Applying clustering algorithms to multicast group hierarchies

Gill Waters and Sei Guan Lim

Technical Report No. 4-03
August 2003

Applying clustering algorithms to multicast group hierarchies

Gill Waters and Sei Guan Lim[1]

Technical Report 4-03
Computing Laboratory, University of *Kent, Canterbury, Kent CT2 7NF, UK*
E-mail: A.G.Waters@kent.ac.uk

April 2003 (Revised August 2003)

Abstract

Multicasting offers group communication a considerable efficiency gain, particularly for large groups. As the size of the group increases, management and protocols become more complex and severe scaling problems can arise, for example in the routers at the network layer, for reliable file distribution at the transport layer and for large scale Peer-to-Peer file sharing or processing sharing systems at the application layer. Hierarchies help to reduce complexity, state space requirements and the number of messages exchanged between participants.

In this paper, we propose the use of clustering algorithms to help in determining hierarchical multicast trees. Clustering algorithms help to partition user populations according to a variety of criteria. Here, we principally consider proximity. We apply the k-means clustering algorithms iteratively to construct successive hierarchical levels. We examine the performance of the technique for multicast groups, first based on the geographical location of the participants and, secondly, based on the usual weighted graph representation of network topology. The results show that it is a promising technique for multicast tree construction. The report concludes with suggestions for applying the clustering technique in a variety of ways. Emphasis is placed on the transport and application layers to produce tree-based overlay networks.

Keywords: Multicasting; clustering algorithm; k-means algorithm; hierarchical; overlay networks; application-layer multicasting.

[1] Now at Department of Mathematics, Faculty of Science, Universiti Brunei Darussalam. The evaluation based on network topology was conducted as Sei Guan Lim's project for the MSc in Distributed Systems and Networks at the University of Kent

Contents

1. Introduction

In this report, we consider the application of clustering algorithms to the construction of large-scale overlay multicast hierarchies. The network layer is the obvious place to concentrate efforts on multicasting, as it can use links as efficiently as possible and does not need to transmit multiple copies down any network link. However, if we cannot assume that multicasting is available throughout the Internet or a large Internet, then it makes sense to consider creating multicasting structures at the application or transport layers. Such structures are called *overlay* networks and should be designed to minimise the impact on the network. Although they are less efficient than network-layer multicasting, they can offer better provision for security, more control over group membership and the opportunity to optimise the multicast trees based on the exact requirements of the application. This includes achieving scalability by appropriate hierarchical organisation.

Clustering algorithms are a way of partitioning data depending on its properties. In our case, we wish the partitions to reflect the distribution of the multicast application's user population so that we can identify closely located groups of users. The technique can be applied iteratively to find subgroups, thus building a layered hierarchy in which a parent system at one layer supports its children in the next layer.

There are many examples of applications that can use multicast overlays. These include:

- Groupware systems to aid collaborative design and evaluation.
- Large-scale dissemination of timely information e.g. for business and medicine
- Publish and Subscribe systems or other content delivery mechanisms
- Multimedia conferencing
- Real time video of international sporting events
- Peer-to Peer applications that share storage or processing capabilities
- Support for the Grid (large user communities in. e-Science, e-Medicine or e-Commerce) enabling them to share expertise, capabilities and information. See for example (Berman, Fox et al.).

Such applications may need several copies of a message to be sent over network links in order to reach all participants and can put a heavy load on the network. A good application layer overlay will minimise this load and indeed can still make use of network–layer multicasting in the pockets of the network where it is available.

We are interested in hierarchical multicast trees capable of scaling to very large groups. Our chosen strategy has the potential for adaptation to a number of different optimisation criteria. These include the number of levels in the hierarchy, the size of the clusters (and thus the number of children served), the efficient use of network layer links and the delay to the recipients.

In (Waters), we proposed clustering algorithms as a means of organising scattered regions of a network that were capable of handling advanced traffic types such as those with Quality of Service constraints. In this report, the work is extended to consider multicast hierarchies, using geographical location and population weightings. We also evaluate the performance of multicast hierarchies created using the usual weighted graph representation of the network topology and examine variations of the clustering algorithms that enable them to be tuned to different performance requirements. As we will demonstrate, our results lead us to believe that our clustering techniques can be efficient and of practical value.

In the following section, we discuss the advantages of hierarchies for multicasting; we also discuss the concept of overlay networks, their constraints and optimisation criteria and relate our work to other work on overlay networks. In section 3, we explain why we have chosen to apply clustering algorithms to the problem; the section also includes an introduction to clustering algorithms. In section 4, we describe our evaluations using the geographical location of group participants. This, of course, assumes that such information is easily available. In section 5, we examine the application of the techniques using standard weighted graph representations of network topologies. In the final section, we summarise our work, suggest the best ways of applying the techniques and discuss further work on applying clustering algorithms to constructing overlay networks.

2. Related Work

2.1 Multicast hierarchies

In general, hierarchies are needed to reduce the burden on routers of keeping large amounts of network state information and to reduce the number of signalling messages exchanged. Provided suitable hierarchies are available, full information need only be kept on systems within the same hierarchical level together with summarized information about adjoining levels. The technique is therefore particularly helpful for routing where large tables must otherwise be maintained. An example of hierarchical routing is ATM’s Private Network Node Interface (ATM-Forum).

For multicast applications, hierarchies are useful for network layer routing and at the transport and application layers. For example, scalable reliable file distribution can be achieved with a hierarchy of systems that provide error recovery on behalf of the users at their level (Reliable Multicast Transport). Real time video distribution is an example of an application that is likely to require optimised hierarchies to scale to very large groups.

A number of authors have considered techniques for networking hierarchies and, although several of these discuss clusters, they do not actually use clustering algorithms to group nodes into clusters as we propose. Most proposals are related to multicasting or to Web caching.

An approach to hierarchical clustering for multicast routing based on Voronoi diagrams is discussed by Baccelli et al (Bacelli, Kofman et al.). (A Voronoi diagram divides a plane into regions each based on a specified node. Within each region, any other node is closer to that region's specified node than to the specified node in any other region.) The technique divides the network into domains each fed by a core and cores are organized into a hierarchy of centre-based trees. Although this approach looks promising, their evaluation uses a uniform distribution of nodes, which may not be realistic. The authors found that optimum tree cost can be found by having much bigger fan-outs nearer the source of the multicast tree, with successively lower fan-outs at lower levels in the hierarchy, whereas practical systems may require similar fan-outs at each level.

Another hierarchical routing scheme is that of Chatterjee and Bassiouni who describe a two level hierarchy (Chatterjee and Bassiouni). Optimisation at the top level (linking the clusters) uses a minimum spanning tree and, at the lower levels, a broadcast tree is found with optimal delay within each cluster.

An example of a reliable multicast transport is the *TRAM* protocol (Chiu, Hurst et al.) with a hierarchical tree structure of "repair heads" for repairing failures in transmission to a multicast group. Tree organization uses advertisement protocols, expanding ring searches and dynamic distributed procedures. The IETF group on Reliable Multicast Transport (RMT) is putting together a number of building blocks which will enable the selection of different techniques suited to a variety of applications (IETF). At the 47th IETF meeting in Adelaide, tree building for RMT was raised as an important issue. Requirements include techniques for optimally organizing hosts into a tree and optionally, re-organizing the trees. Solutions should be capable of scaling to 10,000 receivers with fan-out sizes from 50 to 1000. It is likely that solutions will include top-down techniques (such as discussed in the optimisation approach of this paper) and bottom-up techniques (e.g. by receivers carrying out a local search for the nearest Repair Heads).

Web caching often uses hierarchical structures; these methods combine a method of detecting well-used pages with decisions about the hierarchy. For example, in the LSAM proxy cache (Touch and Hughes), their Intelligent Request Routing uses a neighbour-search operation to configure the proxy hierarchy. Another example is the Squid Proxy Cache (SQUID).

For hierarchical trees, there are many possible optimisation constraints. In this report, we concentrate on minimizing the number of levels in the hierarchy, minimizing the cost of the hierarchical multicast tree and minimizing the load on network links.

2.2 Overlay networks and optimisation criteria

There has been considerable interest recently in overlay networks, i.e. topological arrangements of end-systems that do not rely on knowledge of network-layer routing. Overlay networks can be at the transport or application layer and they access network layer capabilities only through standard network services. The case for end-system multicast was put forward Chu et. al. (Chu, Rao et al.). Although multicast capabilities are offered by many router providers, they are by no means universally installed and so cannot be relied upon in the Internet. Despite techniques for scalable routing e.g. (Estrin and others) (Ballardie), there is still a chicken–and-egg situation whereby applications that need multicasting are hindered by the lack of routing capability and routers are not made multicast-capable because of the lack of applications. It makes sense to organise multicast structures as overlays, provided they do not impose too great a burden on the network. The widespread use of such applications will demonstrate the need for better provision at the network layer. Application layer multicasting has additional advantages: optimisations can be geared to specific requirements and the likely distribution of users may be known in advance.

In this paper, we concentrate on hierarchical overlay structures to support multicast applications. Optimisation criteria include protocol overhead, state overhead, speed of reconfiguration, constrained delays and minimising the excess load on the network.

A number of important measures have emerged. The *Relative Delay Penalty (RDP)* or *stretch* of a path from a source node to a destination node is the ratio of the delay in the overlay network to the shortest delay achievable at the network layer. The S*tress* of a link is defined as the number of copies of the same message sent along that link, due to the overlay network. See (Banerjee, Bhattacharjee et al.) and (Chu, Rao et al.). Chalmers and Almeroth (Chalmers and Almeroth) extend earlier work by Chuang and Sirbu (Chuang and Sirbu) to define a value δ that normalizes the effect of stress throughout a multicast overlay tree.

$$\delta = 1 - \frac{L_m}{L_u} \qquad (1)$$

Here L_m is the sum of the links traversed in the multicast tree and L_u is the sum of the links traversed over all the unicast paths to the multicast group members. For very widely distributed groups with little sharing, δ tends to 0, i.e. there is little to be gained from multicasting. Where the multicast tree offers a very high degree of sharing, δ tends to1. Chalmers and Almeroth show that, over a wide range of groups in typical multicast usage, the trees exhibit a δ of $1 - N^{\varepsilon}$, where N is the number of receivers and ε is a figure between -0.34 and -0.3 (smaller for larger groups).

Proposed techniques for overlay network construction fall into three main categories: centralized, mesh first and tree first, as reviewed in (El-Sayed, Roca et al.). Examples of mesh first protocols are Bayeux (Zhuang and others), Narada (Chu, Rao et al.) and using Delaunay triangulation (Liebeherr, Nahas et al.). In some tree-first protocols e.g. HMTP (Chu, Rao et al.), a joining node contacts the root of the tree, chooses the nearest child and follows down the levels to find a suitable attachment point. For scalability, a number of protocols arrange nodes hierarchically such that a server at each layer supports a "cluster" of children. Examples include NICE which offers a compromise between stress and stretch (Banerjee, Bhattacharjee et al.) and derivatives e.g. Zigzag (Tran, Hua et al.) and the zone-based Scalable Adaptive Hierarchical Clustering (Mathy, Canonico et al.). Clustering in this sense has also been used for reliable multicast transport. Jennings et al (Jennings, Motyckova et al.) compare three graph-theoretic clustering algorithms, some of which rely heavily on Expanding Ring Search which can put an excessive load on the network. They conclude that optimal multicast trees need to have multiple levels of hierarchy and to strike a balance between the number of clusters and cluster size.

Out of these categories, our method is a scalable tree-based approach and our techniques aim to satisfy the interrelated requirements of stress, stretch, cluster-size and cluster number. Our current investigations are through centralized calculation, but should be adaptable to distributed versions. We also believe that maintenance of large multicast trees can be assisted by the presence of a control structure that can assist new members to join or leave. Such a structure can confine the necessary state overhead to that kept by a few application layer servers capable of carrying out the tree calculation.

