Core Migration for Dynamic Multicast Routing

Michael J. Donahoo Ellen W. Zegura

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract

Efficient multicast communication provides one of the best avenues for scaling bandwidth intensive applications. Current multicast routing mechanisms perform well for multicast groups which reach a long-term, steady-state distribution of membership. However, when the membership migrates throughout the network, current routing techniques may perform poorly. Recent techniques for IP multicast involve selection of a "core" router through which all multicast communication is routed. We propose to adapt this routing scheme to efficiently support dynamic groups, by allowing the core to migrate. As part of our work we provide solutions to several problems that are more general, including (1) estimating multicast tree bandwidth using unicast information and (2) a variation on soliciting responses from a limited number of participants to avoid message implosion.

1 Introduction

Core-based multicast routing[3, 7, 6] provides a good mechanism for scalable multicasting; however selecting a "good" core is difficult if the make-up of the group changes during the lifetime of the group connection (e.g., Distributed Interactive Simulation (DIS) [12]). Changes in these types of applications are not predictable *a priori* since movement occurs in response to evolving situations.

1.1 Related Work

Current core-based multicast routing protocols (CBT[2] and PIM[8]) do not adequately address the issue of performance-based core choice, assuming an administrative decision. Work is in progress on a scheme to organize candidate cores into a multi-level global hierarchy [13]. Some discussion is given regarding the migration to a different top-level core based on coarse delay information. Both the performance metrics and the migration options are considerably more limited than those we describe.

Researchers have recently begun to consider the relationship between performance and core placement [5, 11, 16], but with emphasis on the one-time placement of the core based on some *a priori* knowledge of the application. In [14, 17, 18], the authors consider the use of dynamic Steiner trees to handle dynamic multicast routing, but these algorithms are not well suited for large networks or distributed implementation. Several other authors [9, 15] have investigated incremental changes to multicast routing trees to accommodate changes in receivers.

1.2 Contributions

Our contributions include the following:

- A flexible model of dynamic applications, capturing the essential behavior that impacts routing algorithms.
- A scalable method for selecting and evaluating candidate cores for migration. In addition to dynamic multicast, this method would be useful for determining secondary cores to provide robustness, even when a fixed, primary core will be used.
- A method for estimating the bandwidth utilization for multicast distribution, using unicast routing information. This method has broader applicability, for example in deciding whether to deliver information (e.g., from a server) via unicast or multicast.
- A method for efficiently soliciting responses from a selected number of receivers in the multicast group, using a combination of probabilistic multicast [1] and probing rounds [4]. This method can be useful whenever message implosion is a concern.

2 Core Based Multicast Routing

We use a graph $G = (V, E, W)$ to represent the network, where V is the set of vertices, representing routers or switches, E is the set of edges, representing logical connectivity, and $W(u, v)$ is a function mapping edges to weights ($W(u, v) \rightarrow \mathbb{R}$ such that $(u, v) \in E$). We use the weight function to reflect routing policy by controlling the characteristics of the routes found by a shortest-path routing algorithm. Note that the edges represent *logical* connectivity between vertices, for example, four routers connected to a FDDI ring would be represented by a clique of four vertices. Also note that we do not explicitly represent hosts; the participation of a host in a multicast group is reflected via its local router that is responsible for maintaining multicast group information for all hosts in an area.

We define an *instance* of multicast communication to be a set of *sources* ($S \subseteq V$) and a set of *receivers* ($R \subseteq V$). We refer to $S \cup R$ as the *participants* of the multicast group. For each source $s \in S$, the multicast routing algorithm determines a *distribution tree*, $T(s, R)$. This tree is a subgraph of G, that specifies the path from the source to the receivers taken by the multicast packets. The leaves of T should be exactly the source s and the receivers R.

2.1 Metrics

We consider the following quantities of interest in characterizing algorithm performance:

Average(Maximum) Delay The average (maximum) number of links traversed by any packet in traveling from a source to a receiver.

Bandwidth The total number of (single hop) packet transmissions required to deliver a packet from each source node to all receiver nodes.

