Constrained Mirror Placement on the Internet

Paper ID: 383
Pages: 21

Abstract

It is well-known that placing mirrors of web content across the Internet can be an effective way to decrease load on the provider's site. Although there are theoretic studies on similar problems such as center-placement, there have not been many studies that give specifics on how to do mirror placement on the Internet. We show that placing mirrors everywhere is not an optimal solution for most providers because the cost of keeping mirrors consistent is at best linear, and there are diminishing benefits with respect to the number of mirrors. In this study, we take a closer look at mirror placement on the Internet under a realistic setting where the number of mirrors is small, and the placement is restricted to a given set of hosts. Using both simulation and real Internet delay data, we examine a number of algorithms for placing various numbers of mirrors, methods to redirect clients to the closest mirror, and their effects on client response time, reduction in traffic, and server popularity distribution.

1 Introduction

To increase the reliability and performance of frequently accessed Web servers on the Internet, mirror servers, or simply mirrors, have been employed by a growing number of Web sites. These mirrors, which offer replicas of popular data items, can provide much lower latency to clients and higher aggregate throughput to the providers. However, placing such mirrors incurs certain costs to both the content providers and the networks that host them. To achieve a good balance between the cost to the content providers and the performance improvements for clients and providers, it is important to understand some of the fundamental factors in mirror placement, such as the number of mirrors and their placement on the Internet.

There have been some recent works on mirror performance and closest server selection. In [9], 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 down-load time. In [3], the authors present a server selection techniques that can be employed by clients on end hosts. The technique itself involves periodic measurements from clients themselves to all of the mirrors. The authors of [10] 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 [12] and [7], 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 [8], two graph theoretic algorithms, k-HST [1] and Min K-center [11], are used to determine the number and the placement of *instrumentation boxes* for the purpose of network measurement. These theoretic algorithms break the network into a set of partitions with some bounded diameter assuming the network topology is known. 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. Using both simulation and real Internet delay data, we then examine a number of placement and redirection algorithms for placing various numbers of mirrors and their effects in client response time, reduction in traffic, and server popularity distribution. While we observed improved client response time as the number of mirrors increases, we also observed diminishing returns on both the client and the server side, on the usage of network links, and the increasing importance of the closest-server determination. Overall, our results show that increasing the number of mirrors alone is not a cost effective way to improve client response time. In general, random placement combined with closest-server redirection based on some course-grain distance-map is a cost-effective way to achieve good performance.

In the next section we present the mirror placement problem within the constraint imposed by the Internet and describe our Constrained Mirror Placement (CMP) approach. In Section 3 we present our experiment methodology. We show our results and findings based on simulation and real Internet environment in Sections 4, 5, and 6. We summarize this paper and propose our future work in Section 7.

2 Mirror Placement

As important as mirror placement may be, there has not been a detailed study of how to select sites for mirror placement. Theoretical studies on mirror placement are limited in applicability by the fact that mirrors can not be placed on the Internet on a need-to-have basis. In this study, we take a more realistic approach to try to solve the mirror placement problem within the constraint imposed by the Internet itself.

We call this the Constrained Mirror Placement (CMP) problem.

2.1 Problem Definition

Given the Internet represented as a graph $G = (V, E)$, where V is the set of nodes, and $E \subseteq V \times V$ is the set of links. We define $X \subseteq V$ to be the number of clients of a particular service on the Internet, $Y \subseteq V$ to be the number of candidate sites where mirror can be placed, and $Z \subseteq Y$ to be the number of mirrors that the service is willing to support. The objective of the CMP problem is to pick Z mirrors out of the Y available sites such that some optimization condition is satisfied for X clients.

2.2 Parameters X, Y, and Z

In the following paragraphs, we examine the role of each parameter in the performance of the mirror placement problem. We will take a detailed look at the choice of parameters and how we made the selection of parameter values.

X**: Client Population:** There are two important parameters associated with the client population. The number of clients as a percentage of total population, and the distribution of clients over the Internet. In the real-world, both of these two parameters are highly dependent on the nature of the content provided. In our simulation, we have made some assumptions based on our observation of the current state of the Internet.

Y**: Candidate Population:** The choice of candidate sites for mirror placement is usually constrained by administrative agreement and the cost associated with hardware and maintenance. Nonetheless, there are a few technical issues that should be examined: the number of candidate sites must be considered, and so is the whereabouts of such candidate sites. Again, we made assumptions based on observations of current practices on the Internet.

Z**: Mirror Selection:** The number of mirrors and the mirror placement algorithm are two parameters to be considered. The number of mirrors is more or less a parameter to be fine tuned; the placement however deserves an in-depth look. Existing theoretical placement algorithms usually assume that mirrors can be placed anywhere within the client population. In CMP and on the Internet however, the candidate sites and the client sites do not necessarily overlap. Therefore, we are not able to apply some of the center-placement algorithms from existing literature. Instead, we will run a two-phase algorithm in which the first phase selects a list of mirrors from among the client populations only, and in the second phase, for each of the mirror that is located at a client site, we find the closest candidate site and use it instead as the mirror site.

2.3 Optimization Condition

One of the main goals of placing mirrors on the Internet is to reduce client down-load times (the other being the reduction of server load). TCP (Transmission Control Protocol), the transport protocol that underlies all Web down-loads, has well-known biases against connections with long round trip times (RTTs) [4]. Connections with longer RTTs will experience longer recovery periods for packet drops due to congestion, hence smaller throughput and slower down-load. Thus, one metric to optimize for in mirror placement would be the RTTs between clients and their closest mirror. We could either optimize for the worst case, i.e. minimize the maximum client-mirror RTT, or optimize for the average case, i.e. minimize the average client-mirror RTT. We decided to optimize for the worst case since we are looking only at unique clients when evaluating mirror placement algorithms. Having multiple clients at the same address will vary the average case, but not the worst case. However, it would be short-sighted of us to ignore the average case performance. As we will show later, optimizing for the worst-case does not harm average-case performance significantly.

