Spanners and Message Distribution in Networks

Arthur Farley, Andrzej Proskurowski, and Kurt Windisch

CIS-TR-99-03
February 1999

Department of Computer and Information Science
University of Oregon

Spanners and Message Distribution in Networks

Arthur M. Farley, Andrzej Proskurowski
Computer and Information Science Department
and
Kurt Windisch
Advanced Network Technology Center, Computing Center

University of Oregon
Eugene, OR, 97403

We investigate the applicability of k-spanners as shared virtual topologies for intradomain broadcast and multicast on the Internet. A light-weight k-spanner is a spanning subgraph of a network that has both relatively small total edge weight and small multiplicative penalty (at most k) on all point-to-point distances. We define broadcasting and multicasting protocols based on flooding of a spanner, followed by pruning the distribution tree within the spanner and then joining in the network to form sender-based trees, if desired. We evaluate k-spanners (for $2 \leq k \leq 7$) in terms of number of edges, diameter, and average distance on sets of randomly generated graphs having properties similar to intradomain networks. We compare values of these metrics for spanners with those for given graphs and single-source minimum-distance spanning trees. Results show that a 3- or 4-spanner could serve as a good virtual topology for message distribution, having fewer edges than the full network yet lower average distance and diameter than a single-source, minimum-distance spanning tree.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network architecture and design

General Terms: Performance

Additional Key Words and Phrases: Multicast protocol, k-spanner, distribution tree

1. INTRODUCTION

It has become increasingly apparent that there is a need for efficient ways to distribute messages from single senders to more than one receiver within communication networks. Sending a different copy of the message to each receiver (i.e., mailing lists) can work effectively for applications involving infrequent, short transmissions. However, applications such as real-time video distribution or on-line gaming and conferencing, each requiring frequent distribution of relatively long messages, have resulted in the need for more efficient broadcasting and multicasting algorithms and associated protocols.

In this paper, we investigate the applicability of graph spanners as virtual topologies for broadcasting and multicasting in communication networks. First we define graph spanners and review relevant theoretical results. We then discuss how graph spanners could be incorporated into existing broadcast and multicast protocols.

Research supported in part by a National Science Foundation grant NCR-97-14680.

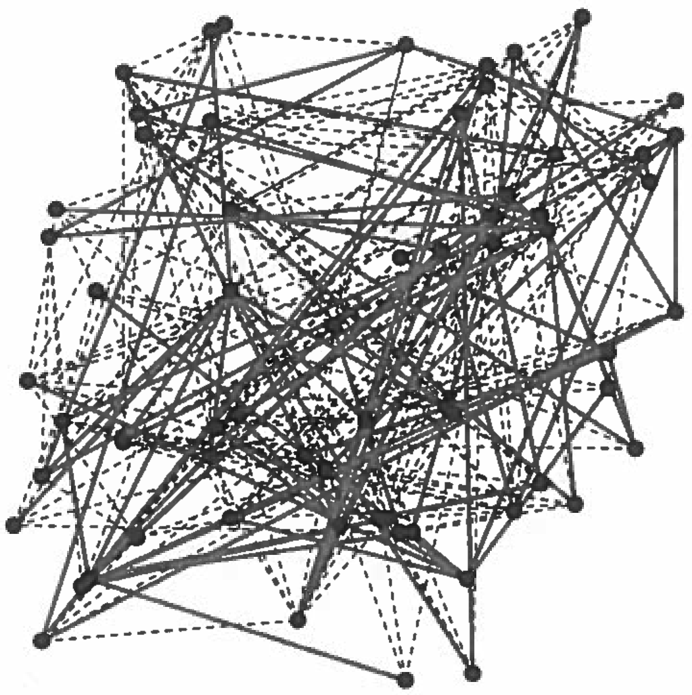


Fig. 1. A 5-spanner of a 64-vertex graph

We present the design of an experiment aimed at determining relevant properties of spanners of randomly generated graphs. We then discuss the results, noting key values that support the use of spanners for message distribution. In terms of Internet applicability, we see spanners playing a possible role in intra-domain contexts, where link-state topology information is reasonably available to the routers for computing the spanners.

