A HYBRID APPROACH TO QUALITY OF SERVICE MULTICAST ROUTING IN HIGH SPEED NETWORKS

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT AT CANTERBURY
IN THE SUBJECT OF COMPUTER SCIENCE
FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY.

By
John S. Crawford
November 1998

Contents

List of Tables

List of Figures

Abstract

Multimedia services envisaged for high speed networks may have large numbers of users, require high volumes of network resources and have real-time delay constraints. For these reasons, several multicast routing heuristics that use two link metrics have been proposed with the objective of minimising multicast tree cost while maintaining a bound on delay. Previous evaluation work has compared the relative average performance of some of these heuristics and concludes that they are generally efficient.

This thesis presents a detailed analysis and evaluation of these heuristics which illustrate that in some situations their average performance is prone to wide variance for a particular multicast in a specific network. It concludes that the efficiency of an heuristic solution depends on the topology of both the network and the multicast, which is difficult to predict.

The integration of two heuristics with Dijkstra's shortest path tree algorithm is proposed, to produce a hybrid that consistently generates efficient multicast solutions for all possible multicast groups in any network. The evaluation results show good performance over a wide range of networks (flat and hierarchical) and multicast groups, within differing delay bounds.

The more efficient the multicast tree is, the less stable it will be as multicast group membership changes. An efficient heuristic is extended to ensure multicast tree stability where multicast group membership is dynamic. This extension decreases the efficiency of the heuristic's solutions, although they remain significantly cheaper than the worst case, a shortest delay path tree.

This thesis also discusses how the hybrid and the extended heuristic might be applied to multicast routing protocols for the Internet and ATM Networks.

Additionally, the behaviour of the heuristics is examined in networks that use a single link metric to calculate multicast trees and concludes one of the heuristics may be of benefit in such networks.

Acknowledgements

Firstly I must acknowledge the Engineering and Physical Sciences Research Council for funding research at the University of Kent at Canterbury into "Multicast Routing for Broadband Networks". I must also thank Gill Waters, my Project Supervisor and Principal Investigator for her encouragement, guidance and support with my work. It was Gill who introduced me to the topic of multicast routing in the first place, and she who made it possible for me to continue working with it. And for reviewing and commenting on my draft, thanks to Tim Hopkins.

Thanks to Maria Stegiou and Florence Benoy who, although they are researching in different areas of Computing, gave me moral support and helped me reflect on my work and how I set about it.

I would also like to thank the now absent postgradute Kent students, who carried out work associated with this thesis, whom I had the opportunity to help supervise. Particularly I would like to thank Matt Salmon for endless conversations about "growing trees", "pathological walkabouts" and "genetics". Gill and I are planning to publish a paper with Matt fairly soon.

Publications

- Heuristics for ATM Multicast Routing. John Crawford and Gill Waters. In Demetres Kouvatsos, editor, *ATM'98 Sixth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks. Participants Proceedings: Tutorial Papers*, pages 5/1-5/18, University of Bradford, July 1998. IFIP Workshop TC6 IFIP Working Groups WG6.3 and WG6.4, UK Performance Engineering Workshop.

- Low Cost Quality of Service Multicast Routing in High Speed Networks. J.S.Crawford and A.G. Waters. Technical Report 13-97, University of Kent at Canterbury, December 1997.

- A Hybrid Approach to Quality of Service Multicast Routing. J.S. Crawford and A.G. Waters. In Demetres Kouvatsos, editor, *ATM'97 Fifth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks. Participants Proceedings: Research Papers*, pages 61/1-61/9, University of Bradford, July 1997. IFIP Working Group 6.3 on Performance of Communications Systems and IFIP Working Group 6.4 on High Performance Networking, UK Performance Engineering Workshop.

- Low-cost ATM Multicast Routing with Constrained Delays. A.G Waters and J.S. Crawford. In J. Pascual and A. Danthine, editors, *Multimedia Telecommunications and Applications*, volume 1185 of *Lecture Notes in Computer Science*, pages 23-40, Barcelona, Spain, November 1996. 3rd International COST 237 workshop, Springer, Heidelberg.

- An Heuristic for Lower Cost Multicast Routing in the Internet. John Crawford and Gill Waters. Technical Report. 25-96, University of Kent at Canterbury, June

1996. This work was supported by EPSRC Grant GR/K55837 and presented at the IDMR of the 36th IETF, Montreal, Canada, June 1996.

- Heuristics for ATM Multicasting. J.S.Crawford and A.G.Waters. Being considered for publication in a special issue of *Computer Networks and ISDN Systems* on *ATM Networks: Performance Modelling and Analysis.*

Glossary

AS	Autonomous System
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CBT	Core Based Trees
CCET	Constrained Cheapest Edge Tree
CSPT	Constrained Shortest Path Tree
CST_c	Constrained Steiner Tree
DAR	Dynamic Alternative Routing
DNHR	AT&T's Dynamic Non-Hierachical Routing
DTL	Designated Transit List
DVMRP	Distance Vector Multicast Routing Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ITU	International Telecommunications Union
LAN	Local Area Network
MOSPF	Multicast Open Shortest Path First
MST	Minimal Steiner Tree
mCCET	Modified Constrained Cheapest Edge Tree
OSPF	Open Shortest Path First
PIM	Protocol Independent Multicast
PNNI	Private Network/Network Interface

QoS	Quality of Service
QoSR	Quality of Service Routing
RP	Rendezvous Point
RPB	Reverse Path Broadcasting
RSVP	Resource Reservation Protocol
SPT	Dijkstra's Shortest Path Tree
sCSPT	stable Constrained Shortest Path Tree
sCST_c	stable Constrained Steiner Tree
ST2+	Internet Stream Protocol, Version 2 +
TOS	Type Of Service
UNI	User/Network Interface
VCI	Virtual Channel Identity
VPI	Virtual Path Identity
WG	Working Group

Note on the bibliography

Several of the documents referenced in the bibliograhy to this thesis are Internet-Drafts, and represent work in progress at the time of their writing (see [48], [28], [50] and [70]). These documents have a lifetime of only six months, after which they must either be superceded or deleted, and so may no longer be available from their referenced source. In this case the author has copies of the original documents, which can be made available on request.

The reader is refered to the Internet-Drafts index at

http://www.ietf.org/internet-drafts/1id-index.txt

for a list of the Internet-Drafts published to date. Where any document has been superceded, a later version may be found at

http://www.ietf.org/internet-drafts/

or, if the document has been superceded by an RFC, at

http://www.rfc-editor.org/

Chapter 1

Introduction

Of the new services envisaged for emerging high speed networks, such as B-ISDN/ATM, many will require point to multipoint connections. Some of these services, particularly those using interactive multimedia communications, will require real-time bounded delays on data delivery and will consume high bandwidths. They may also have large numbers of subscribers, as might be the case with video-on-demand distribution services, video conferencing, and multi-party interactive video games. With the growing use of these and other networked services, network capacity may well become a limiting factor in their deployment [30]. In the future, network connections for these types of applications may need to make efficient use of network resources while maintaining real-time delay bounds on data delivery.

Multicasting uses less network resources than multiple point to point connections between a multicast source and its destination, because routes can be aggregated where they have links in common. In multicasting schemes currently being implemented (see, for example [41] and [61]), routes are aggregated along common links in their shortest paths between source and destinations. Route calculation in these schemes minimises each delay path in the multicast delivery tree. However, further route aggregation may be possible if the constraint on delay required by an application is taken into account. The path taken between a source and each of several destinations may not necessarily have to have the lowest delay, so alternative paths of longer delay may offer more common links for aggregation. In fact, if the delay is unimportant, then maximum route aggregation, and hence minimum network resource usage, may be the optimisation goal.

In deciding which routes to aggregate in a multicast delivery tree, we have a conflict between the efficient use of network resources and the delay between a source and each multicast destination.

The multicast requirements of the applications envisaged above have this conflict because they demand both bounded real-time delays on data delivery and efficient use of network resources.

It is a fundamental assumption of this work that, in high speed networks, the constraints on delay required by the applications envisaged will, in general, be greater than the delays provided by any corresponding shortest path multicast trees. It should, therefore, be possible to find cheaper multicast trees, constructed from longer paths than the shortest, and still maintain a specified arbitrary upper bound on delay.

A second, and equally important constraint, is the issue of how long it might take to calculate a bounded-delay, network-efficient multicast tree. We know that calculation of a shortest path multicast tree can be accomplished with a time complexity of at most $O(n^2)$ by constructing a broadcast tree using Dijkstra's shortest path algorithm [1] which is then pruned back to the multicast group. The efficiency of these solutions (total cost of the edges in the multicast tree) is not taken into account by the algrorithm. We can calculate a most efficient broadcast tree (total cost of the edges in the multicast tree is a minimum) using an algorithm such as Prim's [1] minimum spanning tree, which also has a time complexity of $O(n^2)$. Pruning such a broadcast tree back to a smaller multicast group would not necessarilly result in the most efficient use of network resources. It could, in fact, be rather more expensive than a shortest path tree ! Algorithms such as Prim's ignore delay altogether. The problem of finding an efficient solution for a multicast group smaller than the entire network is known as the Steiner tree problem, which is NP-complete [36]. There are heuristic algorithms of acceptable time complexity that give reasonable solutions for graphs which are proper subsets of a set of network nodes. Our problem is also NP-complete since we want to find a multicast tree that is bound by an upper abitrary delay and that also makes efficient use of network resources.

Kompella et al [38] first addressed the issue of how to calculate efficient multicast trees bound by an arbitrary upper delay bound. They assumed that the efficiency of a network could be calculated using a cost metric for each link; this cost might represent any one of several practical values in a real network, e.g. residual bandwidth or a monetary cost. At the time of route calculation the cost of using a link can be considered to be constant, although it may vary with time and/or network load. Since then there have been a number of other proposals for solutions to this problem, for example [11], [63] and [52].

Previous evaluation work ([11], [52], [63], [64] and [68]) shows that, on average, these heuristics perform well. However, further detailed analysis and evaluation of some of these heuristics has shown that there is a wide variance in the efficiency of their solutions [14]. Whilst on average one heuristic may be more efficient than another, either for all multicast group sizes or for a particular range of multicast group sizes, there are some multicast group and network combinations where this position is reversed. In particular, we have found that as the multicast group membership changes, relative to the size of the network, the heuristic that provides the most efficient multicast solution also changes. The results of our evaluation work indicates that it is difficult to predicit which heuristic provides the most efficient solution for any particular multicast/network combination. The variance in the efficiency of the heuristic solutions is wide enough that on occasions Dijkstra's shortest path algorithm (SPT) calculated on delay is more efficient. By selecting two heuristics that can be efficiently integrated with each other and the SPT algorithm, we propose a hybrid heuristic that produces reasonably consistent and efficient solutions to the multicasting problem, with an acceptable order of time complexity for all possible multicast groups in any network.

In addition to the requirements of the delay bound, low cost, multicast routing algorithms we have identified so far, we have a further multicast tree characteristic that must be considered when looking at the behaviour of multicast groups. If a multicast group remains static throughout its lifetime, as might be the case for a private video meeting, the multicast delivery tree will remain stable for the duration of the communication. For these types of connection virtual channels (or soft circuits) will be established to reserve appropriate network resources which it would be impractical to re-route due to short term fluctuations in network loading. On the other hand some applications, like public multi-party video games, will have dynamic multicast groups where members join and leave throughout the duration of the communication. As Doar and Leslie suggest in [22] there are (at least) three ways to deal with dynamic multicast groups. The first option is to re-calculate the multicast tree everytime the multicast groups changes. This may have unacceptable repercussions on the members already in the group (or remaining in the group) not only because of the tree recomputation time, but also because the multicast tree topology may change. The second option is to start with a near optimal multicast tree, then make as few changes to the tree as possible, as members join or leave. Salmon [53] and Kuzminski [39] have independently explored this approach for one of the heuristics we have evaluated [63]. Salmon used a genetic algorithm to join new members to a multicast tree, while Kuzminski used a deterministic method. Work by Waxman ([21] and [66]) has shown that this approach occasionally results in solutions of very high inefficiency. The third option is to choose a less efficient solution that remains static as members join or leave. We have found that, in general, the more efficient the solutions of an heuristic are, the less stable these solutions will be when multicast group membership is dynamic.

For these reasons we propose an extension to an existing heuristic, so that the solutions it generates are stable, when multicast group membership is dynamic. This extension reduces the cost efficiency of the heuristic (as expected) but its performance is still a significant improvement over that of a shortest delay path tree solution.

For the purpose of evaluating the heuristics we used a simple random graph model proposed by Waxman [66] and modified by Doar [21]. We also developed a hierarchical network model, based on the Doar model, to represent interconnected subnetworks as might be found in the Internet.

While the central theme of this work has been the study of heuristics that calculate low cost multicast trees bounded by an arbitrary delay, we have also investigated the application of the same heuristics to single metric multicasts. That is, we have assessed the performance of the heuristics discussed, in networks that use only a single metric per link to calculate routes, such as is the case in the current Internet. With the exception of the Waters heuristic [63], all the heuristics evaluated have a tendency to converge towards either a minimal cost Steiner tree or a shortest path tree, as the two metrics they use to calculate trees converge. Thus the Waters heuristic may benefit multicast routing schemes in networks that calculate routes using a single link metric. We present this work in a separate chapter, before our conclusions.

The work reported in this thesis developed from previous work by Waters [63] and by the author [11], and was subsequently supported by an Engineering and Physical

Sciences Research Council grant (reference GR/K55837) [14]. The author's original contribution to this work, presented in this thesis, is the

- Analysis of the behaviour, and evaluation of the performance, of heuristics that calculate low cost multicast trees, and are bound by an upper arbitrary delay.
- Development of a new random graph model to represent interconnected sub-networks. The sub-networks may be either stub networks, transit networks, or both stub and transit networks.
- The proposal and evaluation of a hybrid of heuristic algorithms to calculate low cost delay bound multicast trees, that offers consistent performance for all possible multicast groups, in any network.
- The proposal and evaluation of an extension to an existing multicast heuristic to provide a stable solution for dynamic multicast groups.
- Characterisation of multicast groups and how this affects the application of the hybrid heuristic and the extended stable solution heuristic.
- The evaluation of one of the heuristics using a single edge metric, instead of two, and an explanation of how this modification might benefit multicast routing in the Internet. An overview of how the modified heuristic could be integrated into an Internet routing protocol is given.

The rest of this thesis is organised as follows

- Chapter 2 provides an overview of networks, routing and resource reservation. We identify different types of multicast groups and their characteristics, and then describe the multicasting and resource reservation methods used, or proposed for use, in the Internet and B-ISDN/ATM networks.
- In Chapter 3 we define the Bounded Delay, Minimum Cost Multicast Routing Problem. We follow this with explanations of how the heuristics we have evaluated work, and describe their behaviour. We then describe our evaluation method, and introduce the new network model we have used for some of our work. The results of the evaluation of the heuristics are presented.

- Chapter 4 introduces the Hybrid solution to the Low Cost, Bounded Delay Multicast Routing problem. We present the results of the Hybrid evaluation.

- Chapter 5 describes the extension we have made to Sun and Langendoerfer's heuristic [56] in order to calculate stable multicast solutions of reasonable efficiency, for dynamic multicast groups. We assess the performance of this heuristic against the original.

- Chapter 6 addresses how the Hybrid heuristic, and the extended version of the Sun and Langendoerfer heuristic, might be applied in networks.

- Chapter 7 presents an evaluation and proposal for using the Waters [63] heuristic in the current Internet, with a single metric.

- In Chapter 8 we conclude our work and identify further research that needs to be undertaken in this field.

Chapter 2

Multicast Routing

2.1 Networks, Routing and Resource Reservation

There are two major classes of routing methods to which routing algorithms can be applied: fixed and dynamic. Fixed routing assumes that the topology of the network and traffic requirements are known in advance, and remain constant, so that paths through the network can be calculated and downloaded into the network before they are used. Dynamic routing assumes that the network topology and traffic load are likely to change and may not be globally known throughout the network, and that route calculations have to be performed regularly to accommodate dynamic changes within the network.

Routes through a network can be calculated at the data source, or by a route server, or in a distributed manner by each of the nodes along the path. The process that calculates a path through the network is also responsible for establishing the path in the network. Source based dynamic routing methods require full knowledge of the network topology and, possibly, of traffic loading as well, to perform route calculations. In distributed dynamic routing, paths through the network are calculated at multiple points along the path between the source and destination as data is forwarded. As the topology of the network and the traffic load varies, paths through the network may change. The topology and traffic load information required by each routing point in the network varies depending on the routing algorithm used. Distributed dynamic routing algorithms vary from those that require only local information to those that require global knowledge of the network [57].

All routing methods have advantages and disadvantages. For example, in reliable networks that support permanent connections between end points, route calculation can be performed “off-line” and installed in the network prior to use. There is no necessity for the network itself to support either route calculation or maintenance of the information required to calculate routes. In such networks the switches have low functionality and traffic load for each communication is guaranteed. A disadvantage of such a network and routing method is that all changes to network load and end-to-end connections have to be made “off-line”. The network cannot dynamically adapt to changing usage. At the other extreme, networks that support distributed dynamic routing may have considerable functionality in their routers and require regular exchanges between routers of large volumes of topological and traffic loading information. While such a scheme enables the network to adapt to changes in network usage, it requires additional capacity and processing capability to manage its dynamic nature, irrespective of the volume of traffic it is carrying at any time. Source based dynamic routing methods are a compromise between the extremes of fixed routing and distributed dymanic routing. Not all switching points in the network need to know the topology or loading of the entire network, but the route calculating points are responsible for establishing paths through the network. How often this is done depends on the type of network. It would not be a very practical method for routing in a datagram network, but would be for a virtual circuit network.

Introducing multicasting to any routing scheme brings additional complexity to the routing method. In the simplest case this may be the extraction and combination of paths between the multicast source and its destinations to form the multicast tree. In the most complex cases it may involve the exchange of multicast destination location data, in addition to link state data, between all routing points in the network. This is the case with MOSPF [41] which floods link state advertisements and multicast group membership throughout the network. The introduction of multicast routing heuristics to dynamic routing methods may also bring with it the problem of multicast tree reconfiguration as destinations join and leave the multicast group. The resource savings made by using a near optimal minimum cost solution may be outweighed by the multicast tree reconfiguration costs, particularly if the multicast group membership changes frequently.

The main characteristics of networks, that have an influence on the choice of multicast routing algorithm to be used by their routing protocols are :-

- Connectionless networks use distributed route calculations, performed by every router in the network, to forward user data across a network. This allows routers to alter paths taken by user data as changes in the network topology or traffic conditions occur, but incurs the cost of path calculation by each router along the path. User data is carried across the network according to the "best efforts" of the network.

- Resource reservation for flows of user data in connectionless networks requires the establishment of "pseudo" paths between the source and destination routers ([10] and [69]). In order to reserve resources for a flow of user data, and to maintain the ability of the network to re-route user data, the end routers of the communication periodically exchange resource reservation messages. These messages are used by intermediate routers to reserve the requested resources along a path for the flow of user data. If these resource reservation messages cease to be received by intermediate routers along a user path, the resources allocated to the path are relinquished. Resource reservation messages use the same paths as the user data so, as paths change due to link failure, the resource reservation message also change paths. Clark called this concept "soft state" [10].

- Connection oriented networks use source based route calculations, performed at the network ingress router or by a route server, to find paths across the network for user data. These paths are established in the network as "hard state" connections between the source and destinations before any user data is transmitted. This enables switches along the user data path to quickly and efficiently pass incoming data onto the appropriate outgoing links with minimal processing. If links in the network fail then the source/destination path has to be removed and a new path calculated and established in the network.

- Resource reservation for user data paths in connection oriented networks is an integral part of the path calculation and establishment process. Once established, user data paths are held in place by hard-state. That is, the path resources remain intact until they are explicitly removed by the user.

- Fixed networks require user data paths to be pre-calculated and downloaded into the network before they are used. The network does not adapt to changes in topology or load variance.

- Resource reservation in fixed networks also needs to be pre-calculated and downloaded into the network.

2.2 Multicast Group Types

Multicasting protocols, currently implemented or in the process of development, provide support for two types of multicast group communication

1. Heavyweight: Where the sender knows who the receivers are and is responsible for adding them to the multicast (see, for example, [54] and [60]). Applications such as video conferencing and distributed gaming might require this type of service, since it may be important that the sender knows who is participating in the multicast. Heavyweight multicasting protocols are likely to use considerable state data to establish and maintain multicast trees, but may suffer little or no dynamic membership behaviour.

2. Lightweight: Where the sender is unaware of who the receivers are and the recievers are responsible for joining the multicast (see, for example [5], [19], [41] and [61]). Examples of applications that may require this type of service are multimedia lectures and other distribution services, where the sender does not need to know the identities of receivers or where they are located. Lightweight multicast trees require little state data for their establishment and maintenence, and their group membership is likely to be volatile.

These contrasting requirements result in different methods of multicast tree construction.

- For heavyweight multicasts, if all the recipients are known to the sender before data transmission starts and the multicast group is static, the complete multicast tree can be constructed once.

- If a heavyweight multicast group has some dynamic membership behaviour, then a multicast tree can be constructed for the initial group membership. However, the subsequent addition and deletion of destinations from the multicast tree must have as minimal an effect as possible on the topology of the initial multicast. Some consideration has been given to this problem by [53] and [39].

- A lightweight multicast does not know its membership, so its multicast tree construction has to be dynamic. The multicast tree must be readily adaptable to changing group membership.

2.3 Internet Multicast Routing Protocols

While multicasting within local area networks (LANs) is generally available, the ability to multicast across store-and-forward networks that interconnect LANs has not been available until recently, and is still being developed through the working groups (WGs) of the Internet Engineering Task Force (IETF). Much of the initial multicast development has been based on the original work of Deering and Cheriton [18] who proposed protocols that construct multicast trees rooted at a sending source. However, these source specific multicast solutions do not scale well for large internetworks because of the network resources they consume. For this reason multicast protocols that construct trees shared between a number of sources have been proposed by Ballardie et al [5] and by Deering et al [19].

Internet Protocol (IP) multicasting is the delivery of an IP datagram to a group of hosts, where a host group is identified by a unique IP multicast address. Multicast group addresses are either persistent and well-known administratively assigned IP addresses or transient IP addresses which are assigned to multicast groups for as long as they have members. Routers do not need to maintain a mapping between IP host group addresses and the IP addresses of the individual members of the host group because this binding is dynamic.

Delivery of datagrams is “best efforts”, as is the case with unicast IP datagrams. Datagrams are not guaranteed to reach all members of the multicast group, nor is the order of datagram arrival guaranteed.

Membership of an IP host group is dynamic and receiver initiated; host receivers may join and leave a multicast group at any time. A host that is not a member of a multicast group may send datagrams to a group and may have no idea which hosts belong to the group, or where they are located. A host group can have no members.

2.3.1 Internet Group Management Protocol (IGMP)

The Internet Group Management Protocol [17] is used by IP hosts and their local routers to join and leave multicast groups. Support for the protocol is a prerequisite for all hosts and routers that support IP multicasting.

The protocol uses two message types

1. Host Membership Query (Queries).

2. Host Membership Report (Reports).

Multicast routers use these IGMP messages to find out which multicast groups have host members on the subnetworks to which they are directly attached. Periodically, multicast routers broadcast "Queries" onto each of their locally attached subnetworks. Hosts on the subnetworks then reply to the multicast router using "Reports" to indicate to which multicast groups they belong. The protocol uses timers to spread the sending of "Reports" from hosts onto an attached subnetwork over short time intervals to avoid implosions of concurrent messages on the subnetwork. This mechanism also reduces the volume of "Reports" returned to the multicast router as hosts that receive a report for a group to which they belong need not send a report themselves. Multicast routers then forward remotely originated multicast group datagrams onto the subnetworks for which hosts have been registered as group members. If, after several periodic query transmissions on any subnetwork, no host replies are received for a multicast group previously active on the subnetwork, the multicast router assumes that there are no longer any host members for the group on the subnetwork and ceases to forward multicast datagrams to it. When hosts join multicast groups they send a report for the group to the multicast router rather than waiting for a "Query" first. This enables hosts to join groups immediately, rather than waiting for the multicast router to discover that the host wishes to join the group.

There is no explicit message, in the early version of IGMP, for a host to indicate that it no longer wishes to recieve data for a multicast group. The consequence of this is that a multicast router will continue to forward multicast data onto a subnetwork that has no hosts wanting to receive the data. Forwarding will continue until the timers used by the multicast router to wait for Host Membership Reports expire. The router will then know that there are no longer any hosts on the subnetwork that belong to the multicast group and so cease forwarding multicast datagrams. The IETF Inter-Domain Multicast Routing WG has developed a second version of IGMP [24] that will enable routers to immediately cease forwarding multicast datagrams when there are no longer any hosts on attached subnetworks belonging to the multicast group.

2.3.2 Multicast Shortest Open Path First (MOSPF)

The Open Shortest Path First (OSPF) [42] routing protocol is based on the ARPAnet routing protocol proposed by McQuillan in [40]. The protocol uses Dijkstra's shortest path algorithm (SPT)[1] to calculate routes in the network to which it is applied. Also know as Link State Routing, protocols based on Dijkstra's SPT are used in, or have been proposed for use in both connection oriented, e.g. [60], and connectionless network's, e.g. [42] and [45].

The SPT calculation requires complete topological knowledge of the network to which it is applied. In dynamic connectionless networks, routers must periodically flood the state of their incident links to all the other routers in the network so that every router can maintain complete knowledge of the networks topology. To forward unicast data, a router that receives a datagram uses the link state data and the SPT algorithm to calculate the shortest path, from the router to the datagram's destination, on which to send the data. The router does not need to know the source of the data it forwards since it calculates the shortest path from itself to the destination. The calculation requires only the destination identity and the network topology to find the forward path for the datagram.

The OSPF protocol is a link state routing protocol that has been designed to run between groups of routers that exchange routing information using a common protocol. Such a group of routers is termed an Autonomous System (AS) and OSPF is classified as an Interior Gateway Protocol (IGP) because it runs within an AS. To reduce the volume of link state data that must be flooded and maintained by routers throughout an AS, OSPF allows the grouping of contiguous networks and hosts into areas, each of which runs a copy of the basic OSPF protocol. Routers inside an OSPF area each maintain their own topological database of the area, which is not visible to routers in other areas. This means that the topological view a router has depends on the area it resides in, and is different from the topological views routers in other areas have. Areas are interconected either by routers that are present in two or more areas (Area Border Routers) or by networks that are not in any area, but which are connected to two or more areas by attached routers. The "network" that interconnects areas, which can be either contiguous or virtual, is termed the backbone network. A datagram routed between hosts in different areas will travel along an inter-area path from the local router to the area's Border Router. It then follows a backbone path between the source and destination areas and finally reaches the destination router along a path from the destination area's Border Router. Because the SPT calculation is rooted at each router along a path, all paths will be the shortest. Link state data from within each area is summarised and flooded throughout the backbone by the Area Border Routers. These summaries are also advertised internally to areas attached to the Border Routers, thus enabling routers within one area to calculate forwarding paths to destinations in other areas. As we shall see, this hierarchical routing structure impacts multicast route calculations.

For multicast data forwarding, the SPT must be rooted at the multicast source rather than at the forwarding router. To perform route calculation every multicast router in the network must therefore know, in addition to the networks link state data, the location of both the multicast sources and destinations for all multicast groups.

The Multicast Extensions to OSPF (MOSPF) [41] extend the capabilities of OSPF to provide source specific multicasting.

The MOSPF protocol registers multicast destination host group memberships with their local MOSPF router(s) by using the IGMP. The local MOSPF routers then flood the identities of hosts that have registered as receivers of the multicast group on their attached subnetworks to all other MOSPF routers in the area. This flooding process is periodically repeated for all multicast group hosts that remain registered with a local MOSPF router. This mechanism provides every multicast router with the identities of all the destinations for all multicast groups active within an area. MOSPF routers calculate multicast routes "on-demand" when the first datagram is received for a multicast group. This datagram contains the identity of the multicast source; the MOSPF router already has the identities of the multicast destinations and the topology of the network area. From these data the MOSPF router is able to calculate the forwarding path(s) for the multicast datagram, which may be cached for future use as the forwarding path will only need to be re-calculated if multicast destinations either join or leave the multicast group or if there are changes in the network topology. MOSPF supports calculation of multicast trees for each type of Internet Protocol Type of Service (TOS) [3]. However, it should be noted that the TOS feature has been removed from later editions of OSPF [43], and it will be removed from MOSPF [44]. Proposals for QoS OSPF are divided as to whether or not the TOS field should be used to specify QoS requirements [28][70]. Because of this interest TOS may continue to be developed.

