Routing in Overlay Multicast Networks

Sherlia Y. Shi Jonathan S. Turner
Department of Computer Science
Washington University in St. Louis
{*sherlia, jst*}*@cs.wustl.edu*

Abstract— **Multicast services can be provided either as a basic network service or as an application-layer service. Higher level multicast implementations often provide more sophisticated features, and can provide multicast services, where no network layer support is available. *Overlay multicast networks* offer an intermediate option, potentially combining the flexibility and advanced features of application layer multicast with the greater efficiency of network layer multicast. Overlay multicast networks play an important role in the Internet. Indeed, since Internet Service Providers have been slow to enable IP multicast in their networks, Internet multicast is only widely available as an overlay service. This paper introduces several routing algorithms that are suitable for overlay multicast networks and evaluates their performance. The algorithms seek to optimize the end-to-end delay and the *interface bandwidth usage* at the routing sites within the overlay network. The interface bandwidth is typically a key resource for an overlay network provider, and needs to be carefully managed in order to maximize the number of sessions that can be served. The simultaneous optimization of both delay and bandwidth is an NP-hard problem. We propose several heuristic algorithms and simulate their performance under various traffic conditions and on various network topologies.**

I. Introduction

Multicast communication is an important part of many next generation networked applications, including video conferencing, video-on-demand, distributed interactive simulation (including large multi-player games) and peer-to-peer file sharing. Multicast services allow one host to send information to a large number of receivers, without being constrained by its network interface bandwidth. This makes applications more scalable and leads to more efficient use of network resources. The limited network layer support for multicast in the Internet today, has made it necessary for applications requiring multicast services to obtain services at a higher level. In *application layer multicast*, hosts participating in an application session share responsibility for forwarding information to other hosts [7, 9, 11, 12, 15]. While highly flexible, this approach places a significant additional burden on end hosts, and does not scale well to large group sizes. *Overlay multicast networks* provide multicast services through a set of distributed *Multicast Service Nodes* (MSN), which communicate with hosts and with each other using standard unicast mechanisms. Overlay networks effectively use the Internet as a lower level infrastructure, to provide higher level services to end users. The multicast backbone, Mbone [5], is the best-known multicast overlay network, but multicast services are also a part of commercial overlay network services, such as Akamai [1] and iBeam [10].

Because overlay multicast networks are built on top of a general Internet unicast infrastructure, rather than point-to-point links, the problem of managing their resource usage is somewhat different than in networks that do have their own links. One of the principal resources that an overlay network must manage is the access bandwidth to the Internet at the MSNs' interfaces. This interface bandwidth represents a major cost, and is typically the resource that constrains the number of simultaneous multicast sessions that an overlay network can support. Hence, the routing algorithms used by an overlay multicast network, should seek to optimize its use.

In addition to optimizing MSN interface bandwidth, a multicast routing algorithm should ensure that the selected routes do not contain excessively long paths, as such paths can lead to excessively long packet delays. However, the objective of limiting delay in a multicast network can conflict with the objective of optimizing the interface bandwidth usage, so multicast routing algorithms must strike an appropriate balance between these objectives. Reference [13] introduced the overlay multicast routing problem and studied the performance of two algorithms. The results in [13] showed that optimizing the interface bandwidth usage, produced a gain of up to 50% on the overlay network utilization. However, there was still a significant gap between the achieved performance and a computed performance bound, suggesting the possibility of further improvements. In this paper, we briefly review these two algorithms and introduce a new algorithmic strategy that takes a more direct approach to optimizing the MSN interface bandwidth. We describe several specific algorithms based on this strategy and examine the performance of two of them in detail. The algorithms are evaluated using simulation and a range of traffic conditions and network configurations. Our results show that we can improve the performance by 10% to 20% while still satisfing the same end-to-end delay bound.

The rest of the paper is organized as follows: in section II, we briefly present the two multicast routing algorithms that were developed earlier. Our new strategy for overlay multicast routing is presented in section III. In section IV, we compare the performance of these routing algorithms on different network topologies and under various traffic conditions. In section V, we discuss issues related to dynamic membership control and implementation issues, and in section VI we discuss some of the related works. Finally we conclude in section VII.

II. BACKGROUND

An overlay multicast network can be modeled as a complete graph since there exists a unicast path between each pair of MSNs. For each multicast session, we create a shared overlay multicast tree spanning all MSNs serving participants of a session, with each tree edge corresponding to a unicast path in the underlying physical network. The amount of available interface bandwidth at an MSN imposes a constraint on the degree of that node in the multicast tree. We let $d_{max}(v)$ denote this degree constraint at node v.

There are two natural formulations of the overlay multicast routing problem. The first seeks to minimize diameter while respecting the degree constraints.

Definition 1: **Minimum diameter, degree-limited spanning tree problem(MDDL)**
Given an undirected complete graph $G = (V, E)$, a degree bound $d_{max}(v) \in N$ for each vertex $v \in V$ and a cost $c(e) \in Z^+$ for each edge $e \in E$; find a spanning tree T of G of minimum diameter, subject to the constraint that $d_T(v) \leq d_{max}(v)$ for all $v \in T$.