3. Using clustering algorithms for overlay networks

Clustering algorithms are a method of partitioning a set of points based on the characteristics of the points; the aim is to produce clusters of points that are more similar to other points in the same cluster than to points in other clusters. Clustering algorithms have many applications to categorisation for example for biological taxonomy or the study of sociological data. They have been applied to image analysis and to vowel discrimination in speech and language analysis for speech recognition. The description of the use of clustering algorithms for backbone design in telephony networks in Cahn (Cahn) prompted our examination of their application to multicast hierarchies. The principal clustering algorithm that we use for our hierarchical multicast trees is the k-means algorithm that partitions a set of nodes into exactly k clusters.

We have chosen clustering algorithms firstly, for the flexibility they offer in dividing the receivers in different ways e.g. by the maximum number in a cluster, by the total number of clusters, by limiting the number of sub-clustering steps to control the depth of the tree. Secondly, clustering has worked successfully in previous networking design scenarios. We believe that clustering algorithms can be particularly useful in high level design e.g. for overnight reconfiguration for file or content distribution. Such clustering can also reflect large distributed communities and populations, as occur in Grid Computing.

3.1 An introduction to clustering algorithms

There are two main types of clustering algorithm: hierarchical and non-hierarchical. See for example (Sharma). A hierarchical clustering algorithm assigns each node in turn to an appropriate cluster, then repeatedly merges two nearby clusters until all points belong to a single cluster. (Note that the term *hierarchical* is used in a different sense when we discus multicast hierarchies.) Non-hierarchical clustering algorithms typically divide the set of points into a given number, k, of clusters based on a given starting set of cluster-centres and then iterate to optimise the membership of the clusters.

We show each of these techniques applied to the small empirical set of sociological data shown in Fig 1.a), which has three obvious visible clusters. In the *hierarchical technique*, we start with each point in its own cluster; each new clustering is obtained by merging the two clusters that have the nearest pair of neighbours. The clusters are built up as follows:

0) {A},{B},{C},{D},{E},{F};
1) {A},{B},{C,D},{E},{F};
2) {A},{B},{C,D},{E,F};
3) {A,B},{C,D},{E,F};
4) {A,B},{C,D,E,F};
5) {A,B,C,D,E,F}.

The dendrogram of Fig 1.b) (Sharma) shows the order of merging of the clusters, the y-axis being the Euclidean distance between the nearest neighbours at each step. The data from the dendrogram can be used to decide which of the series of clusterings is the most suitable for the application.

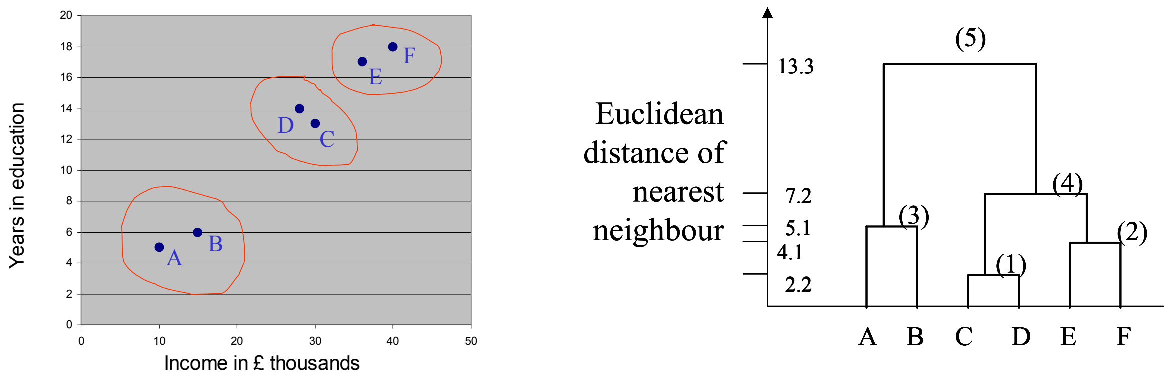

Figure 1. Hierarchical clustering a) Data set; b) Dendrogram

An important example of *non-hierarchical clustering* is the k-means algorithm, that partitions a data set into k clusters as follows:

1. Specify k nodes as initial centres
2. Allocate each node to the cluster containing the nearest centre to the node
3. Re-calculate the centre of each cluster.
4. Repeat steps 3 and 4 until solutions converge (or for a maximum number of rounds).

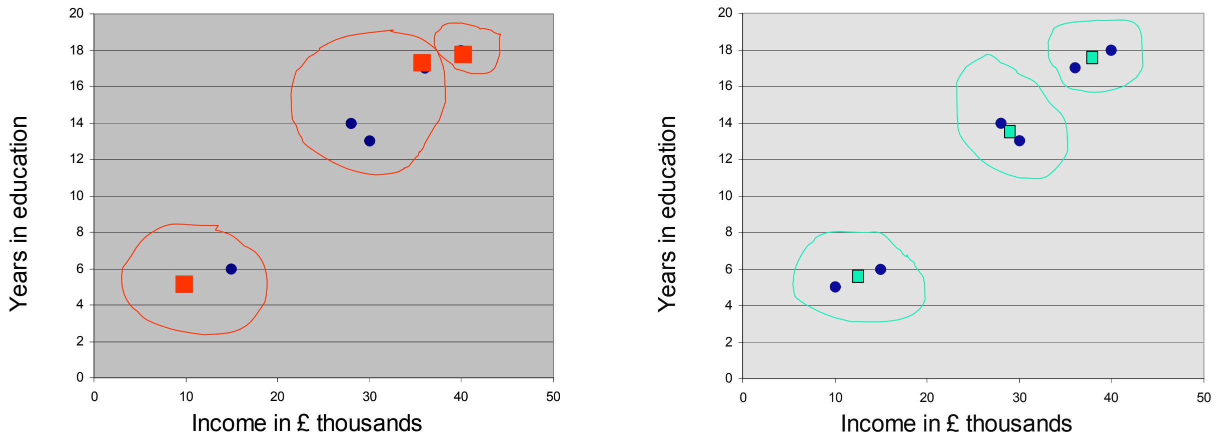

Fig. 2. k-means clustering(squaers indicate centres):
a) With initial centres; b) converged (showing final centres)

Performing k-means clustering on the same empirical data set with k=3 and with initial centres A, E and F (Fig. 2a), it takes three rounds to confirm the final clustering shown in Fig 2b).

For each type of clustering algorithm, we can choose from a number of options, for example, to find the nearest neighbor or to optimise cluster cohesion. One recommended technique is to determine a suitable number of clusters by evaluating the results of a hierarchical technique and then to optimise for this number of clusters by using a non-hierarchical technique. In applying clustering algorithms to network topologies we use k-means clustering. Hierarchical clustering is used determine the initial k centres for step 1 of the k-means algorithm. In our evaluations, we examine the effects of choosing different options for the algorithms.

3.2 Multicast tree formation using clustering algorithms

We form our hierarchical trees by performing k-means clustering at each hierarchical level and choosing a representative member to act as a *server* for each of the k clusters found. Each cluster may then be decomposed using k-means again to form a new layer of sub-clusters, whose servers become children of the server in the cluster just partitioned. The top-level servers become children of the source. The process is repeated until a suitable terminating condition is reached, as discussed below. The non-server members of any cluster that has not been partitioned become children of the cluster's server and are leaf nodes in the tree. Servers receive multicast messages from their parent and pass them on to their children; they may also perform other functions (e.g. retransmission for reliable protocols). We call our technique KMC (K-Means Clustering).

Fig. 3 shows an example of the tree formation for a multicast group with source s, using k=4. The first application of k-means results in four clusters. The nearest multicast member to the centre of each cluster is chosen to act as a server for that cluster. At level 1, these servers are a, b, c and d. The cluster with centre d is then further decomposed into four clusters whose servers (at level 2 in the tree hierarchy) are e, f, g and h.

We have not found other work studying k-means clustering for multicast trees, but related work has been reported for knowledge discovery e.g. by mapping different views of clusterings (Lazarevic and Obradovic) and for spatial data mining, e.g. using k-medoids (Ng and Han). K-medoids extends k-means by examining possible member substitutions that enhance clustering cohesion.

Although our techniques are centralised, we also plan to look at distributed versions. Typical proposals for distributed k-means rely on local algorithms that summarise statistics, resulting in a much-reduced central calculation. Application examples include distributed data clustering (Forman and Zhang) and managing disconnection in a mobile network (Ramanaiah and Mohantny).

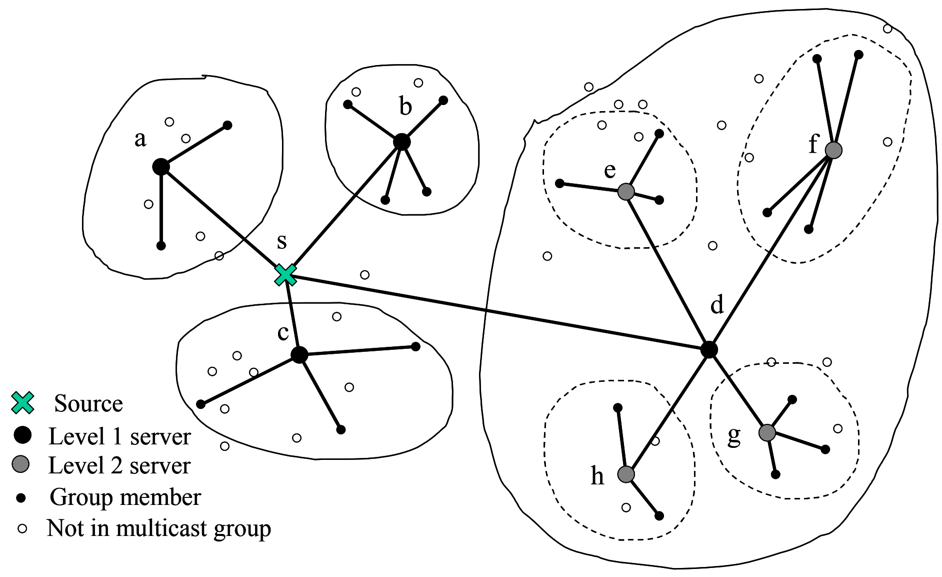

Fig.3 Forming the multicast tree using clustering algorithms

In this report, we describe two sets of experiments that examine the applicability of k-means clustering to finding hierarchical multicast overlay trees. Our preliminary experiments, described in section 4, clustered the nodes based on their geographical location. This technique would be suitable for a high-level arrangement of a large known user community. The second set of experiments, described in section 5, examined applying clustering to network topologies using the shortest path between end-systems as a metric; this scenario can be applied to application-layer multicast trees; the study compared different choices of options for the k-means heuristic.

4. Evaluations using geographical locations

The easiest way to perform the clustering is to use a Euclidean space to determine the closeness of the systems considered. In the first set of evaluations, we used geographical co-ordinates as our Euclidean base. (This in fact assumes a plane where each degree of longitude and latitude is treated as identical and does not take account of the spherical nature f the earth's surface.) If we the distribution of Internet users, they are concentrated on major cities and are certainly not evenly spread around all possible physical locations around the globe. It is therefore appropriate to consider the use of clustering to determine the general location of users. Although the geographical location of *all* Internet users may not be available, it is probable that the geographical location of the members of a global Intranet community would be known.

In section 5, we examine the application of clustering algorithms to network topologies. We believe that the geographical approach is applicable at higher levels in a multicast hierarchy with the topology being applied to achieve more accuracy on the local scale. It may also be applicable to a group with widespread multicast participation at the backbone layer in a large network.

Evaluations were carried out on two principal data sets - both consisting of the geographical locations of North American cities. The first set has 51 cities, all equally weighted. The second set has 46 cities, weighted to the population. The data sets are included as Appendix A.

