Constrained Mirror Placement on the Internet

Sugih Jamin
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
jamin@eecs.umich.edu

Cheng Jin
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
chengjin@eecs.umich.edu

Anthony R. Kurc
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
tkurc@eecs.umich.edu

Danny Raz
Bell Labs
Lucent Technologies
Holmdel, NJ 07733-3030
raz@research.bell-labs.com

Yuval Shavitt
Bell Labs
Lucent Technologies
Holmdel, NJ 07733-3030
shavitt@ieee.org

Abstract—**Internet Service Providers and infrastructural companies often employ mirrors of popular content to decrease client download time and server load. Due to the immense scale of the Internet and decentralized administration of the networks, companies have a limited number of sites (relative to the size of the Internet) where they can place mirrors. Mirrors of popular content are usually replicated on every site to maximize reachability to clients. In this paper, we study the performance improvements as the number of mirrors increases under different placement algorithms subject to the constraint that mirrors can be placed only at certain locations. Although there are extensive theoretical studies on center placement and, more recently, analytical and empirical studies on web cache placement, we are not aware of any published literature on mirror placement especially in the case of constrained mirror placement. Our results show that increasing the number of mirror sites under the constraint is effective in reducing client download time and reducing server load only for a surprisingly small range of values regardless of the mirror placement algorithm.**

I. Introduction

There is a growing number of frequently accessed web sites that employ mirror servers to increase the reliability and performance of their services. Mirror servers (or simply "mirrors") replicate the whole or the most popular content of a web server (the "server" henceforth). Clients requesting the server's content are redirected to the mirrors. Since each mirror sees only a portion of the total requests, clients can be served faster. Furthermore, if clients are redirected to mirrors closer to them than the server, download time can be reduced. In this paper, mirrors refer to locations instead of server machines; in other words, we consider co-located servers to be a single mirror.

At first glance, web caches serve the same purpose as mirrors. We differentiate mirrors from caches in that client access to a mirror always finds the requested content. A client is redirected to a mirror only when the mirror has the requested content. Mirrors can also serve some forms of dynamic content and content customized for each client.

There is a cost to keeping mirrors' content consistent when the content of the server changes. Various algorithms to keep web caches consistent have been proposed in the literature and are applicable to mirrors. We categorize these algorithms as based on time-to-live, e.g. [1], or server invalidation, e.g. [2]. Without going into the details of the algorithms, we note that the cost of keeping mirrors consistent, in terms of the amount of traffic seen at the server (in the case of [1]) or in the total amount of traffic seen on the network (in the case of [2]) increases linearly with the number of mirrors. Thus even if one assumes that larger number of mirrors provides further reduction in server load and client download time, simply increasing the number of mirrors with impunity will result in higher consistency cost. Certainly, one would be willing to pay the cost associated with larger number of mirrors if it is outweighed by the reduction in overall system cost. We show in this paper, however, that on Internet-like settings, increasing the number of mirrors beyond a certain number does not significantly reduce server load nor client download time. Obviously we are not considering the case where there is a mirror on every client host or LAN.

Given a finite number of mirrors, we are then interested in whether their placement on the Internet effects the overall system performance. In Section II we look at various mirror placement algorithms and heuristics. Ideally, a mirror can be placed where there is a large concentration of clients interested in the content of a server [3]. In this paper, however, we only consider a model in which there is a fixed number of candidate machines where mirrors can be placed. We call this the Constrained Mirror Placement (CMP) problem. An ISP (Internet Service Provider) or an Internet infrastructure company, for instance, may have a large number of machines scattered around the Internet capable of hosting mirrors. A content provider with a busy web server can rent resources on these machines to host their mirrors. The question is thus: on which subset of the candidate machines shall a content provider put mirrors of its content?

Before we look at the effects of increasing the number of mirrors and of the placement algorithms, we first present a more formal definition of the CMP problem in the next section.

II. Constrained Mirror Placement

We model the Internet as a graph, $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ the set of links. We define $\mathcal{H} \subseteq V$ to be the set of candidate hosts where mirrors can be placed, $\mathcal{M} \subseteq \mathcal{H}$ the set of mirrors of a particular server S, and $\mathcal{B} \subseteq V$ the set of S's clients. The objective of the CMP problem is to place the set of mirrors on the set of candidate hosts such that some optimization condition $\mathcal{O}(\mathcal{M}, p)$ (defined below) is satisfied for the client set. How well the optimization condition is satisfied depends on the sizes and topological placements of the candidate host and clients. We denote the sizes of the candidate host, mirror, and client sets as $|\mathcal{H}|$, $|\mathcal{M}|$, and $|\mathcal{B}|$, and their topological placements as $\mathcal{P}(\mathcal{H})$, $\mathcal{P}(\mathcal{M})$, and $\mathcal{P}(\mathcal{B})$ respectively. We use the notation $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{B}$ to denote a specific size and placement of the sets. We only consider the case where $|\mathcal{M}| \leq |\mathcal{H}|$.

This project is funded in part by NSF grant number ANI-9876541. Sugih Jamin is further supported by the NSF CAREER Award ANI-9734145 and the Presidential Early Career Award for Scientists and Engineers (PECASE) 1998. Additional funding is provided by MCI Worldcom, Lucent Bell-Labs, and Fujitsu Laboratories America, and by equipment grants from Sun Microsystems Inc. and Compaq Corp.

Danny Raz is currently with the Computer Science Department, Technion, Technion City, Haifa 32000, Israel. http://www.cs.technion.ac.il/~danny

Yuval Shavitt is currently with the Department of Electrical Engineering–Systems, Tel Aviv University, Tel Aviv 69978, Israel

In this paper, we study the effect of changing $|\mathcal{M}|$ and $\mathcal{P}(\mathcal{M})$ while holding $|\mathcal{H}|$ constant, with $\mathcal{H} \cap \mathcal{B} = \emptyset$, and $\mathcal{H} \cup \mathcal{B} \subseteq V$. We experiment with uniformly distributed $\mathcal{P}(\mathcal{H})$ and by placing candidate hosts on nodes with the highest outdegrees (outgoing links) first. We also experiment with both uniformly distributed and trace-based $\mathcal{P}(\mathcal{B})$.

A. Optimization Condition $\mathcal{O}(\mathcal{M}, p)$

We identify two goals commonly associated with placing mirrors on the Internet: reducing client download time and alleviating server load. In the previous section we presented the cost of keeping mirrors consistent as a limiting factor in deploying large number of mirrors. We further stated that even discounting consistency cost, we will show in this paper that increasing the number of mirrors beyond a certain number does not significantly reduce server load nor client download time. Hence, for the remainder of this paper, we will assume zero cost to keep mirrors consistent. With zero consistency cost, we can treat the server itself as simply one of the mirrors. Assuming one can add a mirror with no cost, we ask by how much does adding one more mirror reduce client download time and alleviate load at existing mirrors (including the server). Client download time is effected by load at mirrors and network latency (in terms of round-trip time).[1] Hence we can rephrase the two goals of mirror placement as reducing round-trip time between a client and the closest mirror and alleviating load at mirrors.

