Heuristics for ATM Multicast Routing

John Crawford and Gill Waters

Computing Laboratory
University of Kent at Canterbury
CT2 7NF, UK

e-mail: J.S.Crawford@ukc.ac.uk and A.G.Waters@ukc.ac.uk

Abstract

Several multicast routing heuristics have been proposed to support multimedia services, both interactive and distribution, in high speed networks such as B-ISDN/ATM. Since such services may have large numbers of members and have real-time constraints, the objective of the heuristics is to minimise the 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, although some perform better for small multicast groups and others perform better for larger groups.

We present an introduction to the problem and to some key heuristic solutions. Our detailed analysis and evaluation of some of these heuristics illustrates that in some situations their average performance is reversed; a heuristic that in general produces efficient solutions for small multicasts may sometimes produce a more efficient solution for a particular large multicast/network combination. Also, in a limited number of cases using Dijkstra's algorithm produces the best result. We conclude that the specific efficiency of a heuristics solution depends on the topology of both the network and the multicast, and that it is difficult to predict.

Because of this unpredictability we propose the integration of two heuristics with Dijkstra's shortest path tree algorithm to produce a hybrid that consistently generates efficient multicast solutions for all possible multicast groups in any network. The constituent heuristics are based on Dijkstra's algorithm which maintains acceptable time complexity for the hybrid, and they rarely produce inefficient solutions for the same network/multicast. The resulting performance attained is generally good and in the rare worst cases is that of the shortest path tree. Our results show good performance over a wide range of networks (both flat and hierarchical) and multicast groups, within differing delay bounds. We also study the distribution of path delays to the multicast group members. As might be expected, although the bound is always met, delays are generally longer than those achieved with Dijkstra's shortest path algorithm.

1 Introduction

Many of the new services envisaged for ATM networks involve point to multipoint connections. Distribution services, such as video on demand or continuous information publishing services, are likely to have large numbers of customers. Interactive services such as multimedia conferencing, co-operative working and educational applications can also be well supported by multicasting. ATM offers the integration of data and real-time components such as audio and video. This implies that, for many multicast services on ATM networks, the network must make appropriate Quality of Service (QoS) provision particularly in terms of maintaining agreed bandwidth and minimising delay. Because of the potentially large numbers of users, routing of multicast connections is an important issue. Multicasting itself is much more efficient in its use of the network than multiple point to point connections, as cells are not replicated on individual links.

The topology chosen for the multicast routing tree can also give further efficiency savings, as we discuss in this paper.

Our discussion concentrates on graph-theoretical heuristics for multicast routing which combine bounded delay with efficient use of the network, for large-scale real-time multicast services. For networks with n nodes, the lowest delay from a source to each of the other nodes can easily be found in $O(n^2)$ time using Dijkstra's algorithm. The paths found in the process form a broadcast tree which can be pruned beyond the receiving group members. Provided all of the destinations are reachable within the delay bound this offers a satisfactory solution. Where the predominant requirement is efficiency, the total cost of a broadcast tree can be found using techniques such as Prim's or Kruskaal's algorithms. However, the equivalent problem for a proper subset of the nodes of the network is known as the Steiner tree problem which is NP-complete, although heuristics are available which give reasonable solutions. Finding a multicast routing tree which is both efficient and delay bound is also an NP-complete problem.

Our evaluations, and indeed those in the majority of published work in the area which we review, are done across a range of networks whose edges each have two quantitative parameters: which we call "cost" and "delay". The evaluations take place on a large number of such networks in order to assess different network conditions. The cost function could represent one of several practical values in a real network. In general, at the time of calculation, the cost incurred using each link can be considered to be constant (although it may vary in the life of the network). The cost could be proportional to the monetary cost incurred when a user uses that link. A sensible alternative would be to optimise the cost by relating it to the residual available bandwidth on a link. Certainly, the network considered for routing should first be pruned of any links incapable of carrying the bandwidth of the multicast under consideration. Yet another alternative is for cost to be proportional to distance which may in turn also be related to delay.

Taking the delay as a constant for the purposes of the calculation may well appear more questionable. However, in ATM networks, we are looking at setting up the multicast tree for the duration of a virtual channel (with only occasional modifications due to members joining or leaving the call) and it would be impractical to re-route due to short-term fluctuations in queue size. A fixed value of delay for each link would include an expected component for queueing delay onto the link as well as for the fixed switching, transmission and propagation delays and should be sufficiently representative of the information available at the outset of a connection set-up. For example, the delay could be based on the measured mean and variance of delay over the most recent half-hour period. For certain real-time services, special queueing provision and traffic shaping techniques may also reduce the variability of delay experienced within ATM networks.

The problem of arbitrary delay bound low cost multicasting in networks, where link cost and link delay are different functions, was first addressed by Kompella, Pasquale and Polyzos in [9]. Since then, there have been a number of other proposals for solutions to this problem. Previous evaluation work [16], [11] shows that on average these heuristics perform well. Further detailed analysis and evaluation of some of these heuristics has shown that there is a wide variance in the efficiency of their solutions. 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 a multicast group grows and shrinks the heuristic that provides the most efficient multicast solution also changes. The results of our evaluation work indicate that it is difficult to predict which heuristic provides the most efficient solution for any particular multicast/network combination. The variance in the efficiency of the heuristics 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.