1.1 Graph Spanners

The notion of graph spanners was introduced about a decade ago in [Peleg and Ullman 1987]. 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. In Figure 1 we show an example of a graph with 64 vertices with a 5-spanner displayed in gray edges.

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 [Cai 1994]. 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 [Cai and Corneil 1995; Brandes and Handke 1998].

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($G, k$)
// takes as input an arbitrary connected graph $G = (V, E)$
// and an integer $k$ >1
// gives as 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. [Mansour and Peleg 1994].

2. SPANNERS AND MESSAGE DISTRIBUTION

How might spanners be employed for message distribution in networks? One potential application is as a virtual topology for broadcasting. *Broadcasting* is the communication process whereby a message is sent from one sender to all other sites of a network, such as an administrative domain of the Internet. A need for broadcasting can arise in a number of network management and control contexts, as well as in implementations of distibuted algorithms. Without prior determination of a distribution tree, one common means for completing broadcast is to flood the network with the message. *Flooding* is the process whereby each vertex (site) sends a broadcast message out on every adjacent edge (interface) other than the one(s) on which it received the message. During flooding, the message traverses every edge at least once and some edges twice. Thus, the number of edges in a graph is a good approximation to the amount of traffic generated by a broadcast through flooding.

Given a k-spanner of a network's graph, rather than flooding all edges of the graph, one can just flood the message over edges of the spanner. If the original graph is connected, the spanner will be connected; thus, flooding over edges of the spanner will be sufficient to complete a broadcast. Questions arise as to how much reduction in message traffic this will realize and what will be the increased delay in the communication process. Later, we describe an experiment that we have conducted to provide some answers to these questions.

There are a number of multicast protocols under consideration for use in administrative domains on the Internet [Ballardie et al. 1993; Deering et al. 1996]. All rely upon establishing a distribution tree that covers the receivers of a multicast group. There are two primary ways for characterizing trees used by the protocols. In a protocol using sender-based trees, such as *DVMRP* or *PIM*-DenseMode (*PIM-DM*), each sender establishes a tree that includes shortest paths to all receivers. In a shared-tree protocol, such as *CBT* or *PIM*-SparseMode (*PIM-SM*), a single tree is established that covers all receivers of the group that is used by all senders to distribute messages.

Sender-based trees in *DVMRP* and *PIM-DM* are established by flooding the domain with the message from the sender. Subsequently, branches of the tree reaching no receiving members are pruned back until all leaves of the tree are the intended receivers. As noted above, one benefit of using a spanner to flood a message throughout a network will be a reduction in traffic generated by the initial stage of such dense-mode protocols. We describe important features of a modified version of *PIM-DM* called *PIM-Span*, which limits the set of links used for transporting multicast data to the edges of a spanner. We summarize key points of the *PIM-Span* protocol below:

- Most router implementations use at least two Route Information Bases (*RIBs*), one for unicast and one for multicast. The *RIBs* hold applicable route entries and next-hops to each reachable destination. In *PIM-Span*, a third *RIB* would be added to the router implementation that holds the set of multicast routes traversing only the spanner. Use of this *spanner RIB* ensures that only interfaces belonging to the spanner will be selected as incoming interfaces for a multicast and that flooding will be restricted to the spanner.
- The *PIM-Span* protocol requires the use of a link-state unicast routing protocol (*e.g.*, *OSPF* or *ISIS*). Link-state unicast protocols would distribute full topology information, and then each router would compute a k-spanner, for a particular k (the same for all routers). Routes corresponding to the spanner are then placed in the spanner *RIB* and used for flooding of messages from new senders.
- *PIM-DM* makes *RIB* lookups to determine the incoming interface for a new (S, G) entry and as a criterion for packet acceptance. In the *PIM-Span* protocol, a multicast router must always select an incoming interface on the spanner, one from which it received the message during flooding.
- When flooding data upon creation of a new sender–multicast group entry, (S, G), in each router, the *PIM-Span* protocol creates an outgoing interface list that includes only spanner interfaces from which it did not receive the message during flooding.
- Pruning of interfaces in *PIM-Span* is done as in *PIM-DM*, *i.e.*, when a router has no receivers in its local network and all downstream routers have sent prune messages. Periodically, *PIM-Span* can reflood the spanner to adjust the distribution tree to reflect new group membership, as does *PIM-DM*.