One way to alleviate load at mirrors is to run a load balancing algorithm by which clients are directed to the mirror with the least load [4]. In this paper, we take a different approach and consider reducing round-trip time as our *sole* optimization condition, $\mathcal{O}(\mathcal{M}, p)$. A heavily loaded mirror can always be better provisioned to meet the load requirements, e.g. by forming a server cluster [4], whereas, in the limit, it may not be technically or administratively possible to bring the mirrors any closer to the clients. In this study, we also ignore the time it takes for the client to find its closest mirror; any mechanism to improve this transaction can be equally applied to other redirection schemes. A major consideration that informs our decision to use round-trip time as the sole optimization condition is that TCP (Transmission Control Protocol), the transport protocol that underlies the large majority of Web download, has well-known biases against connections with long round-trip times (RTTs) [5]. TCP uses dropped packets as a signal of network congestion. Upon detection of network congestion, TCP backs off its transmission window size and slowly increases the window again based on successfully acknowledged transmissions. Connections with longer RTTs thus experience longer congestion recovery periods. In this paper we study in turn the use of maximum, 95%-tile, and mean clients' RTTs to their closest mirrors as the optimization condition, denoted as $\mathcal{O}(\mathcal{M}, 1)$, $\mathcal{O}(\mathcal{M}, .95)$, and $\mathcal{O}(\mathcal{M}, \mu)$ respectively.

In order to direct clients to the closest mirrors, we need to know the distances between each client and all of the mirrors. If the network topology is known, the mirror closest to any particular client can be identified by computing the shortest path from the client to all mirrors, using Dijkstra's shortest path first algorithm, for example. When network topology is not known, such as in the case of the Internet, client re-direction can be done randomly. In [6], the authors have shown that closest mirror selection using a distance map[2] invariably gives better performance than random selection. In this paper, in comparing various mirror placement algorithms, we assume that network topology is known and the closest mirror to a client can be deterministically computed. In Section III-C we present a methodology by which a virtual topology can be computed on the Internet where the physical topology is not known.

Algorithm 1 (2-approximate minimum K-center [8])

1. Construct $G_1^2, G_2^2, \ldots, G_m^2$
2. Compute M_i for each G_i^2
3. Find smallest i such that $|M_i| \leq K$, say j
4. M_j is the set of K centers

Fig. 1. Two-approximate algorithm for the minimum K-center problem.

B. Mirror Placement Algorithms and Heuristics

We now present two graph-theoretic algorithms and one heuristics we use in placing mirrors. In the subsequent discussion, we use the term "center" to mean "mirror".

Min K-center: This is a graph theoretic algorithm that finds a set of center nodes to minimize the maximum distance between a node and its closest center. Given this definition, the min K-center problem is relevant only in the case of optimization condition $\mathcal{O}(\mathcal{M}, 1)$. The min K-center problem is known to be NP-complete [7], however a 2-approximate algorithm exists [8]. With the 2-approximate algorithm, the maximum distance between a node and its nearest center is no worse than twice the maximum in the optimal case. For ease of reference, we include here our summary of the 2-approximate algorithm presented in [6]:

The algorithm receives as input a graph $G = (V, E)$ where V is the set of nodes, $E = V \times V$, and the cost of an edge $e = (v_1, v_2) \in E$, $c(e)$, is the cost of the shortest path between v_1 and v_2. All the graph edges are arranged in non-decreasing order by cost, c: $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$, let $G_i = (V, E_i)$, where $E_i = \{e_1, e_2, \ldots, e_i\}$. A *square graph* of G_i, G_i^2 is the graph containing V and edges (u, v) wherever there is a path between u and v in G_i of at most two hops, $u \neq v$. An *independent set* of a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that, for all $u, v \in V'$, the edge (u, v) is *not* in E. An independent set of G_i^2 is thus a set of nodes in G_i that are at least three hops apart in G_i. We also define a *maximal* independent set M as an independent set V' such that all nodes in $V - V'$ are at most one hop away from nodes in V'.

[1] Network round-trip time captures the path's bottleneck bandwidth, to the first degree of approximation.

[2] By "distance map" we mean a virtual topology of the Internet constructed by tracing paths on the Internet. In [6], the authors propose an architecture to build such a distance map.

Algorithm 2 (ℓ-Greedy [9])

1. **if** ($|\mathcal{M}| \leq \ell$)
2. Choose among all sets $\mathcal{M}'$ with $|\mathcal{M}'| = |\mathcal{M}|$
3. the set $\mathcal{M}''$ with minimal $\mathcal{O}(\mathcal{M}'', p)$
4. **return** set $\mathcal{M}''$
5. **end**
6. Set $\mathcal{M}'$ to be an arbitrary set of size ℓ
7. **while** ($|\mathcal{M}'| < |\mathcal{M}|$)
8. Among all sets X of ℓ elements in $\mathcal{M}'$
9. and among all sets Y of $\ell + 1$ elements
10. in $V - \mathcal{M}' + X$, choose the sets $X,\ Y$
11. with minimal $\mathcal{O}(\mathcal{M}' - X + Y, p)$
12. $\mathcal{M}' = \mathcal{M}' - X + Y$
13. **end**
14. **return** set $\mathcal{M}'$

Fig. 2. Algorithm ℓ-*Greedy*.

The outline of the minimum K-center algorithm from [8] is shown in Fig. 1. The basic observation is that the cost of the optimal solution to the K-center problem is the cost of e_i, where i is the smallest index such that G_i has a dominating set[3] of size at most K. This is true since the set of center nodes is a dominating set, and if G_i has a dominating set of size K, then choosing this set to be the centers guarantees that the distance from a node to the nearest center is bounded by e_i. The second observation is that a star topology in G_i, transfers into a clique (full-mesh) in G_i^2. Thus, a maximal independent set of size K in G_i^2 implies that there exists a set of K stars in G, such that the cost of each edge in it is bounded by $2e_i$: the smaller the i, the larger the K. The solution to the minimum K-center problem is the G_i^2 with K stars. Note that this approximation does not always yield a unique solution.

We have to make further approximations in applying the minimum K-center algorithm to the CMP problem. In the construction of the minimum K-center algorithm above, any node in G may be selected to act as a "center". In CMP, only nodes in $\mathcal{H}$ can host a mirror. Thus to apply the min K-center algorithm, we first run the algorithm on G with $V = \mathcal{H} \cup \mathcal{B}$. Should a node in $\mathcal{B}$ be selected as a center, we substitute it with a node in $\mathcal{H}$ that is closest to it. Recall that we assume $\mathcal{H} \cap \mathcal{B} = \emptyset$.

ℓ-Greedy: This algorithm places mirrors on the network iteratively in a greedy fashion. First it exhaustively checks each node in $\mathcal{H}$ to determine the node that best satisfies the optimization condition (see Section II-A) for given $\mathcal{B}$. For $\ell = 0$, after assigning the first mirror to this node, the algorithm looks for an appropriate location for the next mirror, etc. until all $|\mathcal{M}|$ mirrors are placed. For general ℓ, the algorithm allows for ℓ step(s) backtracking: it checks all the possible combinations of removing ℓ of the already placed mirrors and replacing them with $\ell+1$ new mirrors. That is, ℓ number of the already placed mirrors can be moved around to optimize the gain. Figure 2 summarizes the algorithm.

Transit Node: The outdegree of a node is the number of other nodes it is connected to. Assuming that nodes with the highest outdegrees can reach more nodes with smaller latency, we place mirrors on candidate hosts in descending order of outdegree. We call this the *Transit Node* heuristics under the assumption that nodes in the core of the Internet that act as transit points will have the highest outdegrees.