The MDDL problem is NP-hard. Reference [13] introduced a heuristic for MDDL, referred to here as the *Compact Tree* (CT) algorithm. It is a greedy algorithm and builds a spanning tree incrementally. We let $\delta(v)$ denote the length of the longest path from vertex v to any other node in the partial tree T constructed so far. For each vertex v that is not yet in the partial tree T, we maintain an edge $\lambda(v) = \{u, v\}$ to a vertex u in the tree; u is chosen to minimize $\delta(v) = c(\lambda(v)) + \delta(u)$. At each step, we select a vertex v with the smallest value of $\delta(v)$ and add it and the edge $\lambda(v)$ to the tree. Then, for each vertex v, not yet in the tree, we update $\lambda(v)$.

The second natural formulation of the overlay multicast routing problem seeks the "most balanced" tree, that satisfies an upper bound on the diameter. To explain what is meant by "most balanced", we first define the *residual degree* at node v with respect to a tree T as $res_T(v) = d_{max}(v) - d_T(v)$, where $d_T(v)$ is the degree of v in T. To reduce the likelihood of blocking a future multicast session request, we should choose trees that maximize the smallest residual degree. Since the sum of the degrees of all multicast trees is the same for a given session size, this strategy works to "balance" the residual degrees of different vertices. Any tree that maximizes the smallest residual degree is a "most balanced" tree.

Definition 2: **Limited diameter, residual-balanced spanning tree problem(LDRB)**
Given an undirected complete graph $G = (V, E)$, a degree bound $d_{max}(v)$ for each $v \in V$, a cost $c(e) \in Z^+$ for each $e \in E$ and a bound $B \in Z^+$; find a spanning tree T of G with diameter $\leq B$ that maximizes $\min_v res_T(v)$, subject to the constraint that $d_T(v) \leq d_{max}(v)$, for all $v \in V$.

Like the MDDL problem, the LDRB problem is NP-hard. Reference [13] introduced a heuristic for LDRB, referred to here as the *Balanced Compact Tree* (BCT) algorithm. The algorithm can be viewed as a generalization of the CT algorithm. Like the CT algorithm, it builds the tree incrementally. However, at each step it first finds the M vertices that have the smallest values of $\delta(v)$ and from this set, it selects a vertex v with $\lambda(v) = \{u, v\}$, which maximizes the smaller of $res_T(u)$ and $res_T(v)$, where T is the current partial tree. The parameter M may be varied to trade-off the goals of residual degree balancing and diameter minimization. Specifically, when $M = 1$, it is equivalent to the CT algorithm and when M is equal to the number of vertices in the multicast session, BCT concentrates on balancing the residual degrees. Simulation studies have shown that fairly small values of M are effective in achieving good balance, without violating the diameter bound.

We evaluate overlay multicast routing algorithms using a simulation in which new multicast sessions start and end at random times. The primary performance metric is the fraction of sessions that are rejected because no multicast route can be found, either due to the failure to satisfy the diameter bound or due to the exhaustion of interface bandwidth of at least one MSN. We also obtain a lower bound on the rejection probability using a simulation in which a multicast session is rejected only if the MSN interface bandwidth required by the session exceeds the total unused interface bandwidth at all MSNs (including those not involved in the session). Results reported in [13] showed that by distributing the load more evenly across servers, the BCT algorithm rejects substantially fewer multicast sessions than the CT algorithm on the same network configuration. At the same time, there remained a significant gap between the achieved performance and the potential suggested by the lower bound. While the bound was not expected to be tight, the size of the gap suggested that there was room for improvement. In the next section, we introduce a new strategy for overlay multicast routing that leads to an algorithm with uniformly better performance.

III. Balanced Degree Allocation

In the BCT algorithm, a new node is always attached to the tree at the point that yields smallest diameter in the resulting tree. Although the algorithm changes the sequence of node selection by first selecting nodes with larger residual degrees, it does not guarantee the use of these nodes as intermediate nodes in the tree, i.e. nodes with higher degree fanout. This prevents the BCT algorithm from achieving the best-possible residual degree balance. *Balanced Degree Allocation* (BDA) is a strategy for constructing multicast trees that approaches the problem in a fundamentally different way. It starts by determining the ideal degree of each node in the multicast session, with respect to the objective of maximizing the smallest residual degree.

To state the strategy precisely, we need to stretch our definition of the residual degree of a vertex. Let k denote the multicast session size (or interchangeably session fanout). First, we define a *degree allocation* d_A to be a function from the vertices of a multicast session to the positive integers that satisfies two properties: (1) $\sum_v d_A(v) = 2(k-1)$, where k is the number of participants in the multicast session (so, $2(k-1)$ is the sum of the vertex degrees in any tree implementing the multicast session); and (2) there are at least two vertices u and v with $d_A(u) = d_A(v) = 1$. A partial degree allocation is a similar function in which the first property is replaced with $\sum_v d_A(v) \le 2(k-1)$. Now, define the residual capacity of a vertex with respect to a partial degree allocation d_A as $res_A(v) = d_{max}(v) - d_A(v)$.

We can compute a degree allocation that maximizes the smallest residual degree as follows.

- For each vertex v in the multicast session, initialize $d_A(v)$ to 1.
- While $\sum_v d_A(v) < 2(k-1)$ select a vertex v that maximizes $res_A(v)$ and increment $d_A(v)$.

This procedure actually does more than maximize the smallest residual degree. It produces the most balanced possible set of residual degrees, by seeking to "level" the residual capacities as much as possible. This is illustrated in Figure 1.

Given a degree allocation for a tree, we would like to construct a tree in which the vertices have the assigned degrees and which satisfies the limit on the diameter. There is a general procedure for generating a tree with a given degree allocation, which is described below.

