How Bad is Naïve Multicast Routing?

Matthew Doar and Ian Leslie
University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG. U.K.

ABSTRACT

When the problem of routing multicast connections in networks has been previously considered, the aim has been to minimise overall costs, possibly subject to a delay constraint. The emphasis has been on examining the problem with a source transmitting to a fixed set of destinations (the multicast group) which are known when communication is established. There are clearly some applications for which the set of destinations will be dynamic, with destinations joining and leaving the multicast group during the communication. Under these conditions the computation of an "optimal" spanning tree for each new multicast group may not be the best way to proceed. An alternative is to make modest alterations to an existing spanning tree to derive a new one. An extreme variation of this is to use minimal cost source to destination routing for each destination, effectively ignoring the existing multicast tree. This naïve approach leads to a simple implementation but clearly produces non-optimal spanning trees. In this paper we examine just how non-optimal these trees are in random general topology networks and conclude that they are worse by only a small factor. The factor is reduced still further if a hierarchy is imposed upon the random network to give a more realistic model.

1 Introduction

Multicast, which here we consider in the limited sense[1] as the connection of multiple destinations, the multicast group, to a single source, is likely to become increasingly important as networks are used to carry continuous media traffic such as voice and video to the end user and potential bandwidth reductions become more significant. One of the more difficult aspects of providing multicast is deciding how the connections should be routed within a network to minimise or meet constraints for a series of parameters including delay, jitter and the link capacity used. Any scheme which is chosen must also be able to cope with heterogeneous networks [1], where some nodes provide multicast capability and others do not. This situation is likely to arise in the wide area for economic reasons but should be less of a concern in the local area. Another problem is that a route may only have a limited degree of replication available at an intermediate node [2].

Multicast groups may be either dynamic or static in nature. A static group, once set up, remains unmodified until it is discarded. A connection to a static group tends to be set up following a request from the source, for instance, file replication across a network. Unlike a static group, the membership of a dynamic group can vary with time. The state of the dynamic group tends to be driven by destinations, either destinations which are already part of the connection and wish to leave, or destinations which are outside the connection and wish to join it. The modifications are usually made by the source following a request from a destination. An example of an application which has dynamic groups is entertainment video distribution.

The problem of connecting a subset of nodes, $\mathcal{S}$, in an undirected graph $\mathcal{G}$, using the minimal cost subgraph such that there is a path between each pair of nodes in $\mathcal{G}$, is known as the Minimum Steiner Tree (MST) problem. The MST problem is one of the classic NP-complete problems of graph theory [3, 4]. The nodes which are not in $\mathcal{S}$ but which are used in the subgraph to connect the chosen nodes are known as *Steiner nodes*. The construction of a Steiner tree for multicast routing is one extreme form of multicast routing, as shown in Figure 1.

Previous work on heuristic solutions (summarised by Winter [5]) of the MST problem has generally taken one of the two following approaches. In the first approach, likely Steiner nodes are selected and then a minimum spanning tree of these nodes, together with the nodes in $\mathcal{S}$, is constructed [6]. In the second approach, a minimum spanning tree of the whole graph is pruned to remove superfluous nodes and then per-

[1] There are of course many-to-one ('concast') and many-to-many forms of multicast as well.

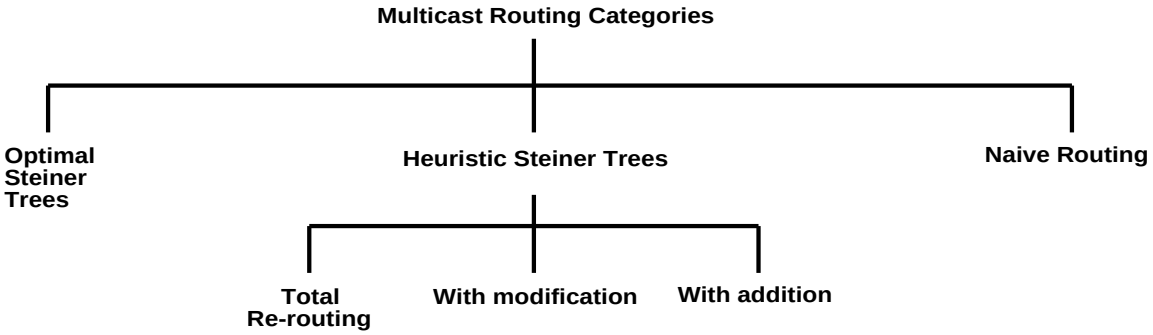

Figure 1: Multicast Routing Categories

turbed to try to include more Steiner nodes, which may reduce the overall cost of the spanning tree [7]. It is one of the original versions of the latter approach by Kou, Markowsky and Berman [8], referred to hereafter as KMB, which has been chosen as the basis of comparison with the simple algorithm detailed below.

The KMB algorithm is described in further elsewhere [7] but briefly, consists of five steps. An all-pairs, shortest path graph of $\mathcal{S}$ is constructed and a minimum spanning tree of this graph is found. This tree is used as a template to add the relevant shortest paths in $\mathcal{G}$, possibly introducing new nodes. A minimum spanning tree of this new subgraph is then constructed. Finally, this new graph is pruned until all the leaves of the tree are destinations.

The performance of heuristic solutions can be described by two properties, their computational complexity and their *inefficiency*. By inefficiency we mean the ratio of the cost of the resulting spanning tree to the cost of the minimal Steiner tree.

The worst case computational complexity of the KMB algorithm is $O(|\mathcal{S}||\mathcal{G}|^2)$ and it has a worst case inefficiency which is bounded by 2(1-1/l), where l is the number of leaves in $\mathcal{S}$, and inefficiency is defined as the ratio of the cost of the heuristic tree to the cost of the optimal tree. The mean inefficiency of the heuristic algorithms is generally considerably better than this, often less than 1.05. The KMB MST heuristic algorithm, whilst not having a computational complexity as low as others, is relatively simple to implement and shows similar inefficiency characteristics when compared with the other more complex heuristics of both approaches [9, 10]. The mean performance of the KMB heuristic is close enough to the optimal solution to permit its use for the purposes of comparison of the simple algorithm introduced in Section 3.1.