Random Placement: Under random placement, each candidate host has a uniform probability of hosting a mirror.

C. Performance Analysis

In this section, we present an analysis of the performance of unconstrained mirror placement to illustrate what could be expected of mirror placement in the ideal setting. In particular, the analysis shows that, under optimal mirror placement, there is a diminishing return in client-mirror distance[4] with respect to the number of mirrors. Despite the diminishing return, the ratio of expected maximum client-mirror distance between optimal and random placement increases logarithmically. However, under random placement, most clients are still close enough to their closest mirrors, and only a small portion of the clients are actually very "far" from their closest mirrors.

To abstract the unconstrained mirror placement problem, we can picture the network as a continuous plane on which clients can be uniformly spread over the infinitely many points ($\aleph$). We want to place a given number of mirrors such that the maximum distance of any client to its closest mirror is minimized. This measure of quality translates to finding a placement such that the radius of the largest circle one can draw in the plane that does not include any mirror is minimized.

Solving this problem analytically is cumbersome, to make the presentation clearer we study the same problem in one dimension. We can transform the problem into one dimension by distributing the clients uniformly on the segment (0,1) and placing mirrors on the same segment. Clearly, the optimal allocation of mirrors given the maximum distance criterion is to separate the mirrors by the same distance apart. Thus, if one needs to place $n-1$ mirrors, the optimal location is at locations $\frac{i}{n}$, $1 \leq i \leq n-1$, and the maximum distance from any client to its closest mirror is $\frac{1}{n}$.[5] It is clear that the gain in reduction of client-mirror distance is diminishing as the number of mirrors increases. We can also see that each mirror site will have approximately the same number of clients if each client is directed to its closest mirror.

The optimal placement could be difficult to achieve in real life. Hence, we would like to quantify how good random placement is compared to the optimal placement in terms of the expected maximum client-mirror distance. Under random placement, $n-1$ points (mirrors) are randomly distributed in the interval (0,1). Using known results from order statistics [10, Section 5.4], we have

$$Pr\{Y_{(n)} > y\} = \sum_{1-iy>0, i\geq 1} (-1)^{i-1} \binom{n}{i} (1-iy)^{n-1} \quad (1)$$

The expected value of the maximum segment between two neighboring points is thus given by

[3] A dominating set is a set of D nodes such that every $v \in V$ is either in D or has a neighbor in D.

[4] client-mirror distance always means the distance between client and the closest mirror.

[5] The actual optimal locations for n mirrors should be at $\frac{1}{2n} + \frac{i}{n}$, but the importance of this boundary condition diminishes with n. For ease of analysis, we consider only the limit case with n going to infinity.

$$
\begin{aligned}
E[Y_{(n)}] &\triangleq \int_0^1 f_{Y_{(n)}}(y)\, y\, dy \\
&= \int_0^1 \frac{d}{dy}\left(F_{Y_{(n)}}(y) y\right) dy - \int_0^1 F_{Y_{(n)}}(y) dy \\
&= F_{Y_{(n)}}(y) y\Big|_0^1 - \int_0^1 Pr\{Y_{(n)} \le y\} dy \\
&= 1 - \int_0^1 \left[1 - Pr\{Y_{(n)} > y\}\right] dy \\
&= \int_0^1 Pr\{Y_{(n)} > y\} dy \\
&= \sum_{1-iy>0, i\ge 1} (-1)^i \binom{n}{i} \frac{(1-iy)^n}{in}\Bigg|_0^1 \\
&= \sum_{i=1}^{n} (-1)^i \binom{n}{i} \frac{1}{in},
\end{aligned}
$$

where $Y_{(n)}$ is the random variable of the longest segment, $f_{Y_{(n)}}$ the density function, and $F_{Y_{(n)}}$ the cumulative distribution function.

Figure 3 depicts the computed expected maximum segment length together with numerical simulation results. Each point in the simulation represents the mean of 1000 experiments; in each, $n-1$ points are uniformly placed on the unit interval and the maximum segment is computed. The confidence interval is negligible in most cases. It is clear that the simulation and the numerical calculation are almost identical. The detailed enlargement in Figure 4 (also in Figure 5) shows that some outliers are observable in different scales. There is a clear knee around $n = 60$ after which the return from adding additional mirrors diminishes.

Comparing the segment length to the optimal length shows that for a large range, $n < 150$, the difference is substantial. Figure 5 shows the ratio of expected maximum segment length between the random placement and the optimal for both the simulated data and the calculated data. Surprisingly, it seems that the ratio increases logarithmically with the number of mirrors (we saw before that the absolute difference diminishes). To check this we fitted the exponent of the ratio with the best (mean square) linear function of the form $\alpha + \beta n$. The resulting fitted curve is $2.675 + 1.78n$. Plotting the fit for the expected maximum length in Figure 3, $\ln(2.675 + 1.78n)/n$, we could not distinguish it from the calculated one in all but the microscopic scale.

One might be tempted to discount random placement algorithm based on the above result. However, we show next that random placement is really not all that bad by examining what portion of the client population is within a "good distance" from its closest mirror given random placement. Let t be the stretch we allow in the distance from the optimal placement distance, which is $1/2n$, we calculate the portion of clients farther away from their closest mirror by more than a factor of t from optimal, i.e., by more than $t/2n$. This is done by looking at the probability that for a random point no mirror is placed at a segment of length t/n around it (a two dimensional ball of radius $t/2n$), which is given by

$$Pr\{distance > t/2n\} = (1 - t/n)^n \qquad (2)$$

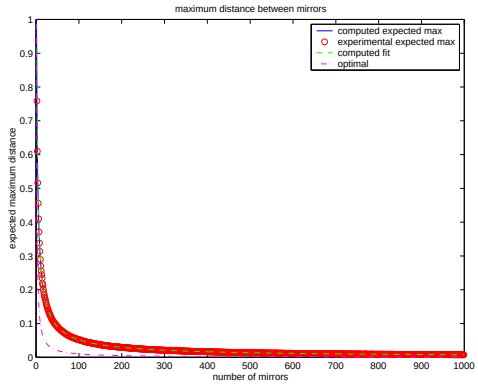


Fig. 3. The expected maximum segment length on the unit interval.

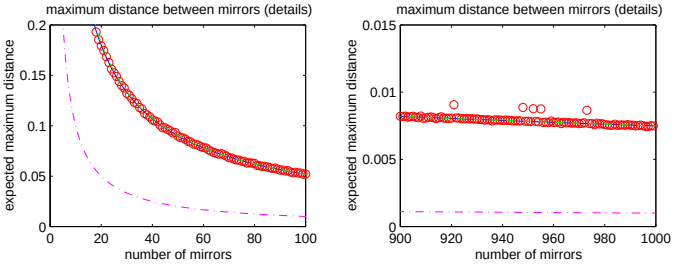


Fig. 4. The expected maximum segment length on the unit interval (details).

As n grows we can write

$$\lim_{n\to\infty} Pr\{distance > t/2n\} = \lim_{n\to\infty} (1 - t/n)^n = e^{-t} \qquad (3)$$

Thus, as the number of mirrors grows, a fixed portion of the clients are away by a certain stretch from optimal. Specifically, $1/e$ of the clients are at distance farther than the worst case of the optimal distance. Figure 6 shows the result of an experiment we conducted to test the above analysis. As one can see, the probability converges to e^{-t} for n values well below 100 (the limit values are plotted in Figure 6 as small symbols at $n = 900$).

