Placement of proxy-based multicast overlays

Article *in* Computer Networks · July 2005

DOI: 10.1016/j.comnet.2004.11.019 · Source: DBLP

3 authors, including:

Min-You Wu
Shanghai Jiao Tong University
317 PUBLICATIONS **5,269** CITATIONS

SEE PROFILE

Wei Shu
University of New Mexico
234 PUBLICATIONS **1,848** CITATIONS

SEE PROFILE

Placement of Proxy-Based Multicast Overlays

Min-You Wu, Yan Zhu, and Wei Shu
Department of Electrical and Computer Engineering
The University of New Mexico

Abstract — Proxy-based multicast is an approach to implementing the multicast function with proxy servers, which has many advantages compared to the IP multicast. When proxy locations are predetermined, many routing algorithms can be used to build a multicast overlay network. On the other hand, the problem of where to place proxies has not been extensively studied yet. A study on the placement problem for proxy-based multicast is presented in this paper. The objectives of proxy placement can be minimization of delay time, minimization of bandwidth consumption, or minimization of both. These performance metrics are studied in this paper and a number of algorithms are proposed for multicast proxy placement. Performance study is presented and practical issues of proxy placement are also discussed.

1 Introduction

Multicast is an important technique that can distribute popular contents to many users with minimal bandwidth consumption. Multicast can be applied to hot event broadcast, audio/video delivery, video conference, and new software dissemination. Multicast is able to minimize the server load and remove "hot spots" from networks when many users access the same content, especially live contents.

IP multicast has been researched and developed for more than a decade [9, 10], but it has not been widely deployed [11, 13]. As an alternative to IP multicast, the rapid development of proxy-based multicast technologies has increased greatly over the last couple of years [6, 7, 17, 32, 28]. Proxy-based multicast challenges the traditional IP multicast by moving up the multicast support from the network layer onto the application layer. It potentially provides better solutions to scalability, flexibility, heterogeneous support, and efficiency for various enabling applications.

A general approach to implementing the proxy-based multicast is to build an overlay network. Many routing algorithms have been proposed for multicast tree building [6, 7, 17, 15, 28, 16]. Some of them proposed to minimize delay between the source and the clients [6]; others proposed to minimize bandwidth consumption [7]. In fact, delay and bandwidth consumption are both important considerations for multicast. The primary goal of multicast is to reduce the bandwidth consumption. On the other hand, delay should not increase too much since multicast is often used for real-time applications. Balance between these two metrics is essential to the best design of the proxy-based multicast. Comparison studies show that the shortest-path tree generates the smallest delay, Minimal Steiner tree generates the lowest

bandwidth consumption, and the core-based tree falls in between the two extremes [34, 33, 12]. However, no metric includes both criteria, delay and bandwidth consumption. In this paper, a hybrid optimal tree will be defined to optimize the trade-off between them.

The routing algorithms assume that the locations of proxies are predetermined. The problem of where to place these proxies has not been extensively studied yet. Where a proxy can be placed, however, is a practical, non-trivial problem. Four practical issues must be addressed: the freedom of proxy placement, the bounded proxy capacity, the dynamic feature of network traffic, and resilience of multicast overlay networks.

A multicast proxy is a replication engine, which may be placed on end-hosts, network edges, or network cores. However, one is not able to place a proxy everywhere. Anyone has freedom to place replication engine to his/her end host, which is called the *end-system multicast* [7, 41]. Although the end-system multicast is easy to deploy, replication engines can only be placed on the locations that *do not* offer maximum benefits. When replication engines can be placed onto network edges or network cores, it is called the *proxy-based multicast*. The proxies placed on network edges provide better performance than those on end hosts and the proxies placed onto network cores can maximize the benefit [19, 39]. Later we will show a comparison of the end-system and proxy-based system.

In fact, an organization network operator may place a proxy close to routers in the organization network; and a network provider may place a proxy in the core network. Here, we use the term "*zone*" to describe the region where a single administrator has the authority to place proxies. The administrator of a zone does not have any knowledge of other zones, nor is able to place proxies onto other zones. Proxies can be placed to optimize the traffic within a zone. A placement strategy needs to be addressed within a single zone to achieve maximum benefits. A simulation study found that when traffic in every zone is optimized, a global optimality may be achieved [19]. This simulation has been conducted with the transit-stub, power-law, and BGP graphs of different sizes. The number of zones is from 5 to 40. The difference between the zone placement and global placement is only about 5%. Thus, with the power of localization, optimal decision within a zone approximates global optimization in a federation of zones. Even if the traffic in other zones is not minimized, a zone can optimize its own traffic with a proper proxy placement. In reality, though proxy placement depends on many factors such as available proxy servers and the locations that are easy to deploy, an optimal placement algorithm provides the administrator useful hints for a right decision.

In the Internet, there might be many multicast streams and each proxy might replicate data for multiple streams. A proxy with a bounded capacity may not be able to handle all the streams. Since the multicast traffic is on top of other traffics, the link capacity and congestion must be considered in routing and placement algorithms. Thus, when a new stream is to be multicast, the residual capacity is considered. An algorithm for a single stream provides a basis to develop algorithms of multiple streams and proxies with

a bounded capacity.

A crucial issue in proxy placement is the change of network traffic. An optimal proxy placement for a certain traffic pattern may become less optimal for a different pattern. Fortunately, the network traffic pattern does not change quickly. Yet another approach to accommodating the traffic change is *proxy subscription* [37]. In this approach, each zone uses the placement algorithm to deploy a number of proxies according to the previous statistics of network traffic. For each multicast event, a subset of proxies are subscribed to form a multicast tree. The proxy placement algorithm is used to determine the subset. This approach adapts to dynamically changing traffic. After a time period, the locations of proxies will be adjusted according to the change of traffic patterns. A credit system can be used to evaluate which location is more important. The proxy subscription approach also can be used to make the overlay resilient. When a proxy fails, some other proxies will be subscribed for a new multicast tree.

In this paper, instead of dealing with practical issues, we will study the basic proxy placement problem. Our goal is to develop the methodology and algorithms for the *multicast proxy placement (MPP)* problem as well as its association with the *multicast proxy routing (MPR)* problem. In particular, our contributions are:

- We formulate the MPP and MRP problems and their performance metrics.
- We introduce a number of MPP algorithms including some advanced algorithms for optimal proxy placement.
- We conduct experiments for performance of MPP algorithms which increases our understanding of the MPP problem. Guidelines are given for general proxy placement.

The results obtained from this work are also applicable to the Mbone router placement. In this paper, we will not work on protocols for proxy deployment. We will not focus on several issues that crop up on practice, such as dealing with joining and leaving participants in the sessions, and building a distributed version of the algorithms. Our goal instead is to provide a well-formulated optimization problem and understand the complexities involved in solving the problem when the constraints are static and known. Also, a set of heuristics for the optimization problem is designed.

The remainder of this paper is organized as the following. Section 2 formulates the MPR and MPP problems and introduces the performance metrics. Underlying routing algorithms are presented in Section 3. Section 4 presents four groups of MPP algorithms: routing-independent, routing-aware, greedy, and local search. Experimental results are presented in Section 5. Related works are given in Section 6. In Section 7, we give a guideline of proxy placement and future research directions.

2 Problem Formulation

The proxy-based multicast is implemented as an overlay network. An overlay network consists of a set of proxies placed on a substrate network. The substrate network is a collection of routers connected by links. Then a multicast tree of proxies is built with a routing algorithm. The multicast tree can be single-sourced or multi-sourced. In this paper, we only consider the single-sourced multicast tree.

From the perspective of a client, delay is the main concern. On the other hand, the cost in terms of the aggregate bandwidth consumption is a main indication of network efficiency from a network provider's point of view. Thus, the quality of a multicast tree can be judged by two metrics: the delay and the cost. The delay of a multicast tree is evaluated in terms of the end-to-end delay between the source and every client. The cost of a multicast tree is the sum of bandwidth consumption of all links. Bandwidth consumption measures the usage of network resources. The more hops a message travels, the more resources that will be consumed.

We first present a model of the substrate network and the overlay network before the objective functions are defined. The substrate network is represented by an undirected graph G, consisting of a set of N nodes S=$\{n_1, n_2, ..., n_N\}$. Each node has a weight $w(i)$ to represent the number of clients concentrated at node n_i. There is one source node, $n_s \in S$. There are two parameters associated with link $l_{i,j}$ between nodes n_i and n_j: $\theta_d(i,j)$ for delay and $\theta_c(i,j)$ for cost. If there is no link $l_{i,j}$ of the substrate network between nodes n_i and n_j, let $\theta_d(i,j) = \theta_c(i,j) = \infty$.

A multicast proxy is a replication engine that is able to replicate the incoming data stream for multiple output links. A multicast overlay network consists of a set of P proxies $\{p_1, p_2, ..., p_P\}$. These proxies are to be placed on P nodes. All nodes with proxies placed plus the source node n_s are called the *replication nodes*. Thus, there are $P+1$ nodes with the replication capability. A routing algorithm connects the replication nodes to a multicast tree based on some criteria, such as minimal delay, minimal bandwidth consumption, or a combination of them. A non-replication node is routed to a proxy according to the same criteria. More specifically, a multicast tree M can be defined by $\{X, P, Y\}$. Here, X is a mapping function of proxy placement

$$x(i) = \begin{cases} 1 & \text{if } n_i \text{ is a replication node} \\ 0 & \text{otherwise} \end{cases}$$

P is a function for each node to know its immediate parent replication node in the tree:

$$p(i) = k \quad \text{if } x(k) \equiv 1 \text{ and } n_k \text{ delivers the stream to } n_i \text{ by using a unicast connection}$$

and Y defines which substrate network links are used when $n_{p(i)}$ delivers the stream to n_i by using a unicast connection,

$$y(i,u,v) = \begin{cases} 1 & \text{if } l_{u,v} \text{ is used for path from } n_{p(i)} \text{ to } n_i \\ 0 & \text{otherwise} \end{cases}$$

First, we define two basic metrics, the aggregate delay and the aggregate bandwidth consumption. For each node n_i, the end-to-end delay from node n_i to the source n_s is

$$d_{\mathrm{M}}(i) = \overbrace{\sum_{u=1}^{N}\sum_{v=1}^{N}\theta_d(u,v)\cdot y(i,u,v)}^{d'} + \underbrace{d_{\mathrm{M}}(p(i))}_{d''} \tag{1}$$

where, d' is the delay from node n_i to replication node $n_{p(i)}$, and d'' is recursively defined as the delay from $n_{p(i)}$ to source node n_s. If the multicast stream is not directly delivered from source n_s to n_i, node n_i must receive the stream from its parent $n_{p(i)}$, and the delay $d_{\mathrm{M}}(i)$ consists of both parts, d' and d''. Otherwise, the delay $d_{\mathrm{M}}(i)$ is equal to d', since $d_M(s) = 0$ for source node n_s. The aggregate delay of a multicast tree is the sum of delays, weighted by $w(i)$, between the source and every clients along the multicast tree M:

$$D_{\mathrm{M}} = \sum_{i=1}^{N} w(i)\cdot d_{\mathrm{M}}(i) \tag{2}$$

where $w(i)$ is the number of clients concentrated at node n_i.

For node n_i, the cost of delivering one stream from its parent node $n_{p(i)}$ is

$$c_{\mathrm{M}}(i) = \sum_{u=1}^{N}\sum_{v=1}^{N}\theta_c(u,v)\cdot y(i,u,v). \tag{3}$$

To calculate the bandwidth consumption for a non-replication node n_i, $w(i)$ streams are replicated from its parent node $n_{p(i)}$. For a replication node n_i, only one stream is sent from its parent replication node $n_{p(i)}$. We use b_{M} to compute the bandwidth consumption,

$$b_{\mathrm{M}}(i) = \begin{cases} c_{\mathrm{M}}(i) & \text{if } x(i) \equiv 1 \\ w(i)\cdot c_{\mathrm{M}}(i) & \text{otherwise} \end{cases} \tag{4}$$

The aggregate bandwidth consumption of multicast data distribution is defined as

$$B_{\mathrm{M}} = \sum_{i=1}^{N} b_{\mathrm{M}}(i) \tag{5}$$

These metrics can be normalized as:

$$D = D_{\mathrm{M}}/D_{\mathrm{IPM}} \qquad\qquad B = B_{\mathrm{M}}/B_{\mathrm{IPM}}$$

where IPM stands for the traditional IP multicast tree, which is commonly used as a reference point for comparison. Due to the SPT-oriented routing algorithm used in IP multicast, D_{IPM} is the minimal value among all possible D_{M}. Based on the same routing, B_{IPM} is obtained by setting every node as an IP multicast router.