All *PIM-Span* message distribution is restricted to the spanner edges. This is a clear benefit during the flooding stage; however, the spanner-based trees established by pruning are not minimum-distance trees from the sender to group members in the given network. While the experimental results we present below indicate that the delay penalties are relatively minor for appropriate values of k, all flooding and subsequent traffic in multicast groups will be concentrated in the spanner, perhaps leading to traffic congestion. One way to deal with this is to create several spanners by randomly permuting the uniformly weighted edges, as considered by the spanner algorithm presented earlier. Different groups could use different spanners, with selection based on the group address; several spanner *RIBs* would be maintained, which is a small overhead.

We could also borrow a useful notion from *PIM-SM* to improve the applicability and scalability of the protocol. The multicast distribution tree can be switched to a minimum-distance tree by having certain routers send a join toward the sender, based upon a lookup in the unicast *RIB*. If the interface that leads to the sender in the network differs from the incoming interface in use on the spanner, a join message is sent on that interface. The join is propagated by upstream routers until either another active, receiving router or the sender is encountered. The protocol must be careful to ensure that no looping structure is created that would disconnect a part of the distribution tree from the sender. Once message data begins to appear on the new, incoming interface, a prune message can be sent up the spanner-based incoming interface, thereby switching (part of) the tree to a shortest-path (instead of spanner-based) tree from the sender in the given network. Switching to a sender-based tree leads to better traffic distribution and somewhat less delay for large messages.

The *PIM-Span* protocol with tree-switching capability represents an interesting combination of dense-mode and sparse-mode behaviors that avoids, on the one hand, the complete flooding of the network associated with dense-mode operations and, on the other hand, the designation of a vulnerable core or rendezvous point associated with sparse mode behavior. Using a common, shared structure for flooding, and switching to a shortest-distance, sender-based tree when traffic load warrants it appears to capture valuable elements of both approaches. A full specification of *PIM-Span*, including the tree-switching mechanism, is still to be completed.

3. EXPERIMENT DESIGN

The *PIM-Span* multicast protocol discussed above is viable only if spanners exhibit considerable savings over the entire network topologies in terms of number of edges, while not incurring significant penalty in terms of increased average or maximum message delay. Our goal is to compare these properties of spanners with those of given network graphs and with single-source, minimum-distance spanning trees of these graphs. In our experiment we choose an arbitrary vertex as the single source of such spanning trees. Such trees are meant to represent shared trees or core-based trees that reflect use of a single rendezvous point, as employed by several, proposed multicast routing protocols [Ballardie et al. 1993; Deering et al. 1996].

We want the given network graphs 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 [Zegura 1997]. 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 ([Zegura et al. 1996]). The locality method we used is the basic Waxman model, [Waxman 1988] 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 incraeses 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 the graphs 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. For each test set of 50 graphs, we compute the average of each comparison ratio, as well as the minimum, twenty-fifth percentile, seventy-fifth percentile, and maximum of the ratios.