Multicasting between MOSPF areas is managed in a different way to OSPF unicast routing. Each area in an AS will flood summaries of multicast group membership into the backbone. However, unlike OSPF, the backbone will not flood these data, nor will it advertise its own group memberships, into attached MOSPF areas. This procedure has been adopted to prevent the broadcasting of the locations of all members of all multicast groups throughout the AS, thus avoiding serious problems with multicast scaling. In order to reach multicast destinations that are outside a source's area, multicast area border routers are designated as "wildcard" multicast routers. This means that they belong to all multicast groups, and so receive user data for all multicast groups. Since the border routers know which multicast groups have members in the areas attached to the backbone, they are able to forward user data appropriately.

If a multicast is between a source in one area and a destination in another area, then the multicast tree no longer uses shortest paths, as it would if the user data were unicast. The reason for this is that a router performing the multicast tree calculation needs to know the distance from the source to itself. If the source is in an area remote to a multicast router this distance is not available, as the router only has the OSPF summary link state data describing the distance from the router to the source [41]. MOSPF therefore has to use the reverse path distance to the source for the SPT calculation across a network backbone. All additional forward links used in the tree calculation are also selected on the basis of their reverse path cost.

The distribution to, and storage of multicast source and destination data at every router in the network, coupled with the processing and storage costs of the SPT at each forwarding router for each multicast, impacts the scalability of link state multicasting.

For this reason MOSPF is considered to be suitable for intradomain use only.

Zhang et al [70] have made a proposal, through the Internet-drafts mechanism of the IETF, to extend both OSPF and MOSPF to include Quality of Service Routing (QoSR) for data flows. Best efforts traffic would be handled in the same manner as it is currently, since it requires no QoSR. These proposals have no authority, and the lifetime of such documents is limited to six months. However, the general direction these proposals are taking is of interest here. They suggest that the link state data exchanged between OSPF and MOSPF routers might include information about a link's available resources and that data flows would request that specified quantities of these resources be allocated to them. The proposals include ideas on what metrics might be used in route calculations, such as link delay and token bucket depth and rate. There is also a suggestion that, because of the significant increase these extensions would make to the volume of link state data maintained by routers, explicit route calculation by the data source router might be used instead of hop-by-hop calculation, and that routes should be fixed (pinned) once established. These proposals indicate a move towards a connection oriented approach for QoSR.

2.3.3 Distance Vector Multicast Routing Protocol (DVMRP)

The DVMRP [61] is based on refinements by Deering and Cheriton [18] of Dalal and Metcalfe's Reverse Path Broadcasting (RPB) method [15]. It is an Interior Gateway Protocol for use within an AS, based on distance-vector routing [31].

In reverse path broadcasting a router forwards a data packet if, and only if, it arrives on the shortest path from the router to the data source. The router forwards the data packet on all its outgoing links except the link on which it arrived. By this means all routers in the network receive a copy of the data. For multicast delivery this is a waste of network and router resources since not all routers require the data. DVMRP uses a modified form of RPB (Truncated RPB) in which routers are able to identify attached "child" and "leaf" routers, and to know which "child" and "leaf" routers are members of multicast groups.

"Child" routers are relative to a multicast source and are identified by their distance from the source. In figure 1, routers A and B have to decide which will forward multicast data from S towards C. All the routers periodically exchange distance-vector data

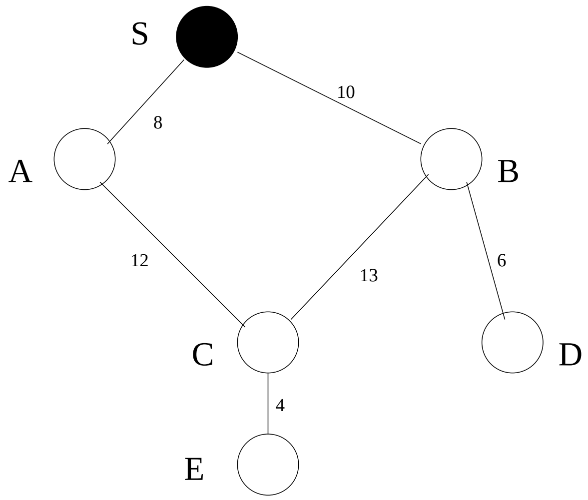

Figure 1: Child and Leaf Routers

indicating their distances from each other. From this data, routers A and B each discover that router A is the closest to the multicast source S and is therefore the "parent" of router C, and hence responsible for forwarding multicast data to C from S. On discovering that router A is the "parent" of C, router B will no longer forward multicast data received from S towards router C. Router D is a "child" of B and router E is a "child" of C. By this method routers will receive only one copy of a data packet, rather than multiple copies as in RPB.

A "leaf" router is a "child" router that no other routers use to reach the multicast source. In figure 1, router C is a "child" of A, but it is not a "leaf" router because the attached router E uses it to reach the source S. Routers D and E are "leaf" routers. "Leaf" routers only forward multicast data onto subnetworks that belong to appropriate multicast groups. By this means the multicast tree is truncated at the "leaf" routers.

Later versions of DVMRP have been proposed and implemented and are documented in IETF draft documents [48]. These later versions of DVMRP implement the Reverse Path Multicast method which allows "leaf" and up-stream "child" routers to be periodically pruned from the multicast tree provided that they have no dependent subnetworks belonging to multicast groups. The multicast tree periodically re-grows to discover if non-member subnetworks have subsequently joined a multicast group. The reader is referred to [18] for an expanation of this method.

The DVMRP only calculates a shortest path multicast delivery tree if the delay on all links is the same in both directions; otherwise some or all of the paths in the tree may not be the shortest. DVMRP routers do not need to know the topology of the network, but only the distances between themselves and all other routers in the network.

The periodic pruning and re-growing of multicast tree branches by DVMRP, irrespective of whether or not a branch has multicast group members attached, renders the protocol unsuitable for large scale internetwork multicasting.

2.3.4 Core Based Trees (CBT)

CBT is an architecture, proposed by Ballardie et al [5], for scalable internet multicasting and is based on the work of Wall [62]. The primary motivation for the development of CBT was to significantly improve the scaling factor for a multicasting method over that of the existing IP source rooted multicasting methods. Source rooted multicasting protocols, such as DVMRP and MOSPF, either use high volumes of bandwidth or require the exchange of large amounts of state data, irrespective of the number or distribution of multicast group members. For source rooted multicasting the scaling factor for each method is the product of the number of sources and the number of receivers in the multicast group, since each sender requires a source rooted multicast delivery tree. The objective of CBT is to reduce this factor to the number of receivers in the multicast.

A CBT consists of a primary core router and an ordered list of additional core routers (included for robustness) for each multicast group. At group initiation time the additional cores join the primary core to form a central hub of the CBT. The protocol does not define how cores are selected and placed in a network.

Hosts wishing to join a multicast group will use a directory service, such as X.500 [29], to obtain a multicast group's identity and the list of the group's core unicast addresses. Using IGMP the host informs its local CBT capable multicast router that it wishes to join a multicast group, and provides it with the group identity and core list obtained from the directory service. The local multicast router then sends a join request towards a core router for the group. The join request is forwarded, at each intermediate router along the path towards the core, until it either reaches the core, or reaches an intermediate CBT capable router that is already a member of the CBT

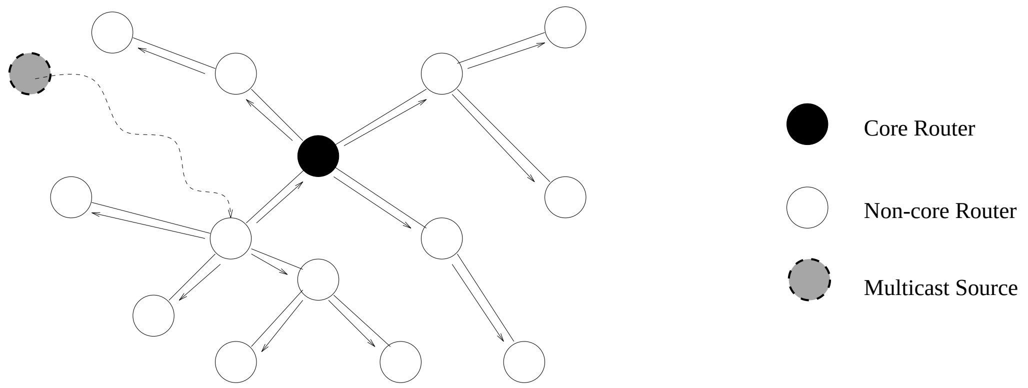

Figure 2: Multicast user data route in a CBT

for the group. In general the CBT router will reply to the join request by sending a join acknowledgement back along the path towards the host's local CBT router, thus establishing a branch in the multicast tree. All the CBT routers along the path become non-core routers for the CBT. There are exceptions to this action which occur for a variety of reasons, such as in loop detection. The reader is referred to [4] for a detailed description of CBT.

Sources wishing to send to a particular multicast group address their user data to the CBT core router for the group, and include the multicast group identity in the option field of the IP datagram. When the user data reaches any of the CBT routers for the multicast group (core or non-core), the core address in the IP datagram destination field is replaced by the multicast group identity from the option field, and the data is multicast across the CBT. Each CBT router will forward the multicast data across all interfaces it has for the group tree (including the path towards the core) except across that on which the data arrived. Figure 2 illustrates how user data is sent towards a CBT for a multicast group, and is then multicast to all the group recipients.

The CBT for any multicast group is maintained by the exchange of "keep alive" messages between adjacent routers along tree branches. If a subtree becomes detached from the main tree it can either clear itself down, in which case each router in the subtree will attempt to re-attach itself to the main tree, or the router that became disconnected from its parent may attempt to re-attach the subtree to the main tree via an alternative core router.

The benefits of this multicasting method are that it utilises the underlying unicast routing protocol for the network, and only requires multicast state data to be maintained by the CBT routers for each tree to which they belong. Also, the exchange of "keep alive" messages is limited to only those routers that are in the multicast tree. Unlike DVMRP and MOSPF, CBT uses hard-state to maintain the multicast tree. This means that branches in the tree have to be explicitly removed when they are no longer required. The costs of using CBT are that

- All user data traffic has to pass through the core router for a multicast group, thus forming a potential bottleneck.

- As Wall [62] shows, the bound on the maximum delay of an optimal centre based tree is twice the shortest path delay. In other words, the delays experienced by CBT multicast deliveries could be up to two times that of an SPT based multicast.

- Because the multicast branch to a receiver is established by sending a join acknowledgement from the existing CBT back along the path from which the join request was received, CBT branches are reverse shortest paths. In networks with asymmetric links the CBT shared tree is therefore not a shortest path delivery tree. This characteristic may add additional delay to the already increased delay incurred by using a shared tree.

2.3.5 Protocol Independent Multicasting (PIM)

The Protocol Independent Multicast method was proposed to address the problems of multicasting to group members that might be sparsely distributed across wide area internets [19]. Like CBT, it was a design goal of PIM to use less resources than those used by source rooted multicasting protocols such as MOSPF and DVMRP. The PIM design also recognised that

- High data rate, low latency applications could only be served by source rooted multicasting (which CBT does not support).

- Low data rate, high latency applications with high numbers of sources would save network resources if supported by shared multicast trees.

- Traditional shared trees, such as CBT, may have a problem with multicast traffic concentration at their cores.

PIM supports two modes of multicasting. Dense mode supports applications that have either

- Low latency requirements or,
- Where the number of simultaneous senders is such that network performance would be unacceptably degraded by the concentration of traffic on a shared tree.

Sparse mode is designed to multicast to a group

- Whose membership size is considerably smaller than the number of routers in the network.
- Which may be widely distributed over a large area.
- For which high latency is acceptable.

A PIM multicast tree initially consists of a set of rendezvous points (RPs), the addresses of which are associated with a multicast group. How RP locations are decided has not been specified, but it will either be a configuration responsibility, or their addresses may be obtained by hosts and distributed to PIM routers in much the same manner as envisaged for core lists in CBT. A host will join a multicast group by sending an IGMP report message to its local PIM router containing the multicast group identity and RP address. The PIM router will then send a join request towards the RP indicating that it wants to receive user data for the multicast, via a shared tree. As the join message travels towards the RP, intermediate PIM routers establish a path from themselves back towards the host's PIM router, using a soft state mechanism, and forward the join request to the next PIM router on a path to the RP. This process is repeated at each intermediate PIM router until the RP is reached. At the RP the join request is dropped and a path from the RP to the receiving host has been established. The RP maintains the path to the receiving host by periodically sending it reachability messages down the established path. Sources, wishing to send to a particular multicast group, address the user data to their local PIM router. The local PIM router then sends a register message (and user data) to all the RPs for the multicast group. The

RPs reply to the source using join messages to set up user data delivery paths from the source to the RPs.

If a receiver requires source-specific tree delivery of user data, it first joins the shared tree for the multicast group. After receiving user data from the source the receiver's local PIM router can switch to the source rooted tree. The local PIM router recognises user data for the multicast group, from which it obtains the source address, and to which it sends a join request. Once the local PIM router starts receiving user data on the source rooted path, it sends a PIM prune towards the RP for the shared tree, indicating that it no longer wants to receive user data from the source via the shared tree.

PIM offers much the same benefits as CBT, with the addition of the option to use a source rooted multicast tree if required. PIM, however, maintains the multicast tree using soft-state. That is, PIM periodically sends refresh messages upstream towards each source to maintain the tree.

2.4 Internet Resource Reservation Protocols

2.4.1 Resource Reservation Protocol (RSVP)

All of the multicast routing protocols described above provide only "best-efforts" delivery of user data to receivers. With the advent of distributed applications such as multimedia conferencing, audio/video multicast delivery and distributed visualisation, "best-efforts" delivery is becoming untenable for some applications. Networks must be able to guarantee a requested quality of service for user data delivery, if the needs of these kinds of applications are to be met. To achieve this objective the Resource Reservation Protocol (RSVP) [69] has been proposed.

Strictly speaking, RSVP does not reserve any network resources for a communication. Rather it is a protocol used to establish router resource reservation state along a communication path between a source and receiver of a "flow" of user data. Sources can always send user data into the network without regard to the resources available for its delivery. This is the primary quality of the Internet Protocol. On the other hand, receivers know what they are capable of receiving and can therefore request the quality of service they want. In a multicast environment, different receivers may be either incapable of receiving, or not require, the same quality of service as other receivers. For these reasons RSVP is a receiver initiated protocol.

In order to reserve network resources, receivers need to know the path that user data takes from the source to reach them, and the transmission characteristics of the data flow that the source will send. The source, therefore, periodically sends path messages to the receivers that contain a specification of the data flow that the source will send into the network (see [47] and [9] for more detail). Path messages are carried towards the destinations by whichever routing protocol the network uses. They are not routed by RSVP. As the path messages pass through intermediate RSVP capable routers, they establish soft-state that describes the incoming and outgoing links for the multicast. A path message is then forwarded to the next-hop router for the multicast, and the process is repeated until the receiver's router is reached. Intermediate routers along the path do not reserve any resources at this stage, they just establish path state between the source and receiver(s). On the basis of the flow specification contained in the path message and the resources available to a receiver, it replies to a path message by sending a reservation message back along the route the path message arrived on. This reservation message contains the description of the resources the receiver wants reserved along the path. As the reserve messages pass through routers on the return path resources are either reserved for the user data flow, or the reservation is rejected. The reserve messages establish the resource reservation state along the path established by the path messages.

RSVP is much more complex than described here. The protocol supports different styles of reservation requests and the merging of requests where paths for the same group meet. The use of soft-state to maintain reservations allows paths to be re-routed to adapt to changes in the topology of the network. The reader is refered to [69] for a detailed description of the protocol.

In addition to a resource reservation mechanism (for example, see [9],[20] and [47]) the protocol requires admission control to manage network resources and routing protocols that will select data paths on the basis of requested quality of service (for example, see [32] and [35]). None of the routing protocols decribed above use admission control or quality of service path selection.

2.4.2 Internet Stream Protocol, Version 2 plus (ST2+)

The Internet Stream Protocol, version 2+, is an experimental protocol that enables applications to establish end-to-end paths, with specifed reserved resources, for real-time data flows (streams) between a source and one or more destinations, across an internet [54]. The resource reservations are for single direction data flows, between a sender and any number of destinations, i.e., multicast.

Like IP, ST2+ is a network layer protocol and is independent of underlying subnetworks. It uses the same addressing schemes as IP to identify hosts and may coexist with IP in network routers. ST2+ has the facility to encapsulate its messages within IP packets so that they can be transported transparently through IP routers that do not support the protocol.

Unlike IP and the multicasting protocols DVMRP and MOSPF, ST2+ is explicitly connection oriented. User data flows cannot be transmitted between a sender and receiver until a path, with the specified resources allocated, has been explicitly set up between them. The protocol has two components

- Path management.
- Real-time data transfer.

ST2+ path management is responsible for setting up, modifying and tearing down the paths used for transmitting data flows. To do this it uses two distinct services

- Routing: to select paths from the source to the destination(s).
- Resource Management: to reserve the appropriate resources for the data flow(s).

The ST2+ protocol specification does not define either of these services since they are considered to be external to the protocol, but it does assume their existence. ST2+ also makes assumptions about how these services are provided. The ST2+ setup protocol assumes that the external routing method it uses calculates unicast routes, on a hop-by-hop basis, as paths are constructed. How the routing method calculates which router is the next hop along any path is not defined by ST2+, but it could be by either a simple shortest path next-hop selection process, or by a more complex method based on network resources and a data flow's quality of service requirements, such as proposed by

Zheng and Crowcroft [71]. The calculation of complete data flow paths by the sender's router (source routing), which was supported by earlier versions of the protocol, has been removed from ST2+. Resource management is invoked at each router along a data flow path, as it is constructed, to allocate local resources to the flow. The local resource manager is supplied with a specification which describes the resources required by the data flow. This it checks against the resources available (for example, buffer space, bandwidth) and rejects the request if enough resources are not available, or accepts it otherwise. Once a path is established, ST2+ path management has mechanisms at each router to efficiently switch data flow packets to the next router along a path and to monitor the status of routes. Data flow paths persist for either the lifetime of the data flow or until a transmission failure occurs.

ST2+ supports construction of multicast trees by senders, receivers or both.

- For sender construction of data flow paths (or multicast), the source knows the addresses of all the receivers. This information is sent to each ST2+ router along paths, as they are constructed. Hence all ST2+ routers know which receivers they have downstream. Sender initiated multicast can be considered heavy weight because of the potentially large destination lists used during path set up, and because this information is maintained by ST2+ routers within a multicast tree.

- For receiver initiated path construction, the sender sets up a data flow with no receivers. That is, the ST2+ router local to the data flow source creates entries in its database for the data flow, but does not setup any paths. For a destination to initiate receipt of a data flow, it must obtain the identity of the data flow (stream ID) and the IP address of the source. How this information is obtained is not within the scope of the ST2+ protocol specification, but it might be via a directory service, for example. With this information the ST2+ router local to a destination sends a join request towards the data flow source. The join message traverses ST2+ routers on a path back towards the source, until it reaches one that is receiving the specified data flow. This ST2+ router can then set up a path between itself and the destination on which to send the data flow. Receiver initiated multicasts are lightweight in comparison to sender initiated ones. Less state data is held at each ST2+ router, since not all routers will necessarily know all of the downstream destinations they are forwarding data flows to.

- Sender and receiver construction of multicast trees is achieved through a combination of both the sender method and the receiver join method.

The ST2+ protocol is far more complex than the overview given above. The protocol has to deal with a variety of problems that may occur during path set up, such as admission failure. It also has mechanisms to deal with failures in the network. The protocol supports groups of streams (data flows) and has the facility for an application to request modification to the resource requirements of a data flow. The characteristics of the ST2+ protocol that are important to our work are

- Multicast trees can be constructed "en masse" by a single protocol message being sent into the network from the data flow source.
- Multicast trees can be built in a piecemeal fashion by receivers joining a data flow arbitrarily.
- Multicast trees are held in place by hard state, that is, they are connection oriented. Path construction is, therefore, a relatively expensive and time consuming process.

The reader is referred to [54] for a detailed description of the protocol.

2.4.3 Tag or Label Switching Multicast

It is recognised that the growing demand for increasing bandwidth in the Internet can, in some part, be achieved by improving the forwarding performance of routers. In IP networks, each router uses a network layer routing calculation and data from a packet's header to determine the path on which to forward the packet. The packet header contains far more data than required to calculate the next hop towards its desination, particularly where a stream or flow of data is being transmitted. Working groups within the Internet community are developing label swapping (or tag switching) architectures ([49] and [50]) to reduce the complexity of the current next hop IP routing method and, thus, improve the forwarding efficiency of network routers.

In essence, label or tag switching is quite similar to cell switching in ATM networks in that each switch (or router, in this case) maintains a table of incoming labels (or

VPI/VCIs, in ATM) which are mapped to outgoing labels. Both architectures consist of two components

- Control: the mechanism used to bind network layer routes to tags. For example, in tag switching [49], a switch uses the routing method to identify the next hop router for a packet and requests a tag binding, from the next hop router, for the path. In label swapping, [50], upstream and downstream routers agree bindings between labels and streams sent between them.

- Forwarding: the mechanism used to switch tags or swap labels and hence forward packets. A packet destination address is mapped to a tag or label as the packet enters the network. From then on, as the packet passes through the network, each router recognises the tag or label, on an incoming packet, which it maps with the corresponding entry in its tag or label database. From this mapping the tag or label and forwarding information to the next router is obtained. In the case of multicast, an incoming tag may map to several outgoing tags.

These architectures also support explicit routing, where the entire route is chosen by a single router. The reader is refered to [49] and [50] as a starting point for further explanations of how these architectures might work, and for references to other work in this area. One observation made in [49] that is worth noting is the applicability of these architectures to ATM networks by the implementation of the Tag Switching Control component.

Like ST2+, neither architecture specifies how routes are calculated (for the binding mechanism), or what routing information needs to be exchanged between routers in order to perform route calculations.

2.5 B-ISDN/ATM Routing Protocols

Asynchronous Transfer Mode (ATM) networks will offer significant improvements over existing local area and wide area network technologies [2]. In particluar, they will support the integration of a wide variety of applications that use both voice, moving image video and other high bandwidth communications. The primary features of ATM networks are that they will offer users a guaranteed quality of service based on a collection of additive and non-additive route metrics (for example, bandwidth and delay) while remaining scalable over wide areas. For these reasons, among others, ATM will be the underlying network technology for Broadband Integrated Services Digital Networks (B-ISDN)

Unlike IP networks, ATM is connection oriented and so requires both a signalling system and routing methods to establish user calls. To support the features of ATM networks, the signalling and routing will be far more complex than those currently used to support IP networks.

2.5.1 Private Network/Network Interface (PNNI)

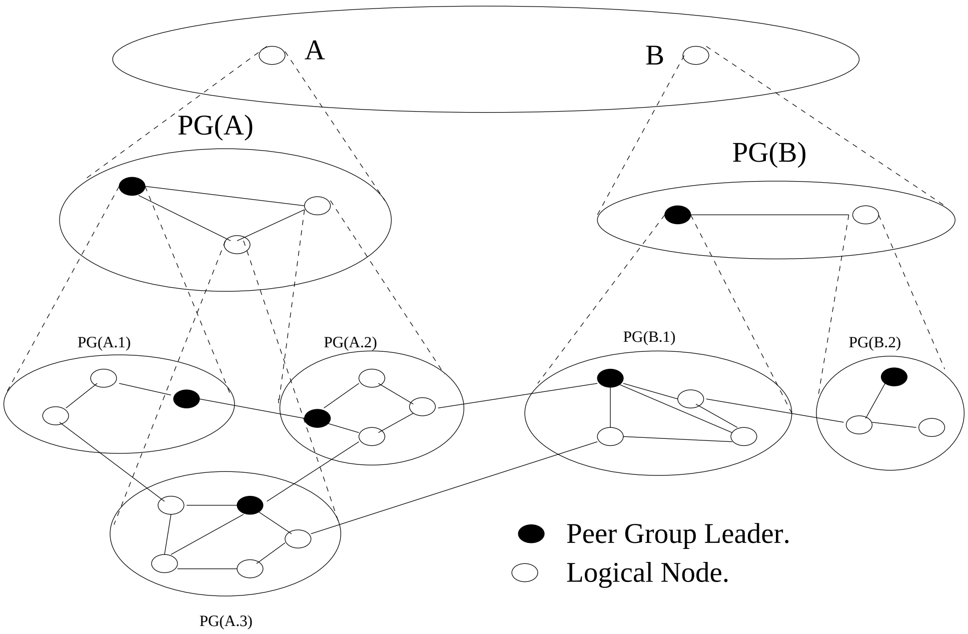

Figure 3: A PNNI hierarchy

The PNNI protocol [60] has been specified by the ATM Forum for use between private ATM switches and groups of private ATM switches. The protocol has two parts; signalling and routing.

PNNI signalling is based on ATM UNI signalling (see, [58] and [59]) and is used to establish both unicast and multicast connections across an ATM network. Version 1.0 of PNNI does not support all the services defined for ATM UNI version 4.0 [59]. In particular, it does not support leaf initiated joins to multicast groups. Multicast destinations can only be added to a multicast group at the explicit request of the sender.

The PNNI routing protocol calculates the paths connections take across a network, and also maintains the data necessary to perform these calculations. Both of these tasks have an influence on the algorithms that can be used by PNNI to calculate multicast delivery trees.

PNNI routing specifies a hierarchical addressing scheme for routing. Within a PNNI switching system, starting from the lowest level (i.e., the switch level), logical nodes are organised as a hierarchy of peer groups. There may be one or more peer groups at each level of the hierarchy. Within each peer group, one logical node is designated the peer group leader. Peer groups may then become logical nodes within the next higher level peer group of the hierarchy. A higher level peer group may also be a logical node of yet a higher level peer group. This structure, which may contain as many levels as deemed necessary for any particular switching system, is illustrated in Figure 3.

Each logical node has an identity, which contains the nodes peer group identity. The peer group identity indicates the level of the peer group in the hierarchy and its parentage. Parent peer identities are always shorter than those of their children and the child peer group identity always contains the peer group identity of its parent.

PNNI is a link state routing protocol, where the link state data consists of topology state data and address reachability state data. Logical nodes exchange this data so that they can build databases of the network topology and tables of address reachability. Logical nodes only exchange link state data within their own peer group.

Peer group leaders aggregate and summarise the topology and reachability information of their peer group into a complex node description and flood it as link state data within the next higher layer peer group. Peer group leaders also pass link state data they recieve at a parent level down to their child level and flood it to the other child logical nodes in their peer group. By this means destination reachability data and (a virtual) network topology permeates the entire network without the necessity of all link state data being flooded to all nodes in the network.

To set up a user call in an ATM network, PNNI has to perform two tasks

- The selection of a call path.
- The establishment of call state along the selected path.

ATM users are able to specify a Quality of Service (QoS) and bandwidth requirement for each call they request. For this reason, path selection in PNNI is based on both the user's requirements and the resources available within the network. Paths are calculated by the connection source node and established across the network using designated transit lists (DTLs). A DTL is essentially a stack of next node identities used by each node along the path to find out where the next forward node is. DTLs are removed from the stack and replaced by other DTLs as the path moves across peer groups. So, although the source node calculates the entire path, parts of the path are virtual in that they traverse complex nodes in higher level peer groups. The sections of the path that cross higher level peer groups have to be mapped onto the underlying lowest level peer groups (at switch level) as peer group boundaries are crossed. Using call admission control, the process of establishing call state along a selected path confirms whether or not the requested resources are available. If during this second step of the call set-up process, the required resources are not actually available along part of the selected path, the route is unwound to a point at which a new path can be calculated. To minimise the likelihood of this happening, the path selection procedure may use a generic call admission control procedure to predict which links are likely to have sufficient resources available, and to use only these links in the path. The PNNI specification does not mandate any particular algorithm for path selection, although it provides an example of an acceptable one (see Appendix H of the PNNI specification, [60]).

User bandwidth and QoS requirements are specified using the UNI SETUP message ATM traffic descriptor and QoS parameter (see [58] and [59]). The traffic parameters specify Peak, Sustainable and Burst Cell rates for the communication, while the QoS parameters specify which class of service is required. For each QoS class of service values can be specified for a variety of performance parameters, such as Cell Transfer Delay and Variation.

UNI/PNNI specify that multicast connections are set up by establishing an initial unicast path between the source node and a destination node. Further destination nodes are then included in the multicast delivery tree by means of "add party" requests. The source node can either wait for each add party request to be acknowledged (serial join) or it may have multiple requests outstanding at the same time (parallel join).

2.5.2 Broadband Integrated Services Digital Network (B-ISDN)

The origins of Integrated Service Digital Networks (ISDNs) are in the services provided by telephone companies, who recognised the need to integrate their separate voice, data and dedicated network services into a single network. The Broadband ISDN (B-ISDN) recommendations of the International Telecommunications Union (ITU) are being developed to further this aim by supporting a wider range of audio, video and data transfer services within the same network [8].