Whilst there are many parameters which are important in setting up a multicast connection, most of which involve some form of heuristic solution to the MST problem for various factors, the dynamic aspect of many multicast connections means that the time and effort expended in setting up a near-optimal solution is likely to be wasted soon after completion, when the configuration must be changed yet again. This is especially true as the size of $\mathcal{S}$ grows or the rate of change of group membership increases.

As shown in Figure 1, there are three ways in which heuristic Steiner trees may be used for routing multicast groups whose membership is dynamic. One is to re-route the whole tree whenever the group membership changes. This may have repercussions for members who remain in the group, since there may be a disturbance in the communication, and in an ATM network [11], such a change may allow cells to arrive out of order at the destination. Another method is to permit partial or local rearrangement of the tree when modifications to the membership occur. This has the same reordering drawback as total re-routing.

Yet another approach is to begin with an optimal (or near optimal) tree and make minimal changes to it as group membership changes — in the extreme causing no disruption to members who remain in the group. This is consequential to the approach described by Waxman in [10]. As membership in the group changes, one would expect the inefficiency of this method to degrade and indeed simulation studies have shown occasional 'spikes' of very high inefficiency [12].

A third category of multicast routing is shown on the right in Figure 1. This is to choose a suboptimal tree which will be resilient to changes. In the extreme case, finding the minimum source to destination path for all destinations independently and taking the union of these paths produces a tree which will be resilient to changes but will produce suboptimal trees. By resilience we mean that the spikes of inefficiency

should be much smaller, since the inefficiencies are independent of the order in which the modifications occur. The computational complexity of a single-source shortest paths algorithm, such as that due to Dijkstra, is $O(|\mathcal{G}|^2)$ but can be reduced to $O(e \log |\mathcal{G}|)$ for sparsely connected graphs, where e is the number of edges in $\mathcal{G}$.

We consider that the scarce resource in future multicast networks will not be the bandwidth, but instead will be the number of signalling messages which must be generated and, in ATM networks, ensuring that no reordering of cells occurs. In this paper we examine the performance of the last approach on randomly constructed networks undergoing stochastic addition and removal of destinations from a multicast connection. The use of well-known routing algorithms such as minimum source-destination path routing also provides a simple way to involve the requisite constraints such as delay, bandwidth and reliability, since these are relatively well understood for point-to-point routing [13].

2 Network Models

2.1 Random Graphs

To ensure that simulations of the effects of the different routing algorithms were independent of the idiosyncrasies of any specific network, random graphs were constructed. These were developed so that their connectivity characteristics approximated those observed in real networks.

The first network model to be considered, following Waxman [10], was one where graphs were constructed by distributing n nodes across a Cartesian coordinate grid. The location of each node had integer coordinates and multiple nodes were permitted to exist at any location. The edges were added to the graph by considering all possible pairs (u, v) of nodes and using the probability function

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

to create an edge, where $d(u, v)$ was the Euclidean distance between the nodes' locations, L was the maximum possible distance between two nodes and α and β were parameters in the range $0 < \alpha, \beta \leq 1$. A large value of α increases the number of connections to nodes further away, whilst a large value of β increases the number of edges from each node. The cost of an edge was defined simplistically as $d(u, v)$, the distance between its nodes, being a typical factor in multicast routing problems. Each graph was tested to ensure that only one connected component existed, i.e. that a spanning tree existed which covered all the nodes in $\mathcal{G}$. Graphs were generated until one with a single connected component had been created.

This method of graph construction produces graphs which model some aspects of real networks. For instance, values of 0.25 and 0.2 for α and β provide graphs which have the appearance roughly resembling that of geographical maps of the major nodes in the Internet. However, one problem with the model as originally proposed is that as the number of nodes increases, the number of edges from each node (*degree* tends to increase as well. The graph on the left of Figure 2 shows a 150 node random graph with a relatively large number (approximately six) of edges from each node. The modification proposed and implemented is to scale P_e by a factor related to the number of nodes in the graph. This is successful since the probability of each edge existing is independent for each node and hence forms a binomial distribution, which has a mean directly proportional to the number of samples, in this case, the number of nodes in the graph. To ensure that the mean degree of each node remains constant, another scale factor related to the mean distance between two random points must be introduced. For $\alpha = 0.25$ and $\beta = 0.2$, this factor, k, is approximately 25, by measurement. The modified probability function is shown in equation 2, where $\overline{e}$ is the mean degree of a node and $|\mathcal{G}|$ is the number of nodes in $\mathcal{G}$.

$$P_e(u, v) = \frac{k\overline{e}}{|\mathcal{G}|} \beta \exp \frac{-d(u, v)}{\alpha L} \tag{2}$$

The graph on the right of Figure 2 shows a 150 node random graph with a mean degree of three edges from each node. As can be seen, the constant degree random graph network model provides networks which more closely resemble real ones in their connectivity.

2.2 Hierarchical Graphs

The second model of networks proposed attempts to model the hierarchy of networks that exist in reality, with multiple LANs connected to a WAN, or a LAN comprising many small switches. A proportion of the n nodes were used as a core, connected in the manner of the modified random graphs of Section 2.1 using the scaled version of the edge probability P_e and checking for the existence of a single connected component. The remaining nodes were divided into equal sized clusters, which were connected internally in the