The above analysis shows that, under the optimal placement, the reduction in client-mirror distance has a diminishing return with a well-defined "knee" as the number of mirrors increases. When clients are uniformly distributed, the optimal placement can achieve good load balancing while directing

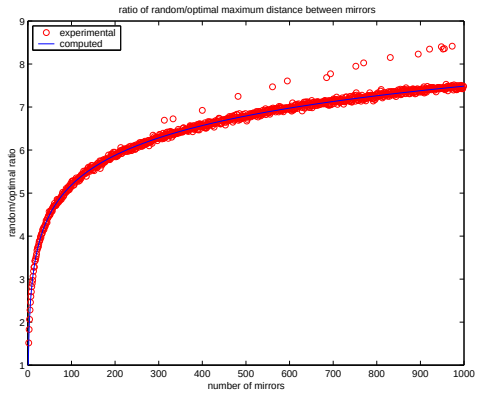


Fig. 5. The ratio of the random placement over the optimal placement.

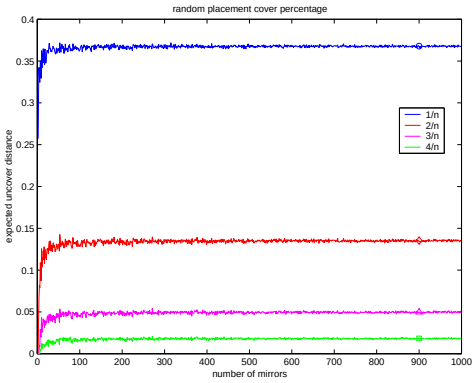


Fig. 6. The probability a client under random placement is farther than a stretch t of the distance bound in the optimal placement.

clients to the closest mirrors. Furthermore, the optimal placement increasingly outperforms random placement in terms of expected maximum client-mirror distance as the number of mirrors increases; however, extremely long client-mirror distances occur very rarely under random placement

III. Performance Evaluation

Our goal in conducting performance evaluation is to study the effect of changing $|\mathcal{M}|$ and $\mathcal{P}(\mathcal{M})$ on the optimization condition $\mathcal{O}(\mathcal{M}, p)$. For our performance evaluation, we conduct both simulations on random topologies and experiments on the Internet. For each set of experiments, we vary either $|\mathcal{M}|$ or $\mathcal{P}(\mathcal{M})$ while holding all the other variables constant. We now describe our simulation setup and scenarios, followed by a description of our Internet experiment setup.

A. Simulation Setup

The random topologies used in our simulations are generated using the Inet topology generator. The Inet topology generator generates random topologies following the observed characteristics of the Internet reported in [11]. A more thorough description of the Inet topology generator is presented in [6]. For this study, we generate several random topologies with 3,037 nodes each.[6] Each generated network is a connected graph on a plane, with nodes representing an Autonomous System (AS); a link between two nodes represents AS connectivity, and its Euclidean distance the latency between the two connected nodes. In our simulations, we place only a single client per network node.

In each simulation, we first select 50 nodes to act as candidate hosts. We experiment with two candidate host selection methods: (1) uniform selection, where each node has an equal probability of being selected, and (2) selection based on outdegree, where the nodes with the largest outdegree is selected first. After the candidate hosts are selected, we randomly, with uniform probability, select 1,000 of the remaining nodes to act as clients. For each mirror placement algorithm, we computed $\mathcal{O}(\mathcal{M}, 1)$, $\mathcal{O}(\mathcal{M}, .95)$ and $\mathcal{O}(\mathcal{M}, \mu)$. We compute each $\mathcal{O}(\mathcal{M}, p)$ for $|\mathcal{M}|$ ranging from 3 to 50. Client redirection to the closest mirror is done by both shortest-path first computation and randomly with uniform probability.

We present results of the simulations in Section IV.

[6]This was the size of the Internet in November 1997; our results with larger networks indicate that observations made in this paper also apply to larger networks.

Location	Number of Hosts	Percentage
North America	58	65.2
Western Europe	15	16.8
Rest of Europe	6	6.7
Australia	6	6.7
Israel	1	1.1
Korea	1	1.1
Mexico	1	1.1
S. Africa	1	1.1

TABLE I
Traceroute Gateway Population Breakdown

B. Internet Experiments

In addition to studying CMP on random topologies, we also evaluate it with a trace-based experiment on the Internet. In particular, we study the effect of optimizing the number and placement of mirrors on client download time when CMP is applied to the Bell Labs web server.

B.1 Candidate Host Set

We do not have access to 50 machines distributed across the Internet which can act as candidate hosts. Given our optimization condition of minimizing the latency observed by the client set, we observed that for purposes of performance evaluation, CMP can be emulated on the Internet as long as we can determine the distance between $|\mathcal{H}|$ sites on the Internet and our client set. We decided to use 89 Traceroute Gateways to serve as our candidate host sites. Traceroute Gateways are web servers made available to the public for measurement purposes by volunteers around the world. Given a host name or address, a Traceroute Gateway runs `traceroute` to that host and reports the result back to the client. Traceroute Gateways can be accessed from http://www.tracert.com/. Table I lists the geographical locations of the Traceroute Gateways used in this paper. The table reflects a reasonable diversity of the geographic locations of the Traceroute Gateways.

The *Transit* heuristics we use in placing mirrors places mirrors on candidate hosts in descending order of outdegrees. Since we do not know the outdegrees of the Traceroute Gateways, we associate with each Traceroute Gateway the outdegree of the AS they reside in. We first map the IP address of a Traceroute Gateway to its AS using a tool `prtraceroute`, which is part of the Routing Arbiter project toolkit (www.irrd.net). Then to determine the AS's outdegree, we use the AS summary information available at NLANR (moat.nlanr.net/AS/), which lists the outdegree of each AS. If the destination traceroute gateway's AS has a single connection to the rest of the Internet, we assign it the outdegree of its closest upstream AS that has outdegree more than 1.

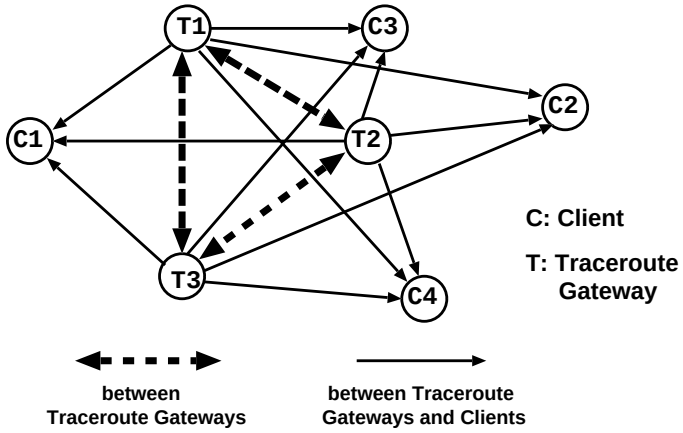


Fig. 7. Experiment Setup

B.2 Client Set

For this experiment, we collected one week (in November 1998) worth of the Bell Labs access log. During this week, the web server saw on average 26,346.9 hits per day. Of these, there were 15,561 unique domain names, which resolved to 10,115 unique IP (Internet Protocol) addresses. Due to the nature of dial-up connections, many of the dial-up clients in the log file were no longer reachable. To prevent clients that are no longer reachable from being traced by the `traceroute` gateways, we use the following procedure to obtain a list of reachable clients. We attempted to open TCP connections to each IP address from two different sites (one in Michigan and the other in California), and eliminated the ones that were not reachable by at least one of them (this is to reduce the number of hosts unreachable by the Traceroute Gateways below). Of the 10,115 unique IP addresses we obtained from the Bell Labs web server logs, 4,980 can be reached through TCP. Finally, we had each of the 89 Traceroute Gateway conduct `traceroute` to all of these IP addresses. Since `traceroute` uses ICMP (Internet Control Message Protocol) instead of TCP, and some networks or hosts do not accept ICMP packets for administrative reasons, the Traceroute Gateways were only able to trace 3,130 of these IP addresses. The client set $\mathcal{B}$ in this experiment thus consists of these 3,130 IP addresses. Table II lists the domains the clients in our client set belongs to (and the percentage thereof).