The original, or Narrowband, ISDN provides the subscriber with a "digital pipe" into an ISDN switch. Connections between ISDN switches are made using either packet switching (data), circuit switching (voice) or non-switched (dedicated) capablities. The B-ISDN will maintain the concept of a "digital pipe", but will integrate the packet switching, circuit switching and dedicated capabilities of N-ISDN into one broadband network, B-ISDN. The transfer mode to be used for the B-ISDN is the Asynchronous Transfer Mode (ATM). The B-ISDN architecture will be described in functional terms and so is implementation independent [8].

Telephone networks, from which ISDN has evolved, in general use one of three routing architectures

- Direct routing: where routes are fixed and pre-established.
- Alternate hierarchical routing: where routes are organised into hierarchies, from local offices, through toll centres and so on, up to regional centres. Trunk routes are provided, beyond the tree structure, as alternative routes to be used when network loading dictates.
- Dynamic two hop alternate routing: where routes are calculated dynamically through more complex network architectures, at call set-up time. Routes are selected on the basis of network loading.

In alternate hierarchical routing, a path between two subscribers follows the lowest level of connectivity in the hierarchy that has the necessary resources available for the call. As resources are consumed on trunks at lower levels, the path is selected from trunks higher up in the network hierarchy. In dynamic two hop alternate routing, networks have a logical link between each pair of switches, and all switches are equally responsible for routing calls. If a call cannot be established along a direct link between a pair of switches, then an alternative, two link route is used, if one is available. Otherwise the call is blocked. Examples of networks that use dynamic two hop alternate routing include AT&T's DNHR scheme and DAR, which is planned for BT's domestic network. Each of these systems uses a different mechanism for the selection of alternative routes [51]. Recent work has addressed mechanisms for dynamic two hop alternate routing in ATM networks [7].

The implementation of such routing schemes does not preclude the use of arbitrarilly connected, multi-hop, dynamically routed, networks as the transfer scheme for B-ISDN. The ITU recommendations for B-ISDN do not specify, nor imply, the network architecture or how a provider of B-ISDN services should route calls through a network, although as Stallings points out in [55], ISDN is evolving from the circuit switching technology of the telephone networks to the packet (or cell) switching technology of broadband networks (such as ATM), as it takes on broadband services. How architectures for B-ISDN/ATM networks evolve compared to ATM networks in the Internet remains to be seen.

Multicast connections in B-ISDN are set up in a manner similar to that of PNNI. A path is first established between a sender and one receiver whilst indicating that the connection is to be point-to-multipoint. Once this connection set up has become alerting or active, additional receivers can be joined to the connection using "add party" requests. Multiple receivers can be pending at any one time. That is, the sender does not need to wait for the response to any other add party request before issuing another [34]. B-ISDN does not currently support receiver initiated joins.

Like PNNI, the sender's ATM traffic descriptor, broadband bearer capability and QoS parameters are specified using the User/Network Interface SETUP message [33].

There are significant differences between the way B-ISDN and PNNI use their underlying ATM networks. For example, in PNNI networks the user data path is the same as the call control path, which is not the case for B-ISDN. The reader is referred to both the ATM Forum and ITU Recommendations for detailed descriptions of these methods (which are beyond the scope of this thesis).

Chapter 3

Low Cost Quality of Service Multicasting

3.1 The Bounded Delay, Minimum Cost Multicast Routing Problem

The bounded delay minimum cost multicast routing problem can be stated as follows.

- Given an undirected connected graph $G = \langle V, E \rangle$ where V is the set of its vertices and E the set of its edges, and the two functions: cost $c(i,j)$ of using edge $(i,j) \in E$ and delay $d(i,j)$ along edge $(i,j) \in E$.

- Find the tree $T = \langle V_T, E_T \rangle$, where $T \subseteq G$, joining the vertices s and $\{M_k\}_{k=1}^{n} \in V$ such that $\sum_{(i,j) \in E_T} c(i,j)$ is minimised and $\{D(s, M_k) \leq \Delta; k = 1,..,n\}$ where Δ is the delay bound and $D(s, M_k) = \sum_{(i,j)} d(i,j)$ for all (i,j) on the path from s to M_k in T.

Note that, if the delay is unimportant, the problem reduces to the Steiner tree problem. The addition of the finite delay bound makes the problem harder; it is still NP-complete as any potential Steiner solution can be checked in polynomial time to see if it meets the delay bound.

3.2 Heuristics with an arbitrary delay bound

Several heuristics have been proposed that use arbitrary delay bounds to constrain multicast trees. Kompella, Pasquale, and Polyzos [38] propose a Constrained Steiner Tree (CST_c) heuristic which uses a constrained application of Floyd's algorithm. Widyono [68] proposed four heuristics based on a constrained application of the Bellman-Ford algorithm. Zhu, Parsa and Garcia-Luna-Aceves [72] based their technique on a feasible search optimisation method to find the lowest cost tree in the set of all delay bound Steiner trees for the multicast. Evaluation work carried out by Salama, Reeves, Vinitos and Sheu [52] indicate that Constrained Steiner Tree heuristics have good performance, but are inhibited by high time complexity.

The proposals for Constrained Shortest Path Trees by Sun and Langendoerfer [56], which we abbrieviate as CSPT and by Waters [65], which we abbreiviate as CCET (Constrained Cheapest Edge Tree), generally have a lower time complexity than Constrained Steiner Trees but their solutions are not as efficient. In this evaluation work we

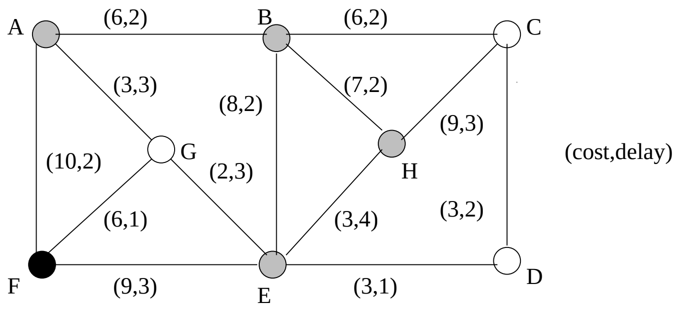

Figure 4: The example network

compare our heuristics against the solutions generated by the Sun and Langendoerfer and the Kompella et al heuristics, both of which we describe below. For consistency the worked examples (see Figures 5, 11 and 14) use the network illustrated in Figure 4. The examples of pathological behaviour, such as where a heuristics sometimes cost more than a shortest path tree, are based on evidence extracted from the evaluation results for each heuristic. The arbitrary delay bound is set to 8 for the Kompella et al and Sun and Langendoerfer heuristics because they both seek solutions with a delay less than Δ. For the Waters heuristic Δ is set to 7 since it seeks a solution where the delay is less than or equal to Δ. It should be noted that the solutions produced by each heuristic in these examples do not illustrate their general performance.

3.3 The CCET and CCPT Heuristics extended

The CCET heuristic was first published in [63] along with some simple preliminary evaluations. In [65] important variations of the heuristic were introduced and comprehensively evaluated.

The original heuristic and its variant, CCPT [11], were bound by either the broadcast delay or the multicast delay. Here we extend the heuristics such that they are bound by an arbitrary delay, Δ. The effect this has upon the heuristic is to vary the size of the search space for the multicast tree in the second stage of the process (steps 4 and 5). The greater value Δ has, the larger the search space becomes. The extended procedure for the CCET heuristic is as follows

1. Use an extended form of Dijkstra's shortest path algorithm, to find for each $v \in V - \{s\}$ the minimum delay, dbv, from s to v. As the algorithm progresses keep a record of all the dbv found so far, and build a matrix $Delay$ such that $Delay(v, k_i)$ is the sum of the delays on edges in a path from s to k_i, whose final edge is (v, k_i), for each k that is adjacent to v.

2. The arbitrary delay bound is Δ. Set all elements in $Delay(v, k)$ that are greater than Δ to ∞. The matrix $Delay$ then represents the edges of a directed graph derived from G which contains all possible solutions to a multicast tree rooted at s which satisfy the delay constraint.

3. Now construct the multicast tree T. Start by setting $T = \langle\{s\}, \emptyset\rangle$.

4. Take $v \ni V_T$, with the largest dbv, that is less than Δ, and join this to T. Where there is a choice of paths which still offer a solution within the delay bound, choose at each stage the cheapest edge leading to a connection to the tree.

5. Include in E_T all the edges on the path (s, v) not already in E_T and include in V_T all the nodes on the path (s, v) not already in V_T.

6. Repeat steps 4 and 5 until $V_T = V$, when the broadcast tree will have been built.

7. Prune any unnecessary branches of the tree beyond the multicast recipients.

The extended procedure for the CCPT heuristic is similar to that for the CCET heuristic.

3.3.1 A Worked Example

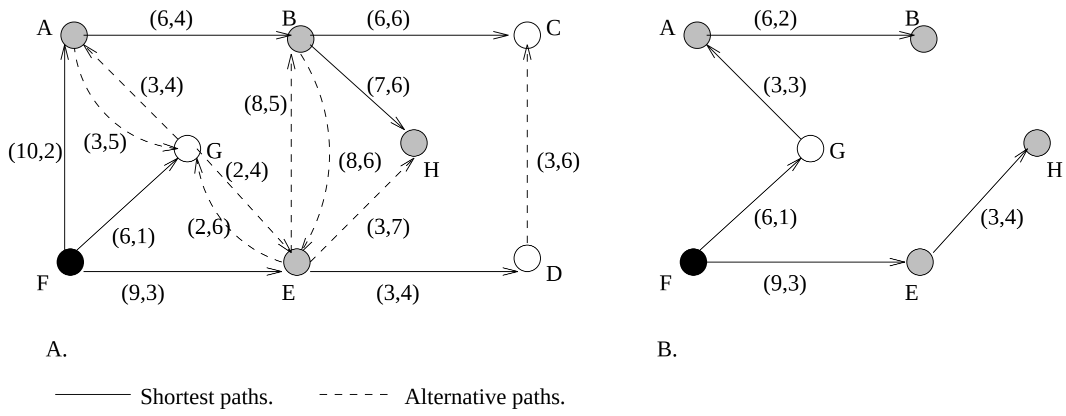

Figure 5: The CCET heuristic: a worked example

To illustrate the working of the heuristic we start with the graph shown in Figure 4. The bracketed parameters for each link indicate $(cost, delay)$. The example finds the multicast route from source F to destinations A, B, E and H.

The application of the extended form of Dijkstra's algorithm pruned to the arbitrary delay bound Δ results in the directed graph shown in Figure 5A where the parameters shown against each link represent the edge cost and total delay from the source F to reach the node at the end of that link. The multicast tree is then constructed starting with $T = \langle F, \emptyset \rangle$. First H is connected to F using the path HE, EF. Node C is connected via the path CD, DE and then node B is connected via path BA, AG, GF. Finally, the edges CD and DE are pruned to give the multicast tree in Figure 5B with a cost of 27 units and a final delay bound of 7.

A shortest delay path tree, when pruned to the multicast, has a cost of 32 units, but a delay bound of only 6 (see Figure 18A). On the other hand a minimum cost spanning tree, when pruned to the multicast, gives a cost of 20 units but a delay bound of 8 (see Figure 18B).

3.3.2 Time Complexity of the CCET Heuristic

The first stage, determining the directed graph, has the same time complexity as Dijkstra's algorithm, $O(n^2)$. The vertices can be put in delay bound order during the construction of the directed graph.

In the second stage, building the multicast tree, requires a depth first search from each leaf node to find a path to the source. As the multicast tree grows the search space for each leaf to source node path becomes smaller. The time complexity of the depth first search is $O(max(N, |E|))$ [26] where N is the number of nodes, and E is the set of edges, in the leaf node to source tree. The values of N and $|E|$ depend on the topology of the network, the position of the multicast source node and the arbitrary delay bound. As the network edge density or the arbitrary delay bound increases so do the values of N and $|E|$. In practice, an optimal upper bound can be placed on the arbitrary delay to limit the values of N and $|E|$.

3.3.3 Pathological Behaviour of the CCET Heuristic

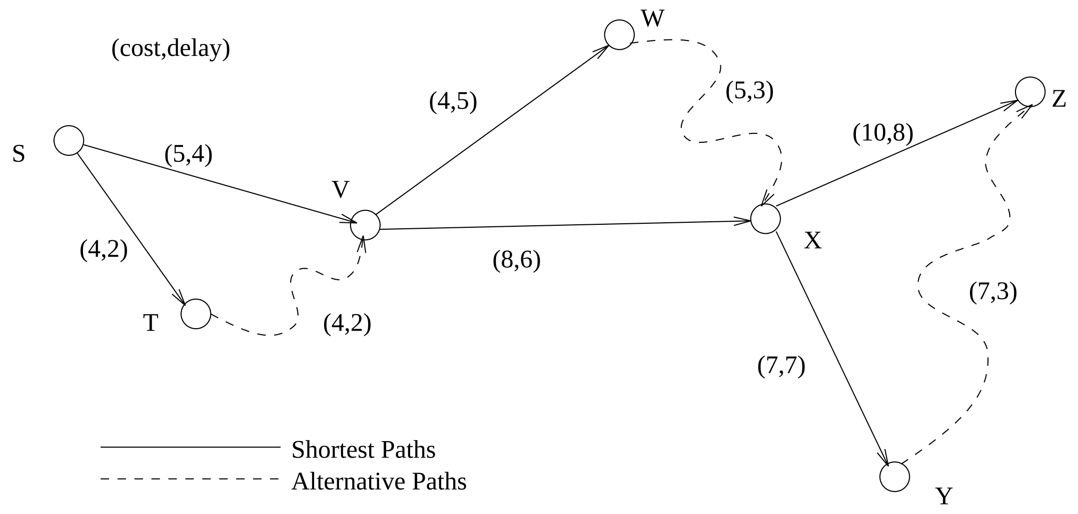

Figure 6: Example of a Rogue Path

The heuristic's first stage constructs a directed graph of paths between the multicast source node and every other node that can be reached within the delay bound. The number of paths between any node and the source offered by this graph depends on the delay bound and the graph density. The higher either of these values are, the more

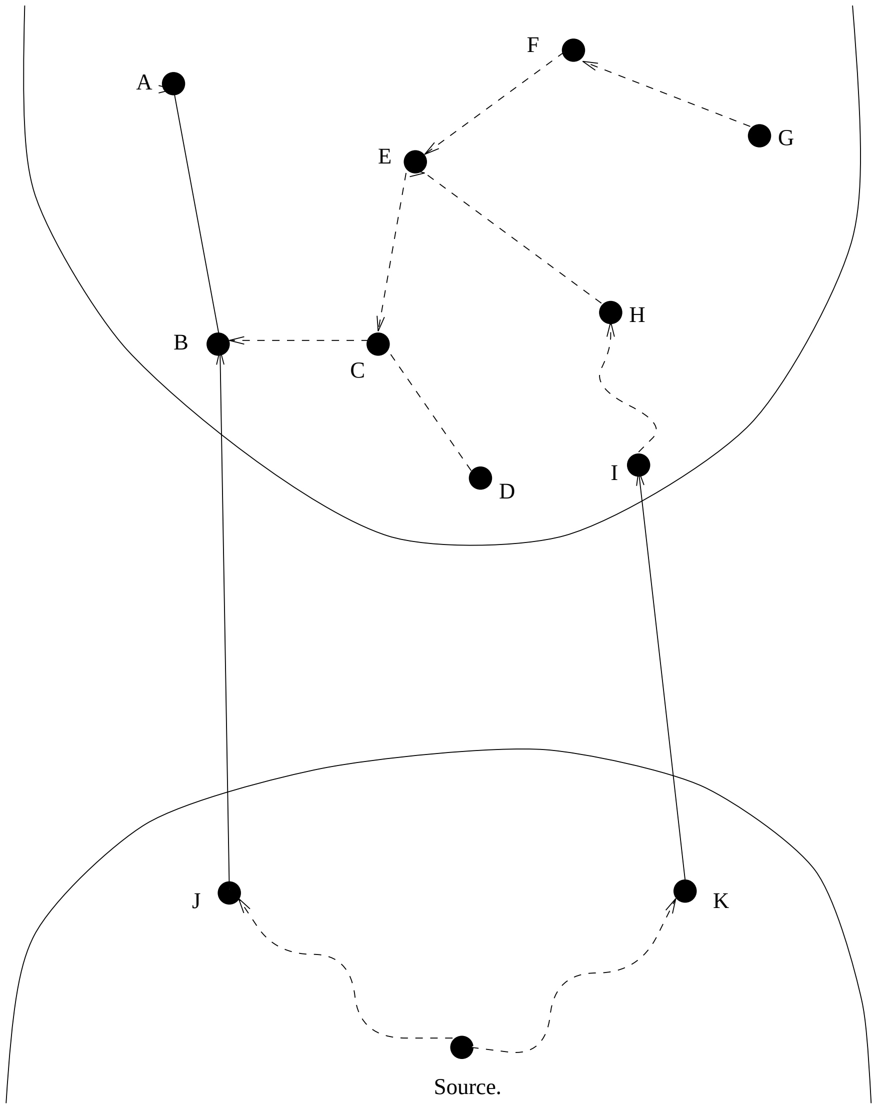

Figure 7: Rogue Paths and the Pathological Walkabout

paths that are available. This graph also contains rogue paths that exceed the delay bound because they include a high proportion of alternative edges. The "cheapest" path in Figure 6 between node Z and the source includes three alternative edges ZY, XW and VT with a delay of 22. If the arbitrary delay bound placed on this multicast were 21 the "cheapest" path is a rogue and would not be detected until the last link, TS, was added to the path.

In the second stage the heuristics extract the bounded delay minimal cost tree from the bounded directed graph constructed during the first stage. To do this the heuristics start by constructing a path between the node furthest from the source and the source node; that is within the delay bound. If the arbitrary delay bound is the broadcast bound or less the path between the first node selected and the source will be a minimum delay path, irrespective of cost. If the delay bound is greater than the broadcast delay then this path is not necessarily a minimum delay path. This first path becomes the trunk of the bounded delay minimal cost tree. The heuristics then add the return paths from each node successively closer to the source until the tree is complete. Two characteristics affect how paths join the existing tree

- Nodes closer to the source have a greater choice of paths back to the source because of the slack between their shortest path delay to the source and the delay bound on the multicast.

- As the tree grows the probability of a path joining the tree at a node closer to itself than the source increases.

The combination of these characteristics generally minimise the probability of loops occuring during tree construction. When the tree is young and sparse, branches are likely to be close to their shortest paths to the source. As the tree grows, branches are more likely to meet the existing tree sooner.

In multi-cluster networks the furthest node from the source and the source node may be in different clusters. If the network comprises three or more clusters then some clusters may not be on the trunk path from the furthest node back to the source. Under these conditions the two characteristics described do not apply to non-trunk clusters. The available delay slack within a cluster may be high and the existing tree may be remote from the node wishing to attach to it. If the cluster is very dense and has few links on a path to the source the second stage may examine a large number of rogue paths that ultimately break the delay bound before finding the path back to the source. Without low delay slack or a close-by existing tree the search for a path to the source may take a very long time. How this problem can occur is illustrated in Figure 7. When constructing the return path from node A to the Source edge BC is chosen instead of BJ because it is the cheapest exit. The heuristic then starts to search the tree rooted at node C for a path back to the source. It is not until this search has been exhausted that the edge BJ is examined.

This behaviour may also apply to single cluster networks where the arbitrary delay bound is very much larger than the network diameter and few edges are removed in the first stage of the heuristic.

3.3.4 When CCET costs increase

The CCET heuristic selects return paths on the basis of the "cheapest" exits from each node, back towards the source, that do not violate the arbitrary delay bound. In some networks this rule can cause multicast trees found by the heuristic to be more expensive than might otherwise be expected.

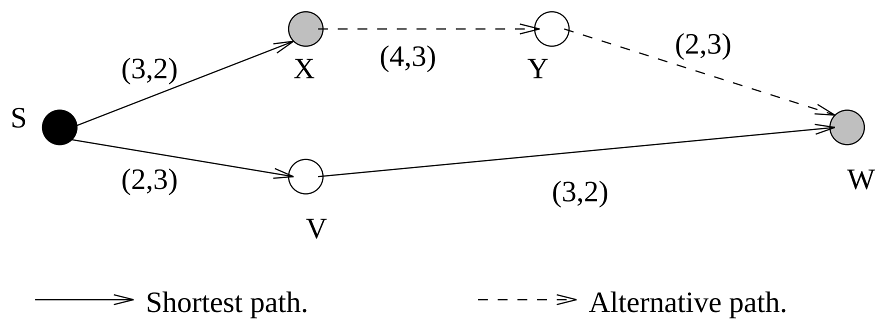

Figure 8: CCET costs increase as Δ is relaxed.

The cost of multicast trees found using the CCET heuristics can increase when the arbitrary delay bound is relaxed. Such a case is illustrated in Figure 8. With a delay bound of $\Delta = 5$ the multicast tree will include the edges SX,SV and VW at a cost of 8 units. This happens because the edge YW will have been excluded as it gives node W a delay of 8 units from the multicast source node, S. If Δ is increased to 8 the edge YW is included and will be selected as the "cheapest" return route from node W towards S.

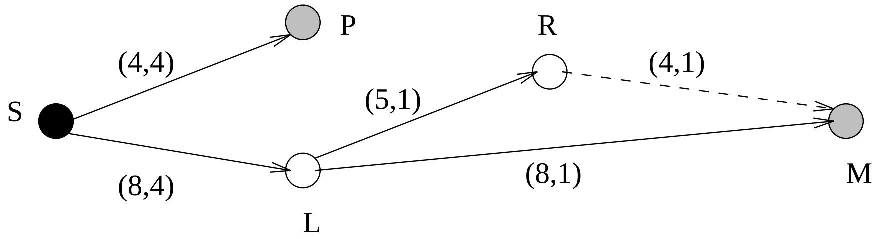

Figure 9: CCET more expensive than SPT.

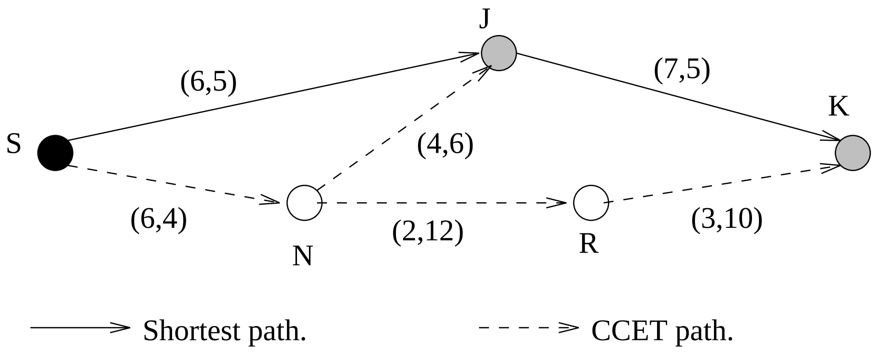

Figure 10: Adding a single node to the CCET tree

The multicast tree then becomes SX, XY, and YW at a cost of 9 units.

The cost of solutions found using Dijkstra's shortest path algorithm can sometimes be cheaper than those found using the Water's heuristic. The multicast tree found using Dijkstra's algorithm for the network in Figure 9 includes the edges SP, SL and LM, the shortest paths. The cost of this tree is 20 units. If the CCET heuristic is used to calculate the multicast tree with an arbitrary delay bound of $\Delta = 6$ the solution will include edges SP, SL, LR and RM because RM offers the "cheapest" exit back to the source from node M. The cost of this tree is 21 units.

As CCET multicast trees grow their cost difference from the corresponding SPT solution will fluctuate. The addition of a single node to the multicast can cause the CCET solution to change from being cheaper than the SPT to becoming more expensive. Figure 10 illustrates the CCET and SPT solutions for a multicast. The multicast source is node S and the delay bound, Δ, is greater than 26. In the first instance the multicast includes only node K. The SPT solution will choose the route SJ, JK to reach K at a cost of 13 units. CCET will choose the route SN, NR, RK to reach K at a cost of 11 units. If the multicast grows by the addition of node J the cost of the SPT solution will remain the same since J is already on the path to node K. The CCET solution has to add the link NJ, increasing the tree cost to 15, to reach node J.

3.3.5 Multicast Tree Stability and Dynamic Groups

The broadcast tree constructed by the CCET heuristic will be the same for all multicast groups with the same multicast source and arbitrary delay bound. This occurs because the heuristic constructs the broadcast tree using only the multicast source and the arbitrary delay bound. The multicast tree is extracted from the broadcast tree by removing unwanted branches. This means that in a dynamic environment where the multicast trees grows and dies, the broadcast tree only needs to be recalculated if the topology of the underlying network changes.

3.3.6 Behaviour of the CCPT Heuristic

The CCPT heuristic has similar behavioural characteristics to the CCET heuristic. However, because CCPT uses cheapest paths, instead of cheapest edges, to find routes back to the multicast source it has a smaller search space in which to find solutions. For this reason, CCPT finds less efficient solutions than CCET, but it is also less likely to encounter the pathological problem of CCET.

We have not analysed, in detail, the behaviour of the CCPT heuristic because of its poor performance relative to the CSPT heuristic.

3.4 The CST_c Heuristic

The algorithm has three main stages.

1. A closure graph (complete graph) of the constrained cheapest paths between all pairs of members of the multicast group is found. The method to do this involves stepping through all the values of delay from 1 to Δ (assuming Δ takes an integer value) and, for each of these values, using a similar technique to Floyd's all-pairs shortest path algorithm (see [25]) to find the cheapest path.

2. A constrained spanning tree of the closure graph is found using a greedy algorithm. Two alternative selection mechanisms are proposed, one based solely on cost, the

other on cost and delay. In our evaluation we use the most efficient of these (cost only) which selects edges for the spanning tree using the function

$$f_C = \begin{cases} C(v,w) & \text{if } P(v) + D(v,w) < \Delta \\ \infty & \text{otherwise} \end{cases}$$

where $C(v,w)$ is the cost of a constrained path from node v to node w, $P(v)$ is the delay from the multicast source to node v and $D(v,w)$ is the delay on the path (v,w).

3. The edges of the spanning tree are then mapped back onto their paths in the original graph. Finally any loops are removed by using a shortest paths algorithm on the expanded constrained spanning tree [37].

3.4.1 A Worked Example

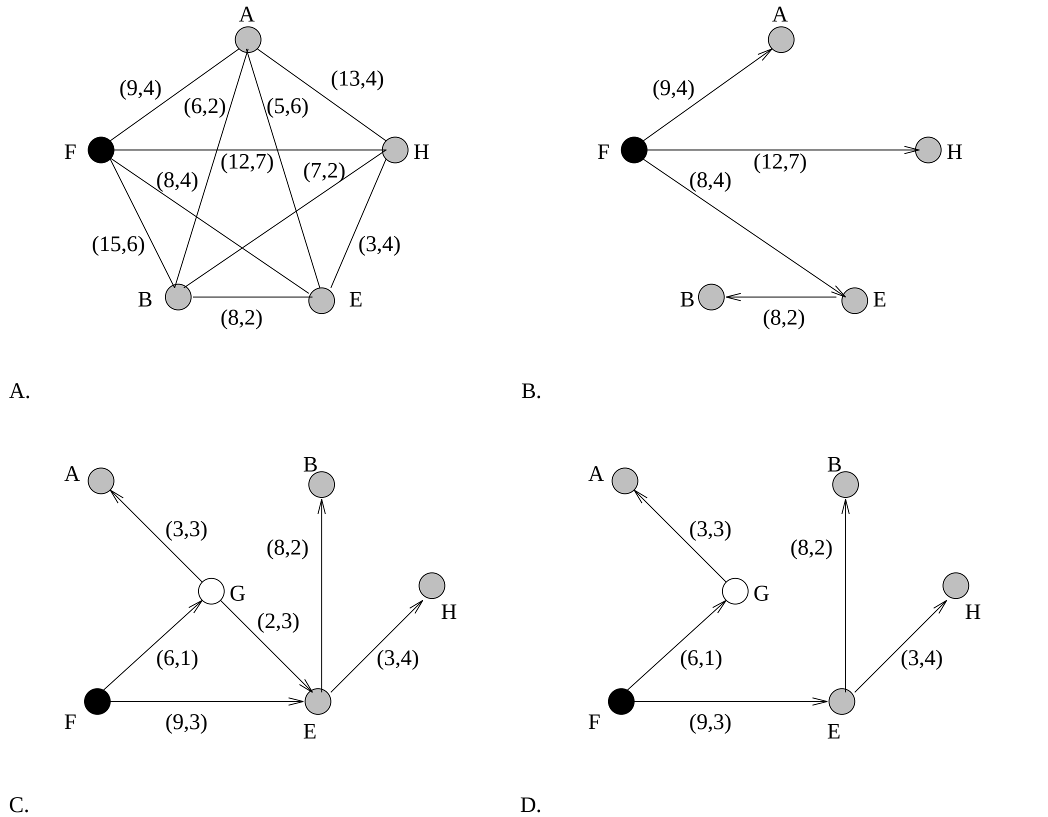

Figure 11: The CST_c heuristic: a worked example

Applying the first stage of the heuristic to the network in Figure 4 produces the constrained closure graph illustrated in Figure 11A. Note that this graph need not be a complete graph so long as there are paths between every multicast node and the source. Path AF includes node G, path HF includes E and path EF includes G. There is a conflict between the paths HF and EF which will result in a loop occurring in the constrained spanning tree which must be removed when the path is mapped back onto the original network. The other paths have no intermediate nodes.