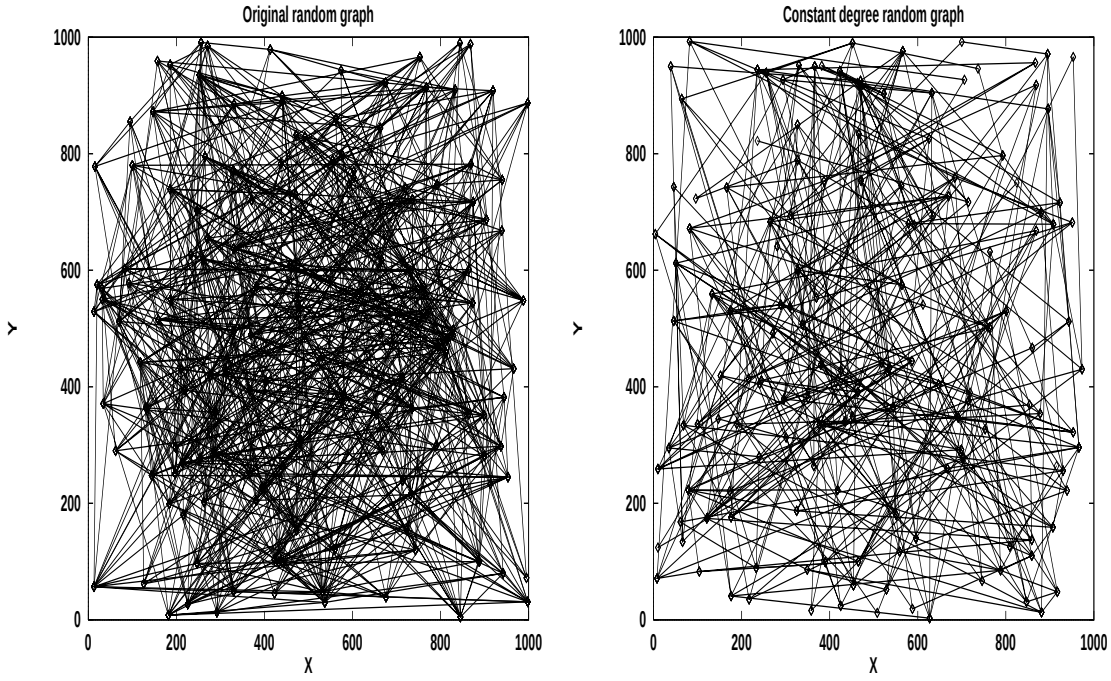

Figure 2: Original random graph and constant degree version

same fashion as the core. The clusters were then attached by a fixed number of edges, typically two, to randomly chosen nodes in the core, not necessarily unique. Both the size of the central core and the size of the clusters on the Cartesian grid were specified. The final parameter was whether the graph was separated or non-separated, i.e. whether the clusters were permitted to overlap the central core. In the simulations presented in Section 4, non-separated graphs were used and the clusters were permitted to overlap each other, modelling the existence of multiple LANs at a site.

Figure 3 shows typical network models constructed using these guidelines. Both graphs contain 100 nodes, 40 of which are in the core and the remaining 60 divided into three clusters of 20 each. The clusters were limited in area to squares of side 200 in a square grid of side 1000. The core was limited to a square of side 500 centred in the grid. The mean degree of each node was set at three and each cluster was connected to the core by two edges.

3 Modification Algorithm

After the graph had been constructed, one node was chosen at random as the notional source of the multicast connection. Another different node was also chosen and the shortest path was found between them. Now that the multicast connection existed, further destinations were added or deleted from the multicast tree using the following probability function.

$$P_c(k) = \frac{\gamma(n-k)}{\gamma(n-k)+(1-\gamma)k} \qquad (3)$$

where P_c is the probability that the modification is an addition, k is the number of destinations in the multicast connection, n is the number of nodes in the graph and γ is a parameter in the range $0 < \gamma < 1$. γ is the mean fraction of destinations/nodes in the connection. If $\gamma = k/n$, then $P_c(k) = 1/2$, if $k/n < \gamma$, then $P_C(k) > 1/2$ and if $k/n > \gamma$, then $P_C(k) < 1/2$. A modification was defined as either an addition or a deletion.

3.1 Algorithms for the Addition of Nodes

Waxman's original algorithms [10, 12] for connecting new destinations to an existing multicast connection were:

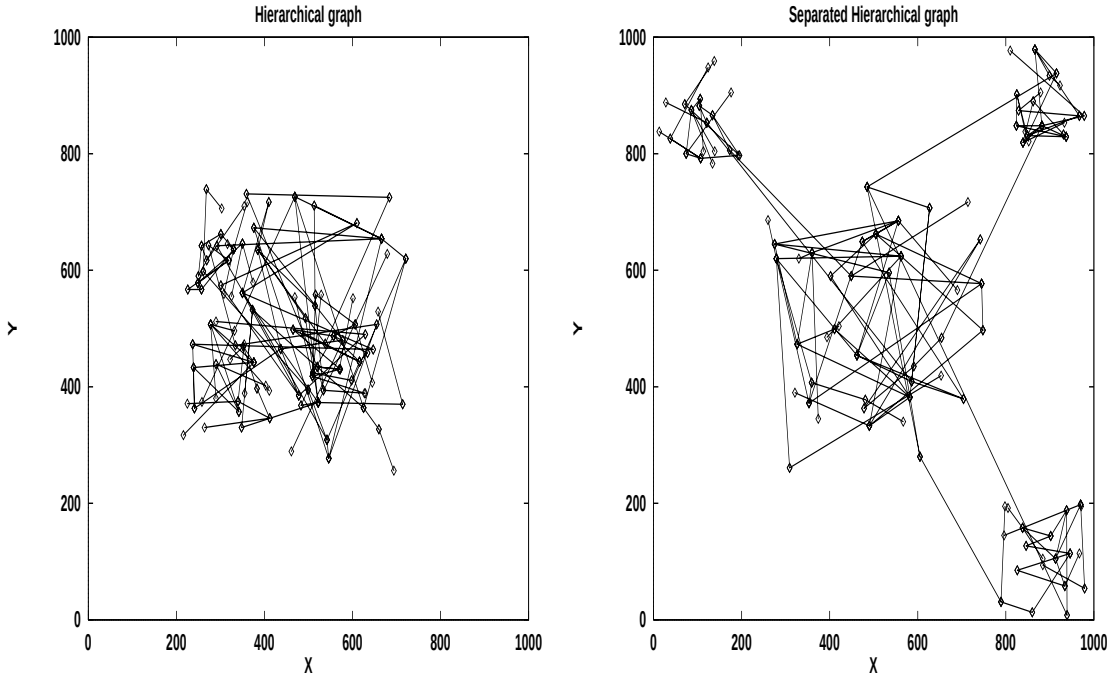

Figure 3: Hierarchical graphs without and with separation

Greedy Find the nearest node already in the connection to the node to be added and connect the two.