Location	Number of Hosts	Percentage
.net	585	18.82
.edu	566	18.53
.com	568	17.96
Germany	121	3.82
Canada	113	3.57
.uk	89	2.81
Japan	80	2.53
Australia	59	1.86
United	38	1.2
France	38	1.2
Sweden	32	1.01
Spain	29	0.91
.org	29	0.91
Italy	25	0.79
Switzerland	22	0.69
.gov	22	0.69
Netherlands	19	0.6
Malaysia	19	0.6
Korea	19	0.6
Hong	18	0.56
India	17	0.53
Denmark	16	0.5
Russian	15	0.47
Finland	15	0.47
Brazil	15	0.47
Belgium	15	0.47
Taiwan	14	0.44
Singapore	14	0.44
Ireland	12	0.37
Greece	12	0.37
Austria	12	0.37
.mil	12	0.37
S. Africa	11	0.34
New Zealand	10	0.31
Mexico	9	0.28
Turkey	6	0.18
Thailand	6	0.18
Portugal	6	0.18
Poland	6	0.18
Israel	6	0.18
China	6	0.18
Argentina	6	0.18
Others	31	0.98
Failed lookup	338	10.69
Total	3161	100

TABLE II
BELL LABS WEB CLIENT SET BREAKDOWN

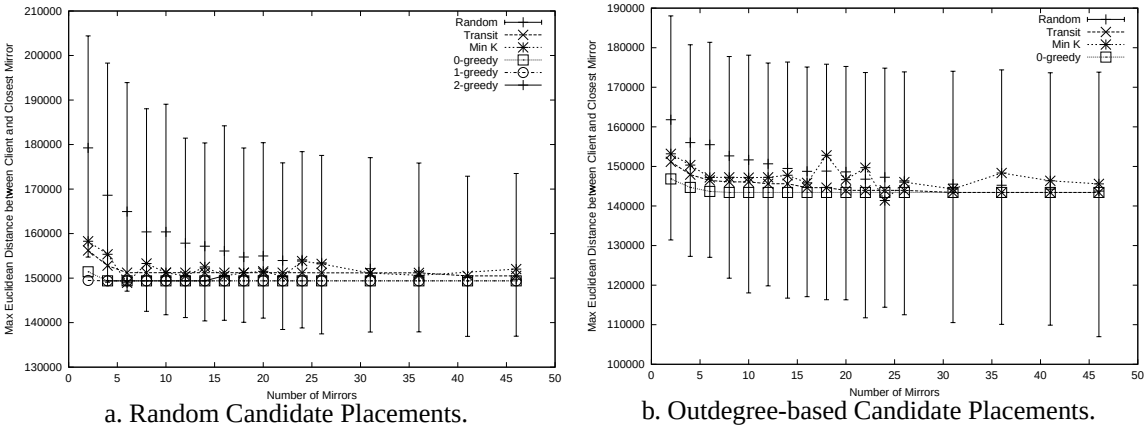
C. Distance Estimation

The virtual network on which we conduct our CMP experiments thus consists of 89 Traceroute Gateways as our candidate hosts and 3,130 IP addresses as our clients. The "edges" of these virtual network consists of round-trip time (RTT) latency measurements from each Traceroute Gateways to all of the other Traceroute Gateways and to all of the clients. For illustrative purposes, Figure 7 shows a sample virtual network consisting of four Traceroute Gateways and two clients. The Traceroute Gateways measure RTTs between each other and RTTs to the two clients. The RTT measurements between Traceroute Gateways are bidirectional, while those between Traceroute Gateways and clients are unidirectional, as indicated in the figure.

Some of the mirror placement algorithms we study require knowledge of distances between clients. In our virtual topology, we estimate the distance between two clients as the sum of the distances from each client to the closest Traceroute Gateway, and the distance between the two Traceroute Gateways. This method is usually called "triangulation" in the literature [12], [13]. In [14], the authors evaluated its efficacy on estimating distance between two points on the Internet.

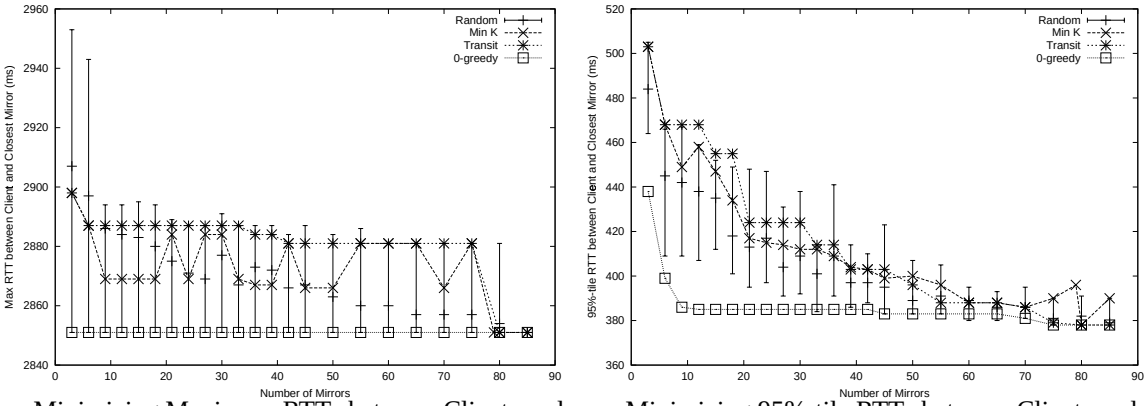
IV. EXPERIMENT RESULTS

Recall from Section III-A, in all of our simulations, we use a network of 3,037 nodes, of which $|\mathcal{H}| = 50$ are selected as candidate hosts. The choice of which host becomes a candidate hosts $\mathcal{P}(\mathcal{H})$ is determined either randomly with uniform probability for all nodes, or by the outdegree of the nodes. The client set consists of 1,000 nodes randomly selected, with uniform probability, from the remaining ones. Recall also that we define three optimization conditions: $\mathcal{O}(\mathcal{M}, 1)$, $\mathcal{O}(\mathcal{M}, .95)$, and $\mathcal{O}(\mathcal{M}, \mu)$. For each optimization condition we run a set of simu-



a. Random Candidate Placements. b. Outdegree-based Candidate Placements.