Figure 11B shows the spanning tree obtained from the closure graph using the edge selection function f_C. Expansion of the spanning tree into their original paths results in a graph with a loop (Figure 11C.) which when removed produces the solution in Figure 11D. This tree has a cost of 29 units and a delay of 7.

3.4.2 Time Complexity of the CST_c Heuristic

The calculation of the constrained shortest paths during the first stage of the heuristic is the most time consuming, with a complexity of $O(\Delta n^3)$, where n is the number of vertices in the graph [25]. The second stage has a time complexity of $O(m^3)$ where m is the number of nodes in the multicast group. Mapping the closure graph back onto the original graph has a time complexity of $O(mn)$. Loop removal using Dijkstra's algorithm has a time complexity of at most $O(n^2)$. This gives the algorithm an overall time complexity of $O(\Delta n^3)$. If the arbitrary delay bound Δ increases, the computation time of the heuristic increases. This characteristic can be overcome by decreasing the granularity of Δ through scaling, although this will compromise the accuracy of the results [68].

3.4.3 When CST_c costs more than SPT

In most cases CST_c calculates multicast solutions that are cheaper than those produced by SPT, but it does sometimes generate more expensive solutions. Figure 12 illustrates such a case. The multicast is from the source node, S, to the destination nodes, M and P. The arbitrary delay bound is 12. The first stage of CST_c constructs a closure graph from the cheapest contrained paths between the multicast nodes and the source in the underlying graph. From the closure graph CST_c selects the solution. In the example the multicast solution selected will be the closure graph edges SM and

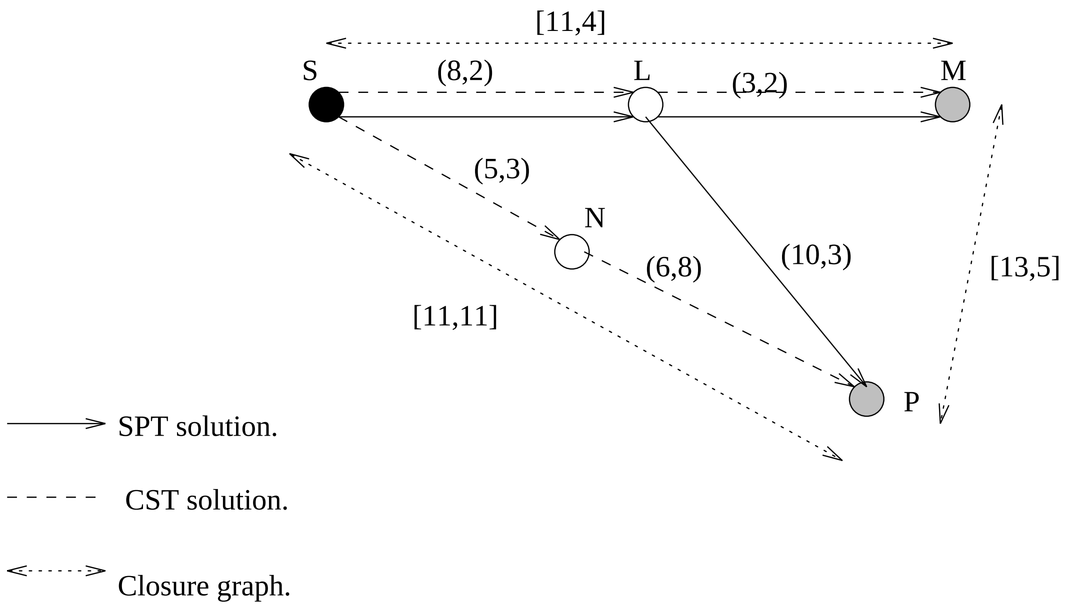

Figure 12: CST_c more expensive than SPT

SP at a cost of 22 and a delay of 11. The final stage of CST_c maps the closure graph solution back onto the original graph, providing the solution SL, LM and SN, NP. The SPT algorithm will select paths solely on the basis of the delay from the source to each node. The solution SPT provides is SL, LM and LP at a cost of 21 and delay 5. By chance the SPT has been able to take advantage of the common edge SL, which was not available in the closure graph for CST_c.

3.4.4 Bounds on Tree Cost

If the arbitrary delay bound applied to the heuristic is effectively infinite then the solutions produced will be similar to those calculated using the Minimum Spanning Tree (MST) for the Steiner Tree Problem [27].

3.4.5 Multicast Tree Stability and Dynamic Groups

The topology of a CST_c multicast tree may be reconfigured as the tree grows or shinks. The second stage of the algorithm applies a greedy process to extract the solution from a closure graph that comprises only the multicast nodes. If a node is added or removed from the multicast the closure graph changes, and so the greedy process has a different set of nodes to consider. This may result in a multicast solution

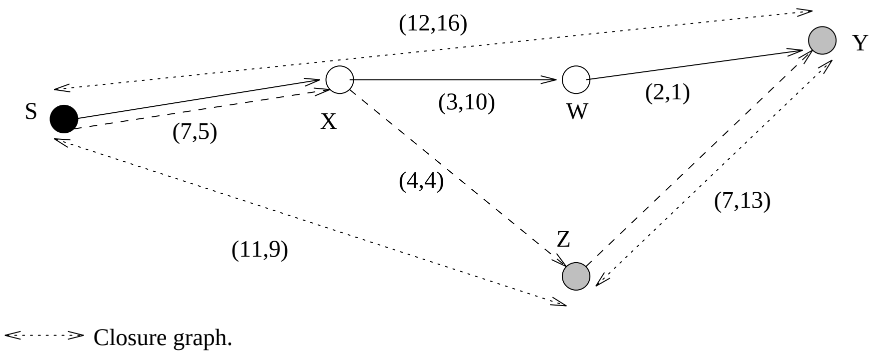

Figure 13: Changing topology of CST_c multicasts

with a topology different from its predecessor. Figure 13 illustrates an example of such a change. Initially the multicast is between the source node, S, and a single multicast node, Y. The closure graph constructed by CST_c will be the single virtual edge SY. When this edge is mapped back onto the original graph we get the solution SX, XW, WY for the multicast. The node Z is then added to the multicast, giving the closure graph of virtual edges SY, YZ, SZ. From this graph the greedy process selects a multicast tree SZ, ZY. When this solution is mapped back onto the original graph the multicast solution becomes SX, XZ, ZY. The addition of a single node to the multicast has caused edges XW and WY to be removed from the solution and replaced by edges XZ and ZY.

3.5 The CSPT Heuristic

This algorithm has three steps.

1. Using Dijkstra's shortest path algorithm compute a lowest cost spanning tree to as many destination nodes in the multicast as is possible without any path breaking the arbitrary delay bound, Δ.

2. Use Dijkstra's algorithm to compute a shortest delay path tree to those multicast nodes not reached in the previous step.

3. Combine the lowest cost spanning tree from the first step with the shortest delay path tree from the second step making sure that the delay to any destination node does not break the delay bound, Δ, and that all loops are removed.

3.5.1 A Worked Example

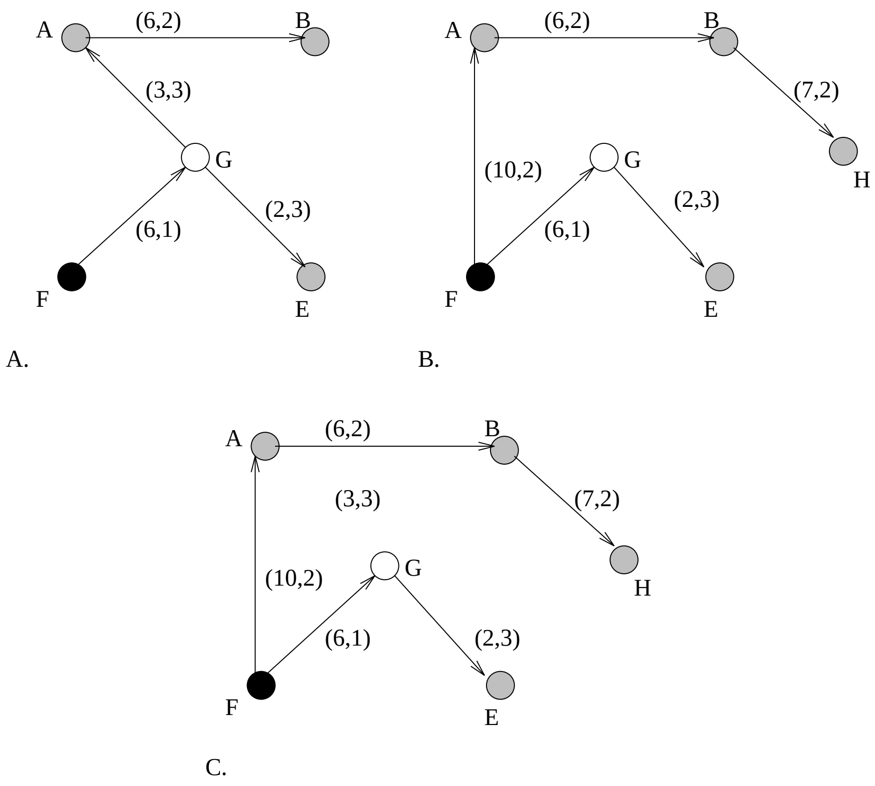

Figure 14: The CSPT heuristic: a worked example

Applying the first step of the heuristic to the network in Figure 4 produces the minimum cost path tree illustrated in Figure 14A. Node H is not included in this tree because its minimum cost path has a delay of 8, which breaks the delay bound. Figure 14B is the shortest delay path tree constructed only as far as node H, the multicast node not yet included in the solution. The combination of the minimum cost path tree and the shortest delay path tree will create a loop connecting nodes F and A. For this reason the edge FA is selected in preference to edge GA to give the final solution in Figure 14C. This tree has a cost of 31 units and a delay of 6. Loop removal in the CSPT heuristic is much simpler than it is with the CST_c heuristic. Because steps 1 and 2 both use Dijkstra's algorithm to compute their trees, a loop may occur. The loop can be avoided by selecting, from the loops downstream node, the shortest delay path tree branch in preference to the minimum cost path branch. This will increase the tree cost, but prevents violation of the delay bound. For example, Figure 15 illustrates how a

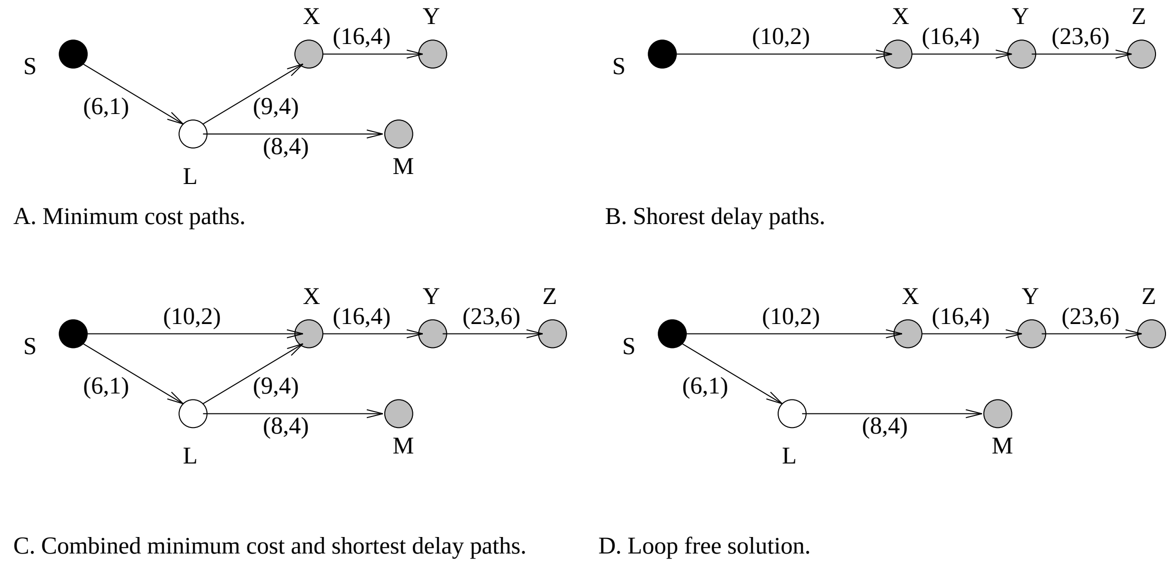

Figure 15: Loop removal in the CSPT heuristic

minimum cost spanning tree (A) and shortest delay path tree (B) when combined create a loop (C). By choosing the path SX from the shortest delay path tree and ignoring the path LX from the minimum cost path tree we obtain a loop free solution that does not violate the delay bound (D).

3.5.2 Time Complexity of the CSPT Heuristic

Each of the first two steps of the heuristic have the time complexity of Dijkstra's algorithm, which is at most $O(n^2)$. Because these two steps are independent of each other they can be performed in parallel. The last step has a time complexity of $O(n)$.

3.5.3 When CSPT costs more than the SPT.

For the majority of multicasts CSPT also calculates solutions that are cheaper than those produced by Dijkstra's SPT algorithm. As with CCET, there are also some cases where the cost of solutions found using the SPT algorithm can be cheaper than those found using the CSPT heuristic. In Figure 16, for a delay bound greater than 8, to connect the multicast nodes Y, W and Z to the source S, the CSPT heuristic will use the path YX, XS at cost 18 and path WZ, ZS at cost 13 because they are the shortest paths based on cost between the multicast nodes and the source. This results in a

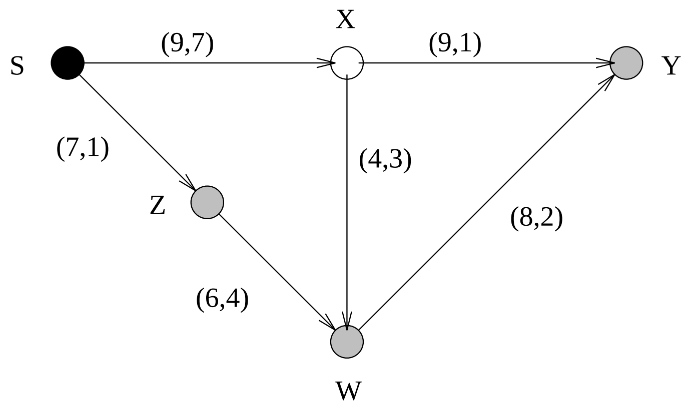

Figure 16: CSPT more expensive than SPT.

multicast tree of cost 31. The SPT algorithm based on delay will choose the path YW, WZ, ZS at a cost of 21 to connect all the multicast nodes to the source.

3.5.4 Bounds on Tree Cost

The minimum cost of a CSPT tree will be obtained where the multicast group can be connected using only the minimum cost spanning tree. On the other hand the maximum cost will be incurred if the multicast group can only be connected using the shortest delay path tree.

3.5.5 Multicast Tree Stability and Dynamic Groups

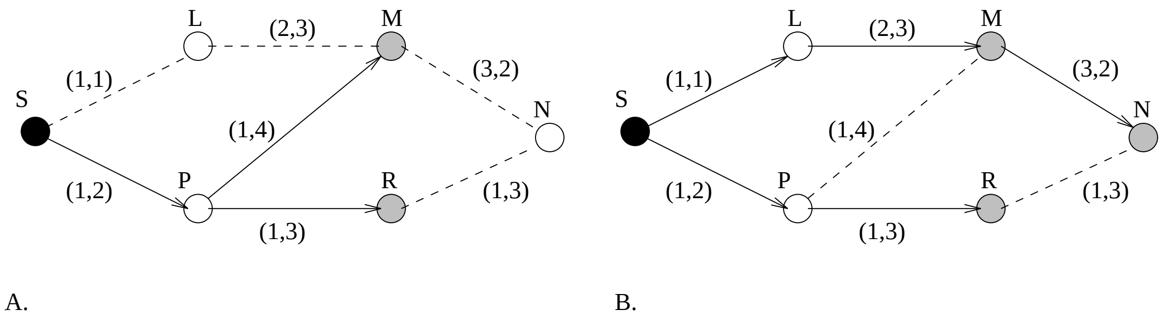

Figure 17: Changing topology of CSPT multicasts

As CSPT multicast trees grow, their topologies are prone to reconfiguration if the arbitrary delay bound is less than the delay along the cheapest path to the new destination node. This happens if the heuristic has to add the new node using a shortest delay path, which may require the removal of a cheaper path from the existing tree. In figure 17A. a CSPT multicast tree has been contructed to the nodes M and R from the source S. The arbitrary delay bound, Δ, is 7 so the multicast tree is constructed entirely of the cheapest paths SP, PM and SP, PR. If node N is added to the multicast it cannot be connected by the cheapest path (SP, PR, RN) because the arbitrary delay bound will be violated. The heuristic calculates a shortest delay path to node N which comprises the links SL, LM and MN which is added to the tree. The link PM is removed from the tree to prevent a forward loop. The resulting multicast topology is significantly different from its predecessor.

3.6 Evaluation Method

3.6.1 Benchmark Algorithms

The ideal benchmark algorithm to use would be one that produces an optimal delay bound, minimum cost, multicast tree. To find such a solution requires the enumeration of all spanning trees in the network that are bound by the arbitrary delay. Kompella [37] bases such a process on Kirchoff's method for enumerating spanning trees but removes all the spanning trees that break the delay bound and selects the cheapest tree as the solution. As Cayley's theorem shows [23] this method becomes intractable if the network size is increased because it has a time complexity of $O(n^{n-2})$. It is only practical for very small networks.

In the evaluation of their own algorithms Kompella et al first compare the performance of their most efficient method against the performance of optimal solutions [38]. The evaluations are limited to a few multicast groups sizes in small networks (20 nodes) because of the complexity of finding optimal solutions. They then use their most efficient algorithm as the benchmark for larger evaluations (50 to 100 node networks).

Salama et al [52] also use the optimal solution method as their benchmark for small networks (also 20 nodes), but use the bounded shortest path algorithm of Zhu et al [72] as the benchmark for larger networks. Salama et al chose this algorithm because in their simulations with small networks the solutions it produced were the closest to the optimal.

Sun and Langendoerfer [56] used Kompella et al's most efficient algorithm as their benchmark, in much the same way as Kompella used it.

We have chosen two different benchmarks for following reasons. Firstly we use the MST heuristic ([6] and [27]); the solutions provided by this heuristic are not bound by any arbitrary delay, but we record their delays and use them in the evaluation. Our reasons for this choice are that we can compute large "minimum" cost trees within a reasonable amount of time ($O(n^3)$) and that the solutions of the Kompella et al algorithm which are the most efficient of all the heuristics under evaluation, tend towards the MST solution as the arbitrary delay increases. The MST heuristic works as follows

1. Use Floyd's all pairs shortest paths algorithm to construct a complete graph, T, of all the nodes in the multicast group, including the source node, from the original graph, G.

2. Use Prim's minimum cost spanning tree algorithm to construct a minimum cost spanning tree, T', of the closure graph, T.

3. Map this minimum cost spanning tree back onto the edges they represent in the original graph G, removing any loops that may occur.

The MST heuristic applied to the graph in Figure 4 results in the tree in Figure 18B.

Secondly we use Dijkstra's shortest path tree as a benchmark to evaluate the cost savings made by using the various heuristics. We do this for the pragmatic reason that there is no optimal (or near optimal) solution to the problem that can be credibly used in real networks. We consider it more realistic to look at the savings a multicasting heurisitic can make in comparison to the shortest path multicasts currently in use [17]. Dijkstra's shortest paths algorithm applied to the graph in Figure 4 results in the tree in Figure 18A.

3.6.2 Network Models

Three models were used for the evaluation of the heuristics. The first is attributed to Waxman [66] and is used to generate single cluster networks such as backbones or autonomous systems. Doar's model [21] produces network clusters interconnected via a central core network. Although this model is intended to represent hierarchical networks, the interconnection between clusters and the network core is rudimentary. It has

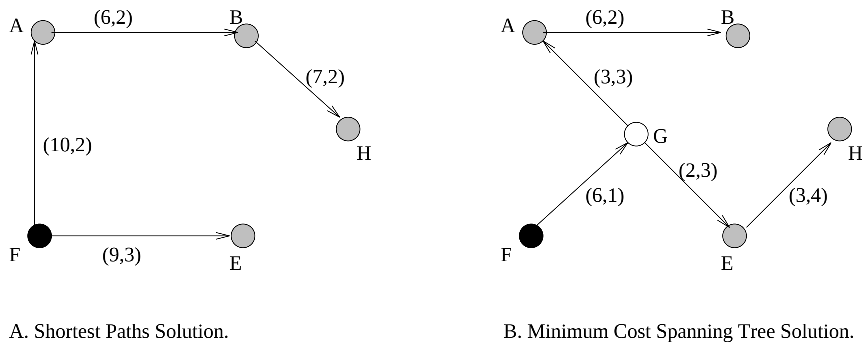

Figure 18: Benchmark solutions

been retained because its use leads to the discovery of rare but extreme behaviour in our heuristics. The third model extends Doar's ideas by generating a random backbone network to interconnect the network clusters. This model is a more realistic representation of real networks. We have not yet considered hierarchical networks with more than two levels nor interconnected adjacent area networks as models for our evaluations, although these have been proposed in both unicast and multicast routing protocols for the Internet ([42] and [41]).

The Waxman model randomly distributes nodes over a rectangular coordinate grid. The Euclidean metric is then used to determine the distance between each pair of nodes. Edges are introduced with a probabilty that depends on their length. The edge probability is given by

$$P(\{u,v\}) = \beta\, exp \frac{-d(u,v)}{L\alpha} \tag{1}$$

where $d(u,v)$ is the distance from node u to v, L is the maximum distance between two nodes, and α and β are parameters in the range (0,1). Small values of α increase the density of short edges relative to longer edges, while larger values of β produce higher node degrees. Edge lengths are used to represent delays. This implies that node queue processing delay can be ignored. Edge costs are selected at random from the range $[1..L]$. Figures 19 and 20 are examples of networks generated using Waxmans model.

The network model suggested by Doar is based on that of Waxman. Doar introduces a factor, related to the number of nodes in the network, to scale the probability of edges

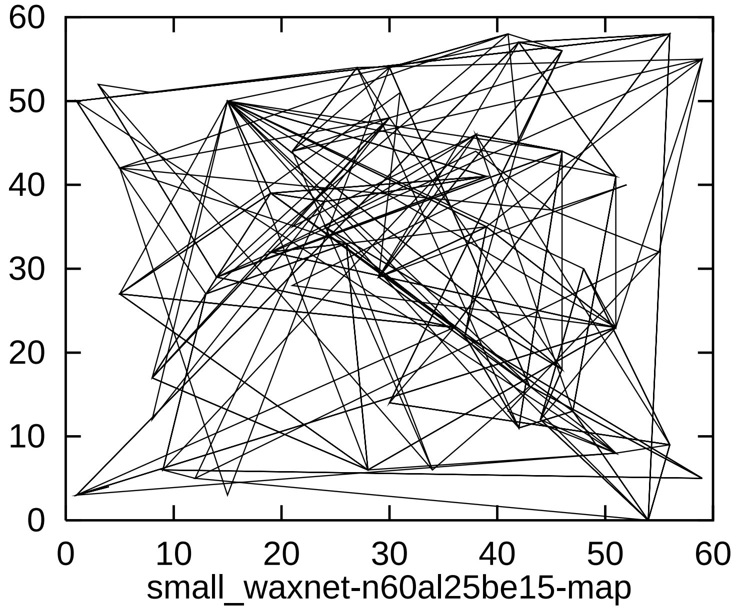

Figure 19: Waxman model with $\alpha = 0.25$, $\beta = 0.15$

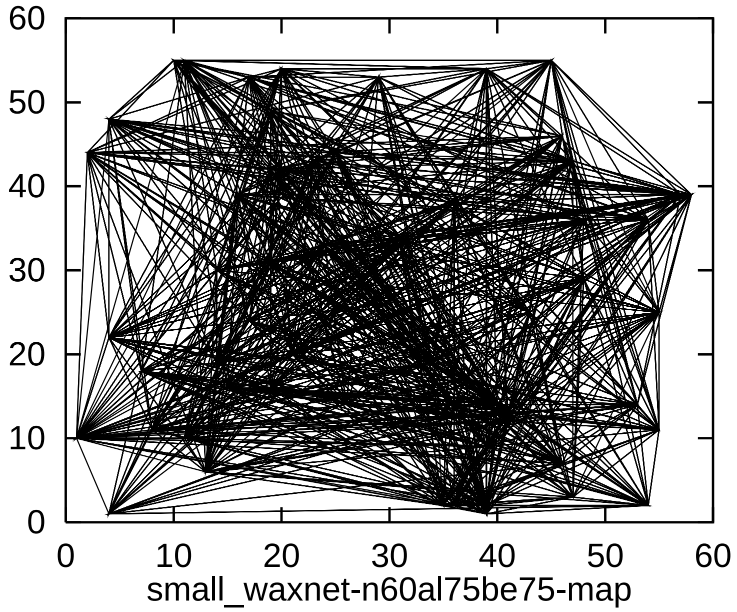

Figure 20: Waxman model with $\alpha = 0.75$, β =0.75

being included. The probability function becomes

$$P(\{u,v\}) = \frac{k\bar{e}}{n}\,\beta\, exp\frac{-d(u,v)}{L\alpha} \tag{2}$$

where k is a scale factor related to the mean distance between two random points, $\bar{e}$ is the mean degree of a node and n is the number of nodes in the graph.

Doar goes further by introducing hierarchical graphs as networks models. These are generated using the modified probability function to generate clusters of networks that can then be connected to a central core network using a fixed number of links. Figure 21 is an example of such a network.

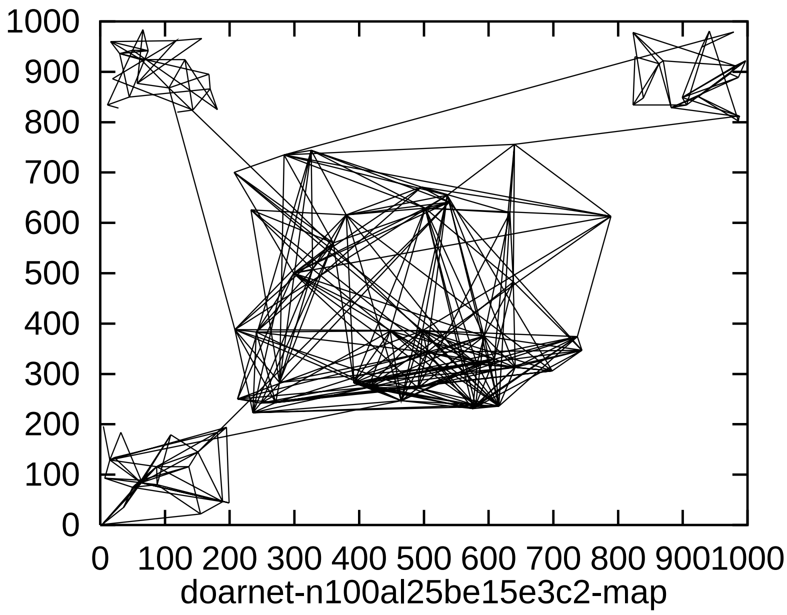

Figure 21: Doar model with $\alpha = 0.25$, β =0.15

The cluster interconnection mechanism proposed by Doar means that the number of links connecting each cluster to the core is predefined and static. We modified this rudimentary method to obtain a third network model. Our network model uses the modified probability function to generate cluster networks. We then use the same function to generate a network with as many nodes as there are clusters. Each node in this network represents one of the cluster networks and its edge lengths are scaled to represent interconnection distances between the cluster networks. This backbone network is then mapped onto the clusters by connecting together nodes selected at

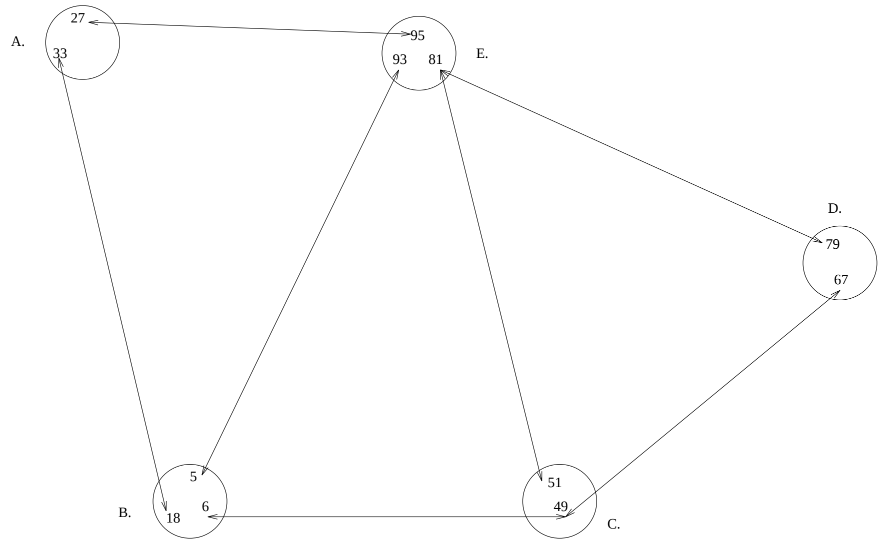

Figure 22: Crawford model network

random from each cluster. Figure 22 illustrates a backbone interconnecting network clusters. Nodes acting as terminators to links interconnecting clusters may be connected to one or more interconnecting links. The internal structure of each cluster will be similar to that in Figures 19 and 20. Because clusters are connected networks there are always paths, within each cluster, between their nodes that terminate interconnecting links. For example, in Figure 22 there will be a path between nodes 27 and 33 in cluster A and between nodes 5, 6 and 18 in cluster B. Thus, there will be a path between node 67 in cluster D and node 33 in cluster A, which traverses clusters C and B. Through this mechanism there exists a path between all pairs of nodes in the network, some of which go via the backbone.