The rest of this paper is organised as follows. In section 2 we define the bounded delay minimum cost multicast routing problem. In section 3 we describe and assess three heuristics and consider them as candidates for integration. Sections 4 describes the network model, benchmark algorithms and arbitrary delay bound we use to evaluate both the candidate heuristics and the hybrid. The candidate heuristics are evaluated in Section 5. Sections 6 describes the hybrid heuristic, which is evaluated in Section 7. We conclude the paper in Section 8 and identify current and further research.

2 Delay Bound Minimum Cost Multicast Routing

The bounded delay minimum cost multicast routing problem can be stated as follows. Given a 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 $\forall k, k = 1,n$; $D(s, M_k) \leq \Delta$, the delay bound, where $D(s, M_k) = \sum_{(i,j)} d(i,j)$ 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, and 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 Heuristics with an Arbitrary Delay Bound

Several heuristics have been proposed that use arbitrary delay bounds to constrain multicast trees. Kompella, Pasquale, and Polyzos [9] propose a Constrained Steiner Tree (CST_c) heuristic which uses a constrained application of Floyd's algorithm [5]. Widyono [18] proposed four heuristics based on a constrained application of the Bellman-Ford algorithm[1]. Zhu, Parsa and Garcia-Luna-Aceves [19] 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 and Vinitos [12] 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 [13], which we abbreviate as $CSPT$ and by Waters [16], which we abbreviate as $CCET$ (Constrained Cheapest Edge Tree), generally have a lower time complexity than Constrained Steiner Trees, but their solutions are not as efficient.

In the following sections, we concentrate on the solutions offered by Kompella (as being representative of a very efficient, but high time complexity technique) and the techniques of Waters and Sun and Langendoerfer, which, because they are based on variations of Dijkstra's shortest path algorithm and are of similar time-complexity, are good candidates for a hybrid capable of combination with Dijkstra's algorithm.

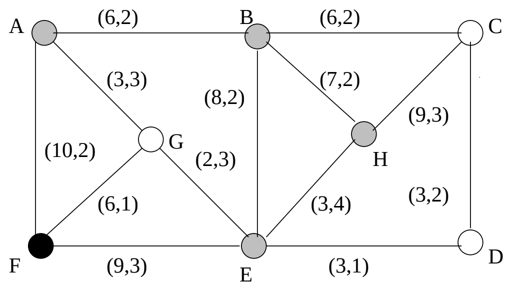

Figure 1: The example network

In the worked examples in the following description of these heuristics, we use the network illustrated in Figure 1, the edges of which are labelled with (cost,delay). The arbitrary delay bound is set to 7 in all cases. Kompella and Sun use a Δ of 8 since they find paths with a delay less than Δ; the Waters heuristic uses a Δ of 7 because it finds paths with a delay less than or equal to Δ. In each case, the worked example finds the multicast tree connecting source F to the destinations A, B, E and H.

(Note that we consider symmetrical cost and delay in either direction on a link. This may well not be the case in practical ATM networks, especially where the cost takes into account residual bandwidth. For asymmetrically weighted networks, the heuristics would have to start from a directed graph.)

3.1 The Constrained Steiner Tree (CST_c) Heuristic (Kompella, Pasquale and Polyzos)

The CST_c algorithm was first published in [9] and has three main stages [8].

1. A closure graph (complete graph) of the delay-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 delay values, using a similar technique to Floyd's all-pairs shortest path algorithm (see [5]).

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 more 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 [8].

   (Note that if the arbitrary delay bound applied to the heuristic is very large compared to delays in the network then the solutions produced will be similar to those calculated using the Minimum Spanning Tree (MST) for the Steiner Tree Problem [7].)

3.1.1 A Worked Example

Applying the first stage of the heuristic to the network in Figure 1 produces the constrained closure graph illustrated in Figure 2A. Again, all links are labelled $(cost, delay)$. 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. The other paths have no intermediate nodes.

Figure 2B 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 2C.) which when removed produces the solution in Figure 2D. This tree has a cost of 29 units and a delay bound of 7.

3.1.2 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

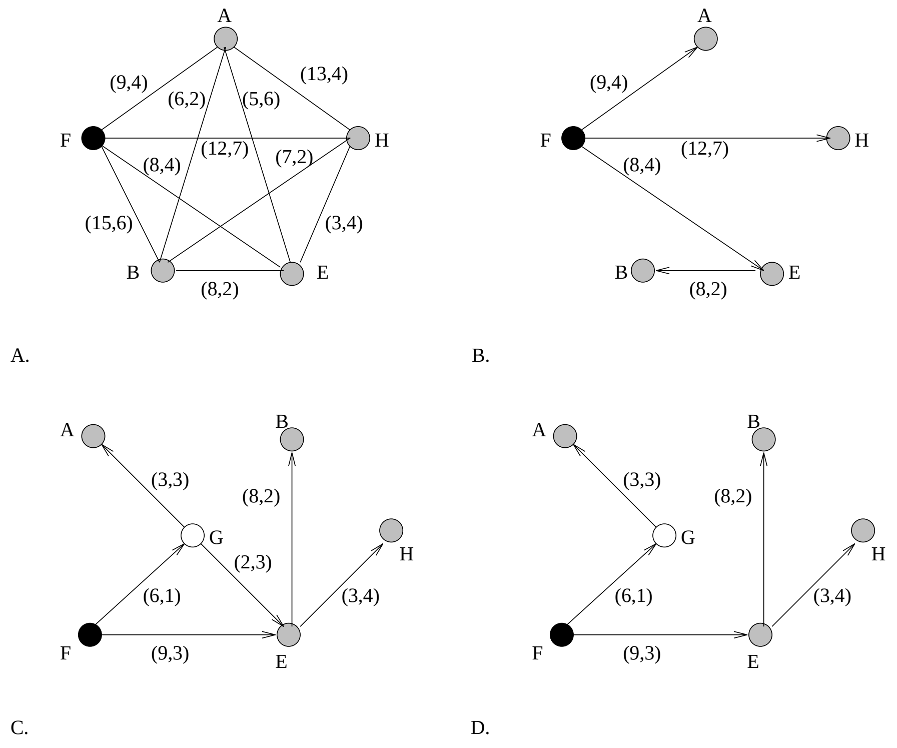

Figure 2: The CST_c heuristic

graph [5]. 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 $O(\Delta n^3)$. The effect of Δ on the time complexity can be reduced by decreasing the granularity of Δ through scaling, although this will compromise the accuracy of the results [18].