2.4 Evaluation

We examine the following performance metrics to evaluate the success of our mirror placement solution:

- the distribution of latency between each client and its closest mirror.
- the average latency between each client and its closest mirror.
- the distribution of clients among mirrors.
- the traffic reduction in term of the number of links (hop count) traversed.

3 Experiment Methodology

Consider the extreme case in which all the clients of a Web server are colocated in the same campus. Should a mirror of the server be needed, it will probably be placed on campus. In the more general case, an efficient placement of a server's mirrors would also depend on the distribution of its client's population across the Internet. To study a mirror placement algorithm thus would require access to raw server logs to extract the client population. Due to privacy concerns, such logs are not easily obtainable. Lacking sufficient number of such logs, we carefully explore the parameter space of the CMP problem through simulation. To somewhat confirm our simulation results, we also conduct experiments using the access logs of our departmental Web server, which is available to us. Given the limited amount of real world client traces, our experimental study is necessarily limited in nature, we present its results only as a sanity check against our simulation.

3.1 Simulation Setup

As with most simulations involving the Internet, our first task in setting up the simulations is to select a topology generator. In this paper we use the topology generator presented in [8]. The topologies are generated based on observed Internet characteristics reported in [2]. Further details of the topology generator can be obtained from [8]. In this study, we generate network topologies of 6000 nodes. Each of the topologies is a connected graph with nodes representing networks, and links between nodes representing connectivities and network latencies. In this study, we do not make a distinction between networks and hosts. Each node can be seen as a single network, and we only pick one client per network.

Once a topology is generated, we have to decide on the parameters of the CMP problem: client population, candidate population, and mirror placement algorithms. We also need a mechanism by which clients are directed to mirrors. We first describe various parameter choices, and then we will present our simulated scenarios.

3.1.1 Simulation Parameters

Candidate Selection: There are two ways to select the candidate sites for mirror placement: random and core-based. Random selection picks the required number of candidates through a uniform random generator. The core-based selection picks nodes with highest out-degrees. We can obtain the out-degree of a node in simulation by simply counting the number of links the node has.

Client Selection: In selecting clients, we exclude all the nodes that have already been selected as candidates. We simulate the coverage of client populations based on the actual geographic separations of the Internet. We simulate the presence of hosts in Asia, Europe, and North/South America by somewhat arbitrarily allocating 80% (in 20%, 20%, and 40% portions) of nodes to represent the three separate groups of hosts, with the remaining clients randomly scattered across the network to account for hosts in other regions. To simulate geographical clustering in each continent, we start by selecting three "continental" nodes that are separated from each other by a distance of at least one third of the network diameter. For each "continental" node, we pick twice the clients assigned to the "continent" starting with nodes closest to the "continental" node; we then randomly select half of the selected nodes as true clients.

Mirror Placement Algorithms: We look at the following mirror placement algorithm: Min K-center placement, which optimizes for the worst case, placement on nodes with high connectivity, and random placement.

- Min K-center is a graph theoretic algorithm that finds a set of center nodes to minimize the maximum distance between a node and its closest center. The problem is known to be NP-complete [6], but a 2-approximation algorithm exists [11]. Given a network, $G = (V, E)$, the approximation algorithm works by finding the smallest maximal independent set, whose size is no more than the number of centers, of the square graph G^2. However, this algorithm can not be directly applied to CMP because the set of "centers" – mirrors is not necessarily a subset of the client population. Instead, we combine the client and candidate population and run the standard approximation algorithm to Min K-center. For each mirror that happens to be colocated with a client, we will substitute the closest candidate site to the client.

- Placing mirrors using node out-degree is straightforward in simulation. We define the out-degree of a node to be the number of links originated at the node. After we compute the out-degree of each node, we sort all nodes by their out-degrees in descending order and select the most connected nodes as centers.

- Under random placement, mirrors are selected using an uniform random generator until the desired

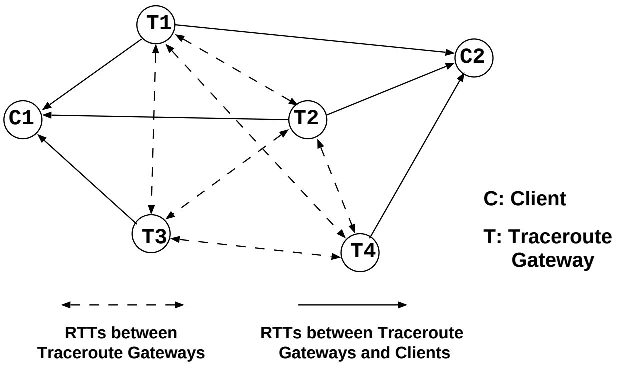


Figure 1: Experiment Setup

number of mirrors is reached.

Client Redirection: Typically when mirrors of a Web site are present, a client will still initially connect to the main Web site and then be redirected to one of the mirrors based on certain criterion. The client will subsequently only retrieve data from this mirror that it is redirected to. In our study, we will ignore the initial latency incurred connecting to the main server and only focus on the latency between a client and the mirror it is redirected to. There are two simple client redirection schemes we consider: redirection to the closest-mirror and random redirection. In the case of the closest-mirror redirection, each client is sent to the mirror that is closest in network latency.