4. EXPERIMENT RESULTS

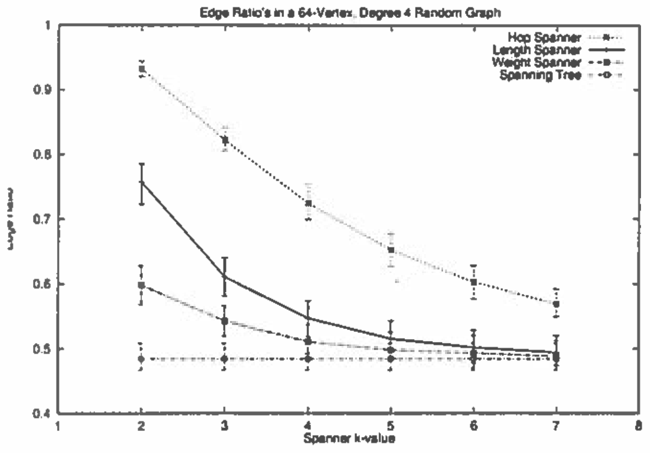
The experiment as designed above, generates a complex array of results. There are 8 types of random graphs, based on number of vertices, average vertex degree, and whether there was a locality preference in generating edges. Then there are 3 edge-distance measures applied to edges: hop, length, and weight. 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). Our results indicate little or no difference between the three test sets for the same parameter settings. So, we report the results from the first set of graphs for each parameter combination. All results are available for viewing at URL [OurWebPage 1999].

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 [Wei and Estrin 1994] 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.

4.1 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 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 constructed for these two distance measures have nearly identical behavior as k varies.



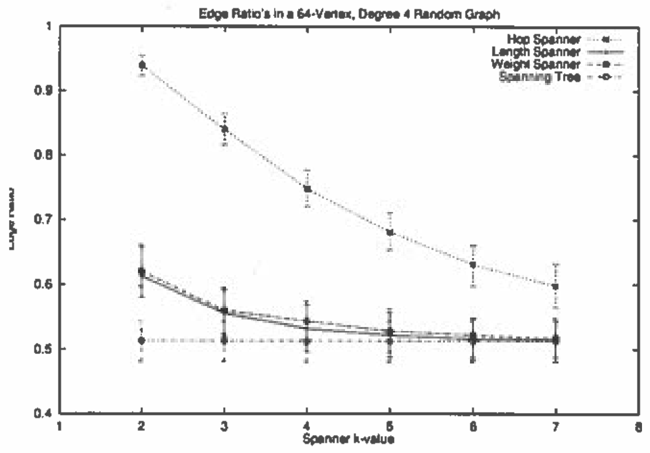


Fig. 2. Edge ratios for the two random graph models (locality preference on the right).

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, 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.

4.2 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.

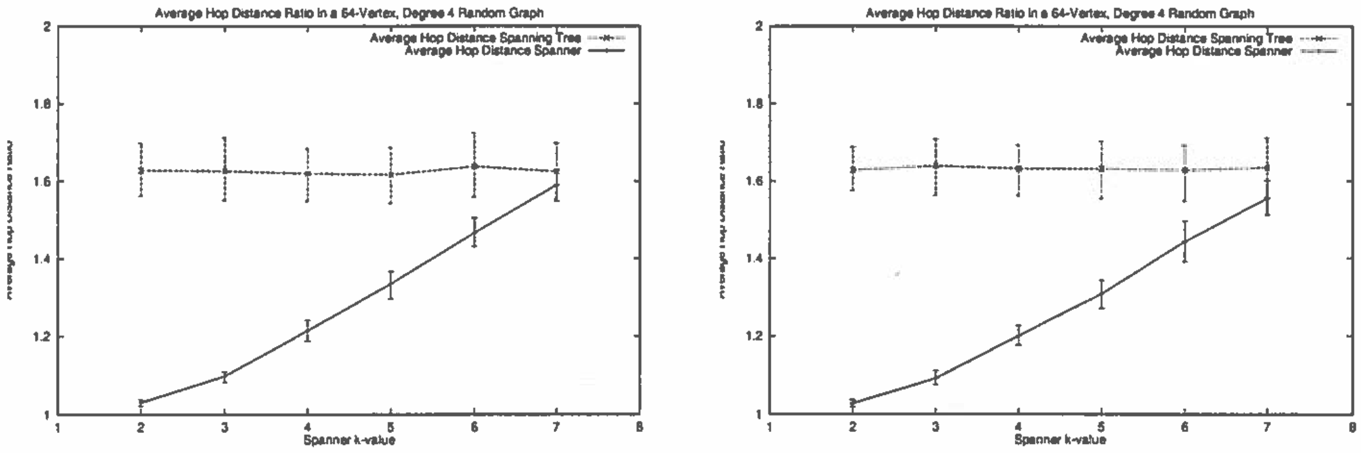


Fig. 3. Average hop distance for the two random graph models (locality preference on the right).