Fig. 8. Minimizing Maximum RTTs between Clients and Closest Mirrors.



a. Minimizing Maximum RTTs between Clients and Closest Mirrors. a. Minimizing 95%-tile RTTs between Clients and Closest Mirrors.

Fig. 9. Internet Experiments.

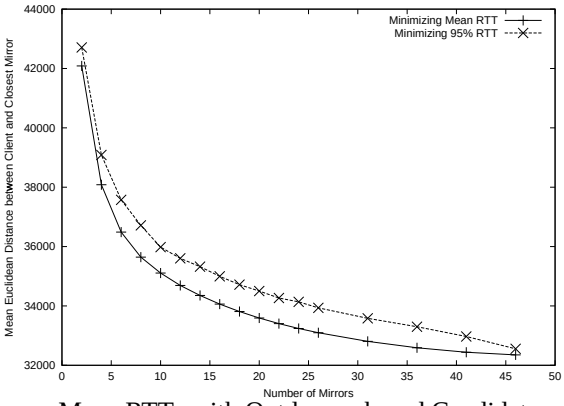
lations. In each set of simulation, we first pick $|\mathcal{M}|$, the number of mirrors. For the given number of mirrors, we run one simulation for each mirror placement algorithm, $\mathcal{P}(\mathcal{M})$: minimum K-center, 0-greedy, 1-greedy, 2-greedy, and Transit. Since random placement of mirrors gives different results based on the sites selected, for random placement we run 10 simulations for a given mirror set size and compute the mean of the observed $\mathcal{O}(\mathcal{M}, p)$. Then we repeat all simulations for the next $|\mathcal{M}|$. In our simulations, we experiment with $|\mathcal{M}|$ ranging from 2 to 50, stepping by 2 up to 26, and stepping by 5 afterwards. We then repeat each set of simulations on 10 different Inet generated networks of 3,037 nodes each. We do the above on 50 randomly selected candidate hosts. Then we repeat everything again on 50 candidate hosts selected based on decreasing number of outdegrees, except that we do not simulate the 1-greedy and 2-greedy algorithms as they do not show marked improvement over the 0-greedy case in the former scenarios. Hence in total we ran 7,350 simulations on randomly selected candidate hosts, and 6,630 simulations on candidate selection based on outdegree.

For the Internet-based experiment, we repeat the above scenario except using the 89 Traceroute Gateways as candidate hosts. Mirror set sizes range from 3 to 89, stepping by 3 up to 45, and stepping by 5 afterwards. Since there is only one virtual network, we do not repeat the set of simulations 10 times; we do, however, still repeat the experiment 10 times for each mirror set size when the mirror placement algorithm used is random placement. This means we run 1,014 experiments on the virtual network.

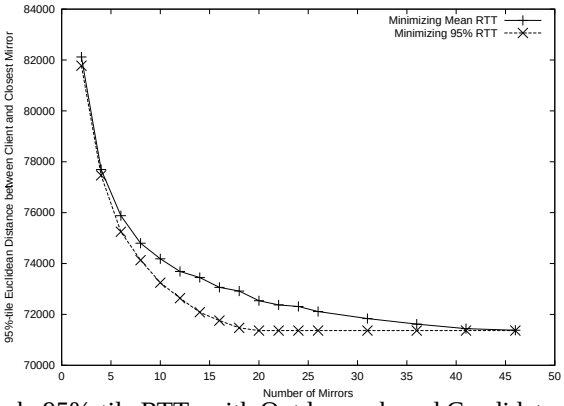
A. Optimization Condition $\mathcal{O}(\mathcal{M}, p)$

We first consider the optimization condition $\mathcal{O}(\mathcal{M}, p)$. Figures 8a, 8b, and 9a show the maximum client-mirror RTTs for $\mathcal{O}(\mathcal{M}, 1)$. The x-axis of each figure lists the number of mirrors, and the y-axis the maximum RTT between clients and their closest mirrors. The x-axes for the simulation results range from 0 to 50, while those for Internet experiments range from 0 to 90. The y-axis in the various figures have different ranges. In the simulation results, the "distance" between two nodes is the Euclidean distance between them on the simulated plane. In the Internet experiments, distance is in milliseconds. The numbers for all placement algorithms, except for random placement, are averaged over simulations on 10 random topologies to obtain the mean, the maximum, and minimum. For clarity, we only show the statistics for random placement in the figures. Recall that for each of the 10 random topologies, we simulate random placement of n mirrors 10 times. From these 10 placements, we get a mean worst case client-mirror RTTs. Each error bar shows the mean, the maximum and the minimum values of these mean values over the 10 random topologies. We see that the maximum and the minimum values are typically within 20% of the mean.

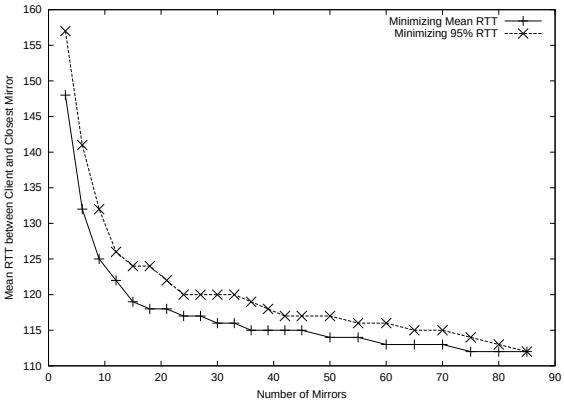
From these figures, we observe that optimizing $\mathcal{O}(\mathcal{M}, 1)$ yields very little improvement as the number of mirrors in-



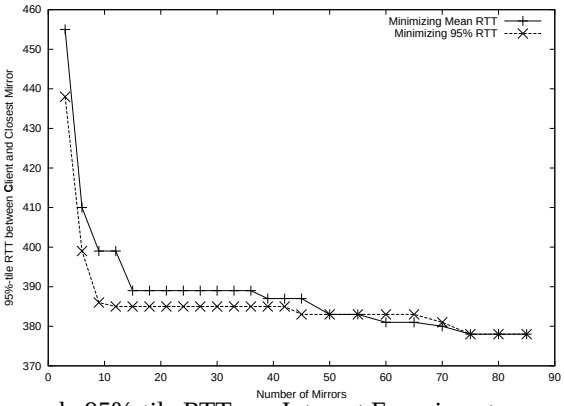
a. Mean RTTs with Outdegree-based Candidate Placement.



b. 95%-tile RTTs with Outdegree-based Candidate Placement.



a. Mean RTTs on Internet Experiments.



b. 95%-tile RTTs on Internet Experiments.

Fig. 10. Mean and 95%-tile RTTs.

creases, both in simulations and actual Internet experiments. In constrained mirror placement, the distance between clients and mirrors can not be improved incrementally at finer and finer granularity because mirrors can not continuously be placed progressively closer to the clients. Both candidate site placement and mirror placement can contribute to this problem. First, the optimal mirror placement is very "location-sensitive" in that it has very specific requirements on where the candidate sites should be, i.e., separated by an equal distance. Also, the optimal solutions for different mirror value n have very little overlap so it is unlikely all $O(n^2)$ optimal locations (at $\{1/2\}$, $\{1/3, 2/3\}$, $\{1/4, 1/2, 3/4\}$, ...) would be included if n mirror candidate sites are randomly selected. Second, adding more mirrors can not improve the minimum distance between a client and its closest candidate site (therefore the client's closest mirror) further, once the candidate site is selected for mirror placement. This problem can be exacerbated when the number of candidate sites is small relative to the client population.