4.1 Evaluations using 51-city data set

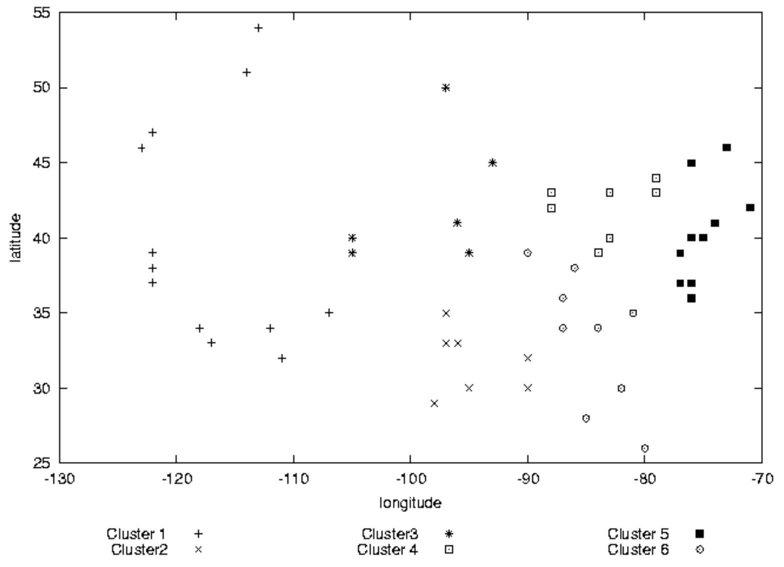

Fig. 4. Six clusters from 51 cities with initial centres distributed throughout the range

Our first evaluations create broadcast trees with different numbers of hierarchical levels. Figure 4 shows the 51-city data set partitioned into six clusters using k-means and with the intial centres distributed throughout the range of city locations. The next few figures show sample multicast trees constructed using our technique. Our implementation of the k-means algorithm returns the point at the "Euclidean centre" of each cluster i.e. the point that minimises the average distance over all cluster members. The nearest city to this centre is taken as the city that serves the other cities in the cluster; this server is also the attachment point to the root of the tree (or to the server at the next higher level for hierarchies with greater than two levels). In these examples, the root of the tree is calculated as the Euclidean centre of the six servers.

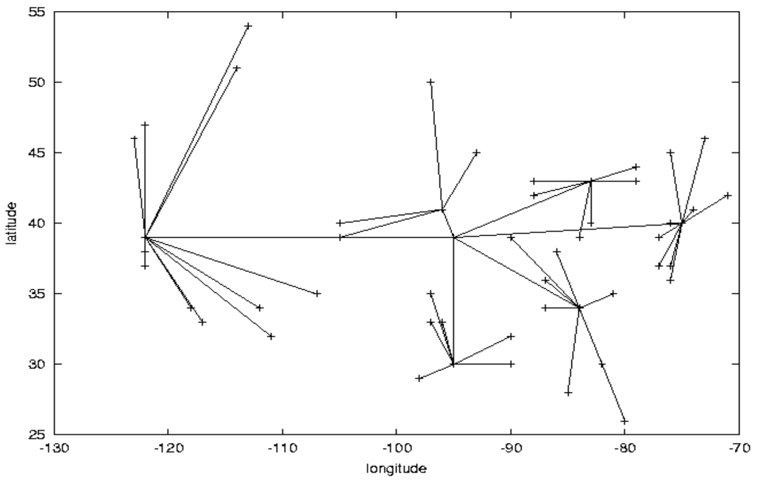

Fig. 5. Two-level hierarchical multicast tree from clusters of Fig.3

Figure 5 shows the two-level hierarchical tree formed by this technique. For clusters with greater than seven members (a server plus six other cities), k-means is used again to obtain the three-level hierarchy shown in Figure 6. Here, the number of sub-clusters is chosen to try to keep the number of members of each sub-cluster within the fan-out of 6. This produces a tree with a maximum of six children of each server (except possibly at the lowest layer) at a maximum of three hops.

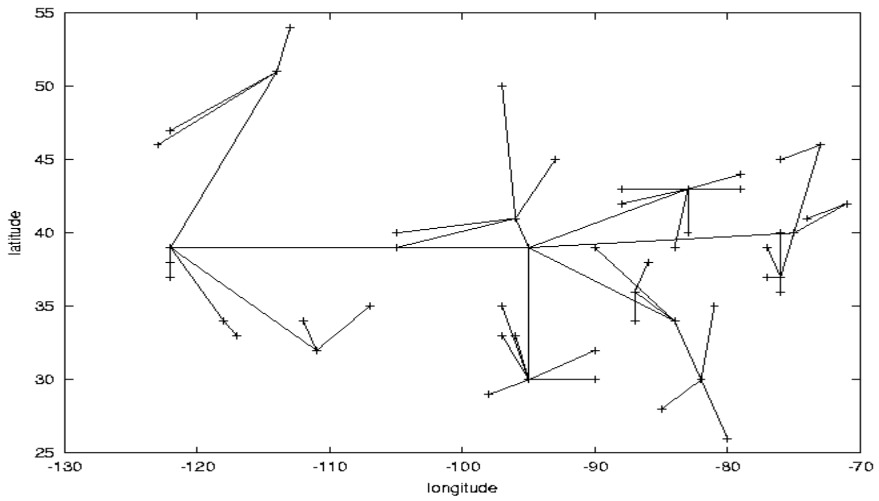

Fig. 6. Three-level hierarchical multicast tree from clusters of Fig.a

The k-means clustering algorithm is a heuristic technique and is influenced by the choice of initial centres. This can be seen in Figure 7, which shows a two-level hierarchy produced from six initial centres from the South West of the range of locations. Clusters and their centres can and have changed during the execution of the kmeans algorithm, but the resultant cluster centres farthest away from the starting centres are large. Cluster sizes with this starting set range from 2 to 19, whereas the tree shown in Figure 4 has even-sized clusters ranging from 6 to 12 members.

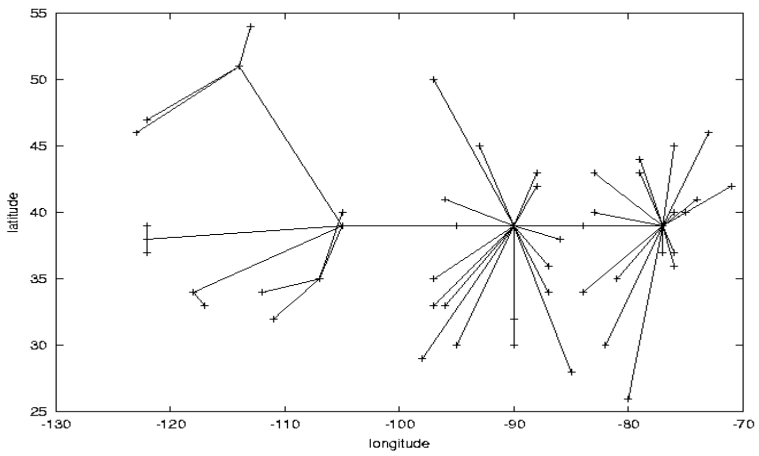

Fig. 7. Two-level hierarchical tree for 51 cities, using six SW cities as starting centres

For the South-West initial centres, k-means converges only after twelve iterations; for the set of initial staring centres used in Figs a and b, only three iterations are needed. We can therefore conclude that the choice of initial centres is crucial, not only to the quality of the clustering formed, but also to the time taken to achieve a solution.

Initial centres	**Source**	**number of levels**	**cost**
(No clustering)	Overall centre	1	749.64
6 distributed throughout data set	Centre of first level servers	3	296.34
As above	As above	2	345.53
As above	As above	1	768.82
6 in South West	Centre of first level servers	3	316.02
As above	As above	2	373.06
As above	As above	1	953.96

Table 1. Costs for 2 and 3 level hierarchies using 51- city data set

Table 1 shows the results of using KMC to find a broadcast tree for the 51-city data set using k-means with k = 6. Two selections of initial centres were chosen. The server of each cluster is the city nearest to the geographical centre of the cities in the cluster. Clusters were only subdivided if they contained more than seven members (a server plus six other cities). The two- and three-level trees have depths of two and three hops respectively. The source and each server in the tree have a maximum of six children, except at the lowest layer. The cost of the tree is the sum of the edge-lengths, which are calculated as the Euclidean distance between a city and its parent in the tree. The one-level hierarchy is simply a star directly connecting the source to each of the cities in the map. Results for both two and three levels of hierarchy are shown.

The k-means clustering algorithm is a heuristic technique and is influenced by the choice of initial centres. Table 1 shows that better results were obtained using cities distributed throughout the map than taking six cities located in the South-West corner. The first-level clustering for the distributed choice of centres resulted in reasonably even-sized clusters ranging from 6 to 12 members, whereas the clusters formed from the centres located in the corner had sizes ranging from 2 to 19 With distributed initial centres, the two-level hierarchy offers substantial gains of approximately 55% over a single level and the three-level a further improvement of 14% over the two-level hierarchy.

4.2 Evaluations using the 46-city data set with population weights

To investigate the effect of large city populations, the second data set has 46 cities, each weighted at one unit for a population of approximately 500,000; this gives a total population weighting of 100 over all the cities. The data set is illustrated in Fig. 8 with the population sizes indicated by the size of the circles shown centred on the cities.

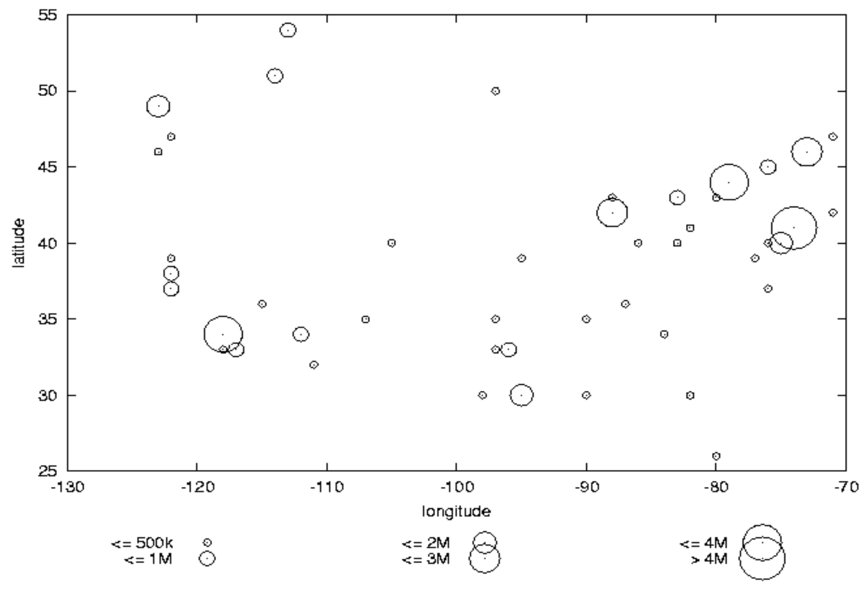

Fig. 8. Geographical location of 46 American cities showing population weightings

A modified version of KMC, which calculates its centres for k-means based on the weight of the nodes under consideration was used for this data set. The resultant three-level broadcast hierarchy is shown in Figure 9. Starting centres are distributed randomly throughout the data set and the source is positioned at the geographical centre of all the first-level servers. The resultant tree reflects the population distribution well and this encouraged us to continue with further work on clustering algorithms.

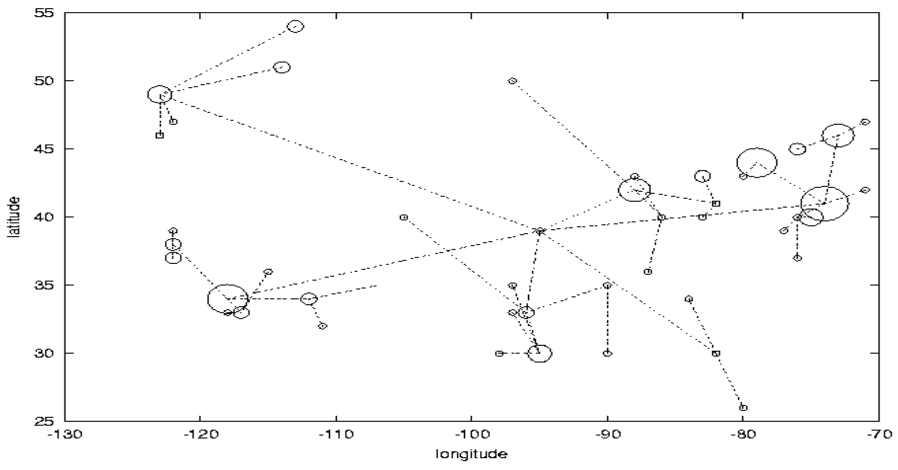

Fig. 9. Three-level hierarchical tree using weighted k-means clustering based on the geographical location and population of 46 American cities

From these simple results, we can get an indication of the stress and stretch inherent in multicast trees produced in this way. Fig. 9 shows that, in the case considered, the maximum number of children is six throughout the tree. In the evaluations, we always assume that there is a link of weight the Euclidean distance between any pair of cities. This means that a single copy of a packet would be needed along any link in the tree (giving a stress of 1). In most practical networks, restricting the number of a server's children to six implies a maximum stress of 6, but, in practice, lower values can be expected.