Both D and B depend on the routing algorithm used to construct the multicast tree M. Achieving minimal delay and reducing bandwidth consumption are usually contradictory to each other. It is known that the Minimal Spanning Tree (MST) minimizes the bandwidth consumption, but involves a longer delay than that of the Shortest Path Tree (SPT). On the other hand, SPT minimizes the delay, but consumes more bandwidth than the MST routing. A theoretic bound was given for balancing minimum spanning trees and shortest-path trees [22]. That is, given a ε, the delay is at most $1+\varepsilon$ times the shortest-path tree delay and the total cost is at most $1+2/\varepsilon$ times the cost of a minimum spanning tree. In order to optimize the both metrics simultaneously, we define a new metric, *Weighted Sum of Delay and Bandwidth consumption* (*WSDB*) as

$$WSDB = \lambda D + B \tag{6}$$

where λ is a weight indicating the relative importance of the delay. In a scenario such as interactive video conference where delay is considering as a central issue, weight λ should be raised. Under another scenario such as broadcasting a hot event where control of the network bandwidth is more important, weight λ should be decreased. We will provide an algorithm that dynamically adjusts the value of λ so that the realtime requirement can be satisfied.

In general, a multicast tree M defined by $\{X, P, Y\}$ is relevant to both the proxy placement and routing, where X defines the proxy placement, and P and Y define the routing. Therefore, the problem can be formulated as

- PROBLEM 1. Given the proxy placement X, *design a routing algorithm* A_R to construct the multicast tree, P and Y, subject to minimizing D, B, or *WSDB*
- PROBLEM 2. By limiting the number of proxies to be placed, *design a placement algorithm* A_P to generate a mapping function X, subject to optimizing D, B, or *WSDB* with a routing algorithm A_R

Two design problems have been formulated for a multicast overlay network, the multicast proxy routing (MPR) problem and the multicast proxy placement (MPP) problem, together with three important performance metrics, D, B, and *WSDB*. Solving PROBLEM 2 usually is more difficult than PROBLEM 1. In some cases, preference is given to the co-design between PROBLEM 1 and PROBLEM 2, which is called the routing-aware placement algorithm in this paper. Routing algorithms play an important role, but without an appropriate placement, routing algorithms alone will not be able to produce good performance.

3 Underlying Routing Algorithms

Given a placement X, a routing algorithm will build a multicast tree M optimized toward certain metrics. Based on these metrics, these multicast trees can be classified into the following three categories.

3.1 Shortest Path Tree (SPT) — M_{SPT} minimizing D

In a M_{SPT}, every node $n_i \in S$ connects to the source n_s through the shortest path. If the shortest path from n_i to n_s includes some proxies, node n_i is connected to the nearest proxy along the path; otherwise it is connected to the source directly. M_{SPT} minimizes the delay $d_{\text{M}}(i)$ from every node n_i to n_s, respectively, therefore optimizing the aggregate delay D_{M}. Thus, regardless of placement X, the metric D remains the same since the shortest paths for every node will not change. Figure 1 shows the routing algorithm A_{SPT}. In step 1, We use the Dijkstra's algorithm to construct a single-source M_{SPT} and retain its complexity as $O(N^2)$. In step 2, the M_{SPT} tree has been traversed to establish P and Y. Since every link on the M_{SPT} tree will be visited at most N times and the number links on M_{SPT} is in order of N, the time spent in step 2 is within $O(N^2)$ too.

Function: given X, generate P and Y
Objective: minimize D
step 1) Apply the Dijkstra's algorithm to the entire graph G to generate a single-source SPT with:
$sp(i)$ representing the shortest path between n_i and n_s
nextNodeOnSPT(i) representing the next node on the shortest path towards n_s
step 2) for each node n_i excluding n_s, let $k_1 = i$
do $k_2 = \text{nextNodeOnSPT}(k_1)$;
$y(i,k_1,k_2) = 1$;
$k_1 = k_2$;
while $(x(k_2) \neq 1)$
$p(i) = k_2$;
$p(s) = s$;

Figure 1: Routing Algorithm A_{SPT} for M_{SPT} minimizing D

3.2 Minimal Spanning Tree (MST) — M_{MST} minimizing B

The Minimal Spanning Tree (MST) is a minimal-cost tree connecting every node in a graph. Figure 2 shows the routing algorithm A_{MST}. In order to minimize B, a subgraph with all replication nodes is constructed by utilizing the shortest path among them. In step 1, for simplicity, all-pair shortest paths are constructed with a complexity of $O(N^3)$. In fact, step 2 needs shortest paths among all replication nodes and step 5 needs shortest paths from every replication nodes to all non-replication nodes. Step 2 constructs the subgraph and step 3 applies the Prim's algorithm on this subgraph to find the MST with a complexity of $O(P^2)$ [1]. In this way, bandwidth consumption among replication nodes may be minimized. The commonly used Steiner Minimal Tree (SMT) does not work here since any included non-replication nodes cannot replicate the traffic. In step 4, the MST tree has been traversed to establish P and Y for

all replication nodes. In step 5, all non-replication nodes are directed to the nearest replication nodes, which guarantees optimal bandwidth consumption. Among the five steps in Figure 2, step 1 dominates in complexity, therefore the complexity of A_{MST} is $O(N^3)$.

Function: given X, generate P and Y
Objective: minimize B

step 1) apply the Dijkstra's algorithm to the entire graph G to generate all-pair shortest paths with:
  $sp(i,j) = sp(j,i)$ representing the shortest path between n_i and n_j
  nextNodeOnSP(i,j) representing the next node on the shortest path toward n_j

step 2) construct a subgraph O with
  all nodes with $x(i) = 1$
  use $sp(i,j)$ as the virtual link cost $v(i,j)$ between n_i and n_j, where $x(i) = x(j) = 1$

step 3) apply the Prim's algorithm to the subgraph O to generate a MST with
  $p(i) = j$ if $v(i,j)$ is included in MST towards n_s

step 4) for each node n_i with $x(i) = 1$ excluding n_s, let $k_1 = i$
  do $k_2 = \text{nextNodeOnSP}(k_1, p(k_1))$;
  $y(i,k_1,k_2) = 1$;
  $k_1 = k_2$;
  while $(x(k_2) \neq 1)$
  $p(s) = s$;

step 5) for each node n_i with $x(i) = 0$, let $k_1 = i$
  let $p(i) = q$ be a replication node such that $sp(i,q) \leq sp(i,j)$, where $x(q) = x(j) = 1$
  do $k_2 = \text{nextNodeOnSP}(k_1, q)$;
  $y(i,k_1,k_2) = 1$;
  $k_1 = k_2$;
  while $(k_2 \neq q)$

Figure 2: Routing Algorithm A_{MST} for M_{MST} minimizing B

3.3 Hybrid Optimal Tree (HOT) — M_{HOT} minimizing *WSDB*

The *Hybrid Optimal Tree (HOT)* minimizes *WSDB*, a compromise between SPT and MST. Many applications can promote this newly defined metric *WSDB*. For example, realtime video multicast streaming can tolerate neither long delay nor high bandwidth consumption. Two algorithms are used here to generate the multicast tree M_{HOT}. The first one is the Prim-Dijkstra's algorithm.

Prim-Dijkstra's (P-D) algorithm $A_{\text{P-D}}$ — The Prim-Dijkstra's algorithm [21] may be used to find the M_{HOT} tree. It builds a tree by starting at a node and bringing in nodes by picking the one with the best label: $MIN_k(\beta \times dist(n_s, p_k) + dist(p_k, n_i))$, where $0 \leq \beta \leq 1$ is a constant used to parameterize the calculation. The complexity of Prim-Dijkstra's algorithm $A_{\text{P-D}}$ is $O(N^2)$ and details are described

in [21].

Since the P-D algorithm $A_{\text{P-D}}$ is unable to guarantee a minimal *WSDB* value and a parameter needs to be adjusted, we use a greedy algorithm to directly minimize the *WSDB* value.

Direct-DB (DDB) algorithm A_{DDB} — DDB routes proxies one by one as shown in Figure 3, starting from the nearest proxy to the source. The first proxy connects to the source through the shortest path. The following proxies connect to the source or a placed proxy, whichever minimizes *WSDB*. After the multicast tree is established, each non-replication node connects to the replication node that generates the smallest *WSDB* value. The complexity of this algorithm is $O(PN^2)$.

Function: given X, generate P and Y
Objective: minimize *WSDB*

step 1) apply the Dijkstra's algorithm to the entire graph G to generate all-pair SPTs with:
  $sp(i,j) = sp(j,i)$ representing the shortest path between n_i and n_j
  nextNodeOnSPT(i,j) representing the next node on the shortest path toward n_j
step 2) sort all replication nodes in ascending order of length of their shortest path towards n_s.
  reset $mark[i]$ = false for all n_i and $mark[s]$ = true
step 3) establish an init routing with all n_i following the SPT towards n_s
  compute the init value $WSDB_{\text{cur}}|_{p(i)=s}$
step 4) for each node n_i with $x(i) = 1$, $mark[i]$ = false, according to their order in step 2
  let $k_1 = i$ and $p(i) = q$ be an already-routed replication node such that $gain(i,q) \geq gain(i,j)$,
    where $mark[q] = mark[j]$ = true
    $gain(i,j) = WSDB_{\text{cur}} - WSDB_{\text{new}}|_{p(i)=j}$
  do $k_2 = \text{nextNodeOnSPT}(k_1, q)$;
    $y(i,k_1,k_2) = 1$; $k_1 = k_2$;
  while ($k_2 \neq q$)
  update $WSDB_{\text{cur}}|_{p(i)=s \to p(i)=q}$
step 5) for each node n_i with $x(i) = 0$
  let $k_1 = i$ and $p(i) = q$ be a replication node such that $gain(i,q) \geq gain(i,j)$,
    where $x(q) = x(j) = 1$
    $gain(i,j) = WSDB_{\text{cur}} - WSDB_{\text{new}}|_{p(i)=j}$
  do $k_2 = \text{nextNodeOnSPT}(k_1, q)$;
    $y(i,k_1,k_2) = 1$; $k_1 = k_2$;
  while ($k_2 \neq q$)
  update $WSDB_{\text{cur}}|_{p(i)=s \to p(i)=q}$

Figure 3: Routing Algorithm A_{DDB} for M_{HOT} minimizing *WSDB*

This algorithm generates a tree that balances delay and cost requirements. As will be found in the experimental result, this algorithm generates a delay that is only a few percent longer than that of SPT, and a bandwidth consumption that is a few percent larger than that of MST.

In summary, design of routing algorithms has been motivated by various performance metrics. Depending on characteristics of applications, selection of a right routing algorithm is crucial to overall performance. Usually, routing algorithms are applied after proxies are placed. Sometimes, routing algorithms can help the decision of proxy placement, as discussed in the following section.

4 Proxy Placement Algorithms

In an end-system multicast, though each end-system can replicate streams, they cannot be placed in strategically important locations in most cases. On the other hand, proxies can be placed to obtain the best performance. Table 1 shows comparison of end-system multicast and proxy-based multicast using the DDB routing, where the number of nodes in the graph is 200 and the number of replication engines is 40. It indicates that the proxy-based multicast performs much better than the end-system multicast. The end-system has about 40% longer delay and almost 100% larger cost.

Table 1: Comparison of end-system multicast and proxy-based multicast

	D	B
End-system multicast	1.44	2.62
Proxy-based multicast	1.02	1.37

An MPP algorithm finds the optimal locations for proxies. The objective functions may be minimization of D, B, or *WSDB*. In general, the MPP problem can be reduced to the P-median problem, which is an NP-hard problem [14]. Thus, heuristics are used for the MPP algorithms. Here, the placement algorithms are presented in four categories: routing-independent, routing-aware, monotonic incremental, and local search.

4.1 Routing-independent algorithms

The routing-independent placement algorithms do not rely on routing information. Four algorithms are presented here. Random and Max-degree placement algorithms are for arbitrary topology, and Strategic and Advanced Strategic Placement (ASP) algorithms are for the Internet topology.

Random. The random algorithm randomly chooses P locations among $N-1$ nodes (excluding n_s) to place proxies. Its complexity is $O(N)$.

Max-Degree (Max-D). The max-degree algorithm places P proxies to the nodes with the highest node degrees. The complexity of this algorithm is $O(N\,logN)$ due to sorting of nodes.

Strategic. The strategic placement selects strategically important nodes to place proxies. In the Internet topology, proxies are placed on the transit nodes first, then the stub nodes as described in [17]. Ties

are broken randomly. The complexity of this algorithm is $O(N)$.

Advanced Strategic Placement (ASP). The ASP algorithm is also for the Internet topology. First, we select the transit nodes in the order of the number of stub nodes connected to it. Then, we select the stub nodes that are connected to transit nodes directly in the order of the number of stub nodes connected to it. Finally, other stub nodes are selected in the order of the node degree. The complexity of this algorithm is $O(N\,logN)$.

Since all of the routing-independent placement algorithms do not utilize the traffic information, they are traffic-independent too. Thus, the placement generated by these algorithms does not change with the traffic pattern.