3.1.2 Simulated Scenarios

We have designed the following scenarios to simulate two kinds of Web sites: one with medium popularity and one with high popularity on the Internet. We made the size of the candidate sites to be a significant percentage of the total client population (15%) to investigate the effect of having large number of mirrors.

Scenario I: 1500 geography-based clients, randomly placed 225 candidate sites, and 1%, 5%, 15%, 30%, 50%, 75% and 100% candidates are chosen as mirrors separately.

Scenario II: 4500 geography-based clients, randomly placed 675 candidate sites, and 1%, 5%, 15%, 30%, 50%, 75% and 100% candidates are chosen as mirrors separately.

In both of these two simulation scenarios, we perform Min K-center, node out-degree based and random placement algorithms to place mirrors among candidates. We use both closest-mirror and random redirections under all placement algorithms to redirect clients to mirrors. For each simulation, we record

3 statistics: the distance between each client and its closest mirror, the mirror each client is redirected to, and the hop count between each client and its closest server.

3.2 Internet Delay Data

This section describes the procedure we use to conduct experiments on the Internet. Given the limited access to Web logs, we are not claiming general applicability of the results obtained from these experiments. We present these results as sanity checks against our simulation results.

3.2.1 Traceroute Data Collection

In order to conduct experiments under CMP, we need to have access to sites that can be mirror candidates. More precisely, we need to have access to hosts that can measure RTTs to the rest of the Internet so we can tell how good mirrors are if they were to be placed on these sites. The Multiple Traceroute Gateways (MTG) are a collection of Web sites, each of which is capable of running `traceroute` to clients that can be specified through HTTP request. The output of such a `traceroute` contains the path information from the `traceroute` gateway to a host including the name of each intermediate router, its hop-count and three RTT measurements from the `traceroute` gateway.

Next, we need to find realistic client population. We obtained the WWW log file of our EECS department Web server and used the clients in the log file as our basis for this study. Due to the nature of dial-up connections, many of the dial-up clients in the log file were no longer reachable. To eliminate the unreachable clients, we attempted to open TCP connections to each client IP address from three different sites (Two in US and one in Israel) on the Internet and eliminated ones that were not reachable by at least one of them.

We need to know how far away each client is from each candidate site, i.e., a `traceroute` gateway, so we know the quality of the placement algorithm by examining how far each client is from its closest mirror. We set up each `traceroute` gateway (by sending HTTP requests) to trace the entire client list simultaneously. From the `traceroute` output for each client, we record the hop count and the minimum of the three RTTs. Figure 1 illustrates the experiment setup where four `traceroute` gateways measure RTTs between each other and RTTs to two clients. The RTT measurement between `traceroute` gateways is bidirectional, while the RTT measurement between `traceroute` gateways and clients is

Location	Number of Hosts	Percentage
North America	19	43.1
Europe	16	36.4
Japan	3	6.8
South Africa	2	4.5
Australia	2	4.5
Russia	1	2.3
United Kingdom	1	2.3

Table 1: Traceroute Gateway Population Breakdown

unidirectional as indicated in the figure.

3.2.2 Distance Estimation

The approximation algorithm to Min K-center placement needs to have the complete end-to-end distance-map between pairs of clients. Since there are no provisions for us to measure end-to-end RTTs between pairs of clients. We have to resort to other techniques for distance estimation.

The authors in [5] presented a technique called triangulation that estimates end-to-end delays between hosts assuming shortest path routing and routing symmetry. The triangulation technique approximates a single RTT between host A and host B by summing up the RTT from host A to another host C, the RTT between host C and D, and the RTT between host D and host A. We borrow this technique here to estimate end-to-end distances between pairs of clients.

As pointed out by the authors in [5], in order for triangulation to be successful, we need to have multiple triangulated paths between pairs of clients to extract the minimum path. We discovered that many of the clients were only reachable by a small number of `traceroute` gateways, which would make the number of multiple paths very small. Therefore, we filtered out clients that were not reachable by at least N `traceroute` gateways. Furthermore, we selected a subset of `traceroute` gateways that could reach each other so the RTTs between `traceroute` gateways were exact. Out of the 74 `traceroute` gateways that we started with, we were able to find 44 of them that could reach each other; for clients, we were able to find 285 out of a total of 447. We then constructed a virtual topologies between `traceroute` gateways and clients and ran Dijkstra's algorithm to compute the minimum RTTs between pairs of clients.

Location	Number of Hosts	Percentage
net	42	14.7
APNIC (Asia)	37	13.0
RIPE (Europe)	36	12.6
com	33	11.6
AOL	28	9.9
Authors' domain	24	8.4
edu	20	7.0
Europe	13	4.6
Asia	11	3.9
USA	11	3.9
EXTREME	7	2.5
TNET	5	1.8
Canada	5	1.8
EROLS	4	1.4
org	2	0.7
Australia	2	0.7
Mexico	2	0.7
South Africa	2	0.7

Table 2: EECS Web Client Population Breakdown

3.2.3 Host Locations

Since we filtered out some hosts and traceroute gateways for reachability reasons, we may have inadvertently skewed the candidate and the client population. Therefore, it is important to analyze both populations to ensure the reasonable diversity of host locations. From the Web pages of the multiple `traceroute` gateways, we obtained the geographical location information of the `traceroute` gateways used in this paper, which is shown in Table 1. We developed a tool called `getASbyIP`, which uses the IP address of the client and the `whois` database to locate the client's AS. Unfortunately, we were not able to locate the AS information for all clients, and in some cases, we were only able to tell which regional authority a particular client IP address was delegated to. For all other undetermined hosts, we group them by their root domain names under net, com, edu, org, or geographically to USA, Europe, and Asia depending on the convention of root domain names.