3.1.3 When CST_c costs more than a Shortest Path Tree

In most cases CST_c calculates multicast solutions that are cheaper than those produced by a Shortest Path Tree algorithm (SPT), but it does sometimes generate more expensive solutions. Figure 3 illustrates such a case. The underlying graph is not shown, but sufficient information is given for the purposes of illustration. The multicast is from the source node, F, to the destination nodes, B and D. The arbitrary delay bound is 12. The first stage of CST_c constructs a closure graph from the cheapest constrained 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 FB and FD 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 FA, AB and FC, CD. The SPT algorithm will select paths solely on the basis of the delay from the source to each node. The solution SPT provides is FA, AB and AD at a cost of 21 and delay 5. By chance the SPT has been able to take advantage of the common edge FA, which was not available in the closure graph for CST_c.

3.1.4 Multicast Tree Stability and Dynamic Groups

The topology of a CST_c multicast tree may be reconfigured as the tree grows or shrinks. 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,

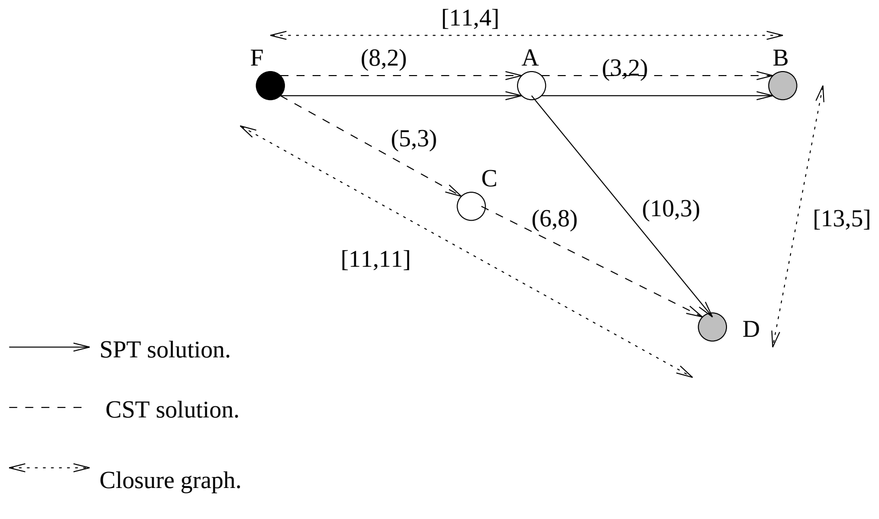

Figure 3: CST_c more expensive than SPT

the closure graph changes, and so the greedy process has a different set of nodes to consider. This may result in a multicast solution with a different topology from its predecessor.

3.2 The Constrained Cheapest Edge Tree ($CCET$) Heuristic (Waters)

The CCET heuristic was first published in [15] along with some simple preliminary evaluations. In [16], important variations of the heuristic were introduced and comprehensively evaluated.

The original heuristic and its variant [2] 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 many 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 $v \in V_T$, with the maximum 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.

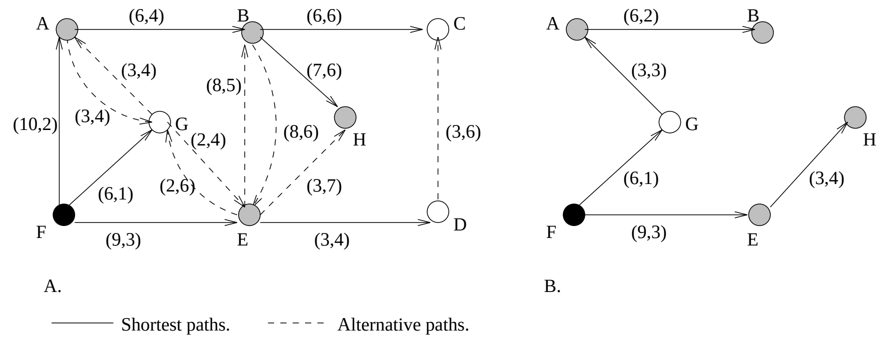

Figure 4: The CCET heuristic

3.2.1 A Worked Example

To illustrate the working of the heuristic we start with the graph shown in Fig. 1. 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 Fig. 4A 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 Fig. 4B) with a cost of 27 units and a final delay bound of 7.

3.2.2 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 $O(max(N, |E|)$ [6] where N is the number of nodes, and E is the set of edges in the search tree from the leaf node to the source. 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 increase 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 $|E|$. See section 5.1, which discusses the performance of the heuristic.

3.2.3 Pathological Behaviour of the CCET Heuristic

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 is, the more paths 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 5 between node G and the source includes three alternative edges GH, FB and CD 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, DS, was added to the path, necessitating the finding of an alternative path from G without this link.