Weighted Greedy Take a proportion of the weight from the greedy algorithm and a proportion (e.g. 0.3) of the distance from a node known as the notional owner (the source) of the connection. Use the minimum of the sum of these two weights to decide which existing node in the multicast connection to route to the new node from.

Other proposed costs include the distance from the centre of the tree (see [14], pp. 213–214 for a definition of centrality) or the number of connections attached to a node.

The above algorithms all fall into the category of heuristic Steiner trees with addition, as shown in Figure 1. The algorithm used in the simulations described in this paper is one in which the tree which is constructed is independent of the order of the modifications. The particular algorithm chosen was to find the shortest path from the source to the node which had been chosen to be added and to add all the nodes along that path into the multicast connection as Steiner nodes.

An example of another simple algorithm would be to find the shortest path from the source using dynamic costs on the edges. If k paths to destinations used an edge in the graph, then its cost would be scaled by k, encouraging Steiner trees with minimal diversion until near the destinations.

3.2 Algorithm for the Deletion of Nodes

Multiple destinations were allowed to exist at the same node in the graph, modelling multiple destinations on the same LAN, for instance. If a node chosen for deletion was found to have multiple destinations, then the number of destinations was simply decremented by one. However, if there was only one destination at the node chosen for deletion and the node was a leaf, then, after deletion of the node, its immediate predecessor was considered recursively for deletion. The recursive deletion ceased when a node was encountered which was part of an internal path to another destination. In this case the node was left in the Steiner tree, but marked as deleted for later action if the other dependent destinations subsequently became deleted. The source node was never deleted in this scenario, and if the lower limit of two nodes in the connection was reached and the modification function

indicated the deletion of a node, then no action was taken.

This series of decisions and recursions is realistic, since routing tables at each node have to contain enough information about the numbers of multicast destinations using the node to permit timely recovery of resources when a destination is removed from the multicast connection.

4 Results

Multiple simulations of the above models were performed and the results are shown in Figures 4 to 11. For each run a graph was generated according to one of the network models described in Section 2 and 2000 modifications were simulated. The ratio of the cost of the multicast tree obtained by the simple algorithm to that of the KMB Steiner heuristic tree constructed afresh for the current set of destinations was evaluated every 10 modifications. This ratio is close to the inefficiency mentioned in Section 1. Ten series of generation of a graph and a run of modifications were performed for each point to obtain the Student-t distribution confidence intervals shown in the graphs. Longer runs indicated little reduction in confidence intervals in most cases.

Figure 4 shows how the inefficiency changes with the number of modifications made. The graph shown on the left is for a 100 node constant degree random graph with a mean of 50 destinations in the multicast connection and a mean degree of three. The graph on the right shows how the number of nodes involved in the multicast connection changes during the simulation. The shape of these graphs is typical of the changes in the inefficiency and number of nodes as modifications are made for many different sizes of graphs, fractions of the graph nodes involved in the multicast connection and different network models. The spikes are not so sharp as those seen by Waxman [12] due to the nature of the naïve algorithm, since the multicast tree constructed by it is independent of the order in which the modifications occurred. The inefficiencies which do occur are due to occasions when a new node being added is close to one already in the multicast tree, but a (partially) disjoint path is used to reach it and the path is only slightly shorter than the path to the new node via the existing node.

Figure 5 shows how the mean and maximum values of the inefficiency changes with an increasing number of destinations involved in the multicast connection for various sizes of the original random graphs network model. As can be seen, increasing the graph size results in an increase in the inefficiency whilst the changing proportion of destinations makes only a minor difference to the inefficiency. The inefficiency is unity when only two nodes are involved in the connection, since the shortest path between the two is also the (trivial) Steiner tree. When all the nodes are involved in a connection, the KMB tree is the minimum spanning tree of the network. This does not occur in the graphs shown since multiple destinations can exist at a single node. The increase in confidence intervals as the size of the graph increases is probably due to the increased possibility of the existence of long paths mostly separated from the rest of the multicast tree, which affect the performance of the simple algorithm drastically. This effect can also be seen in the graph of the maximum values, though the general trend in both cases is that the inefficiency increases as the number of nodes, and in this case the degree, increases.

Figure 6 shows how the mean values of the inefficiency changes with an increasing number of destinations involved in the multicast connection for various sizes of random graphs with the constant degree network model. As can be seen in the graph on the left, increasing the graph size results in little change in the inefficiency if the degree is constant. It also shows that the inefficiency is relatively independent of the number of nodes in the network for a given proportion of destinations involved in a multicast connection. The graph on the right in Figure 6 shows the results of varying the degree of a 100 node constant degree random graph. It suggests that the performance of the simple addition algorithm is indeed directly related to the mean degree of the graph. Figure 8 confirms that this relationship is approximately linear.

Figure 7 shows how the mean value of the inefficiency changes with an increasing number of destinations involved in the multicast connection for various sizes of the hierarchical graphs network model. As can be seen in the graph on the left, increasing graph sizes again result in little change in the inefficiency. The values of the inefficiency obtained with the hierarchical model are less than those obtained with the constant degree model, probably due to a more limited set of inter-cluster paths from node to node, so the naïve algorithm is more likely to choose the same ones as an optimal MST algorithm. The graph on the right in Figure 7 shows the effects of varying the degree of a 100 node non-separated hierarchical graph. It suggests that, as with the constant degree model, the performance of the simple addition algorithm is related to the mean degree of the graph.

