Spanners and Message Distribution in Networks*

Arthur M. Farley[1], Andrzej Proskurowski[1], Daniel Zappala[1] and Kurt Windisch[2]

[1] Computer and Information Science Department,
University of Oregon, Eugene, OR, 97403
[2] Advanced Network Technology Center, Computing Center,
University of Oregon, Eugene, OR, 97403

Abstract. We investigate the applicability of spanners as shared structures that offer both low cost and low delay for broadcast and multicast. A k-spanner has the potential to offer lower delay than shared trees because it limits the distance between any two nodes in the network to a multiplicative factor k of the shortest-path distance. Using simulation over random topologies, we compare k-spanners to single-source minimum-distance spanning trees and show that a k-spanner can have similar cost and lower delay. We illustrate that by varying the value of k a spanner can be made to gradually favor cost over delay. These results indicate that it is feasible to build a multicast routing protocol using spanners as a shared communication structure.

1 Introduction

Current multicast routing protocols can be classified as constructing either a separate shortest path tree for each sender to a group or a single shared tree for all senders. The tradeoffs between these two approaches have been well-studied [15]; generally speaking, shared trees have lower cost than shortest-path trees, but incur higher delay. Ideally, a multicast routing protocol could utilize a single shared substructure for each group that has the advantages of both approaches, namely both low cost and low delay.

We are investigating the use of graph spanners as a shared substructure that can obtain a cost and delay similar to building a separate shortest path tree for each sender. In particular, we are studying k-spanners, a structure that guarantees that the distance between any two nodes does not exceed the shortest path distance by a multiplicative factor k. Unlike other commonly-used shared structures, such as Steiner Minimal Tree approximations [16, 13, 9] or core-based trees [1, 7], the k-spanner can bound the path length between any two vertices in the graph. This bound can take into account hop count, Euclidean distance, or link weights.

In this paper we are concerned primarily with broadcast, rather than multicast, because algorithms for spanners (where all nodes are group members) are

* Research of all authors supported in part by National Science Foundation grant NCR-97-14680.

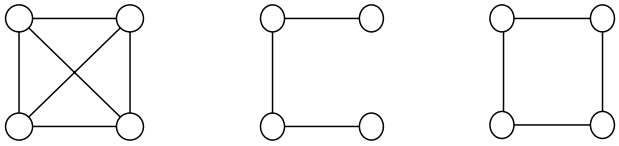

Fig. 1. Examples of a spanning tree and a 2-spanner of a 4-vertex graph

more tractable than algorithms for stanners (where only a subset of nodes are members). Broadcast is also the basis for bootstrapping multicast communication in several multicast protocols, namely DVMRP [12] and PIM [7]. Both of these protocols build a separate shortest-path tree for each sender; eventually we hope to use spanners to bootstrap multicast over a shared structure. More details on our planned future work with multicast are given in Section 6.

Our primary focus in this paper is a performance evaluation of spanners relative to shortest-path spanning trees. While the theoretical bounds of k-spanners are well-known, their average behavior is not. Our goal is to characterize the average behavior of k-spanners as a function of k to assess their applicability for group communication. We demonstrate through simulation that a k-spanner may gradually favor cost over delay by adjusting k.

The rest of the paper is organized as follows. Section 2 defines graph spanners and reviews relevant theoretical results. We then discuss in Section 3 how graph spanners can be used for group communication. In Section 4 we present our experiment design for studying spanner characteristics and in Section 5 we present our results. Finally, in Section 6 we discuss our conclusions and several areas for future research, particularly in applying this work to multicast.

2 Graph Spanners

The notion of graph spanners was introduced about a decade ago in [11]. Each spanner is parameterized by a constant k, as follows: A k-spanner of a graph $G = (V, E)$ is a graph $S = (V, E')$, where E' is a subset of E, such that the distance in S between any pair of vertices x and y of V is not more than k times the distance between x and y in G. The *distance* between two vertices of a graph is defined as the minimum, over all paths connecting the two vertices in the graph, of the sum of edge lengths on the path.

To illustrate the differences between spanning trees and spanners, we show an example of a minimum distance spanning tree and a 2-spanner of a 4-vertex graph in Figure 1.

A k-spanner is formed by removing certain edges from G while guaranteeing that the distance between any pair of vertices is not stretched by more than the multiplicative factor k. Unfortunately, the problem of deciding whether a k-

spanner exists that has total weight (*i.e.*, sum of weights associated with edges) less than W in an arbitrary graph G has been shown to be $\mathcal{NP}$-complete for most values of k [3]. The problem remains $\mathcal{NP}$-complete for any planar, biconnected graph G, with lower limits for k depending on whether edges of G are weighted or unweighted (*i.e.*, weights all equal to one). Finally, determining whether an arbitrary graph has a k-spanner that is a tree or that is planar is also $\mathcal{NP}$-complete [4, 2].