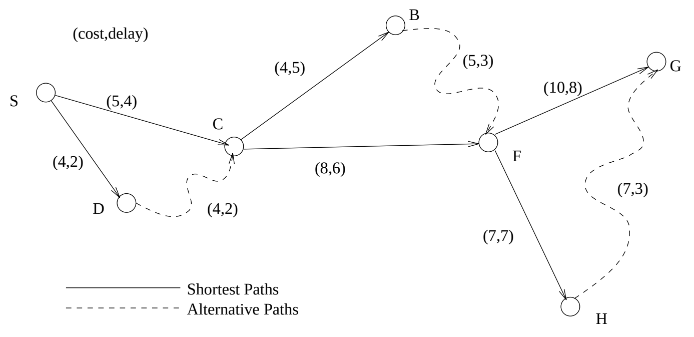

Figure 5: Example of a Rogue Path

In the second stage, the heuristic extracts the bounded delay minimal cost tree from the bounded directed graph constructed in 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 minimises the probability of loops occurring during tree construction and minimises the number of rogue paths found, thus radically reducing the portion of the search tree which is actually considered. 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.

The necessity of repeatedly rejecting rogue paths (and thus increasing search time) may 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. It may also apply in multi-cluster networks and an example is given in [3].

3.2.4 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 Δ. In some networks this rule can cause multicast trees found by the heuristic to be more expensive than might otherwise be expected.

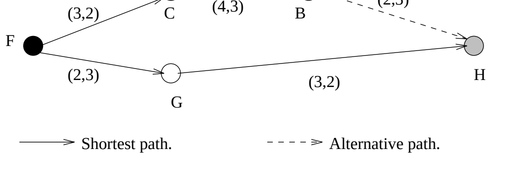

Figure 6: Costs increase as Δ is relaxed.

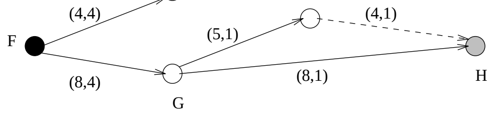

Figure 7: CCET more expensive than SPT.

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 6. With a delay bound of $\Delta = 5$ the multicast tree will include the edges FC, FG and GH at a cost of 8 units. This happens because the edge BH will have been excluded as it gives node H a delay of 8 units from the multicast source node F. If Δ is increased to 8 the edge BH is included and will be selected as the "cheapest" return route from node H towards F. The multicast tree then becomes FC, CB, and BH 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 Waters heuristic. The multicast tree found using Dijkstra's algorithm for the network in Figure 7 includes the edges FB, FG and GH, 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 FB, FG, GA and AH because AH offers the "cheapest" exit back to the source from node H. The cost of this tree is 21 units.

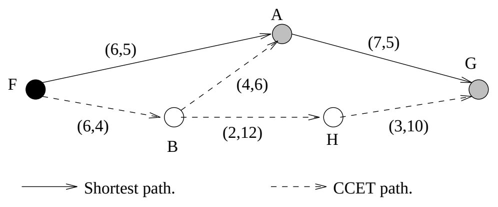

Figure 8: Adding a single node to the tree

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 8 illustrates the CCET and SPT solutions for a multicast. The multicast source is node F and the delay bound, Δ, is greater than 26. In the first instance the multicast includes only node G. The SPT solution will choose the route FA, AG to reach G at a cost of 13 units. CCET will choose the route FB, BH, HG to reach G at a cost of 11 units. If the multicast grows by the addition of node A the cost of the SPT solution will remain the same since A is already on the path to node G. The CCET solution has to add the link BA, increasing the tree cost to 15, to reach node A.

3.2.5 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 tree grows and dies, the broadcast tree only need be recalculated if the topology of the underlying network changes.

3.2.6 Constrained Cheapest Path Tree (CCPT)

A variation on the Waters heuristic, proposed by Crawford [2] uses the cheapest path back to the source rather than the cheapest edge leading to the existing tree as its selection mechanism. The idea is similar to a variation developed independently by Salama [11]. We have included the CCPT heuristic in the first of our evaluations, but as it generally produces more expensive results than the CSPT heuristic described below, we did not include this in the majority of our evaluations.

3.3 The Constrained Shortest Path Tree ($CSPT$) Heuristic (Sun and Langendoerfer)

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.

3.3.1 A Worked Example

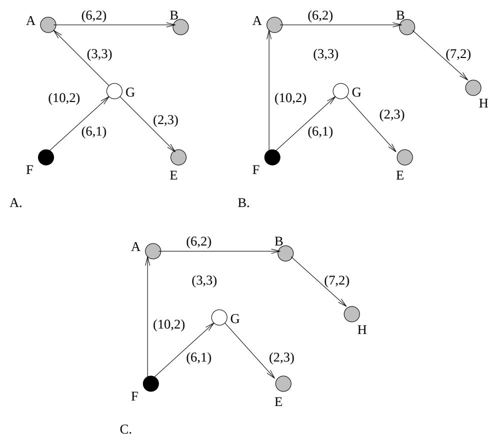

Figure 9: The CSPT heuristic

Applying the first step of the heuristic to the network in Figure 1 produces the minimum cost path tree illustrated in Figure 9A. Node H is not included in this tree because its minimum cost path has a delay of 8, which breaks the delay bound. Figure 9B 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 with nodes F,G and A. For this reason the edge FA is selected in preference to edge GA to give the final solution in Figure 9C. 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 occurs. The loop can be avoided by selecting, from the loop's 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 10 illustrates how the lowest cost spanning tree (A) and shortest delay path tree (B) when combined create a loop (C). By choosing the path AB from the shortest delay path tree and ignoring the path EB from the minimum cost path tree we obtain a loop free solution that does not violate the delay bound (D).