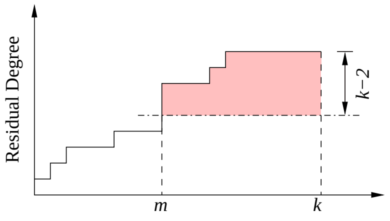


Fig. 1. Balanced Degree Allocation

The procedure builds a tree by selecting *eligible pairs* of vertices, and adding the edge joining them to a set of edges F that, when complete, will define the tree. The *spare degree* of a vertex v, is its allocated degree $d_A(v)$ minus the number of edges in F that are incident to vertex v. At any point in the algorithm, the edges in F define a set of connected components. A pair of vertices $\{u, v\}$ is an eligible pair if the following conditions are satisfied.

- u, v are not in the same component;
- both u, v have spare degree ≥ 1;
- either, there are only two components remaining, or the sum of the spare degrees of the vertices in the components containing u and v is greater than 2.

The last condition above is included to ensure that each newly formed component has a spare degree of at least one, so that it can still be connected to other components in later steps. Of course, this condition is not needed in the last step.

This process is guaranteed to produce a tree with the given degree allocation, and all trees with the given degree allocation are possible outcomes of the process. We can get different specific tree construction algorithms by providing different rules for selecting the vertices u and v.

One simple and natural rule is to select the closest pair u and v. Call this the *Closest Pair* (CP) algorithm. Since the CP algorithm does nothing to directly address the objective of diameter minimization, it may not produce a tree that meets the diameter bound. An alternative selection rule is to select the pair $\{u, v\}$ that results in the smallest diameter component in the collection of components constructed as the algorithm progresses. This algorithm is referred to as the *Compact Component* (CC) algorithm.

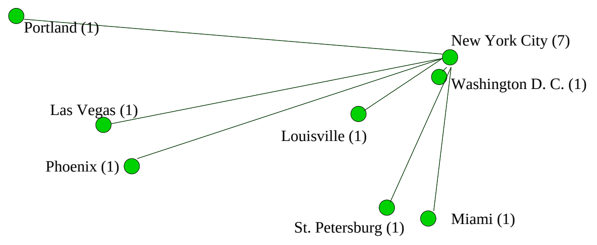
One can also use a selection rule that builds a single tree incrementally. In this rule, we consider all eligible pairs of vertices u and v, for which either u is in the tree built so far, or v is in the tree, but not both. Among all such pairs, we pick the one that results in the smallest diameter tree. This procedure is repeated for every choice of initial vertex, and the smallest diameter tree kept. This algorithm is the same as the CT algorithm described before, but with the original input degree constraint d_{max} replaced with the much tighter degree allocation d_A.

Any of the selection rules described, will produce a tree with the desired degree allocation. However, the resulting tree may not satisfy the bound on diameter. We can reduce the diameter of the trees by using a less balanced degree allocation. To reduce the diameter, we can increase the degree allocation of "central vertices" while decreasing the degree allocation of "peripheral vertices", where central and peripheral are relative to vertex locations. A vertex u is more central than a vertex v if its radius $max_w c(\{u, w\}) < max_w c(\{v, w\})$. Unfortunately, we have found that natural strategies for adjusting degree allocations lead to only marginal improvement in the diameters of the resulting trees. It appears difficult to find good degree allocations, independently of the tree building process. We have found that a more productive approach is to use a "loose degree allocation" and allow the tree-building process to construct a suitable tree satisfying the degree limits imposed by the loose allocation. Loose degree allocations are derived from the most balanced allocation by allowing small increases in the degrees of vertices.

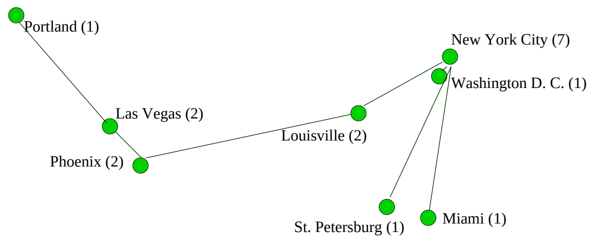
By combining the tree building procedure, with a specific rule for selecting eligible pairs and a procedure for "loosening" a degree allocation, we can define iterative algorithms for the overlay multicast routing problem. We start with a balanced degree allocation and build a tree using that allocation. If the resulting tree satisfies the diameter bound, we stop. Otherwise we loosen the degree allocation and build a new tree. We continue this process until we find a tree with small enough diameter, or until a decision is made to terminate the process and give up.

The degree loosening procedure increases by 1, the degree allocation of up to b vertices, where b is a parameter. The first application of the procedure adds 1 to the degree allocation of the b most central vertices. If incrementing the degree allocation for a vertex v would cause its degree allocation to exceed the degree bound for v, then v's degree allocation is left unchanged. The second round of the procedure adds 1 to the next b most central vertices (again, so long as this would not cause their degree bounds to be exceeded). Subsequent applications affect the degree bounds of successive groups of b vertices, and the process wraps around to the most central vertices, after all vertices have been considered. The process can be stopped after some specified number of applications of the degree loosening procedure, or after all degree allocations have been increased to the smaller of their degree bounds and $k - 1$ where k is the session fanout.

If we select eligible pairs for which the connecting edge has minimum cost, the resulting algorithm is called *Iterative Closest Pair* (ICP) algorithm. If we select eligible pairs so as to minimize the diameter of the resulting component, the algorithm is called *Iterative Compact Component* (ICC) algorithm. In the *Iterative Compact Tree* (ICT) algorithm, we select eligible pairs, with one vertex of each pair in a single tree being constructed, and the other selected to minimize the tree diameter. This procedure is repeated for all possible initial vertices.