Given the apparent intractability of finding minimum-weight spanners, a greedy algorithm that finds relatively light-weight spanners can be used. The algorithm is defined as follows:

Algorithm LightWeightSpanner(G, k)
// Input: an arbitrary connected graph $G = (V, E)$ and an integer $k > 1$
// Output: a k-spanner $S = (V, E')$ of G

Step 0. Let E' be empty.
Step 1. Sort the edges E in increasing order of weight.
Step 2. For each edge $e = (x, y)$ in E (considered in sorted order)
    if { the distance between x and y in $S = (V, E')$
        is greater than k times the distance in G }
    then { add edge e to E' }
Step 3. Report $S = (V, E')$ as result.

The algorithm considers light edges first and only adds an edge to the spanner when it is necessary to connect neighbors so as to maintain the given stretch factor. The resultant graph is a k-spanner because, if each edge in G has not been stretched by more than k, then any path between non-neighbors in G has not been stretched by more than k. The algorithm is clearly implementable in polynomial time. Mansour and Peleg have determined some properties of the spanners generated by this algorithm; they are guaranteed to be sparse (with $\mathcal{O}(n)$ edges) and light-weight (within $\mathcal{O}(\log n)$ of the minimum weight of a spanning tree) for a $(\log n)$-spanner. [10].

3 Broadcast and Multicast over Spanners

There are several alternatives for broadcasting in a network. DVMRP bootstraps multicast communication by first flooding over a shortest-path spanning tree rooted at the source of the data. Leaf nodes that are not members of the group or that do not want data from the source may prune themselves off, resulting in a shortest-path multicast tree. DVMRP builds the shortest-path spanning tree by incorporating its own distance-vector unicast routing protocol, a variant of RIP [8]. To reduce this dependence on a particular unicast routing protocol, dense-mode PIM [6] floods the entire network, before pruning from leaf nodes as in DVMRP.

Clearly, flooding the network as dense-mode PIM does is expensive, as every edge in the graph is visited at least once by each message. Broadcasting over a shortest-path spanning tree as in DVMRP is much less expensive, but a separate tree must be used for each sender to the group. Once leaf nodes begin to prune themselves from the tree, source-specific state is stored in routers.

Broadcasting over a k-spanner of a network may potentially be as inexpensive as a spanning tree in that not every edge is touched. Unlike a spanning tree, however, a k-spanner has the advantage of being a shared structure, usable by any sender to the group.

A spanner can also be used to bootstrap multicast communication. To do this, leaf nodes that are not members of the group may prune themselves from the k-spanner, storing only group-specific state in routers. This can be done trivially by pruning non-member leaf nodes that have degree 1; further work is needed to develop an algorithm for pruning non-member nodes with greater degrees.

Using a shared structure usually has a disadvantage compared to sender-rooted trees, namely traffic concentration. However, the results from [15] illustrating this problem affect only core-based trees, which have single router at their center. Because traffic from all senders to the group must pass through the core, it has the potential to overload nearby links. A k-spanner, on the other hand, does not have a single core and thus should not suffer from this problem to the same extent.

Another type of traffic concentration problem can occur when the same core, or a small set of cores, is used for all core-based trees built by a multicast routing protocol. An analogous problem can plague the k-spanner if the same spanner is used for all multicast groups. This problem can be mitigated by creating a different spanner for each multicast group. This can be done by randomly permuting the uniformly-weighted edges considered by the spanner algorithm presented earlier. To ensure that each node computes the same spanner for a group, the group address can be used as the seed for the random permutation.

4 Experiment Design

In order to be useful for bootstrapping multicast, a k-spanner must first have desirable broadcast properties. To assess the utility of broadcasting over spanners, we have designed a set of experiments to measure average cost and delay over a set of random topologies. We compare spanners to flooding the entire graph, which gives maximum cost and minimum delay, and to transmitting on a single, shortest-path spanning tree rooted at a random node (which reduces cost but increases the delay). We have chosen this latter style of communication because it has been suggested as the basis for forming a low-cost core-based tree [7]. Details on how other types of shared trees perform based on these two metrics may be found in [15].

We want the randomized topologies to be representative of topologies for administrative domains of the Internet. As such, we generate graphs having either

64 or 128 vertices and average vertex degrees of approximately 4 or 8. In addition, we use two vertex connection schemes: one representing a strictly random adjacency pattern and a second preferring "nearby" vertices over "distant" connections. This results in eight different classes of random graphs, one for each combination of the above three conditions. Our test sets consist of 50 graphs for each type. We generated 3 sets of graphs for each type to get some indication as to the stability of our results relative to the random number generator used to create the network graphs.