3.3.2 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 $O(n)$.

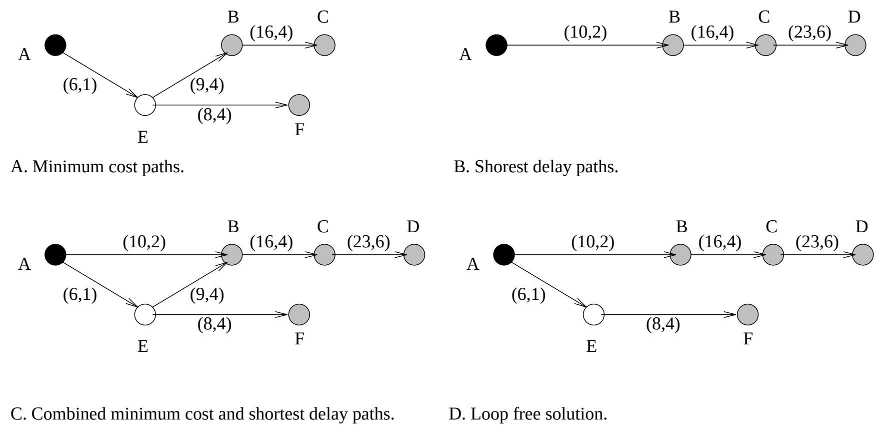

Figure 10: Loop removal in the CSPT heuristic

3.3.3 When CSPT costs more than the SPT.

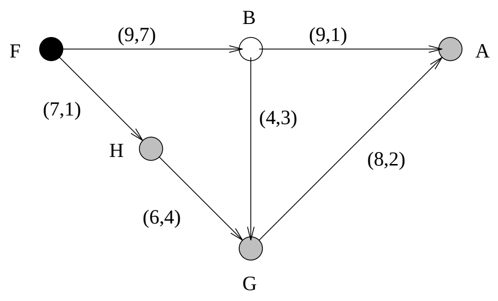

Figure 11: CSPT more expensive than SPT.

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 11, for a delay bound greater than 8, to connect the multicast nodes A,G and H to the source F, the CSPT heuristic will use the path AB, BF at cost 18 and path GH, HF at cost 13 because they are the shortest paths based on cost between the multicast nodes and the source. This results in a multicast tree of cost 31. The SPT algorithm based on delay will choose the path AG, GH, HF at a cost of 21 to connect all the multicast nodes to the source.

3.3.4 Multicast Tree Stability and Dynamic Groups

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 cheapest path from the existing tree. We propose a minor modification the the CSPT heuristic which eliminates its instability. Instead of calculating a solution for the multicast group, the calculation includes all nodes in the network, as is the case with the CCET heuristic. The multicast tree is extracted from the broadcast tree calculated in this way. We refer to the modified version of the CSPT as the stable CSPT, or sCSPT. The two techniques are compared in [3]. For smaller multicast group sizes the original heuristic produces, on average, more efficient solutions than sCSPT. As the multicast group size increases the performance of the heuristics converges, as expected. The difference between the two techniques is small enough to consider sCSPT as a valid alternative to CSPT in dynamic routing situations.

4 Evaluation Environment

Two network models are used to generate random networks in the evaluations described in this paper. In most cases, and where not stated explicitly, the network models are single cluster systems such as backbones or autonomous systems. These are generated using Waxman's model [17] which distributes nodes randomly over a rectangular co-ordinate grid. The Euclidean metric is then used to determine the distance between each pair of nodes and this is used for the delay metric. Edges are introduced with a probability depending on their length. We also use a factor, introduced by Doar [4] related to the number of nodes in the networks, to scale the probability of edges being included. The cost assigned to each edge is selected at random from the range [1,,L] where L is the maximum distance between any two nodes.

We also use a cluster network (connecting a number of clusters via a backbone network) for some of our evaluations, based on the hierarchical model of Doar [4]. The cluster interconnection mechanism proposed by Doar means that the number of links connecting each cluster to the cores is pre-defined and static. Our model is made more general by first generating a number of cluster networks, then a "backbone" network is produced using a node to represent each cluster. The "backbone" is then mapped back onto the clusters by connecting together nodes at random from each cluster. Further details are given in [3].

4.1 Benchmark Algorithms and Arbitrary Delay Bounds

The ideal benchmark algorithm to use would be one that produces optimal delay bound minimum cost multicast trees which, being an NP-complete problem, is impractical for large graphs. Instead we use the Minimum Steiner Tree heuristic (MST) of Gilbert and Pollack [7] which approaches a minimum cost for multicast trees, although they are of unbound delay. We also use the SPT as a benchmark to evaluate the cost savings made by using the various heuristics.

We chose the network diameter as the arbitrary delay bound for the evaluation of the multicast algorithms. This purely arbitrary choice provides an evaluation "mid-point" between the multicast delay, which is the tightest bound and the MST which is the delay at which the maximum improvement in network utilisation for each heuristic will be achieved.

5 Evaluation of the Candidate Heuristics

Within this section, the following acronyms are used.