3.6.3 Link Metrics

Arbitrary Delay Bounds

In our network models we use Euclidean edge lengths to represent link delays. These delays have no units of time associated with them. Node queue processing delay is assumed to be negligible, as implied by the network models.

In his simulations of multicasting algorithms Salama et al [52] use a similar network model, but assign delay on the basis of a propagation speed of two-thirds the speed of light in networks distributed over a 3000 by 2400 km^2 grid. They further assume that each network node is a non-blocking ATM switch and that node queue processing delay can be ignored when calculating link delay. Their simulations use an arbitrary delay bound of 0.03 seconds, as might be required by interactive video and voice applications. This delay was chosen to allow enough time for higher level end-to-end protocols to process transmissions without degrading the required quality of service.

In Distributed Interactive Simulation applications human reaction times may require delay bounds of 200-300 ms. Tightly coupled applications may require bounds as low as 100 ms. If these delays are applied over networks of typically 50 ms diameter then the range of arbitrary delay bounds required by applications may vary from being very close to the multicast bound to several times the network diameter.

We choose three arbitrary delay bounds to evaluate multicast algorithms. The tightest delay bound is the multicast delay. Evaluations using this arbitrary delay bound will be used to indicate the minimum improvement in network utilisation achievable by each heuristic. Our second choice is to use the network diameter as the arbitrary delay bound. This purely arbitrary choice provides an evaluation mid-point. As the arbitrary delay bound increases in relation to the network diameter so the maximum improvement in network utilisation for each heuristic will be achieved. Our third delay bound is that of the MST, the lowest cost tree that can reasonably be calculated.

Interpretation of the evaluation results may be adapted to time based link delays by scaling network edge lengths appropriately.

Costs

Edge cost is assumed to be a different function to edge delay, and might be a measure of some network resource used in a communication, to which a cost value can be made. For example, there may be a cost in using a satellite link rather than a terrestial link.

3.7 Evaluation of the Candidate Heuristics

3.7.1 Performance Averages

Figures 23 and 24 illustrate the percentage excess costs and delays of using the four heuristics described above. The excess costs are measured relative to the MST

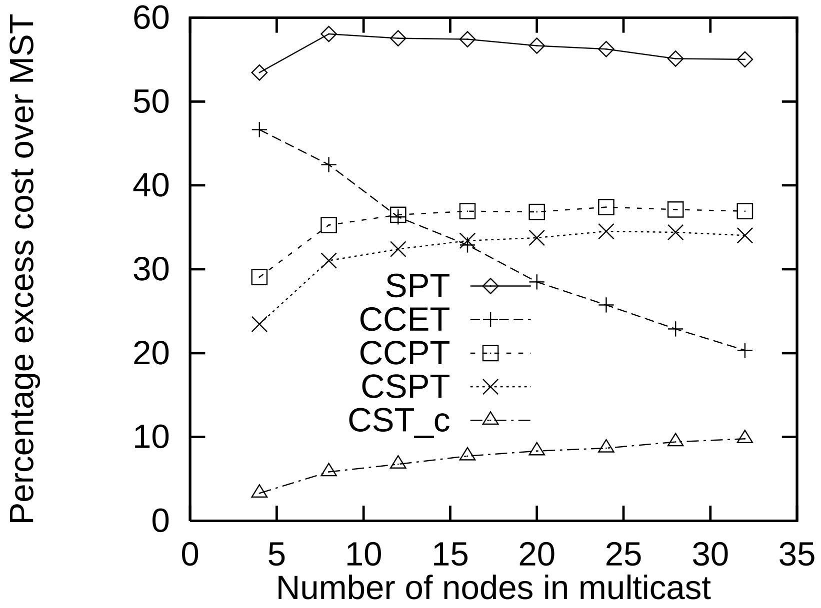

Figure 23: Average comparative cost: small networks

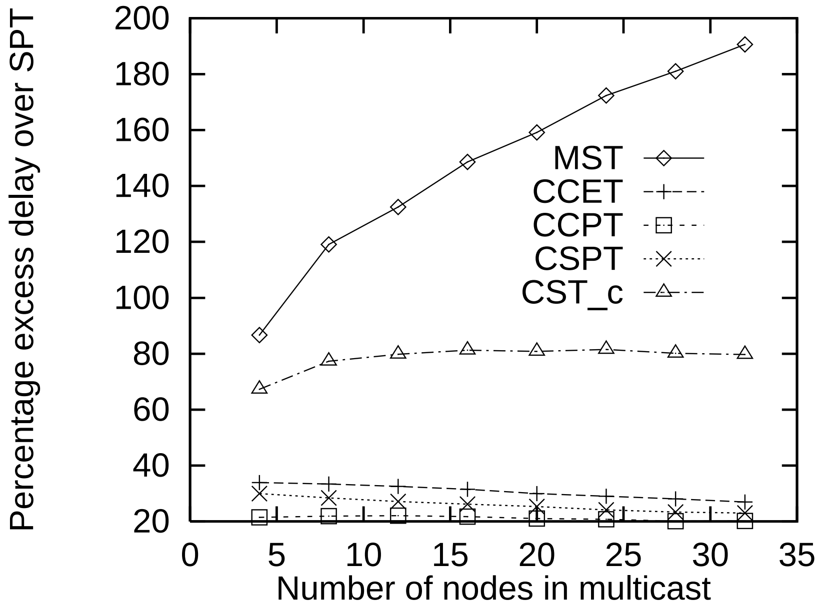

Figure 24: Average comaparative delay: small network

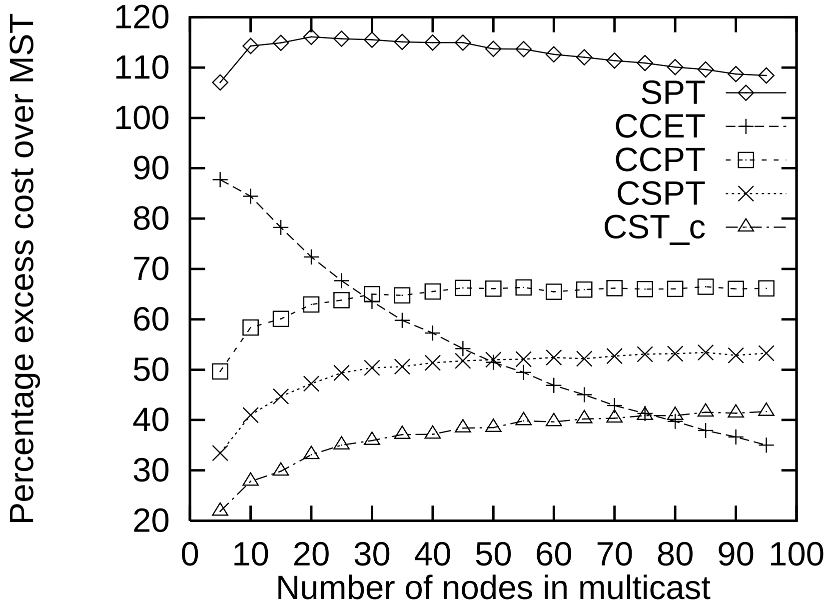

Figure 25: Average comparative cost: large network

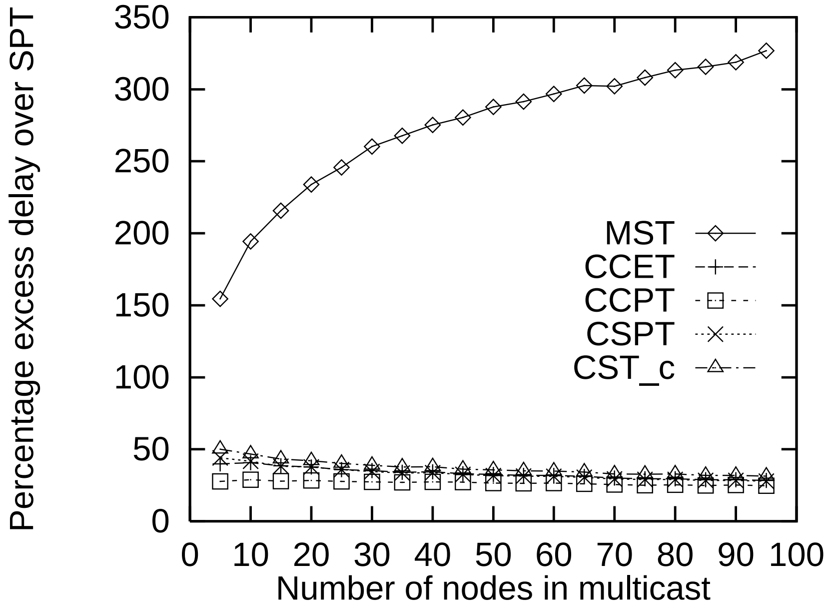

Figure 26: Average comparative delay: large network

benchmark and the excess delays are measured relative to Dijkstra's SPT benchmark. The evaluation networks have 35 nodes each and are of low edge density. The arbitrary delay bound is set to the diameter of the network.

The algorithm of Kompella et al (CST_c) generates multicast solutions that are on average cheaper than the other heuristics, although as the size of the multicast group size increases the Waters heuristic's solutions (CCET) coverge with those of Kompella et al. The performance of the CCET heuristic is much better than CSPT and CCPT heuristics beyond a multicast of size 20, but is worse for smaller multicasts. The CCPT heuristic shows poor performance in comparison with that of CSPT. As illustrated in Figure 24 Kompella et al and CCET make greater use of the delay bound to find cheaper paths. The performance of the heuristics is confirmed for larger networks in figures 25 and 26, although as the network size increases so does the percentage excess cost of each heuristic.

It is reasonably clear from the above evaluation results, albeit that they only cover low density single cluster networks, that the CCPT heuristic is in general significantly less efficient than the others, so it will not be considered further. It is also clear that the algorithm of Kompella et al is the most efficient, but is hampered by its high order time complexity.

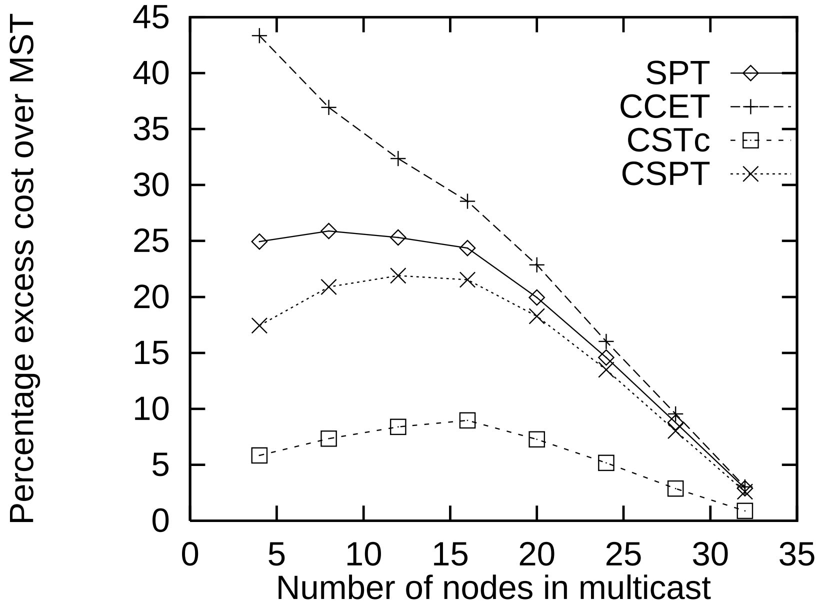

Figure 27: Average comparative cost: small network with unit link cost

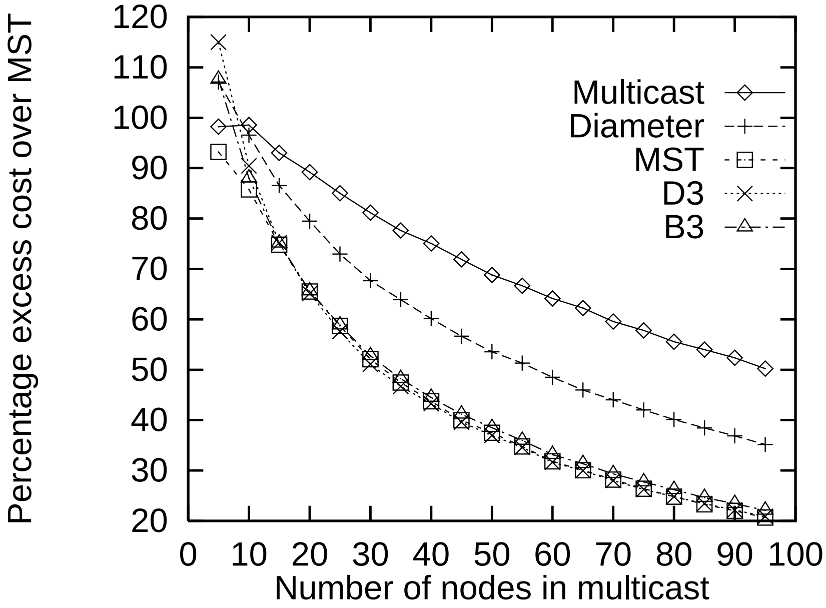

Figure 28: Average comparative cost for CCET as delay bound increases

Figure 27 illustrates the behaviour of the heuristics when the cost of each link in the network is the same. The CST_c and CSPT heuristics maintain a greater efficiency than SPT, but CCET becomes more expensive. Both CST_c and CSPT attempt specifically to minimise the cost of their solutions by growing their trees from the multicast source out to the destinations, so their behaviour is as expected. The CCET heuristic adds links by working back from each destination towards the source using the cheapest return links, so it has little control over the number of links a path might require. Rather it relies on the likelihood that cheaper links will lead to cheaper paths, irrespective of the number of links in the path. It also relies on the chance of paths meeting at the earliest possible point. If link costs are all the same the heuristic will choose the first return link from each node on a path back to the source, without minimising the number of links in the path. In this case path cost is a product of the number of its links and so this is not minimised. Figure 28 illustrates the improvement in solution efficiency of the CCET heuristic, as the arbitrary delay bound is increased. We have observed that as the delay bound approaches the MST delay, improvements in solution efficiency become negligible. Figure 28 shows how the costs of multicast trees with delays of three times the network diameter (D3), three times the broadcast delay (B3) and the MST delay almost coincide.

Up to these delay bound limits the number of nodes visited during the tree search in the heuristic's second stage is, by observation, less than $2n$. If the delay bound goes much beyond these limits the heuristic is occasionally prone to very long execution periods which suggests that either N or $|E|$ (or both) can become unacceptably large.

3.7.2 Specific Multicast Comparisons

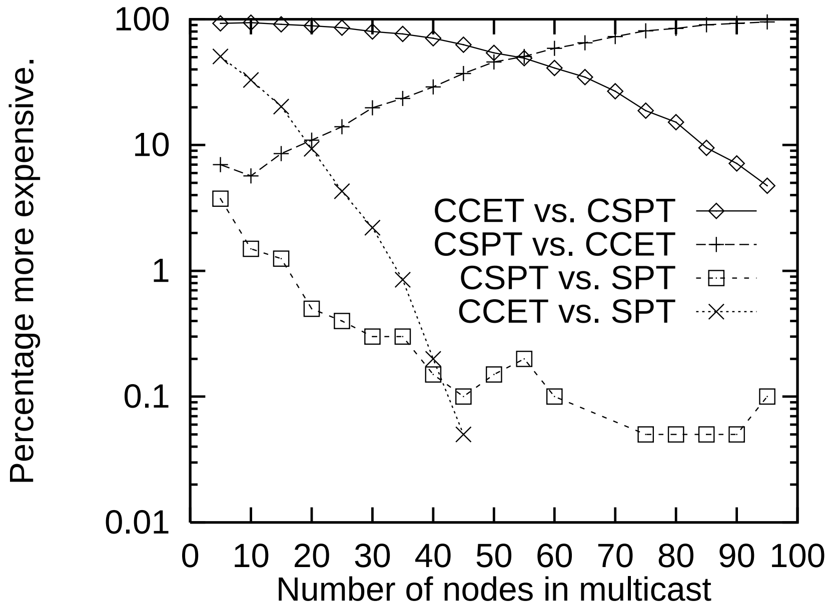

Figure 29: Exceptional comparative costs

The CSPT heuristic is generally better for smaller multicast group sizes, while the CCET heuristic is more suited to larger multicasts, although this is not always the case. Figure 29 illustrates the percentage of times CCET soutions are more expensive than those of CSPT and when the solutions of both CSPT and CCET are more expensive than Dijkstra's SPT. On average, the graph shows the expected results, where CCET is cheaper for larger multicast groups and CSPT is cheaper for smaller groups. In nearly 5% of the sample solutions for multicast groups of 95 nodes the solutions generated by CCET were more expensive than those generated by CSPT. Similarly, in 7% of the solutions for multicast groups of 5 nodes the solutions generated by the CSPT solution were more expensive than those generated by CCET. For smaller multicast groups sizes both CSPT and CCET generated some solutions that were more expensive than the

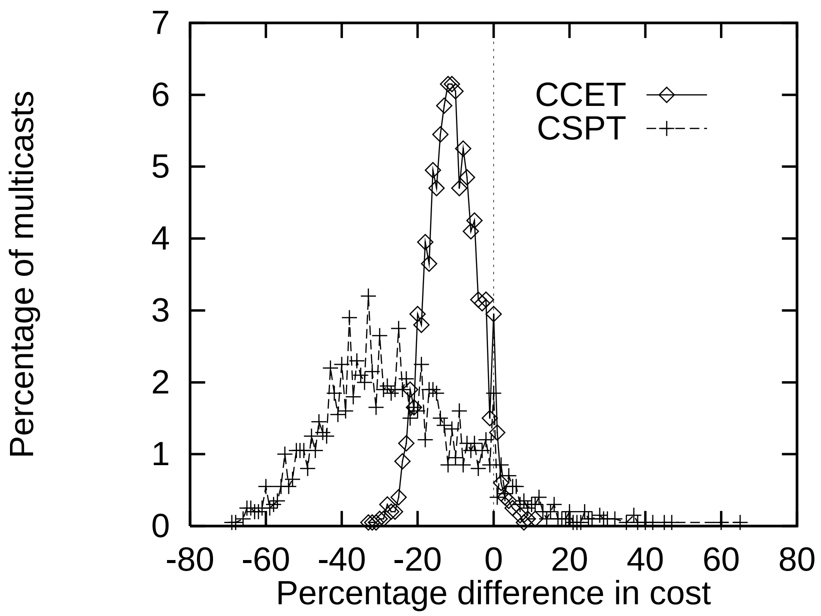

Figure 30: Cost distributions

SPT solutions. For larger multicast groups CSPT still generates some solutions that are more expensive than SPT, while CCET does not. Figure 30 indicates just how large and varied these differences can be. The graph for CSPT plots percentage cost savings of CSPT over CCET for small multicasts. While the majority of CSPT solutions are up to 69% cheaper, some can be up to 65% more expensive. Similarly, for CCET the majority of larger multicasts are up to 33% cheaper than CSPT, although some can be as much as 11% more expensive.

Although the CST_c heuristic has generally much better performance than any of the other heuristics, it is also prone to generating some inconsistent solutions, as illustrated in Figure 31.

This behaviour suggests, as might be expected, that the solutions each heuristic generates depend both on the algorithm and the topology of the network to which they are applied. None of these heuristics can realistically provide the "cheapest" multicast solutions in all networks for all multicast groups.

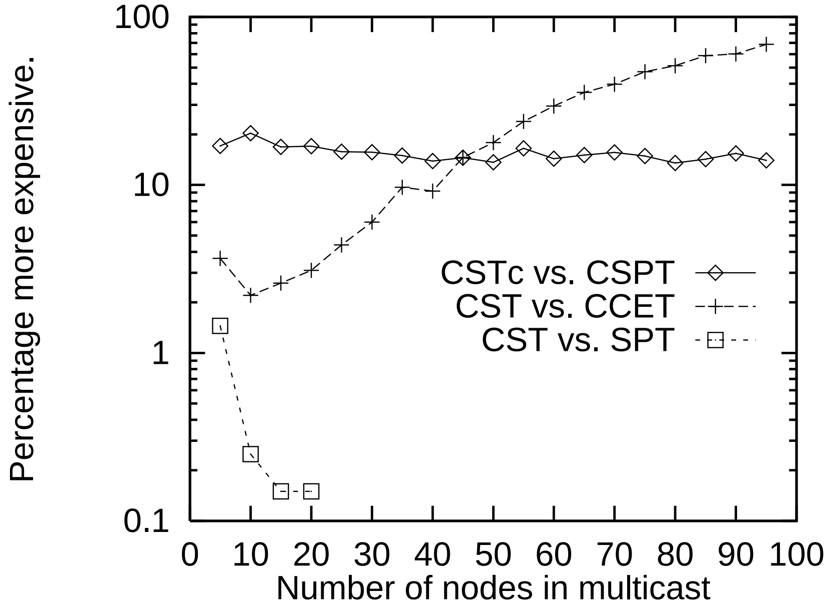

Figure 31: Exceptional comparative costs

3.7.3 Network Load and Multicast Failures

Figure 32 illustrates how each heuristic consumes network resources. In this evaluation multicast groups of differing sizes are successively applied to networks and the resources they consume are removed from the network before the next multicast is applied. Each multicast specifies a bandwidth requirement for the connection. Any links in the network that do not have sufficient bandwidth available for the connection are removed before the multicast is applied. The SPT heuristic then finds a multicast solution based solely on delay. All the other heuristics select links that have the highest available bandwidth, provided the delay constraint is not violated. As successive multicasts are applied bandwidth is used up and multicasts begin to fail because either the network becomes disconnected or, as the diameter of the network effectively widens, the delay bound is violated. This latter condition applies to the SPT sooner than the other heuristics because it does not necessarily spread the multicast tree evenly over the network [16]. Shortest path links that centre on the multicast source will become heavily used while those not centred on the source will not be utilised. As the bandwidth of the centred shortest paths is used so the paths become unusable, thus widening the network diameter and hence path delays. The other heuristics spread the paths they use for the

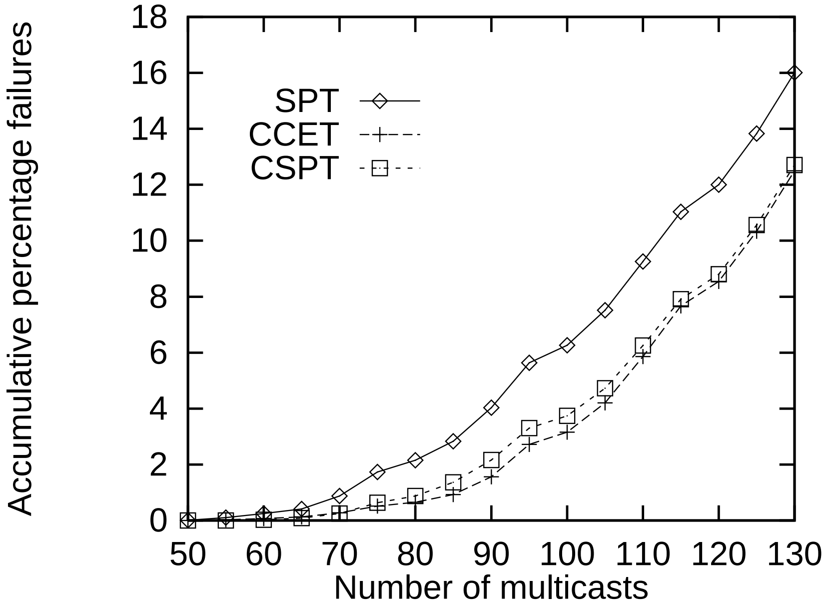

Figure 32: Network load and multicast failures

multicast tree over a wider area than the SPT because they use the slack between the shortest path delay and the delay bound to try and minimise tree cost.

Chapter 4

Hybrid Approach

4.1 Hybrid Multicast Heuristic

We conclude from our analysis and evaluation work that none of the heuristics we have considered can provide the "cheapest" multicast solutions in all networks for all sizes of multicast groups. They either take too long to find their solutions or are vulnerable to generating unacceptable solutions that depend on the network topology and/or the multicast topology. We propose that by combining heuristics of acceptable time complexity that can be efficiently integrated, the resulting Hybrid [13] will generate solutions that are predominantly cheaper than SPTs for all network topologies, for all multicast group sizes.

Kompella et al (CST_c) [38] generate good solutions but the algorithm has a high order of time complexity. As suggested by Widyono [68] this can be ameliorated by reducing the granularity of the arbitrary delay bound, but at the cost of compromising the algorithms accuracy. We have already discarded the Crawford (CCPT) heuristic because of its poor performance and we now discard the Kompella et al algorithm as well, because its time complexity is too high to be practical.

Because the Waters (CEPT) and Sun and Langendoerfer (CSPT) heuristics generate their most efficient solutions at opposite ends of the multicast group size range, their combination as a Hybrid might result in an heuristic of acceptable time complexity that produces solutions of significantly improved efficiency over shortest path trees. The absolute guarantee of minimal efficiency can be made if Dijkstra's algorithm is included in the Hybrid to cater for the rare instances where both Waters and Sun and

Langendoerfer produce solutions that are more expensive than the shortest path tree.

Integration of the three heuristics is simple. As all three heuristics use Dijkstra's shortest path tree for delay, the Waters heuristic can be modified to output both it's own solution (CCET) and the shortest path tree, based on delay (SPT), which is calculated by the first stage of the heuristic. The Sun and Langendoerfer heuristic also needs Dijkstra's shortest path tree for cost, a calculation which can be conducted simultaneously with the Waters calculation. When the SPTs based on delay and cost are available they can be combined, if necessary to give the Sun and Langendoerfer (CSPT) solution. Once all three solutions (SPT, CCET and CSPT) have been obtained their costs can be easily calculated and the cheapest tree selected as the solution.

The Hybrid heuristic procedure is as follows

- Execute the modified Water's heuristic to return the Water's solution and Dijkstra's SPT for the multicast.
- Execute the first step of Sun and Langendoerfer's heuristic to obtain a lowest cost spanning tree to as many destination nodes in the multicast as is possible without any path breaking the arbitrary delay bound.
- If not all of the multicast nodes have been reached in the previous step combine the shortest paths to these nodes from Dijkstra's SPT with the lowest cost spanning tree, making sure that the delay to any destination node does not break the delay bound. This is the last step of Sun's heuristic.
- Calculate the cost of the Waters solution.
- Calculate the cost of Dijkstra's SPT.
- Calculate the cost of the lowest cost spanning tree, with additional SPT paths.
- Select the cheapest tree from the above three steps as the multicast solution.