We use the suite of random graph generators available at Georgia Tech [5]. These algorithms all place a set of n vertices on a square in the plane, and then consider each pair of vertices in turn, deciding whether an edge is to be added between them. The purely random scheme uses an equal probability between all pairs of vertices; the locality preference scheme has the probability decreasing exponentially with the distance between the two vertices on the plane ([17]). The locality method we used is the basic Waxman model, [14] in which the probability of an edge being added between two vertices is $\alpha e^{-d/(\beta L)}$ where d is the distance between the vertices on the plane and α and β are parameters of the method. An increase in α increases the number of edges, and an increase in β increases the ratio of "long" to "short" edges. We use the following parameter settings: for degree 4, 64 vertex graphs, $\alpha = .42$, $\beta = .14$; for degree 4, 128 vertex graphs, $\alpha = .21$, $\beta = .14$; for degree 8, 64 vertex graphs, $\alpha = .85$, $\beta = .15$; and for degree 8, 128 vertex graphs, $\alpha = .42$, $\beta = .14$.

The primary metrics we are interested in are number of edges, average distance, and maximum distance in a given graph, its spanners, and spanning tree. We consider three measures for the distance along an edge in a given graph. The first is uniform over all edges; we assume its value is one. Given this measure, the distance between two vertices equals the number of edges ("hop count") in a shortest path between the pair. The second measure is the (Euclidean) distance between the end-vertices of the edge wrt. their location on the plane (determined when the graph is generated). The third measure is a random weight, assigned from the set $\{1, 2, 4, 8, 16\}$, reflecting the fact that for traffic management purposes in networks like the Internet, the cost of an edge may be set to an arbitrary value irrespective of its physical length. We refer to these three distance measures as the hop, length, and weight measure, respectively.

For each given graph G, we determine the following subgraphs: k-spanners $S_k(G)$, $2 \leq k \leq 7$, and a minimum-distance spanning tree $T(G)$ from vertex labeled 0, for all three edge-distance measures (a total of 21 subgraphs). Note that each subgraph has the same vertex set as the original graph, but has only a subset of the edges. We then compare each subgraph to the original graph in terms of three metrics: number of edges E, diameter D, and average distance A between vertices. We make these comparisons in terms of the edge-distance measure (either hop, length, or weight) used in constructing the spanner.

We compare subgraphs by computing the ratios of parameter values for a subgraph to the values for the given graph; for example, $E(S_2(G))/E(G)$ would be the ratio of number of edges in a 2-spanner of G to the number of edges in

a given graph G. We use the notation $G_{v,d}$ to represent the set of graphs with v vertices and average degree d; for example $G_{64,4}$ represents the set of graphs with 64 vertices and average degree 4.

We generated three test sets of 50 graphs for each of the above profiles. Our results indicate little or no difference between the three test sets for the same parameter settings. Thus, we report the results from the first set of graphs for each parameter combination.

5 Experiment Results

The experiment as designed above, generates a complex array of results. There are 8 types of random graphs, based on number of vertices, average vertex degree, and whether there was a locality preference in generating edges. Then there are 3 edge-distance measures applied to edges: hop, length, and weight. For each graph and distance measure, we compute k-spanners for k from 2 to 7 and minimum-distance spanning trees rooted at an arbitrary vertex. The metrics are computed for all graphs are: number of edges, average distance between vertices, and greatest distance between vertices (*i.e.*, graph diameter).

The results of our experiment are expressed in terms of the ratios that compare values of the metrics for a spanner or a spanning tree to the corresponding values for the original graph. This use of ratios to compare graph metrics is similar to that used in [15] to evaluate options for multicast shared trees. The ratios of the numbers of edges are less than 1.0, while distance-related ratios are greater than 1.0 (reflecting the increased distance due to excluding some edges). We plot the values of these metric ratios for k-spanners as functions of k ranging from 2 to 7. Each plot presents the average ratio, with error bars indicating the twenty-fifth and seventy-fifty percentile value, over the 50 graph sample. The ratio of the corresponding parameter value for spanning trees is shown on each plot for comparison purposes.

5.1 Number of Edges

Given a graph with n vertices, every spanning tree has $n-1$ edges. As such, the edge ratio for this parameter is a simple function of the average degree of the original graph (*i.e.*, about twice its inverse). For a random graph with average degree 4, we expect the edge ratio to be $(n-1)/2n$, or a little less than .50; for degree 8 graphs, we expect $(n-1)/4n$, or a little less than .25. There will be some variability due to the randomness of the graphs generated, more so for the locality preference graphs as the parameter settings that influence average degree were determined by trial and error. The spanning trees set the lower bound for the number of edges in the spanners. Edge ratios for the spanners will range from a possible high of 1.0 (all edges included) down to the ratios associated with the spanning trees.

