group's receivers. The server has pre-established connections to each of a groups receivers, and therefore acts as an intermediary by distributing any received multicast traffic to the group.

There is a proposal for running IP multicast over a subnetwork whose underlying technology is ATM (IP over ATM) [1], and this proposal suggests using *both* of the above methods, but each method for a different purpose. The subnetwork's ARP server acts as the *multicast server*, and is responsible for sending IGMP[5] queries (sent to the *all-systems* multicast address) onto the subnetwork. ARP requests for the *all-systems* multicast address are handled specially by the ARP server, with the ARP server returning its own ATM address for such requests, resulting in the requestor creating a VC to the ARP server (multicast server). Since all end-systems are required to join the *all-systems* group, all end-systems will have a VC to the multicast server. Similarly, the multicast server creates a VC to each host.

When a host wishes to join some random multicast group (to receive, and possibly send to the group), the new member, instead of multicasting a group-specific IGMP membership report on the subnetwork as is the case with Ethernet subnetworks, registers its join with the multicast server, which in turn, re-transmits the IGMP report to all other members of the *all-systems* multicast group, i.e. all hosts.

Only an active group sender need act on receipt of an IGMP report – it establishes whether it already has a VC open to the new receiver, and if it does not, it creates one, thereby adding the new receiver to the sender's source-based distribution tree.

In order for a new *sender* to be able to send, it must request the (IP, ATM) address pairs of all group members from the multicast server which stores them as part of its local

[5]IGMP referred to here requires adaptation for ATM, and implements several new message types specifically for ATM.

group membership information. Before sending, the sender must ensure that a VC is open to all group members.

It should be clear from our description above, that the source-based multicast tree approach, when the underlying technology is ATM, involves considerable overhead, especially for group *senders*. This approach looks particularly unattractive compared with the connectionless IP approach. What now follows is a more detailed explanation of the overhead and complexities involved in setting up source-based ATM multicast trees.

8.2.1 Why Source-Based ATM Multicast Trees are Bad

In ATM, the routing function and delivery tree building, whether multicast or unicast, is a function of the ATM layer, i.e. the data-link layer. In IP these are functions of the network layer. A *multicast routing protocol* is responsible for these very functions, and therefore, if ATM is the underlying technology, the multicast routing protocol must operate at the ATM layer itself. We now itemize the main reasons why, if the multicast routing protocol at the ATM layer builds source based trees, this is not a good idea, and why certain IP multicast algorithms would be unacceptable if operated at the ATM layer:

- the proposal for subnetwork multicasting, outlined above, is very inefficient in its use of resources. Not only must a group *sender* set up individual VCs to each group member, but, in order to do so, a group sender needs knowledge of who the receivers are. This contradicts IP multicast's *host group model* [26] paradigm, whereby senders are not required to know who/where receivers are. Rather, it is left to the multicast routing protocol to route group traffic efficiently to the group's receivers.

  In our thesis abstract we emphasized the primary benefits of multicast as being threefold: *optimization of transmitter costs*, *bandwidth/resource optimization*, and *the exploitation of parallelism in the network*. All of these benefits are compromised by the described ATM approach – if we make an analogy of ATM's operation on IP subnetworks, then IP multicasts would be *unicast* by a sender n times, once to each receiver – this is effectively what the described IP over ATM approach is doing.

  Returning to the point of a sender having to know who the group receivers are, this requires the presence of a *multicast server*, which essentially performs a group management function unnecessary in IP multicast.

- group set-up in source-based ATM multicast requires establishing a *mesh* of connections, which involves a complexity of $O(n^2)$. *Sending* to such a group involves a complexity of *O(n)*.

  The same complexities in IP multicast are *O(n)* and *O(1)*, respectively.

- *flood and prune* IP multicast protocols, such as DVMRP, do not have a place in ATM multicasting, since, in order to flood, a sender-initiated VC would need to be created to every destination within the domain of operation, and data subsequently sent over each VC, irrespective of whether the remote end of a VC were a group member. This is an unacceptable use of resources.

We now go on to describe another recently proposed scheme for creating *core based* ATM multicast trees, based closely on the concepts of the CBT multicast protocol for datagram networks, which is the centrepiece of this thesis. It is far more efficient than the scheme described above in its use of network resources, and the protocol design follows closely that of our own design, not surprisingly, since it was based on our architecture and protocol.

8.3 CBT for ATM Multicast

An ATM multicast proposal based on the CBT multicast protocol (see chapter 5) appeared recently [1].

The reasons for CBT being chosen as the basis of the ATM proposal include [63]:

- the CBT "connection-oriented" approach complements the ATM paradigm. For example, an already-established multicast tree branch does not change as underlying routes change, unless a node failure affecting the same branch causes partial tree reconfiguration.

- CBT shared delivery trees span only a group's receivers, and therefore, are not wasteful of network resources. This is particularly important in connection-oriented networks.

- CBT is unicast routing protocol independent.

- the CBT design allows for the potential emulation of source-based delivery trees.

We now give a brief background, followed by a description of how CBT has been adapted for use in a (wide-area) ATM network. The modified protocol is known as the *ACBT protocol*, to distinguish it from our CBT protocol, described in chapter 5.

ATM PNNI (private network-network interface) routing operates a multi-level hierarchical link-state routing protocol [63]. Nodes are divided into *peer groups*[6], which are groups of nodes that exchange link-state information with other nodes in the same group, so that all peer group members compose an identical topological database covering the peer group. Certain nodes within a peer group are permitted to exchange routing information with nodes in a *parent* peer group(s), i.e. a peer group in a next-higher hierarchy level. To make an analogy, this is somewhat similar to IS-IS intra- and inter-area routing [92].