If we compare the lengths of the paths from the source to each city in the tree with the Euclidean distance from the source to that city, we get a measure of stretch. Looking again at Figure 9, we can calculate approximate values of stretch for the South-West cluster and its descendants between 1 and 2.7; for the five remaining clusters, the stretch is between 1 and 1.8. We would expect the technique to give similar results on real network topologies, but defer this discussion to section 5.

4.2.1 Effects on random multicast groups using geographical location

The above findings are based on single runs for broadcast trees. We next evaluated multicast trees with a randomly selected source and random groups of between 20 and 100 users located at the cities in our 46-city data set. The weighted node technique was again applied and the results are averages of 100 samples at each group size. Initial cluster centres are the same in all cases and are distributed throughout the 46 cities.

Figure 10 compares one, two and three-level hierarchical trees averaged over all 100 samples for each group size, using a fan-out of 6. The total cost of a tree uses the geographical distance, as described above for the 51-city data set. The figures are normalised to the cost of a minimum spanning tree connecting the source to the destinations. (Note that the cost of connecting nodes within the same city is taken as zero based on this measure; this is true for all the trees used in the comparison.)

The results again show a substantial reduction in cost from a single level tree. The two-level hierarchy costs rise steadily but slowly with group size. The three-level hierarchy can be expected to give a cost close to that for the two-level hierarchy for small group sizes, as in many cases it will not be necessary to form sub-clusters at all.[2]

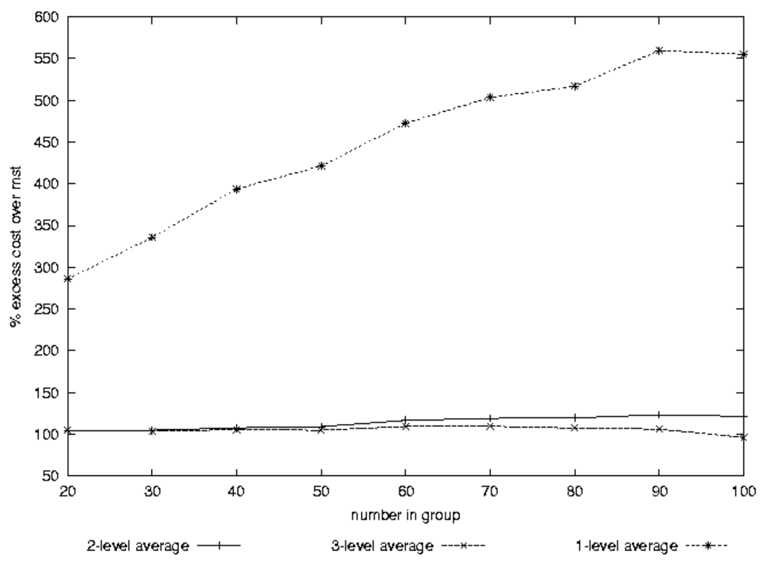

Fig. 10. Average costs of one-, two- and three-level trees normalised to cost of minimum spanning tree

As the number of nodes increases, the three-level hierarchy outperforms the two-level hierarchy and this effect is more noticeable with larger groups. This is due to the relationship between the fan-out size and the group population and is not an effect of comparing dense or sparse groups, although it can be seen that the technique works well for both sparse and dense groups within our evaluation.

Both the two and three level costs are roughly twice the cost of the minimum spanning tree. Note that the hierarchical tree costs can be expected to be much more than those for the minimum spanning tree, as each tree reaches every destination within two or three hops for these hierarchies. In contrast, a typical example of a minimum spanning tree of all 46 cities took 20 hops to reach one of the destinations. Here we are optimising first for a hierarchical structure with a maximum number of levels and secondly for cost. The technique can therefore be expected to offer low delay to all destinations.

Figure 11 shows the maximum and minimum costs achieved at each group size out of the 100 samples, for each of the hierarchies. These represent groups that happen to be particularly widely-spaced or closely spaced respectively. For more dense groups, groups are more likely to be cohesive and there is less variation.

[2] Note that if all clusters were of even size, which cannot be guaranteed using the technique described, a two-level hierarchy could support six clusters each of seven nodes (six nodes plus the centre) i.e. 42 nodes.

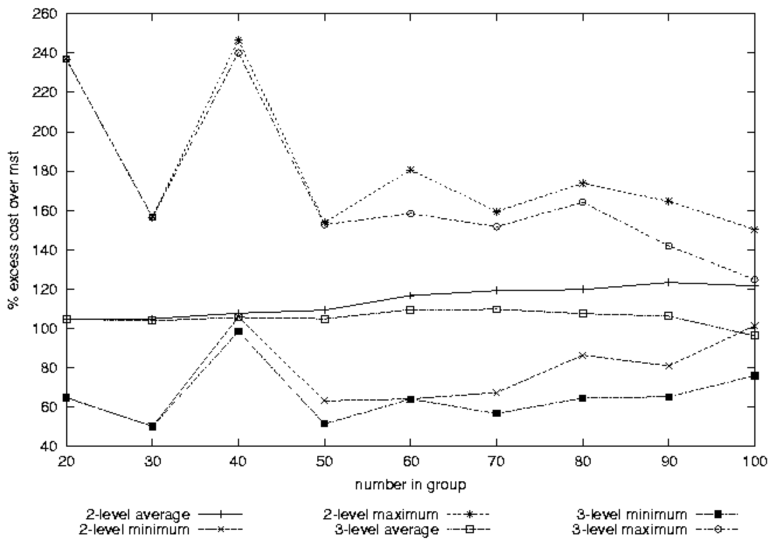

Fig 11. Average, maximum and minimum costs over 100 samples: two- and three-level trees

Figure 12 compares a fan-out size of 5 with a fan-out of 6 (with k set to 5 or 6). The results show greater variation between two- and three-level hierarchies with a fan-out of 5. This is to be expected, as the three-level hierarchy is better able to take advantage of sub-clusters. For most group sizes, the two-level hierarchy costs for a fan-out of 5 are greater than the costs for a fan-out of 6. The three-level fan-out of 5 however outperforms all the remaining curves. This indicates that careful choice of fan-out and number of hierarchical levels is required.

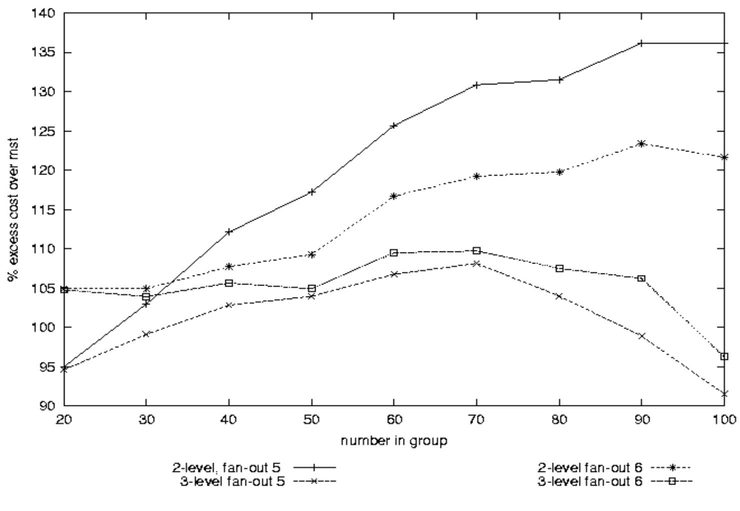

Fig. 12. Comparison of fan-out sizes of 5 and 6 for two- and three-level hierarchies

.

4.2.2 Computational cost

The computing cost of k-means is O(n*k) for each iteration, where n is the number of nodes under consideration and k is the number of clusters. It can be halted after a fixed number of iterations if convergence has not been reached. In the experimental runs used for Figures 10 and 11, k-means converged in between 2 and 6 iterations; a more detailed breakdown can be seen in Figure 13. By incorporating several nodes into a weight at a single geographical location, we reduce the time take to calculate the tree and it may not be necessary to recalculate the tree every time a node joins the group.

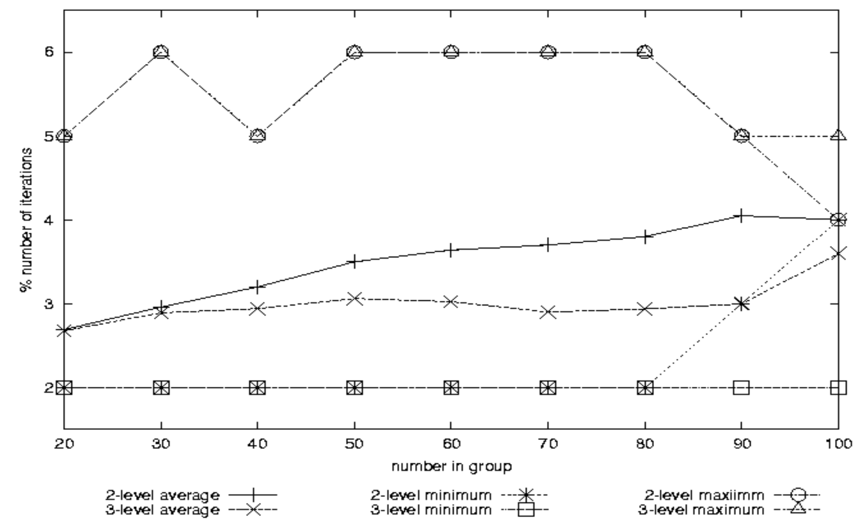

Fig. 13 Average, maximum and minimum number of iterations in k-means for Figs 7 and 8

4.3 Summary

The results show the benefit of using hierarchical levels not only for reducing message and state information, but also in providing trees at reasonable cost. The evaluations also show that KMC trees are applicable for a range of group sizes. The stretch can be contained by limiting the number of levels in the hierarchy. The evaluations use clustering based on weighted nodes that *represent* the number of users at each geographical location[3] and which may be particularly useful in supporting distributed communities working on a common project as in the Grid community.

Unfortunately, geographical location may not be available; our remaining evaluations are therefore all conducted using more realistic network topologies.

[3] We have also carried out some preliminary work in which multiple user nodes were placed at the same location. This was found to be less effective, as large numbers of nodes can skew the decisions on sub-clustering.

5. Evaluations using network topology

5.1 Evaluation context

The previous evaluations were based on simple geographical information. For a more realistic view, we have examined the construction of hierarchical multicast trees based on more readily available network topology information. The evaluations described constitute the main findings in Guan Lim's MSc thesis (Lim).

In these evaluations, we assume that application users can measure their round-trip delay to other users. We assume source-based trees and apply KMC clustering to the multicast receivers only. The delay between potential users is the shortest path between any two nodes in the network and its distance matrix can be found using Floyd's all-pairs shortest path algorithm (Floyd). At the top layer, k clusters are; for lower layers, the clusters are partitioned only if their size, minus one for the server, is more than the specified *maximum allowable number of child nodes, c*. The process continues until all lowest level clusters have at most c child nodes. The resultant tree will have a maximum stress of k above the lowest level, or c at the lowest level. All parent-child connections in the tree follow the shortest path between the parent and the child. The stretch is related to the number of levels in the hierarchy and depends on the distribution of the nodes.

In the following sub-sections, we discuss the options for the variations of KMC that we considered: the method of selection of the initial seeds, assigning nodes to clusters and the choice of server from each cluster or sub-cluster.