Let us start with results for $G_{64,4}$, the 64 vertex graphs having vertices with average degree 4. Figure 2 shows two graphs depicting edge ratios for k-spanners

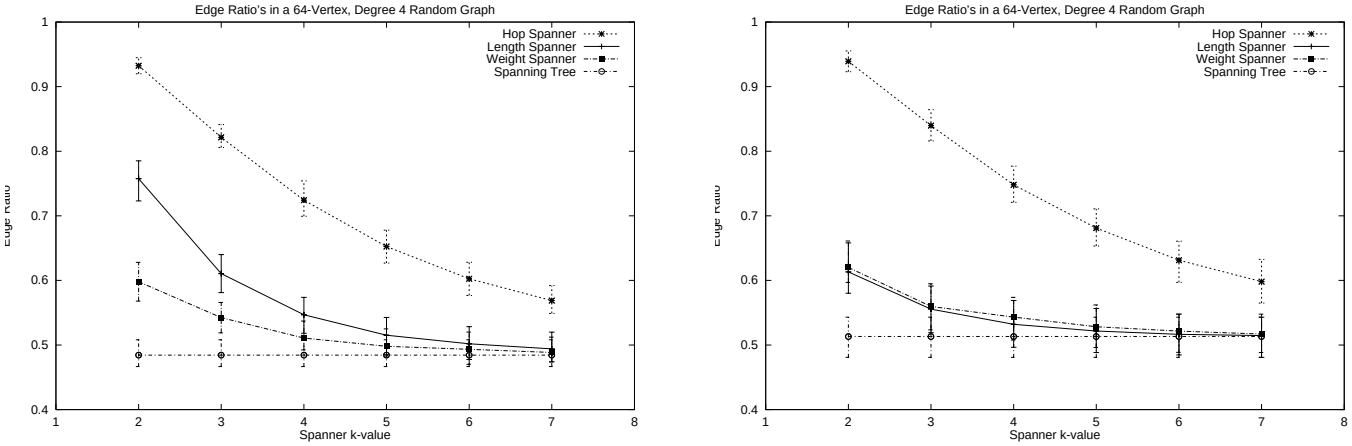

Fig. 2. Edge ratios for the two random graph models (locality preference on the right).

with k from 2 to 7; one graph showing the values of metrics for purely random graphs and the other showing the same values for graphs with a locality preference. We first note the general shape of the curves. The number of edges in k-spanners approach the number of edges in the spanning tree as k increases. This is consistent with the theoretical results mentioned earlier regarding properties of the light-weight spanners generated by the spanner algorithm we used. The decrease in number of edges is especially pronounced for the length and weight edge-distance measures. In addition, the numbers of edges in k-spanners of locality preference graphs for these two distance measures have nearly identical behavior as k varies.

The k-spanners constructed for the hop edge-distance measure (*hop spanners*) consistently have more edges than those constructed for length and weight edge-distance measures (*length* and *weight spanners*). This phenomenon has a straightforward explanation. Recall the algorithm only adds edges (considered in increasing order of their length or weight) to the spanner as needed to maintain an increase in distance by a factor of less than k between neighbors in the original graph. A "long" edge in the length or weight spanners may not be needed, as it can be effectively "replaced" by a path of possibly more than k "short" edges. In a hop spanner, all edges have equal distance and are considered in a random order. Thus, an edge can only be omitted from the hop spanner when a path of k or fewer edges between its end-vertices has already been considered, which can be expected to happen less often.

As far as any differences between purely random and locality preference graphs, edge ratios for the hop and weight spanners are affected little, if at all. For the length edge-distance measure, some differences do appear. The edge ratio for length spanners is consistently lower in the locality preference graphs. This indicates that the locality preference for connecting neighbors in the given,