Figure 8 confirms the approximately linear relation-

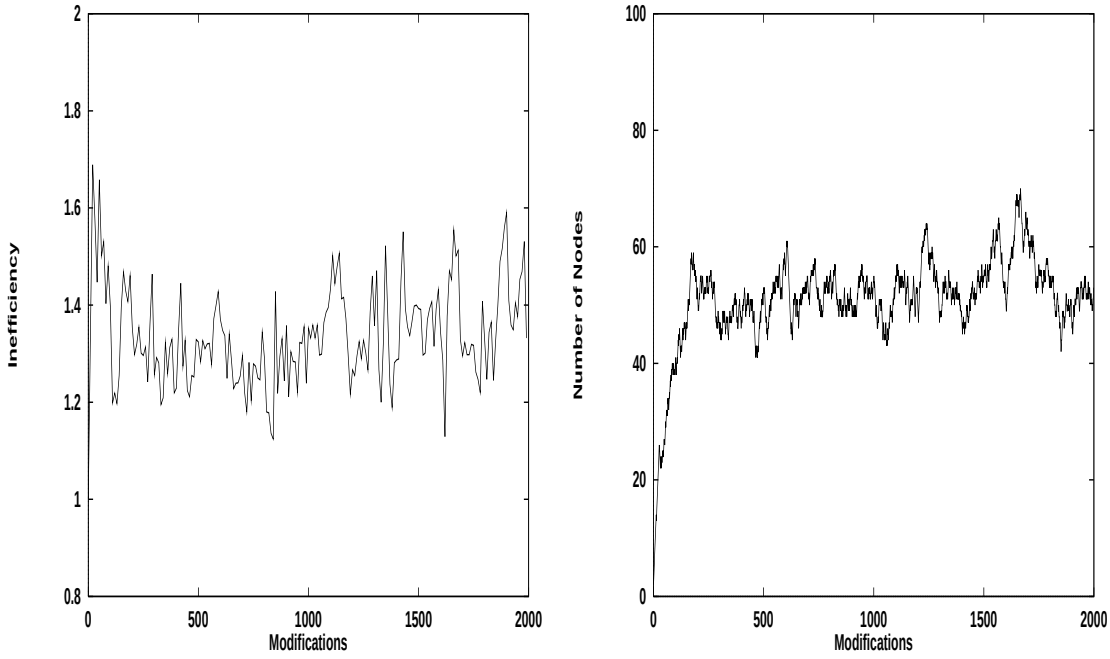

Figure 4: Inefficiency and number of nodes for a single run

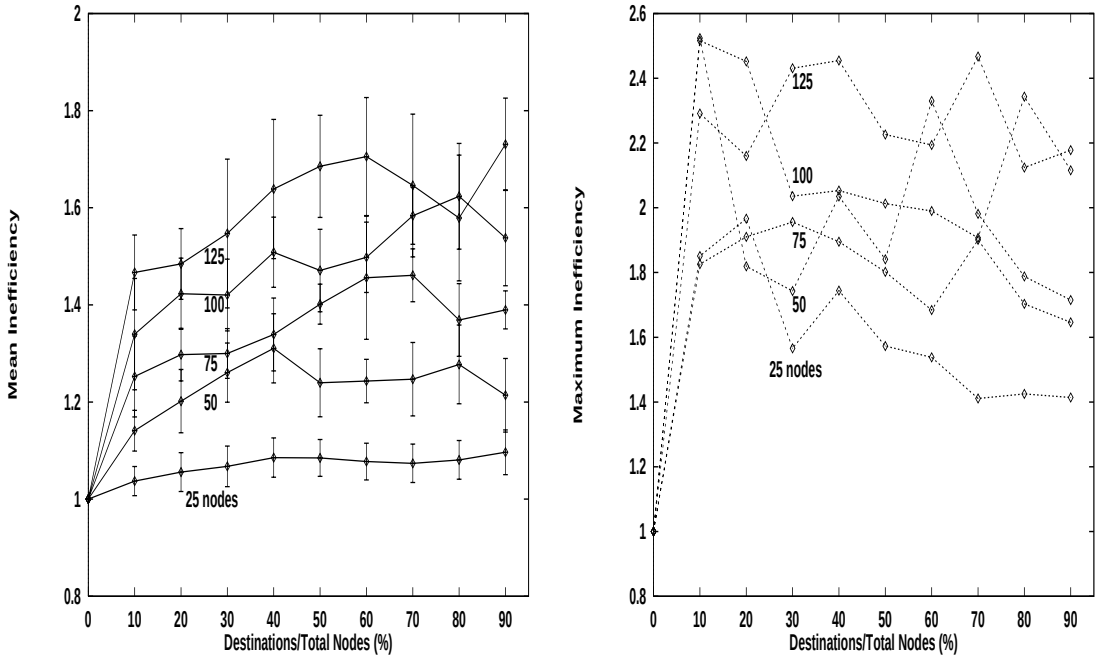

Figure 5: Mean and maximum inefficiency of original random graph

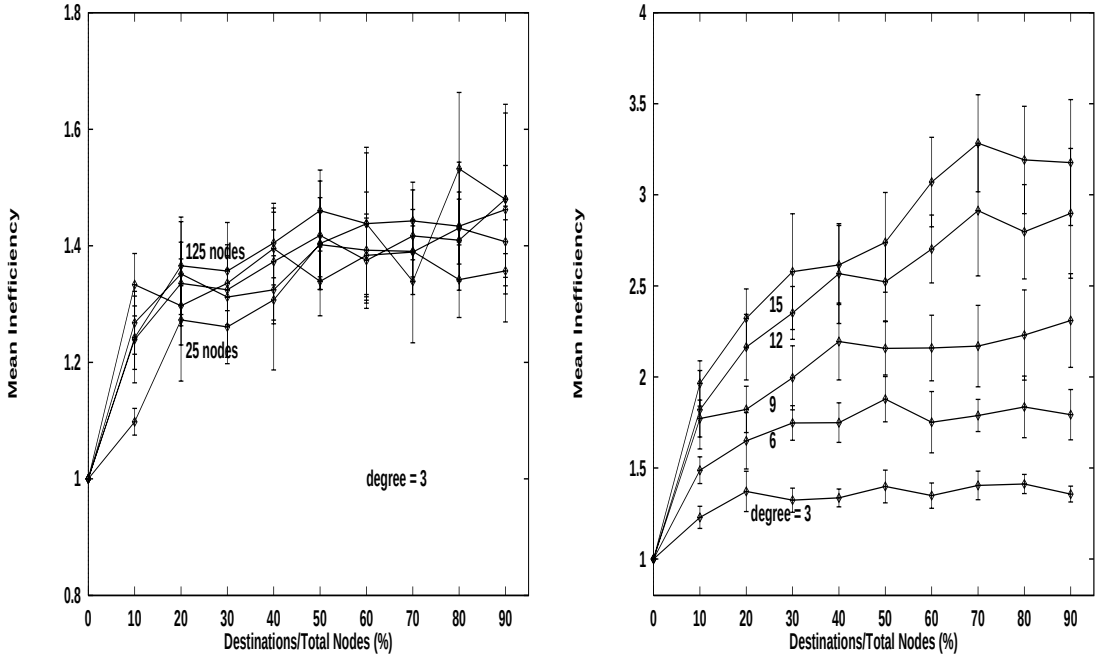

Figure 6: Inefficiency of constant degree random graph for various graph sizes and degrees