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. 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

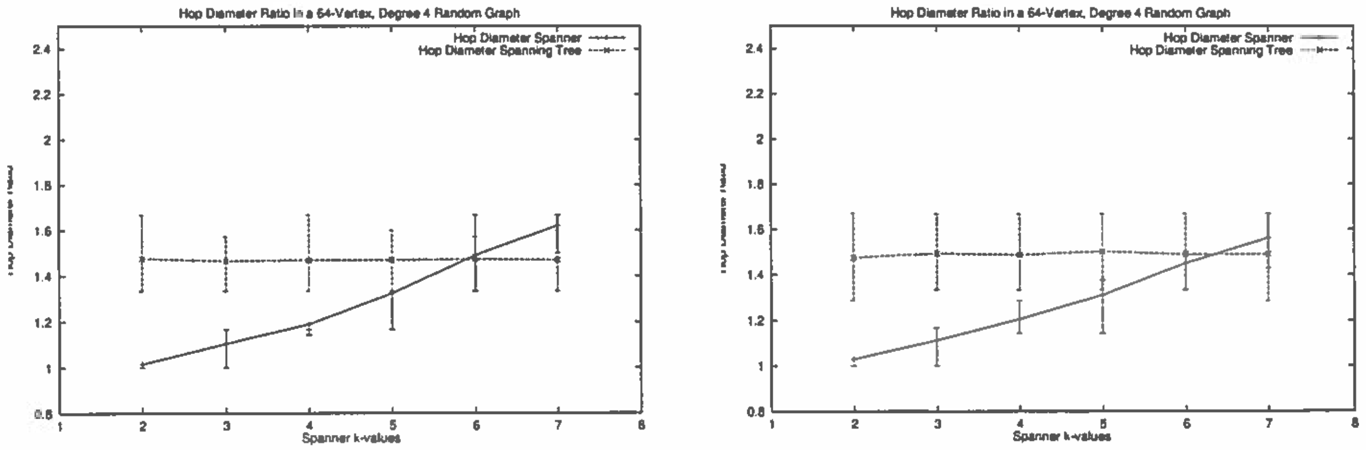


Fig. 4. Average hop diameter for the two random graph models (locality preference on the right).

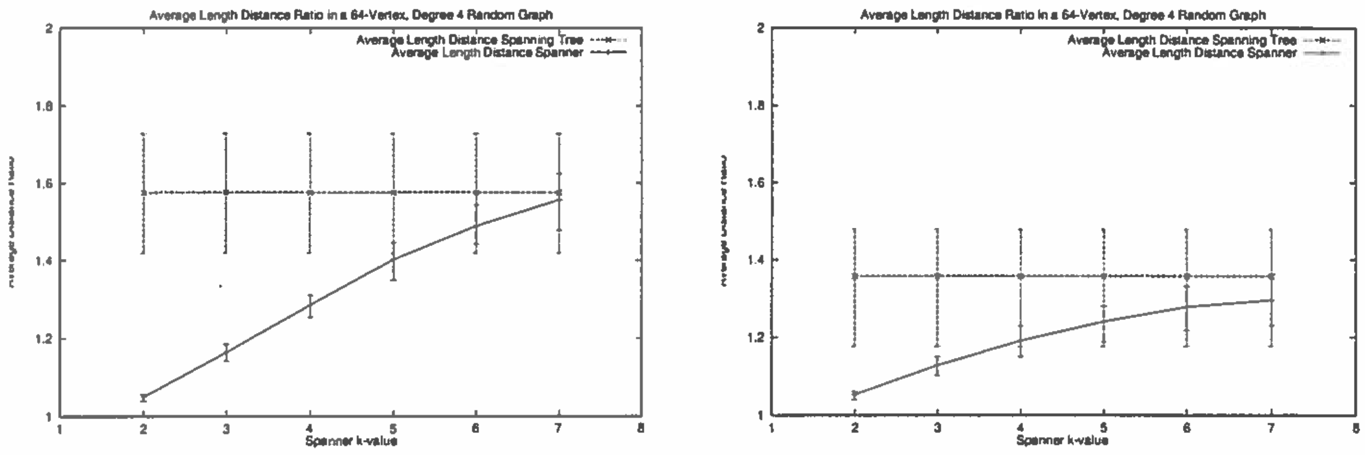


Fig. 5. Average length distance ratios (locality preference on the right).