(a) Longest Path: Portland → New York → Phoenix; Length = 9416.67 km



(b) Longest Path: Portland → Las Vegas → Phoenix → Louisville → New York → St. Petersburg; Length = 7823.64 km

Fig. 2. An Example of the ICT Algorithm with Degree Adjustment

An example execution of the ICT algorithm appears in Figure 2. For simplicity, we used geographical distance as routing cost and a diameter bound of 8000 km. Initially, the BDA output dictates the creation of a star topology with New York City as the center with degree of 7; this exceeds the diameter bound. In the second round, the degree allocation of the three nodes with small radius, Las Vegas, Phoenix, Louisville, is loosened by 1; this results in a smaller diameter tree satisfying the diameter bound. We observe that the actual degree allocation is still close to the balanced degree allocation.

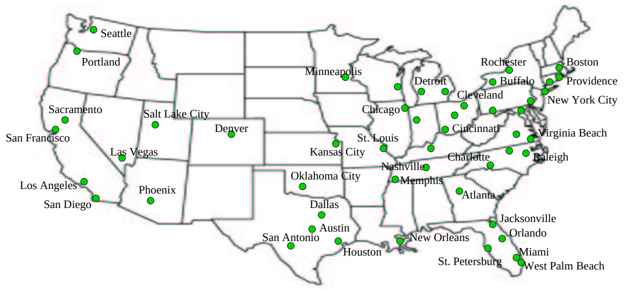
IV. Evaluation

This section reports simulation results for the overlay multicast routing algorithms described above. We report results for three network topologies and a range of multicast session sizes. The principal performance metric is the multicast session rejection rate. We also evaluate the multicast tree diameter and computation times of the algorithms. Comparisons with the CT and BCT algorithms are also included.

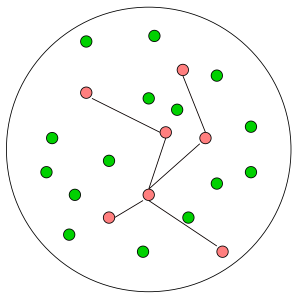
A. *Simulation Setup*

We have selected three overlay network configurations for evaluation purposes. The first (called the metro configuration) has an MSN at each of the 50 largest metropolitan areas in the United States [14]. The "traffic density" at each node is proportional to the population of the metropolitan area it serves. We use a Poisson session arrival process and the session holding times follow a Pareto distribution. Session fanouts follow a truncated binomial distribution with a minimum of 2 and maximum of 50, and means varied in different result sets. All multicast sessions are assumed to have the same bandwidth. Different MSNs were assigned different interface bandwidths, depending on their traffic density and their location. MSNs in more central locations are assigned higher interface bandwidths than those in less central locations, since it is more efficient for multicast sessions to branch out from these locations than from the more peripheral locations. The assignment of interface bandwidth at MSNs is critical to the performance of the routing algorithms. We have dimensioned the network to best carry a projected traffic load given a specific routing algorithm. This is done by routing a projected traffic load on a network configuration and computing the average carried load for each MSN. Then we assign access link bandwidth to each server proportional to its carried load subject to a fixed total bandwidth capacity for the entire network. The dimensioned network is further fine-tuned by additional rounds of routing with the actual routing algorithm. Details of the MSN dimensioning process can be found in [13].

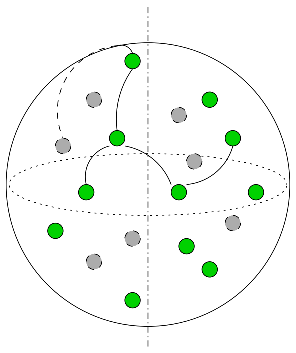
The metro configuration was chosen to be representative of a realistic overlay multicast network. However, like any realistic network, it is somewhat idiosyncratic, since it reflects the locations of population centers and the differing amounts of traffic they produce. The other two configurations were chosen to be more neutral. The first of these consists of 100 randomly distributed nodes on a disk and the second consists of 100 randomly distributed nodes on the surface of a sphere. In both cases, all nodes are assumed to have equal traffic densities. In the disk, as in the metro configuration, the MSN interface bandwidths must be dimensioned, but in this case it is just a node's location that determines its interface bandwidth. In the sphere configuration, all nodes are assigned the same interface bandwidth, since there is no node that is more central than any other.



(a) 50 Largest U.S. Metropolitan Areas



(b) Disk Configuration



(c) Sphere Configuration

Fig. 3. Overlay Network Configurations

The three network configurations are illustrated in Figure 3. In all configurations, the geographical distance between two nodes is taken as the edge cost in the multicast session tree.

B. *Comparison of Tree Building Techniques*

In the previous section, we suggested three basic tree building techniques: selecting the closest pair (CP), selecting the pair that minimizes the component diameter (CC), and selecting the pair that minimizes the single tree diameter (CT). The iterative versions of these algorithms, namely ICP, ICC and ICT, seek to satisfy the diameter bound by loosening the degree allocation produced by BDA. In this section, we examine their performance sensitivities to different diameter bounds and to the number of rounds allowed for degree adjustment. The simulation uses the metro configuration as the network topology and a session fanout of 10.