5.1.1 Selecting the k initial seeds using hierarchical clustering

Methods of selecting the initial seeds for k-means clustering include:

i. Selection based on expert opinion in the subject area; it is too early in the analysis of the technique to apply this selection method.
ii. Techniques that maximise the distance between seeds; these are quite straightforward, but the usefulness of the seed set depends on the distribution of the data set (Mirkin) (Hartigan).
iii. Random selection; this is easy to implement, but often gives poor performance (Sharma).
iv. Hierarchical clustering; this can reflect the dataset under consideration and tend to produce well-spaced clusters (Sharma)

Because of its advantages, we choose hierarchical clustering. The clustering is applied until k clusters are found and the centres of these clusters are the seeds used. At each iteration, the two nearest (least dissimilar) clusters are merged. We have evaluated our clustering techniques using the following three methods for computing the dissimilarity D(I,J) between cluster I and cluster J, where d(i,j) is the distance measure between i and j.

1. *Single-linkage method:* D(I,J) = $\min_{i \in I, j \in J} d(i, j)$ the minimum dissimilarity between all possible pairs of entities in the two clusters.
2. *Complete-linkage method:* D(I,J) = $\max_{i \in I, j \in J} d(i, j)$ the maximum dissimilarity between all possible pairs of entities in the two clusters.
3. *Average-linkage method:* D(I,J) = $\sum_{i \in I, j \in J} d(i, j) / |I||J|$ the average dissimilarity between all pairs of entities in the two clusters.

The strengths and weaknesses of hierarchical methods are extensively discussed in the literature e.g. (Sharma) (Mirkin). Single-linkage is more prone to the formation of unbalanced, straggly and elongated structures, especially in large datasets. Complete-linkage, however, tends to find compact clusters with uniform diameters; it is also less affected by outliers. Single-linkage tends to produce clusters that are isolated but less cohesive; whereas complete-linkage tends to produce very cohesive clusters that are not quite isolated from others. Average linkage offers a compromise between the two.

5.1.2 Assigning nodes to clusters

At each iteration of k-means, the new set of cluster centres (or *centroids*) is computed and the entities are reassigned to these centroids. In our evaluations, a clustering has converged when the cluster sets of two successive iterations are identical; we also have a maximum allowable number of iterations. In practice, any oscillation between two clusterings can be established by checking back over previous generations.

We have run our tests with the following three methods of computing the centroids. Let I be the whole dataset and S be a cluster where $S \subseteq I$. A centroid is defined as follows:

1. *Average-within* A centroid is an entity $i \in S$ minimising the dissimilarity $diss(i,S) = \sum_{j \in S} d(i,j)/|S|$
2. *Nearest-within* A centroid is an entity $i \in S$ minimising the dissimilarity $diss(i,S) = \min_{j \in S} d(i,j)$
3. *Farthest-away* A centroid is an entity $i \in S$ maximising the total dissimilarity $diss(i, I-S) = \sum_{j \in (I-S)} d(i,j)$

5.1.3 Choosing the servers in the hierarchical tree

Two separate methods of finding the servers are implemented. These are:

1. The *most-central-receiver* server is a receiver that is closest to the receiver most distant from it (Aho, Hopcroft et al.) (Evans and Minieka). It is defined as follows:
   Let R be a set of receivers. $r \in R$. The centre of R is a receiver of minimum eccentricity, where
   the eccentricity of $r \in R$ is $\max_{j \in R - \{r\}}$ {shortest path from j to r}

2. The *closest-to-parent* server is the receiver in the cluster that has the shortest path to the parent in the hierarchical tree.

5.2 Evaluation environment

We have evaluated KMC on random networks using the Waxman model (Waxman) modified according to Doar and Leslie's suggestions (Doar and Leslie) by scaling the probability functions according to the size of the network so that the models do not have unrealistically high node degrees. We used visualization techniques to assist in developing and verifying our programs.

Our simulations used 20 samples of 100 node networks and for each of these there are 10 randomly chosen multicast groups for each group size. In this paper we present the principal results from the full set of evaluations covered in (Lim).

The full evaluation considered the following parameters.

a) Weighted and unit edged networks (to examine both delay and hop counts);
b) Broadcast trees and multicast groups of different sizes;
c) All nine combinations of the three seed selection techniques for the k initial seeds with the three methods of k-means centroid calculation. We label our methods
   KMC-XY, where
   $X = S$ (Single-), C (Complete-) and A (Average-) linkage for finding the initial centres and
   $Y = A$ (Average-within), N (Nearest-within) and F (Furthest-away) for centroid calculation.
d) Tree building using both most-central receiver and closest to parent receiver as the centre;
e) Different values of k; (unless otherwise indicated c is also set to k)
f) Multi-level clustering (as described above) and two-layer clustering;
g) Servers-to-receivers ratio;
h) Number of k-means iterations.

The principal performance metric was the normalised value δ defined in Chalmers and Almeroth (Chalmers and Almeroth) and shown in Equation (1) discussed in section 2. This definition of δ is for unit-edged networks. For weighted-edge networks we used the following definitions of L_m and L_u . L_m is the total cost of the application multicast overlay (the sum of the shortest paths of all the receiver-to-server conections in the tree). L_u is the sum of the weights of the edges in the end-to-end shortest paths from the source to every receiver.

5.2.1 Implementation of Dijkstra's algorithm

Some of our graphs show the results of finding an overlay tree using Dijkstra's algorithm alone. As expected, this starts by finding the shortest path to each receiver. If we were to take the cost of the resultant tree by counting the weight of each constituent link once only, we would get the cost of a network-layer multicast tree in which we assume that all nodes are multicast-capable. Instead, our "Dijkstra-based" cost is of an application layer multicast tree. Only members of the multicast group under consideration can act as servers. The servers will then repeatedly unicast to their children in the overlay tree. The outgoing links from a server will be traversed at most n times, where n is the number of children of the sever in the overlay tree. Such a tree may be able to achieve solutions that are more efficient than our KMC trees, but is likely to carry higher stress, especially on the links directly connected to the source.

5.3 Evaluation results and discussion

5.3.1 Weighted edge networks: multicast efficiency

Figure 14 shows the values of δ achieved using multi-level networks with k = 4 and with multicast trees constructed using the most central receivers as servers, for the nine cases mentioned in c) above. In Figure 15, we show the four best performing and the one worst performing case. The best three all use average-within centroid calculation, the fourth is KMC-CN and the worst is KMC-SF.

Each case exhibits similar characteristics throughout the range and KMC-CA gives about 9.4% improvement in δ over KMC-SF at a group size of 80. The choice of centroids is the most important factor for efficiency, in order from best to worst: A, N and S, then for each of these three, the choices are, in order from best to worst: C, A, S –linkage when determining the k seeds using hierarchical clustering. The efficiency increases with group size, demonstrating how dense groups can take advantage of shared transmissions high up the tree.

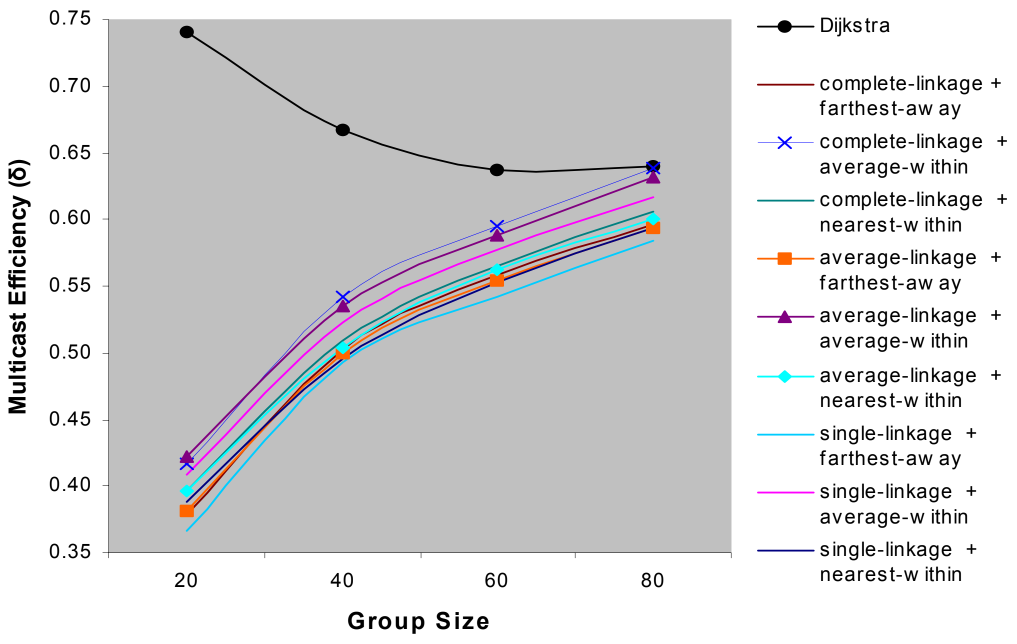

Fig. 14. Efficiency of all 9cases studied for weighted edge networks, k=4

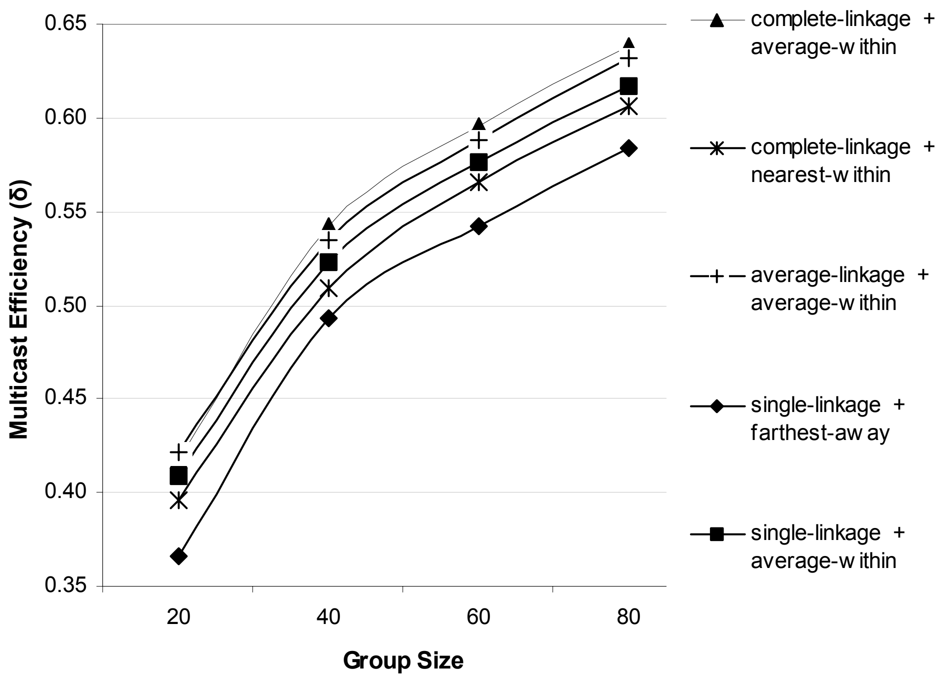

Fig. 15. Efficiency of clustering algorithms: best 4 and worst performing

Figure 16 shows the effect of using *closest to parent centres* as servers and exhibits similar overall efficiency trends for most of the cases studied. Comparing Figure 16 with Figure 14, we see that using most central receivers as servers gives between 1.2% and 4.3% efficiency gain over using closest-to parent servers. This effect is expected to be more marked for large values of k (larger fan-out) as the most central vertex minimizes every path from the server to its child receivers in the cluster, whereas the closet-to-parent vertex only minimizes the path from the server to its parent.

For *broadcast* trees (where all 99 non-source nodes are receivers) the same ordering of the nine cases was apparent, with results for δ ranging from 0.609 to 0.662, better than for all the multicast groups shown in Figure 14.