4.2 Routing-aware algorithms

The routing-aware placement algorithms find a better placement by utilizing the routing information. First, a routing algorithm is applied to provide the routing paths by building a multicast tree M_{INIT} assuming all nodes are replication nodes. Once we have the multicast tree M_{INIT}, traffic T of a node is defined as traffic passing through the node plus the traffic terminated at the node assuming no proxy is placed. If T of a node is more than 1, the node is a *branching node*. The number of branching nodes $Q \leq N-1$. If the number of proxies to be placed $P = Q$, there is no repeated traffic on every link, and the bandwidth consumption is minimal. When $P < Q$, a placement algorithm selects nodes to place P proxies. Two routing-aware algorithms are presented here, Max-Traffic and Max-Traffic-Distance (Max-TD). In the following, we will use O_{routing} to represent the complexity of the routing algorithm to build the multicast tree, where O_{routing} is $O(N^2)$ for SPT, $O(N^3)$ for SMT, and $O(PN^2)$ for HOT.

Max-Traffic (Max-T). The traffic is computed for each node based on routing in M_{INIT} as shown in Figure 4. The proxies are placed at the top P locations with the most concentrated traffic. Its complexity is $O(N\,logN) + O_{\text{routing}}$.

Max-Traffic-Distance (Max-TD). TD is the product of the node traffic and the distance between the node n_i and its parent replication node $n_{p(i)}$. As shown in Figure 5, the first proxy is placed on the node with the highest TD. Then the value of $t(i)$ is recomputed for each non-replication node and the next proxy is placed on the node with the highest TD. This process continues until P proxies are placed. The complexity of this algorithm is $O(N^2) + O_{\text{routing}}$.

Routing-aware placement algorithms utilize traffic information. Therefore, they are able to produce better placement. However, the placement will change when the traffic pattern changes.

Function: for a routing algorithm A_{R}, generate placement X
Placement priority: nodes with Max-Traffic
step 1) Let $x(i) = 1$ for all i, apply A_{R} to generate a M_{INIT} with
nextNodeOnTree(i) representing the next node on the path towards n_s
step 2) Let $t(i) = 0$ for all i
step 3) For each n_i, let $k_1 = i$ and $t(i) = t(i) + w(i)$
do $k_2 = \text{nextNodeOnTree}(k_1)$
$t(k_2) = t(k_2) + w(i)$
$k_1 = k_2$
while $(k_2 \neq s)$
step 4) Sort all nodes in descending order of $t(i)$
step 5) Let $x(i) = 0$ for all i, and $x(s) = 1$
For the first P node in order let $x(i) = 1$

Figure 4: Placement Algorithm: Max-Traffic

Function: for a routing algorithm A_{R}, generate placement X
Placement priority: nodes with Max-TD
step 1) Let $x(i) = 1$ for all i, apply A_{R} to generate a M_{INIT} with
nextNodeOnTree(i,p(i)) representing the next node on the path towards $n_{p(i)}$
step 2) Let $x(i) = 0$ for all i, and $x(s) = 1$
step 3) For $k = 0$ to P, let $t(i) = 0$ for all i
step 4) For each n_i, let $k_1 = i$
do $k_2 = \text{nextNodeOnTree}(k_1, p(i))$
$t(k_2) = t(k_2) + 1$ if $x(i) = 1$
$t(k_2) = t(k_2) + w(i)$ if $x(i) = 0$
$k_1 = k_2$
while $(k_2 \neq p(i))$
step 5) find a node q such that $t(q) * c_{\text{M}}(q) \geq t(j) * c_{\text{M}}(j)$, where $x(q) = x(j) = 0$
step 6) let $x(q) = 1$

Figure 5: Placement Algorithm: Max-TD

4.3 Monotonic Incremental algorithm

The *Monotonic Incremental Proxy Placement (MIPP)* algorithm is designed for a near-optimal solution. Especially when an exhaustive search for the optimal solution is not feasible, it can be used as a measure of how other algorithms perform.

MIPP. Suppose P proxies are to be placed. We choose one location at a time. In the first iteration, we presume a proxy to be placed on each of the $N-1$ locations and evaluate their objective functions, respectively. The location that yields the best value of the objective function is selected to place the first proxy. In the second iteration, we search for the next best location that, considering both the source and the proxies already placed, yields the smallest value of the objective function. This process continues until all P proxies are placed. Its complexity is $O(N^2) \times O_{\text{routing}}$.

Function: for a routing algorithm A_{R}, generate placement X
Placement priority: minimize *WSDB*
step 1) Let $x(i) = 0$ for all i, apply A_{R} to generate a M_{INIT} and
compute the init value $WSDB_{\text{cur}}|_{p(i)=s}$
step 2) Let $x(s) = 1$
step 3) For $k = 0$ to P
let n_q be node with $x(q) = 0$ such that $gain(q) \geq gain(j)$,
where $x(q) = x(j) = 0$ and
$gain(j) = WSDB_{\text{cur}} - WSDB_{\text{new}}|_{x(j)=1}$ by applying A_{R}
Let $x(q) = 1$
update $WSDB_{\text{cur}}|_{x(q)=0 \to x(q)=1}$

Figure 6: Placement Algorithm: MIPP

Among the above placement algorithms, all routing-independent algorithms are traffic independent, therefore, proxies are not relocated when traffic pattern changes. However, their performance is not satisfactory in general. Other algorithms, including Max-Traffic, Max-TD, and MIPP, produce good performance but depend on the traffic pattern. The *proxy subscription* method can be applied to deal with the dynamically changed traffic [37].

Finally, a local search algorithm will improve the placement generated by these placement algorithms.

4.4 Local search algorithm

Local search was one of the early techniques for combinatorial optimization. It has been applied to solve NP optimization problems [31]. The principle of local search is to refine a given initial solution point in

the solution space by searching through the neighborhood of the solution point. We may apply the local search technique to refine the proxy placement solution.

Local search. Any of the above algorithms can be used to generate an initial solution. A randomly selected proxy is removed and then placed on a randomly selected non-replication node. If the initial solution is improved, the proxy remains in the new location. Otherwise, the proxy is restored back to its original location. This process repeats for B times. The complexity of the local search is $B \times O(N)$ for SPT tree and $B \times O_{\text{routing}}$ for other type of trees since the multicast tree needs to be rebuilt for each search. Here, B is a constant.

Example 1. To illustrate various MPP algorithms, a sample graph G_1 is shown in Figure 7 with $N = 15$, which is generated by the Georgia Tech Internetwork Topology Generator (GT-ITM) [2]. The graph

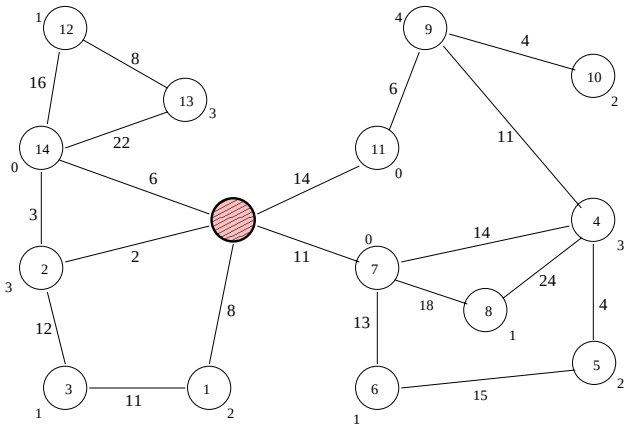
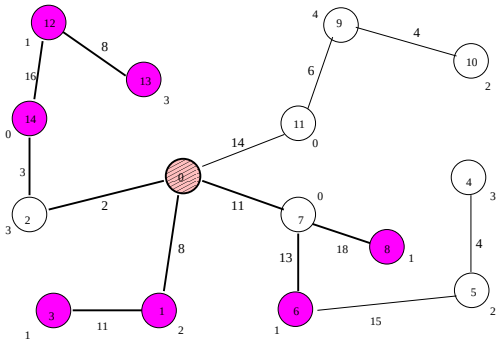
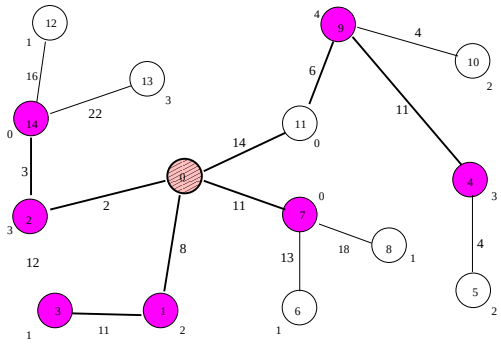


Figure 7: Sample graph G_1 for MPP algorithms.

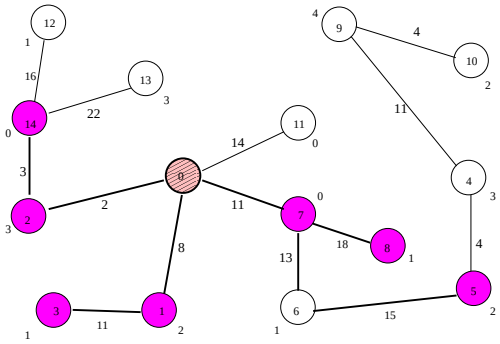
generated by GT-ITM with the transit-stub model closely resembles the Internet hierarchy. Nodes n_0 and n_2 are transit nodes. Node sets $(n_4, n_5, n_6, n_7, n_8), (n_9, n_{10}, n_{11}), (n_{12}, n_{13}, n_{14}), (n_1, n_3)$ are four stubs. Links $l_{2,14}$ and $l_{0,1}$ are transit-stub links. Link $l_{4,9}$ is a stub-stub link. The source is at node n_0. The numbers listed outside of nodes are $w(i)$ of the nodes, and the numbers on links are the $e(i, j)$. The optimal solution is obtained by the exhaustive search. The routing algorithm A_{DDB} is used in this example, subject to minimizing *WSDB*. The results of various MPP algorithms are compared with the optimal solution. Figure 8 shows the proxy placement generated by the Random, Max-degree, Strategic, ASP, Max-traffic, Max-TD, and MIPP algorithms. The optimal solution by exhaustive search is the same as the one generated by MIPP. Table 2 shows the delay D, bandwidth consumption B, and *WSDB* of these MPP algorithms, where $\lambda = 1$. The value of D_{IPM} is 452 and the value of B_{IPM} is 147. In fact, the B_{IPM} value is not minimal since IP multicast uses the shortest-path routing. The Random placement has high D and B values. The



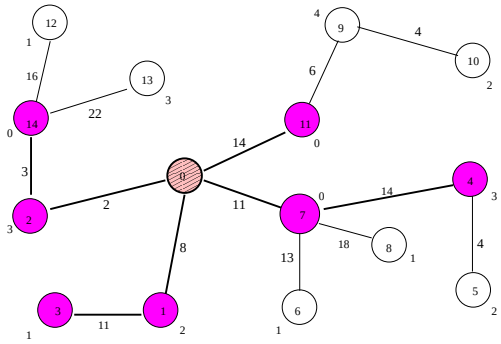
(a) Random



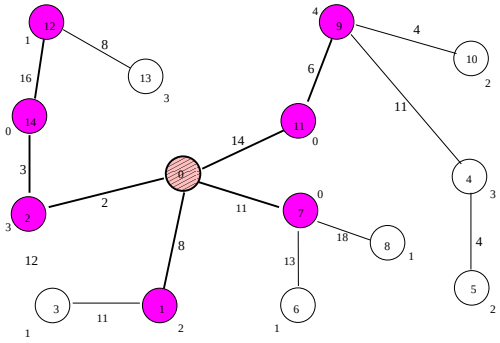
(b) Max-Degree



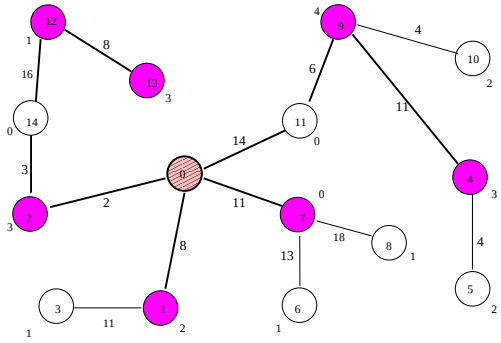
(c) Strategic



(d) ASP



(e) Max-Traffic, Max-TD



(f) MIPP, Optimal

Figure 8: Different proxy placements for G_1.

Max-Degree placement shows a lower delay and bandwidth consumption. In the strategic placement, non-replication node n_4 is connected to replication node n_5 instead of n_7, resulting in a lower *WSDB* value. ASP exhibits the smallest delay but its B value is higher than that of the Max-Degree, Max-Traffic, Max-TD and MIPP algorithms. The Max-Traffic and Max-TD algorithms generate the same placement in this small graph. The MIPP algorithm generates the same tree as the optimal placement. Both of them exhibit the smallest *WSDB* value, which is the objective function of routing algorithm A_{DDB} used here.