Figure 4 shows the session rejection rates versus the ratio of the diameter bound to the maximum inter-city delay (6000 km). In this simulation, we allow each algorithm to loosen the degree allocation as much as it needs to (stopping when the degree allocation reaches the smaller of nodes' degree bounds or $k-1$, where k is the session size). The horizontal line labeled BDA, shows the rejection rate using the balanced degree allocation strategy, but ignoring the diameter of the resulting tree.

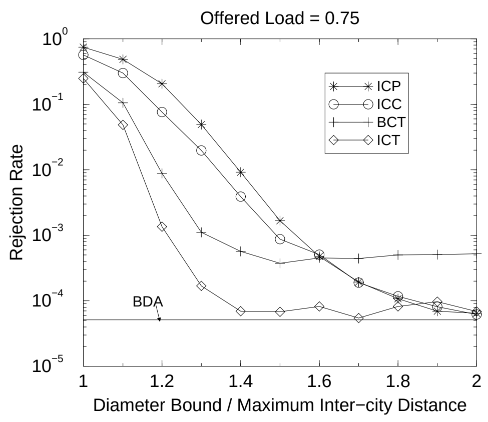


Fig. 4. Sensitivity to Diameter Bound

As the large cities in this map are along the coastal areas, the majority of the sessions will span across the continent. Therefore, it is difficult to find a multicast tree for these sessions when the diameter bound is tight, resulting in very high rejection rates for all algorithms. However, as the diameter bound is relaxed, the rejection rate improves for all the algorithms, with the iterative algorithms all achieving essentially the same performance for diameter bounds of more than 1.8 times the maximum inter-city distance. At intermediate diameter bounds, the ICT and BCT algorithms perform better than the ICC and ICP algorithms. This suggests that building from a single tree, as both ICT and BCT do, is more effective in minimizing the tree diameter. The BCT algorithm does not allocate its node degree before building the tree; rather, it seeks to maximize the residual bandwidth as the algorithm progresses. For large diameter bounds, BCT is not able to reduce its rejection rate as much as the algorithms using balanced degree allocation.

Figure 5 shows the rejection ratio versus the maximum number of degree adjustment rounds allowed in ICP, ICC and ICT. The diameter bound is fixed at 8000 km for this simulation. In each degree adjustment round, the number of vertices being adjusted is 3. Generally, the ICP and ICC algorithms only benefit from the very first few rounds of degree adjustment, which allows nearby nodes to be joined together to form collections of small forests. The additional rounds of degree adjustment have no effect on them and the rejection rates remain relatively high. Contrarily, the ICT algorithm benefits greatly from the additional rounds of degree adjustment. It is able to utilize the increased degree allocation at the centrally located nodes and form smaller diameter trees.

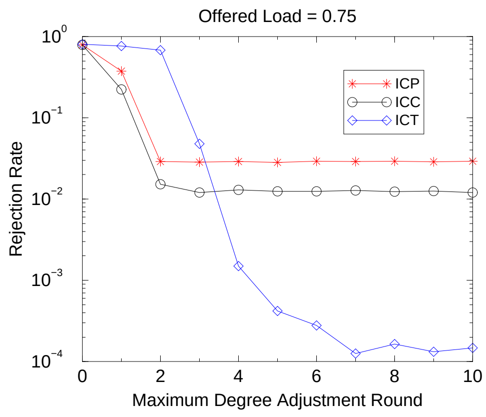


Fig. 5. Sensitivity to Degree Adjustment Round

We conclude in this subsection that the ICT algorithm, when combined with the degree loosening procedure, is more effective at producing small diameter trees than ICP and ICC. In the rest of this paper, we will focus our evaluation mainly on the ICT algorithm. However, ICT's greater effectiveness comes with a cost of added complexity, as it iterates through each possible starting vertex in order to find the best tree. We will analyze the computational cost of the ICT algorithm later in the section.

C. Performance Results – Rejection Rate

Figure 6 shows the session rejection rates versus offered load for a subset of the overlay multicast routing algorithms presented earlier. The charts also include a lower bound on the rejection fraction that was obtained by running a simulation in which a session is rejected only if the sum of the degree bounds at all nodes in the network is less than $2(k-1)$ where k is the number of nodes in the session being set up. The lower bound curves are labeled LB. The results, labeled BDA, are obtained using the balanced degree allocation strategy, and ignoring the diameter of the resulting tree. We conjecture that this also represents a lower bound on the best possible rejection fraction that can be obtained by any on-line routing algorithm. It is certainly a lower bound for algorithms based on the balanced degree allocation strategy.

For these charts, the diameter bound for the metro configuration is 8000 km which is approximately 1.5 times the maximum distance between nodes. For the disk and sphere topology, the bound is two times the disk diameter and three times the sphere diameter; each is about twice the maximum distance between any two nodes in the respective topology. The total interface bandwidth for all MSNs is 10,000 times the bandwidth consumed by a single edge of a multicast session tree. So, in the sphere, each MSN can support an average of 100 multicast session edges and