A description of how CBT has been adapted for ATM multicast follows: each peer group has an elected *primary core*, whose identity is distributed to all other members of the peer group. This is done by means of a special "announce" message, sent periodically. The *backbone* (analogous with the CBT *core tree*) of a delivery tree consists of primary cores of different peer groups (but of the same multicast group) who share a common parent peer group. Hence, there may be several distinct "backbones" across a multicast group. So-called *exposed nodes* communicate primary core information up and down the peer group hierarchy using the special "announce" messages.

CBT-style *join-request* and *join-acknowledgement* (ACBT-JOIN, and ACBT-JOIN-ACK) messages are used to establish nodes on a particular distribution tree, but the resulting PNNI VCs, which are bi-directional, are *not* used for forwarding data – they effectively identify the nodes that are part of a CBT-style tree.

When there are senders, the sender must first request connection-establishment by sending a SETUP/ADD-PARTY message to the network. This message is targeted at a core the sender knows about, and follows the paths created by the tree-joining process, i.e. the flow set-up message is distributed by those nodes that have sent joins and received

[6]Note that these have nothing to do with multicast groups.

corresponding join-acks. When the SETUP message has been acknowledged, data can flow from the sender. On reaching an *on-tree* node, the data is disseminated to all outgoing links corresponding to that particular group tree. Tree links are thus shared between the different senders to the group.

Explicit *quit* and *re-join* (ACBT-QUIT, and ACBT-REJOIN) messages are used by members wanting to leave a group tree, or re-join a tree subsequent to failure, respectively. The semantics of these messages, as well as other maintenance messages, are very similar to those of the CBT protocol, described in chapter 5.

8.3.1 ACBT Examples

The diagrams below serve to demonstrate how CBT trees are used in ATM. With each diagram we provide a brief explanation of the specific process in the ACBT protocol being shown.

The first illustration (over) shows our example ATM network topology:

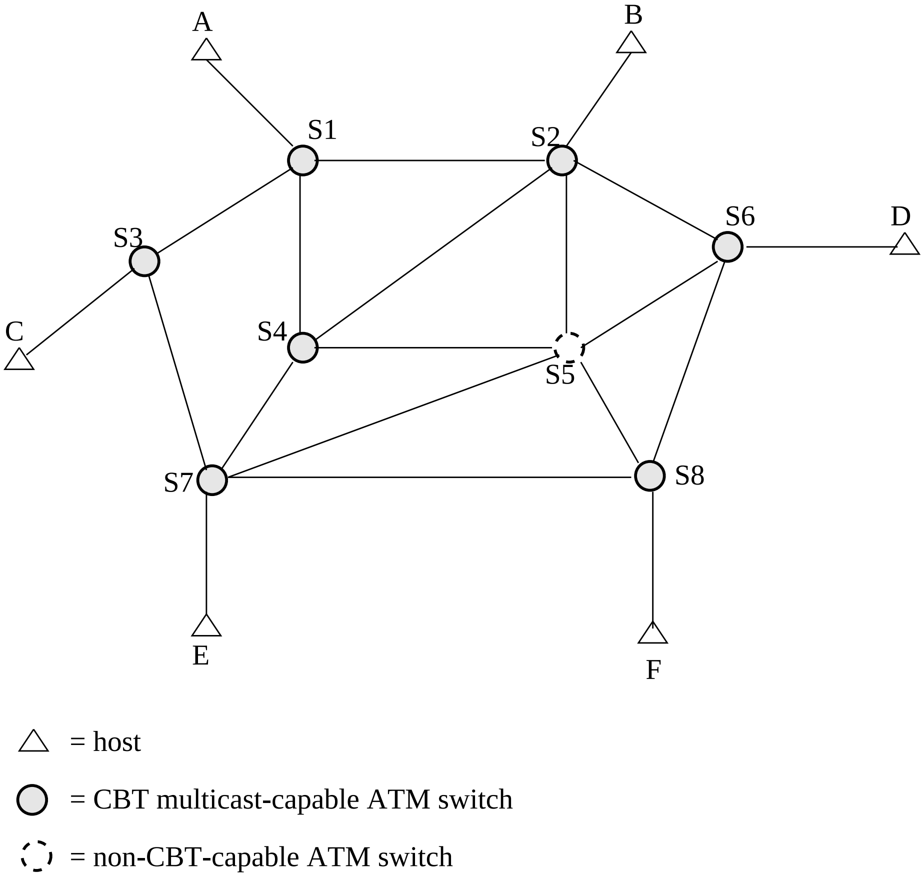

Figure 8.1: Example ATM Network Configuration

The following diagram shows how a host, in this case *C*, *joins* a core based multicast delivery tree. The "host-join" is a message sent by host *C*, requesting membership of the group specified in the message. Let's call this group *G*. The "host-join" is acknowledged by switch *S3* (but the acknowledgement is not shown in the diagram). After acknowledging the "host-join", switch *S3* formulates an ACBT-JOIN, which is unicast to switch *S1* – the primary core for group *G*. Each ACBT-capable switch on the unicast path processes the ACBT-JOIN, and caches relevant incoming and outgoing interface state. Upon reaching *S1*, *S1* responds to the join with an ACBT-JOIN-ACK, which follows the reverse-path of the corresponding join. The ack terminates where the join originated (*S3*).