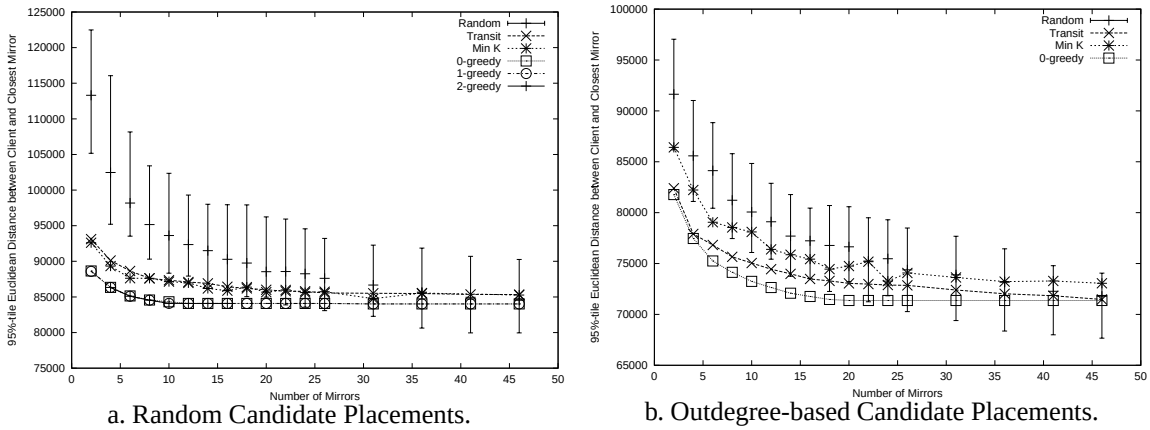
Figure 10 shows the mean and 95%-tile of client-mirror distances when candidate sites are selected based on outdegrees, and mirror placement is by the 0-greedy algorithm. Recall that solution to the min K-center problem is applicable only in the case of optimization condition $\mathcal{O}(\mathcal{M}, 1)$. Hence for optimization conditions $\mathcal{O}(\mathcal{M}, \mu)$ and $\mathcal{O}(\mathcal{M}, .95)$ we consider only the l-greedy algorithm, in particular 0-greedy. Both the 95%-tile and mean client-mirror graphs show diminishing return and a well-defined "knee", which confirms the theoretical analysis and our intuition. We observe very similar performance between the two curves, reflecting $\mathcal{O}(\mathcal{M}, .95)$ and $\mathcal{O}(\mathcal{M}, \mu)$ optimization conditions, and attribute this to the potentially long, but nonetheless not heavy tail of the client-mirror RTT distribution in our setups (which means that the 95%-tile is not that far from the mean). In the remainder of this paper, we use $\mathcal{O}(\mathcal{M}, .95)$ as our optimization condition.

B. *Effect of $|\mathcal{M}|$ and $\mathcal{P}(\mathcal{M})$ on $\mathcal{O}(\mathcal{M}, p)$*

Figures 9b, 11a, and 11b show the observed 95%-tile RTTs between clients and their closest mirrors when $\mathcal{O}(\mathcal{M}, .95)$ is used. Note that in most cases, especially when the 0-greedy algorithm for mirror placement is used, there is little improvement in 95%-tile RTT beyond 10 mirrors.

One important observation with regard to $\mathcal{P}(\mathcal{M})$ is that placement is very important when the number of mirrors is small. In all cases, when $|\mathcal{M}|$ is small, there is a significant difference in observed latency between using the greedy placement algorithm and random placement. When $\mathcal{P}(\mathcal{H})$ is uniform, non-random $\mathcal{P}(\mathcal{M})$ outperforms random placement. Even when $\mathcal{P}(\mathcal{H})$ is non-random, as in the case of outdegree-based candidate selection, using greedy placement improves $\mathcal{O}(\mathcal{M}, .95)$ by 10% to 20% as shown in Figure 11 (note the difference in y-axis ranges).

We conclude that increasing the number of mirrors beyond a small portion of the candidate sites (10, in our examples) does not necessarily improve client to closest mirror latency. Furthermore, careful placement of mirrors on a small candidate sites can provide the same performance gain as placing mirrors on all candidate sites. These preliminary results seem to suggest that



a. Random Candidate Placements. b. Outdegree-based Candidate Placements.

Fig. 11. Minimizing the 95%-tile RTTs between Clients and Mirrors.

candidate site placement can be just as important and possibly more important than mirror placement itself. We note that practically candidate host sites are often decided by administrative and financial constraints rather than technical ones.

C. Mirror Load Distribution

We now show that using $\mathcal{O}(\mathcal{M}, .95)$ as the optimization condition, mirror load distribution is not improved with larger number of mirrors. Figure 12 plots client distribution among mirrors when the number of mirrors is increased from 2 to 50 (3 to 89, in the Internet experiment). The x-axis is the popularity rank of each mirror, and the y-axis is the number of clients redirected to a particular mirror, with the most popular one getting the most redirections. Each curve in the graphs represent a specific mirror set size. For the simulations, the candidate sets are chosen based on decreasing outdegrees. In all cases, the optimization condition is $\mathcal{O}(\mathcal{M}, .95)$, and the mirror placement algorithm is 0-greedy. In the simulation, only a small number of clients (less than 1% mirrors) get redistributed with each additional mirror once the number of mirrors is above 15 mirrors. Client redistribution is also very infrequent in our Internet experiments.

Again, we point to our analysis in Section II-C, where we showed that the optimal placement produces good load-balancing among mirrors as the number of mirrors increases. We have already shown that it is difficult to reproduce the ideal setting when mirror placement is constrained so perhaps it is not surprising that we also lose the ability to load-balance. However, we want to point out that one can still achieve load-balancing if the requirement that each client be directed to the closest server is ignored.