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.

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.

4.3 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 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 trese 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 sapnners of degree-8 graphs.

Now we turn our attention to 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.

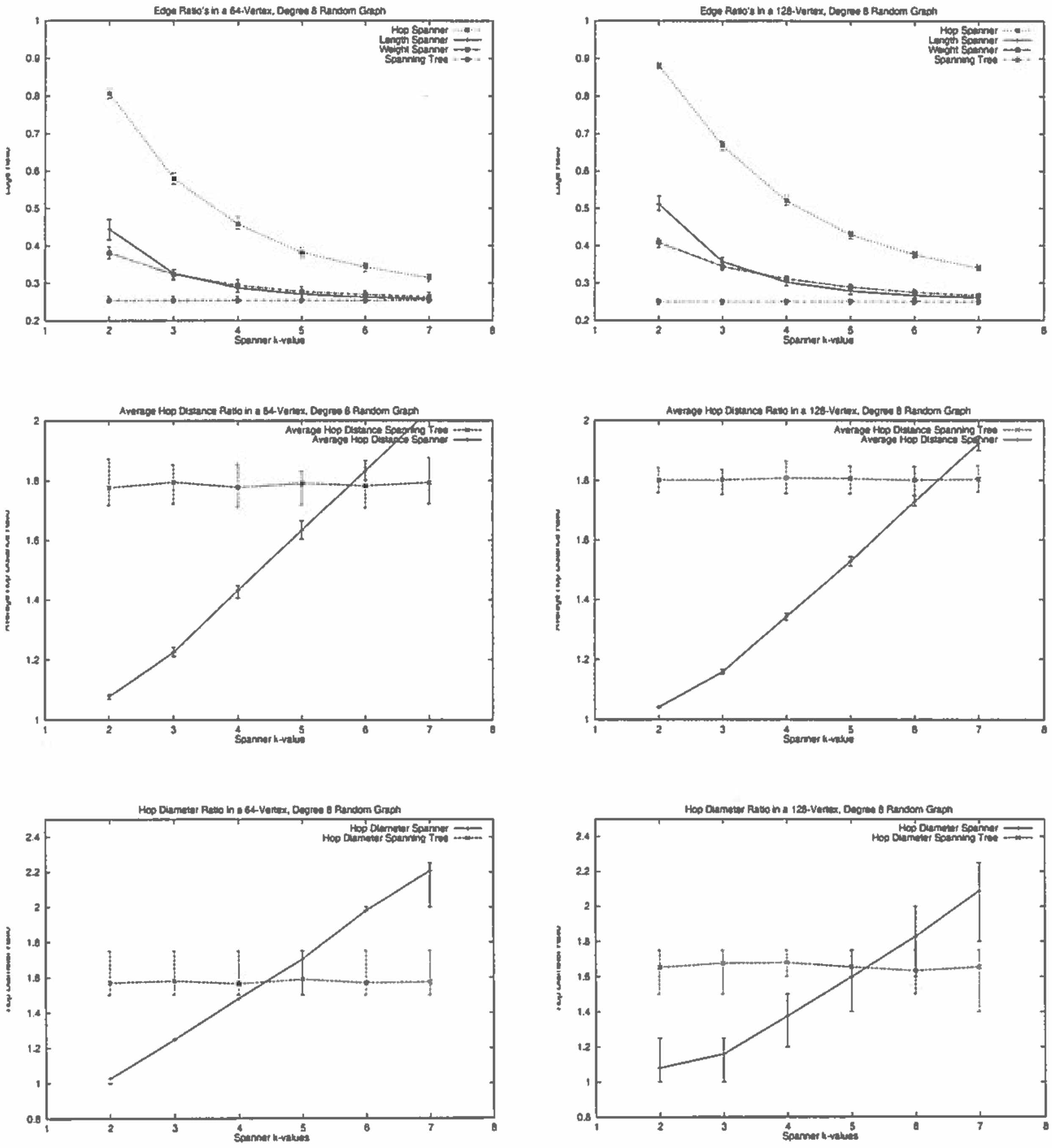


Fig. 6. Representative results for degree-8 graphs.