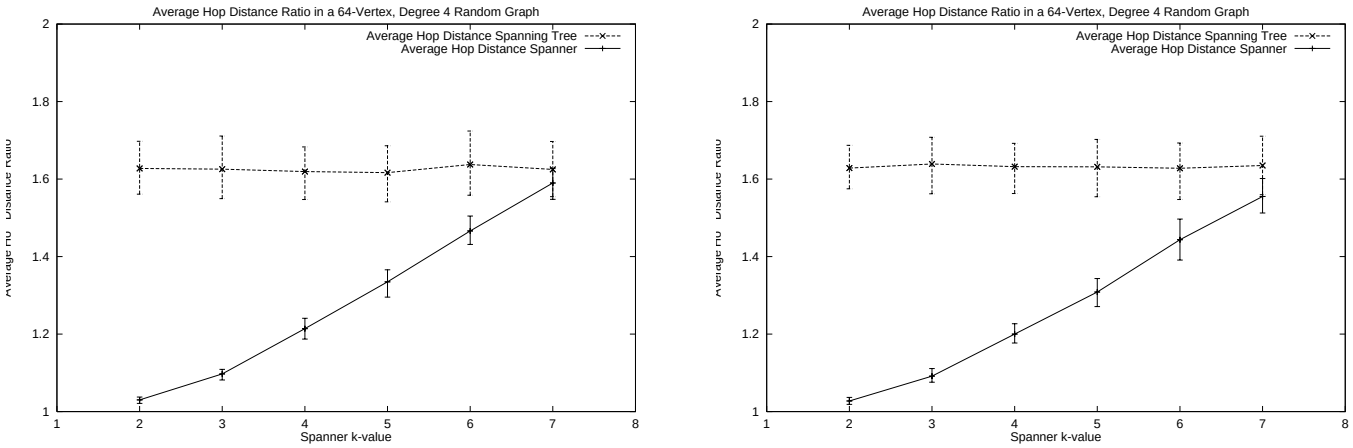

Fig. 3. Average hop distance for the two random graph models (locality preference on the right).

randomly generated graph does provide a small advantage when forming spanners based on the length distance measure in terms of number of edges required. As there are more "short" edges in such graphs, a higher percentage of what "long" edges there are can be eliminated by the spanner algorithm. Note that when all edges are considered to have the same edge-distance, as with the hop measure, or when edge-distances do not correspond to the basis for the locality preference, as with the weight measure, there is essentially no effect on number of edges.

5.2 Distance Measures

We next turn to distance-related results for the graphs with 64 vertices and average degree 4. As spanners have fewer edges than the original graphs, distances between vertices will increase. By how much they increase and how they compare to increases for minimum-distance spanning trees are the key questions. While the number of edges is related to traffic generated during message distribution, distance corresponds roughly to communication delay in the network. Let us first consider the hop distance measure, as this is usually considered to be most relevant to distance and delay in the Internet.

First, we consider average hop distance between vertices. Figure 3 presents results for the hop spanners, the first plot for purely random graphs and the second one for locality preference graphs. In a minimum-distance spanning tree the average distance ratio between the tree and the original graph, $A(T(G_{64,4})))/A(G_{64,4}))$, is slightly greater than 1.6. We see that average hop distance increases by over 60%, regardless of locality preference in neighbor selection. Recall that this is for group communication using the spanning tree as a shared structure, so we are considering all nodes, not just the root of the tree, as potential sources.

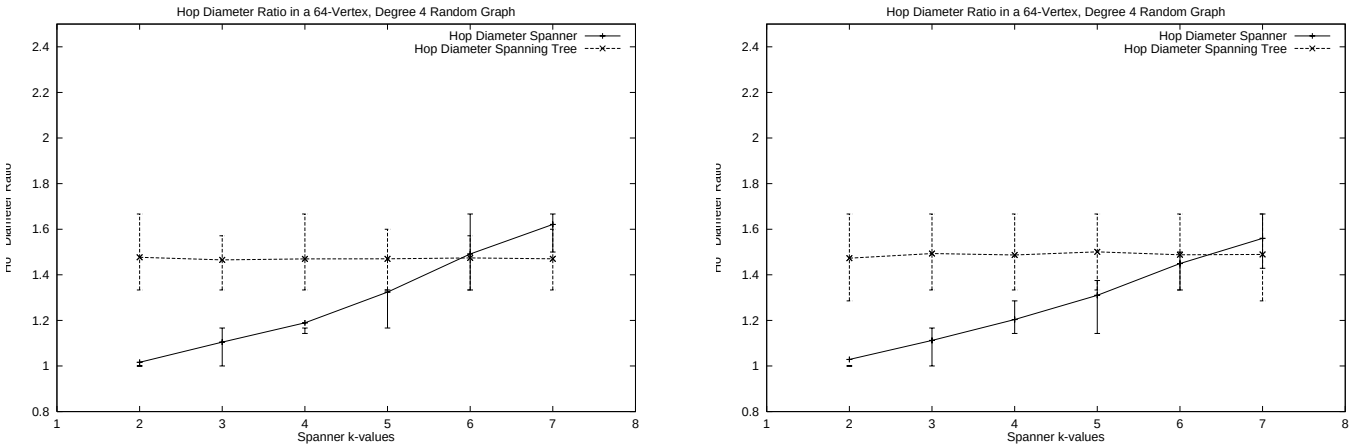

Fig. 4. Average hop diameter for the two random graph models (locality preference on the right).