2.2 Core Migration Algorithm

The outline of the core migration algorithm is as follows:

1 Determine a set of candidate cores.
2 Request from each candidate core, c', the expected performance, were c' to be the core.
3 If the candidate core with the best performance is significantly better than the current core, migrate to this best candidate.

There are two major issues associated with this algorithm:

Candidate core selection: Select a pool of candidate cores large enough to include core likely to improve performance but small enough to keep control message overhead small, preserving scalability.

Candidate core evaluation: Perform distributed evaluation of the each core in the candidate core pool.

In Section 6 we bring together the details of each step into a complete version of the algorithm, and explore the performance with all components working together.

3 Modeling

Previous work in multicast routing relies on very simple specifications of group activity[9, 18, 17]. We extend the concept of group activity specification to allow evaluation of our algorithm in the contexts of various "real-world" scenario simulations. We use the term *scenario* to describe a time-dependent characterization of an application's behavior. Scenarios, as opposed to simulating with purely random behavior, provide a framework for classification of applications. We could use this application classification to allow the migration algorithm to adapt on an application-specific basis (e.g., to determine how often to consider core movement) instead of trying to develop one general migration algorithm that works well for all applications. In this paper, however, we focus on a general algorithm that contains some adaptation based on on-line observations. See [10] for a complete description of the scenarios.

We characterize a scenario by the following parameters:

Migration Probability The probability that a new participant is generated in the domain that is not represented by the current set of participants.

New Participant The probability a new participant appears in a time interval.

Life The percentage of the total simulation time that a participant exists.

Participant Source The probability a new participant is a source.

All of these parameters can be time-dependent distribution functions. For example, the New Participant probability is a function mapping the percentage of time that has elapsed in the simulation to the probability of a new participant generation event. This can be used, for example, to model applications that have many participants join early in the lifetime of the group but few or none that join after a certain percentage of time has elapsed.

We select four particular scenarios for study.

Scenario VC (Video Conferencing)
Scenario IS (DIS)
Scenario RC (Relay Chat)
Scenario SM (Seminar)

4 Selecting Candidate Cores

Before suggesting a policy for selecting candidate cores, we examine the question of where the good cores are located. Utilizing simulation results, we found that for all scenarios and metrics, the average stub makes the worst choice. For the most part, the optimal transit node performs as well as any optimal node. The next best cores vary between the average transit and optimal stub nodes. Further investigation reveals that when the group has a small number of receivers and sources concentrated in a single stub domain, the optimal core tends to be a stub node, specifically a node from the domain with all the participants. See [10] for a detailed analysis.

We now propose the following candidate selection policy which attempts to exploit the previous observations regarding core location and performance: *one candidate core is chosen from each transit domain. In addition, one candidate is chosen from the stub domain with the most representation by sources and the stub domain with the most representation by receivers.* If the network has a prohibitively large number of transit domains, we augment this policy with the specification that candidates be chosen from randomly selected transit domains. By selecting a random core from each transit domain, we approximate using the average transit node from each transit domain. Selecting a random node from the most populated stub domains approximates selecting the best stub node since, if the participants are highly localized, we know that a very good core likely comes from that locale.

5 Evaluating Candidate Cores

We obtain information[1] to use in performance evaluation by soliciting help from the participants (i.e., the

[1] Routers may have distance information for some participants. In this paper, we do not address using this type of information, but adapting our approach to use routing distance information is trivial. Note that to use router distance infor-

designated routers for the sources and receivers). Thus we propose augmenting the control messages that are part of the multicast protocol to include messages that facilitate core migration (See [10] for message details). This allows a candidate core to know the hop count from itself to each participant in the group.[2] Each candidate core then evaluates and reports its cost to the current core.