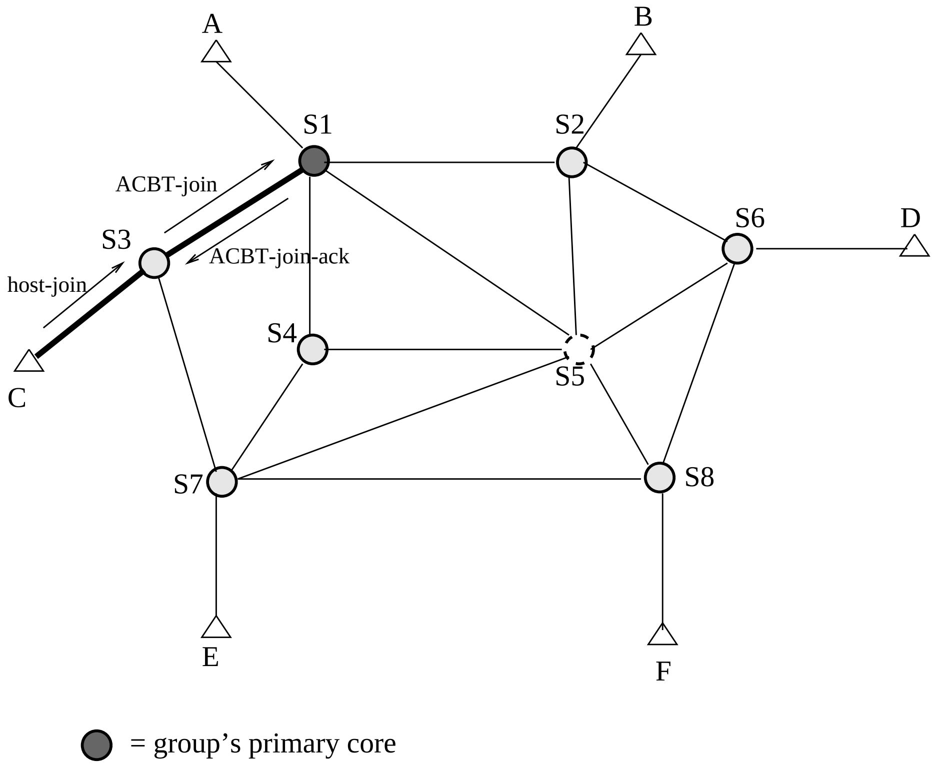

Figure 8.2: Host Joining CBT Group in ATM Network

The next diagram illustrates the procedures involved in a host (not necessarily a group member), in this case host E, sending to group G.

In order to *send* to group G, E must initiate a *connection*, which involves sending a SETUP message towards the primary core, *S1*. Unlike ACBT-JOINs, SETUP messages are acknowledged on a hop-by-hop basis *before* forwarding, i.e. each hop-by-hop connection is established before a switch receiving a SETUP message forwards it on towards the primary core.

On reaching a switch, a SETUP message is forwarded along each outgoing tree interface, should that switch have already-established tree interfaces. If not, the SETUP message is forwarded to the next-hop on the path to the primary core. In the diagram, the primary core (*S1*), forwards the SETUP message over an outgoing tree interface (the one leading to switch *S3*).

In this way, connections are established between a sender and all group members.

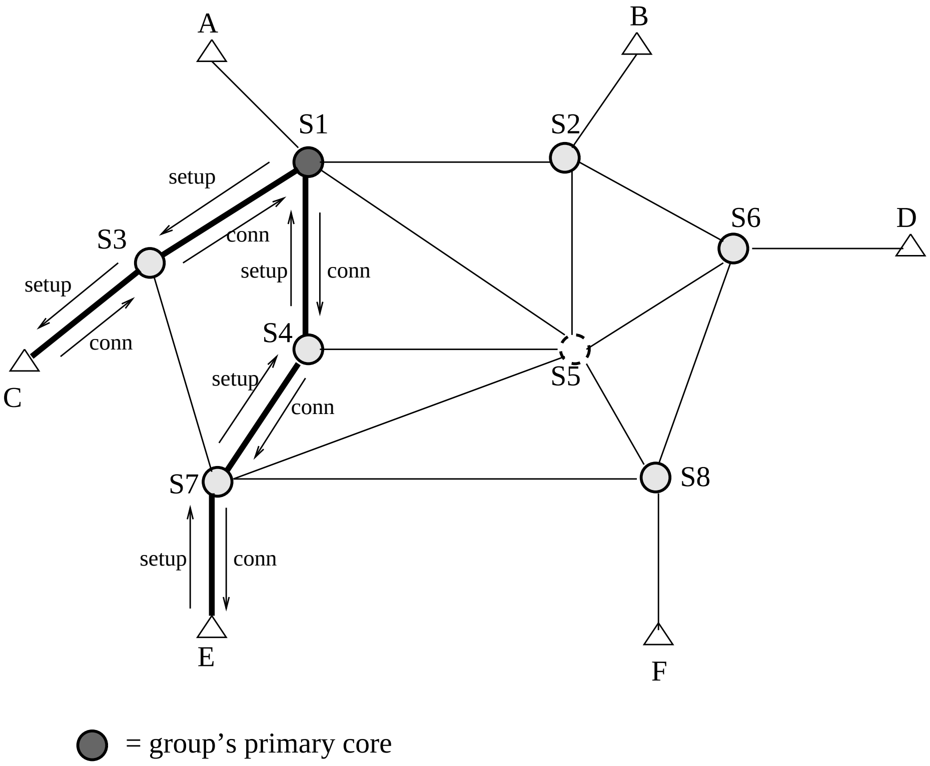

Figure 8.3: Host Sending to CBT Group in ATM Network

What if a group member appears whilst a sender is sending? The answer is provided in the next two diagrams below, since the process involves two stages.