For a 4-spanner of such graphs, the ratio $A(S_4(G_{64,4})/A(G_{64,4}))$ is approximately 1.2, representing only a 20% increase in average hop distance. As spanners do not approximate trees for small k, there are alternative paths that provide shortcuts between vertices, resulting in average distances that are significantly lower in such spanners. We see average hop distances are not impacted by locality preference in the network topology.

In Figure 4, we consider hop diameters in degree 4 graphs, again by two plots, one for purely random graphs and one for locality preference graphs. In a minimum-distance spanning tree, the ratio $D(T(G_{64,4}))/D(G_{64,4})$ is about 1.5, representing a 50% increase over hop diameters of the original graph. For a 4-spanner of such graphs, the ratio $D(S_4(G_{64,4})/D(G_{64,4}))$ is approximately 1.2, for only a 20% increase in hop diameter. Again, we see diameters associated with hop distances are not changed by locality preference in modeling the network topology. Overall, we see 4-spanners perform very well in comparison to minimum-distance spanning trees when considering average hop distance; the difference is less for the hop diameter measure, but still significant.

Our results show that, as was true for number of edges, only results associated with the length measure for edge distance are impacted by locality preference in forming edge connections. Figure 5 shows how average distance ratios in spanners and spanning trees for the length edge-distance measure is indeed improved in the locality preference networks. Even though the number of edges in spanners of the locality preference graphs is less than in spanners for the pure random model, their diameters and average distances are also smaller. Thus, if locality preference based on the edge-distance measure plays a role in creating a network graph, then spanners of that graph perform even better than indicated by the results for hop spanners.

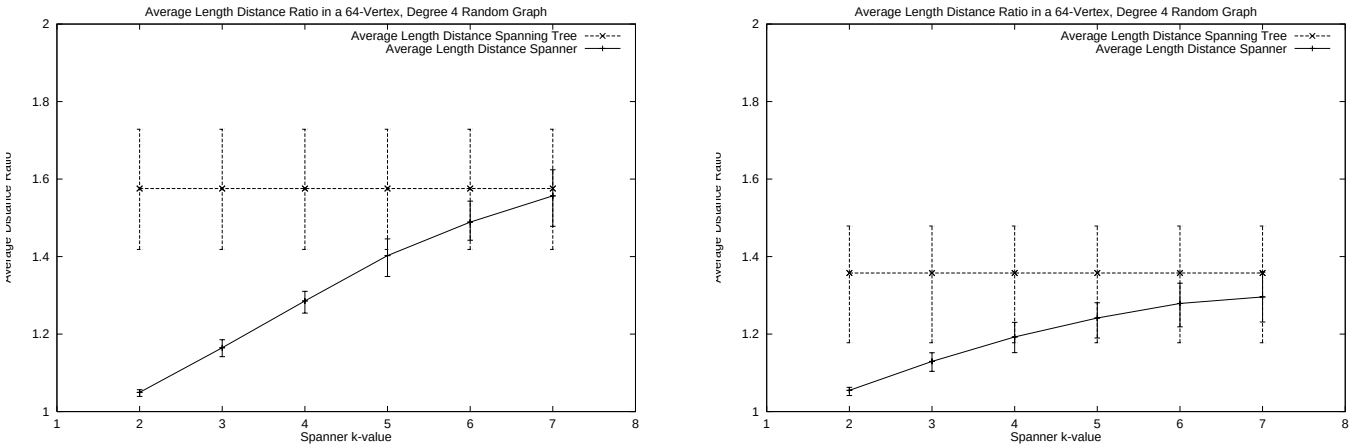

Fig. 5. Average length distance ratios (locality preference on the right).

Note that the distance metrics for spanning trees do not represent an upper limit on those metrics for k-spanners. Spanners do not focus on minimizing any distances, only limiting maximum distance growth. Spanners for higher values of k tend to permit greater distances between vertices than do the minimum-distance spanning trees. This is especially noticeable when considering the diameter measure. The k-spanners we generated have higher hop diameter than the corresponding spanning trees when k was greater or equal to 5.

5.3 Vertices and Degrees

As average vertex degree increases, both spanning trees and spanners have lower edge ratios. In Figure 6, we turn our attention to $G_{64,8}$, graphs having 64 vertices with average degree of 8. In Figure 2 we saw that for a degree 4 graph, a 4-spanner has less than 55% of the edges of the original graph for the length and weight edge-distance measures; a 4-spanner for the hop edge-distance measure has about 75% of the edges. Now, in a degree 8 graph, a 4-spanner has only approximately 30% of the edges of the original graph for the length and weight edge-distance measures; for the hop edge-distance measure, it has less than 50%. Recall that if we use a spanner to broadcast a message by flooding within a domain, the number of edges corresponds roughly to the message traffic generated. We can see that using a 4-spanner to flood the network results in significant reduction in traffic, and that this benefit increases as average degree of vertices in the network graph increases.