From the above information, we believe that the clients are reasonably scattered throughout different parts of the Internet with no major concentration in one localized region or a single institution. Furthermore, we also looked at the client IP addresses, and no large blocks of IP addresses with identical top 24 bits are identified except AOL.

3.2.4 Mirror Experiments

The mirror experiments are conducted using the same placement algorithms as in the simulation. However there are practical issues that we have to resolve in order to apply these algorithms.

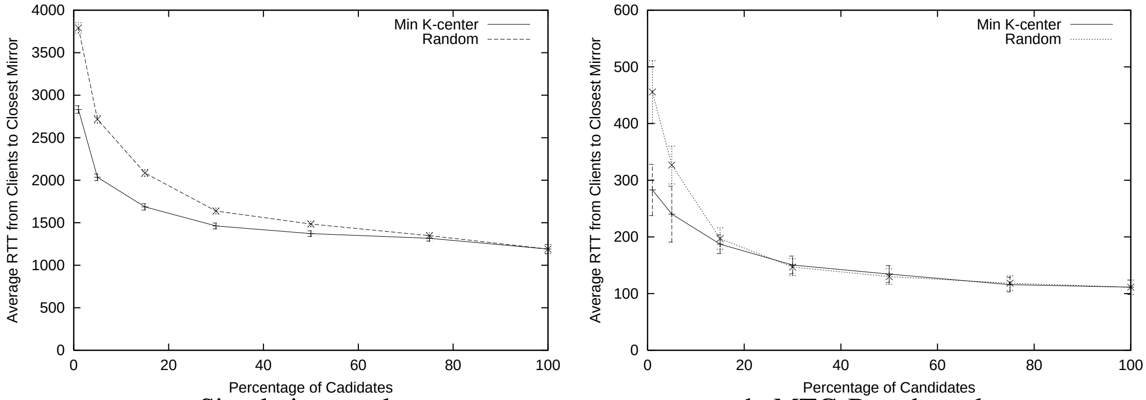
- The approximation algorithm for Min K-center placement requires complete distance information between all pairs of hosts. We had to approximate end-to-end distances using triangulation as described in Section 3.2.2. We realize that when the number of candidate sites is large, measuring end-to-end distances between all candidate sites and clients becomes difficult and costly especially if we have to perform the measurements over time. For client redirection, we will have to rely on some kind of distance-map, built either by ourselves or provided by others, which is likely to be coarse-grain. We picked 10 most connected `traceroute` gateways to act as triangulation agents, and we conducted experiments using the rest of the `traceroute` gateways and clients. For client redirection, We approximate distances between mirrors and clients following the method outlined in Section 3.2.2.

- The node out-degree is considered a useful metric for determining the placement of mirrors. Nodes with high out-degrees generally have good connectivity to the rest of the Internet, which makes them attractive for mirror placement. In our MTG-based experiments, it is difficult for us to find out the exact out-degree of every `traceroute` gateway. Instead, we estimate the connectivity of every `traceroute` gateway using the following procedure: for each `traceroute` gateway, we cut off the first 10 hops of the output of `traceroute` to each client; we then determine the number of unique links present between the 10th and the 11th hop routers and use that as the out-degree. Based on the rank of the out-degrees of the candidate sites, we choose the ones with the highest out-degrees as mirrors.

- We can simply apply the random placement in this case without any change.

4 Diminishing Returns

In this section we focus on studying the effectiveness of increasing the number of mirrors. We look at the improvement in terms of latency between clients and mirrors, server popularity distribution, and reduction in traffic. One solution to the mirror placement problem is to blanket the Internet with a server's mirrors.



a. Simulation result b. MTG-Based result

Figure 2: Average RTT between Clients and Mirrors using Min K-center and Random Placement Algorithms

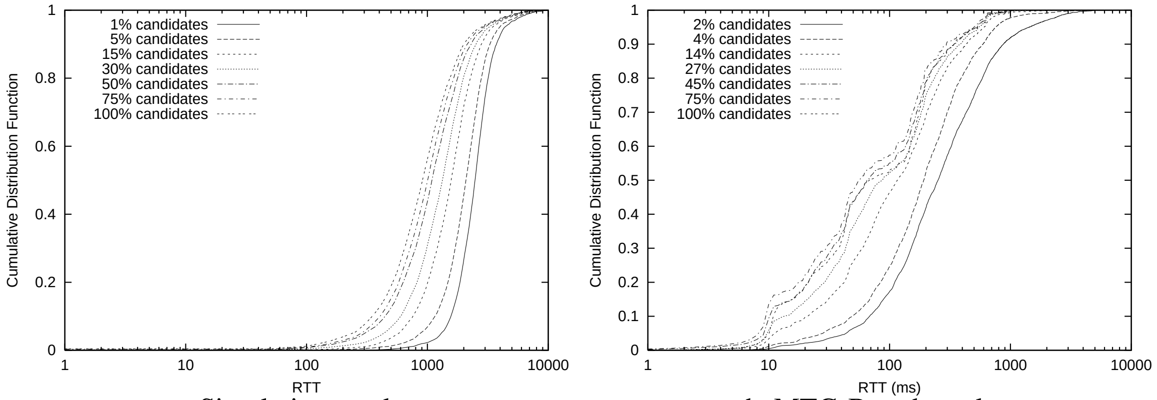
Our results here show that there is a diminishing return in the advantages of having larger number of mirrors. Furthermore, we show that there is a cost, other than financial, to have a large number of mirrors.

4.1 Diminishing Return on the Client Side

We used different mirror placement algorithms in both simulation and Internet MTG-based experiment to observe the improvement in latency between clients and mirrors as the number of mirrors increases. We show our results using two different metrics: the average latency and the latency distribution.