In this case, host *F* wishes to join group *G*, which currently has an active sender, *E*. The 2-stage process involves *F* first joining the tree for *G*, followed by connection establishment which is initiated by the join-terminating switch (since the new receiver cannot know a sender is currently active).

The figure below shows how *F*'s join propagates towards the primary core (*S1*), but is terminated by switch *S4*, since *S4* is already part of the ACBT distribution tree. *S4* propagates an ACBT-JOIN-ACK along the reverse-path of the corresponding ACBT-JOIN. What happens next is described preceding figure 8.5.

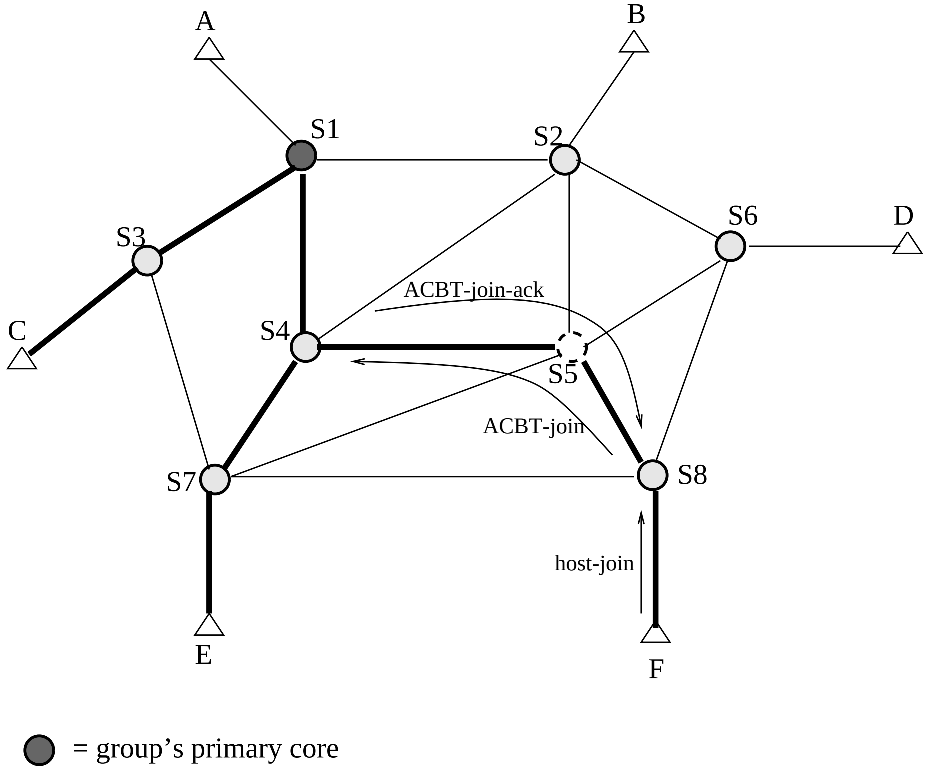

Figure 8.4: Host Joining Whilst Sender has Group Connection – Part 1

Subsequent to switch *S4* sending an ACBT-JOIN-ACK to switch *S8* (note that switch *S5* is non-ACBT-capable, and so ACBT messages must be "tunnelled" through *S5*), because *S4* is aware of an active connection for group G, it sends a SETUP message along the same path as the ACBT-JOIN-ACK just sent. On reaching *S8* (the next ACBT-capable hop), *S8* acknowledges the SETUP messages and the segment connection is established. This is done segment-for-segment all the way back to the receiver F. This is illustrated in figure 8.5.

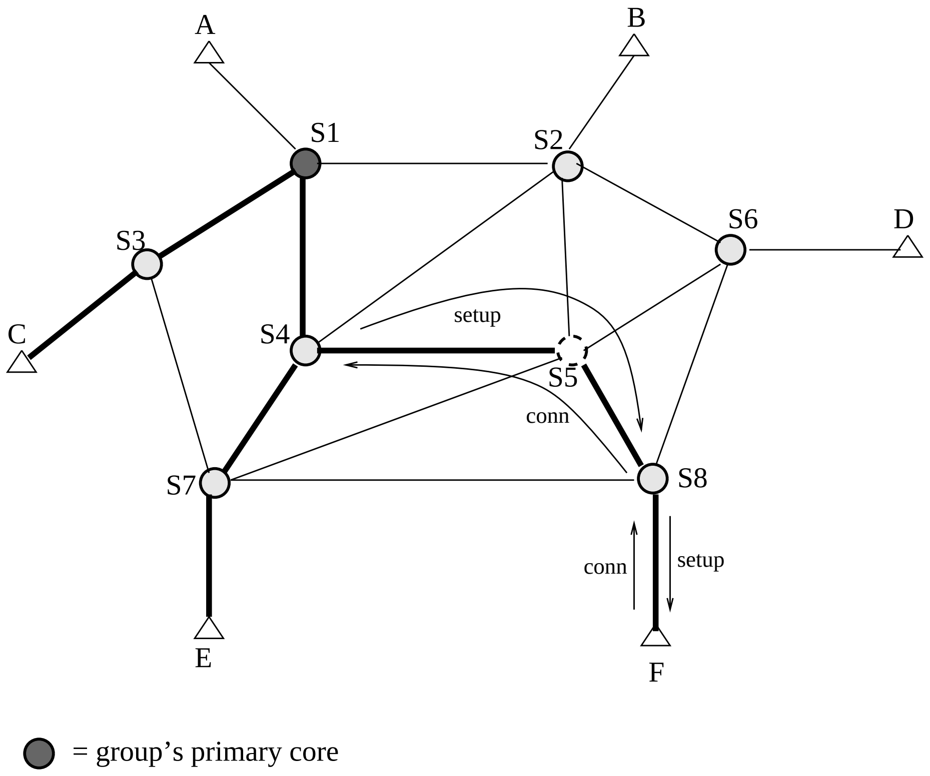

Figure 8.5: Host Joining Whilst Sender has Group Connection – Part 2

Finally, the last diagram shows how no-longer-used parts of the delivery tree, i.e. those no longer leading to any group members, are removed from the group tree.