Let r denote the number of receivers and d_i denote the distance (in hops) from the candidate core to receiver i. Let s denote the number of senders and s_i denote the distance (in hops) from sender i to the candidate core. The delay performance metrics are easy to compute. The average and maximum delay for messages from source i is:

$$\begin{aligned} D_{avg}(i) &= s_i + \frac{1}{r}\sum_{j=0}^{r-1} d_j \\ D_{max}(i) &= s_i + \max_j\{d_j\} \end{aligned}$$

The bandwidth is not directly computable from the distance information, since bandwidth depends on the amount of overlap in the shortest paths from the core to the receivers. In order to estimate the multicast bandwidth, we first derive an upper bound, and then relax the bound to obtain an estimate.

5.1 Evaluating Multicast Bandwidth

We want to determine an upper bound on the bandwidth performance at a candidate core, corresponding to the *least* amount of overlap in the shortest paths reaching the receivers. Note that in addition to the distance information, the candidate core also knows the outgoing interface for each receiver. Thus the set of receivers can be partitioned according to outgoing interface, and the bandwidth can be evaluated separately over each interface. We also assume that the candidate core knows the maximum degree of any node in the network, denoted Δ.

Since each interface can be treated separately, we develop an upper bound for bandwidth to the receivers reached over a given interface. We modify the earlier notation to let r denote the number of receivers reached over the given interface, and d_i denote the distance from the *interface node* to receiver i. (Note that this is one less than the distance from the candidate core to receiver i.) The least amount of path overlap (and hence the largest multicast bandwidth) results when the shortest path tree from the candidate to the receivers branches as much as possible as close to the core as possible. The degree of branching at a node is limited to $\Delta - 1$ by the node degree constraint.

The worst case distribution tree is one which is "full" for as many levels as possible (each node branches by $\Delta-1$). A multicast distribution tree containing a full subtree, S, with i levels reaches at least $(\Delta-1)^i$ nodes downstream from the leaves of S. The maximum depth of S is then the maximum value h such that $|\{d_i | d_i \geq h\}| \geq (\Delta-1)^h$. The intuition is that there must be enough receivers at distance h or greater from the candidate to allow full branching to a depth of h. The link $(C, 1)$ from the candidate core to the interface node, plus the full subtree S, contribute multicast bandwidth equal to $\sum_{i=0}^{h}(\Delta-1)^i$.

After building a full subtree, S, to depth h, there may be some "extra" receivers that are at distances greater than h. Note that the number of extras is strictly less than $(\Delta-1)^{h+1}$, otherwise the full tree could be extended by one more level. Since there are fewer than $(\Delta-1)^{h+1}$ extra receivers, they can be reached from the leaves of S, using disjoint paths to each extra receiver. Thus the multicast bandwidth to the receivers out the given interface is bounded by:

$$\sum_{i=0}^{h}(\Delta-1)^i + \sum_{j\ s.t.\ d_j > h}(d_j - h)$$

where the first term accounts for the full subtree, S, and the second term accounts for the extra receivers.

The total bandwidth, B, for a candidate core is the sum of the bandwidth from each source to all receivers. Each source unicasts to the candidate then multicasts using the receiver distribution tree. Recall that s_i denotes the distance from source i to the candidate, and s denotes the total number of sources. Let B_j denote the maximum multicast bandwidth to the receivers out interface j for the candidate core. The total bandwidth for a multicast from all sources, B, is bounded by $\sum_i s_i + s\sum_j B_j$. Each candidate core computes the upper bound and sends this as an estimate of bandwidth to the current core. If the current core decides to perform core migration, this occurs using a core change protocol, such as in PIM [8].

We evaluate the effectiveness of this method by inspecting the ratio of our estimated multicast bandwidth to the actual multicast bandwidth over randomly generated instances. Figure 1 shows the ratio as a function of the number of receivers. Each data point corresponds to a randomly chosen core and set of receivers. The striking feature of this plot is the tendency of the data to fall into three bands. Further investigation reveals that the bands are distinguished by the *proximity* of the core to a transit domain. The points marked by the diamond are transit nodes, those marked by the plus are stub nodes directly connected to transit nodes, those marked by the box are stub nodes two hops away from a transit node, and those marked by the "x" are three or more hops from a transit node. The lowest band consists entirely of transit nodes and directly connected stub nodes. The middle band is made up of stubs that are two hops away from a transit node. The intuition is that the closer the core is to a transit domain, the more accurate the assumption about maximum node degree (since transit nodes have higher average node degree than stubs). In addition, transit nodes are nearer to the "middle" of the network, thus the shortest paths from a transit

mation requires knowledge of receivers which the current IP multicast architecture does not provide.