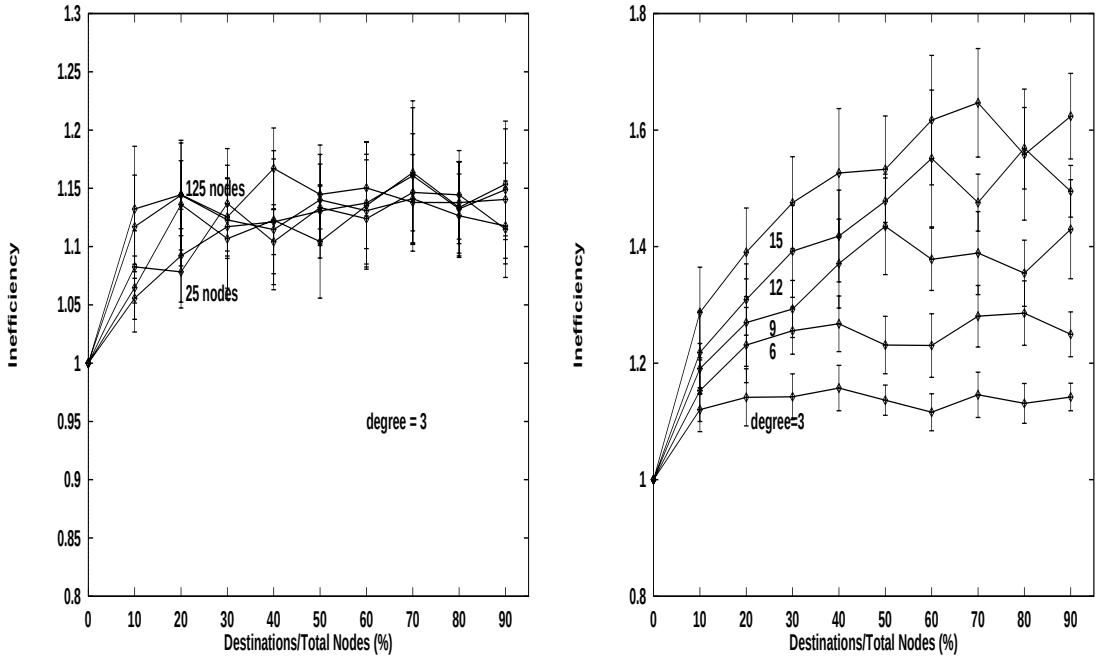

Figure 7: Mean Inefficiency of non-separated hierarchical graph for various sizes and degrees

ship between the degree and the inefficiency for both network models. The values shown are for 20% and 60% destinations/nodes in the 100 nodes of the graph being involved in the multicast connection (on average). As the degree increases, the number of available paths between any two nodes increases and hence the possibility of the shortest path tree coming close to the heuristic KMB tree decreases and the inefficiency increases.

Figures 9 to 11 all refer to random graphs with 100 nodes and give further insight into the difference in performance between heuristic Steiner tree algorithms such as KMB and the naïve but robust shortest path algorithm.

Figure 9 shows how the maximum path length varies with the percentage of destinations/total nodes and also with the mean degree. As may be seen, the paths chosen by the naïve algorithm are indeed shorter than those chosen by KMB and the length of the paths decreases as the degree of connection of the graph increases. The maximum path length tends to increase in KMB trees as the proportion of destinations increases. In shortest path trees, the change is less marked. This is probably due to the extended nature of the trees constructed from pruned minimum spanning trees, compared to the more compact nature of a tree constructed from shortest paths.

Figure 10 shows the maximum hop count for each path in the two sets of trees, and how it varies with the percentage of destination nodes/total nodes and also with the mean degree. Again, very little difference is seen in the values from the shortest path graphs as the proportion of destinations changes. A small decrease with respect to the degree can be seen in the graph on the right. The changes for the KMB graphs are even more marked than the maximum path length seen in Figure 9, though the reasons for them are much the same. The maximum hop count for the KMB trees is observed to change very little with respect to the degree of the underlying tree, increasing slightly as the degree increases.

Figure 11 shows the mean bandwidth used by each tree, defined as mean of the reciprocal of n_e for each edge in the multicast tree subgraph, where n_e is the number of paths which use the edge e. The bandwidth is scaled to the bandwidth which would be required if no multicast were possible except at the source. Thus an edge which is only used by one path has a bandwidth of 100%, compared to one used by two paths, which uses 50%, and so on. As may be seen from the figure, the KMB algorithm gives a substantial reduction in the bandwidth used, compared to the shortest path algorithm. The reduction is again dependent upon the percentage of destination nodes/total nodes involved in the multicast tree, this time improving as the proportion increases. The naïve shortest path algorithm on the other hand gives a relatively small sharing of bandwidth resources, though it is almost independent of the mean degree of the graph.

5 Improvements

Whilst the simple algorithm described above provides a set of multicast connections which appear to tend to a fixed multiple of the cost of an optimal solution in the long run, the erratic nature of call modification means that there may be opportunities for an intermittent process to exist, whose purpose would be to examine the existing tree and construct a set of modifications which would transform the tree into a near-optimal Steiner tree. The algorithm used for this process would be a modification of one of the previously mentioned heuristic algorithms, possibly limited by the number and type of routes that it would be permitted to alter. Alternatively, one of the heuristics could be used as they were presented and the whole multicast tree could be periodically torn down and set up again, probably closer to an optimum solution.