Once a switch registers no further group presence on any of its interfaces, it sends a message to its parent switch, potentially resulting in the parent removing the child from the shared tree.

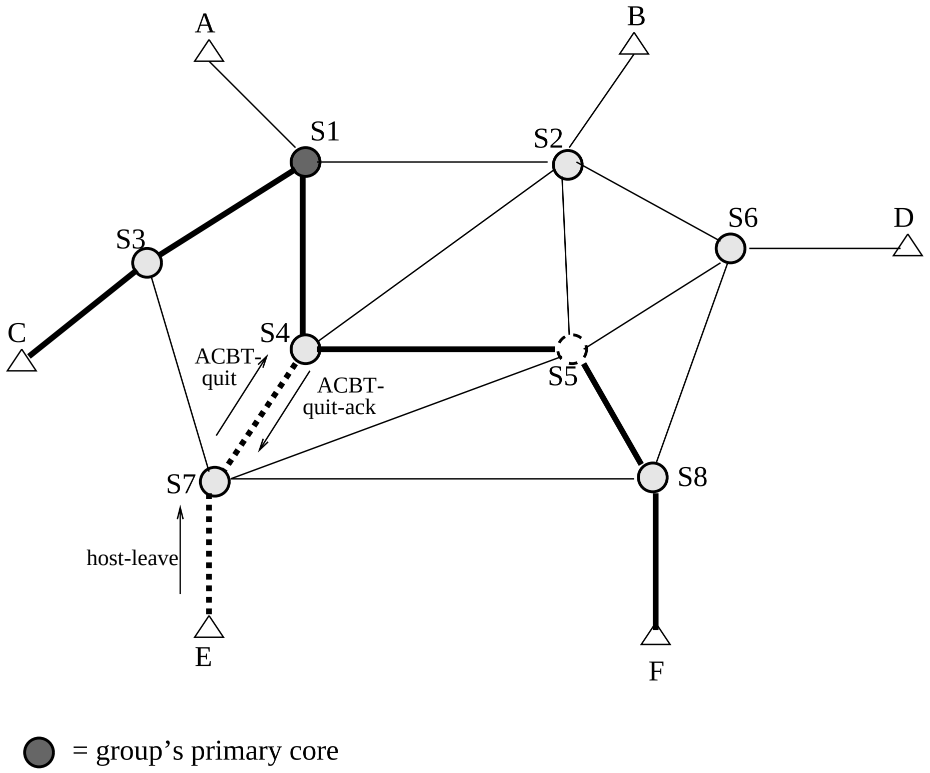

Figure 8.6: Pruning No-Longer-Used Branches

In this case, switch *S7* receives a "leave" message from a host on an attached subnetwork, which results in that subnetwork having no further members of group *G* present on it. Since *S7* is responsible for registering group presence, it sends an ACBT-QUIT for group *G* to its parent, *S4*.

On receipt of the ACBT-QUIT, *S4* carries out an interface check to establish if tree branch(es), or attached subnetwork group presence, still exist for *G*. In this case, *S4* does have a remaining child (*S8* through *S5*), and so the ACBT-QUIT is not forwarded further. ACBT-QUIT-ACKs are sent on a segment-to-segment basis in response to receiving an ACBT-QUIT.

A detailed description of the ACBT protocol, together with further examples, can be found in [63].

8.4 Chapter Summary

In this chapter we discussed how and why traditional IP multicast algorithms are not suited to ATM networks. The primary reason for this incompatibility is that the design philosophy of traditional multicast algorithms was founded upon the assumption that the underlying network offers a connectionless datagram delivery service. Furthermore, "flood and prune" techniques are wasteful of resources in a connection-oriented environment, and exhibit complexities for group set up and group sending that are orders of magnitude greater than they would be in connectionless environments.

We went on to describe an ATM multicast proposal that has recently been published. It is based heavily on our CBT multicast protocol, described in chapter 5. CBT is considered complementary to ATM-type networks for several reasons, for example, because its design philosophy emulates a connection-oriented approach, and also, CBTs span only a group's receivers, thereby making more efficient use of network resources.

Finally, we provided some examples to demonstrate how CBT is used for ATM multicasting. The modified CBT protocol has been termed the *ACBT protocol* in ATM.

Chapter 9

Conclusions

The focal point of this thesis 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 thesis 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 case where there are many senders to a multicast group.

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. However, CBTs offer no guarantees that this will be the case.

The definition of the CBT protocol includes provisions for *security*. Currently, CBT is the only multicast protocol with optional security features. Furthermore, the CBT architecture and security features combined have provided a low-cost, scalable solution to the *multicast key distribution problem*.

We have also shown how CBT has made explicit provision in its network-layer header for *flow-ids*, and we have described why CBT complements the Resource Reservation Protocol, RSVP.

CBT has also provided an effective solution to "anycasting", which is a recently proposed service typically used for resource-discovery.