In Figure 2 we show the average RTTs from clients to mirrors as the number of mirrors increases using both Min K-center and random mirror placement algorithms and closest-mirror redirection. The x-axis is the number of mirrors normalized by the total number of candidates; the y-axis is the RTT. For any given number of mirrors, we compute the average RTT between clients and mirrors and the standard deviation from the average RTT. In Figure 2.a, we show the simulation result of Scenario I with 1500 geography-clustered nodes as clients and 225 random placed candidates. In Figure 2.b, we have the MTG-based experiment with the similar mirror-candidate ratio as in the simulation. Both simulation result and the Internet MTG-based result show that by increasing the number of mirrors, the average RTT between clients and mirrors improves, but with diminishing return. When the number of mirrors is beyond 15% of candidate sites, there is almost no perceptible improvements in latency.

We also plot the client latency distributions with different percentages of candidate sites chosen as mirrors. Both simulation Scenario I and MTG-based results are shown in Figure 3. The x-axis is the RTT



a. Simulation result b. MTG-Based result

Figure 3: Client latency distribution for different number of mirrors

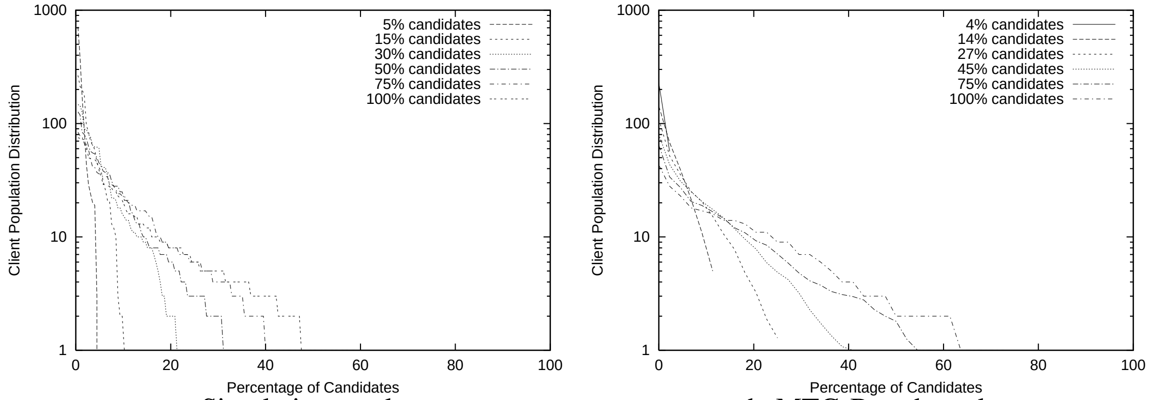
in log scaled; y-axis is the fraction of total pairs of clients. From this figure, we also observe diminishing return for large number of mirrors. In the MTG-based experiment, the curves for 27%, 45%, 75% and 100% candidates show increasingly smaller differences. The same situation happens in simulation as the mirror population grows beyond 30% of the candidate sites.

4.2 Diminishing Return on the Content Provider Side

We are also interested in the gain on the provider side when the number of mirrors increases. We use the server popularity distribution, which gives the number of clients each server receives with the most popular server being the one receiving the most clients, as the metric for measuring the gain on the content provider side. Different number of mirrors are used in both simulation Scenario I and MTG-based experiment. Figure 4 plots the result, where the x-axis is the number of mirrors normalized by the total number of candidates and the y-axis is the client population distribution depicted in log scale. We observe that although the number of mirrors increases, most of the clients are still directed to the most popular mirrors. That means increasing the number of mirrors can not reach the objective of spreading the loads from popular mirror servers. There is still a diminishing return on the content provider side.

4.3 Diminishing Return on Link Usage

The usage of network links is also an important factor when we consider the effectiveness of increasing the number of mirrors. In this experiment, by using Min K-center placement algorithm, we sum up the total hop counts from clients to their closest mirrors with different percentage of candidate sites chosen



a. Simulation result b. MTG-Based result

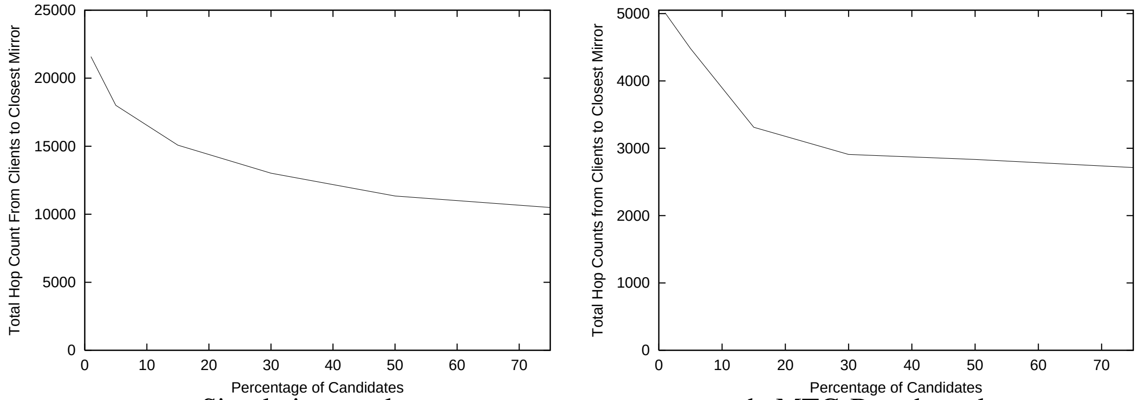
Figure 4: Client population distribution among popular mirrors

as mirrors. If increasing the number of mirrors has a promising effect on reducing the network link usage, we could expect the total hop count to go down with more mirrors placed. However, from both simulation Scenario II and MTG-based results shown in Figure 3, where the x-axis is the number of mirrors normalized by the total number of candidates and y-axis is the total hop count from clients to closest mirrors, we find that link usage does not improve much beyond a certain percentage of candidates chosen as mirrors.