[2] The hop counts to the receivers are actually for the reverse path; in this paper, we assume paths are symmetric and defer dealing with the issue of asymmetric paths.

core to a set of random receivers are more likely to be disjoint.

This result is significant since our candidate cores come primarily from transit domains. While the estimation can be off by a factor of four or more for some stub nodes, it is generally within a factor of two for transit nodes.

Our analysis of multicast bandwidth above is an upper bound that assumes branching by the maximum possible (i.e., $\Delta - 1$) at each level of the tree. A single node with large degree can influence the tightness of our upper bound. Relaxing this, we can use the average node degree to get a closer estimation to the actual multicast bandwidth. (Of course, we no longer have a guaranteed upper bound.) Repeating the previous experiment using that average node degree instead of the maximum, we see the same banding effect when using average node degree; we note that the estimation for transit nodes is now generally within a factor of 1.25.

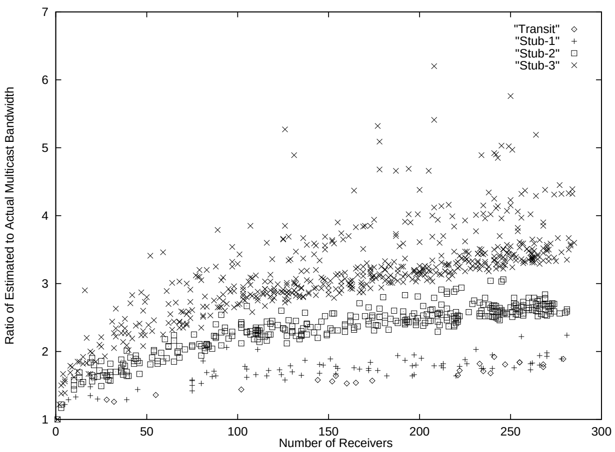

Figure 1: Performance of Multicast Estimation using Maximum Degree

We note that Thaler and Ravishankar [16] also give a method to estimate bandwidth from information on distances to receivers. Their estimate is the average of a lower and upper bound; their upper bound is not as tight as ours, using only the degree at the tree root to tighten the naive bound consisting of the sum of the distances to all receivers. Their lower bound is fairly straightforward based maximizing the amount of path overlap.

5.2 Random Participant Evaluation

At this point, we have presented an algorithm which, using participant feedback, evaluates the performance at candidate cores. Unfortunately, the message complexity of the algorithm increases with the number of participants. This not only creates additional network load but can lead to message implosion at the candidate cores. In this section we introduce Random Participant Evaluation, which works in the same manner as above except that it utilizes probabilistic multicast[1] to control the number of responses sent by participants to each candidate core. The header of the multicast packet contains an acceptance probability, α, which each participant uses to determine whether or not to respond. That is, each receiver will send a message to the candidate core with probability α. The candidate cores will use the information obtained from the limited set of participants to estimate performance. In what follows we describe our method for estimating the number of participants, and then evaluate the accuracy of the performance estimates.

5.2.1 Number of Participants Determination

To maintain scalability, we must determine α based on the number of participants. We adapt the receiver congestion discovery protocol proposed by Bolot [4] to determine the number of receivers. Our algorithm chooses an initial value of α, and the core sends a probabilistic multicast to the receivers. The initial value of α is chosen conservatively, allowing for few or no responses. We modify the receiver response to send a message to the current core, in addition to sending a unicast message to each candidate core. If no responses are received at the current core in the initial round,[3] α is increased, and a probabilistic multicast, again, solicits responses from receivers. This process is repeated until a response is received or $\alpha = 1$. (Note that when $\alpha = 1$, every receiver will respond.) We estimate the number of receivers based on the first round in which a response is received. Our analysis demonstrates that the expected first response round and probability of implosion for our technique is similar to Bolot's results. See [10] for analysis of expected first response round and probability of implosion.