When considering the effects of average vertex degree on distance measures, we find that the average hop distance ratio for the spanning tree is approximately 1.75 in graphs with average degree 8; for 3-spanners, it is less than 1.25, and for 4-spanners it is less than 1.45, regardless of locality preference. In these same graphs, the hop diameter ratio for the spanning tree is approximately

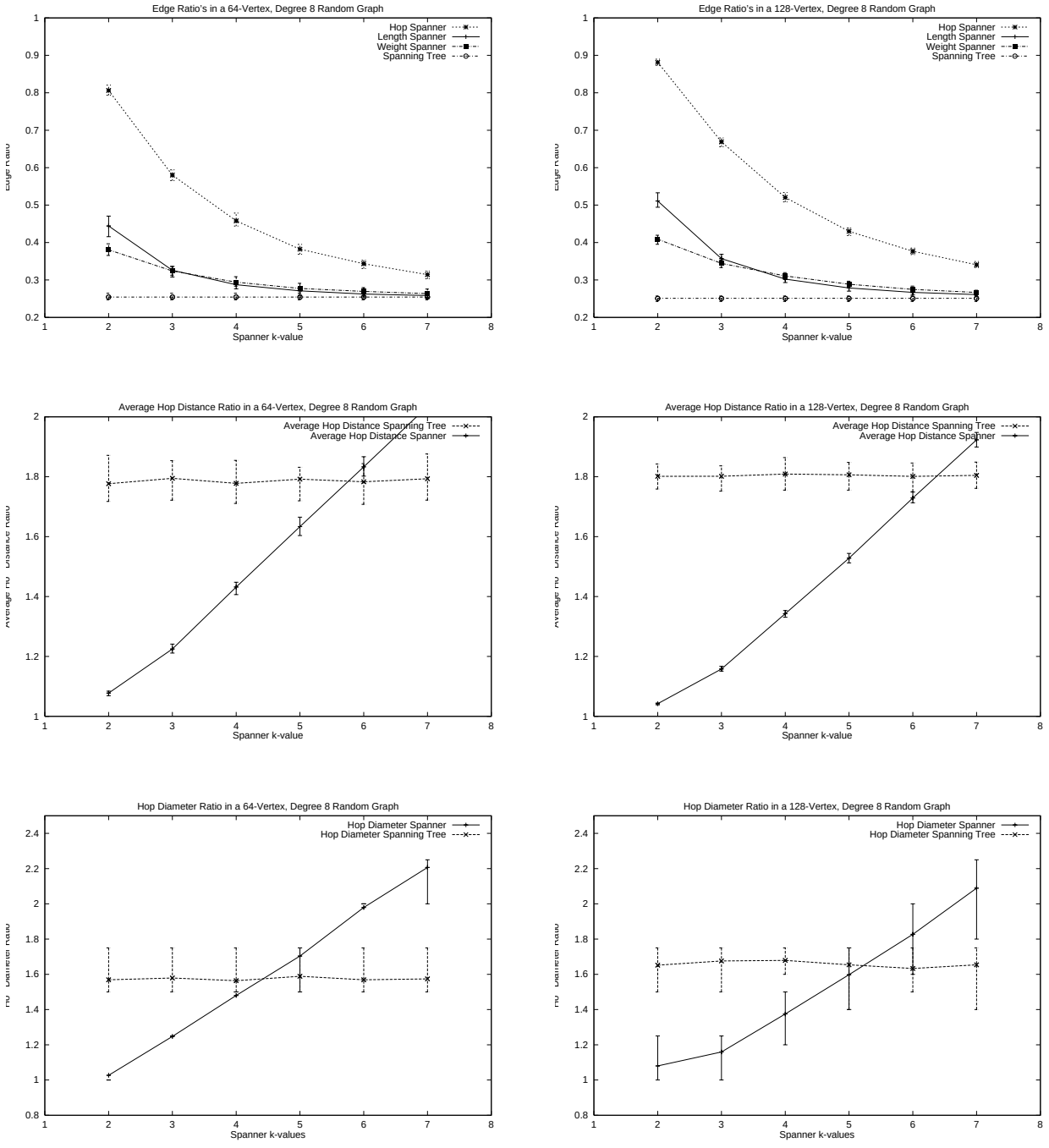

Fig. 6. Representative results for degree-8 graphs.

