# Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks

Sherlia Y. Shi Jonathan S. Turner Marcel Waldvogel
Applied Research Lab
Washington University
St. Louis, MO 63130
{sherlia, jst, mwa}@arl.wustl.edu

## ABSTRACT

Application-level multicast is a new mechanism for enabling multicast in the Internet. Driven by the fast growth of network audio/video streams, application-level multicast has become increasingly important for its efficiency of data delivery and its ability of providing value-added services to satisfy application specific requirements. From a network design perspective, application-level multicast differs drastically from traditional IP multicast in its network cost model and routing strategies. We present these differences and formulate them as a network design problem consisting of two parts: one is bandwidth assignment in the overlay network, the other is load-balancing multicast routing with delay constraints. We use analytical methods and simulations to show that our design solution is a valid and cost-effective approach. Simulation results show that we are able to achieve network utilization within 10% of the best possible utilization while keeping the session rejection rate low.

## Keywords

application-level multicast, network planning, load balancing routing

## 1. INTRODUCTION

IP multicast and its various companion problems such as reliable transport and multicast security, have been hot research topics in recent years. Although many innovative approaches have been developed, the deployment of IP multicast in the Internet has not been easy. In fact, except for the Mbone [6], there is no global multicast infrastructure available. The most cited problems preventing ISPs from deploying a multicast-enabled network include: the complexity of most multicast routing protocols and their implementations; the lack of a scalable inter-domain routing protocol; and the lack of support in access control and transport services.

Despite these difficulties, it is undeniable that multicast is an efficient transmission mechanism to reduce network load for very large groups and save transmission time and bandwidth for data sources even in small multicast groups. Recently, research efforts have emerged in two areas: one is to simplify the IP multicast model to enable very large scale single source multicast [12, 13]; and the other is to build an overlay multicast tree among session participants (end-systems or proxy servers) and unicast data along tree links. We categorize the latter as *application level multicast*[1] [1, 2, 7, 14].

Besides the push from content distribution networks, application level multicast also suits other multicast applications such as video conferencing, data replication services, etc. These applications typically have many-to-many semantics and/or interactive sessions, and session entities can vary widely in their processing power and network connectivity. In [16], we proposed *AMcast* as a common service layer to facilitate multipoint applications without the need of native network support. The key idea in AMcast is to deploy application servers to aggregate datagrams from end users and tunnel packets to servers in other domains. Logically, servers create a virtual multicast tree among all server participants of a session, and spawn a star topology from each server to its end users. The advantage of such an architecture is that end users send or receive exactly one copy of all packets disseminated over the session, and the work of duplicating packets is shifted from data sources to all session servers. Given the fast development of optical communication infrastructure, the capacity of backbone or core networks are progressing much faster than interconnections from home-users or business corporations to their ISPs. Meanwhile, server clusters, such as web servers, caching servers and others are also speeding up their network connections into gigabit ranges, thereby creating incentives for our AMcast model.

In this paper, we focus on the network design aspects of multicast overlay networks. In AMcast, bandwidth in the backbone network is assumed to be plentiful, while the limitations on server processing power and network interface bandwidth are system bottlenecks because servers behave as application-level routers, which have to both forward and duplicate packets. For the purpose of this study, we also assume that these servers have enough CPU power for process-

*NOSSDAV'01* June 25-27, 2001, Port Jefferson, New York, USA.
Copyright 2001 ACM 1-58113-370-7/01/0006 ...$5.00.

[1] Although data relay does not necessarily happen purely at the application layer, we use it as a general term referring to self-organizing multicast overlay network

ing packets and saturating their interfaces at link rate, but they may not have enough access bandwidth when traffic load is heavy. This assumption is typically true for most of today's server clusters which negotiate with ISPs for service level agreements at certain prices.

The design process described in this paper includes two components: one is to quantify traffic load at servers according to a session traffic model and assign proper access bandwidth to each server site. We refer to this step as the *dimensioning process*; the other component is to devise multicast routing algorithms that make the best use of the above dimensioned network, subject to routing constraints such as end-to-end delay bounds. We show that by closely combining the dimensioning process with a load balancing routing algorithm, we can achieve high overall network utilization of within 10% of the theoretical lower bound with low session rejection rate, i.e. only a small fraction of sessions fail to satisfy the delay constraint.

The rest of the paper is organized as follows: in section 2, we describe our design objectives and steps taken in our design approaches; in sections 3 and 4, we formalize the problems and present our algorithms for multicast routing and bandwidth dimensioning, respectively. In section 5, we use simulation to evaluate these algorithms; in section 6, we compare our work to other related works and then conclude in section 7.

## 2. DESIGN OBJECTIVES AND APPROACHES

Overlay multicast networks differ from traditional networks in several ways, leading to differences in how they are best configured and operated. These main differences are:

- **Network reachability:** The overlay network among end points in application-level multicast is a fully meshed network, as each node is able to reach everybody else in the network via unicast connections. Therefore, unlike in IP multicast where a path from one router to another is defined by its physical connectivity, an n-node application multicast session could have $n^{n-2}$ different spanning trees.

- **Network cost:** Historically, the cost of a network is determined largely by the summation of individual link costs. This is certainly true for network providers who have to physically deploy the links or lease them from others. But from an application or application server's point of view, network cost is actually the total amount paid to gain *access bandwidth* at each service provider's site to the backbone network. This divergence of cost metric has a deep impact on both design and routing strategies.