Togethering with the experiment results shown in previous subsections, we observe diminishing returns in terms of latency between clients and mirrors, server popularity distribution, and reduction in traffic as the number of mirrors increases. These results suggest us that we could expect to place certain number of mirrors in the CMP problem to get an acceptable average performance of the mirrors. According to above experiment results, we assume that placing mirrors on 15% candidate sites is a possible solution.

4.4 Consistency Cost

In this subsection we consider another factor of increasing the number of mirrors on the Internet – what is the cost of keeping the contents on all of these mirrors consistent? If we have more mirror servers on the Internet, we can expect that clients can get the requested data more quickly. But trying to keep all the replicas on large number of mirrors consistent could be expensive. In order to study the relationship between number of mirrors and the cost for keeping them consistent, we studied several Web cache consistency approaches.



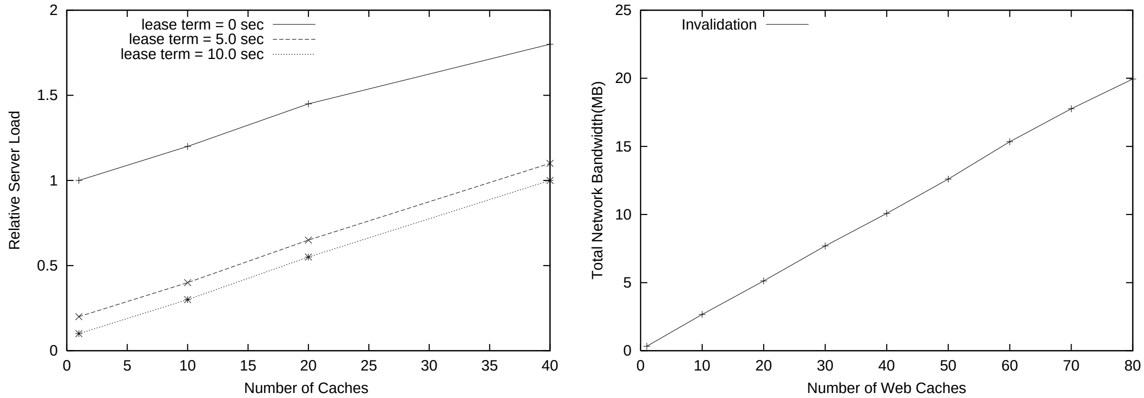
a. Simulation result b. MTG-Based result

Figure 5: Usage of network links vs. number of mirrors

First we studied the `leases` approach for distributed file consistency [7], where `leases` are proposed as a time-based mechanism that provides efficient consistent access to cached data in distributed systems. In this approach whenever a cache stores a page, it requests a `lease` from the server. While the `lease` is valid (within the `lease` term), the page is regarded as valid by its cache. If a cache receives a request for a page with an expired `lease`, it sends a message to the server to determine whether this page is still valid and gets a new `lease` for this page from the server. Figure 6.a presents the relative server load as the function of both the number of Web caches and `lease` terms. The unit relative server load is defined as the server load when the `lease` term is 0 second and the cache number is 1. And all the data used in this figure are from [7]. From this figure we find that the network load at server side in the `lease` approach is increased linearly with the increasing of the Web cache number.

Another effective proposal to achieve Web caching consistency is the `invalidation` approach. In this mechanism, servers send explicit `invalidation` signals to caches when pages are modified. In [12], authors describe a scalable Web cache consistency architecture based on `invalidation` approach. We repeated and modified their simulation. Figure 6.b shows the result. The x-axis is the number of Web caches and y-axis is the total network bandwidth measured in the simulation. Our observation from this figure is that when considering the total network bandwidth as the Web caching consistency overhead, there is also a linear relationship between the Web caches number and the network bandwidth.

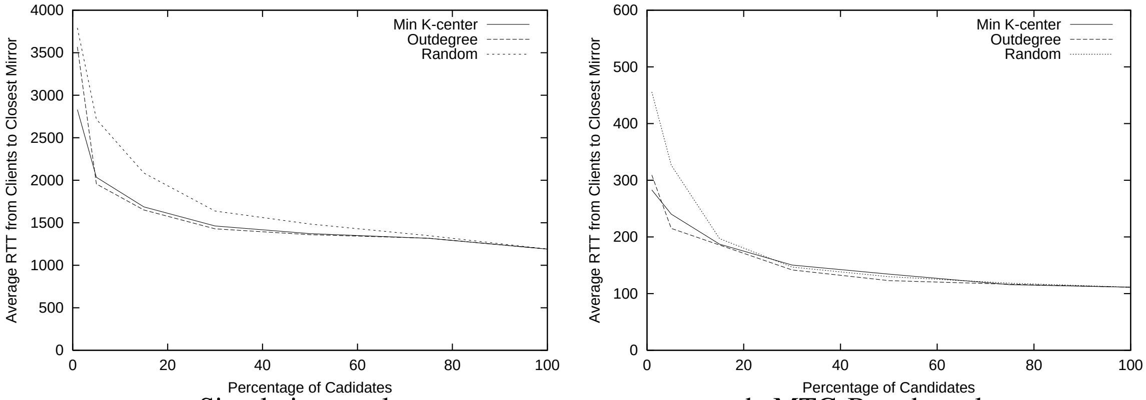
Upon studying these Web cache consistency approaches, we feel that there is also a diminishing return in the advantages of having larger number of mirrors when considering the linearly increased cost for