1.55; for 3-spanners it is less than 1.25, and for 4-spanners it is less than 1.50, regardless of locality preference. We see that distance ratios tend to increase overall, reflecting a penalty incurred by being able to discard more edges when forming the spanners in higher degree graphs. The pattern of results we find when comparing spanners to the minimum-distance spanning trees are essentially unchanged, however. The 3- or 4-spanners significantly reduce distances over the spanning tree. As in degree-4 graphs, spanners show better relative performance on the average distance metric than on the diameter metric. Figure 6 shows some representative results for spanners of degree-8 graphs.

Finally, we consider network graphs having 128 vertices. The first thing we note is that the general pattern of results remains unchanged. Overall, the edge ratios tend to be slightly higher for the spanners. This increase in edge ratios produces slightly better distance-related ratios in the spanners. The results for graphs with 128 vertices are somewhat more supportive of the usefulness of 3-, 4-, or 5-spanners as distribution topologies; spanner distance-related ratios are slightly better when compared to minimum-distance spanning tree results. However, we feel it is best to say that our results indicate that spanner properties are, for the most part, independent of the number of vertices. This is in contrast to average degree and, in the case of the length edge-distance measure only, locality preference, which have clear, more significant impacts on spanner performance.

6 Conclusions and Future Research

In this paper, we conduct what we believe to be the first experimental analysis of k-spanners with respect to cost and delay. We have shown the k-spanner to be an intermediate between flooding over the entire network and using a single-source, minimum-distance spanning tree. Using a value of 4 or 5 for k produces a k-spanner with cost approaching that of the spanning tree, yet with significantly lower delay. These characteristics make the spanner a good candidate for a shared structure in a multicast routing protocol.

As part of our ongoing research, we are attempting to create a multicast routing protocol that bootstraps multicast using spanner broadcast, essentially a flood-and-prune protocol over a shared structure rather than a shortest-path tree. There are several challenges in this area. First, we are developing a protocol that allows non-members to prune themselves off the spanner. This is a difficult problem because removing a node may disconnect the spanner or at the very least increase the delay for some nodes beyond the multiplicative k factor. Second, we must develop and evaluate a mechanism for randomizing the selection of a spanner based on the group address. This can alleviate the problem of traffic concentration when a single spanner is used for all multicast groups. Finally, we are working on a distributed algorithm for computing a spanner that does not require full knowledge of the topology. Current spanner algorithms require a centralized computation over the entire graph of the network; developing a distributed algorithm will allow the multicast routing protocol to be independent of any underlying unicast routing protocol.

7 Acknowledgments

We would like to thank Iyer Sivaramakrishna for producing appealing plots from an unappealing slew of numbers, and Nita Viswanath for running the program producing those numbers.

References

1. A. J. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees. In *ACM SIGCOMM*, August 1993.
2. U. Brandes and D. Handke. NP-Completeness results for minimum plannar spanners. *Discr. Math. and Theor. Comp. Sci.*, 3:110, 1998.
3. L. Cai. NP-completeness of minimum spanner problem. *Discrete Applied Mathematics*, 48:187–194, 1994.
4. L. Cai and D. G. Corneil. Tree spanners. *SIAM J. Discrete Mathematics*, 8:359–387, 1995.
5. K. Calvert and E. Zegura. Georgia Tech Internetwork Topology Models. http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.
6. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, and L. Wei. Protocol Independent Multicast Version 2 Dense Mode Specification. Internet Draft: work in progress, June 1999.
7. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An Architecture for Wide-Area Multicast Routing. In *ACM SIGCOMM*, August 1994.
8. C. Hedrick. Routing Information Protocol. RFC 1058, STD 0034, June 1988.
9. L. Kou, G. Markowsky, and L. Berman. A Fast Algorithm for Steiner Trees. *Acta Informatica*, 15:141–145, 1981.
10. Y. Mansour and D. Peleg. An approximation algorithm for minimum–cost network design. Technical Report CS94-22, Weizmann Institute of Science, Faculty of Mathematical Sciences, Jan. 1, 1994.
11. D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 77–85, Vancouver, British Columbia, Canada, 10–12 Aug. 1987.
12. D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. RFC 1075, November 1988.
13. D. W. Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Department of Electrical Engineering, Stanford University, 1980.
14. B. M. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988.
15. L. Wei and D. Estrin. The Trade-offs of Multicast Trees and Algorithms. In *1994 International Conference on Computer Communications Networks*, September 1994.
16. P. Winter. Steiner Problem in Networks: A Survey. *Networks*, 17(2):129–167, 1987.
17. E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *IEEE INFOCOM*, 1996.