CSTc	Constrained Steiner Tree (Kompella, Pasquale and Polyzos)
CSPT	Constrained Shortest Path Trees (Sun and Langendoerfer)
CCET	Constrained Cheapest Edge Tree (Waters)
CCPT	Constrained Cheapest Path Tree (Crawford)
SPT	Shortest Path Tree (Dijkstra)
MST	Minimum Steiner Tree (Gilbert and Pollack)

5.1 Performance averages

For each evaluation, 200 networks of 100 nodes of low edge density were used. Multicast groups were selected for sizes from 5 to 95 nodes, at steps of 5. There were 10 multicast samples for each multicast group size, for each network.

Figure 12 illustrates the percentage excess costs of using the four heuristics described above, relative to the MST and SPT benchmarks. For the CST_c heuristic we use a granularity of $\Delta/5$ to step through possible delay values (see Section 3.1).

The algorithm of 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 $CCET$ heuristic's solutions become cheaper than those of CST_c. The performance of the $CCET$ heuristic is much better than $CSPT$ and $CCPT$ for larger multicasts, but is worse for smaller multicasts. The

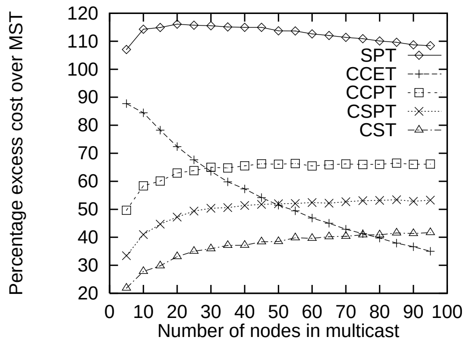

Figure 12: Average comparative costs

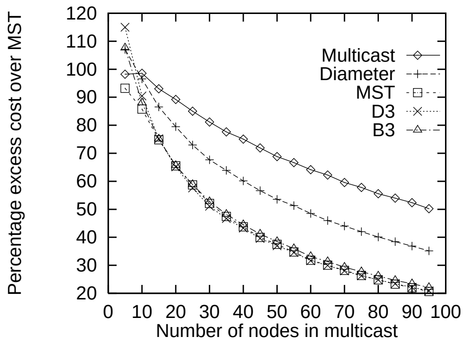

Figure 13: CCET excess costs as Δ increases

CCPT heuristic shows poor performance in comparison with that of *CSPT*. The solutions of *CSPT* and *CCPT* are similar because they are tightly constrained through their construction of paths using Dijkstra's *SPT* algorithm. CCPT is not considered further in this paper. Although *CCET* uses an extension of the *SPT* algorithm to construct its search space, it is not constrained by the algorithm when finding its solution. Rather, it relies on the chances of the selected edges leading to existing paths in the solution tree. This approach can result in small multicast solutions being relatively expensive, while large multicasts solutions are generally much cheaper.

We have observed that as the delay bound approaches the *MST* delay, improvements in solution efficiency of the *CCET* heuristic become negligible. (See figure 13 which plots the percentage excess cost achieved over MST for five different delay bounds, where D3 = 3*network diameter; B3 = 3*broadcast delay from the source.) Up to these delay bound limits the number of nodes visited during the tree search in the heuristic's second stage is $< O(2n)$, by observation, where n is the number of nodes in the network. If the delay bound goes much beyond these limits the heuristic is occasionally prone to very long execution periods which suggests that either the number of nodes or the number of edges in the search tree can become unacceptably large. (See the discussion in section 3.2.2.)

5.2 Specific multicast comparisons

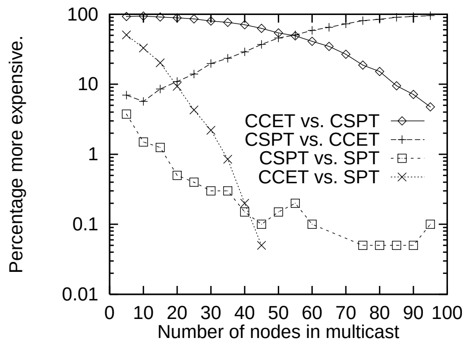

Figure 14: Exceptional comparative costs

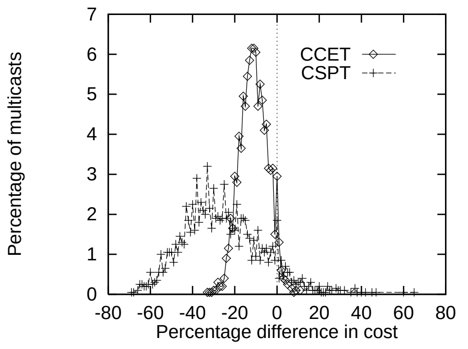

Figure 15: Cost distributions

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 14 illustrates a sample of the percentage of times *CCET* solutions are more expensive than those of *CSPT*, *CSPT* solutions are more expensive than *CCET*, and when the solutions of both *CSPT* and *CCET* are more expensive than the *SPT*. On average, the graph shows the expected results where CCET is cheaper for large groups and CSPT is cheaper for smaller groups. In nearly 5% of the sample, for multicast groups of 95 nodes, the solutions generated by *CCET* were more expensive than those generated by *CSPT*. Similarly, in 7% of the sample, for multicast groups of 5 nodes, the solutions generated by *CSPT* 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 SPT solutions. For larger multicasts $CSPT$ still generates some solutions that are more expensive than SPT, while $CCET$ does not. Figure 15 indicates just how large and varied these differences can be. The graph for $CSPT$ plots the 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. This behaviour shows, as might be expected from our analysis, that the solutions each heuristic generates depend on the algorithm, the topology of the network and the topology of the multicast. There is also a wide variance in the cost of solutions between the heuristics for the same size multicasts.