- **Routing constraints:** Traditional IP multicast routes through a shortest path tree to minimize average delay from source to members, i.e. reducing the number of links needed to carry session traffic. Building the network at the application layer gives the flexibility of matching routing strategies to application needs. For applications such as streaming media or conferencing, a routing strategy that produces bounded delay between any pair of participants results in significantly higher quality from an application's perspective.

This paper addresses two problems. First, given a server-network topology and a set of traffic assumptions, we want to find an assignment of network access bandwidth to each server subject to a fixed overall bandwidth constraint. Second, given the above dimensioned server network, we want to devise a routing algorithm that dynamically routes multicast sessions and makes the best use of the available network resources so as to accommodate the maximum number of sessions as well as satisfying the application delay constraint. The two problems interact. The dimensioning process must know the intrinsic property of a routing algorithm such as the possible traffic concentration points, and assign bandwidth to servers accordingly. On the other hand, the performance of a routing algorithm is significantly affected by the difference between the bandwidth assignment in the underlying network and the actual traffic load.

It is reasonable to question the network efficiency of our approach vs. traditional IP multicast. It is obvious that by tunneling multicast packets through unicast connections, there are duplicated packets on physical links, notably from a server's local interface to the branching point of two unicast connections in the network. This discrepancy in efficiency is potentially significant if the size of multicast session is very large [3]. However, for AMcast virtual network, we envision the number of server clusters is within the range of tens or hundreds to at most lower thousands world-wide, with each cluster consisting of a large number of processing units. In [14], we have quantified through simulation the efficiency ratio of virtual overlay multicast trees vs. IP multicast trees. In a 6000 node network, the cost of a virtual multicast tree on 50 randomly distributed nodes, is within 1.5 times the cost of a IP multicast tree, where the cost is measured as the number of links traversed by each packet. While it would certainly be more efficient to provide multicast as a native IP service, in the absence of a widely deployed IP multicast service, the overlay approach can be useful.

## 3. MULTICAST ROUTING ALGORITHMS

In this section, we present two multicast routing algorithms for the AMcast overlay network. There are two main performance objectives for the routing algorithms. First, they should use network resources efficiently, in order to carry as much traffic as possible; Second, they should keep the end-to-end delay as low as possible, i.e. keeping the tree diameter small. Unfortunately, these two objectives are orthogonal: a small diameter tree creates traffic concentration on nodes that are at the center of the topology, and consequently these nodes become bottlenecks of the system; on the other hand, increasing overall utilization typically means to distribute load more evenly across servers which results in longer path and longer delay. Although it is impossible to optimize both parameters at the same time, we can instead fix a target bound for one objective while optimizing on the other. This leads us to design two alternative routing algorithms.

### 3.1 Algorithm for Delay Optimization

We first formulate the routing problem to minimize the end-to-end delay of a multicast tree while satisfying each server's access bandwidth constraint. Each link in the multicast tree is assumed to require some specific amount of bandwidth $b$, so if a server has degree $d$ in the multicast tree, it will require at least $d * b$ units of access bandwidth. This leads to the following problem formulation.

Definition 1. **Minimum diameter, degree-bounded spanning tree (MDDBST)**
*Given an undirected complete graph $G = (V, E)$, a degree bound $d_{max}(v) \in N$ for each vertex $v \in V$; a cost $c(e) \in Z^+$ for each edge $e \in E$. Find a spanning tree $T$ of $G$ such that for each $v \in T$, degree of $v$ satisfies $d_T(v) \leq d_{max}(v)$ and the diameter of $T$ $dia(T)$, which is the cost of the longest simple path in $T$, is minimized.*

Much previous research has been done on related problems. In [11], Ho et al. proved that in geometric space, there exists a minimum diameter spanning tree in which there are at most two interior points (non-leaf nodes) and the optimal tree can be found in $O(n^3)$ time. Hassin and Tamir established in [10], that for a general graph, a minimum diameter spanning tree problem is identical to the absolute 1-center problem introduced by Hakimi [9] and as such, a solution can be found in $O(mn+n^2 logn)$, where $n$ is the number of nodes and $m$ the number of edges. In [11] and [15] respectively, they prove that minimum diameter, minimum spanning tree and minimum maximum degree, minimum spanning tree are both NP-complete.

Theorem 1. *The decision version of* MDDBST*– finding a spanning tree with diameter bound $B$ and a degree constraint $d_{max}(v)$ for each node, is NP-complete, for $2 \leq d_{max}(v) < |V| - 1$.*

*Proof:* Clearly, the problem is in NP, since we can verify in polynomial time if a candidate solution satisfies both the diameter and degree constraints. For the special case where $d_{max}(v) = 2$ for all $v \in V$, the problem is the same as the *Traveling Salesman Problem(TSP)* [8]. We reduce from the TSP problem for the general case of $d_{max}(v) \geq 2$. Let $G = (V, E)$ be the graph of a TSP instance. We transform $G$ to $G' = (V', E')$ by adding $d_{max}(v) - 2$ vertices $u_1, \ldots, u_{d_{max}(v)-2}$ to each $v \in V$. We join each of these new vertices $u_i$ to $v$ with an edge length of 0; All other edges from $u_i$ have length $B + 1$, so that $G'$ is still a complete graph. Now, the MDDBST instance in $G'$ has a spanning tree of diameter $B$ if and only if the TSP instance in $G$ has a path joining all the vertices of length $B$. □

### 3.1.1 Heuristic Algorithm for MDDBST

We have developed a heuristic algorithm for the MDDBST problem, which is a greedy algorithm similar to Prim's algorithm for *Minimum Spanning Tree* [4]. Figure 1 shows the steps of the algorithm. We denote $\delta(v)$ as the longest path of $v$ to any other nodes in $T$. Similarly to Prim's algorithm, we start from a single root node. At each step when adding a new node $u$ to the existing component $T$, we select the node that has the smallest $\delta(u)$. Then, we update the nodes in the existing component that have changed their longest path because of the new node, $\delta(v) = max(\delta(v), dist_T(u, v))$. Finally, for each node $v$ not in the component, we update its parent to node $q$ which, without violating the degree constraint, gives $v$ the smallest longest path.