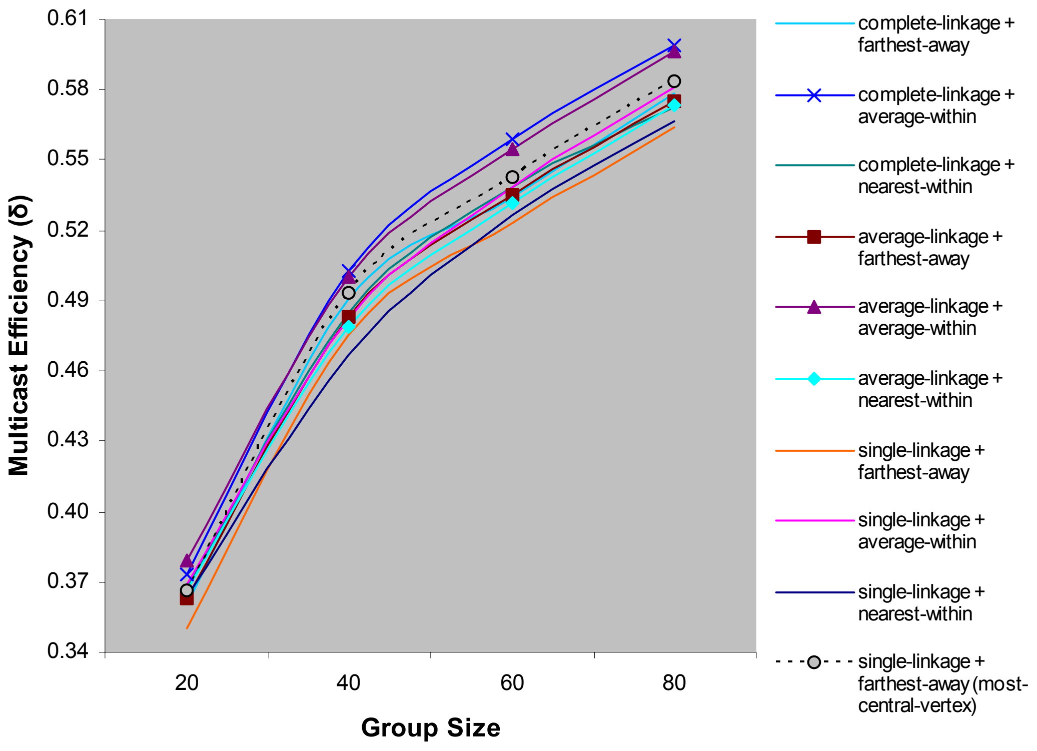

Fig. 16. Efficiency of all 9 cases studied for weighted edge networks (closest to parent centres)

5.3.2 Weighted edge networks: number of clusters and number of centres

To investigate the influence of the value of k on our results (i.e. the number of clusters found at each layer in the KMC hierarchy), we examined the broadcast case. Figures 17 and 18 show the results for values of k ranging from 4 to 10, using the same nine and five cases as for Figures 14 and 15 respectively. This shows that the efficiency gain decreases as the number of clusters per level increases. Trees with higher fan-outs tend to be short and trees with smaller fan-outs tend to be taller (in our data, trees with 4 clusters per level had a maximum depth of 7 levels). Tall trees tend to increase sharing of links and hence efficiency over the shorter trees; see also (Chalmers and Almeroth).

We would expect similar results for multicast to those of the broadcast case.

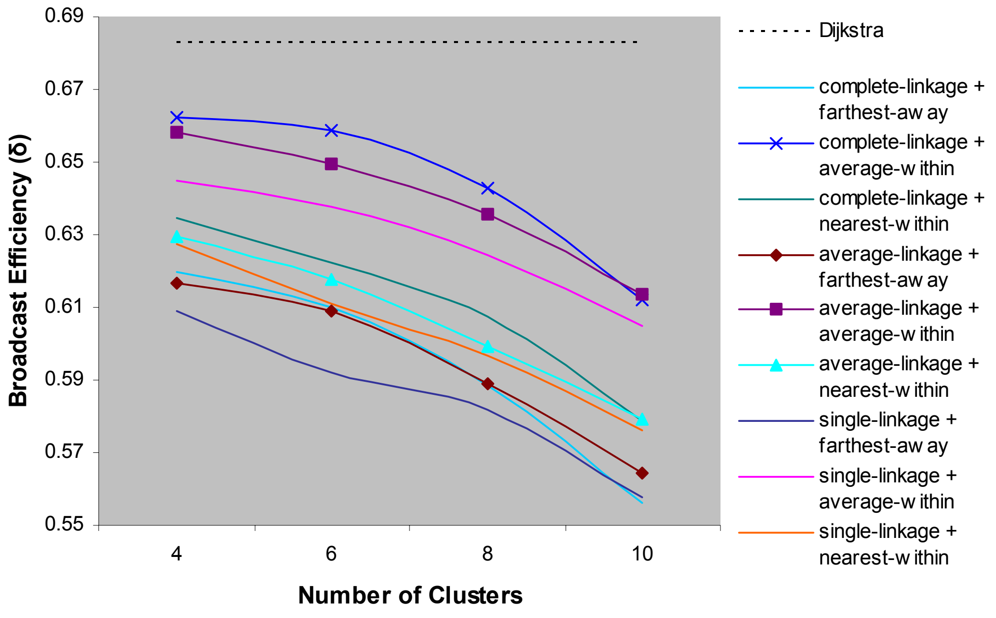

Fig. 17. Efficiency of broadcast trees against number of clusters per layer (all 9 cases)

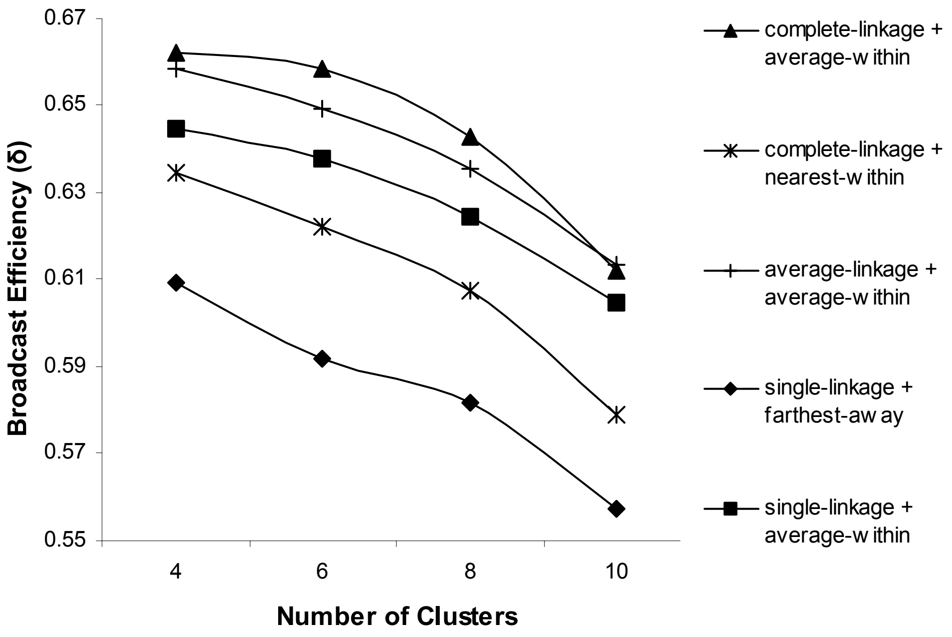

Fig. 18 Efficiency of broadcast trees against number of clusters per layer

Despite the efficiency gain with smaller k, we need to be aware that more servers are required. Typically, we find that the 6-cluster partitioning needs 9 more servers than the 10-cluster partitioning for an approximate average efficiency gain of 5%. We may wish to minimize the ratio of servers to total receivers, as servers must do extra work and are more costly. Figures 19 and 20 show the ratio of servers to the total number of receivers in a group (for all 9 cases and the selected 5 cases respectively). Here, the lower ratios imply that fewer servers need to be deployed. In this respect, four other techniques slightly out-perform our most efficient technique (KMC-CA). So, although

KMC-CA is the most efficient technique, it can be more expensive than some of the alternatives in reducing stress on the network.

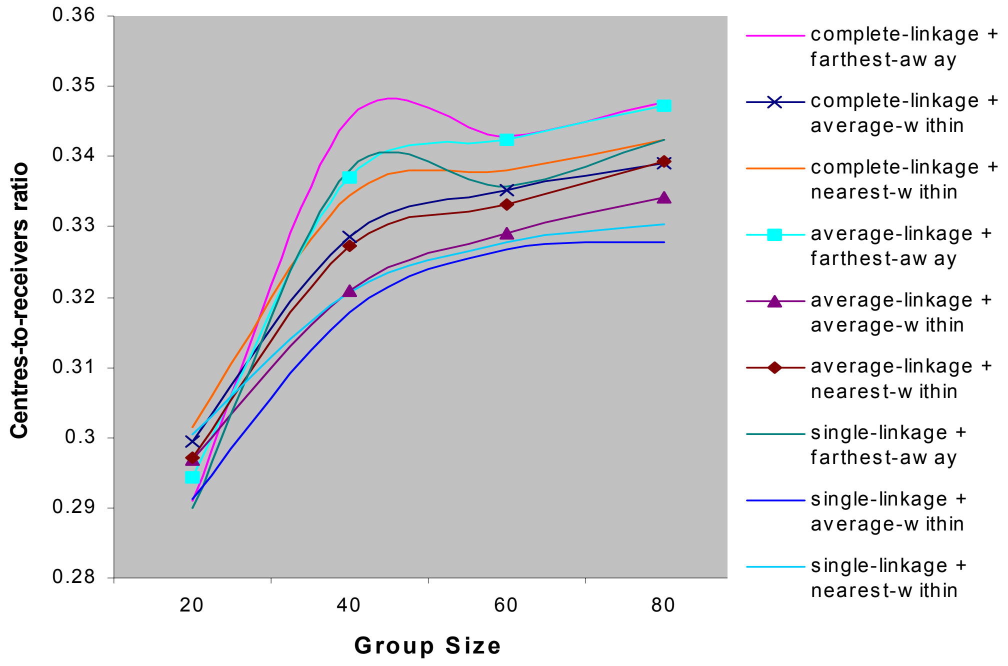

Fig. 19. Server to receiver ratio for multi-level clusters (all 9 cases)

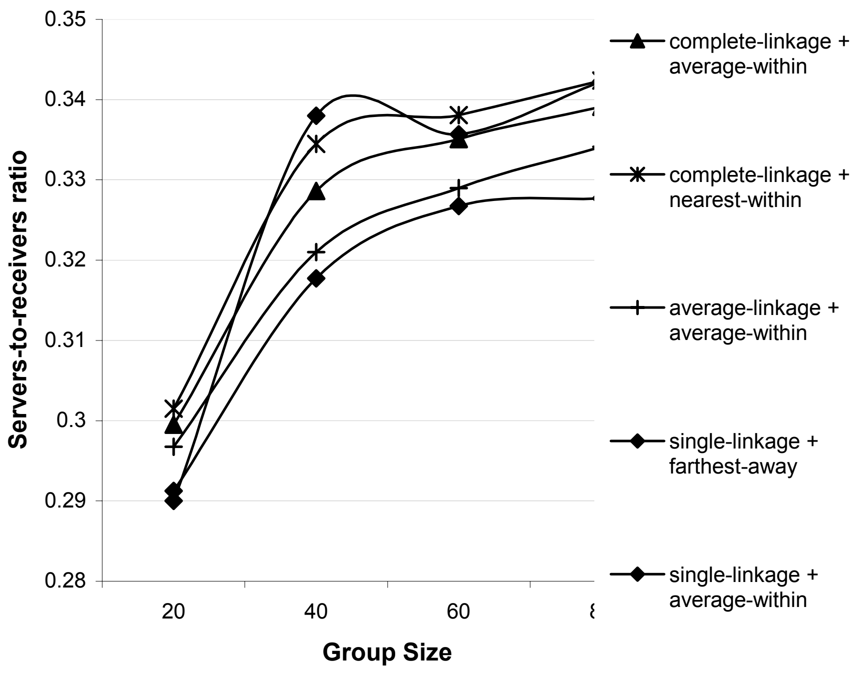

Fig. 20. Server to receiver ratio for multi-level clusters (5 selected cases)

General comments on multicast tree shape

The results of this section are based on all the experiments carried out on weighted-edge networks.