6 Hybrid Approach to Multicast Routing Heuristics

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 will generate solutions that are predominantly cheaper than SPTs for all network topologies, for all multicast group sizes.

CST_c on average generates good solutions but has an order of time complexity which may be too high for practical use. Also its calculation is based upon a variant of Floyd's All Pair Shortest Paths algorithm [5], making it unsuitable for integration with the other heuristics. We discard $CCPT$ because of its poor overall performance.

The $CCET$ and $CSPT$ heuristics generate the majority of their most efficient multicast solutions at opposite ends of the multicast group size range, and both base their calculations on trees generated by the SPT algorithm. Individually, each is vulnerable to generating some inefficient solutions throughout the full range of multicasts, but rarely will both heuristics generate an inefficient solution for the same network/multicast group pair. We combine the $CCET$ and $CSPT$ heuristics to obtain a hybrid of acceptable time complexity that produces solutions of significantly improved efficiency over SPTs. The hybrid will select the "cheapest" tree provided by each of these heuristics as the multicast solution. To guarantee maximum efficiency the SPT algorithm is also included in the hybrid to cater for the rare instances where both $CSPT$ and $CCET$ produce solutions that are more expensive than the SPT. The $CCET$ function, within the hybrid, must place a maximum limit on the delay bound it uses to calculate its multicast solution in order to limit its execution time, as previously suggested. This maximum value does not apply to the $CSPT$ function.

Integration of the three heuristics is simple. All three calculate the shortest path tree for delay, which is extended for the second stage of the $CCET$ heuristic. The $CSPT$ heuristic also calculates the SPT shortest path tree for cost, a task which can be conducted concurrently with the delay calculation. Once the trees have been obtained for each method their costs can be easily calculated and the cheapest tree selected as the solution.

The time complexity of the hybrid is dominated by the $CCET$ function. The first stage of this function has time complexity of at most $O(n^2)$. The second stage, the construction of the broadcast tree, has a time complexity of $O(max(N, |E|))$. In practice, the time taken by this stage is limited by maximum value on the delay bound it uses, as discussed in Section 3.2 and observed in Section 5. The $CSPT$ and SPT functions have a time complexity of $O(n^2)$.

7 Evaluation of the Hybrid Heuristic

Figure 16 illustrates the cost performance of the hybrid heuristic in comparison to $CCET$ and $CSPT$. The hybrid outperforms or equals both $CCET$ and $CSPT$. It is interesting to note that

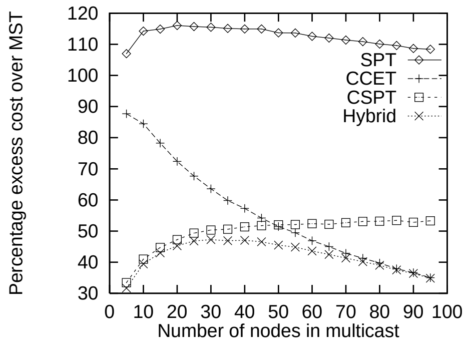

Figure 16: Average comparative costs

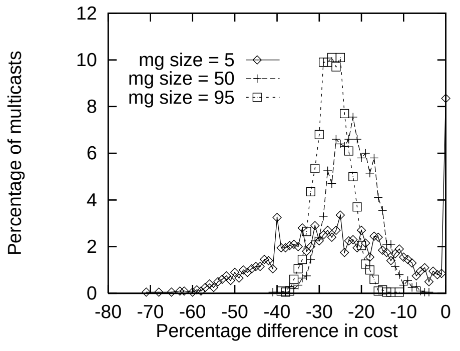

Figure 17: Cost distribution

for mid-sized multicasts the hybrid is able to provide solutions that are better than either *CSPT* or *CCET* can do separately. This occurs because the hybrid is able to choose the most efficient heuristic for each particular multicast. The efficiency of hybrid solutions for small multicasts is still subject to a fairly wide variance as figure 17 shows. These graphs plot the cost savings distributions of the hybrid over *SPT* for multicast group sizes of 5, 50 and 95 respectively. The dominance of *CSPT* for small multicast groups and *CCET* for large multicasts is obvious, as is the narrow but sharp intervention of *SPT* when required.

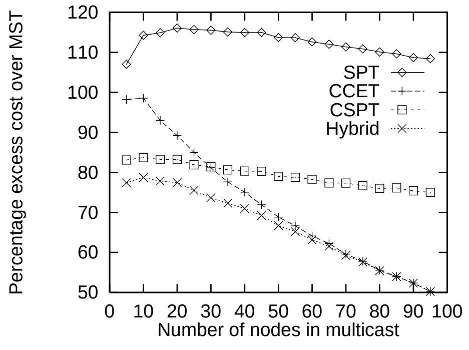

Figure 18: Multicast bound; single cluster

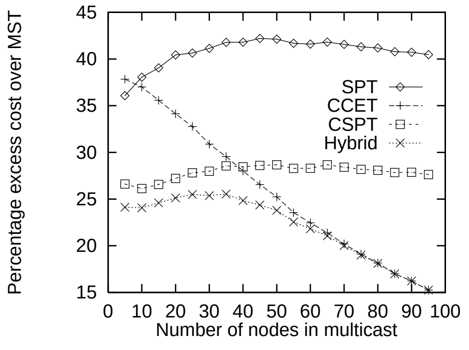

Figure 19: Multicast bound; multi-cluster

Figures 18 and 19 show the performance of the heuristics at the tightest possible delay bound, the delay to the furthest member of the multicast group. Figure 18 is plotted for a single cluster network and Figure 19 is a two-level hierarchy with clusters connected by a backbone. Results are similar for both hierarchical and non-hierarchical networks and the improved performance of the hybrid is confirmed. Note that, within this tight delay bound, the CSPT gives much smoother performance across the range of multicast group sizes and it is hard to achieve a very efficient solution for the smaller groups. The hybrid reflects this situation.