The algorithm fails when it finishes with some vertices having $\delta(v) = \infty$, meaning that we cannot build a spanning tree with the specified set of degree constraints. There are two occasions for this to happen: one is that the total degree constraints can be less than $2 * (|V| - 1)$, which is the minimum total degree required for a session spanning tree; the other is that during the progress of the algorithm, a leaf node maybe added to the current tree component and consumes all the spare degrees of the component, leaving the rest of the nodes disconnected. Both of these failures do not occur very often in a real system, as the degree constraints for each session are usually generous enough to avoid them. Only when the system is extremely highly loaded, some or all of the nodes may have stringent degree constraints which cause the algorithm to fail. We can perform a simple feasibility test on the summation of the degree constraints to identify the first type of failure. To remedy the second type of failure, we can add a count of the spare degree of the tree component and defer the addition of a leaf node if it reduces the count to zero.

```
Input:
  G = (V, E)
  Edge cost c(u,v), for $u, v \in V$
  Degree constraints $d_{max}(v)$
Output:  T with the smallest diameter
  foreach $r \in V$
    foreach $v \in V$
      $\delta(v) = c(r, v)$;
      $p(v) = r$;
    T = (W={r}, L={});
    while $(W \neq V)$
      let $u \in V - W$ be the vertex with smallest $\delta(u)$;
      $W = W \bigcup \{u\}; L = L \bigcup \{\{u, p(u)\}\}$;
      foreach $v \in W - \{u\}$
        $\delta(v) = max\{\delta(v), dist_T(u, v)\}$;
      foreach $v \in V - W$
        $\delta(v) = \infty$;
        foreach $q \in W$
          if $degree(q) < d_{max}(q)$ and $c(v, q) + \delta(q) < \delta(v)$
            $\delta(v) = c(v, q) + \delta(q)$;
            $p(v) = q$;
```

**Figure 1: Heuristic Algorithm for MDDBST**

During the updating phase, it requires $O(n)$ time to update the new longest path for each $v \in W$; and $O(n^2)$ time to find the new parent for each $v \notin W$. And the total running time of the greedy algorithm is $O(n^3)$. We analyze its performance ratio in terms of the tree diameter. Let $\lambda_{Greedy}$ and $\lambda^*$ denote the tree diameter constructed by the greedy algorithm and an optimal solution, respectively. And let $d^{min} = min(d_{max}(u))$ and $d^{max} = max(d_{max}(u))$.

Lemma 1. *If the ratio of edge weights is bounded by $\varepsilon \in Z^+$, and the degree constraints satisfy $2 < d^{min} \leq d^{max} < d$ for a constant $d < |V| - 1$, then $\lambda_{Greedy} < O(k)\lambda^*$, where $k = \varepsilon \log_{d^{min}} d^{max}$.*

*Proof:* Without loss of generality, let the smallest edge weight be 1 and the largest edge weight $\varepsilon$. The optimal solution achieves $\lambda^* > 2 \log_{d^{max}} n$. Now, let's assume a simple algorithm $A$ for constructing a spanning tree: at each step of adding a new node, $A$ simply selects a node to maximize the degree of $u \in T$ without consideration of the tree diameter. In the worst case, $\lambda_A < 2\varepsilon(\log_{d^{min}} n)$. We observe that $\lambda_{Greedy} \leq \lambda_A$, since when adding a new node, the greedy algorithm always attempts to select the one node which will result in the smallest diameter increase. Therefore, $\lambda_{Greedy} < O(k)\lambda^*$, where $k = 2\varepsilon \log_{d^{min}} d^{max}$. □

We further evaluate the MDDBST algorithm through simulation in section 5. There, we used a topology close to a real network. We observe that the MDDBST algorithm is capable of creating multicast tree with small delay but is lack of the ability to distribute traffic load across servers. Consequently, the utilization of the system is low and the session rejection rate is high. A session request is rejected if it arrives at a server which does not have any spare bandwidth. This suggests that if we can distribute the work on heavily loaded servers to others who are nearby but are less loaded, we can prevent the bottleneck servers from "choking" and reduce the session rejection rate. We introduce a load-balancing routing algorithm that utilizes this idea.

### 3.2 Load Balancing Routing Strategy

DEFINITION 2. **Bounded diameter, residual-balanced spanning tree (BDRBST)**
*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 $e \in E$; a bound $B \in Z^+$. Find a spanning tree $T$ of $G$ that $d_T(v) \leq d_{max}(v)$, for each $v \in V$; diameter of $T$ $dia(T) < B$ and maximize $min(d_{max}(v) - d_T(v))$.*

The above problem is also NP-complete, since its special case when every node has a degree of two, corresponds to the decision version of the TSP problem.