Table 2: D, B, and *WSDB* of G_1 for MPP algorithms

MPP Algorithms	D_{M}	B_{M}	D	B	*WSDB*
Random	537	322	1.19	2.19	3.38
Max-Degree	452	199	1.00	1.35	2.35
Strategic	476	307	1.05	2.09	3.14
ASP	452	229	1.00	1.56	2.56
Max-Traffic, Max-TD	458	165	1.01	1.12	2.13
MIPP, Optimal	458	141	1.01	0.96	1.97

5 Algorithms for Dynamic Realtime Applications

Though the delay is not very critical for some applications such as one way broadcast of events, most interactive applications require minimization of delays, such as the video conference and online course discussion. Furthermore, multicast streams start at different times; clients join and leave; and network traffic changes dynamically. A proxy placement for a traffic pattern might not be the best for a different traffic pattern. These realtime and dynamic issues need to be considered. In this section, we discuss the method to control the delay and to handle the dynamic features of applications and traffic patterns.

Multicast is mainly used to minimize the server load and the cost of content delivery. Normally it will cause longer delay compared to unicast. However, increase of delay must be under control so that the realtime requirement of interactive applications is satisfied. We dynamically tune the value of λ in Equation 6 to meet this requirement. By default, the value of λ is 1. An application may set a threshold as the upper limit on increase of delay for each client. The routing algorithm could check the value of D. If the result is not satisfactory, the value of λ will be adjusted until the requested threshold is met. Of course, reduction of D will cost more bandwidth consumption in most cases. The λ value also can be adjusted to reduce the bandwidth consumption. This method makes the routing algorithm adaptive to different types of applications. For interactive applications, the threshold can be low, such as 10%, which means the delay increase is no more than 10% of the minimal delay. For non-interactive applications, such as playback of

the stored video, the threshold can be set high, say, 50%, so that the cost will be minimized.

Multiple multicast groups may simultaneously exist in the Internet. When a multicast stream starts, a source tree is built, which will be continuously constructed as the new clients' requests arrive. An event can be scheduled or impulse. Many people will watch a scheduled event, such as U.S. open final, from the beginning. On the other hand, for an impulse event, such as Terror Strikes America, requests will arrive after the event begins. A better multicast tree could be built for a scheduled event since more information is available when the tree starts to build. We assume a non-preemptive approach so that the route of a client will not change frequently. When a client joins a multicast stream, if its proxy has been part of the multicast tree, the proxy simply replicates another stream for the newly joined client. Otherwise, the proxy will first join the multicast stream using a routing algorithm that minimizes the delay, the cost, or the combination of them. When a client leaves a multicast stream, its proxy checks if this is the last replication stream. If so, the proxy will tear down the connection to the multicast tree as well.

The multicast groups and the network traffic will change dynamically over the time. Routing can adapt to the dynamics promptly, however, proxies cannot be replaced very often. The proxies can be replaced after a time period, such as a few months. If we were able to know the network traffic in the next time period in advance, an optimal placement could be achieved. However, we only can make a prediction based on the previous time periods. An *Adaptive Placement* algorithm is used. During a regular running phase, the multicast traffic is continuously monitored and recorded. Periodically, the system will enter an adjusting phase, in which the data from the traffic monitor will be analyzed. Proxies will be replaced based on the traffic analysis. The system will be back to the regular running phase again as shown in Figure 9. Simulation will be conducted in the experiment section to evaluate the consequences of this prediction and the degree of proxy migration.

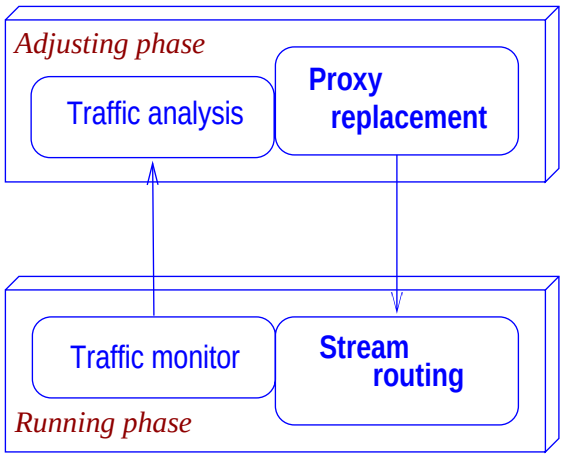


Figure 9: Adaptive Placement

6 Experimental Results

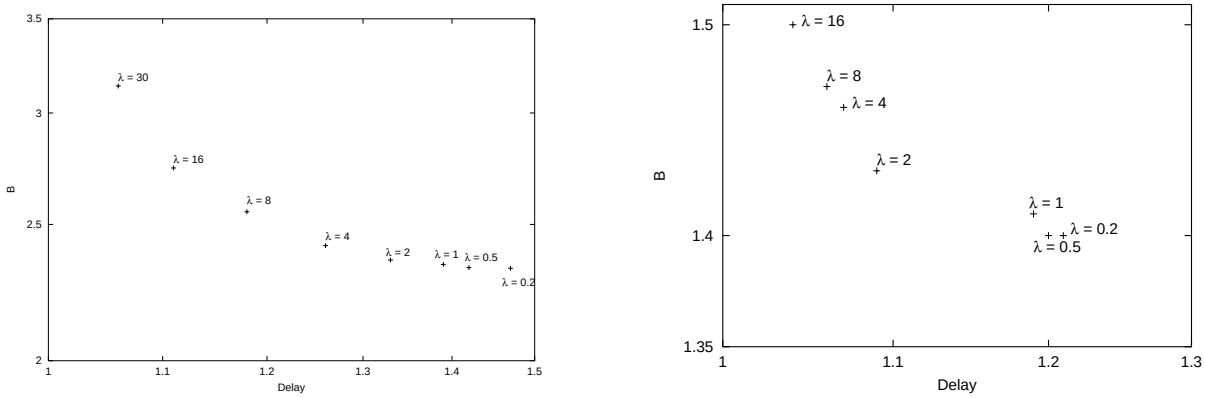
In this section, proxy placement algorithms with different routing strategies are evaluated. We use three topology models in our simulation study, *transit-stub, power-law, and BGP* graphs. The transit-stub model is analogous to the current Internet backbone transit networks and local networks. It has a similar property of power-law model in its stub domain and a similar topology of BGP graph in its transit domain. We mainly use the transit-stub graph of 200 nodes in our performance test. Results of 1,000 nodes are also presented. In addition, the performance of power-law and BGP graphs will be presented for comparison.

The transit-stub graphs are generated by the Georgia Tech Internetwork Topology Generator (GT-ITM) [2]. We construct ten different graphs, each consisting of 200 nodes. Each graph includes two transit domains with four transit nodes each. Three stub domains were connected to each of the transit domain nodes and each stub domain, on the average, has eight nodes. The α values of the transit and stub domains are set to 0.2 which controls the connectivity in a transit or stub domain. The delay of edges is between 1 to 100, with an average of 20. Shown in each figure below, all the results are averaged over the ten graphs. In each instance, a random node is assigned as the source, which remains the same for different placement and routing algorithms. The number of proxies ranges from 1 to 40.

6.1 Routing algorithms

We first evaluate multicast trees built by various routing algorithms, in terms of three performance metrics D, B, and *WSDB*. We select two MPP algorithms, Random and MIPP, to generate placement for this experiment. Four routing algorithms, SPT, MST, P-D, and DDB, are evaluated. The SPT algorithm provides the shortest delay and the MST algorithm the lowest cost. The P-D and DDB algorithms compromise between the delay and the cost. The DDB algorithm adjusts the λ value to control the increase of delay when minimizing the cost. Figure 10 shows the relationship between the delay and the cost when adjusting the λ value. Figure 10(a) is for the random placement of proxies and Figure 10(b) is for the MIPP placement. For the same delay, the bandwidth consumption of random placement is much higher than that of MIPP. The setting of the threshold of maximum delay is significant for the random placement but not for MIPP. For a threshold of 10%, λ is 16 for random placement and 2 for MIPP placement. For a threshold of 50%, λ can be as small as 0.2 for both placements.

Figure 11 illustrates D, B, and *WSDB* for the given Random and MIPP placements, respectively. The λ value is set to be 1 so the performance of different routing algorithms can be compared. Evidently, the MIPP placement on the right has produced a better placement than the Random placement on the left. With a good proxy placement, the difference between four routing algorithms is not significant. The difference of the *WSDB* value is within 8%. Given a Random placement, good routing algorithms can help



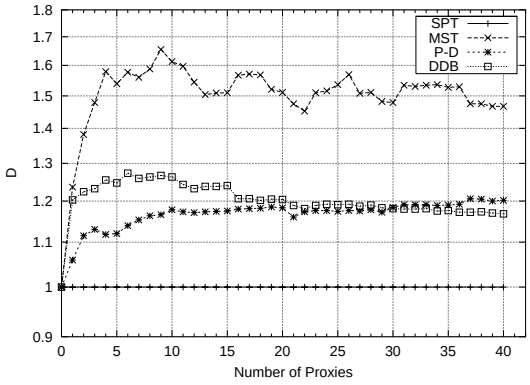
(a) Random Placement

(b) MIPP Placement

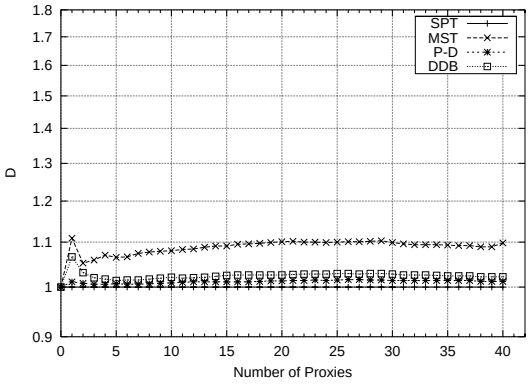
Figure 10: The relationship between delay and cost when adjusting λ.

to some extent. SPT routing has the minimal delay and the highest bandwidth consumption. MST routing consumes the least bandwidth but has the highest delay. SPT results in a higher bandwidth consumption, particularly for the Random placement, where it consumes up to 120% more bandwidth than MST does. The bandwidth consumption of DDB routing is only slightly higher than MST routing. DDB exhibits a much better performance than SPT. The difference of the *WSDB* value can be as large as 80%. DDB is the best in terms of its *WSDB* value which is always the lowest. However, it is the slowest algorithm among the four routing algorithms since it recomputes the metric every iteration. P-D performs slightly worse than DDB with a larger *WSDB* value.

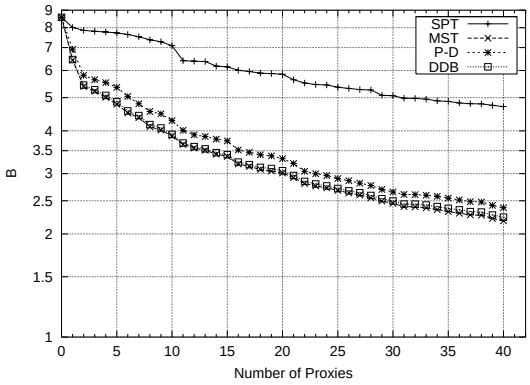
Considering delay only, SPT is obviously superior to the others by its definition, and it is independent as to where proxies are placed. Other routings exhibit an increased delay in general, however, the increase in delay should be kept minimal when reducing the bandwidth consumption. An inappropriate placement, such as the Random placement shown in the left column, together with MST routing can lead to up to 67% increase in the aggregate delay. For details, Figure 12 shows the cumulative distribution of delay when 20 proxies are placed. The X-axis represents the increase of relative delay and the Y-axis the percentage of individual nodes whose relative delay were less than the X-value. The Random placement and MIPP placement behave substantially different in terms of delay. For MST, more than 90% of the nodes have less than 17% delay increase with the MIPP placement, but most nodes have more than 50% delay increase with the Random placement. MST certainly cannot tolerate an inappropriate placement. In contrast, the DDB algorithm is less sensitive to different placements, where with the MIPP placement, more than 90% of the nodes have less than a 8% delay increase; and with the Random placement, more than 90% of the nodes have less than a 30% delay increase. An important observation is that, with a proper proxy