5.2.2 Estimating Performance with Limited Participation

Each candidate core must estimate performance given data about the unicast routes between itself and a limited set of group participants. We now must estimate average and maximum delay, in addition to bandwidth, since we do not have the full knowledge necessary to compute exact delays. We use the average delay of the limited participants as the estimate of average group delay, relying on the assumption that the limited participants are, on average, a good representation of the total group. Our experimentation shows that even a small percentage of participants can give a good approximation to the average delay. For maximum delay, we know that the limited participants will not give us a accurate estimate, unless we happen to have heard from the furthest source and destination. However, all of the candidates cores will be similarly limited in accuracy, thus for the purposes of comparison we use the maximum delay of the limited participants.

[3] A timeout of twice the maximum round trip time is used to determine the end of a round. See[4] for a description of the round trip time discovery.

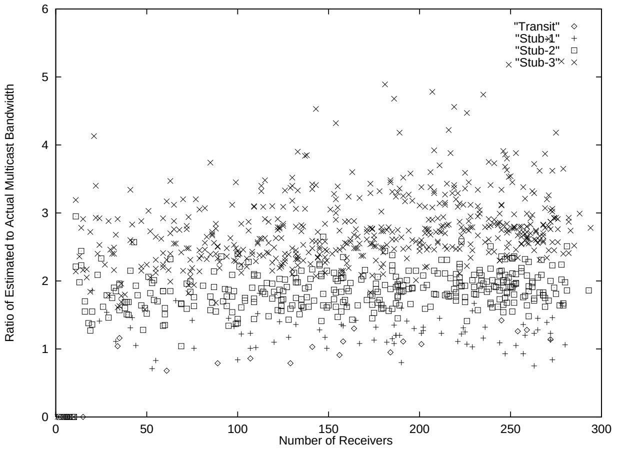

Figure 2: Multicast Bandwidth Estimation using 10% Participation

To estimate bandwidth, we assume that each responding participant serves as a "proxy" for the participants that have not responded. Thus, if $\alpha = .1$ and we hear from a receiver that is five hops away, we assume there are 10 receivers ($1/\alpha$) at distance five hops. We then calculate multicast bandwidth using the algorithm of Section 5.1.

Figure 2 shows the ratio of estimated to actual multicast bandwidth, with 10% of the participants providing data to the candidate core. As before, we have distinguished the data based on proximity of the core to a transit domain. The bands are still evident, although they are not as cleanly separated as with full participation. The accuracy for transit and directly connected stub nodes is still quite good, with most values in the range 0.75 to 1.25. It is somewhat surprising to note that the accuracy is relatively independent of the group size, whereas it depended more on group size for full participation.

6 Putting It All Together

In this section, we merge all of the concepts presented in the previous sections into an evaluation algorithm, and we examine the performance of the core migration scheme including the overhead required for distributed execution. We address the following issues to accomplish this integration:

Performance Evaluation The performance metric of interest depends on the application e.g., an interactive video conference may be very concerned about maximum delay, while the participants in a video lecture may be more concerned with bandwidth. Our simulation supports a composite performance metric that is a weighted combination of delay, maximum delay and bandwidth.

Evaluation Algorithm The full core evaluation algorithm brings together all of the pieces described earlier, including core selection policy, estimating the number of receivers using probabilistic multicast within probing rounds, and estimating performance at candidate cores. (Note that the parameter *threshold* in Step 3 is application specified. It determines what constitutes "significantly" better performance to justify core migration.)

1. **Determine the estimated number of receivers (Subsection 5.2.1)**
2. **Select and notify C, the set of candidate cores (Section 4)**
3. **Decide if and where to migrate core (Section 5)**

Evaluation Frequency The core evaluation overhead scales linearly with the frequency of its use. Prematurely performing core evaluation (the core evaluation does not select a new core) incurs unnecessary overhead cost; while overly infrequent core evaluation (the core evaluation would have selected a significantly better core) results in inefficient multicast routing. The correct frequency of evaluation depends on the dynamics of the multicast application. For example, the video-conferencing scenario has significant membership changes in the first 10% of the application and few changes afterwards; therefore, core evaluation should happen frequently at the start of the connection, diminishing over the lifetime of the connection. Conversely, the relay chat scenario has significant membership changes throughout the life of the connection; consequently, core evaluation should happen often during the connection lifetime.