The residual bandwidth of a server is $d_{max}(v) - d_T(v)$. By maximizing the minimum of the residual bandwidth, we give the bottleneck server a better chance to serve other sessions if requested. Overall, this increases the total load that the system can handle at the cost of increased end-to-end delay. In order to distribute load while still satisfying the constraint of end-to-end delay, we introduce a *balance factor* $M$ to denote the tradeoff between diameter and load balancing. We vary the previous MDDBST algorithm to take into account this balance factor: at each step when adding a new node, instead of selecting the one node that has the smallest $\delta(v)$, we select a set of $M$ smallest nodes and choose one of them that maximizes the minimal residual bandwidth of these M nodes and their parent nodes. If $M = 1$, this algorithm is the same as the one shown in Figure 1. On the other hand, if $M$ equals the number of servers in a multicast session, then the algorithm considers load balancing as the sole routing criteria and serves as an approximation algorithm for BDRBST. For intermediate values of $M$, it takes both parameters into account. We have found that small values of $M$ (e.g. 5) provide good load balance while still meeting the diameter bound.

We have also considered a slightly different version of the BDRBST algorithm which maximizes the proportional residual bandwidth at each server. We define it as follows.

DEFINITION 3. **Bounded diameter, residual fraction-balanced spanning tree (BDRFBST)**
*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 $e \in E$; a bound $B \in Z^+$. Find a spanning tree $T$ of $G$ that $d_T(v) \leq d_{max}(v)$, for each $v \in V$; diameter of $T$ $dia(T) < B$ and maximize $min(\frac{d_{max}(v) - d_T(v)}{d_{max}(v)})$.*

However, in most of the simulations, BDRFBST consistently performs worse than BDRBST, as shown in Figure 11. Therefore, we will not discuss it any further in the rest of the simulations.

## 4. DIMENSIONING SERVER ACCESS BANDWIDTH

In this section, we describe the *dimensioning process*. The main objective is that given a topology and a traffic model, we want to assign access bandwidth to individual servers (subject to a fixed total), such that a specific routing algorithm will best utilize the allocated bandwidth. In other words, the dimensioning process creates the degree constraints, as described earlier in the routing algorithms, for individual servers.

There are two important parameters involved in the process: a) the location, or the topology of AMcast servers; and b) the traffic model used for bandwidth dimensioning. For the purpose of this study, we have made statistical assumptions on these parameters since closely modeling them as they are in the real networks is a separate and complex issue beyond the scope of this paper.

### 4.1 Dimensioning to Minimize Delay

When there is no degree constraints on servers, the quality of a multicast tree is characterized by its diameter, which is to provide the best possible multicast tree for applications. Therefore, we construct the multicast tree as a *minimum-diameter spanning tree.*

Figure 2 shows an example of a dimensioned server network. The topology is derived from a geographic map of 50 largest metropolitan areas in US [17] and link delay is calculated as the geographical distances between cities. We model session requests as a Poisson process and session fanout as a Binomial distribution with mean equal to 10. We compute the average traffic load of each server during the whole simulation time. The total network bandwidth capacity is 10,000 units, which is allocated to individual servers proportional to its traffic load. There are two factors impact the traffic load distribution across the servers. One is the number of sessions that a server will participate in (a server participates in a session if one of the users in its service area is a participant); the other is the location of a server which decides the degree of a server in a multicast tree. A geocentric server typically has a larger degree since it acts as a transit hop for nodes on both sides. Therefore, the traffic load on servers in larger metro areas and servers in the center of the topology is expected to be heavy. Figure 2 illustrates this effect.

The servers are ordered on x-axis by their population. The overall trend is that servers in larger areas receive higher bandwidth, while servers at the center of the US continent also receive significantly more bandwidth. For example, Chicago has about 43% of New York's population but receives 1.3 times more bandwidth.

### 4.2 Dimensioning for Specific Routing Algorithms

For a specific routing algorithm to perform well, we need to allocate bandwidth as close as possible to how the routing algorithm will use them under the presumptive traffic load. The previous delay-based dimensioning has created a configured network, over which we can route the traffic once more using the actual load balancing routing algorithm.

Respectively, let $C_i$ and $C'_i$ be the capacity assigned to each server after one (delay-based dimensioning) and two (routing algorithm specific dimensioning) rounds. We reassign bandwidth capacity $C'_i = L^c_i + \frac{\sum R^c_i}{n}$, where $n$ is the

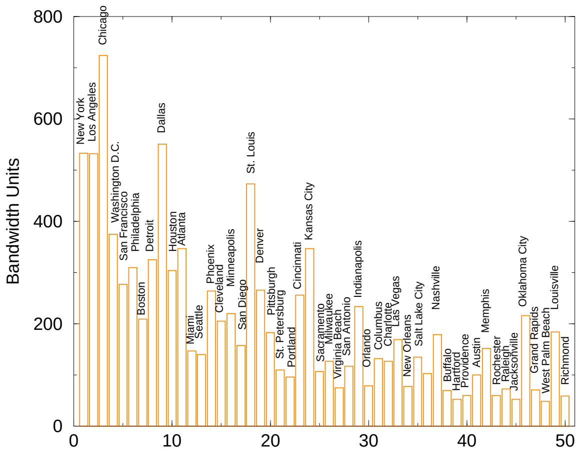


**Figure 2: Dimension of Server Access Bandwidth**

number of servers, $L_i^c$ the carried load of each server during the second round of dimensioning and $R_i^c$ the average residual bandwidth of server $i$ in the second round. Intuitively, the algorithm converges since the load balancing algorithm always tries to equalize the residual bandwidth of each server by adding more transit traffic to servers with available capacities while offloading smaller servers, and the dimensioning algorithm also reduces the excess capacities of big servers and add them to those whose bandwidth are less abundant.