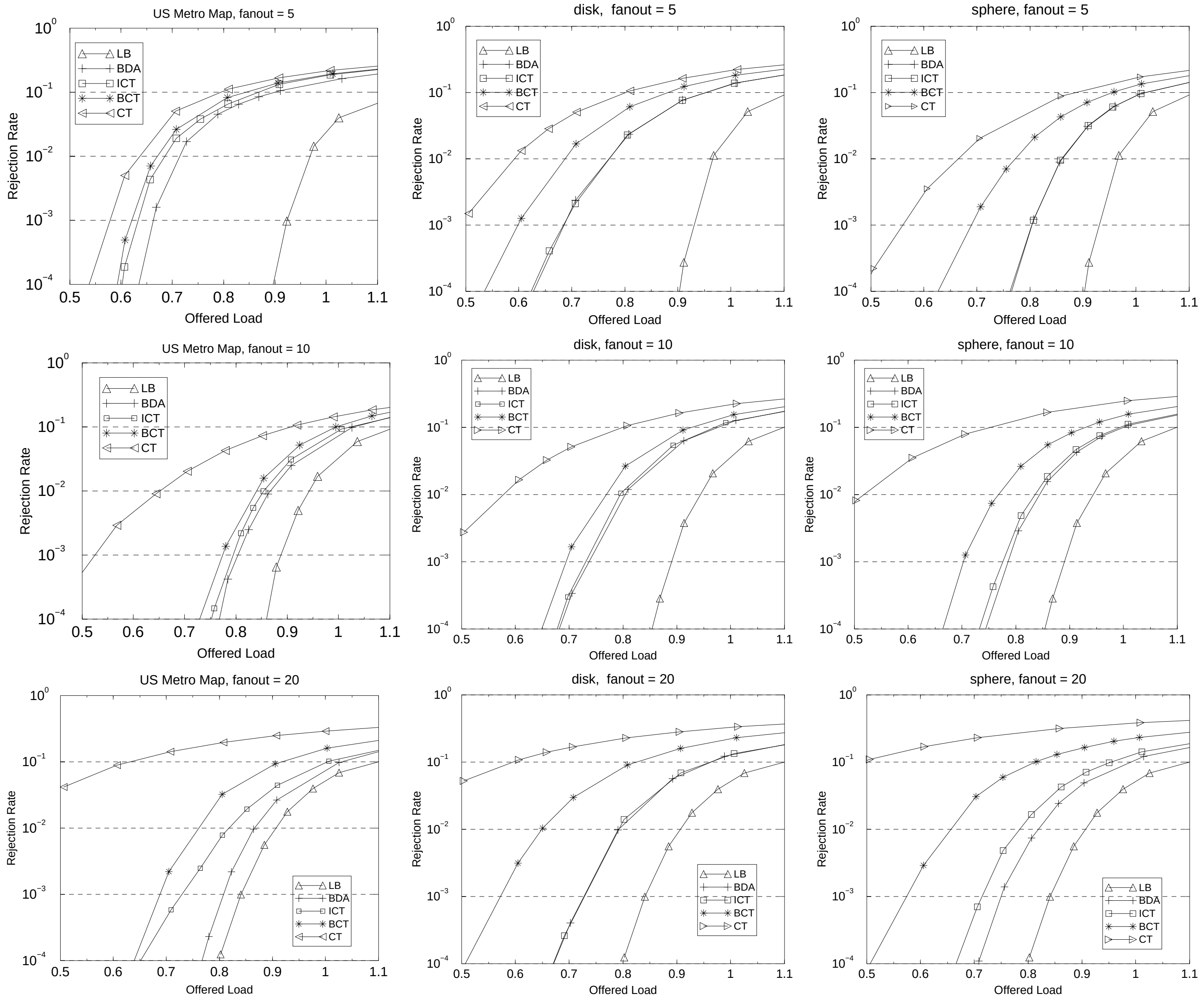


Fig. 6. Rejection Fraction Comparison

for the metro configuration, the average number is 200.

Overall, the results show that the algorithms that seek to balance the residual degree usually perform much better than the CT algorithm, which merely seeks to minimize the diameter subject to a constraint on the maximum degree bound of a node ($d_{max}(v)$). The performance of CT is particularly poor in the fanout 20 case, since it tends to create nodes with large fanout leading to highly unbalanced residual degree distributions.

Looking down each column, we see that the lower bound increases with the fanout. This simply reflects the fact that each session consumes a larger fraction of the total interface bandwidth. For the sphere, the rejection fraction also increases with fanout for the BDA curve. This makes sense intuitively, since as the fanout grows, one expects it to be more difficult to find balanced trees with small enough diameter. In the disk and metro configurations, it is less clear why the BDA curve changes with fanout as it does. Part of the explanation for the observed behavior is that the network dimensioning process is based on an assumed traffic load, and in particular, an assumed multicast session fanout of 10. When the simulated traffic has the same fanout distribution as the one used to dimension the network, we get smaller rejection rates. However, there is a somewhat surprising deterioration of the rejection rate for small fanout, particularly in the metro network case. The apparent explanation for this is that with small fanout, we often get sessions involving MSNs near the east and west coasts, but none in the center of the country. Such sessions are unable to exploit the ample unused bandwidth designed into the more central MSNs, based on a larger average fanout. A similar effect is observed with the disk, but it is more extreme in the metro configuration because of the greater population densities on the coasts, and also the smaller ratio of the diameter bound to the maximum inter-MSN distance (1.5 vs. 2).

The curves for the ICT algorithm are generally quite close to the BDA curves, leaving little apparent room for improvement. There are small but noticeable gaps in a few cases. For the metro configuration, the gaps for fanout 5 and 20 are most probably due to the difference between the average fanout of the simulated traffic and the fanout used for dimensioning. This explanation does not account for the gap in the sphere configuration for fanout 20, since in the sphere, all nodes have the same interface bandwidth. The most plausible explanation seems to be that with large fanout, it just becomes intrinsically more difficult to find trees that satisfy the diameter bound.

In three out of the nine cases shown, the BCT algorithm performs nearly as well as the ICT algorithm. Its performance relative to ICT is worst for the sphere and best for the metro configuration.