We propose a simple, adaptive mechanism for core evaluation frequency based on the result from the previous core evaluation.

> For each core evaluation which moves the core, we additively increase the frequency of core evaluation up to a maximum. For each core evaluation which does not move the core, we multiplicatively decrease the frequency of core evaluation.

This approach is conservative about increasing the frequency of core evaluation, and quick to back off if core evaluations do not result in migration.

7 Conclusion

We evaluate the performance of our core migration algorithm on each of the four scenarios described earlier (See [10] for detailed simulation description). Figure 3 shows the ratio of the bandwidth, average delay, and maximum delay performance with no migration to our core migration scheme for scenarios RC (left) and scenario SM (right). Results for the other scenarios are similar. See [10]. The migration decisions are based exclusively on the bandwidth metric. The B/W(Amortized) plot is the ratio of the non-migration bandwidth to the sum of the migration bandwidth and the amortized core evaluation overhead (amortized over the all multicasts during the same interval).

Table 1 gives the average improvement of migration over non-migration for each scenario and performance

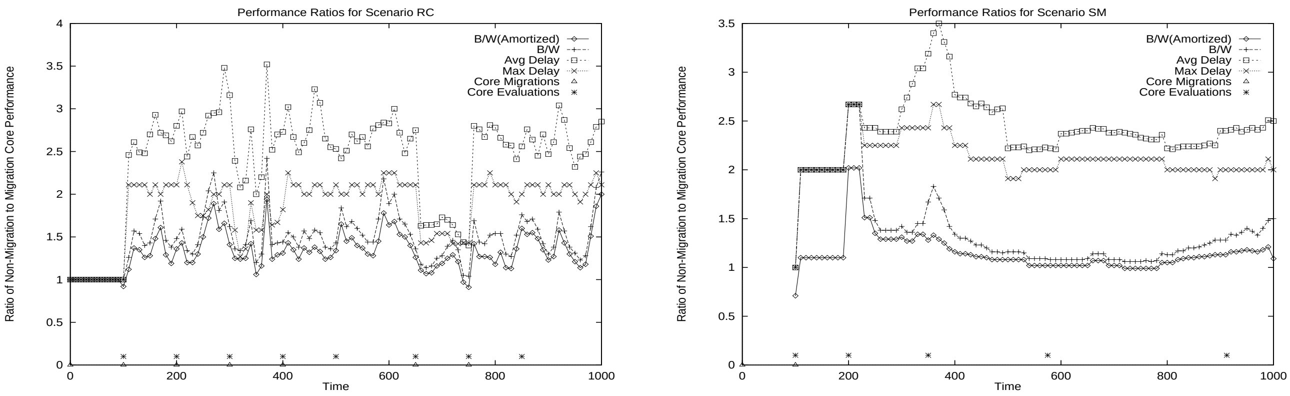

Figure 3: Performance Ratios with Bandwidth Migration Criteria

Scen.	B/W (Amor.)	B/W	Avg Delay	Max Delay
IS	22%	28%	137%	74%
RC	32%	48%	139%	85%
SM	15%	37%	142%	112%
VC	8%	10%	117%	74%

Table 1: Average Relative Scenario Performance

measure. We use the simulation results to reach the following conclusions about core migration:

Bandwidth Our core migration algorithm provides a modest, consistent improvement in bandwidth performance. In fact, even migration bandwidth with amortized overhead costs are always at least as good as non-migration bandwidth costs.