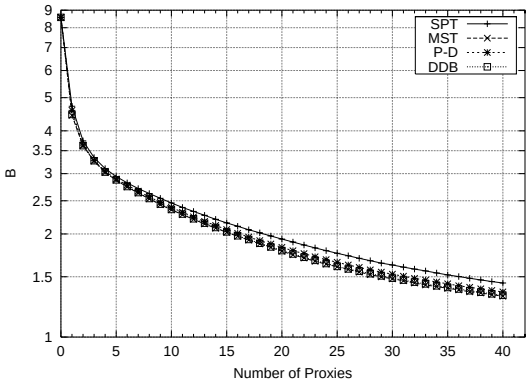
(a) Delay with Random Placement



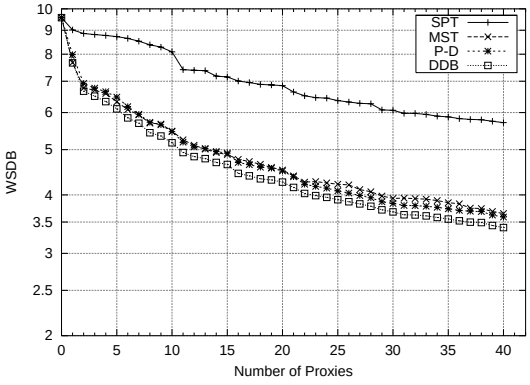
(b) Delay with MIPP Placement



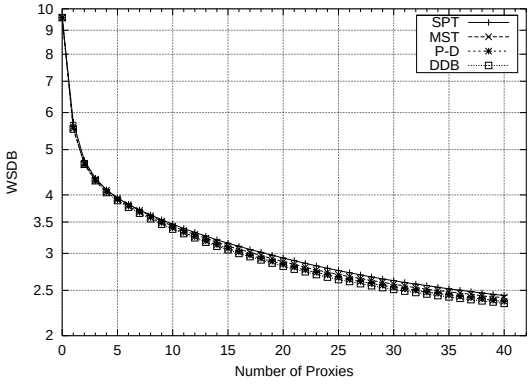
(c) Bandwidth Consumption with Random Placement



(d) Bandwidth Consumption with MIPP Placement

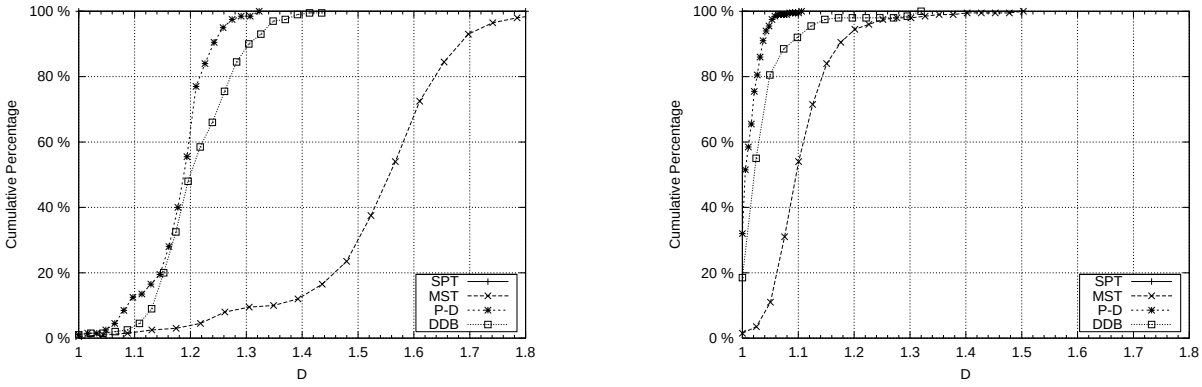


(e) WSDB with Random Placement



(f) WSDB with MIPP Placement

Figure 11: D, B and *WSDB* Comparison of Routing Algorithms.



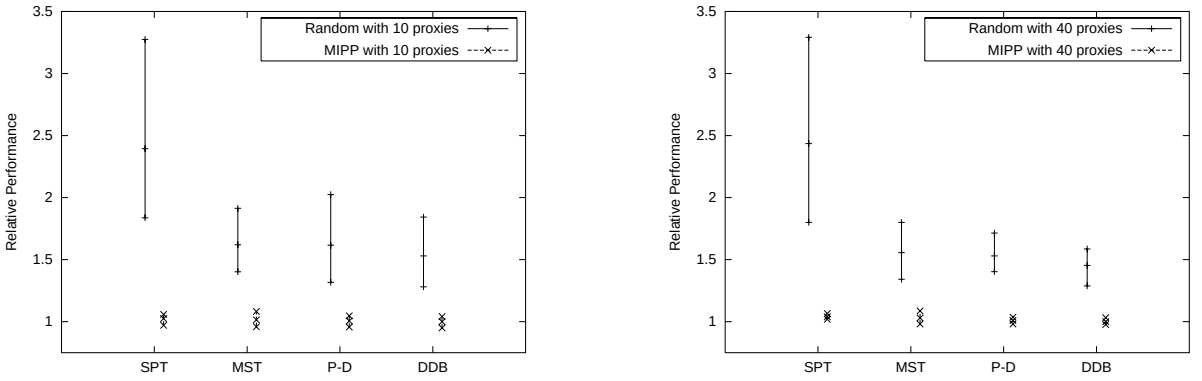
(a) Random Placement

(b) MIPP Placement

Figure 12: Cumulative Delay Percentage from Routing Algorithms.

placement, the increase of delay is minimal, in most situations within 10%. Thus, a "bounded delay" routing algorithm might not be necessary since the delay increase is not significant.

In Figure 13, all routing algorithms are compared with reference to the best case, MIPP placement with DDB routing. Here, the relative performance is defined as the ratio of the corresponding *WSDB* value to the best *WSDB* value across all routing and placement algorithms with the same number of proxies placed. The error-bars in Figure 13 correspond to the maximum, average, and minimum, respectively, of the relative performance of the routing algorithms. An evident observation is that with a good placement, such as MIPP, all four routing algorithms perform well. With an inappropriate placement, such as Random



(a) P = 10

(b) P = 40

Figure 13: Relative Performance of Routing Algorithms.

placement, performance of different routing algorithms is substantially divergent and much worse. Hence, the proxy placement has its essential importance.

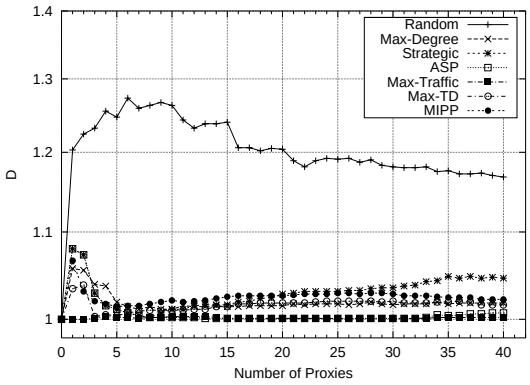
6.2 Placement Algorithms

Here, the multicast trees resulting from various placement algorithms are compared. After the proxy placement is generated, two routing algorithms, SPT and DDB, are used to construct the multicast trees. The resultant D, B, and *WSDB* metrics are shown in Figure 14.

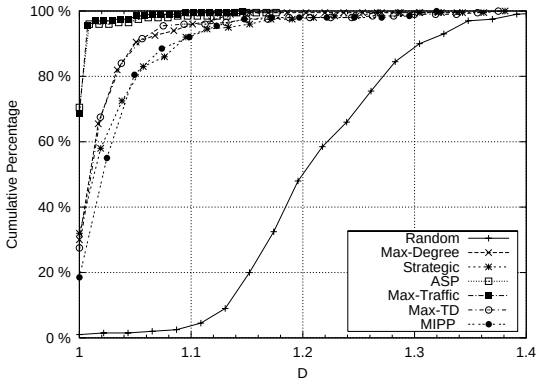
The delay for SPT routing is invariant, so only the delay for DDB is shown in Figure 14(a). The delay of the Random placement is much worse than others. The multicast trees resulting from the Max-Traffic and ASP placement algorithms show the lowest delay. These placement algorithms, except the Random placement, have delay increases of no more than 10%. Figure 14(b) shows the cumulative distribution of relative delay when 20 proxies are placed. The X-axis represents the increase of relative delay and the Y-axis the percentage of individual nodes whose relative delay were less than the X-value.

Figures 14(c) and (d) show the bandwidth consumption. Most placement algorithms significantly reduce the bandwidth consumption when the first five proxies are placed. The bandwidth consumption with five proxies is about two times that of 40 proxies. The Random placement with SPT routing shows large bandwidth consumption. The Max-Traffic placement algorithm exhibits higher bandwidth consumption when a few proxies are placed. It outperforms the Max-Degree placement algorithm as more proxies are placed. The Strategic placement and ASP are the same with eight proxies since there are eight transit nodes in the graphs. For other numbers of proxies, ASP outperforms the Strategy placement. The Max-TD placement is exactly the same as MIPP with the SPT routing since the SPT tree is invariant for different numbers of proxies. It is very close to MIPP with the DDB routing.

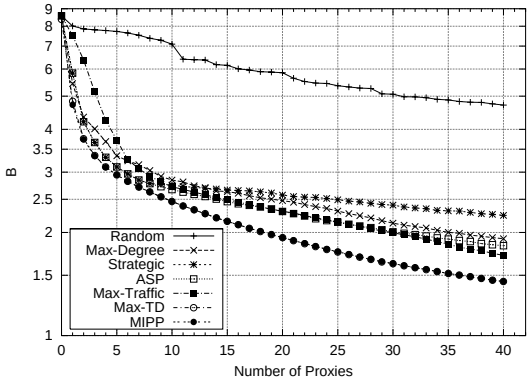
The *WSDB* results are illustrated in Figures 14(e) and (f). Since the delay does not change significantly for most placement algorithms, these curves are similar to bandwidth consumption curves. The MIPP placement produces the best performance, as expected. The Random placement does not produce satisfactory results when SPT routing is used. However, DDB routing can improve the performance of the Random placement, but the *WSDB* value of the Random placement is about 40% larger than that of the MIPP placement. Again, the *WSDB* value of Max-TD is very close to that of the MIPP algorithm. Since Max-TD is faster, it can serve in place of the MIPP algorithm. In Figure 15, all placement algorithms are compared with reference to the best case, MIPP placement with DDB routing. Here, the relative performance is defined as the ratio of the corresponding *WSDB* value to the best *WSDB* value across all routing and placement algorithms with the same number of proxies placed. The error-bars correspond to the maximum, average, and minimum. The Random placement is much worse than other algorithms while Max-TD and MIPP perform better than others. The routing algorithms have made a significant impact in Random placement but not for others.



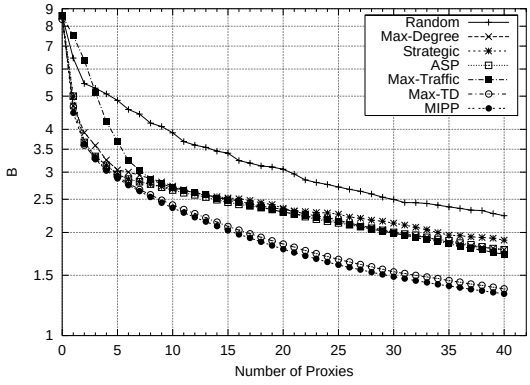
(a) Delay with DDB Routing



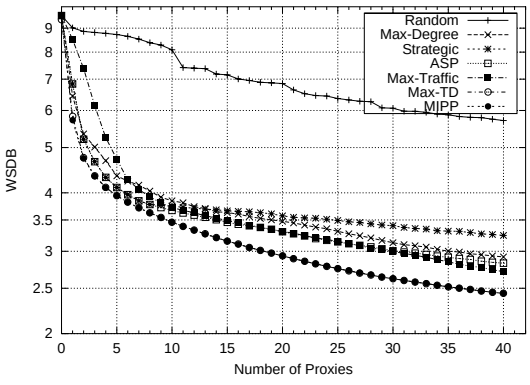
(b) Cumulative Delay Distribution with DDB



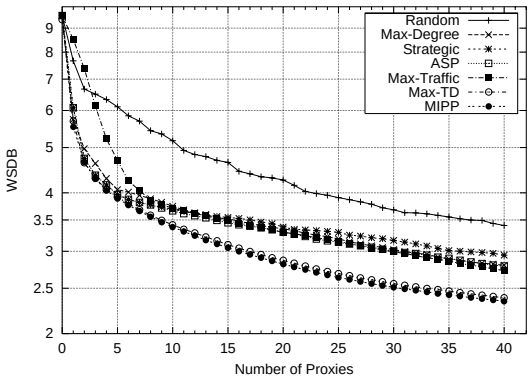
(c) Bandwidth Consumption with SPT Routing