Through simulation, we show that by tightly coupling the dimensioning process to the routing algorithm, we improve on the overall network utilization.

## 5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the two routing algorithm in two aspects: one is the tree diameter, the other is the network utilization. We have selected the 50 most populated metropolitan areas in the United States [17] as the server network topology. Throughout the simulation, we also used geographic distance between cities as edge cost.

### 5.1 Performance on Tree Diameter

Figure 3 shows the simulated multicast tree diameter of MDDBST by restraining a node's degree as a binomial random distribution with mean $p$. In each simulation run, we randomly select over the 50 cities a fixed fanout (or session size), shown as x-axis. The y-axis shows the ratio between multicast tree diameter and the geographical distance of two furthest apart cities in a session, which is also the optimum of end-to-end delay for the session. The results clearly shows that our heuristic algorithm works very well with the largest end-to-end delay no more than 1.25 times the optimum. For smaller degree bound, the spanning tree is more likely to be long legged, thus longer diameter. On the other hand, as the density of the session increases, closeby nodes are connected to each other if its degree bound allows, which results in a shorter diameter.

Figure 4 shows the performance on tree diameter of the BDRBST algorithm. We use the same 50-city network as described previously, and we use a binomial distribution with mean = 7 as degree bound for each node and a diameter bound of 8,000 kilometers (or roughly 40ms one-way delay).

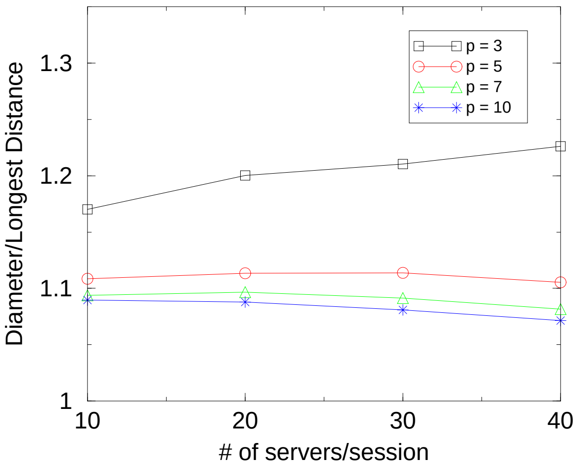


**Figure 3: MDDBST Performance on Tree Diameter**

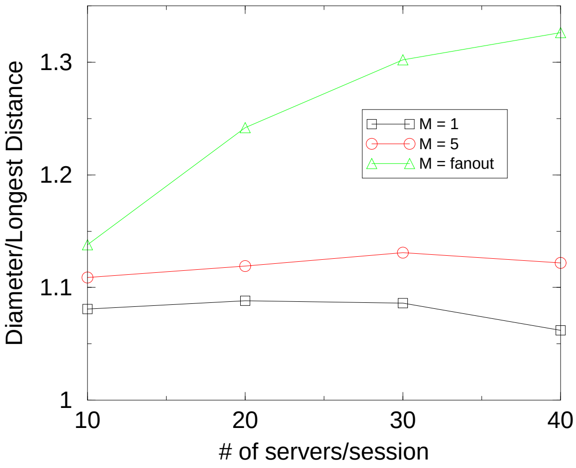


**Figure 4: BDRBST Performance on Tree Diameter**

By varying the balance factor $M$, we illustrate the tradeoff between load-balancing and tree diameter in the BDRBST algorithm. Clearly, we can see that the more we lean towards load balancing, i.e. a larger $M$, the longer the tree diameter. Nevertheless, we are able to construct a spanning tree to meet the diameter bound for all $M$. This is partly due to that 8,000 km is a generous bound enough to cover coast-to-coast propagation delay for the entire nation, yet it is also a reasonable bound from an application stand point of view. Therefore, for the rest of simulation sets, we mainly focus on the load balancing part of the algorithm and merely ensure that the diameter bound is always satisfied.

### 5.2 Performance on Load Balancing

In this simulation, we assume traffic density on a server is proportional to the population of its service area and accordingly, we select the probability of a server participating in a given AMcast session. Although such assumption is quite simplistic and may not reflect the actual traffic pattern, it sets up a traffic model that we can apply consistently in both the dimensioning and routing process. Neither of the two algorithm's operation depends on the traffic model, and we also demonstrate network performance when the projected traffic in network dimensioning differs from the actual routed traffic. We model the dynamic session requests and removals as Poisson session arrivals with Pareto session service time. We generate session fanout as a binomial distribution with mean equals to 10. Each session is assumed to consume one unit bandwidth per tree link.

In order to focus on more important aspects of the simulations, we have fixed several parameters through various simulations, shown in appendix A.

- Value of load balancing factor $M$. We found that with a small $M$, typically 2 or 5, the load balancing achieves full-fledged performance gain, while further increase of $M$ only has diminishing effects. As a small $M$ also has advantage in reducing the complexity of the algorithm, we chose $M$ equal to 5 in following simulations.
- Overall bandwidth capacity. This parameter affects directly the session rejection rate, the larger the total capacity the smaller rejection rate. From Figure 11, we chose to use the mid-range capacity of roughly 200 session bandwidth unit per server for further simulation.

### 5.2.1 *Performance under Different Dimensioning Strategies*

We evaluate AMcast network performance by examine session rejection rate under a dynamic traffic load. The network is configured using the dimensioning algorithm described in section 4. We show the performance of BDRBST algorithm over two differently dimensioned network, one is delay-based dimensioning, the other uses BDRBST-specific dimensioning strategy.