D. Performance Results – Tree Diameter

Next, we investigate the performance of these algorithms in terms of the average diameter the trees created. Figure 7 shows the cumulative distribution of the tree diameter scaled to the diameter bound used in the algorithms. The fanout used here is 10 per session. Also, we show in a table the mean and variance of the tree diameter using unscaled values.

We observe that the diameter performance of the BCT algorithm is as good as that of the CT algorithm; the difference is almost indiscernible. Since the ICT algorithm makes no attempt to minimize the diameter (it simply attempts to meet the diameter bound), it does not perform as well as the other algorithms, with respect to diameter. Especially for the sphere, ICT generates trees with diameter significantly larger than those generated by the other two algorithms (the median diameter is approximately 20% larger). The explanation seems to be that in the sphere, degree allocation and traffic is evenly distributed, which means that BDA tends to produce trees in which all vertices have small degree, and many have degree 2. Such trees, while meeting the diameter bound, have significantly larger diameter than the trees produced by algorithms that explicitly seek to minimize diameter.

These results suggest that there may be further room for improvement. It might be possible to develop algorithms that match the low rejection rates achievable with balanced degree allocation, while also matching the diameter performance of CT. In some settings, it may makes sense to adopt a hybrid approach in which BCT is used for application sessions that are highly delay sensitive, while ICT is used for others.

E. Complexity of the ICT Algorithm

Although the ICT algorithms gives superior performance in terms of rejection fraction and overall system utilization, and can satisfy the diameter bound in most cases, its computational complexity is a potential disadvantage, particularly when the diameter bounds are tight. In this case, the iterative loosening of the degree allocation might continue for many rounds.

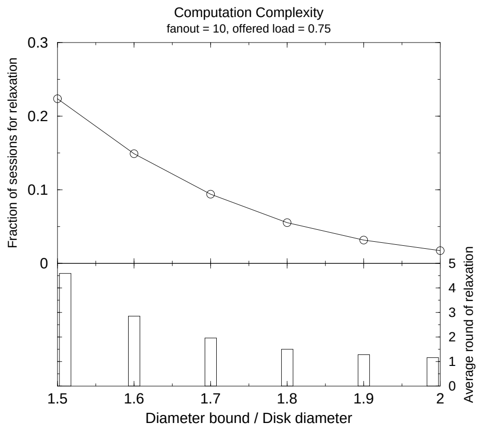
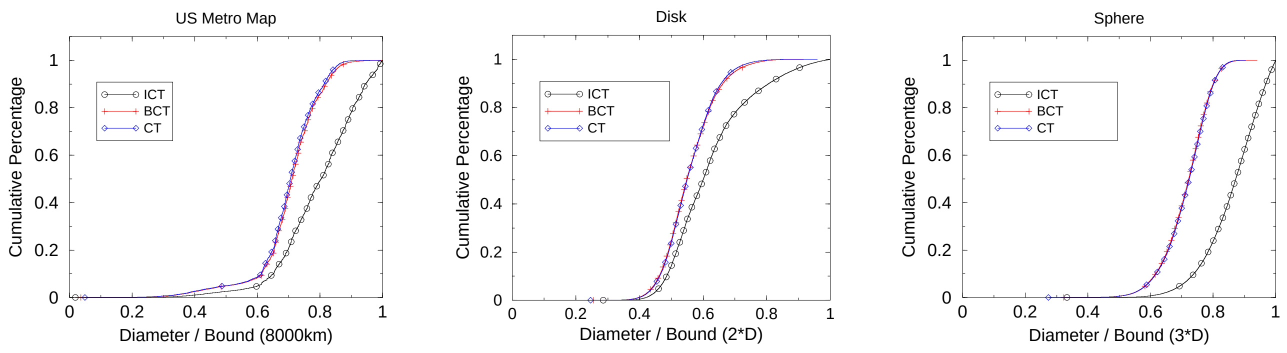


Fig. 8. Evaluation of Iterative Loosening in ICT Algorithm

Figure 8 evaluates the impact of the iterative loosening on the performance of the ICT algorithm. The top half of the figure shows the percentage of sessions that require degree adjustment; and the bottom half shows the average number of rounds iterated for the session that do require it. We observe that for degree bounds of at least 1.7 times the diameter, fewer than 10% require degree adjustment. Among those, that do require adjustment, the average number of rounds is 2 or less. However, if the diameter bound is too stringent, the extra rounds of degree adjustment do not help much in reducing the session rejection rate. Therefore, it is important to pick a suitable diameter bound for a topology so that the algorithm can operate more efficiently and more effectively. From our experience, a bound of twice the maximum distance between any pair of nodes seem to be a good choice in all three network configurations.

V. Discussion

In this study, we have not considered dynamic session memberships. When vertices can join or leave a session dynamically, there are fewer choices available to the session routing algorithms. While the strategies of degree



	metro		disk		sphere	
	mean	std.	mean	std.	mean	std.
CT	5581.22	14.4%	222.38	13.9%	429.16	10.0%
BCT	5627.77	14.5%	222.67	14.5%	428.17	10.1%
ICT	6347.36	15.7%	248.17	19.9%	516.18	10.4%

Fig. 7. End-to-end Delay performance

balancing and diameter minimization can still be applied, the need to apply them incrementally can be expected to degrade performance. An important qualitative difference between the static and dynamic problems is that in the dynamic problem, nodes may have to remain in a session, even when they are no longer active participants. This may be required to prevent a disruption in the flow of packets to other nodes participating in the session. If we allow only incremental changes to a session configuration (disallowing more global rearrangement), then a node which is no longer an active participant may drop out of the session only if it is a leaf in the tree. Alternatively, we might allow a localized rearrangement when a node's degree drops to 2. In a case like this, it would be relatively straightforward to ensure a continuing smooth flow of packets to the participants, while the rearrangement is taking place.