Our discussions on the shape of the trees constructed from the clustering techniques focus mainly on the depth of the trees. It is important to be aware that the constraints placed on the experimental network size and the maximum allowable number of child nodes limit the possible depth achievable by the clustering techniques.

Firstly, the maximum depth of the multicast trees formed by multi-level clustering is 7 with a group size of 80 (and very rarely, with 60 receivers too). We could consider an ideal balanced tree to be one in which all parents support k children and thus, for a given k gives us the shortest possible tree. With k set to 4:

A tree of depth 2 can accommodate up to 20 receivers.
A tree of depth 3 can accommodate up to 84 receivers.
A tree of depth 4 can accommodate up to 340 receivers.

Even the shortest of the trees achieved with group size 20 is still one level taller than an ideal balanced tree. Hence, we can conclude that the trees built from clustering are not ideal balanced trees. It should be noted that it is also difficult to achieve such a tree with other algorithms.

Secondly, comparing the maximum tree depth obtained by each variation of KMC, we found that, we found that farthest-away K-Means generally built shorter trees than other techniques and so formed trees nearer to ideal balanced trees. Considering the relationship between efficiency and tree depth, shorter trees achieve more branching near the top, whereas longer trees (such as those produced by MKC-CA) share the links near the top more effectively and thus give more efficient trees according to the normalized measure for stress that we used.

5.3.3 Unit-edged networks: efficiency

The results for unit-edged networks showed a similar pattern to the weighted edged ones but they achieved lower efficiencies. Initial clustering was often found to be the same for average linkage and single linkage. KMC-CA again emerged as the best technique over most evaluation criteria.

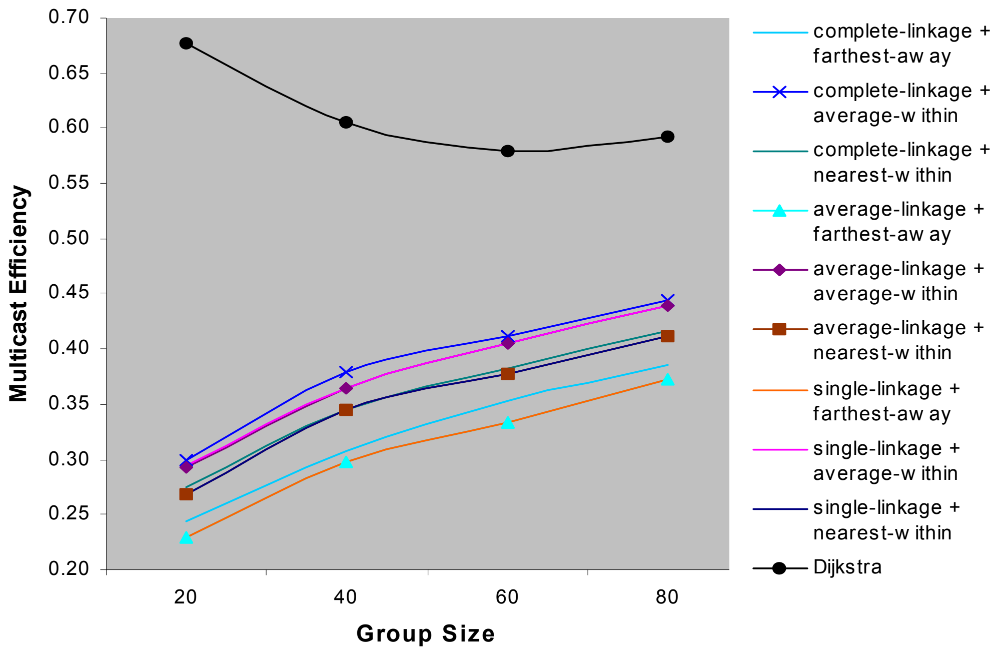

Fig. 21. Multicast efficiency of unit-edge trees constructed using multi-level clustering

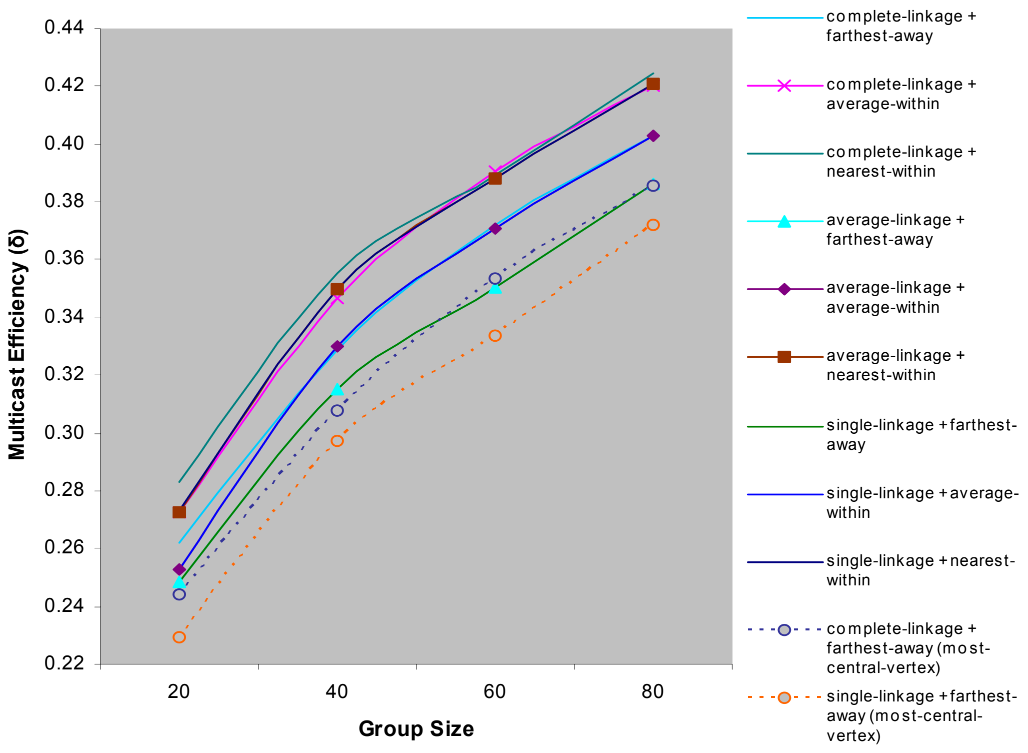

Fig. 22. Multicast solutions with closest-to-parent centres: Unit-edge networks

5.3.4 Unit-edged networks: number of servers

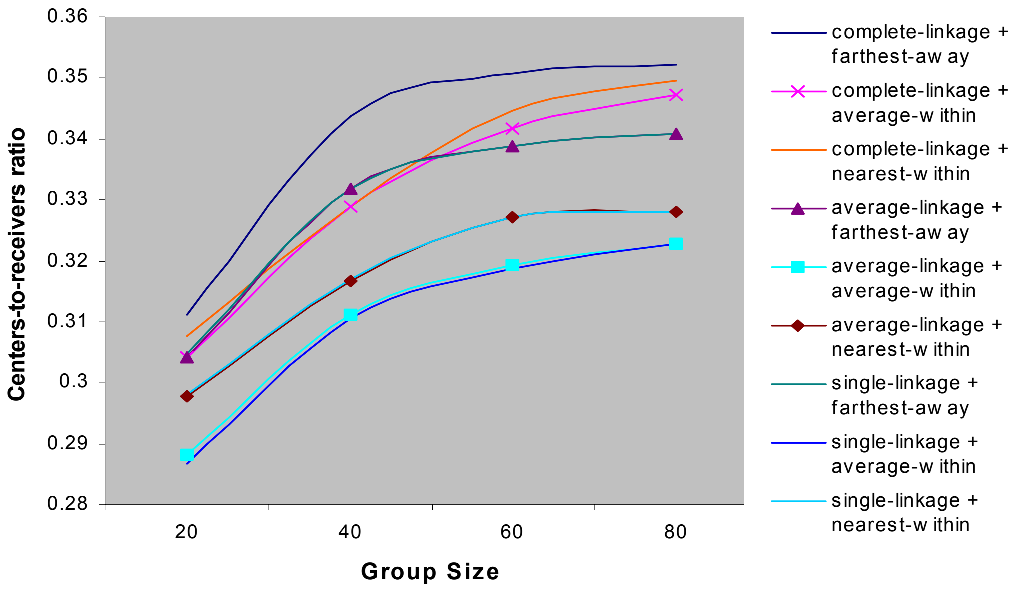

Fig. 23. Centres-to-receivers ratio obtained from multi-level clustering unit-edge networks for multicasting

5.3.5 k-means convergence

Figure 24 shows the maximum number of k-means iterations for the results presented in Fig. 14 for weighted edge networks; this was seven for groups of 20 and eleven for groups of 80. The lowest maxima were achieved by KMC options using the furthest away choice of centres. By choosing a centre that is furthest away from the nodes in all other clusters, it is less likely that nodes will move between clusters at each iteration.

Figure 25 is the result of averaging the number of iterations collected from ten simulated runs on each network. Averages over all samples ranged up from about 2.8 to 5.3 iterations. KMC-CA took on average between 2.8 to 3.7 iterations with an overall maximum of 8. Some of the other algorithms were quicker, but we believe that the computation time for our best algorithm is certainly acceptable for the networks that we considered.

For unit-edged networks, the average of the maximum over ten runs gave results broadly similar to weighted-edge networks with KMC-CA being quicker on average than all other techniques at from 3 to 3.6 iterations. There was a slightly smaller range of values compared with weighted edge networks; with averages from about 3 to 5.1. This may be because there is a smaller range of values in the all-pairs shortest path matrix for unit-edged networks than for weighted-edge networks.