(d) Bandwidth Consumption with DDB Routing

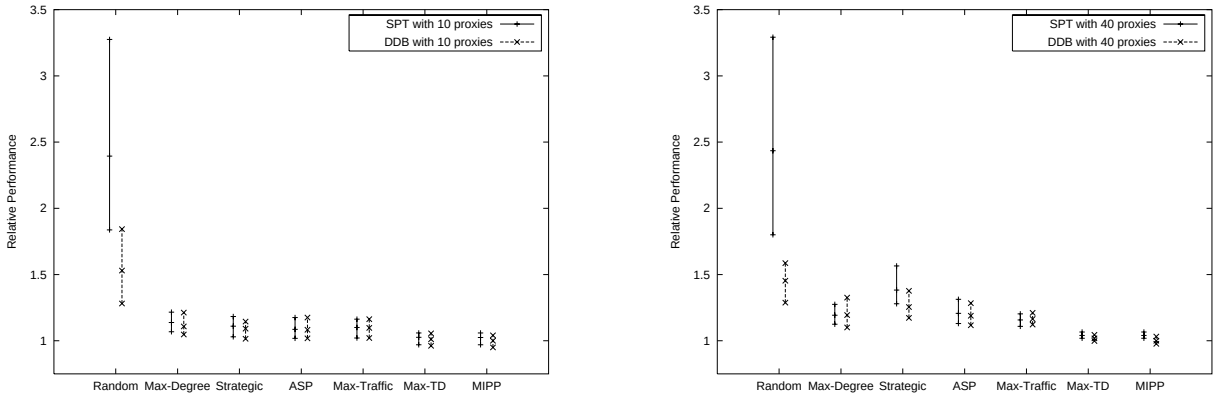


(e) WSDB with SPT Routing



(f) WSDB with DDB Routing

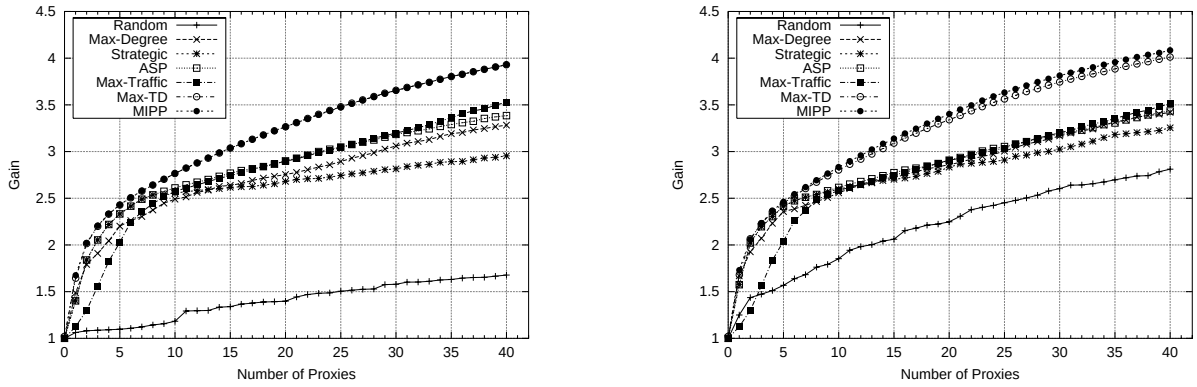
Figure 14: D, B and *WSDB* Comparison of Placement Algorithms.



(a) P = 10 **(b)** P = 40

Figure 15: Relative Performance of Placement Algorithms.

The benefit of placing proxies can be measured by the *gain*, which is defined as the ratio of the *WSDB* value with no proxy placed to the *WSDB* value with *P* proxies placed. Figure 16 illustrates the gain of various placement algorithms with SPT and DDB routing. The gain of Random placement is very low, especially for the SPT routing. The popular Strategic placement performs well for a few proxies, but not as good as other placement algorithms for more proxies. MIPP and Max-TD produce the highest gains.



(a) Gain with SPT Routing **(b)** Gain with DDB Routing

Figure 16: Gain with Proxy Placement.

Most existing schemes of multicasting, especially IP multicast, use strategic placement with SPT routing. It is a practical approach and simple to implement. Compared strategic placement with SPT to the best scheme, MIPP placement with DDB routing, it can be found that the bandwidth consumption of the former can be 80% larger. Furthermore, many multicast overlays [6, 7] do not implement true SPT due

to practical considerations, so the delay can increase significantly [41]. MIPP placement may become practical with the zone approach [19]. With a good proxy placement, even the simplest SPT routing can generate satisfactory performance.

Local Search

The MIPP placement algorithm produces satisfactory results but with a high complexity. Max-TD can produce good performance with a moderate complexity. Max-Degree, Max-Traffic, Strategic and ASP are fast, but their performance is not as good as Max-TD and MIPP. The Random placement is not satisfactory. However, with local search, their performance can be significantly improved. Figure 17 shows improvement of Random, Max-Degree, and Max-Traffic placements with local search. An improvement of 12% to 37% can be obtained in 300 iterations. The local search algorithm has also been applied to MIPP and Max-TD but improvement is limited, possibly due to the fact that their placement is quite close to the optimal.

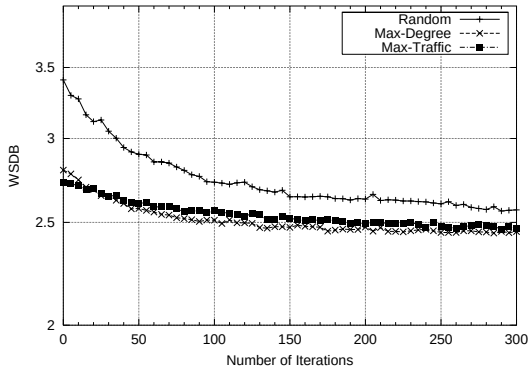


Figure 17: Local search for Random, Max-Degree, and Max-Traffic.

Performance on Large Graphs

The performances shown so far are for graphs of 200 nodes. Here, graphs of 1,000 nodes are tested. Again, the network graphs are generated by GT-ITM. Ten graphs, each consisting of 1,000 nodes are constructed. The large graph is made up of ten transit domains. All the other parameters are the same. Figure 18 shows *WSDB* values for various placement algorithms by taking the average of the ten graphs. Comparing this figure and Figures 14 (f), the curves are similar. Thus, the results from the graphs of 200 nodes can be extended to graphs of 1,000 nodes.

Power-Law Graphs and BGP Graphs

The *WSDB* values for the power-law graph are shown in Figure 19(a). The power-law graph is gen-

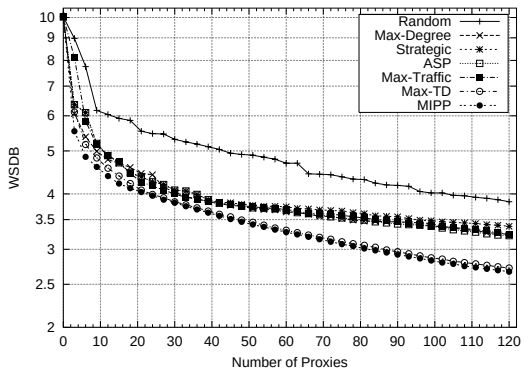
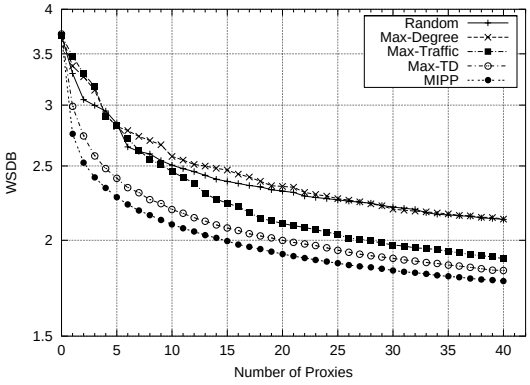


Figure 18: *WSDB* for Graphs of 1,000 Nodes with DDB routing.

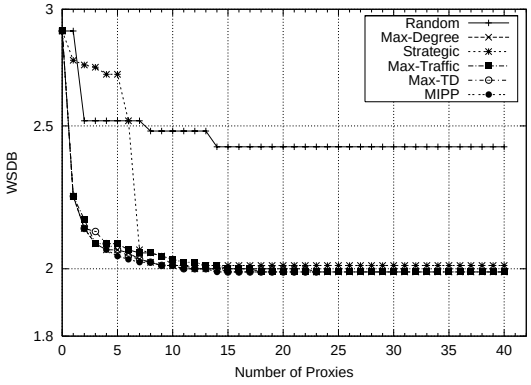
erated by *Router Waxman* in BRITE [26]. It is a random topology using Waxman's probability model to connect the nodes, which is $P(u,v) = \alpha * e^{-d/\beta * L}$, where, d is the distance from node u to v, and L is the maximum distance between any two nodes. α and β are parameters set between zero and one. The strategic and ASP algorithms are designed for transit-stub graphs and not directly applicable here. Different from the transit-stub graph, the power-law graph has higher connectivity. As a consequence, the *WSDB* value is lower in general. With more proxies placed, the *WSDB* values are even less than 2 because more alternative paths make bandwidth consumption B less than 1. Furthermore, the Random placement performs better. The difference between Max-TD and MIPP becomes larger, therefore, MIPP is able to produce better placement in different situations.

Figure 19(b) shows the performance of BGP graphs with $N = 1,000$. This model reflects the backbone connectivity in the AS level. The information was collected from the routing tables with BGP connections to multiple geographically-distributed routers [27]. We have implemented a strategic algorithm for BGP graphs, which places proxies randomly to high-degree BGP nodes. The strategic algorithm does not produce good performance for the first five proxies but performs well afterwards. The connectivity of BGP graphs is not as high as the power-law graphs, so ten proxies placed in right locations can lead to almost optimal performance. Random placement does not perform well for the BGP graph. In general, the less the connectivity, the worse the Random placement. On the other hand, the less the connectivity, the less the difference between the placement schemes except the Random placement. It is because most placement algorithms can find the right locations in a low-connectivity graph.

A more realistic setting in the transit-stub graph with different number of receivers is shown in Figure 20. Here, we assume that no receiver is attached to the transit nodes. In the graph of 200 nodes, there are eight transit nodes and 192 stub nodes. Shown in Figure 20(a) are *WSDB* values for 192 receivers, each receiver is randomly assigned to a stub node. Therefore, some nodes have one or more receivers and the other 78 nodes (including 8 transit nodes and 70 stub nodes) have no receiver attached. Figure 20(b)



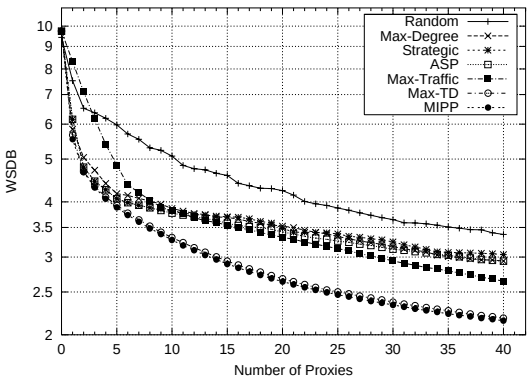
(a) PowerLaw Graphs



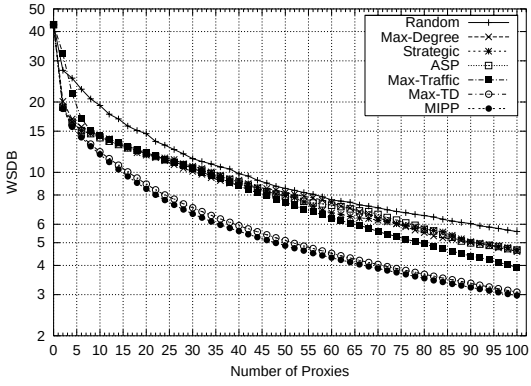
(b) BGP Graphs

Figure 19: *WSDB* for Power-law Graphs and BGP Graphs with DDB routing.

shows the *WSDB* values for 960 receivers. All stub nodes have at least one receiver except the eight transit nodes. On a node with zero weight, though there is no receiver, a proxy might be placed if it will help to replicate streams for other receivers. The performance of 192 receivers is similar to Figure 14(f), where the gain of MIPP with 40 proxies is 3.9 and WSDB is 2.4. With 960 receivers, the gain of MIPP increases to 8.1 with 40 proxies, and increases to 14.5 with 100 proxies. The *WSDB* values are 5.6 with 40 proxies and 3.0 for 100 proxies.



(a) 192 Receivers



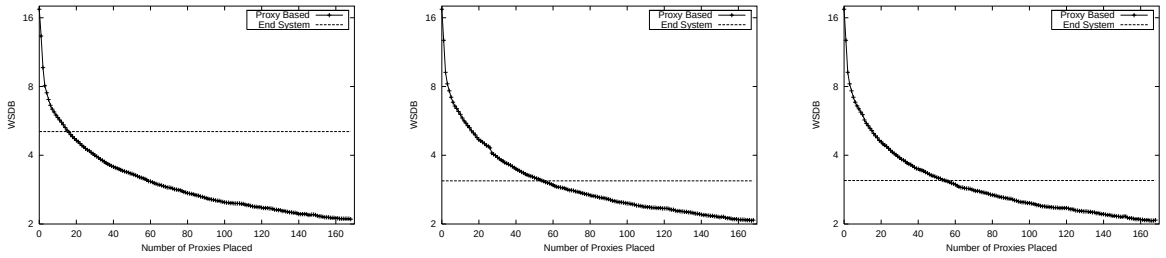
(b) 960 Receivers

Figure 20: *WSDB* for different number of receivers with DDB routing.

In the following three subsections, we will present a set of experiments to show the difference between the end-system based multicast and proxy-based multicast, the placement adjustment for dynamic traffic, and the performance with limited proxy capacity.