Delay Even when using the bandwidth metric to make core migration decisions, maximum and average delay improve markedly (e.g. on average 142% for average delay in the SM scenario) [4]

Adaptation The points near the x-axis indicate when core evaluation and migration occurs. Note from Figure 3 that the more dynamic scenarios (e.g. IS and RC) have frequent core evaluation while less dynamic scenarios (e.g. SM and VC) have significantly fewer core evaluations. This demonstrates that the adaptive core evaluation frequency mechanism presented in Section 6 does adapt itself to application behavior, reducing core evaluation overhead for less dynamic applications.

Overhead We allow flexibility in controlling overhead by determining the frequency and scope of evaluation.

Avenues for future research include improving upon the set of parameters used for multicast scenario control, investigating the relationship between performance and graph topology, further refining the candidate core selection policy and comparing the core migration performance to other core selection methods.

[4] Reducing multicast bandwidth correlates closely with reducing delay; using delay as the core migration criteria produces results similar to these.

Acknowledgments

The authors would like to acknowledge useful discussions of this work with Ken Calvert.

References

[1] Mostafa H. Ammar. Probabilistic multicast: Generalizing the multicast paradigm to improve scalability. In *Proceedings of INFOCOM 1994*, pages 848–55, 1994.

[2] A. J. Ballardie. Core-based trees (cbt) multicast architecture. Technical report, IDMR Internet Draft, February 1996.

[3] Tony Ballardie. Core based tree (CBT) multicast - architectural overview and specification. Internet Draft, November 1994.

[4] Jean-Chrysostome Bolot, Thierry Turleeti, and Ian Wakeman. Scalable feedback control for multicast video distribution in the internet. In *Proceedings of SIGCOMM 1994*, pages 58–67, 1994.

[5] Kenneth L. Calvert, Ellen W. Zegura, and Michael J. Donahoo. Core selection methods for multicast routing. In Kia Makki and Niki Pissinou, editors, *Proceedings of the ICCCN '95*, pages 638–642. IEEE, IEEE Computer Society Press, Sept. 1995.

[6] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching gung Liu, and Liming Wei. Protocol independent multicast (PIM): Motivation and architecture. Technical Report draft-ietf-idmr-pim-arch-01, IDMR Working Group, Jan. 1995.

[7] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An architecture for wide-area multicast routing. In *Proceedings of SIGCOMM 1994*, pages 126–135. ACM, ACM Press, 1994.

[8] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, Liming Wei, Puneet Sharma, and Ahmed Helmy. Protocol independent multicast - sparse mode (PIM-SM): Protocol specification. Technical report, IETF-IDMR, July 1995.

[9] Matthew Doar and Ian Leslie. How bad is naive multicast routing? In *Proceedings of INFOCOM 1993*, pages 82–9, 1993.

[10] Michael J. Donahoo and Ellen W. Zegura. Core migration for dynamic multicast routing. Technical Report GIT-CC-95-28, Georgia Institute of Technology, 1995.

[11] Deborah Estrin. Draft note on RP selection for PIM-SM. IETF-IDMR, Mar 1994.

[12] Institute for Simulation and Training. Standard for Distributed Interactive Simulation — application protocols (version 2.0.4 draft IEEE standard). Technical Report IST-CR-94-50, University of Central Florida, March 1994.

[13] M. Handley, J. Crowcroft, and I. Wakeman. Hierarchical protocol independent multicast (HPIM). Technical Report http://www.cs.ucl.ac.uk/staff/M.Handley/hpim.ps, University College London and University of Sussex, 1995. Work in progress.

[14] Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM Journal Disc. Math*, 4(3):369–84, August 1991.

[15] James Kadirire and Graham Knight. Comparison of dynamic multicast routing algorithms for wide-area packet switched (Asynchronous Transfer Mode) networks. In *Proceedings of INFOCOM 1995*, pages 212–219, 1995.

[16] D.G. Thaler and C.V. Ravishankar. Distributed center-location algorithms: Proposals and comparisons. In *IEEE Infocom '96*. IEEE, March 1996.

[17] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–22, December 1988.

[18] Bernard M. Waxman. Performance evaluation of multipoint routing algorithms. In *Proceedings of INFOCOM 1993*, volume 3, pages 980–6, March 1993.