The timing of this optional process of improvement could be controlled by the source to avoid re-routing during transmission and the consequent risk of re-ordering. It might be encouraged by a possible reduction in the charge for the multicast connection in a WAN.

6 Conclusions

The proof that multicast routing was equivalent to constructing a Steiner tree, which is an NP-complete problem, lead to various heuristics being proposed, with worst case performance generally twice that of the optimum solution but with average performance much closer to the optimum. These heuristics were modified to optimise the tree for various parameters, all of which are important, but none of which is crucial in itself.

In the long-term operation of any real network which provides multicast capability, the dynamic nature of multicast applications is such that the previously proposed heuristic algorithms are likely to provide solutions whose success is only ephemeral.

The simulations in Section 4 imply that even one of the simplest routing algorithms can provide a mean in-

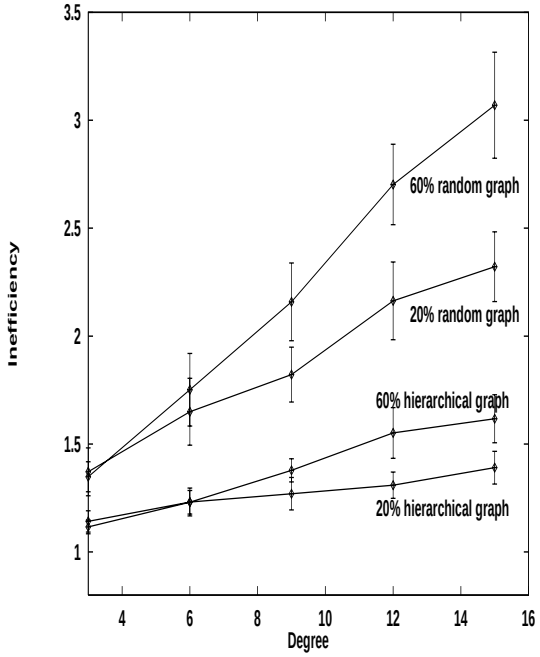

Figure 8: Mean Inefficiency for both network models and various degrees at 20% and 60% destination nodes/total nodes

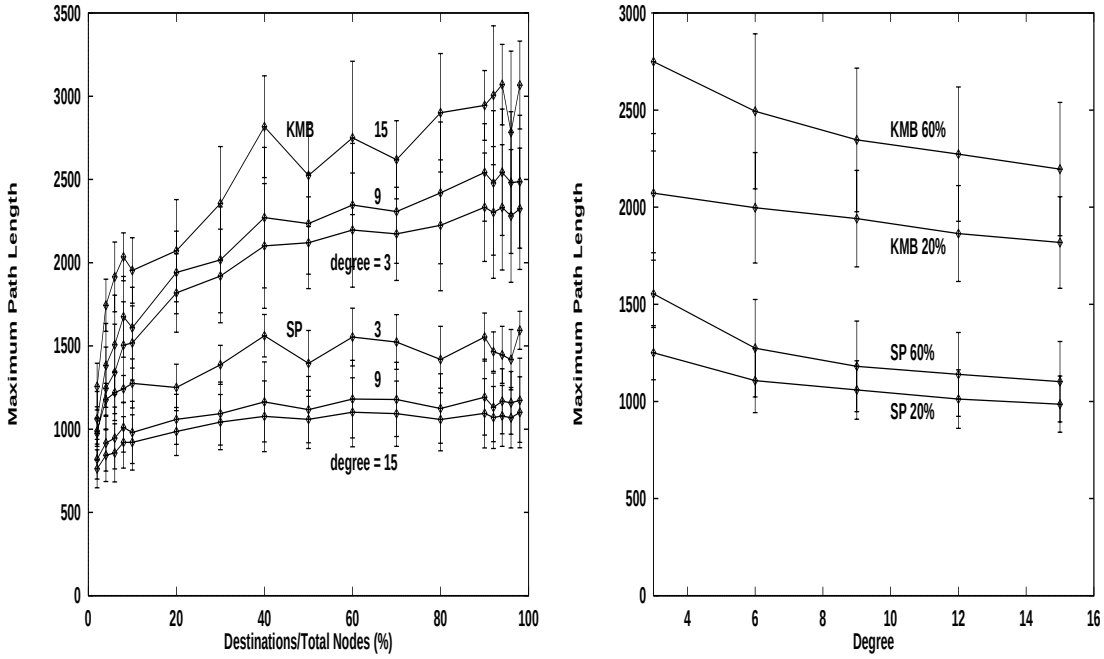

Figure 9: Maximum path length for Shortest Path (SP) and KMB trees with various percentages of destination nodes/total nodes and degrees