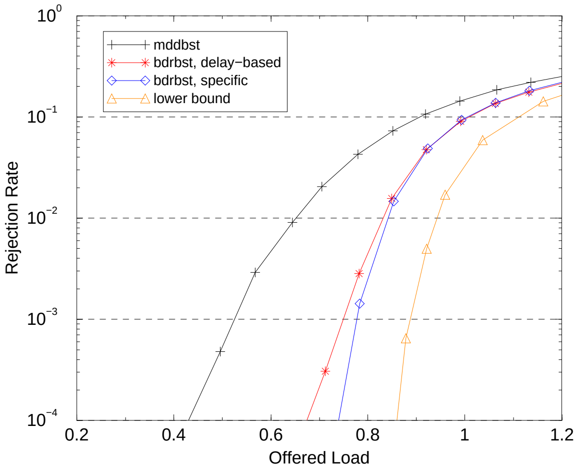


**Figure 5: Performance Comparison of Dimensioning Strategies**

Figure 5 shows the performance of MDDBST and BDRBST algorithms. Additionally, we computed a theoretical lower bound of the whole system. The lower-bound is computed by emulating a single queuing system where the total server bandwidth amounts to the queue size and each session corresponds to a message whose size equals to the session size. Although this lower bound is not very tight, it gives an indication of the best achievable utilization for any routing algorithm.

As described earlier, MDDBST generates the multicast tree without any consideration of server load balancing but only considers the maximum server capacity constraint. Clearly, it does not have any adaptiveness to load variance across servers and performs poorly as expected. If we expect our system to operate at rejection ratio of 1 every 10,000 sessions, we can only carry traffic amounts to 44% of total network capacity using vanilla MDDBST. On the other hand, BDRBST outperforms MDDBST significantly. For the network to operate at the same rejection ratio, BDRBST gives a 50% gain over MDDBST and allows the network to operate at a more reasonable load. This suggests that it is worthwhile to endure a little higher delay in order to achieve a higher system utilization. Additionally, the different bandwidth allocation strategies also indicate that the closer the dimensioning process is tied with the routing algorithm, the better the performance. When dimensioned with BDRBST-specific assignment, the performance of BDRBST algorithm approaches within 10% of the lower bound.

### 5.2.2 *Performance of Handling Traffic Noise*

The real challenge of designing a robust routing algorithm is when the assumed traffic pattern does not agree with the actual load. This is certainly the case here as our assumption of traffic proportional to city population is quite simplistic. In order to examine the network performance under varied conditions, we have added Gaussian noise to each city's population, thus changing their probabilities of participating a multicast session and traffic intensity across the network.

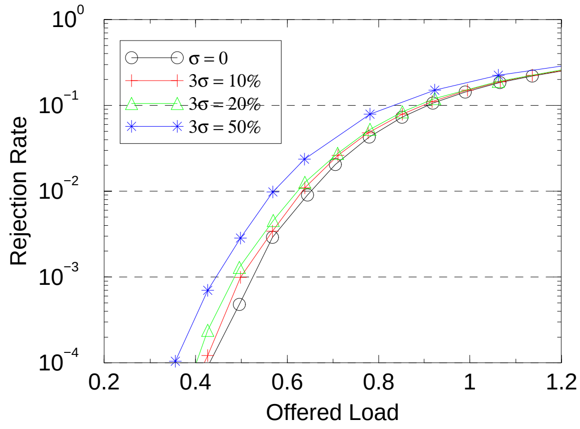


**Figure 6: MDDBST Algorithm Under Traffic Noise**

Figure 6, 7 and 8 depict the performance of the two algorithms when handling traffic noise. For the desired network operation point of $10^{-4}$ rejection ratio, a 50% noise results about 15% performance drop at most for both algorithms. The relative performance across two network configurations is also preserved as traffic noise increases.

## 6. RELATED WORK

*Application-level multicast* is a general term referring to self-organization of overlay multicast network. It is also a very young area with much potentials, both for research and commercial users, and thus has received increasing attention recently. Previously, we have proposed a version of AMcast at end systems called ALMI [14], which employs centralized

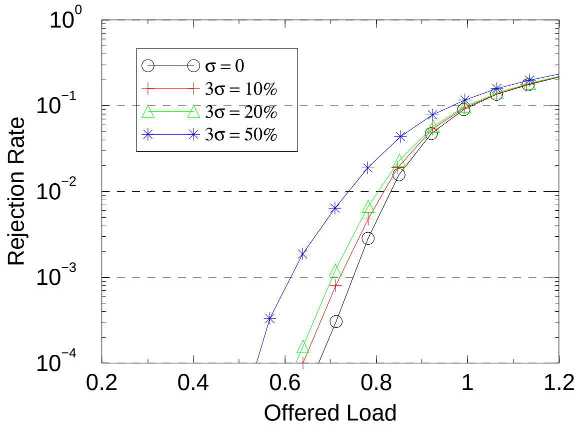


**Figure 7: BDRBST Over Delay-based Dimensioned Network with Traffic Noise**

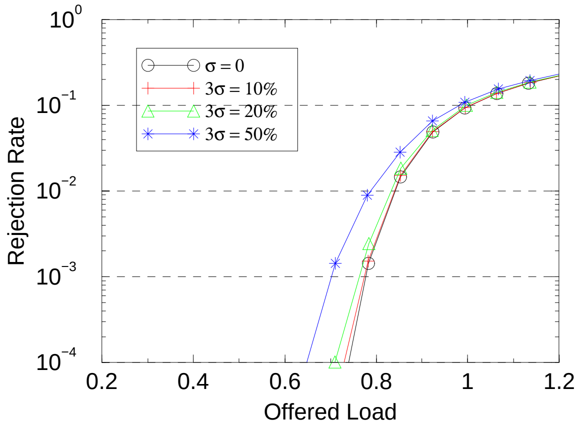


**Figure 8: BDRBST Over Specifically Dimensioned Network with Traffic Noise**

control and focuses more on value added middleware functions. The goal then was to allow small groups create interactive sessions with low to moderate bandwidth. AMcast is a logical extension after that to provide an infrastructure service for the Internet. There are several other overlay multicast architecture in the recent literatures, which we discuss below.