Finally, we presented a proposal made recently that provides a solution to ATM multicasting. This is a generic solution based on the CBT architecture that allows IP multicast over ATM networks. This proposal demonstrated CBT's relevance to ATM networks, and the proposed protocol *(ACBT protocol)* is based closely on our CBT protocol, presented in chapter 5.

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.

9.1 Ongoing and Future Work

Throughout this thesis we have identified several key areas relating to the CBT architecture and protocol that require more extensive investigation than has been possible within the scope of this thesis. These include, in no order of preference, the following:

- **core management:** various aspects of core management need investigating. These include: core selection criteria, core placement strategies, and core announcement.

  With regard to core announcement, a further question is: should core identities remain hidden from users, which would require the protocol itself to establish core identities, or should users be involved in core selection?

- **dynamic core placement:** it is arguable whether a group's core(s) should change as group membership dynamics change. The trade-offs are periodic loss of connectivity vs. possibly more efficient distribution trees. Extensive testing should give an indication as to the validity of dynamically placing core routers.

- **interoperability:** this is one of CBT's most important issues, and several issues remain to be resolved, or concretely specified in the case where there may be several approaches that can be taken.

  Currently, the most significant unresolved issue is that of "overlapping" delivery trees, discussed in chapter 5.

- **control traffic aggregation:** potentially, control traffic imposes a significant burden in two areas. Firstly, CBT routers incur the processing costs of control traffic from "neighbouring" tree routers, which increases linearly with the number of groups traversing a CBT router. Secondly, as control traffic increases, the bandwidth consumed on a particular link increases. As a result, strategies for alleviating control traffic need to be investigated.

- **group access control:** finer-grained (host-level) group access control needs to be introduced to multicast. Because of the nature of multicast (having a single transmission received by multiple stations) it is uncertain whether host-level group access control is possible. Should host-level group access control be possible, it will undoubtedly require changes to the internetwork service interface of IP hosts [26].

- **"the global picture":** we concluded that each of the multicast protocols described in this thesis has its domain of applicability in the Internet. This means that in reality, the Internet will comprise a collection of heterogeneous "clouds", each operating any one of the given multicast protocols. It is therefore essential that interoperability between all schemes is investigated in order to provide seamless, Internet-wide multicast operation.

9.2 Summary of Main Contributions

The main contributions of this thesis are the following:

- the design and development of a new multicast architecture and associated network layer protocol for datagram (inter)networks. This new architecture offers considerable scaling improvements over existing multicast schemes, in particular for *many sender* multicast applications and scenarios.

  Our work motivated, and was catalytic, in the development of PIM.

- an analysis of multicast security, identifying why certain threats are compounded by the multicast paradigm.

  We introduced specific counter-measures to alleviate the threats we identified. These counter-measures are primarily subnetwork-level *group access control*, and *multicast transit traffic control*.

- we presented the CBT security architecture, which provides optional network layer security which is integrated into the CBT protocol. This security architecture provides for the authentication of hosts and routers that wish to become part of a CBT delivery tree.

  The CBT security architecture also provides a low-cost, scalable solution to the *multicast key distribution problem*, by delegating key distribution capability to already-verified on-tree nodes.

- we presented a proposal (that is not a contribution of the author) that offers a solution to ATM multicasting. This solution is based closely on our CBT architecture and protocol. It demonstrates CBT's flexibility in offering a multicast solution to both connectionless and connection-oriented networks.

Glossary

soft state the term used to describe state in the network (routers) which is transient, i.e. it requires periodic refreshing, otherwise times out.

hard state the complement of soft state, hard state refers to information that must be put explicitly, by means of some protocol, into the nodes of a network. It must also be explicitly removed when it is no longer required.

domain synonymous with *routing domain*, *administrative domain (AD)*, and *autonomous system (AS)*, a domain is a topological region administered by a single entity. For example, a domain may comprise a campus network.

subnet(work) the result of the division of a single network into smaller, more manageable physical subnetworks. In the Internet Protocol (IP), this is done by extending the network portion of an IP network address to encompass some of the host portion of the address.

leaf subnetwork a subnetwork that is not used by any router to reach a particular destination, i.e. a subnetwork with only one router attached.

anycast *anycasting* is a proposed best-effort, stateless, datagram delivery service which is used by hosts primarily to locate particular services in an internetwork. Its goal is to optimize the use of network resources in that a client should only have to transmit a single request to an "anycast address", and the nearest server bound to that address will respond.

dense mode the term used by PIM (Protocol Independent Multicast protocol) to describe "flood and prune" multicasting. Because of the nature of the "flood and prune" technique, it considered only appropriate for use within resource-rich environments, for example, campus networks.

sparse mode the term used by PIM (Protocol Independent Multicast protocol) to describe multicasting between sparsely distributed members of a group, typically in the wide-area context. That is, the number of members interested in receiving multicast traffic is far fewer than those *not* interested in receiving the traffic.

strong authentication *strong authentication* is authentication based on the use of public key (asymmetric) cryptosystems to protect the exchange of validating information.

digital signature the mechanism by which a message originator appends a compressed string (which is typically a function of the message itself) encrypted with the originator's secret key. On receipt, the digital signature is verified using the sender's public key. A verified signature proves message integrity and origin authenticity.

re-keying a strategy used by communicating parties that provides a means to change an encryption key regularly. This involves the initial distribution of a *key encrypting key (KEK)*, which is used periodically to encrypt a new data encryption key. This strategy alleviates the problem of an encryption key being "broken" by cryptanalysis.