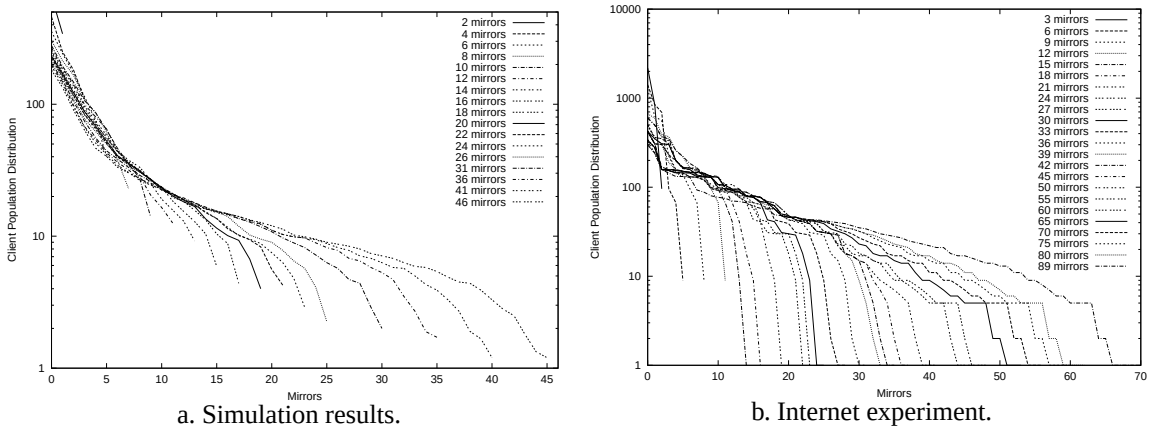
D. Effect of Redirection Methods

Up to now we have assumed that client-mirror distances can be deterministically computed using Dijkstra's shortest path first algorithm. In this section we consider the case where only 10 of the highest outdegree Traceroute Gateways are able to do `traceroute`. Hence distances between the other Traceroute Gateways and between a Traceroute Gateway, other than these 10, to a client must be estimated by doing triangulation on the distances measured by these 10 Traceroute Gateways only. This simulate the case where the underlying network topology is not known (such as the case with the Internet) and a "distance map" of the underlying topology must be estimated by placing measurement boxes on the network. We have shown by simulations in [6] that when the underlying network topology is not known, nearest mirror redirection using some form distance map outperforms random redirection. We now show that similar results can also be observed on the Internet. Figure 13 shows the 95%-tile of client-mirror RTTs under $\mathcal{O}(\mathcal{M}, .95)$ when distances are known, with random redirection, and with redirection using a distance map. The results were obtained from Internet-based experiments, when mirrors are placed using the 0-greedy algorithm.

V. Related Work

There have been some recent works on mirror performance and closest server selection. In [15], the authors measured 9 clients scattered throughout the United States retrieving documents from 47 Web servers, which mirrored three different Web sites. They presented findings that revealed good stability of mirror rankings according to download time. In [16], the authors present a server selection techniques that can be employed by clients on end hosts. The technique itself involves periodic measurements from clients to all of the mirrors. The authors of [17] proposed a server selection scheme based on shared passive end-to-end performance measurements collected from clients in the same network. There are also related works that focus on maintaining consistency among cache servers, which can be applicable in keeping mirrors consistent. In [2] and [1], the authors studied different scalable web cache consistency approaches and showed various overhead of keeping caches consistent.

There has not been, however, any study we are aware of that gives specifics on how to do mirror placement on the Internet. In [6], two graph theoretic algorithms, k-HST [18] and Min K-center [8], are used to determine the number and the placement of *instrumentation boxes* for the purpose of network measurement. While the authors of the paper use nearest mirror selection as a motivating problem, the 3 mirrors they consider are manually placed on arbitrarily selected locations. In this paper we take a closer look at mirror placement on the Internet under a realistic setting where the number of mirrors is small, but generally larger than 3, and the placement is restricted to a given set of hosts.



a. Simulation results. b. Internet experiment.

Fig. 12. Client population distribution under 95%-tile RTT optimization

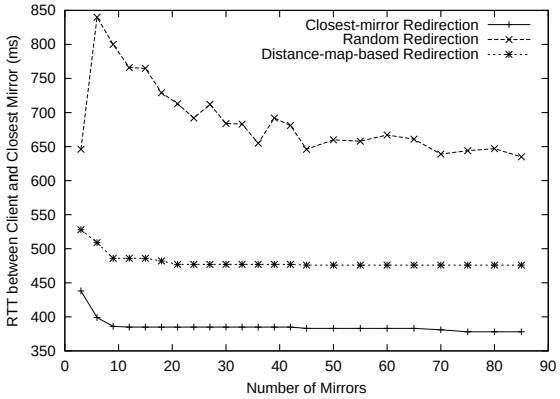


Fig. 13. 95%-tile RTT optimization under different redirection schemes.

VI. Conclusion

In this paper, we take a detailed look at the problem of placing mirrors of Internet content on a restricted set of hosts. Using both simulation and real Internet delay data, we examine a number of placement and redirection algorithms for placing various numbers of mirrors and their effects in client response time and mirror load distribution. We observed that there is a rapid diminishing return to placing more mirrors in terms of both client latency and server load-balancing. We hypothesize that the presence of the locality constraint has eliminated some of the necessary conditions for obtaining the optimal solution and the subsequent performance benefits. Even under the more elaborate placement schemes, simply increasing the number of mirrors yields very little performance improvement beyond a relative small number of mirrors.

VII. Acknowledgments

We thank all volunteers who donated their time and resources to host the `traceroute` gateway service and make it available to the public (http://www.tracert.com/cgi-bin/trace.pl).

References

[1] C. Gray and D. Cheriton, "Lease: An efficient fault-tolerant mechanism for distributed file cache consistency," *Twelfth ACM Symposium on Operating Systems Principles*, pp. 202 – 210, 1989.

[2] H. Yu, L. Breslau, and S. Shenker, "A scalable web cache consistency architecture," *Proc. of ACM SIGCOMM*, Sept. 1999.

[3] B. Krishnamurthy and J. Wang, ""On Network-Aware Clustering of Web Clients"," *Proc. of ACM SIGCOMM 2000*, Aug. 2000.

[4] V. Cardellini, M. Colajanni, and P.S. Yu, "Dynamic Load Balancing on Web Server Systems," *IEEE Internet Computing*, pp. 28–39, May-June 1999.

[5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *ACM/IEEE Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[6] Sugih Jamin, Cheng Jin, Yixin Jin, Dan Raz, Yuval Shavitt, and Lixia Zhang, "On the placement of internet instrumentation," *Proc. of IEEE INFOCOM*, Mar. 2000.

[7] Michael R. Garey and David S. Johnson, *Computers and Intractability*, NY, NY: W.H. Freeman and Co., 1979.

[8] Vijay Vazirani, *Approximation Methods*, Springer-Verlag, 1999.

[9] P. Krishnan, Danny Raz, and Yuval Shavitt, "The cache location problem," *ACM/IEEE Transactions on Networking*, vol. 8, no. 5, Oct. 2000, to be published.

[10] Herbert A. David, *Order Statistics*, Wiley, second edition, 1981.

[11] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos, "On power-law relationships of the internet topology," *Proc. of ACM SIGCOMM*, Aug. 1999.

[12] S. Hotz, "Routing information organization to support scalable interdomain routing with heterogenous path requirements," Tech. Rep. Ph.D. Thesis, Univ. of Southern California, CS Dept., 1994.

[13] James D. Guyton and Michael F. Schwartz, "Locating nearby copies of replicated internet servers," *Proc. of ACM SIGCOMM*, Aug. 1995.

[14] Francis P. et al., "IDMaps: A Global Internet Host Distance Estimation Service," Submitted for publication, 2000.

[15] A. Myers, P. Dinda, and Hui Zhang, "Performance characteristics of mirror servers on the internet," *Proc. of IEEE INFOCOM*, Mar. 1999.

[16] Zongming Fei, Samrat Bhattacharjee, Ellen W. Zegura, and Mostafa Ammar, "A novel server selection technique for improving the response time of a replicated service," *Proc. of IEEE INFOCOM*, 1998.

[17] Srinivasan Seshan, Mark Stemm, and Randy H. Katz, "Shared passive network performance discovery," *Proc 1st Usenix Symposium of Internet Technologies and Systems (USITS '97)*, Dec. 1997.

[18] Yair Bartal, "Probabilistic approximation of metric space and its algorithmic applications," in *37th Annual IEEE Symposium on Foundations of Computer Science*, Oct. 1996.