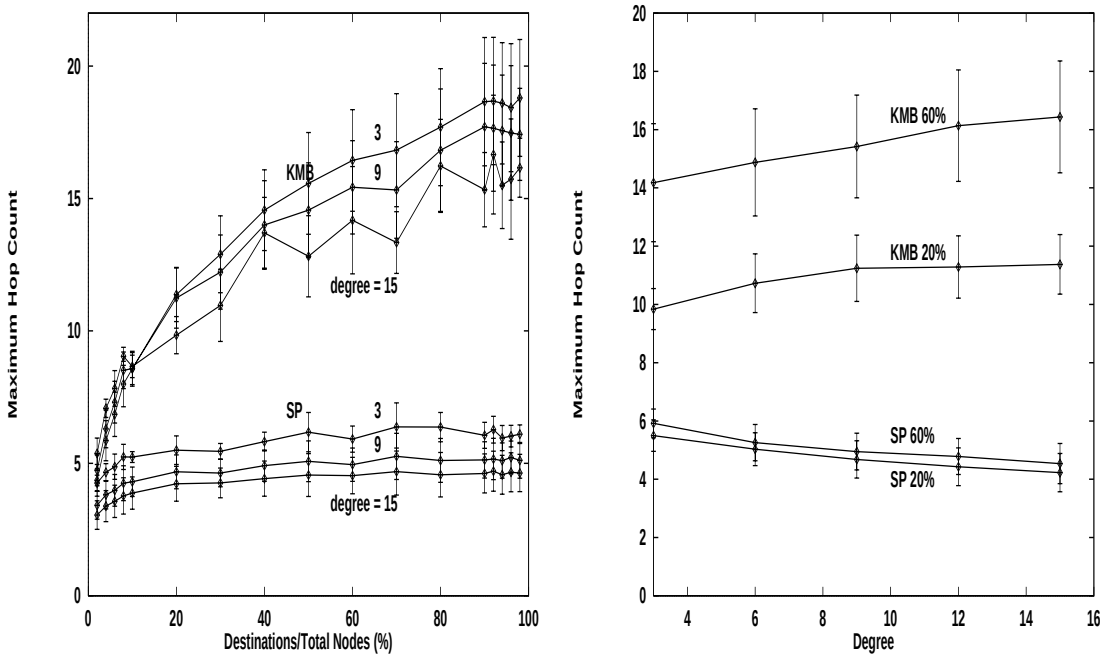

Figure 10: Maximum number of hops for Shortest Path (SP) and KMB trees with various percentages of destination nodes/total nodes and degrees

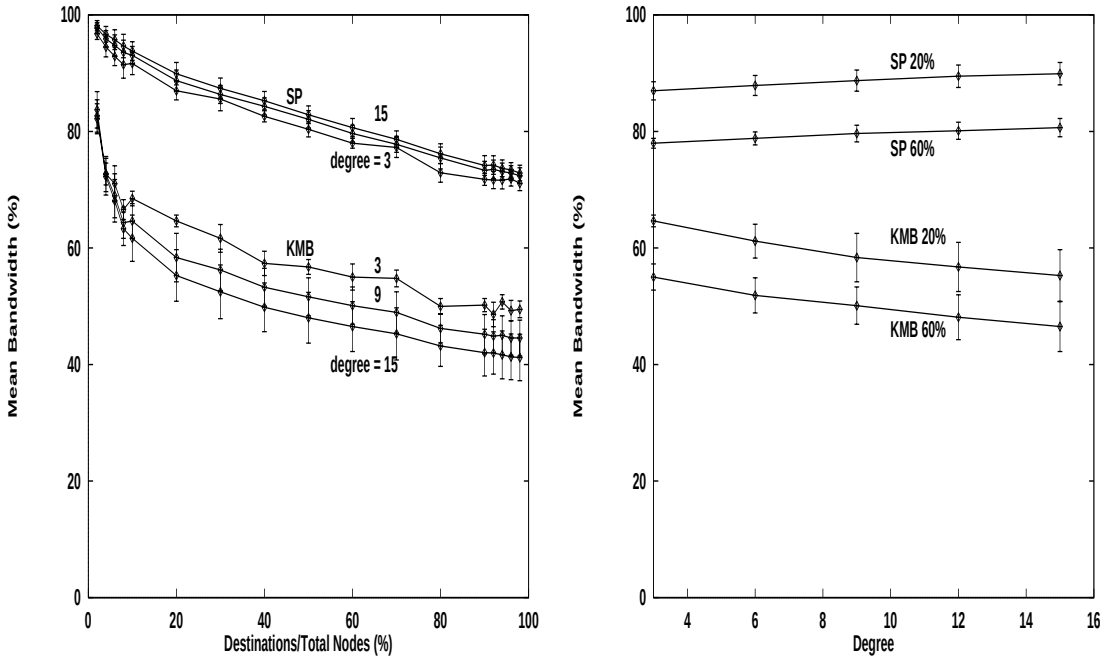

Figure 11: Mean bandwidth relative to unicast network, for Shortest Path (SP) and KMB trees with various percentages of destination nodes/total nodes and degrees

efficiency (for a mean number of three edges per node) which is within a factor of 1.5 of typical heuristic solutions, which themselves have an upper-bound of being within a factor of 2 of the optimal solution (and typically within a factor of 1.2). More realistic network models such as the hierarchical model indicate even better performance.

It is suggested that any future work in this area should concentrate upon the ability to add and remove connections from a multicast tree and attempting to improve the routes chosen in an incremental fashion, rather than constructing monolithic solutions to a problem specification that may only be valid for a unprofitably short duration. The further development of hierarchical random graph network models is also of interest for future studies in this area.

References

[1] C H Chow, *On Multicast Path Finding Algorithms*, Proceedings of IEEE Infocom 1991, Miami.

[2] H Tode, Y Sakai, M Yakamoto, H Okada and Y Tezuka, *Multicast Routing Algorithm for Nodal Load Balancing*, Proceedings of IEEE Infocom 1992, Florence.

[3] E N Gilbert and H O Pollak, *Steiner Minimal Trees*, SIAM Journal of Applied Mathematics, Vol.16, No.1, pp.1–29, 1968.

[4] R M Karp, Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R E Miller and J W Thatcher (editors), pp.85–103. Plenum Press, New York, 1972.

[5] P Winter, *Steiner Problem in Networks*, Networks, Vol.17, pp.129–167, 1987.

[6] V J Rayward-Smith and A Clare, *On finding Steiner Vertices*, Networks, Vol.16, pp.283–294, 1986.

[7] D Wall, Selected Broadcast in Packet-Switched Networks. In *Multicast Communication in Distributed Systems*, M Ahamad (editor), IEEE Computer Society Press, 1990.

[8] L Kou, G Markowsky and L Berman, *A Fast Algorithm for Steiner Trees*, Acta Informatica Vol.15, pp.141–145, 1981

[9] X F Jiang, *Multipoint Digital Video Communications*, PhD Thesis, University of Cambridge Computer Laboratory, 1991.

[10] Bernard M Waxman, *Routing of Multipoint Connections*, IEEE J. Selected Areas in Communications, Vol.6, No.9, December 1988.

[11] CCITT, Draft Recommendation I.150, *B-ISDN ATM Aspects*, Geneva, January 1990.

[12] Private communications with B M Waxman.

[13] Z Wang, *Routing and Congestion Control in Datagram Networks*, PhD Thesis, University College, London, January 1992.

[14] Aho, Hopcroft and Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1983.