7.1 Distribution of path delays

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 solutions to the multicast problem addressed in this paper, 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 20 and 21 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 bound, while the number of occurrences is the actual number of paths of each

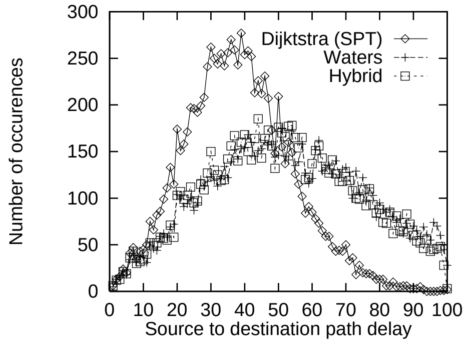

Figure 20: Path delays, 5 node multicasts

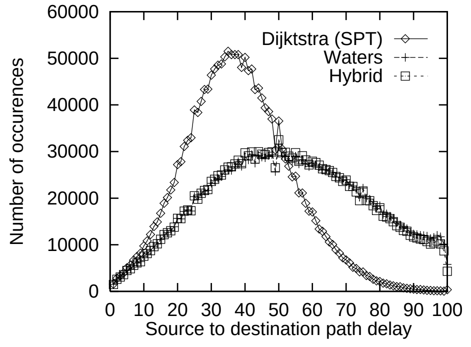

Figure 21: Path delays, all multicasts

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 holds for all multicast group sizes. As expected, the distribution shows that the paths produced by the shortest path tree 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 path tree algorihtm is used.

8 Conclusions and Further Research

We have identified problems of time complexity and performance variability in heuristics that have been proposed to calculate low-cost multicast trees that are bound by an arbitrary delay. By combining appropriate heuristics we propose a hybrid that produces efficient solutions over all multicast group sizes within an acceptable order of time complexity.

The hybrid heuristic uses metrics for every link in a network to perform its route calculation and so is amenable for implementation in link-state routing protocols such as the Internet's Multicast Open Shortest Path First protocol[10] or that used by the ATM Forum's Private Network-Network Interface [14]. The MBone currently uses Distance Vector routing, and may use the Resource Reservation protocol (RSVP) which reserves the requisite resources; this approach does not attempt to reduce the overall cost of the tree. Optimising the cost of the tree for the MBone would involve a move to a link-state approach. The evaluations of the hybrid have included both flat and hierarchical networks over a range of group sizes and using an "average" and a tight delay bound. The hybrid is shown to perform well under all these circumstances. The multicast tree produced by the hybrid reduces the total network bandwidth required to support multicast transmission. We have also shown, in our study of path delay distribution, that less buffering will be needed in the destinations using the hybrid than when using SPT, where it is necessary to play back the information at the same time at all recipients.

Note that the hybrid, in common with CSTc (Kompella) will sometimes involve reconfiguration of the multicast tree where group membership is dynamic. Where it is particularly important to have a stable tree, which can be pruned and regrow branches, we suggest the use of the constituent heuristics: CCET (Waters) for large groups relative to the size of the network and the broadcast and prune version of CSPT (Sun) which we propose in Section 3.3.4.

An important result of this work, and a departure from current routing solutions, is the integration of several heuristics which are individually unstable (as might be expected in an heuristic approach) into a stable hybrid. Hybrid methods may also have an application in other multicast or load sharing route calculation algorithms.

Further work is needed to evaluate the effect of using the heuristics within individual networks which form part of a larger internet.

9 Acknowledgements

We would like to acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) for our work on multicast routing (Grant ref. GR/K55837). We would also like to thank the anonymous reviewers for their constructive comments.

References

[1] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall,Inc., 1987.

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

[3] 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.

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

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

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

[7] E.N. Gilbert and H.O. Pollack. Steiner Minimal Trees. *SIAM Journal on Applied Mathematics*, 16, 1968.

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

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

[10] J. Moy. Multicast Extensions to OSPF. RFC 1584, March 1994.

[11] H.F. Salama, D.S. Reeves, I. Vinitos, and Tsang-Ling. Sheu. Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks. In *Proceedings of the 6th IFIP Conference on High-Performance Networks (HPN'95)*, 1995.

[12] H.F. Salama, D.S. Reeves, and Y. Vinitos. Evaluation of multicast routing alogorithms for real-time communication on high-speed networks. *IEEE Journal on Selected Areaa in Communications*, 15(3):332–345, April 1997.

[13] Q. Sun and H. Langendoerfer. Efficient Multicast Routing for Delay-Sensitive Applications. In *Second Internatiopnal Workshop on Protocols for Multimedia Systems (PROMS'95)*, pages 452–458, 1995.

[14] The ATM Forum Technical Committee. *Private Network-Network Interface Soecification, Version 1.0.* The ATM Forum, March 1996.

[15] 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.

[16] A.G. Waters and J.S. Crawford. Low-cost ATM Multimedia Routing with Constrained Delays. In *Multimedia Telecommunications and Applications (3rd COST 237 Workshop, Barcelona, Spain)*, pages 23–40. Springer, November 1996.

[17] B.M. Waxman. Routing of Multipoint Connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.

[18] 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.

[19] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting. In *Proceedings of INFOCOM*, pages 377–385, 1995.