4.2 Evaluation of Hybrid Heuristic

4.2.1 Performance Averages

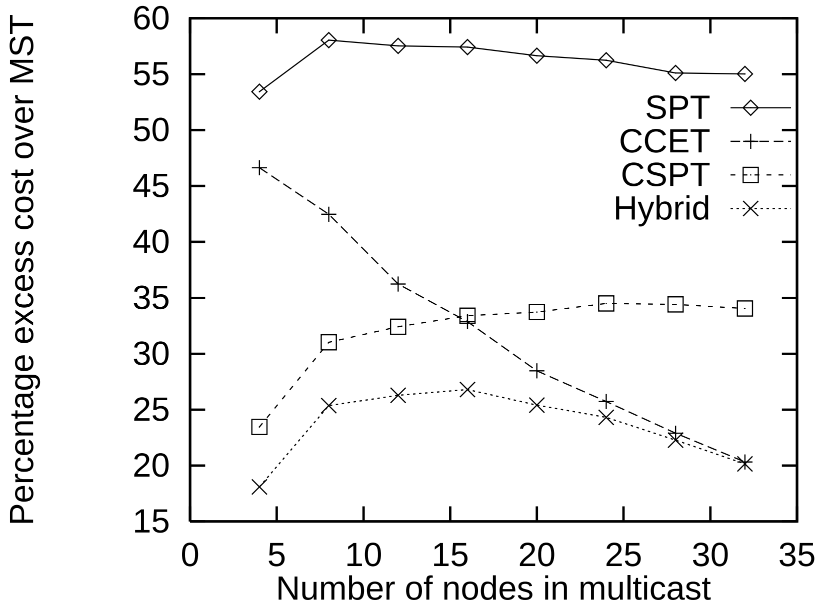

Figure 33: Average comparative cost: Δ = network diameter, small network.

Figure 33 and 34 illustrate the cost performance of the Hybrid heuristic in comparison to CCET and CSPT for single cluster networks of 35 nodes and 100 nodes respectively. The arbitrary delay bound is set to the network diameter, our chosen "mid-point" delay bound. Figures 35 and 36 show the performance of the heuristic for the tightest delay bound (when Δ is set to the multicast delay) and the loosest delay bound (Δ is set to the MST delay bound), in 100 node single cluster networks. Figures 37, 38 and 39 illustrate the cost performance of the heuristic in hierarchical networks of 100 nodes for the three arbitrary delay bounds we used.

4.2.2 Specific Multicast Comparisons

In all examples the Hybrid outperforms or equals both CCET and CSPT; the improvement is particularly noticable in smaller networks or where the multicast group size is small. This occurs because both CCET and CSPT are at their most volatile in these cases whereas the Hybrid is able to select the better solution of either heuristic. As the network size increases there is a greater coincidence between the Hybrid and

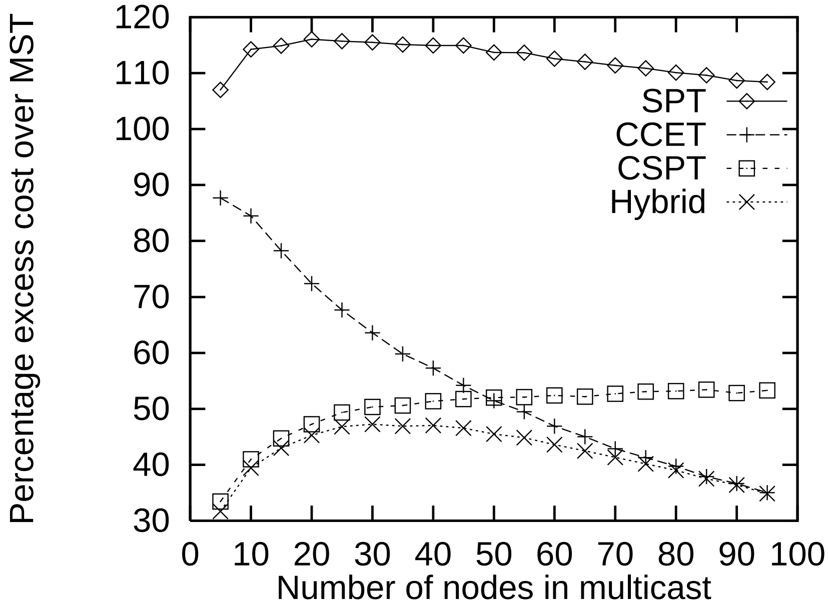

Figure 34: Average comparative cost: Δ = network diameter, large network.

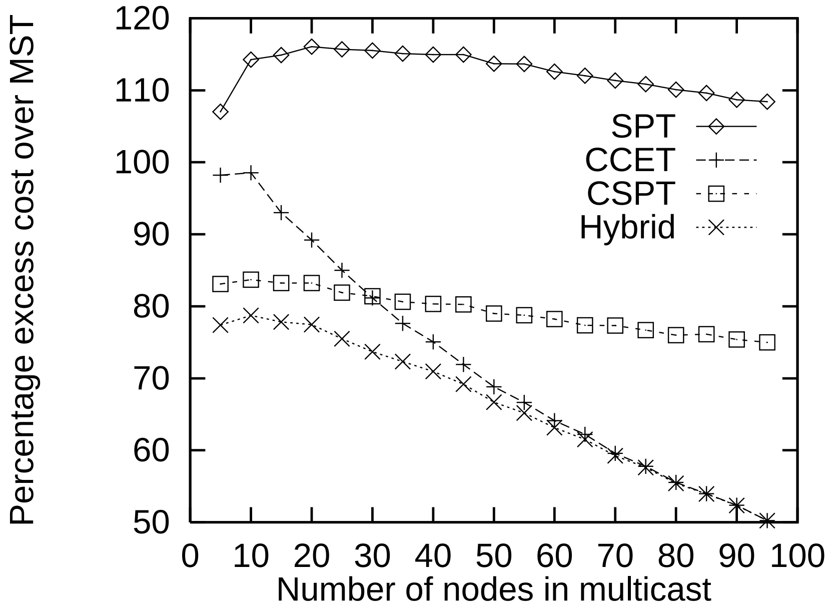

Figure 35: Average comparative cost: Δ = multicast delay bound, large network.

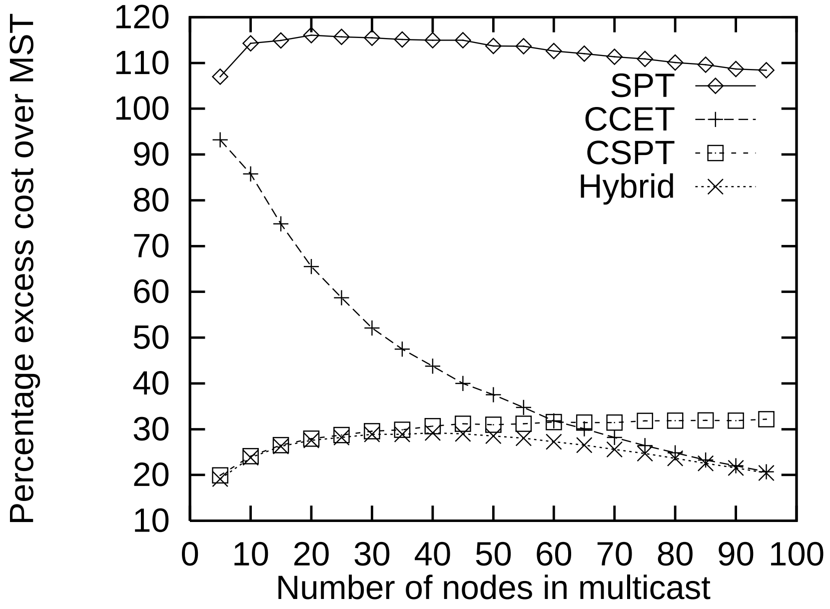

Figure 36: Average comparative cost: Δ = MST delay bound, large network.

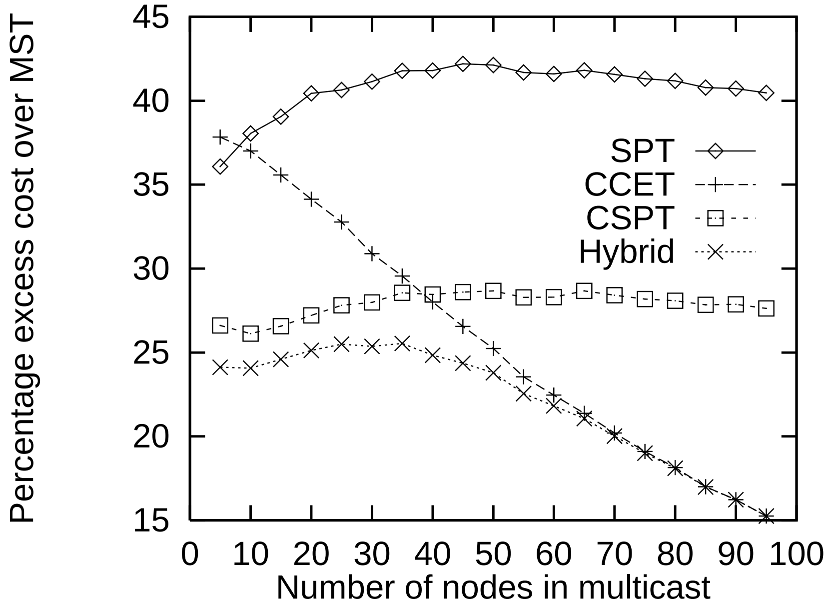

Figure 37: Average comparative cost: Δ = multicast delay bound, hierarchical network.

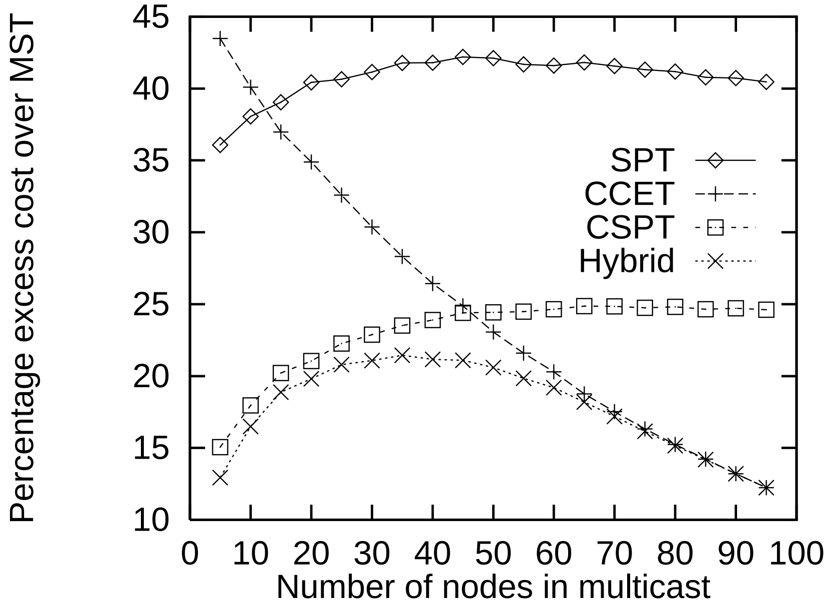

Figure 38: Average comparative cost: Δ = network diameter, hierarchical network

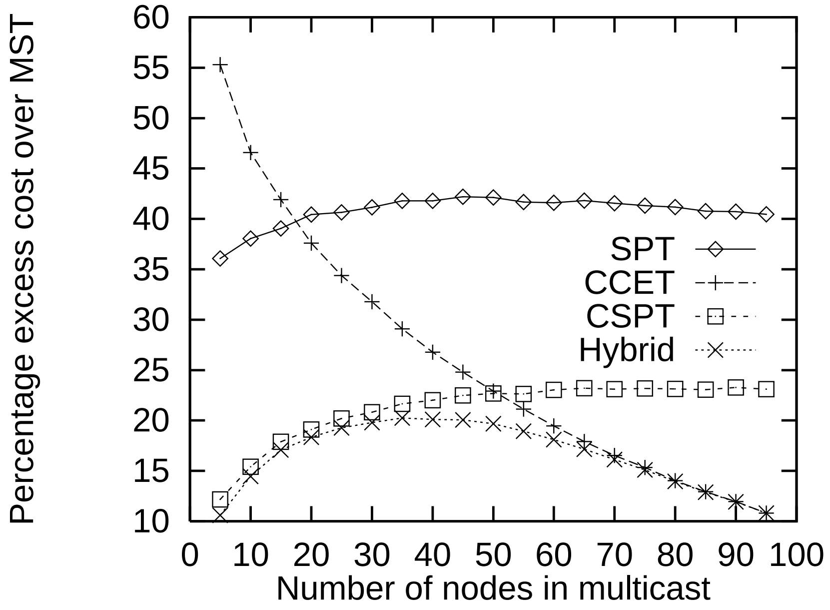

Figure 39: Average comparative cost: Δ = MST delay bound, hierarchical network.

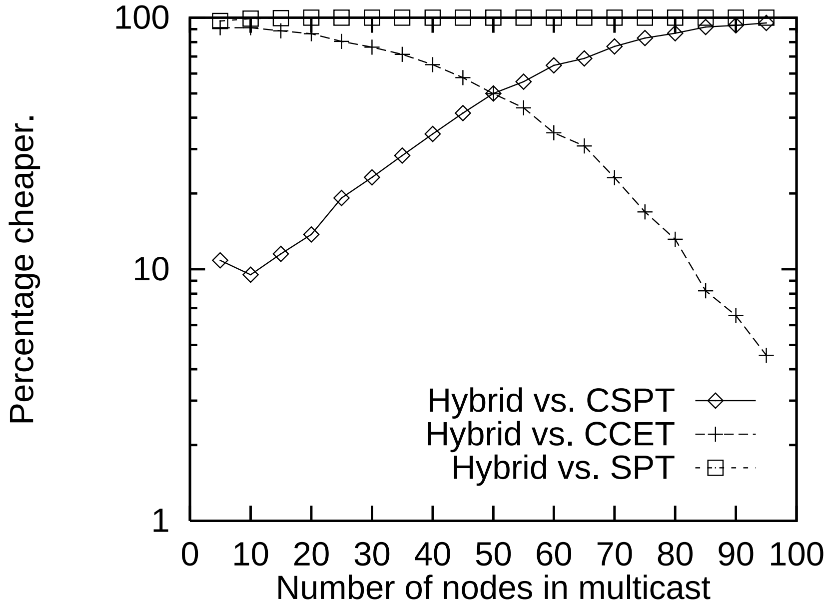

Figure 40: Hybrid cost saving

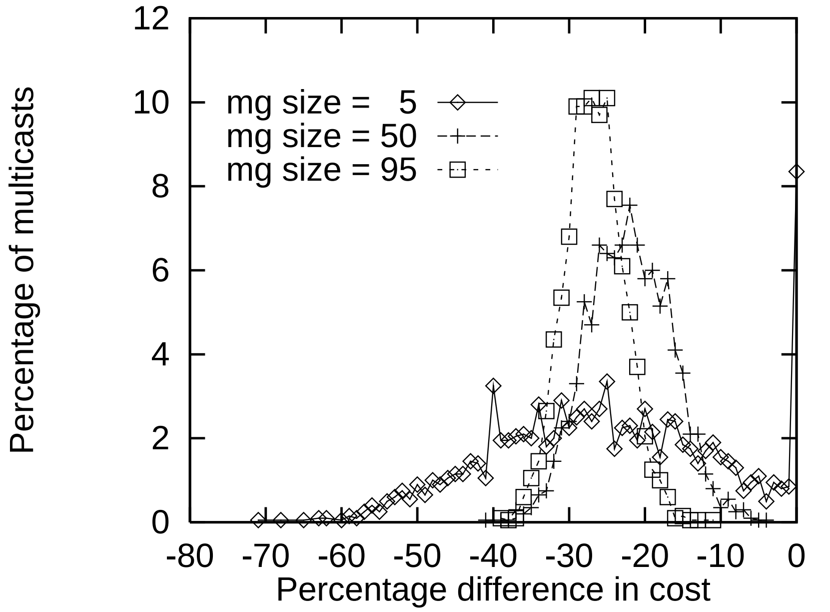

Figure 41: Hybrid cost variance

its constituent heuristics. Figure 40 illustrates the percentage of times the Hybrid is cheaper than CCET, CSPT and SPT. The SPT function in the Hybrid only achieves cheaper solutions in 0.33% of the cases.

Figure 41 shows how much cheaper the solutions calculated by the Hybrid can be, than those calculated using the SPT. The figure also illustrates the variance in cost savings of using the Hybrid over SPT, for different sizes of multicast groups. For smaller multicast groups the variance is, as expected, quite wide. As the multicast group size increases, the variance narrows. In all cases the Hybrid calculates solutions that are either cheaper than those calculated using the SPT, or are the same. In the sample illustrated, just over 8% of the five node multicast solutions calculated using the Hybrid had the same cost when calculated using the SPT.

4.2.3 Network Load and Multicast Failures

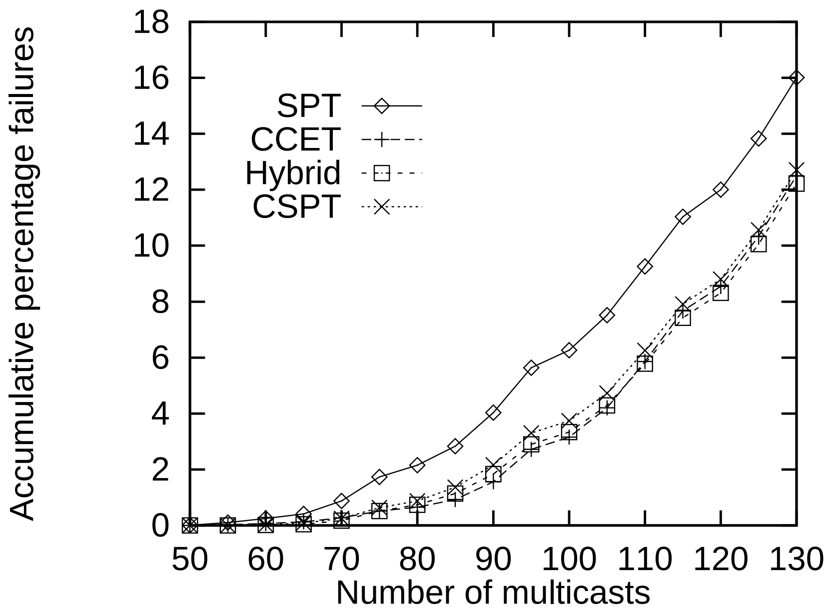

Figure 42: Network load and multicast failures

As noted in section 3.7.3 the SPT uses up network resources faster than CCET or CSPT. We see, In Figure 42, that as we would expect, the Hybrid consumes much the same resources as its constituent heuristics.

4.2.4 Multicast Tree Stability and Dynamic Groups

Heuristic	Hybrid	Hybrid using $sCSPT$
% multicast tree reconfigurations	85.13	93.74
Average % number of paths changed per reconfiguration	28.79	37.9

Table 1: Percentage of Hybrid trees reconfigured

The Hybrid heuristic is prone to reconfigure the multicast tree as nodes join and leave the multicast group. This happens for either of the following reasons. The Hybrid may switch the heuristic used for the solution when a node joins or leaves the multicast group, thus calculating an entirely "new" solution for the "new" multicast group. On the other hand, the Hybrid may not need to switch heuristics, but might already be using CSPT which is prone to reconfiguration as the multicast tree grows. Table 1 gives the percentage of multicast trees that were reconfigured at least once during their growth, and the average percentage number of path changes per reconfiguration. We also include results for a variant of the Hybrid that uses the sCSPT heuristic (the static, and less efficient, version of CSPT) which reduces the number of times the multicast tree is reconfigured due solely to the use of CSPT, but increases the percentage of multicast tree reconfigurations overall. From these results it would appear that the number of reconfigurations saved by using sCSPT in the Hybrid, instead of CSPT is significantly outweighted by the increase in the number of reconfigurations caused by reselection of the calculating heuristic, as the multicast group changes.

By aggregating paths between the multicast source and multicast destinations, extra delay is introduced along some of the paths in the multicast tree. The closer a destination node is to the source, the greater is the chance of its source-destination path being aggregated with a longer path to a more distant node. This does not present any problems for the multicast solution addressed in this work, since the arbitrary delay bound is not violated by the extra delay introduced. Nor does the extra delay imply that data remains in the network any longer than it would otherwise do. The very purpose of the aggregation of paths is to reduce the replication of data across the multicast tree without violating the arbitrary delay bound. Figures 43 and 44 illustrate the distribution of path delays across all 5 node multicast groups, and all multicast groups, respectively. The delay (the x-axis) is normalised against the arbitrary delay

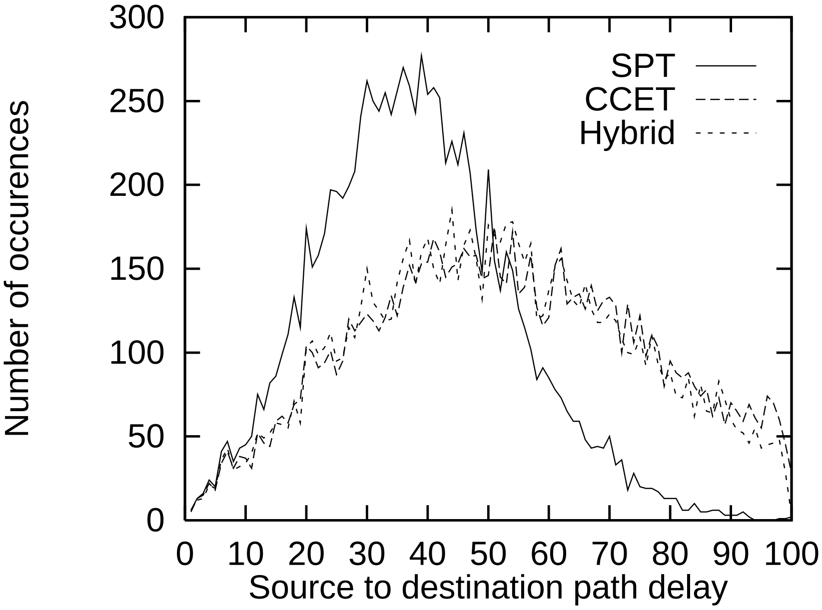

Figure 43: Distribution of path delays across all 5 node multicasts

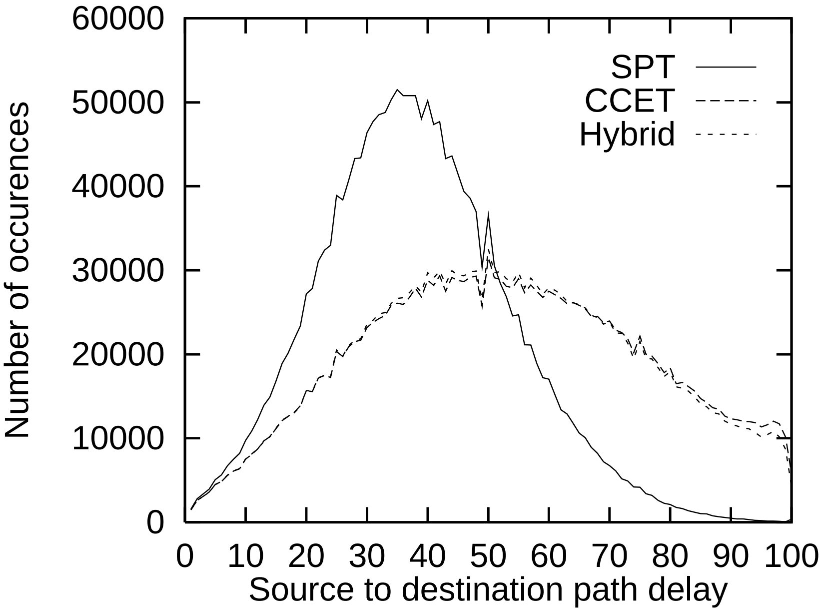

Figure 44: Distribution of paths delays across all multicasts

bound, while the number of occurences is the actual number of paths of each length for all the multicasts. There is little difference in the shape of the distributions for the 5 node multicasts and all the multicasts. We deduce from this that the distribution probably holds for all multicast group sizes. As expected, the distribution shows that the paths produced by the SPT algorithm have lower delays. In both cases, different delays will be perceived by different recipients. It is not practical to take remedial action in the network to equalise the delays (e.g., by buffering cells at the switches or taking slower paths). Where it is necessary to play information back at the same time, buffering must be provided in the destination stations. In this case, on average, less total buffer storage will be required for the hybrid than if the shortest paths tree algorithm is used.

Chapter 5

Multicast tree calculation for Dynamic Groups

With the exception of the CCET, the heuristics we have discussed so far may reconfigure the multicast tree topology when multicast group membership is dynamic. This characteristic is also true of the benchmark MST algorithm. As suggested in our introduction, this behaviour may have an unacceptable impact on destinations already in the multicast group. Although the CCET heuristic does not reconfigure solutions as group membership changes, it is not particularly efficient for small groups. The more efficient the solution of an heuristic, the more likely it is to reconfigure as group membership changes.

Table 2 gives the percentage of multicast tree reconfigurations that occured during the tree growth, and the average percentage number of path changes per reconfiguration. These results were obtained by growing a sample of 20 multicasts per network from group size 1 to group size 34 over 200 single cluster 35 node networks. The arbitrary delay bound was set to the network diameter.

Heuristic	MST	CST_c	CSPT
% multicast trees reconfigured	89.22	100	67.53
Average % number of paths changed per reconfiguration	23.98	20.66	11.56

Table 2: Percentage of MST, CST_c and CSPT trees reconfigured

The CSPT and CST_c heuristics can be modified to eliminate the instability of

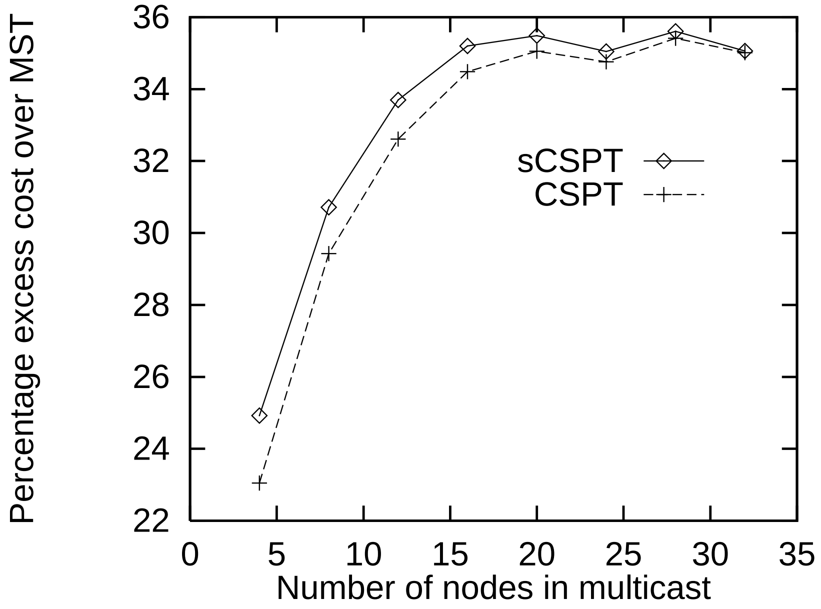

Figure 45: Average comparative cost of sCSPT and CSPT

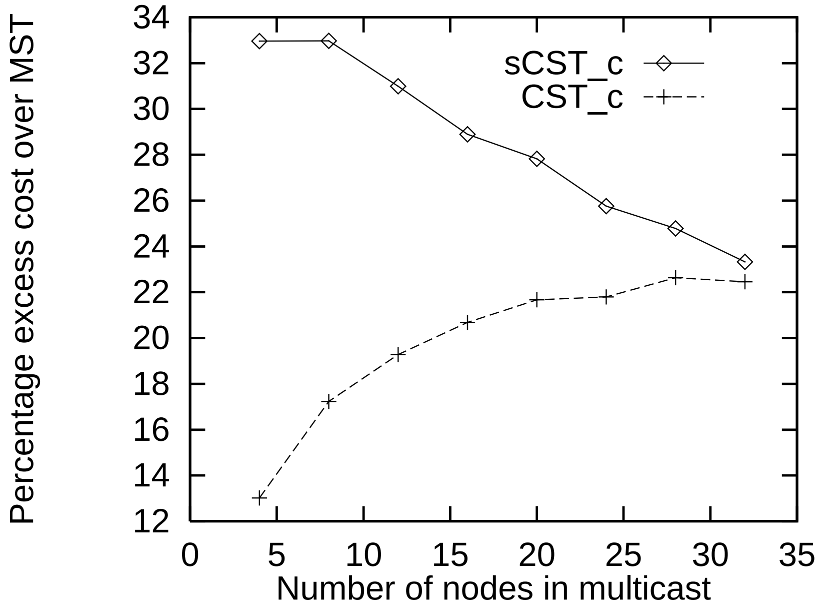

Figure 46: Average comparative cost of sCST_c and CST_c

their solutions as multicast group membership changes, by initially including all nodes of the network in the calculation and then extracting the multicast solution from the result. This method is used by the CCET heuristic. We label the modified versions of these two heuristics sCSPT and sCST_c, respectively. Figure 45 illustrates the cost performance for CSPT and sCSPT. The cost difference between the variants is greater for smaller multicasts than larger ones, as would be expected. The actual cost difference, as opposed to the difference in percentage excess over the cost of an MST solution, is on average less than 1%, making the sCSPT an acceptable alternative to CSPT where required. Figure 46 illustrates the cost performance for CST_c and sCST_c. In this case, for smaller multicasts, the percentage excess cost over the MST solution is markedly different for each of the heuristic variants. In fact the sCST_c cost performance tends towards that of CCET, as illustrated in Figure 47. By using a cheapest cost path

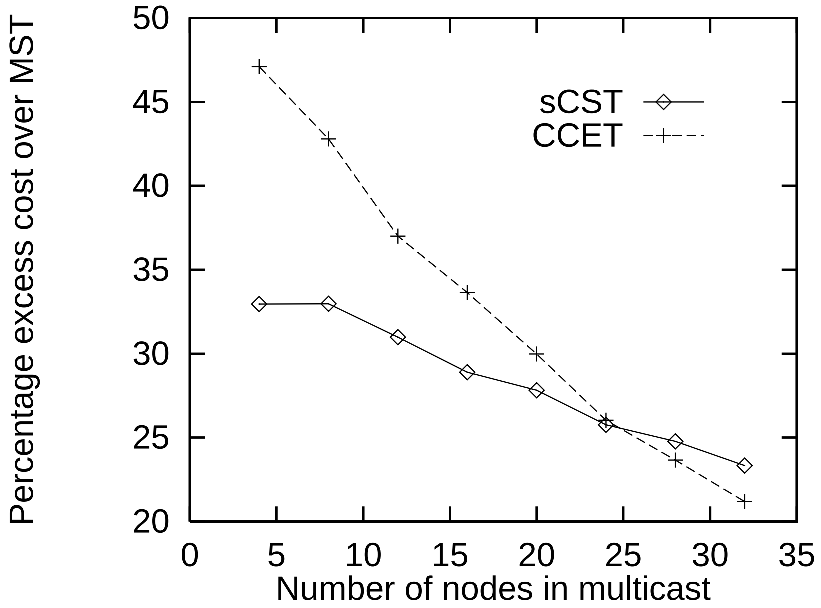

Figure 47: Cost perfromance of sCST_c verses CCET

algorithm to attach nodes to the multicast tree the CSPT and sCSPT heuristics find cheaper multicast trees for sparse multicasts. As the multicast tree grows, paths to additional nodes are included in the multicast tree on the basis of either their cost or delay from the source. There is no direct attempt to aggregate routes and so the tree cost steadily increases. Both sCST_c and CCET aggregate cheapest paths to produce efficient solutions for the broadcast case. This means that paths to many nodes are not the cheapest in themselves. The cost saving of the multicast occurs because expensive paths are aggregated to reach more destinations. Consequently, as the broadcast tree is pruned to find solutions for multicast groups of reducing size, the result is a tree that has an increasing proportion of expensive paths to fewer nodes, i.e. there is less aggregation of paths that are not the cheapest routes to the destination nodes.