Yallcast [7], aims to extend the Internet multicast architecture and defines a set of protocols for host-based content distribution either through tunneled unicast connections or IP multicast wherever available. It uses a *rendezvous host* to bootstrap group members into the multicast tree. The functionality of the *rendezvous host* is to inform new members about several current members in the tree but the rendezvous host is not connected to the multicast data paths. Yallcast creates a shared multicast tree using a distributed routing protocol. It also maintains a mesh topology among group members to ensure that the multicast group is not partitioned. Overall, Yallcast envisions the deployment of IP multicast within small "network islands" and provides a rudimentary architecture for global multicast.

In contrast to Yallcast, Endsystem Multicast [2] is aiming towards small and sparse group communication applications much like ALMI does. In Endsystem Multicast, group members are self-organized into multicast trees using a routing protocol similar to DVMRP [5] that creates source-based multicast tress. It require members to periodically broadcast refresh messages to keep the multicast tree partition free. A companion protocol of Endsystem Multicast is called Narada, which focuses on optimizing the efficiency of the overlay in terms of delay bounds based on end-to-end measurements.

Scattercast [1] is an application-level infrastructure service engineered for content distribution. It uses shortest path routing to build source-rooted distribution trees. In order to build a routing table at the application level, a mesh is first built among multicast proxies using a protocol called Gossamer for neighbor discovery. Additionally, a customizable transport is defined in Scattercast, which prioritizes application data based on their content, for example text data is prioritized for reliability, while losses in image data are ignored to some extent.

All three of the above schemes try to leverage the existing multicast routing protocols and re-apply them at the application level. Although, at the application level, the complexity of IP routing is greatly reduced, since the number of nodes involved is much fewer than the number of routers all over the Internet, there is additional complexity introduced by the sensitivity of end host measurement and the potential of end host unreliability. Additionally, the cost of building an application-level multicast tree differs greatly from the cost of building a network level multicast tree and results in very different network design and routing perspectives. And it is these aspects that we try to address in this paper.

## 7. CONCLUSION AND FUTURE WORK

In this paper, we have taken a systematic approach for designing an overlay network for application-level multicast. We first defined the unique network cost metric and routing constraints different from conventional networks. According to these performance criteria, we then provisioned the network assuming certain traffic characteristics. Last, we devised routing strategies which take advantage of our dimensioning process to maximize network utilization and to satisfy application service requirement. Through simulation, we evaluated the performance of our routing algorithms under various traffic distributions.

As part of future work, we are looking into other network configurations to further prove the validity of our approaches. Specifically, it is worthwhile to look at more controlled network topology such as geocentric and equal-distance topology as they emphasizes or deemphasizes the importance of geographical location, respectively and can let us further understand the intrinsic characteristics of the routing algorithms.

We are currently underway defining a complete AMcast architecture including schemes for multicast addressing, initialization and interaction between end users and servers. In order to put these algorithms into use, we also need to define a distributed routing protocol and exploit various issues in routing procedures, such as update frequency, forms of information exchanges and etc. Furthermore, one of the advantages of using servers instead of routers is the feasibility and flexibility of providing value-added service for applications. We are looking into issues in providing such a service platform.

# APPENDIX

## A. SIMULATION FOR PARAMETER SELECTIONS IN THE BDRBST ALGORITHM

### A.1 Matched Projected Traffic and Routed Traffic

As a first step to understand the performance of our algorithms, we generates routed traffic exactly the same as we projected. Ideally, this should give a utilization close to the offered load. Figure 9 shows the network utilization for this case. The main parameter we used to cause traffic variation is the session size. In Figure 9, we used a binomial distribution on session size with mean equal to 10 for both projected and routed traffic load.

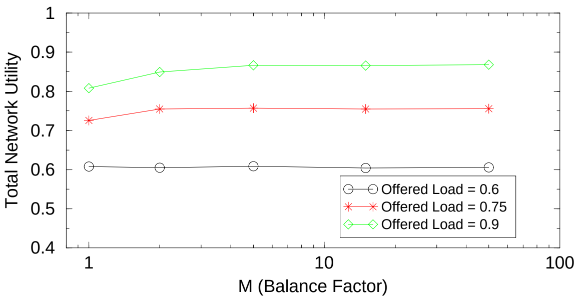


**Figure 9: Network Utilization of Matched Traffic**

When the network is less loaded, all sessions are accepted and a delay-bounded multicast tree can be successfully constructed for each of them. When the offered load increases, the network utilization lags the offered load when no load-balancing (M = 1) is considered. With the increase of the balance factor M, the overall utilization improves.