MBONE the set of workstation routers in the Internet that are running an instance of the distance-vector multicast algorithm called DVMRP. As such they create an overlay, or *virtual topology*, of the Internet.

multicasting a technique that enables a single packet transmission to reach one or more destinations, or *group*.

multi-access network (LAN) a network (or subnetwork) with multiple communicating routers attached.

designated router (DR) in the context of a multicast routing protocol, a DR is a router on a multi-access network that is responsible for forwarding multicast packets on and off the LAN.

rendezvous point (RP) the term used by PIM (Protocol Independent Multicast protocol) to describe a set of pre-defined network routers that form part of a shared delivery tree which are used for the purpose of receivers *meeting* senders – senders send to all RPs of a set, and each receiver joins one RP.

core router the term used by CBT (Core Based Tree protocol) to describe a pre-defined router (one of a small set) that forms part of the backbone of a CBT delivery tree.

Other routers wishing to become part of the same delivery tree use one of the core routers as the target for join requests.

non-core router the term used by CBT (Core Based Tree protocol) to describe those routers that comprise a CBT delivery tree but are not core routers.

IGMP (Internet Group Management Protocol) the protocol implemented in hosts and multicast routers, and used by the latter to establish multicast group presence on all directly-attached subnetworks.

RPF (Reverse-Path Forwarding) designed originally as a broadcast technique (and later adapted for multicasting by Deering), the RPF technique is as follows: 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 (child) links. Otherwise, discard the packet.

Key Distribution Centre (KDC) a centralized network entity responsible for maintaining and securely distributing conventional encryption keys to requesting parties.

Bibliography

[1] G. Armitage. IP Multicast over UNI 3.0 based ATM Networks. *Working draft*, August 1994.

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

[3] R. Atkinson. *Private communication*, July 1994.

[4] R. Atkinson. IPv6 Authentication Header. *Work in progress*, August 1994.

[5] R. Atkinson. IPv6 Security Architecture. *Work in progress*, September 1994.

[6] A. Ballardie and J. Crowcroft. Multicast-Specific Security Threats and Counter-Measures. In *ISOC Symposium on Network and Distributed System Security*, February 1995.

[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] A. J. Ballardie. Scalable Multicast Key Distribution. *Working draft*, November 1994.

[9] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, and A. Secret. The World Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.

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

[11] 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.

[12] R. Braden, D. Clark, S. Crocker, and C. Huitema. Security in the Internet Architecture. *RFC 1636*, June 1994.

[13] T. Bradley and C. Brown. RFC 1294, Multiprotocol Interconnect over Frame Relay. *SRI Network Information Center*, January 1992.

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

[15] J. Bibb Cain. PIM vs. CBT Comparison. *IETF IDMR wg presentation*, April 1995.

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

[17] CCITT. X.25, Interface between Data Terminal Equipment (DTE) and Data Communicating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks. *Blue Book*, 1988.

[18] D. Cheriton. Sirpent: A High Performance Internetworking Approach. In *Conference Proceedings of ACM Sigcomm'89*, pages 158–169. ACM SIGCOMM, September 1989.

[19] 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.

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

[21] J. Crowcroft. RFC 1165, The Network Time Protocol (NTP) over the OSI Remote Operations Service. *SRI Network Information Center*, June 1990.

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

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

[24] 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.

[25] 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.

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

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

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

[29] M. Douglas. An Interoperable Analysis of SNDS SP3 and ISO NLSP. In *8th Annual Computer Security Applications Conference*, pages 193–201, November 1992.

[30] D. Eastlake. Practical IP Security (PIPS). *Work in progress*, March 1994.

[31] D. Eastlake and C. Kaufman. DNS Security Protocol Extensions. *Work in progress*, March 1994.

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

[33] D. Estrin and G. Tsudik. End-to-End Argument for Network-Layer Access Controls. *Journal of Internetworking: Research and Experience*, 2(2), May 1991.

[34] D. Estrin and G. Tsudik. Secure Control of Transit Internetwork Traffic. *Computer Networks and ISDN Systems*, 22:363–382, 1991.

[35] R. Braden et al. Slide Presentation. *RSVP working group meeting*, July 1994.

[36] P. Francis. *Addressing in Internetwork Protocols.* PhD thesis, University College London, University of London, 1994.

[37] P. Francis and R. Govindan. Flexible Routing and Addressing For a Next Generation IP. In *Conference Proceedings of ACM Sigcomm'94*, pages 116–125. ACM SIGCOMM, September 1994.

[38] L. Gong. New Protocols for Third-Party-Based Authentication and Secure Broadcast. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 176–183, November 1994.

[39] L. Gong and N. Shacham. Elements of Trusted Multicasting. In *Proceedings of the IEEE International Conference on Network Protocols*, pages 23–30, October 1994.

[40] R. Grimm. Security on Networks: Do We Really Need It? *Computer Networks and ISDN Systems*, 17:315–323, 1989.

[41] N. Haller and R. Atkinson. RFC 1704, On Internet Authentication. *SRI Network Information Center*, October 1994.

[42] H. Harney, C. Muckenhirn, and T. Rivers. Group Key Management Protocol (GKMP) Architecture. *Work in progress*, 1994.

[43] J. Hyman, A. Lazar, and G. Pacifici. Joint Scheduling and Admission Control for ATS-based Switching Nodes. In *Conference Proceedings of ACM Sigcomm'92*, pages 223–234. ACM SIGCOMM, October 1992.

[44] J. Ioannidis, M. Blaze, and P. Karn. swIPe: Network Layer Security Protocol for IP. *Working draft*, December 1993.