6.3 Comparison of End-system based and Proxy-based Multicast

In this simulation topology, 168 nodes are stub nodes and the other 32 nodes are transit or bridge nodes. A bridge node is a node in a stub domain that connects to a transit domain. We compare the performance of the proxy-based system over the pure end-system case. In a pure end-system, all the stub nodes are enabled to be replication engines, whereas in a proxy-based system, a proxy can be placed onto any node. In Figure 21, performance of a pure end-system, where all of the 168 stub nodes are replication nodes, is illustrated by a horizontal line. Performance of a proxy-based system varies as the number of proxies increases. An end-system with SPT, as built by many popular routing algorithms, is not efficient compared to that with MST or HOT. With SPT, a client has a smaller chance to connect to its peer stub nodes than that with MST or HOT. A proxy-based system shows a better performance since a client can connect to a proxy at a bridge or transit node. Thus, with SPT, after 16 proxies are placed, the performance matches with the pure end-system case. With MST or HOT, about 55 proxies match with the pure end-system performance. As more proxies are placed, the performance continues improving. By intuition, the proxy-based multicast is more efficient than the end-system multicast since the proxies can be placed in more strategically important locations. However, adding a replication proxy is more expensive than an end host.



(a) SPT **(b)** MST **(c)** HOT

Figure 21: Comparison of proxy-based multicast and end-system multicast.

6.4 Placement and its adjustment

A long term simulation has been conducted with dynamic traffic patterns. The topology is the same as before. Among the 200 nodes, twenty nodes are randomly chosen to be candidates of source. The number of streams issued by each node is determined by the power-law, so is the multicast group size. The stream starting time follows the Poisson arriving model and the stream duration follows the normal distribution with an average of 120 minutes and standard deviation of 20. Receivers join and leave at runtime, whose arrival patterns follow the Poisson distribution and whose duration is evenly distributed. Overall, the stream arrive-rate is, by average, two streams every ten minutes, but also changes during the peak and

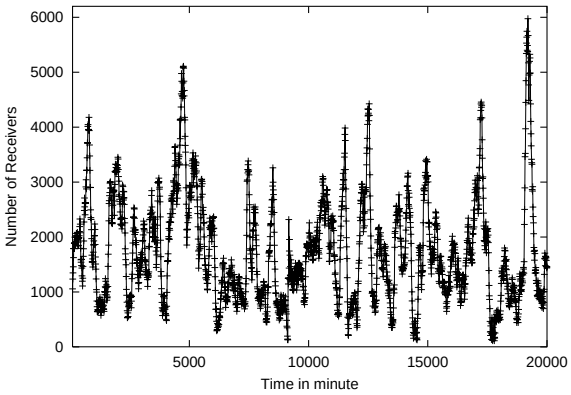


Figure 22: A sample generated traffic: the number of receivers being serviced

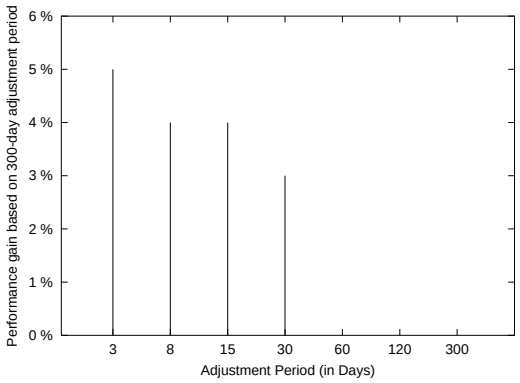


Figure 23: Performance with different adjustment periods of proxy replacement.

non-peak intervals in a day. Figure 22 shows a randomly generated traffic during about 15 days. As illustrated, the generated traffic varies in a wide range and closely represents the reality, based on which, our simulation is conducted and results are obtained. The simulation is running for 300 days. During these days, how often does the placement of proxies need adjustment? A more frequent adjustment could lead to more improvement in performance, but may incur unnecessary overhead. It is in general a trade-off problem. An experiment has been conducted to study the length of the adjustment period. Figure 23 shows the performance gain relative to the performance with the 300-days adjustment period. We select the adjustment period of 30 days as the *time period*. An adaptive algorithm is used to dynamically replace the proxies in the adjustment phase. That is, the proxy placement is adjusted at the beginning of every period, based on the traffic pattern of the previous time period (PTP).

Figure 24 shows the bandwidth consumption in each period. The results of the Max-Traffic and Max-TD are shown, each of them uses the average traffic to determine the proxy placement. The bandwidth

consumption of Max-Degree is also shown as a reference as the Max-Degree placement is independent of the traffic. It can be seen that the Max-Traffic algorithm improves the performance in all cases. The Max-TD algorithm reduces the bandwidth consumption to more than half. Figure 25 shows the same set of simulations as Figure 24 but assuming an ideal case that the traffic pattern for the next time period is known in prior. This assumption of current time period (CTP) is used to show how much improvement could be possible. We found the prediction of traffic pattern based on the previous time period can be acceptable and works well, as Figures 24 and 25 do not differentiate considerably. Figure 26 shows the number of proxies that are replaced in each time period. Table 3 shows the average bandwidth consumption of ten time periods and Table 4 is the average number of proxies that are replaced. It can be seen that Max-Traffic is slightly better than Max-Degree, which is static. Max-TD performs much better and its replacement cost is lower than that of Max-Traffic. Thus, with the Max-TD algorithm, the system is more stable and fewer proxies are replaced each time.

Table 3: Average bandwidth consumption of ten time periods

	PTP	CTP
Max-TD	1.402	1.403
Max-T	1.534	1.529
Max-D	1.582	

Table 4: Average number of proxies replaced of ten time periods

Max-TD	Max-T	Max-D
1.7	12.7	0

6.5 Proxy Capacity Constraint

The overlay multicast, implemented at the network application layer, can be more adaptive and responsive to dynamic changes of the Internet traffic. In order to sustain these advantages, the capacity constraint of an individual proxy must be incorporated into the system design. In general, the proxy placement and dynamic subscription encourage deployment of proxies with high capacity when and where as needed. However, every proxy has a limited capacity. Its cost, feasibility, as well as its limitation imposed by the underlying network infrastructure have to be taken into consideration. Here, we briefly discuss the impact of limited proxy capacity.

We have designed a capacity-aware algorithm by modifying the distributed algorithm described in [3] to build a capacity constraint SPT. Initially, every proxy forms a fragment by its own, where the fragments

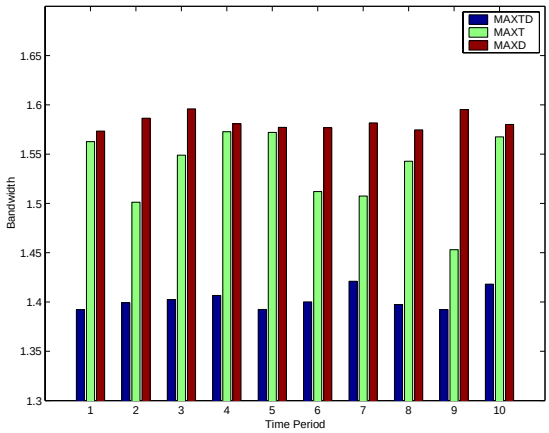


Figure 24: Simulation of ten time periods with traffic prediction.

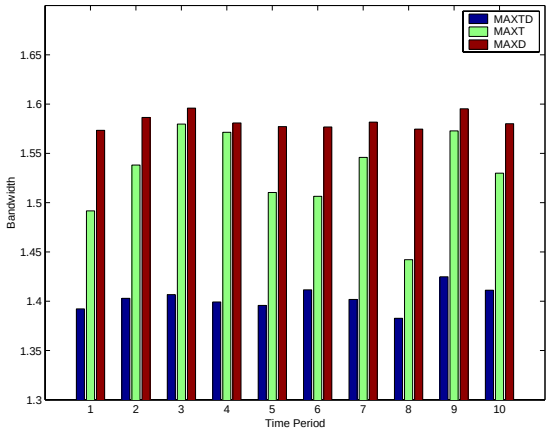


Figure 25: Simulation of ten time periods assuming the traffic is known priorly.

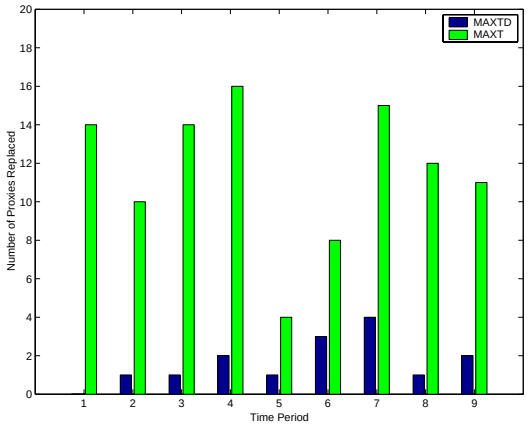


Figure 26: The number of proxies that are replaced for each time period.

will be extended or merged during processing. After the *init* state, the proxy keeps sending *request* to the source fragment. Once *accept*ed by the source fragment, the *connect* operation is conducted. Meanwhile, the source fragment performs a *discovery* step to search for the closest fragment to merge with without violating the capacity constraint. Finally, a capacity-aware SPT is formed. Details of the algorithm can be found in [40].

In the following simulation, the proxy capacity is randomly generated and measured in terms of its bandwidth. The constraints of the proxy capacity is reinforced by setting the proxy delay as a function of available capacity. The proxy delay can largely increase when the proxy is heavily loaded and therefore its remaining capacity becomes limited. That is, when congestion happens due to the limited proxy capacity, the increased proxy delay would encourage the multicast stream being routed to an alternative path. Figure 27 shows the impacts in performance for three cases: case 1) every proxy has an infinite capacity; case 2) each proxy has a limited capacity, the multicast overlay is constructed without incorporating capacity constraints; and case 3) each proxy has a limited capacity, the multicast overlay is constructed by the capacity-aware algorithm. It can be seen that the delay may increase significantly when the proxy capacity is limited, however, a capacity-aware algorithm can minimize its impact.

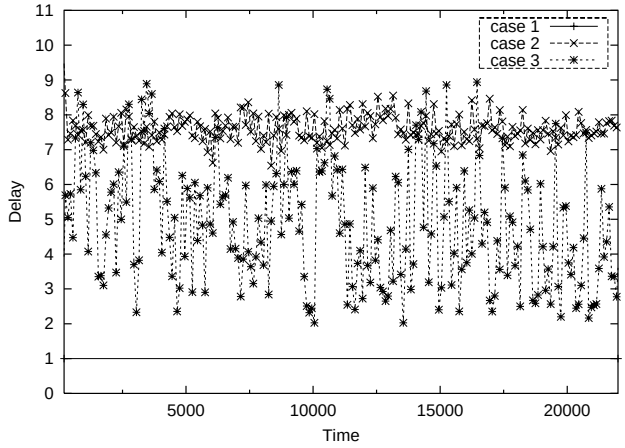


Figure 27: Comparison of case 1) every proxy has an infinite capacity; case 2) each proxy has a limited capacity, the multicast overlay is constructed without incorporating capacity constraints; and case 3) each proxy has a limited capacity, the multicast overlay is constructed by the capacity-aware algorithm.

7 Related Work

Compared to the IP multicast, the proxy-based multicast may solve the congestion control problem on heterogeneous networks. Retransmission may be handled locally for reliable multicast. The proxy-based

multicast may also work across both space and time by using proxy servers as caches [38, 36, 35]. Our work has been informed and inspired by a variety of pioneering works on overlay networks [6, 7, 17, 28], which focus on various scopes and requirements. Many of them were designed for distributed implementation, therefore, not necessarily looking for the optimal. Our work has been targeted at a thorough understanding of both multicast routing and proxy placement problems. Such a study can help us to evaluate various implementations and their optimality. Furthermore, a better understanding of the problem in general can benefit many people in practical design, implementation, and deployment in the near future.

Existing works on the proxy-based multicast use the strategic placement and various routing algorithms to build multicast trees. A comparison study showed that these routing algorithms normally involve a 200% to 300% longer delay than IP multicast [41], and sometimes are unable to reduce the bandwidth consumption at all. Our work has investigated both routing and placement problems in a systematic approach. Various algorithms and their system integration are discussed and evaluated to illustrate their performance impacts. These results can be valuable references for other proxy-based multicast overlays. Moreover, with the power of localization, these optimal routing and placement algorithms become feasible [5, 23].