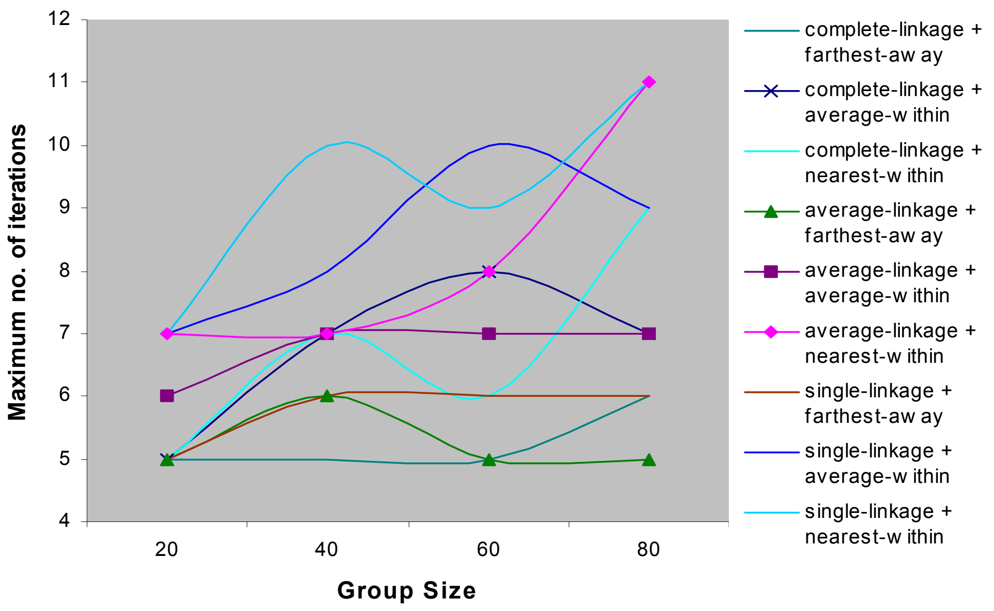

Fig. 24 Maximum no. of K-Means iteration: weighted-edge networks

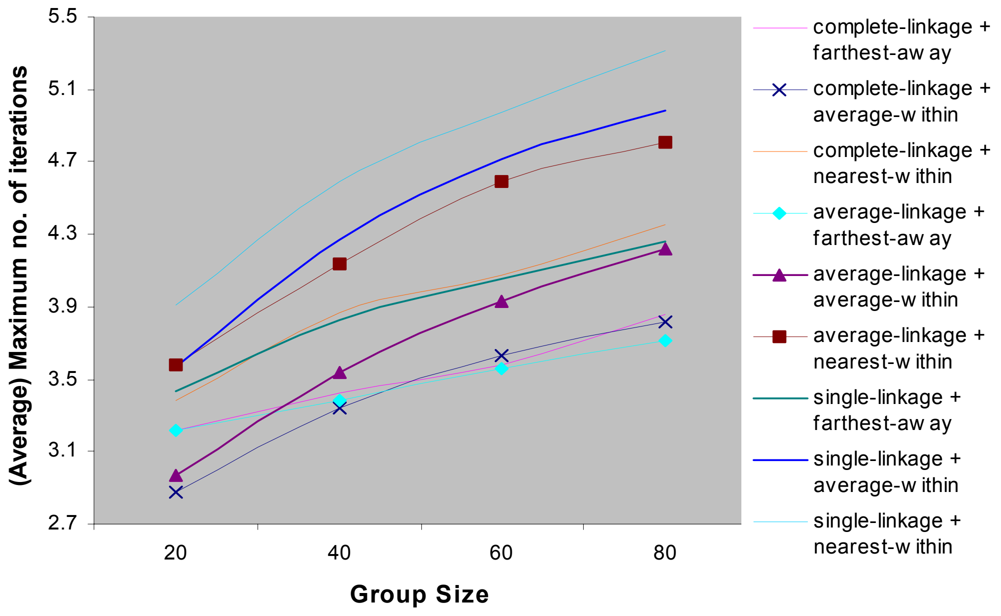

Fig. 25 Maximum no. of K-Means iterations: weighted-edge networks

5.3.6 Summary

We have shown that clustering can be applied to multicast hierarchies and that it can achieve significant gains over unicasting especially for weighted networks. From our results, we suggest for both weighted-edge and unit-edges networks:

- hierarchical clustering using complete linkage to determine the initial seeds;
- the average-within method for centroid calculation for k-means;
- finding servers that are the most-central nodes of a cluster when constructing the hierarchical trees.

The results for the unit-edged networks showed that it is possible to apply the KMC technique when using hop-counts rather than round-trip time. Unit-edged network solutions were less time-consuming, but also generally less efficient than weighted edge network solutions.

The techniques evaluated are centralised in that they assume complete knowledge of all-pairs shortest paths. We discuss the significance of this in the next section. The results discussed in this section are again promising and we are now comparing our techniques with other multicast overlay network proposals.

6. Conclusions and further work

Discussion

We have shown that KMC can be used to find multicast tree hierarchies both where geographical location is known and applied to network topologies modelled as graphs. Clustering is a heuristic technique that must be adapted to a specific problem, so we have also examined the best parameters for use in the k-means clustering heuristic. It is particularly interesting to see how well the population distribution of the participants can be matched. It is therefore likely to be applicable to other multicast group patterns e.g. those supporting distributed communities of users, as may be found for Reliable Multicast Transport. The results show the benefit of using hierarchical levels not only for reducing message and state information, but also in providing trees at reasonable cost. The evaluations also show that these trees are applicable throughout a range of group sizes.

The time-complexity of the technique is similar to Dijkstra's algorithm. This will not be a problem where it is applied as a top-level optimisation technique for which it may not be necessary to recalculate the tree every time a node joins the group. Indeed, by incorporating several nodes in a weight at a single geographical location, we are already reducing the time take to calculate the tree. For nodes situated very closely together, it is likely that other techniques could be used. e.g. Some nodes may belong to the same broadcast LAN and could be served in a single transmission.

To optimise hierarchies for specific classes of users, it is likely that, in addition to geographical location, other parameters may need to be taken into account, such as the services offered and whether subscription and charging are supported. The better the information available, the more optimal the solution found will be. Such information may be found within an Intranet, but is unlikely to be available at the network layer in the Internet. Working at the application layer can potentially ensure the privacy needed to provide such information.

The most appropriate applications of KMC are likely to be:

- Using geographical location where known, for high-level organisation of user communities. The technique may also be applicable for more fine grain clustering where global positioning information is known.
- For organising distributed communities, where it is helpful to take the user population into account
- For partitioning potential users into regions for a mesh-first approach to overlay networks.
- For organising support for large multicast groups in two stages; first using KMC for long-term configuration of the expected user populations into regions and then shorter timescale refinement of the local regions using KMC with topological information or using an alternative overlay technique.

Several of the above applications can be seen to be particularly useful for Grid Computing.

The advantages of considering KMC and its variations include:

- Control over the number of partitions at each layer; the value of k could be set to a different value at different layers and could be tuned to the size of the set to be partitioned.
- Control over the number of levels for which partitioning is carried out; this may help with bounding delay for delay-sensitive applications.
- Applicability to both weighted-edge (round-trip delay) and unit-edged networks (hop count based).

Further Work

We are currently comparing our clustering techniques with the NICE protocol, particularly in relation to stress and stretch. Early comparisons indicate that KMC performs well in this context.

We plan to examine the KMC method in more detail in the following ways.

- Further comparisons with the NICE protocol. This will include using models with the GT-ITM topology generator (Zegura, L. et al.) and using alternative values of k.
- Evaluating the performance of KMC for shared trees as well as for source-based trees;
- Adapting KMC to offer delay-bound solutions for real-time conferencing and other applications;
- Developing distributed versions of KMC.
- Identifying suitable control structures, for example using special servers that collect round trip delay information from users in a local region and carry out the tree calculation based on their overall knowledge of the region.
- Mappings (embeddings) from network topologies to Euclidean space to enable clustering to be used in a "virtual" geographical context with realistic measures between participants.

7. References

Aho, A. V., J. E. Hopcroft, et al. (1983). Data Structures and Algorithms, Addision-Wesley.
ATM-Forum "Private Network Node Interface."
Bacelli, F., D. Kofman, et al. (1999). Self Organizing Hierarchical Multicast Trees And Their Optimization. INFOCOM, IEEE.
Ballardie, A. (1997). Core Based Trees (CBT) Multicast Routing Architecture, IETF. Internet RFC 2201.
Banerjee, S., B. Bhattacharjee, et al. (2002). Scalable Application Layer Multicast. ACM Sigcomm.
Berman, F., G. Fox, et al. (2003). Grid Computing: Making the Global Infrastructure a Reality, Wiley.
Cahn, R. S. (1998). Wide Area Network Design: Concepts and tools for optimization, Morgan Kaufmann.
Chalmers, R. C. and K. C. Almeroth (2000). Modeling the Branching Characteristics and Efficiency Gains in Global Multicast Trees. IEEE GLOBECOM.
Chatterjee, S. and M. A. Bassiouni (1992). "Hierarchical message dissemination in very large WANs." IEEE Computer Soc. Press: 397-403.
Chiu, D.-M., S. Hurst, et al. (1998). TRAM: A tree-based reliable multicast routing protocol, Sun Microsystems Laboratories. SML TR-9896,.
Chu, Y.-H., S. G. Rao, et al. (2000). A Case for End System Multicast. ACM SIGMETRICS 2000, Santa Clara, CA.
Chuang, J. and M. Sirbu (1998). Pricing multicast communication: a cost-based approach. Proc. INET 98, Geneva, Switzerland.
Doar, M. and I. Leslie (1993). How Bad is Naïve Multicast Routing? IEEE INFOCOM.
El-Sayed, A., V. Roca, et al. (January/February 2003). "A survey of proposals for an alternative group communication service." IEEE Network: 2-7.
Estrin, D. and others (1998). Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification June 1998, IETF. RFC 2362.
Evans, J. R. and E. Minieka (1992). Optimization Algorithms for Networks and Graphs, Marcel Dekker Inc.
Floyd, R. W. (1962). "Algorithm 97: Shortest path." Communications of ACM **5**: 345.
Forman, G. and B. Zhang (2000). Distributed Data Clustering can be Efficient and Exact, HP Labs**:** 1-6. HPL-2000-158.
Hartigan, J. A. (1975). Clustering Algorithms, John Wiley & Sons.
IETF Working group on Reliable Multicast Transport. http://www.ietf.org/html.charters/rmt-charter.html.
Jennings, E., L. Motyckova, et al. (2001). Evaluating graph theoretic clustering algorithms for reliable multicasting. IEEE Globecom.
Lazarevic, A., Pokrajac, D and Z. Obradovic (2000). Distributed clustering and local regression for knowledge discovery in multiple spatial databases. 8th European Symposium on Artificial Neural Networks.
Liebeherr, J., M. Nahas, et al. (2001). "Application-Layer Multicasting with Delaunay Triangulation Overlays."
Lim, S. G. (2002). Constructing hierarchical multicast trees. thesis for MSc in Distributed Systems and Networks, University of Kent, UK.
Mathy, L., R. Canonico, et al. (2002). "Scalable adaptive hierarchical clustering." IEEE Communications Letters.
Mirkin, B. (1996). Mathematical Classification and Clustering, Kluwer Academic Publishers.
Ng, R. T. and J. Han (1994). Efficient and effective clustering methods for spatial data mining. 20th International Conference on Very Large Data Bases,, Santiago, Chile, Morgan Kaufmann Publishers.
Ramanaiah, O. and H. Mohantny (2001). Managing disconnection of mobile hosts in distributed clustering. 9th international conference on advanced computing and communications, Bhubanshwar, India.
Sharma, S. (1996). Chapter 7 Clustering Algorithms. Applied Multivariate Techniques, Wiley.
SQUID The Squid Web Proxy Cache. http://squid.nlanr.net/.
Touch, J. and A. S. Hughes (1998). "LSAM proxy cache: a multicast distributed virtual cache." Computer Networks and ISDN Systems **30**: 2245-2252.
Tran, D. A., K. A. Hua, et al. (2002). ZIGZAG: An efficient peer-to-peer scheme for media streaming. Orlando, University of Central Flofida. CS-UCF 2002.
Waters, G. (2000). Hierarchies for network evolution. 16th UK Teletraffic Symposium on Management of Quality of Service - the New Challenge, Harlow, UK.
Waxman, B. M. (1988). "Routing of Multipoint Connections." IEEE Journal on Selected Areas in Communications **6**(9): 1617-1622.
Zegura, E. W., C. K. L., et al. (1996). How to Model an Internetwork. IEEE Infocom.
Zhuang, S. and others (2001). Bayeux: an Architecture for Scalable Fault-tolerant Wide Area Dissemination. 11th International Workshop on Network and Operating Suystem Support for Digital Audio and Video.

Appendix A
Data sets of North American Cities

51 cities,
Latitude and longitude to nearest degree.

-118	34
-117	33
-122	38
-122	37
-122	39
-112	34
-107	35
-123	46
-122	47
-111	32
-96	33
-97	33
-98	29
-95	30
-97	35
-76	36
-105	40
-105	39
-95	39
-96	41
-93	45
-88	42
-88	43
-90	39
-97	50
-71	42
-73	46
-79	44
-83	43
-83	40
-79	43
-76	45
-114	51
-113	54
-74	41
-75	40
-76	40
-77	39
-76	37
-77	37
-81	35
-84	34
-87	36
-87	34
-90	32
-86	38
-84	39
-90	30
-82	30
-80	26
-85	28

46 cities:
Longitude, latitude and population weight (1 per 500k inhabitants)

35	-107	1
34	-84	1
30	-98	1
40	-76	1
42	-71	1
51	-114	2
42	-88	6
41	-82	1
40	-83	1
33	-96	2
40	-105	1
43	-83	2
54	-113	2
33	-97	1
43	-80	1
30	-95	4
40	-86	1
30	-82	1
39	-95	1
36	-115	1
33	-118	1
34	-118	7
35	-90	1
26	-80	1
43	-88	1
46	-73	6
36	-87	1
30	-90	1
41	-74	15
35	-97	1
45	-76	2
40	-75	3
34	-112	2
46	-123	1
47	-71	1
39	-122	1
33	-117	2
38	-122	2
37	-122	2
47	-122	1
44	-79	8
32	-111	1
49	-123	4
37	-76	1
39	-77	1
50	-97	1