[45] ISO. Communication of Open Systems, The Basic Reference Model, Part 2: Security Architecture.

[46] ISO. OSI Recommendation X.509: The Directory – Authentication Framework. pages 48–80, 1988.

[47] V. Jacobson. Multimedia Conferencing on the Internet. *ACM Sigcomm'94 Tutorial, London*, September 1994.

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

[49] V. Jacobson and S. McCanne et al. "vat" manual page (under unix). 1994.

[50] P. Janson and R. Molva. Security in Open Networks and Distributed Systems. *Computer Networks and ISDN Systems*, 22:323–346, 1991.

[51] J. Kadirire. Minimising Packet Copies in Multicast Routing by Exploiting Geographic Spread. *ACM Computer Communication Review*, 24(3):47–62, 1994.

[52] B. Kahin. RFC 1192, Commercialization of the Internet Summary Report. *SRI Network Information Center*, November 1990.

[53] R. M. Karp. *Reducibility among combinatorial problems (In: Complexity of Computer Computations)*. Plenum Press, 1972.

[54] S. Kent. Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. *RFC 1422*, February 1993.

[55] S. T. Kent. Internet Privacy Enhanced Mail. *Communications of the ACM*, 36(8):48–59, August 1993.

[56] S. Kille, E. Huizer, V. Cerf, R. Hobby, and S. Kent. RFC 1430, A Strategic Plan for Deploying an Internet X.500 Directory Service. *SRI Network Information Center*, February 1993.

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

[58] L. Kou, G. Markowsky, and L. Berman. A Fast Algorithm for Steiner Trees. *IBM Research Report No. RC-7390 (31782)*, November 1978.

[59] B. Kumar and J. Crowcroft. Integrating Security in Inter-Domain Routing Protocols. *ACM Computer Communication Review*, 23(5):6–16, 1993.

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

[61] M. Laubach. RFC 1577, Classical IP and ARP over ATM. *SRI Network Information Center*, January 1994.

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

[63] F. Liaw. Multicast Routing and Signaling Protocol – Architectural Overview. *ATM Forum draft*, October 1994.

[64] J. Linehan. Comparison of Network-Level Security Protocols. *Draft report, contact: linehan@watson.ibm.com*, June 1994.

[65] M. Macedonia and D. Brutzman. MBONE Provides Audio and Video Across the Internet. *IEEE Computer*, pages 30–36, April 1994.

[66] Sun Microsystems Inc. RFC 1057, Remote Procedure Call Protocol Specification version 2. *SRI Network Information Center*, June 1988.

[67] P. Mockapetris. RFC 1035, domain names - implementation and specification. *SRI Network Information Center*, November 1987.

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

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

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

[71] R. Needham and M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.

[72] C. Partridge. *Gigabit Networking*. Addison-Wesley Professional Computing Series, 1994.

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

[74] C. Partridge and S. Pink. An Implementation of the Revised Internet Stream Protocol. *Internetworking: Research and Experience*, 3(1):27–54, March 1992.

[75] R. Perlman. An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN. In *9th Data Communications Symposium*, pages 44–53, September 1985.

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

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

[78] J. Postel. RFC 791, Internet Protocol. *SRI Network Information Center*, 1981.

[79] J. Postel. RFC 793, Transmission Control Protocol. *SRI Network Information Center*, September 1981.

[80] Jon Postel. RFC 791 internet protocol. *DDN Network Information Centre*, September 1981.

[81] B. Rajagopalan. Reliability and Scaling Issues in Multicast Communication. In *Conference Proceedings of ACM Sigcomm'92*, pages 188–198. ACM SIGCOMM, August 1992.

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

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

[84] Y. Rekhter and T. Li. An Architecture for IP Address Allocation with CIDR. *RFC 1518*, September 1993.

[85] R. Rivest. RFC 1321, The MD-5 Message Digest Algorithm. *SRI Network Information Center*, April 1992.

[86] J. Saltzer, D. Reed, and D. Clark. End-to-End Arguments in System Design. *ACM Transactions on Computer Systems*, pages 277–288, June 1984.

[87] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *Work in progress*, November 1994.

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

[89] S. Shenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines. In *Conference Proceedings of ACM Sigcomm'94*, pages 47–57. ACM SIGCOMM, September 1994.

[90] 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.

[91] K. Sollins. RFC 1350, The TFTP Protocol (Revision 2). *SRI Network Information Center*, July 1992.

[92] International Standards Organisation. Intermediate System to Intermediate System Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-Mode Network Service (ISO 8473), ISO 10589. 1991.

[93] International Standards Organization. Network Layer Security Protocol, draft standard 11577. November 1992.

[94] S. Stubblebine. Security Services for Multimedia Conferencing. In *Proceedings of 16th National Computer Security Conference*. IEEE, September 1993.

[95] C. Topolcic. RFC 1190, Experimental Internet Stream Protocol v2. *SRI Network Information Center*, October 1990.

[96] Gene Tsudik. *Access Control and Policy Enforcement in Internetworks*. PhD thesis, University of Southern California, 1991.

[97] V. Voydock and S. Kent. Security Mechanisms in High-Level Network Protocols. *ACM Computing Surveys*, 15(2):135–171, 1983.

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

[99] I. Wakeman, J. Balot, and T. Turletti. Scalable Feedback Control for Multicast Video Distribution in the Internet. In *Conference Proceedings of ACM Sigcomm'94*, pages 58–67. ACM SIGCOMM, September 1994.

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

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

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

[103] Lixia Zhang. Private communication. January 1995.