a. Relative Server Load of the Main Server to Keep Caches Consistent

b. Bandwidth Consumed to Keep Caches Consistent Using Invalidation

Figure 6: Overheads in Keeping Caches Consistency



a. Simulation result

b. MTG-Based result

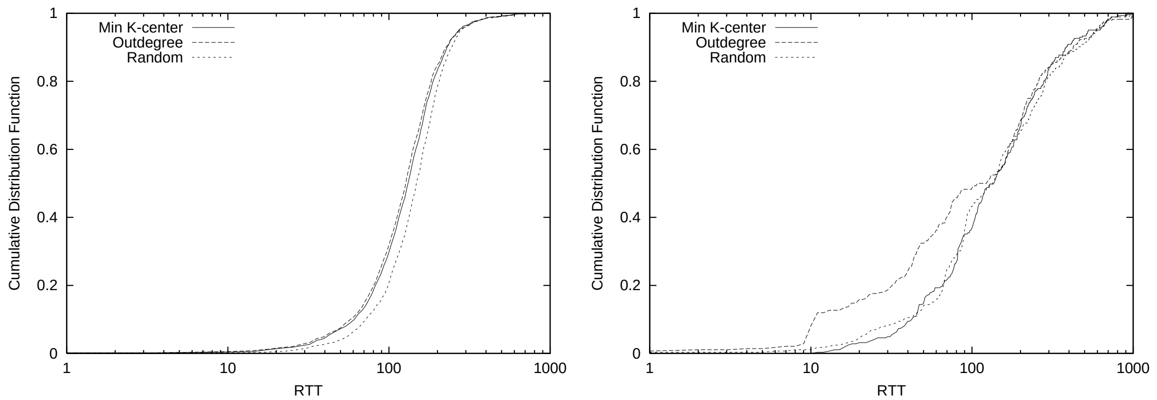
Figure 7: Effect of different mirror placement algorithms on Average latency

keeping them consistency.

5 Comparison of Mirror Placement Algorithms

There are several algorithms or approaches which can be used to determine the placement of mirror servers among the available candidate sites as described in Section 3.1.1 and 3.2.4. In this section we want to evaluate and compare the performances of constrained Min K-center and out-degree placement approaches with the random placement approach, and propose how to use these algorithms or approaches on the constrained mirror placement.

Figure 7 shows the simulation and MTG-based experiment results of average RTT from clients to closest mirrors with different mirror placement algorithms. The x-axis is the percentage of candidates chosen as mirrors; the y-axis is the average RTT from clients to their closest mirrors. From the MTG-based

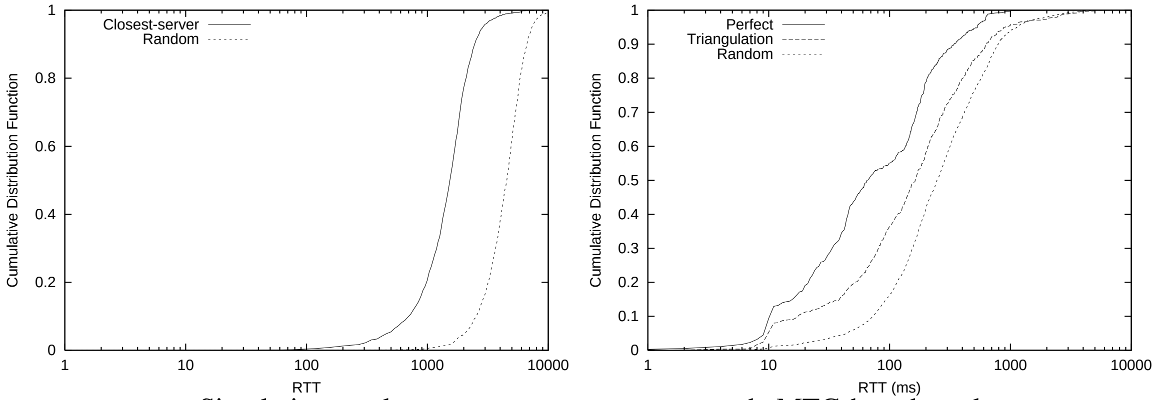


a. Simulation result b. MTG-Based result

Figure 8: Effect of different mirror placement algorithms on client latency distribution

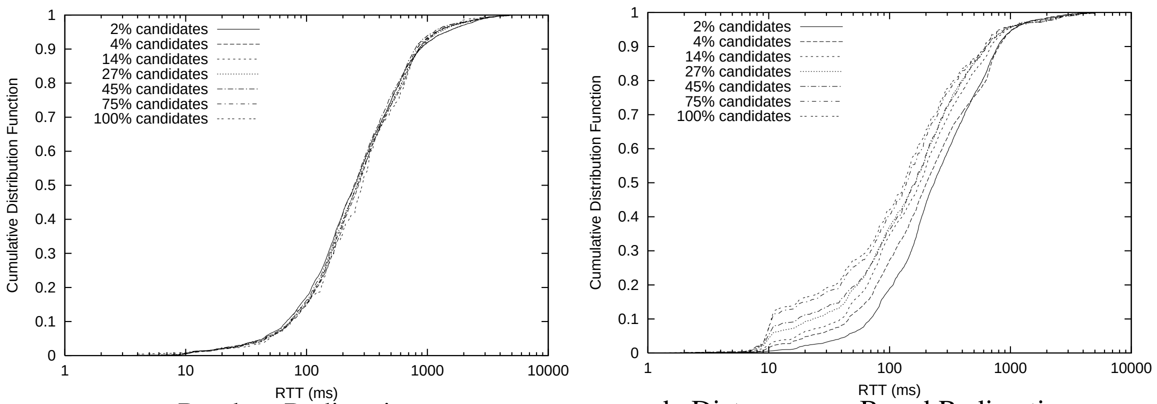
experiment we observe that when the number of mirrors is relative small, the theoretic algorithm of Min K-center outperforms out-degree approach by a little. But all these approaches have better performance than random placement. When more than 15% candidates are chosen, however, random placement performs as well as any of the other placement algorithms. In the simulation experiment, we use Scenario I with 1500 clients and 225 candidates and try to place the same percentage of mirrors out of candidate sites as in the MTG-based experiment. Although, when large number of mirrors are placed, all the placement approaches have similar performance, random placement always has the worst performance. Especially when less than 15% candidate sites are chosen as mirrors, every placement algorithm has much better performance than randomly placing.