The advantage of extracting multicast solutions from static broadcast trees is that the multicast tree does not need to be re-configured as nodes join and leave the group. When a node joins, if it is not already in the solution, its path of attachment to the multicast tree can be easily extracted from a cached broadcast tree. Nodes leaving the multicast are removed only if they are not on a path between the source and another node. Path removal also uses the broadcast tree to trace the route to be removed from the multicast solution.

Chapter 6

Application of the Heuristics

6.1 Characteristics Required of an Heuristic

To be of practical use for multimedia applications in high speed networks, a bounded delay low-cost multicast routing algorithm should aim to have

- Low time complexity, since route calculation may have to be performed in "real-time" at call set-up or as part of a distributed route calculation as user data crosses a network.

- Narrow variance in the efficiency of its solutions to minimise the occurence of expensive multicast solutions being calculated.

- Uniformity of performance over all possible multicast groups.

- Minimal or no reconfiguration of a multicast tree caused by receivers joining or leaving a multicast group because of the impact it may have on established user data flows. Switching paths in a multicast delivery tree may increase transmission jitter or cause user data packets in flows, temporarily, to arrive out of sequence. The degree of disruption to existing flows will depend on the scale of any reconfiguration. In this case, reconfiguration does not include the addition of links required to join the new receiver to the multicast tree. It only refers to links that are dropped from an existing solution and replaced by other links, because more efficient paths are to be found as a consequence of a receiver joining or leaving the multicast group.

Heuristics that do not satisfy all of these criteria may still have limited application.

6.2 Characteristics of the Heuristics Evaluated

Our work has identified a number of characteristics that appear to be generally applicable to the low-cost multicast heuristics we evaluated in Chapter 3. These characteristics, which do not apply to the Hybrid, are

- The closer to the optimal the solution of an heuristic is, the higher its time complexity.
- Single metric shortest path algorithms, such as Dijkstra's SPT, add paths to multicast destinations as delivery trees grow. This is not the case for delay constrained multiple metric algorithms, such as those we have evaluated here. The CCET heuristic, for instance, constructs a broadcast tree which it then prunes back to the multicast tree. The CSPT and CST_c heuristics have to remove any forward loops from their delivery trees before they can be used as multicast solutions. This characteristic of constructing the complete multicast tree before any path can be used, needs to be taken into account if any of the heuristics evaluated is being considered for implementation in connectionless networks. Unlike routers that use single metric path calculations, routers that construct delay constrained multiple metric paths will not be able to update their user data forwarding tables until the entire multicast tree has been calculated.
- The solutions of the heuristics studied are prone to wide variance. This variance can be so large that, sometimes, the solution provided by a shortest delay path tree is more efficient than that found by using a heuristic. Further, it is not possible to easily predict for which multicast/network combinations this extreme variance occurs.
- The efficiency of each heuristic solution is not uniform across the range of all multicast group sizes. In general some heuristics find efficient multicast solutions for small multicast groups, while others find better solutions for large groups.
- The heuristics we have evaluated select their solutions from search spaces that contain either broadcast trees or multicast trees. Broadcast trees include all nodes

in the network. Once selected, a broadcast tree is pruned back so that its leaves are the multcast group. Multicast trees include only those nodes in the multicast group, and the intermediate nodes on the paths from the source to the multicast destinations.

  - Once calculated, heuristics that use broadcast tree based solutions are able to join new receivers to the multicast tree without having to recalculate their solution. Heuristics that use only multicast trees either have to recalculate their solutions to include the new reciever or use some other method to perform the join. Mechanisms for joining new destinations to multicast trees vary from those that use simple shortest delay paths, but ignore cost [39] to genetic algorithms that attempt to find joining paths which balance cost and delay [53].
  - Heuristics that select their solutions from a search space based on multicast trees produce more uniform results over the range of possible multicast group sizes than those that use broadcast trees.
  - Heuristics that select their solutions from a search space based on broadcast trees find very efficient solutions for large multicast groups but appear to do badly for small multicast groups.

- Combining several heuristics into a hybrid reduces the occurence of inefficient multicast solutions, which are limited to the worse case performance of a shortest delay path tree. However, the hybrid approach means that the multicast solution often requires reconfiguration as nodes join or leave the multicast group, because there may be jumps from one to another of the constituent heuristics.

- All the heuristics evaluated require up-to-date knowledge of the entire topology of the network to which they are applied. For this reason implementation of the heuristics will require the use of "link-state" routing protocols, such as PNNI. When implementing a protocol to support a particular multicast tree calculation, consideration should be given to the cost of constructing and maintaining the tree in comparison to the network resources the chosen heuristic may save by its use [67].

6.3 Combining Heuristics, Multicast Types and Network Types

As discussed in Chapter 2, we identified two extreme types of multicast groups and three types of network

- Multicast group types.
  - Heavyweight; which will tend to have little or no changes of group membership. Closed multicast user groups are heavyweight.
  - Lightweight; which may have considerable changes of group membership. Open multicast user groups are lightweight.
- Network types.
  - Static: connections are reconfigured within the network.
  - Hard state: connections are expensive to reconfigure.
  - Soft state: paths reconfigure themselves at releatively low cost.

We also note that multicast tree reconfiguration will be minimised or eliminated where multicast group membership is (almost) static or a single, broadcast tree based, heuristic is used for the tree calculation. From our observations we conclude that the heuristics evaluated might satisfy a number of different low cost, delay bound, multicast routing scenarios, as summarised in Figure 3.

- Heavyweight multicasts are (almost) static and so might best be supported by a hybrid heuristic, since this will calculate the lowest cost solution for the multicast tree. Reconfiguration is only likely to occur if the multicast group membership changes. The level of disruption to user data flows due to any rare reconfiguration caused by nodes joining or leaving multicast groups would need to be assessed.
- A lightweight multicast may have extremely dynamic membership behaviour. For this reason, in a hard state network the hybrid heuristic approach is inappropriate. A single heuristic, based on a broadcast search space, might be the best solution because it will not reconfigure the multicast tree as receivers join or leave the

network type / multicast type	hard state (e.g. connection oriented)	soft state (e.g. connectionless)	static (e.g. pre-configured)
heavyweight (sender join)	Hybrid	Hybrid	Hybrid or SuperHybrid
lightweight (receiver join)	sCSPT	sCSPT or Hybrid	not applicable

Table 3: Combining heuristics, network types and multicast types

multicast group. The level of risk associated with the occurence of inefficiency spikes of a single heuristic would need to be assessed.

- A lightweight multicast in a soft state network could be supported by a hybrid heuristic. Although the multicast tree will probably require reconfiguration quite often, the costs of doing so are unlikely to be high.

- In the case of static routing or the calculation of permanent connections, such as permanent virtual circuits in ATM networks, we would propose the extension of the hybrid to a super hybrid that includes a larger variety of multicasting heuristics than we have in the hybrid discussed here. For example, the heuristic of Kompella et al [38] might be suitable as part of a super hybrid heuristic. It is unlikely that time complexity would be a predominent consideration where multicast solutions are calculated "off-line" and downloaded into the network. If time complexity were a significant contribution to the process of creating or modifying multicast trees, then the hybrid heuristic as previously proposed would suffice.

For the hybrid we would choose the Hybrid, described in Chapter 4, as it generates efficient multicast trees and has an acceptable time complexity. The choice of a single, broadcast tree based heuristic would be sCSPT. This is because it generates fairly uniform solutions of reasonable efficiency, across the entire range of multicasts, and it has an acceptable time complexity. Although sCSPT, like all the single heuristic solutions, is prone to generating spikes of inefficent solutions, the percentage of times it does so is generally low, as illustrated in Figure 48. For small multicast group sizes, sCSPT can be more expensive than SPT in up to 7.3% of the solutions. This figure rapidly drops to less than 2% of multicasts when the group size is 20% of the network size, and is below 1% after the group size exceeds 35% of the network size. The other heuristics have not been chosen either because of their time complexity or because their performance is not sufficiently uniform across all possible multicasts.

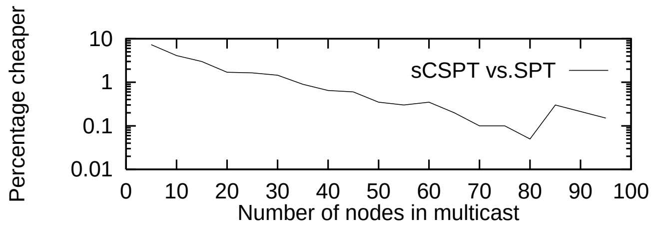

Figure 48: SPT cheaper than sCSPT

6.4 Application of the chosen Heuristics in Multicast Routing Protocols

Clearly, neither of the chosen heuristics will be applicable to routing protocols that do not use some form of link state data for route calculation. The heuristics need to know the topology of the network, even if it is aggregated in places, to calculate their solutions. For this reason implementation of either heuristic in protocols such as DVMRP is not possible. The heuristics cannot be implemented in the shared tree protocols CBT and PIM either, because both these protocols use their underlying unicast routing methods, irrespective of what types they may be, to establish a shared multicast tree. PIM and CBT do not perform any multicast tree calculation; their primary purpose is the reduction of the volume of state data and route computation in comparison to protocols such as DVMRP and MOSPF. However, PIM and CBT could provide the basic framework for shared low-cost multicast trees. Using our techniques would require the cores or RPs to calculate the paths to the receivers (once they have received the join request). Although this is a significant departure from the philosophy of PIM and CBT, we believe it is worth considering. If explicit routing from the cores/RPs were also introduced, the scaling properties of MOSPF and DVMRP would, to some extent, be avoided. The other protocols discussed in Chapter 2, with the exception of ST2+, Tag Switching and Label Swopping, are all based on link state protocols and so are candidates that, subject to any necessary modifications, might be able to use the Hybrid and sCSPT heuristics. The ST2+ protocol, Tag Switching and Label Swopping, like PIM and CBT, depend on an external unicast route calculation mechanism. However, unlike PIM and CBT, the purpose of these methods is not, primarily, to minimise the volume of state data used by multicast protocols, and so they might also be able to use an external multicast route calculation, such as the Hybrid or sCSPT heuristics.

MOSPF as a single link metric routing protocol, may reserve networks resources, if used with a protocol, such as RSVP. If the protocol were to be extended to cater for QoSR, as has been proposed by Zhang et al [70] then both of the chosen heuristics could be used within MOSPF areas. Although the heuristics could also be applied across the inter-area backbone of MOSPF Autonomous Systems, the calculation of remotely sourced multicast trees would need to be altered to use forward path link state data, rather than the reverse path data currently used. This problem could be overcome by modifying MOSPF to include "come from" link metrics in backbone advertisements, rather than relying on the OSPF backbone summary link state advertisements [42], or by means of explicit route calculation. However, the current multicast paradigm used by MOSPF is receiver initiated, in which destinations may join and leave the multicast group arbitrarilly. If the Hybrid heuristic is used under these circumstances, then the multicast tree may be continually reconfigured, which could have a serious impact on the data flow it is carrying. On the other hand, the sCSPT heuristic maintains a stable multicast tree irrespective of the multicast groups join/leave behaviour. For this reason it would be suitable as a route calculation method for a QoSR MOSPF network. The Hybrid heuristic might be used in a QoSR MOSPF network if features such as explicit routing and pinning [70] were introduced in conjuction with heavyweight multicasting.

The ST2+ protocol specifies both source initiated, destination initiated and source and destination intiated multicast tree construction. Although the protocol does not mandate the routing calculation it uses to construct paths, since it relies on an external unicast route calculation, it could do so. The support of ST2+ for both heavyweight and lightweight multicasting means that it could use a route calculation based on either the Hybrid heuristic (heavyweight, source initiated) or the sCSPT heuristic (lightweight, source or destination initiated). This argument applies to Tag Switching and Label Swopping architectures, although as these support explicit routing, and hence heavyweight multicasting, the Hybrid might be a more appropriate choice. This would be particularly so if the architectures were applied to ATM networks.

The PNNI specification does not mandate how routes through ATM networks are to be calculated, although it does illustrate how they might be by giving a sample route generation algorithm. Route calculation is multiple metric and based on the exchange of link state data, both of which are requirements for delay bound, low-cost, multicast routing algorithms. However, PNNI (and UNI) specifies that multicast connections are established by first connecting the source to a single destination and then joining additional destinations to the connection by means of an "add party" request. This mechanism is in direct conflict with the characteristics of some delay bound, low-cost, multicast routing algorithms which reconfigure multicast trees as they grow. They can only avoid this reconfiguration if all the multicast destinations are known when the multicast tree is calculated. This argument applies to B-ISDNs which also construct their multicast connections using an add party mechanism. Given the evolutionary nature of ISDN architectures neither of the heuristics proposed may, initially, be of use, particularly if B-ISDN uses dynamic two-hop alternate routing. On the other hand, if evolving ATM networks use dynamic routing for arbitrarily connected switches, and support for both heavyweight and lightweight multicasting becomes necessary (as is likely), then both the Hybrid and sCSPT heuristics provide an efficent means of calculating low-cost delay bound multicast trees. In the interim, while PNNI and B-ISDN use add party mechanisms for receivers to join multicast trees, if the group membership is known by the sending router prior to the establishment of a multicast tree, the join paths could be taken from a pre-calculated multicast tree. Such a procedure would enable the use of heuristics, such as the Hybrid, in B-ISDN and PNNI networks.

A potential application of the Hybrid in any link state network is in the construction of "permanent" multicast trees for closed user groups, where the sender wants the resources in the network to be reserved, for all receivers, even if they are not active. Receivers may join and leave the multicast as they wish, but their paths from the sender would remain intact.

The Hybrid and sCSPT heuristics could be used in a shared tree protocol, but the savings in state data, required to construct and maintain the tree, would not be as great as that achieved by PIM or CBT. The heuristics require multicast group membership link state data to be maintained by all routers that perform tree calculation, as is the case with both MOSPF and PNNI but not for CBT and PIM. However, with an appropriate explicit routing or centre based route calculation, shared trees using either the Hybrid or the sCSPT heuristics need not require as much state data to be used as is required by MOSPF or DVMRP. Where either of the heuristics is used in a shared tree protocol, the arbitrary delay bound has to span the diameter of the multicast group rather than the height, as is the case for source based multicast trees. This is because all user data goes via the centre of the tree. Some evaluation work has been carried out to assess the savings that the CCET heuristic used in a shared tree might have over CBT and this is documented in [46]. This work remains to be carried out for the Hybrid and sCSPT.

In summary, the Hybrid heuristic is more applicable to heavyweight multicasting protocols that support closed multicast user groups. The sCSPT heuristic is better suited to lightweight multicasting protocols, that support open multicast user groups.

Chapter 7

Lower Cost Multicast Routing in the Internet

7.1 Introduction

The main theme of this work has been the evaluation of heuristics that calculate delay bound, low cost, multicast trees. That is, we have examined the calculation of multicast trees in networks where each link has two metrics. Here, in a brief aside, we investigate the application of the same heuristics to calculate multicast trees in networks that use only one metric per link for route calculation.

In their comparison of multicast trees and algorithms Wei and Estrin [67] judge the quality of multicast trees according to three criteria

1. Low delay.
2. Low cost.
    - Cost of total bandwidth consumption.
    - Cost of tree state information.
3. Light traffic concentration.

Low delay in a multicast tree is generally accomplished at the expense of tree cost, whereas low cost trees are generally accomplished at the expense of delay. Traffic concentration in a multicast tree depends on the number of sources that send data across the tree. For example, if source based multicast trees are used, there is one multicast tree per sender. Traffic will only be concentrated at nodes that are common to two or more of the multicast trees established for the communication. Minimal cost trees, such as the MST we have used as our evaluation benchmark ([6] and [27]), and shared trees, such as CBT [5] and PIM [19] provide a common tree for all senders and are, therefore, vulnerable to traffic concentration, particularly at their centres.

Previously we have described heuristics that calculate low cost, multicast trees that are constrained by a bounded delay. These heuristics were designed to meet the needs of multicast routing in future high speed networks, such as ATM/B-ISDN, where link costs and delays may be independent metrics. Further, the heuristics evaluated calculate source based trees, not shared trees, so they do not suffer the traffic concentrations alluded to here.

Current Internet multicasting protocols use either link-state [41] or distance vector [61] routing to construct multicast trees that are based on a single metric, such as delay. Tree cost is ignored by the route calculation, but is reduced by route aggregation.

If the constraint of a single link metric is applied to some of the heuristics we have evaluated their performance varies dramatically. Under these circumstances the CSPT, sCSPT and CCPT heuristics will attempt to minimise the additive value along each path in the multicast tree and so calculate a shortest path tree, based on the link metric. They will effectively ignore the total additive value of the edges in the multicast tree. This characteristic renders these heuristics to be of no benefit in networks that use a single metric per link for route calculation. Both the CCET and CST_c heuristics will attempt to minimise the additive value of the multicast tree while maintaining an upper bound on the additive value of the metric along any path in the multicast tree. The time complexity of the CST_c heuristic, however, precludes its use in current routing protocols.

Evaluation of the CCET heuristic, in a modified form that uses only a single edge metric, indicates that it may provide lower cost multicast trees than current source based shortest path multicast trees, in networks that have only a single metric per link. With further adaptation this modified heuristic could be integrated into a distributed link-state protocol, such as MOSPF [41], for use in connectionless networks.

To further constrain the modified CCET heuristic, such that its delay performance is similar to that of the current Internet multicast tree route calculations, we bound the multicast tree by the additive value along the path to the multicast node furthest from the multicast source.

7.2 Defining the bounded delay, minimum cost multicast routing problem

In our definition of the bounded delay, minimum cost multicast routing problem [63], we assume that all links in a network have independent delay and cost metrics. In this definition we use a single edge metric, which can represent delay or cost or any other link metric. For convienience we use the link delay metric.

- Given a connected graph $G = \langle V, E \rangle$ where V is the set of its vertices, E the set of its edges, and $d(i,j)$ the delay along each edge $(i,j) \in E$.
- Find the tree $T = \langle V_T, E_T \rangle$, where $T \subseteq G$, joining the vertices s and $\{M_k\}_{k=1}^{n} \in V$ such that $\sum_{(i,j) \in E_T} d(i,j)$ is minimised and $\{D(s, M_k) \leq \Delta; k = 1,..,n\}$ where Δ is the delay bound and $D(s, M_k) = \sum_{(i,j)} d(i,j)$ for all (i,j) on the path from s to M_k in T.

Note that, if the delay is unimportant, the problem reduces to the Steiner tree problem. The addition of the finite delay bound makes the problem harder although it is still NP-complete.

7.3 Modified CCET (mCCET)

CCET was first published in [63] along with some simple preliminary evaluations. The original heuristic uses two link metric functions, $d(i,j)$ and $c(i,j)$. This variant, mCCET [12], assumes there is only one metric function, $d(i,j)$.

1. Use an extended form of Dijkstra's shortest path algorithm, to find, for each $v \in V - \{s\}$ the minimum delay, dbv, from s to v. As the algorithm progresses, keep a record of all the dbv found so far, and build a matrix $Delay$ such that $Delay(k, v)$ is the sum of the delays on edges in a path from s to v, whose final

edge is (k, v). When the algorithm is complete, the dbv to the multicast node furthest from the multicast source is the multicast delay bound dbM.

2. Set all elements in $Delay(k, v)$ that are greater than dbM to ∞. The matrix $Delay$ then represents the edges of a directed graph derived from G which contains all possible solutions to a multicast tree rooted at s which satisfy the delay constraint (i.e., $\Delta = dbM$).

3. Now construct the multicast tree T. Start by setting $T = \langle \{s\}, \emptyset \rangle$.

4. Take $v \ni V_T$, with the largest dbv and join this to T. Where there is a choice of paths which still offer a solution within the delay bound, choose at each stage the lowest delay edge leading to a connection to the tree. This step involves a depth first search to find a path to the multicast source that does not exceed the delay bound dbM. Such a path will always exist, although this step may require backtracking to find it. (Our evaluation of the heuristics indicates that this is not, generally, a problem).

5. Include in E_T all the edges on the path (s, v) not already in E_T and include in V_T all the nodes on the path (s, v) not already in V_T.

6. Repeat steps 4 and 5 until $V_T = V$, when the multicast tree will have been built.

7. Prune any unnecessary branches of the tree beyond the multicast recipients.

7.3.1 A worked example

To illustrate the working of mCCET, we take the graph shown in Figure 49. The bracketed parameters for each link indicate the associated cost. The example finds the multicast route from source G to destinations A, B, C, D and F.

The application of the extended form of Dijkstra's algorithm results in the directed graph shown in Figure 50 where parameters shown against each link represent the path cost from the source, G, to the node at the end of that link and the link cost. The multicast bound, dbM, is the cost to the multicast node furthest from the source, which is node D with cost 56.

Figure 51 illustrates the bounded directed graph, where all edges with a path cost greater than the cost bound dbM have been removed.

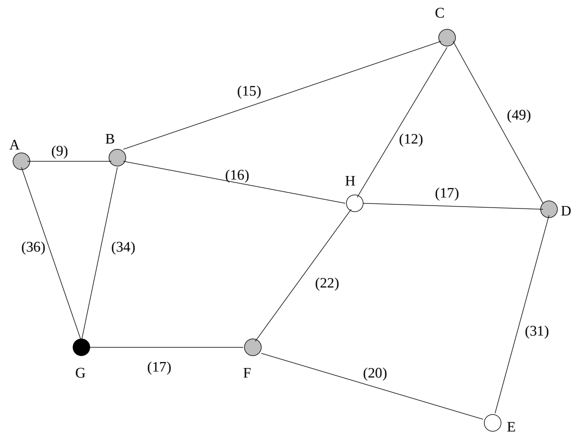

Figure 49: Sample graph to illustrate mCCET

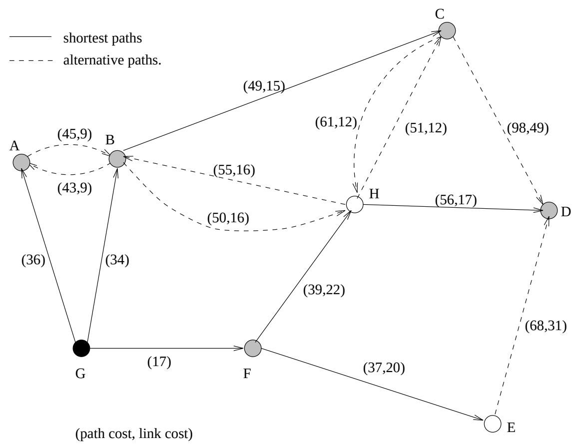

Figure 50: Directed graph of paths within the delay bound

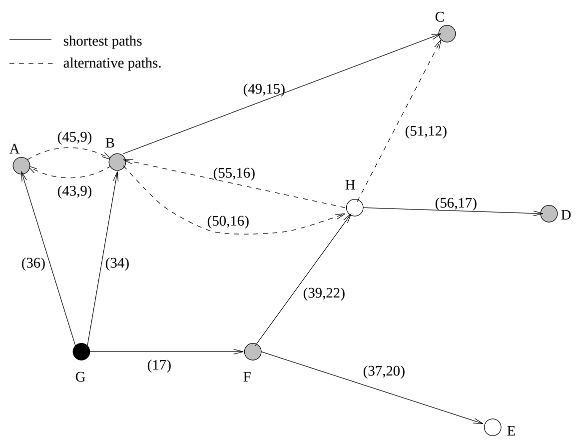

Figure 51: Directed graph of paths within the delay bound

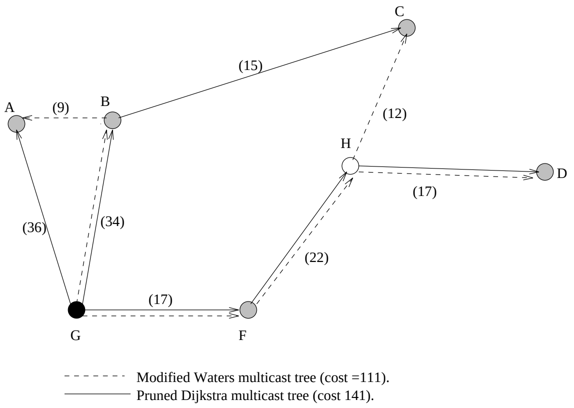

Figure 52: Multicast tree

Starting with the shortest path node furthest from the source, node D, choose the cheapest link back to the source that remains within the cost bound. There is only one path, to node H. From node H choose the cheapest link back to the source that remains within the cost bound minus the cost of link HD. This is link HF. Repeat this process until the source is reached. The first path constructed will always be a shortest path. Nodes D, H and F are now in the multicast tree. The next furthest node on a shortest path from the source is node C. This is connected to node H without violating the cost bound dbM. In this manner nodes E, A and B are added to the multicast tree. Finally, the edge FE is pruned to give the multicast tree of Figure 52 whose total cost is 111 units and has a cost bound of 56 (to D).
The tree produced by the standard use of Dijkstra's algorithm and then pruned would result in a solution with a cost of 141. This simple example shows how, with a single edge metric, the mCCET heuristic is able to find aggregated paths between the multicast source and its destinations.

7.3.2 Time complexity mCCET

The mCCET heuristic has the same time complexity as the CCET heuristic, and is vulnerable to the same pathological behaviour (see Section 3.3.2 and Section 3.3.3).