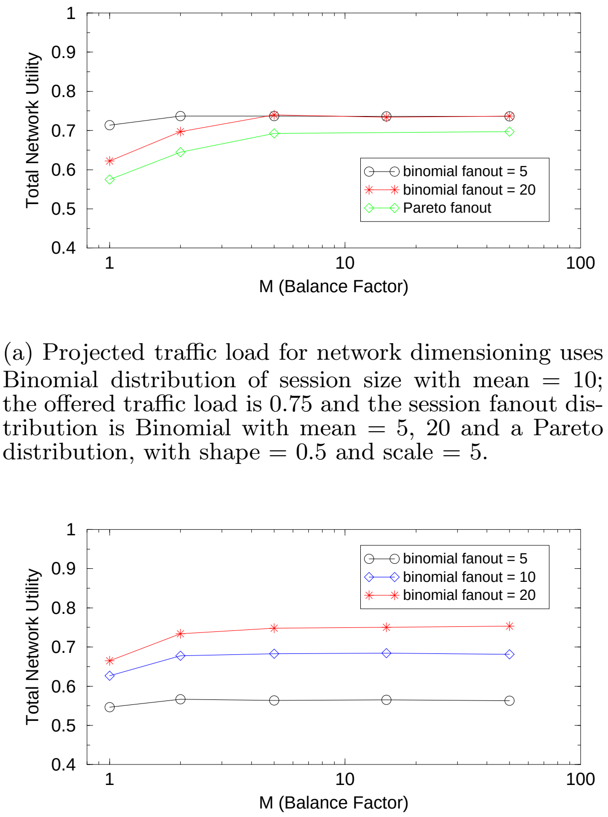
### A.2 Varying Traffic with Session Fanout

In Figure 10, we vary the traffic load by using different distribution on session size. Again, with the increase of the balance factor M, the total network utilization is improved. Especially for small M, the gains of load balancing is already noticeable.

### A.3 Impact of Total Network Capacity

Figure 11 shows session rejection rate vs. offered load. Generally, the higher the overall capacity, the lower the rejection rate. Assuming each session has unit bandwidth, and we dimension network with different capacity ranging from 5,120 bandwidth units to 25,600 bandwidth units. If each session transmits at rate of 10 Mb/s, a total of 25,600 units amounts to 5 Gb/s access bandwidth per server, which can be cost-effectively provisioned using Gigabit Ethernet or corresponding leased lines.

$$\texttt{Offered Load} = \frac{\texttt{Session fanout} * \texttt{Arrival rate}}{\texttt{Service rate} * \texttt{Total bandwidth}}$$



(a) Projected traffic load for network dimensioning uses Binomial distribution of session size with mean = 10; the offered traffic load is 0.75 and the session fanout distribution is Binomial with mean = 5, 20 and a Pareto distribution, with shape = 0.5 and scale = 5.

(b) Projected traffic load for network dimensioning uses a Pareto distribution of session size; the offered traffic load is 0.75, and the session size is Binomial distribution with mean = 5, 10 and 20.

**Figure 10: Network Utilization Under Varied Traffic load**

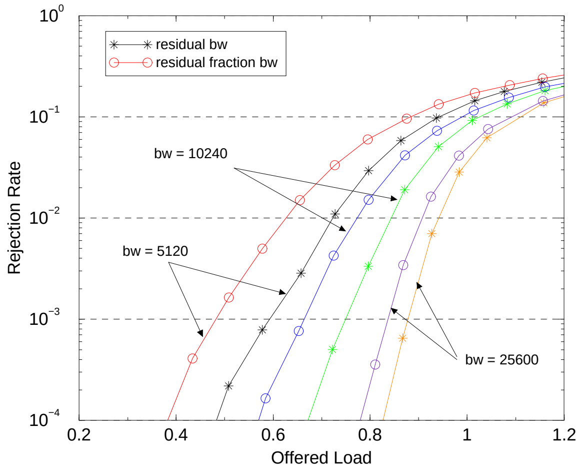


**Figure 11: Dynamic Session Generation and Removal**

## B. REFERENCES

[1] 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.

[2] Y. Chu, S. Rao, and H. Zhang. A Case For EndSystem Multicast. In *Proceedings of ACM Sigmetrics*, Santa Clara, CA, June 2000.

[3] J. Chuang and M. Sirbu. Pricing Multicast Communications: A Cost-based Approach. In *Proc. of INET'98*, 1998.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[5] Distance Vector Multicast Routing Protocol - DVMRP. RFC 1812.

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

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

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness.* San Francisco : W. H. Freeman, 1979.

[9] S. Hakimi. Optimal Locations of Switching Centers and Medians of A Graph. *Operations Research*, 12:450–459, 1964.

[10] R. Hassin and A. Tamir. On the minimum diameter spanning tree problem. *Information Processing Letters*, 53(2):109–111, 1995.

[11] J. Ho, D. Lee, C. Chang, and C. Wong. Minimum Diameter Spanning Trees and Related Problems. *SIAM J.Comput.*, 20(5):987–997, 1991.

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

[13] H. Holbrook and D. Cheriton. IP Multicast Channels: EXPRESS Support for Large-scale Single Source Applications. In *Proc. of ACM SIGCOMM*, Boston, MA, September 1999.

[14] 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*, San Francisco, CA, March 2001.

[15] R. Ravi, M. Marathe, S. Ravi, D. Rosenkrantz, and H. H. III. Many birds with one stone: Multi-objective approximation algorithms. In *25th ACM STOC'93*, 1993.

[16] S. Shi. A Proposal for A Scalable Internet Multicast Architecture. Technical Report WUCS-01-03, Washington Universtiy in St. Louis, March 2001. (Proposal for Dissertation Research in Partial Fulfillment of the Requirements for the Degree of Doctor of Science).

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