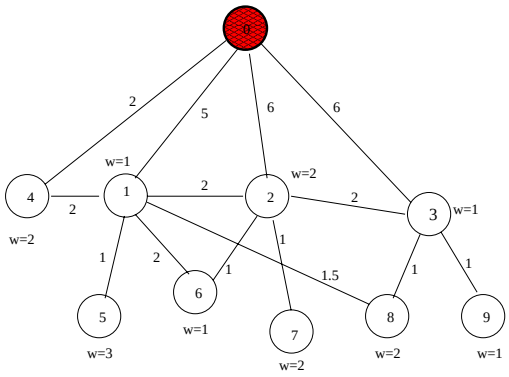
Formulation and evaluation of the Multicast Proxy Placement (MPP) problem is a focal point of this paper, which to our knowledge is the first work formulating the MPP problem. Recently, another work [30] considered the proxy placement problem in overlay networks. Its objective function is to minimize the number of proxies that cover the entire network with a bounded delay from clients to a proxy. However, since the delay from the source to proxies is unbounded, there could be a long delay between the source and clients. Furthermore, the bandwidth requirement was not considered in that work.

Our work carries many ideas from the works of placing cache replicas or proxies [29, 25, 24, 20, 18]. The cache placement problem has been formulated as the two well-known problems: the facility location problem [8] and the *k*-median problem [4]. Though the multicast proxy placement and the cache proxy placement share some similarities, these problems have different requirements in other places. The cache proxy placement problem assumes that contents are cached in the cache proxies and may be accessed at different times. The MPP problem assumes that the content is distributed from the source to many users at the same time. The bandwidth requirement must be minimized in a multicast network. The delay time in a cache network depends on the distance between the client and its nearest cache proxy holding the desired content; whereas, in a multicast network, it depends on the distance between every client and the source. Thus, when a client connected to the nearest proxy as in cache proxy routing, it may have a longer delay from the original server. These different requirements result in different routing and placement strategies. Therefore, the optimal placement of multicast proxies may be different from that of cache proxies. In Figure 28, we use a small sample graph G_2 with $N = 10$ to show the difference between the optimal cache proxy placement and optimal multicast proxy placement. Figure 29 shows the optimal placement of cache

proxies and multicast proxies. The cache placement problem can be modeled as the minimum k-median problem [29]. Its objective function is the system cost

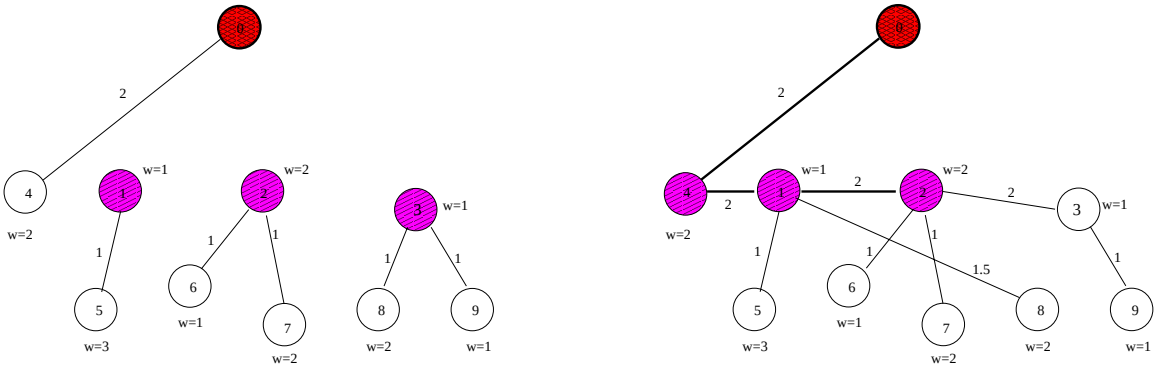
$$C = \sum_{i=1}^{N} w(n_i) \times d(n_i, c(n_i, P)),$$

where $w(n_i)$ is the node weight representing the traffic generated by node n_i, $d(n_i, n_j)$ is the length of the shortest path from n_i to n_j, and $c(n_i, P)$ is the proxy in the proxy set P that is closest to n_i. Figure 29(a) shows optimal cache proxy placement for three proxies. The proxies are placed on nodes n_1, n_2, and n_3 and the system cost $C = 13$. On the other hand, the optimal multicast proxy placement that generates the



n_i	$w(i)$	$e(i,0)$	$e(i,1)$	$e(i,2)$	$e(i,3)$	$e(i,4)$	$e(i,5)$	$e(i,6)$	$e(i,7)$	$e(i,8)$	$e(i,9)$
n_0	1	-	5	6	6	2	-	-	-	-	-
n_1	1	5	-	2	-	2	1	2	-	1.5	-
n_2	2	6	2	-	2	-	-	1	1	-	-
n_3	1	6	-	2	-	-	-	-	-	1	1
n_4	2	2	2	-	-	-	-	-	-	-	-
n_5	3	-	1	-	-	-	-	-	-	-	-
n_6	1	-	2	1	-	-	-	-	-	-	-
n_7	2	-	-	1	-	-	-	-	-	-	-
n_8	2	-	1.5	-	1	-	-	-	-	-	-
n_9	1	-	-	-	1	-	-	-	-	-	-

Figure 28: n_i, $w(i)$, and $e(i, j)$ of a sample graph G_2



(a) Optimal cache proxy placement

(b) Optimal multicast proxy placement

Figure 29: Placement of Cache Proxy and Multicast Proxy.

smallest bandwidth consumption is shown in Figure 29(b). The proxies are placed on nodes n_1, n_2, and n_4 and the aggregate bandwidth consumption $B_M = 20$ and aggregate delay $D_M = 84$. Note that, $C = 14$ when proxies are placed on nodes n_1, n_2, and n_4. On the other hand, $B_M = 21$ and $D_M = 91$ when proxies are

placed on nodes n_1, n_2, and n_3. In summary, the MPP problem has a different objective function from that of the cache proxy placement problem. First, a multicast tree must be built among the replication nodes. Second, the non-replication nodes are not necessarily connected to the nearest replication node except that for the MST tree. It cannot be modeled by the minimum k-median problem in general. On the other hand, caching and multicasting share many common characteristics and their integration can easily gain benefits. In fact, it is attractive to use the same proxy to serve as both cache proxy and multicast proxy. Then, an interesting question is: where are the best locations to place proxies that are for both caching and multicast?

8 Conclusion

This paper addressed the Multicast Proxy Placement problem. The problem was formulated and objectives were presented. Based on the performance metrics, a number of routing and placement algorithms have been presented. The performance study showed that the proxy placement is more crucial than routing in terms of their impact on performance. From our study, important observations are listed below:

- With a good proxy placement, the difference between various routing algorithms is not significant. Within a zone, an optimal placement can be practical with the simple SPT routing.
- With the most common strategic placements, DDB routing can improve performance up to 20%.
- From our experiment, random placement is hardly acceptable. However, in some situations where proxies cannot be placed on proper locations, such as that in an end-system, a good routing algorithm such as DDB can improve the performance.
- The DDB routing algorithm generates a satisfactory result. The increase of delay and bandwidth consumption by DDB is within a few percents compared to that generated by SPT and MST, respectively. It balances well the requirements of low delay and low bandwidth consumption.
- The MIPP placement algorithm is the best, and Max-TD is very close to MIPP. A combination of Max-TD placement and DDB routing is able to produce satisfactory performance, with relatively low complexity. Compared to the popular combination of Strategic placement and SPT routing, an 80% gain in performance can be achieved.
- The Max-TD algorithm performs well for dynamically changing traffic. Only a few proxies need to be replaced each time period.

The routing and placement algorithms for a single multicast stream provide a basis for construction of overlay networks. More works are necessary to make multicast proxy placement as a practical approach.

First, a mechanism must be designed so that each zone has incentives to place their proxies. With the power of localization, optimal decision within a zone may result in global optimization in a federation of zones. Second, a method is to be investigated for dynamically changing traffic. Proxy subscription could be a promising approach. Third, routing and placement algorithms are to be designed for bounded proxy capacity with multiple multicast streams on top of the existing Internet traffic. Algorithms for self-organizing, self-diagnosing and self-healing overlays are also important. Another issue to be addressed is the distributed decision on the proxy placement. Distributed versions of routing and placement algorithms, which we are currently working on, may produce near-optimal results.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thorough comments.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[2] GT-ITM at Georgia Tech University. GT-ITM: Modeling topology of large internetworks. http://www.cc.gatech.edu/projects/gtitm/.

[3] Fred Bauer and Anujan Varma. Degree-constrained wultcasting in point-to-point netwroks. In *IEEE Infocom*, 1995.

[4] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *IEEE 40th Annual Symposium on Foundations of Computer Science*, October 1999.

[5] Y. Chawathe and M. Seshadri. Broadcast federation: An application-layer broadcast internetwork. In *the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2002.

[6] Yatin Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, Department of EECS, UC Berkeley, December 2000.

[7] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *ACM Sigmetrics*, 2000.

[8] F.A. Chudak and D. Shmoys. Improved approximation algorithms for capacitated facility location problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[9] S. Deering. *Multicast routing in a datagram internetwork*. PhD thesis, Stanford University, 1991.

[10] S. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2):85–111, May 1990.

[11] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Networks*, pages 78–88, January 2000.

[12] M. Doar and I. Leslie. How bad is naive multicast routing? In *INFOCOM*, April 1993.

[13] Paul Francis. Yoid: Your own internet distribution. Technical Report at www.aciri.org/yoid, UC Berkeley ACIRI Tech Report, April 2000.

[14] M.R. Gary and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[15] David Helder and Sugih Jamin. Banana tree protocol, an end-host multicast protocol. Technical Report CSE-TR-429-00, Univ. of Michigan, 2000.

[16] H. Holbrok and D. Cheriton. IP multicast channels: EXPRESS support for large-scale single source applications. In *ACM SIGCOMM*, September 1999.

[17] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable multicasting with an overlay network. In *5th Symposium on Operating System Design and Implementation (OSDI)*, December 2000.

[18] X. Jia, D. Li, X. Hu, and D.Z. Du. Placement of read-write web proxies on the internet. In *21st International Conference on Distributed Computing Systems*, 2001.

[19] Y. Jiang, M.Y. Wu, and W. Shu. A hierarchical overlay multicast network. In *IEEE International Conference on Multimedia and Expo (ICME2004)*, June 2004.

[20] K.M. Kamath, H.S. Bassali, R.B. Hosamani, and L.Gao. Policy-aware algorithms for proxy placement in the internet. In *ITCOM*, 2001.

[21] A. Kershenbaum, P. Kermani, and G. Grover. Mentor: An algorithm for mesh network topological approximation and routing. *IEEE Trans. on Communications*, 39:503–513, 1991.

[22] S. Khuller, B. Raghavachari, and N.E. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–322, 1995.

[23] Christopher Kommareddy. Scalable application layer multicast. In *ACM SIGCOMM*, August 2002.

[24] P. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transaction on Networking*, 8:568–582, October 2000.

[25] B. Li, M.J. Golin, G.F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. In *IEEE Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, 1999.

[26] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Universal topology generation from a user's perspective. Technical Report BUCS-TR-2001-03, Computer Science Department, Boston University, April 2001.

[27] National laboratory for applied network research. routing data. http://moat.nlanr.net/Routing/rawdata/.

[28] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *3rd Usenix Symposium on Internet Technologies and Systems (USITS)*, March 2001.

[29] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the placement of web server replicas. In *INFOCOM*, April 2001.

[30] S. Shi and J. Turner. Placing servers in overlay networks. In *Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, July 2002.

[31] R. Sosič and J. Gu. Local search for the satisfiability (SAT) problem. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(3):1108–1129, July 1993.

[32] Ion Stoica, T. S. Eugene Ng, and Hui Zhang. Reunite: A recursive unicast approach to multicast. In *INFOCOM*, 2000.

[33] Liming Wei and Deborah Estrin. A comparison of multicast trees and algorithms. In *INFOCOM*, April 1994.

[34] Liming Wei and Deborah Estrin. The trade-offs of multicast tree and algorithms. In *International Conference on Computer Communications and Networks*, August 1994.

[35] M.Y. Wu, S. Ma, and W. Shu. Scheduled video delivery for scalable on-demand service. In International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSS-DAV), May 2002.

[36] M.Y. Wu and W. Shu. Efficient support for interactive browsing operations in clustered CBR video servers. *IEEE Trans. on Multimedia*, 4(1), March 2002.

[37] M.Y. Wu and W. Shu. Proxy subscription for dynamic multicast overlays, 2003. submitted.

[38] M.Y. Wu and W. Shu. Video distribution with edge stations and Wi-Fi delivery networks. In *the IEEE International Conference on Multimedia and Expo (ICME2003)*, July 2003.

[39] M.Y. Wu, Y. Zhu, and W. Shu. Optimal multicast overlay placement for realtime streaming media. In *the IEEE Int'l Conference on Multimedia and Expo (ICME2004)*, June 2004.

[40] Y. Zhu. *Routing and Placement Problems in Proxy-based Multicast Overlay*. PhD thesis, Department of Electrical and Computer Engineering, University of New Mexico, 2004.

[41] Y. Zhu, W. Shu, and M.Y. Wu. Comparison study and evaluation of overlay multicast network. In *the IEEE International Conference on Multimedia and Expo (ICME2003)*, July 2003.