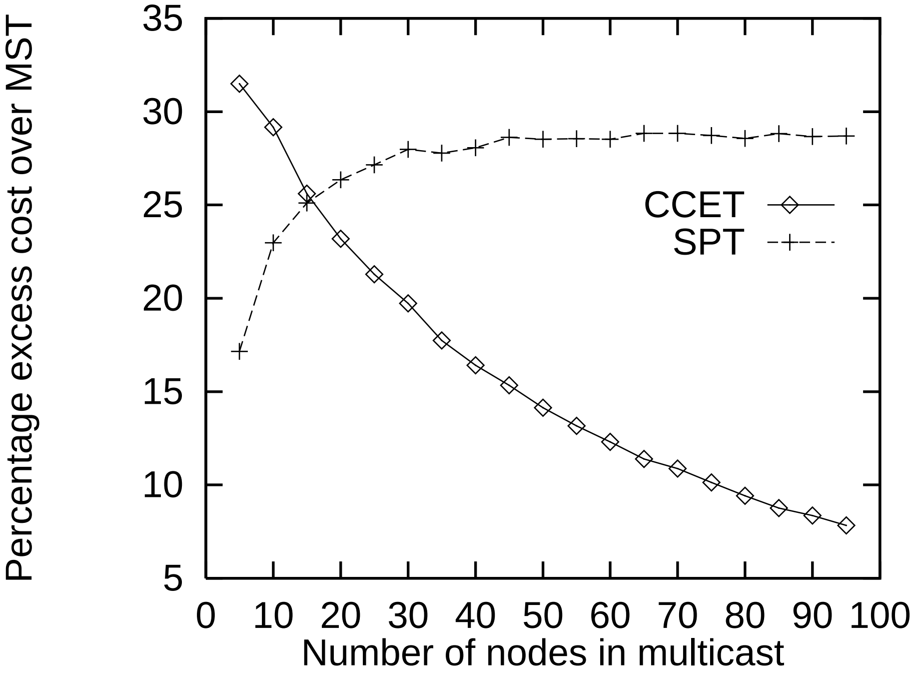

Figure 53: Multicast tree cost of mCCET and pruned Dijkstra

7.4 Evaluation of the mCCET heuristic

The network model used in the evaluation of the mCCET heuristic is attributed to Waxman [66], and is described in Section 3.6.2. Two hundred networks of 100 nodes each were used to obtain the evaluation results presented in Figure 53. The cost of multicast trees constructed using both mCCET and Dijkstra's SPT algorithm are compared with the cost of multicast trees built using the benchmark MST algorithm (see Section 3.6.1). We have not presented an extensive evaluation of the mCCET heuristic because its behaviour is similar to its parent heuristic, CCET. Rather, we only wish to illustrate how the heuristic behaves when it uses only a single metric for each link.

For small multicast groups mCCET produces, on average, more expensive multicast trees than Dijkstra's SPT algorithm, while for larger multicasts the heuristic achieves much lower cost trees than Dijkstra. A mCCET solution is generally cheaper than that found by Dijkstra's because it has a choice of paths back to the source that are within the multicast delay bound. Dijkstra's algorithm only has the shortest path from each node back to the source. As the size of the multicast group increases so does the number of path choices available for mCCET to select, hence the improved performance.

7.5 Adaptation of mCCET to Distributed Routing Protocols

For any network, Dijkstra's SPT algorithm may calculate one of several different shortest path trees routed at the same source node, depending on the order in which links are processed. This happens when there is more than one shortest path between the root and any destination node. In distributed unicast routing protocols such as OSPF [42], where every router calculates a shortest path tree rooted at itself, this is not a problem. Nor is it a problem for source based multicast routing. In distributed multicast routing protocols this attribute of Dijkstra's SPT algorithm is a problem because every router in the network must calculate exactly the same shortest path tree for each source/multicast group in order to create consistent routing tables throughout the network.

The Multicast Extension to OSPF [41] is a distributed routing protocol which uses Dijkstra's SPT algorithm in the "on-demand" construction of multicast routing tables. To ensure that all routers calculate exactly the same shortest path tree for a source/multicast group, MOSPF employs a tie-breaker mechanism to select links for inclusion in the tree. For mCCET to be used in a distributed multicast routing protocol it must be constrained in much the same manner as Dijkstra's algorithm is in MOSPF.

In addition to the link-state data and multicast group membership data required to calculate a multicast tree in a distributed routing protocol (see Section 2.3.2), calculation of forwarding paths by routers using the mCCET heuristic requires each router in the network to have one, additional, primary data structure. This structure, which is illustrated in Figure 54, we call the Tree Structure, and is initially empty. It has eight fields for each entry it may contain, which are

- Parent; a router at a link source.
- Child; a router at a link sink.
- Link cost; the value of the metric associated with the link between the Parent and Child routers.
- Path cost; the sum of the values of the metric on a path between the multicast source router and the Child router.

- SPT; A number that indicates the postion of a Parent/Child link in the shortest path tree between the multicast source router and the Child router.
- T; a flag which is set if the Parent/Child link is in the multicast tree.
- Assocaited Interface (AI); the downstream interface a router will forward incoming multicast data to.
- MC; a flag which is set if the Child router is in the multicast group.

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC

Figure 54: Tree Structure

In the following description of how the Tree Structure would be used by a router calculating forwarding paths for multicast data, we use the example from Section 7.3.1. The calculating router is F (see Figure 49). The multicast source is node G, and the multicast destinations are nodes A,B,C,D and F.

1. Initialise the Tree Structure to contain all the edges from the multicast source router, G, to its neighbours (Figure 55). None of the links included in the Tree Structure have yet been added to the multicast tree, so the T field remains blank for all entries. This also applies to the associated interface (AI) field.

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	-	-	-	X
G	B	34	34	-	-	-	X
G	F	17	17	-	-	-	-

Figure 55: Initial state of the Tree Structure

2. Set the link with the lowest Path Cost in the Tree Structure to be the first link in the shortest path tree (SPT) (Figure 56).

3. Add all the links from the Child router of the last link added to the SPT to the Tree Structure, except the link leading back from the Child to its Parent. In this example, add all the edges from the router F to its neighbours, except the link

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	-	-	-	X
G	B	34	34	-	-	-	X
G	F	17	17	1	-	-	-

Figure 56: Set link GF to be the first in the SPT

FG. The Path Cost for the new links added to the Tree Structure are the sum of the link costs along the path from the multicast source router, G, to the Child router of the link being added (Figure 57).

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	-	-	-	X
G	B	34	34	-	-	-	X
G	F	17	17	1	-	-	-
F	H	39	22	-	-	-	-
F	E	37	20	-	-	-	-

Figure 57: Update the Tree Structure with next forward links

4. Set the link with the lowest Path Cost in the Tree Structure, that is not already in the SPT, to be the next link in the shortest path tree (SPT). In this example the link GB with a Path Cost of 34 is added as the 2nd link in the SPT (Figure 58).

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	-	-	-	X
G	B	34	34	2	-	-	X
G	F	17	17	1	-	-	-
F	H	39	22	-	-	-	-
F	E	37	20	-	-	-	-

Figure 58: Update the Tree Structure with next forward links

5. Repeat steps 3 and 4 until all the routers in the network are in the SPT. The Tree Structure finally becomes as illustrated in Figure 59.

6. The bound for the multicast tree is the highest Path Cost along a shortest path to a router that is a multicast destination. From Figure 59 we see that router D has

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	3	-	-	X
G	B	34	34	2	-	-	X
G	F	17	17	1	-	-	X
F	H	39	22	5	-	-	-
F	E	37	20	4	-	-	-
B	A	43	9	-	-	-	X
B	C	49	15	6	-	-	X
B	H	50	16	-	-	-	-
A	B	45	9	-	-	-	X
E	D	68	31	-	-	-	X
H	D	56	17	7	-	-	X
H	C	51	12	-	-	-	X
H	B	55	16	-	-	-	X
C	D	98	49	-	-	-	X
C	H	61	12	-	-	-	-

Figure 59: Tree Structure with SPT and alternative links

the highest Path Cost (56) along a shortest path (SPT indicator is set). Remove all entries from the Tree Structure that have a Path Cost greater than 56 (Figure 60).

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	3	-	-	X
G	B	34	34	2	-	-	X
G	F	17	17	1	-	-	X
F	H	39	22	5	-	-	-
F	E	37	20	4	-	-	-
B	A	43	9	-	-	-	X
B	C	49	15	6	-	-	X
B	H	50	16	-	-	-	-
A	B	45	9	-	-	-	X
H	D	56	17	7	-	-	X
H	C	51	12	-	-	-	X
H	B	55	16	-	-	-	X

Figure 60: Pruned Tree Structure with SPT and alternative links

7. Set the current node, CN, to the multicast destination furthest from the multicast source along the shortest path tree, that is not already in the multicast tree. From Figure 60, router D is in the SPT and has the largest Path Cost from the multicast

source, CN = D.

8. Initialise a Branch Table to contain the link, from the Tree Structure, with the lowest Link Cost from the router in CN to one of its parents (Figure 61). In this table the Path Cost is the cost of the path from the Child back towards the multicast source, not the Path Cost taken from the Tree Structure.

Parent	Child	Path Cost	Link Cost
H	D	17	17

Figure 61: Initial Branch Structure

9. If the Path Cost of the last entry in the Branch Structure is less than the bound then set the next router, NR, to the Parent of the last entry. In this example the bound is 56, so NR = H.

10. Add the link, from the Tree Structure, with the lowest link cost from the router in NR to one of its parents, to the Branch Structure.

Parent	Child	Path Cost	Link Cost
H	D	17	17
B	H	33	16

Figure 62: Branch Structure with two links

11. Repeat steps 9 and 10 until either the multicast source router is reached, or a link already marked as being in the multicast tree is reached (Tree Structure entry field T = X) or the Path Cost in the Branch Structure exceeds the bound (56, in this example). If the Path Cost in the Branch Structure exceeds the bound the path must be unwound and an alternative path back to the source must be found. As a path is unwound, its links are removed from the Branch Structure. In this example the bound is broken when searching for a path from router D back to the source, G, (Figure 63).

    The path has to be unwound as far as router H before an alternative is found. Figure 64 illustrates the final path from D to G. (Note that this is a reverse path.

Parent	Child	Path Cost	Link Cost
H	D	17	17
B	H	33	16
A	B	42	9
G	A	80	36

Figure 63: Path Cost breaks bound

The actual path, and its path cost, is from the multicast source router, G, to the multicast destination router, D.)

Parent	Child	Path Cost	Link Cost
H	D	17	17
F	H	39	22
G	f	17	56

Figure 64: Final path between router G and D

12. Set the multicast indicator, MI, to false.

13. Starting with the first entry in the Branch Structure, mark the corresponding entry in the Tree Structure as being in-tree (T = X). If the Child router in the Branch Structure is a multicast destination set the multicast indicator, MI, to true. If the Parent router for the entry is also the calculating router, and the Multicast Indicator is set (MI == true), set the Associated Interface in the Tree Structure, for the link, to the Child router. Repeat this procedure for all the entries in the Branch Structure. In this example the corresponding entries of the Tree Structure will be as illustrated in Figure 65.

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
..	..	..	..	..	..	..	..
G	F	17	17	1	X	-	X
F	H	39	22	5	X	H	-
..	..	..	..	..	..	..	..
H	D	56	17	7	X	-	X
..	..	..	..	..	..	..	..

Figure 65: Path GD set in-tree in the Tree Structure

14. Once a branch has been added to the Tree Structure delete all entries from the Branch Structure, set CN to the router furthest from the multicast source router, along the shortest path, that is not already marked in-tree and repeat steps 9 to 13 until all the multicast routers in the Tree Structure have been marked as in-tree (each has the field T = X). The final Tree Structure for this example will be as illustrated in Figure 66.

Parent	Child	Path Cost	Link Cost	SPT	T	AI	MC
G	A	36	36	3	X	-	X
G	B	34	34	2	X	-	X
G	F	17	17	1	X	-	X
F	H	39	22	5	X	H	-
F	E	37	20	4	-	-	-
B	A	43	9	-	-	-	X
B	C	49	15	6	X	-	X
B	H	50	16	-	-	-	-
A	B	45	9	-	-	-	X
H	D	56	17	7	X	-	X
H	C	51	12	-	-	-	X
H	B	55	16	-	-	-	X

Figure 66: Final Tree Structure

15. In this example there is only one forwarding associated interface for the calculating router, F. This is the interface to router H. The multicast forwarding table for the multicast group is set according to the values of the AI field in the Tree Structure entries.

7.6 Conclusions

The CCET heuristic was not designed for use in connectionless networks that have only single metric links. Rather, it was designed to minimise the cost of multicast delivery trees that are delay bound in connection oriented networks.

Our evaluation work has shown that by making a simple modification to the CCET heuristic, the cost of dense multicast trees in single metric networks can be reduced, without increasing the cost of any path beyond that to the most distant multicast destination. In circumstances where the modified heuristic generates sparse multicast trees that are more expensive than those built using Dijkstra's algorithm, the protocol can simply fall back to the pruned Dijkstra SPT, since this will have already been calculated.

The cost saving of mCCET multicast trees is significant enough for it to be considered as an alternative to Dijkstra's SPT algorithm in appropriate multicast routing protocols, such as MOSPF [41].

The problem of scaling, and hence tree construction costs, remains with this approach as it does with MOSPF and DVMRP, making it unsuitable for sparse multicasting. As the results presented here indicate, mCCET generates greater cost savings as the multicast tree size grows, making it ideal for dense multicasting [19].

The directed graph calculated in the first stage of the algorithm contains alternative paths within the cost bound which could be used for load sharing, although there will be an increase in control traffic to do this.

Because the CCET heuristic was designed for networks with two metrics per link, the modified version offers an inherent evolutionary path from single metric to two metric multicast routing. Such a requirement might arise where there is a need to minimise the bandwidth usage of a multicast tree while keeping a bound on the maximum delay to any recipient. Such a requirement could be specified by combining IP Types of Service in a multicast delivery request (a feature removed by [3]). It would also mean that network routers would have to exchange two metrics per link to construct routing tables for the network.

Chapter 8

Conclusions and Further Research

In this work we have analysed the behaviour and evaluated the performance and time complexity of several heuristics that have been proposed to calculate low-cost multicast trees that are bound by an arbitrary delay. The evaluation used flat networks, based on a model proposed by Waxman and improved by Doar, and networks constructed from clusters interconnected via backbone networks, developed by ourselves, from the the Waxman/Doar model. The networks have been both densely and sparsely connected, large and small. The multicast groups used range in size from small to large, randomly distributed across the networks. We chose the network diameter as our primary arbitrary delay bound.

We have compared the average excess cost performance of the heuristics evaluated against a benchmark, MST. We have also compared the performance for individual multicasts against each other, for each heuristic. The variance in the efficiency of the heuristics solutions has also been examined. Finally, we have assessed the efficiency of the heuristics as network load increases, and how often and how much the topology of multicast trees changes as they grow. We have identifed problems of time complexity and performance variability in the heuristics evaluated.

By combining appropriate heuristics we propose a hybrid that produces efficient solutions within an acceptable order of time complexity, for all multicast group sizes.

Our evaluations indicate that this hybrid performs well for both single cluster and hierarchical networks. We have also shown, in our study of path delay distribution, that less buffering will be needed in the destinations using the Hybrid than using Dijkstra's algorithm. An important result of this work, and a departure from current routing solutions, is the integration of several heuristics which are individually prone to occasional results of very high inefficiency (as might be expected in an heuristic approach) into a hybrid that generates efficent solutions for all multicast groups.

The Hybrid, in common with other heuristics that provide very efficient multicast solutions (e.g., CSTc) will sometimes reconfigure a multicast tree as group membership changes. This characteristic may not be acceptable to applications that have dynamic multicast group membership. For large dynamic multicast groups the CCET heuristic provides very efficient solutions that can be calculated with an acceptable order of time complexity. In order to provide a reasonably efficient heuristic for all sizes of dynamic multicast groups we constrain the CSPT heuristic to obtain the sCSPT heuristic; this new heuristic will not reconfigure multicast trees as members join or leave the multicast group. Note that the solutions of sCSPT for large dynamic multicast groups are not as efficient as those provided by the CCET heuristic.

Both the Hybrid and sCSPT heuristics use metrics for every link in a network to perform route calculation making them amenable for implementation in link-state routing protocols such as the Internet's MOSPF or that used by the ATM Forum's PNNI.

In assessing how the heuristics can be applied, we have identified two classes of multicast types; heavyweight and lightweight. We have also examined existing and proposed multicasting protocols in order to identify where the Hybrid and sCSPT heuristics can be best applied. The Hybrid heuristic is applicable to heavyweight multicasting for closed user groups in robust networks, whereas the sCSPT heuristic is suitable for lightweight multicasting for open groups, in less robust networks. With appropriate extensions, the Hybrid is more suited to architectures such as PNNI (and, possibly, B-ISDN and ST2+) and the emerging Internet tag switching and label swapping architectures. The sCPST heuristic is applicable to most multiple metric, link state, routing protocols, including the potential QoSR MOSPF.

We have also considered how the heuristics could be used in shared tree routing protocols and consider this a viable application area.

As an aside, we examined the performance of the original heuristics in networks that use only a single metric for route calculation. We concluded from this evaluation work that a modified version of the CCET heuristic (mCCET) might provide lower cost multicast trees when applied to routing protocols such as the Internet's MOSPF.

8.1 Further Research Issues

The work to date leads into the more practical issues of implementing the heuristics in real networks, such as QoSR MOSPF etc. This work requires further study as QoSR protocols are evolved through the working groups of the IETF and the ATM Forum.

The application of the heuristics to shared trees is a second area which needs more practical work. While the use of the heuristics in protocols such as CBT and PIM is precluded, these protocols provide a basic framework from which low-cost, delay bound, shared trees might evolve. There is a need, in this area, to study how the heuristics might construct shared trees with multiple centres. Other application areas that may benefit from this work are the calculation of load sharing paths in networks.

An area of continued research is the study of join/leave mechanisms for very efficient multicast heuristics, such as the Hybrid, that will minimise tree reconfiguration. These methods may take non-deterministic approachs, as illustrated by Salmon [53].

Bibliography

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data structures and algorithms.* Addison Wesley, Reading, Mass., 1987.

[2] A. Alles. *ATM Interworking.* Cisco Systems, Inc. White Papers, http://cio.cisco.com:80/warp/public/614/12.html, May 1995.

[3] P Almquist. *Type of Service in the Internet Protocol Suite, RFC1349.* Internet Engineering Task Force, http://www.ietf.org, July 1992.

[4] A.J. Ballardie. *A New Approach to Multicast Communication in a Datagram Internetwork.* PhD thesis, University College London, 1995.

[5] A.J. Ballardie, P.F. Francis, and J. Crowcroft. Core based trees. *Computer Communications Review*, 23(4):85–95, 1993.

[6] K. Bharath-Kumar and J.M. Jaffe. Routing to Multiple Destinations in Computer Networks. *IEEE Transactions on Communications*, COM-31(3):343–351, March 1983.

[7] R. Bolla, P. Castelli, F. Davoli, and M. Marchese. Dynamic Distributed Two-hop Alternate Routing in ATM Networks. In Demetres Kouvatsos, editor, *ATM'97 Fifth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, pages 59/1–59/10, University of Bradford, 21st-23rd July 1997. UK Performance Engineering Workshop 1997.

[8] CCITT. *Broadband Aspects of ISDN, Recommendation I.121.* International Telecommunications Union, Geneva, 1991.

[9] CIP Working Group, editor C.Topolcic. *Experimental Internet Stream Protocol, Version 2 (ST-II), RFC1190.* Internet Engineering Task Force, http://www.ietf.org, October 1990.

[10] D.D Clark. The Design Philosophy of the DARPA Internet Protocols. *Computer Communication Review SIGCOMM*, pages 106–114, 16th - 19th August 1988.

[11] J.S. Crawford. Multicast Routing: Evaluation of a New Heuristic. Master's thesis, University of Kent at Canterbury, 1994.

[12] J.S. Crawford and A.G. Waters. An Heuristic for Lower Cost Multicast Routing in the Internet. Technical Report 25-96, University of Kent at Canterbury, June 1996.

[13] J.S. Crawford and A.G. Waters. A Hybrid Approach to Quality of Service Multicast Routing. In Demetres Kouvatsos, editor, *ATM'97 Fifth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, pages 61/1–61/9, University of Bradford, 21st-23rd July 1997. UK Performance Engineering Workshop 1997.

[14] J.S. Crawford and A.G. Waters. Low Cost Quality of Service Multicast Routing in High Speed Networks. Technical Report 13-97, University of Kent at Canterbury, December 1997.

[15] Y.K. Dalal and R.M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communictaions of the ACM*, 21(12):1040–1048, December 1978.

[16] D.W. Davies, D.L.A. Barber, W.L. Price, and C .M. Solomonides. *Computer Networks and their Protocols.* John Wiley and Sons, New York, 1981.

[17] S. Deering. *Host Extensions for IP Multicasting, RFC 1112.* Internet Engineering Task Force, http://www.ietf.org, August 1989.

[18] S.E. Deering and D.R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[19] S.E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G. Liu, and L. Wei. An Architecture for Wide Area Multicast Routing. *Computer Communications Review*, 24(4):126–135, October 1994.

[20] L. Delgrossi, C. Halstrick, R.G. Herrtwich, and H. Stuttgen. HeiTP - A Transport Protocol for ST-II. In *GLOBECOM'92*, pages 1369–1373, Orlando, Florida, 6th-9th December 1992. IEEE Computer Society.

[21] J.M.S. Doar. Multicast in the Asynchronous Transfer Mode Environment. Technical Report No. 298, University of Cambridge Computing Laboratory, April 1993.

[22] M. Doar and I. Leslie. How Bad is Naive Multicast Routing. In *Proceedings of IEEE INFOCOM'93*, volume 1, pages 82–89. IEEE Computer Society Press, 28th March - 1st April 1993.

[23] Alan Dolan and Joan Aldous. *Networks and Algorithms, An Introductory Approach.* John Wiley and Sons, New York, 1995.

[24] W. Fenner. *Internet Group Management Protocol, Version 2, RFC2236.* Internet Engineering Task Force, http://www.ietf.org, November 1997.

[25] R.W. Floyd. Algorithm 97: Shortest Path. *Communications of the ACM*, 5(6):345, 1962.

[26] Alan Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, Cambridge, 1989.

[27] E.N. Gilbert and H.O. Pollack. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[28] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. *QoS Routing Mechanisms and OSPF Extensions, draft-guerin-qos-routing-ospf-01.txt.* Internet Engineering Task Force, http://www.ietf.org/internet-drafts/, March 1997.

[29] S.E. Hardcastle-Kille. *X.500 and Domains, RFC1279.* Internet Engineering Task Force, http://www.ietf.org, 1991.

[30] V. Hardman, M.A. Sasse, and I. Kouvelas. Successful Multiparty Audio Communication over the Internet. *Communications of the ACM*, 41(5):74–80, May 1998.

[31] C. Hedrick. *Routing Information Protocol, RFC1058.* Internet Engineering Task Force, http://www.ietf.org, 1988.

[32] M. Hyman, A.A. Lazar, and G. Pacifici. Joint Scheduling and Admission Control for ATS-based Switching Nodes. *Computer Communication Review SIGCOMM*, 22(4), 17th - 20th August 1992.

[33] ITU-T Study Group 11. Reccommendation Q.2931 (02/95) - B-ISDN Integrated Services Digital Network (B-ISDN) - Digital Subscriber Signalling System No. 2 (DSS 2) - User-Network Interface Layer 3 Specification for Basic Call/Connection Control.

[34] ITU-T Study Group 11. Reccommendation Q.2971 (10/95) - B-ISDN Integrated Services Digital Network (B-ISDN) - Digital Subscriber Signalling System No. 2 (DSS 2) - User-Network Interface Layer 3 Specification for Point-to-Multipoint Call/Connection Control.

[35] S. Jamin, S. Shenker, L. Zhang, and D. Clark. Admission Control Algorithms for Predictive Real-Time Service. In *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video.*, November 1992.

[36] R.M. Karp. *Reducibility among combinatorial problems.* Plennum Press, New York, 1972.

[37] V.P. Kompella. *Multicast Routing Algorithms for Multimedia Traffic.* PhD thesis, University of California, San Diego, USA, 1993.

[38] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Multicast Routing for Multimedia Communications. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.

[39] T Kuzminski. Alternatives for Multicast Routing in ATM Networks. Master's thesis, University of Kent at Canterbury, 1996.

[40] J.M. McQuillan, I. Richer, and E.C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[41] J. Moy. *Multicast Extensions to OSPF. RFC 1584.* Internet Engineering Task Force, http://www.ietf.org, March 1994.

[42] J. Moy. *OSPF Version 2. RFC1583.* Internet Engineering Task Force, http://www.ietf.org, March 1994.

[43] J. Moy. *OSPF Version 2. RFC2178.* Internet Engineering Task Force, http://www.ietf.org, July 1997.

[44] J. Moy. Private e-mail correspondence, 1997.

[45] D. Oran. *OSI IS-IS Intra-Domain Routing Protocol, RFC1142.* Internet Engineering Task Force, http://www.ietf.org, February 1990.

[46] S. Parkinson. Multicast Routing in the Internet: Evaluating Proposed Routing Mechanisms. Master's thesis, University of Kent at Canterbury, 1995.

[47] C Partridge. *A Proposed Flow Specification, RFC1363.* Internet Engineering Task Force, http://www.ietf.org, September 1992.

[48] T Pusateri. *Distance Vector Multicast Routing Protocol - draft-ietf-idmr-dvmrp-v3-07.* Internet Engineering Task Force, http://www.ietf.org/internet-drafts/, August 1998.

[49] Y. Rekhter, B. Davie, E. Rosen, and G. Swallow. *Cisco Systems' Tag Switching Architecture Overview, RFC2105.* Internet Engineering Task Force, http://www.ietf.org, February 1997.

[50] E.C. Rosen, A. Viswanathan, and R. Callon. *A Proposed Architecture for MPLS. draft-ietf-mpls-arch-00.txt.* Internet Engineering Task Force, http://www.ietf.org/internet-drafts/, August 1997.

[51] K.W. Ross. *Multiservice Loss Models for Broadband Telecommunications Networks.* Springer-Verlag, London, 1995.

[52] H.F. Salama, D.S. Reeves, I. Vinitos, and T-L. Sheu. Evaluation of Multicast Routing Algorithms for Real-Time Communications on High-Speed Networks. In *Proceedings of the 6th IFIP Conference on High-Performance Networks (HPN'95)*, pages 27–42, Palma de Mallorca, Spain, 1995.

[53] M Salmon. Survivable Multicast Routing for ATM Networks. Master's thesis, University of Kent at Canterbury, 1997.

[54] ST2 Working Group. *Internet Stream Protocol, Version 2 (ST2) Protocol Specification - Version ST2+, RFC1819.* Internet Engineering Task Force, http://www.ietf.org, August 1995.

[55] W. Stallings. *ISDN and Broadband ISDN.* MacMillan Publishing Company, Oxford, 2nd edition, 1992.

[56] Q. Sun and H. Langendoerfer. Efficient Multicast Routing for Delay-Sensitive Applications. In *Second International Workshop on Protocols for Multimedia Systems (PROMS'95)*, pages 452–458, Saltzburg, Austria, 9th-12th October 1995.

[57] A.S Tanenbaum. *Computer Networks.* Prentice-Hall International, London, 1989.

[58] The ATM Forum Technical Committee. *ATM User-Network Interface Specification, Version 3.1.* The ATM Forum, http://www.atmforum.com, September 1994.

[59] The ATM Forum Technical Committee. *ATM User-Network Interface Specification, Version 4.* The ATM Forum, July 1996.

[60] The ATM Forum Technical Committee. *Private Network-Network Interface Specification Version 1.0.* The ATM Forum, http://www.atmforum.com, March 1996.

[61] D. Waitzman, C. Partridge, and S.E. Deering. *Distance Vector Multicast Routing Protocol, RFC1075.* Internet Engineering Task Force, http://www.ietf.org, 1988.

[62] D. Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Stanford University, 1980.

[63] A.G. Waters. A New Heuristic for ATM Multicast Routing. In *2nd IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, pages 8/1–8/9, July 1994.

[64] A.G. Waters. *Multi-Party Communication over Packet Networks.* PhD thesis, University of Essex, UK, 1996.

[65] A.G. Waters and J.S. Crawford. Low-cost ATM Multimedia Routing with Constrained Delays. In *Third COST 237 Workshop on Multimedia Telecommunications and Applications*, November 1996.

[66] B.M. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[67] L. Wei and D. Estrin. The Trade-offs of Multicast Trees and Algorithms. Technical Report 93-560, Computer Science Department, University of Southern California, 1993.

[68] R. Widyono. The Design and Evaluation of Routing Algorithms for Real-time Channels. TR-94-024, University of California at Berkeley and International Computer Science Institute, September 1994.

[69] L. Zhang, S.E. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 8(17), September 1993.

[70] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley. *Quality of Service Extensions to OSPF, draft-zhang-qos-ospf-00.ps.* Internet Engineering Task Force, http://www.ietf.org/internet-drafts/, June 1996.

[71] W. Zheng and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, September 1996.

[72] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting. In *Proceedings of IEEE INFOCOM'95*, pages 377–385, Boston, Massachusettes, 2nd-6th April 1995. IEEE Computer Society Press.