Besides the average RTT, we also use the client latency distribution to evaluate the effect of different mirror placement algorithms. In both simulation and MTG-based experiments we place mirrors on 15% candidate sites. Figure 8 shows the fraction of total pairs of clients as a function of log scaled RTT for different placement algorithms. In the simulation figure, we observe again better performance of different placement algorithms than that of random placement. Although MTG-based result shows that random placement has a comparable performance with Min K-center in the case of 15% candidate sites chosen, using out-degree approach we still can get a better performance. According to above experiment results, we propose that in the constrained mirror placement problem, when the number of mirrors needed is relatively small, such as below 15% of candidate sites, using different placement algorithms can reduce the client latency efficiently compared with using random placement. Whereas, employ random placement without knowing the network topology is an efficient way to place large number of mirror servers.



a. Simulation result b. MTG-based result

Figure 9: Latency Distribution under Different Redirections with 50% Candidates as Mirrors



a. Random Redirection b. Distance-map Based Redirection

Figure 10: Latency Distribution under Random and the Distance-map Based Redirection

6 Effect of Redirection Methods

In this section, we look at the effect of the redirection algorithm on the overall performance of mirrors. Prior to this, we have assumed the closest-mirror redirection; however, we need to have a complete distance database between clients and mirrors in order to use the closest-mirror redirection. Such luxury can not always be assumed, and when the complete distance information is unavailable, we have to resort to schemes that may not be much better than random redirection. Therefore, it is important to at least obtain a qualitative understanding of the difference between the closest-mirror redirection and random redirection.

6.1 Random Redirection vs. Closest-Mirror Redirection

Figure 9 presents both the simulation and the MTG-based experiment results of placing mirrors on 50% candidates with random redirection and the closest-mirror redirection. The x-axis is RTT in log scale and the y-axis is the fraction of the total number of clients. It is clear that the closest-mirror selection has superior performance in both the simulation and the MTG-based experiment.

Perhaps not so surprisingly, under random redirection, having more mirrors produces little or no improvements in term of latency between clients and mirrors. Figure 10.a gives the latency distribution between clients and randomly placed mirrors under random redirection in the MTG-based experiment. We observed that the performance of having more mirrors essentially did not improve. Under random redirection, each client is equally likely to be redirected to a close mirror as a faraway mirror, and such decision is independent of the number of mirrors assuming randomly placed mirrors. However, when there is a skewed mirror population where there are more close mirrors than faraway ones, we will likely still see some improvements as more mirrors are added.

6.2 Redirection Using Distance-Maps

Without the actual latency information between mirrors and clients, the redirection itself will be somewhat arbitrary. As we have seen in the previous subsection, random redirection will render the placed mirrors totally ineffective. Therefore, some kind of distance information has to be obtained for mirror placement to succeed.

In Figure 9.b, redirection using a distance-map is plotted alongside random redirection and closest-mirror redirection. We used the 10 most connected `traceroute` gateways to collect latency information and generated a distance-map using triangulation of these latencies. Redirection using this distance-map performs much better than random redirection and relatively good comparing to the closest-mirror redirection. Furthermore, random mirror placement under the distance-map based redirection doesn't exhibit the problem that random redirection has, namely, no improvement in performance as the the number of mirrors increases. It is clear that adding more mirrors under the distance-map based redirection improves performance albeit with a diminishing return as shown in Figure 10.b. Considering the relative small number of hosts used to build this ad-hoc distance-map and the diversity of the client population, we expect that even a coarse-grain distance-map can be used for the purpose of client redirection.

7 Conclusion

In this paper, we take a detailed look at the problem of placing mirrors of a Web site on a restricted set of hosts. We conducted experiments in both simulation and using actual Internet delay data. We observed that there is a diminishing return to placing more mirrors in terms of client latency, server popularity distribution and reduction in traffic. Furthermore, placing more mirrors incurs cost in terms of keeping mirror content consistent and gathering delay data to determine the closest mirror for each client. Overall, increasing the number of mirrors alone is not a cost effective way to improve performance. We found that random placement of a small number of mirrors combined with closest-mirror redirection based on some coarse-grain distance-map can usually provide adequate preformance improvement over the main Web site. For future work, we like to conduct multiple Internet-wide experiments using our proposed mirror placement scheme and evaluate its cost and performance under more realistic settings.

References

[1] Yair Bartal. Probabilistic approximation of metric space and its algorithmic applications. In *37th Annual IEEE Symposium on Foundations of Computer Science*, October 1996.

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

[3] 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.

[4] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance". *ACM/IEEE Transactions on Networking*, 1(4):397–413, Aug. 1993.

[5] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel Gryniewicz, and Yixin Jin. An architecture for a global internet host distance estimation service. *Proc. of IEEE INFOCOM*, March 1999.

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

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

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

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

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

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

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