The issue of implementing the algorithms in a distributed fashion also needs to be addressed. If implemented in a fully distributed fashion, the proposed algorithms require synchronized update at the end of each node addition to the tree, which is potentially inefficient and unscalable. Alternatively, if they are implemented in a centralized way, i.e. let one of the MSNs compute the routes and inform others to connect as a tree, we can eliminate the many message exchanges required to coordinate a distributed computation. We should point out that the centralized version does not create a single point of failure or even a performance bottleneck, as each session may select a different delegate to perform the tree computation. During periods of heavy network load, we can expect there to be lots of session routing computations being performed concurrently. This means that the overall computational load can be effectively distributed by having different servers do the computation for different sessions. We believe that the greater efficiency of this approach, relative to a distributed routing computation for each session, will more than compensate for any inequities in the load distribution that are likely to arise in practice. However, we note that there may be some benefit to be gained by introducing explicit load balancing mechanisms.

VI. Related Work

The Multicast Backbone (Mbone) [5], is the best known and widely adopted multicast overlay network. The Mbone is implemented as tunnels at the network layer and implements the distance vector multicast routing protocol [2]. Other standard routing protocols include: core-based tree (CBT) [3], protocol independent multicast (PIM) [6] and most recently, source specific multicast (SSM) [8]. All of these routing protocols build shortest path trees from data sources or from the core node of a session, to minimize network delay (although not necessarily, the total traffic in the network).

There are many application-level multicast services appearing in the recent literatures, mostly due to the dwindling usage of the Mbone and the slow deployment of network multicast services. The flexibility of application-level multicast services allow the routing policy to be changed based on the target application requirements. For example, Scattercast [4] uses delay as the routing cost and builds shortest path trees from data sources; Overcast [11] explicitly measures available bandwidth on an end-to-end path and builds a multicast tree that maximizes the available bandwidth from the source to the receivers;

and Endsystem multicast [9] uses a combination of delay and available bandwidth, and prioritizes available bandwidth over delay when selecting a routing path.

In this paper, we have defined *interface bandwidth* as our primary routing metric. The path selection policies seek to optimize the usage of interface bandwidth of MSNs while satisfying the end-to-end delay performance of individual session. We believe ours is the first work to focus on optimizing the MSN interface bandwidth.

VII. Conclusions

In this paper, we have introduced several multicast routing algorithms that are specifically designed for overlay networks, where the optimization of the interface bandwidth at multicast service nodes is a primary focus. This leads to rather different routing considerations than in conventional networks. Our algorithms seek to balance the available MSN interface bandwidth while keeping the tree diameter small. Our evaluation showed that it is possible to achieve a large gain in system utilization without a significant reduction in the end-to-end delay performance. The algorithms perform well across a range of network configurations and traffic conditions.

Perhaps the most promising direction for future work on overlay multicast routing relates to the dynamic version of the problem. Another direction worth pursuing is the development o routing algorithms that provide fault tolerance in the presence of server failures.

References

[1] Akamai Technologies, Inc. http://www.akamai.com.

[2] F. Baker. Distance Vector Multicast Routing Protocol - DVMRP. RFC 1812, June 1995.

[3] A. Ballardie. Core Based Trees (CBT version 2) Multicast Routing - Protocol Specification. RFC 2189, September 1997.

[4] Y. Chawathe, S. McCanne, and E. Brewer. An Architecture for Internet Content Distribution as an Infrastructure Service. Unpublished, available at http://www.cs.berkeley.edu/ yatin/papers.

[5] H. Erikson. MBONE: The Multicast Backbone. *Communication of the ACM*, pages 54–60, August 1994.

[6] D. Estrin, V. Jacobson, D. Farinacci, L. Wei, S. Deering, M. Handley, D. Thaler, C. Liu, S. P., and A. Helmy. Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture. Internet Engineering Task Force, August 1998.

[7] P. Francis. Yallcast: Extending the Internet Multicast Architecture. http://www.yallcast.com, September 1999.

[8] H. Holbrook and B. Cain. Source Specific Multicast. IETF draft, draft-holbrook-ssm-00.txt, March 2000.

[9] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proc. ACM SIGCOMM 2001*, San Diago, CA, August 2001.

[10] iBeam Broadcasting Corp. http://www.ibeam.com.

[11] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. OSDI'01*, 2000.

[12] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An **A**pplication **L**evel **M**ulticast **I**nfrastructure. In *3rd USENIX Symposium on Internet Technologies and Systems (USITS'01)*, San Francisco, CA, March 2001.

[13] S. Shi, J. Turner, and M. Waldvogel. Dimension Server Access Bandwidth and Multicast Routing in Overlay Networks. In *11th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, June 2001.

[14] U.S. Census Bureau. http://www.census.gov/population/www/estimates/metropop.html.

[15] S. Zhuang, B. Zhao, A. D. Joseph, R. H. Katz, and J. Kubiatowicz. Bayeux: An Architecture for Wide-Area, Fault-Tolerant Data Dissemination. In *Proc. NOSSDAV'01*, June 2001.