5. CONCLUSION AND FUTURE RESEARCH

In this paper, we have presented two main results regarding the potential use of graph spanners as virtual topologies for message distribution in networks. First, we describe a multicast protocol *PIM-Span* that employs a spanner as basis for forming the distribution tree from any sender. While not all details of the protocol design have been decided, its close relationship to existing *PIM* protocols makes completing this protocol specification a reasonable next step.

Second, we conduct what we believe to be the first experimental analysis of k-spanners with respect to edge ratio and distance ratio metrics. While a k-spanner guarantees maximum stretch less than or equal to k, as expected, average distances and diameters increase by much less. We also address the issues of how these metrics are affected by number of vertices, average vertex degree, and any locality preference in constructing the given graph.

The results show that a spanner represents an interesting intermediate virtual topology between the original network graph, which minimizes distances, and a spanning tree, which minimizes number of edges. For the ranges of graph sizes (number of vertices) and average vertex degrees that typify adminstrative domains on the Internet, k-spanners, for $3 \leq k \leq 5$, appear to be the most applicable.

We have a number of research questions to pursue as a result of this study. One is clearly to complete a specification of *PIM-Span*. Another is to investigate the number of k-spanners that exist in a given graph and to determine the extent to which they share edges. An ability to select a number of partially edge-disjoint spanners would be valuable for improving the scalability of spanner use as traffic levels increase and possible congestion develops in a network. Finally, spanner-related problems for selected sender and receiver sets within a network also offer the potentials for new, interesting results.

ACKNOWLEDGMENTS

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

REFERENCES

Ballardie, A. J., Francis, P. F., and Cowcroft, J. 1993. Core based trees. In *Proc. ACM SIGCOMM, San Francisco* (1993).

Brandes, U. and Handke, D. 1998. Np-completeness results for minimum plannar spanners. *Discr. Math. and Theor. Comp. Sci. 3*, 110.

Cai, L. 1994. Np-completeness of minimum spanner problem. *Discrete Applied Mathematics 48*, 187–194.

Cai, L. and Corneil, D. G. 1995. Tree spanners. *SIAM J. Discrete Mathematics 8*, 359–387.

Deering, S., Estrin, D., Farinacci, D., Jacobson, V., Liu, C., and Wei, L. 1996. An architecture for wide-area multicast routing. *IEEE/ACM Trans. Networking.*

Mansour, Y. and Peleg, D. 1994. An approximation algorithm for minimum-cost network design. Technical Report CS94-22 (Jan. 1,), Weizmann Institute of Science, Faculty of Mathematical Sciences.

OurWebPage. 1999. http://www.cs.uoregon.edu/. URL.

Peleg, D. and Ullman, J. 1987. An optimal synchronizer for the hypercube. In *Proc. Sixth Annual ACM Symp. on Princ. Distr. Comp.* (August 1987), pp. 77–85. ACM.

Waxman, B. M. 1988. Routing of multipoint connections. *IEEE J. Selected Areas Comm. 6.*

Wei, L. and Estrin, D. 1994. The trade-offs of multicast trees and algorithms. In *Proc. Int'l Conf. Comp. Comm. Netw.* (Sept. 1994).

Zegura, E. W. 1997. http://www.cc.gatech.edu/fac/ellen.zegura/gt-itm/. URL.

Zegura, E. W., Calvert, K. L., and Bhattacharjee, S. 1996. How to model an internetwork. In *Proceedings of Infocom, San Francisco* (1996).