Efficient Optimal Algorithms for Locating Web Proxies in Linear and Ring Networks

Li-Chuan Chen, Amrinder Arora and Hyeong-Ah Choi  
Department of Computer Science  
The George Washington University  
Washington, DC 20052  
{chen,amrinder,choi}@seas.gwu.edu

Abstract:

With the increasing popularity of the World Wide Web (Web), it is very difficult for a single popular Web server to handle the explosive demand from its clients. Web caching (proxy) is an effective technique to reduce a Web server's load, alleviate Internet traffic, and reduce user's response time when accessing a Web service. A fundamental question is where to locate proxies throughout the network to minimize total access time.

We consider the problem of locating $m$ Web proxies on a network of $n$ nodes. The objective is to minimize the overall cost for accessing Web services to satisfy a given level of demand. In [LDGS98], it was shown that if the network is a linear chain with unidirectional request, then an optimal solution can be computed in $O(n^2m)$ time. We extend and improve their results, by considering both the linear and ring topologies, and also allowing bidirectional requests.

We present $O(n^2 \log m)$ time algorithms which computes optimum proxy locations for unidirectional and bidirectional linear chains, and an $O(n^3(\log m)/m)$ time algorithms for unidirectional and bidirectional rings.

Keywords: Web proxies, optimal locations, algorithms, optimal placement of resources.

Introduction

With ever increasing demand for Internet services, network traffic has grown explosively. This is true for the World Wide Web (WWW) in particular. One recent study [UT99] states that the Internet has generated more than 301 billion dollars in U.S. revenue in 1998 and is growing faster than previously expected. The growth is expected to continue. As load growth exceeds network capacity, network congestion and server overloading will become serious problems in the future and will result in increased traffic delays.

A number of approaches to solve this problem has been proposed recently. The first is to mirror (replicate) popular Web sites in different locations throughout the world. The original Web site's homepage would contain a list of mirror sites. This allows users to choose a site based upon their location. However the user does not typically have access to information about underlying network and server load. This issue was considered in [SBSV98,LM97,MDZ99,JBM99] by taking network latency and server load into account, basing decisions on prior performance, or based on erasure codes.

A second approach is to cluster servers together as a single Web server. Web documents are distributed among servers, and only one Universal Resource Locator (URL) is published to the clients. Since many servers are working together, load balance is the main issue. This has been studied in [NRY97,SNCC96,GGMP95,KMT96,AYHI96].

A third approach is to cache Web objects. Web caching is a mechanism to place copies of frequently accessed Web...
objects closer to the users. Web caching can be classified into three categories:

- **Browser-based caching**: Most of the current popular Web browsers such as Netscape Navigator and Internet Explorer have a built-in cache. This is only useful for one client, who repeatedly accesses the same information.

- **Server-based caching**: An accelerator sits in front of one or more Web servers. If the requested objects is found in the cache, the accelerator returns the cached objects back to the client. Otherwise, it routes the request to a back-end Web server [LAISD99].

- **Proxy-based caching**: A proxy server sits between clients and the original Web server. It is typically closer to the clients than the remote servers. Unlike browser caching and accelerators, a proxy server can serve a large community of users and relieve multiple remote Web server loads.

Both browser-based and server-based caching do little to alleviate both Internet congestion and server load. The cached object in the client-based caching can only be used by a single person for the future of the same request. The drawback of server-based caching is that it only serves a particular Web site and client still has to travel the same distance to reach the server. Like Web servers, a group of proxy servers can work together and act as a single proxy server to reduce proxy server load [MLB96, ACSW96]. An arbitrary proxy takes a client's request and returns the cached object if available. Otherwise the proxy asks other proxies in the group for the requested object. If none contains the object, the request is forwarded to the original Web server. Since proxy-based caching serves a larger community of users, the probability of getting a cached object is much greater. Because it is closer to the users, fewer requests for the same object are going to the original Web server. Thus, the advantages of using Web caching are (1) better user response time, (2) reduced Internet backbone congestion, (3) reduced Web server load, and (4) increased reliability when remote network segments or the original server are down.

One difficulty in Web caching is the possibility of accessing stale Web objects. Cache coherence [LC98, Din96] deals with problem of keeping Web objects consistent with the original copy. Web objects vary in size, unlike traditional caching systems, in modern computer memory systems, where a cache-line is of a fixed size. Replacement algorithms deal with this issue where more than one objects might be removed to replace the current object [Ira97, SWAF96].

In this paper, we consider the problem of locating \( m \) proxy servers throughout the network of \( n \) nodes with the objective of minimizing the overall cost for accessing documents. A similar problem was considered in for linear [LDGS98] and tree topologies [LG1999]. In [LDGS98, LG1999], it was assumed that the target web server is located at one end of the network for a linear topology and the root for a tree topology, and each node requests cache objects from the nearest proxy located between itself and the target web server, i.e., unidirectional. In our model, the unidirectional requirement is relaxed so that a node can request cache objects from proxies in either direction, i.e., bidirectional. The assumption of unidirectional information flow significantly simplifies proxy location, since the direction to the nearest proxy is fixed. The main reason for introducing the bidirectional model is that it provides a more accurate modeling of true network behavior, in which information may flow in any direction in the network. Thus response times will improve and fewer network segments need to traversed.

The rest of this paper is organized as follows. The formulation of our problem is described in Section 2. In Section 3, we consider linear topologies model and present an \( O(n^2 \log m) \) time optimal algorithm. In Section 4, we consider ring topologies and present an \( O(n^3(\log m)/m) \) time optimal algorithm. In Section 5, we use a similar technique, and present an \( O(n^2 \log m) \) time optimal algorithm for unidirectional linear networks model which considered in [LDGS98]. This is an improvement over the \( O(n^2 m) \) time presented in [LDGS98]. We also present an \( O(n^3(\log m)/m) \) time optimal algorithm for unidirectional ring network model. Concluding remarks are given in Section 6.
Problem Formulation

Consider a network $G$ with $n$ nodes, where $V = \{v_i \mid 1 \leq i \leq n\}$. ($v_i$ is sometimes denoted by simply $i$.) Each node $v_i \in V$ is associated with a positive real number $w(v_i)$ denoting the amount of traffic generated by $v_i$. $w(v_i)$ may be the expected number of documents requested by $v_i$ for a certain time period. Each link $(v_i, v_j) \in E$ is associated with a positive real number $d(v_i, v_j)$ denoting the cost of traversing link $(v_i, v_j)$. $d(v_i, v_j)$ may denote the latency of link between $v_i$ and $v_j$, where each link is assumed to be bidirectional and symmetrical, meaning that communication may take place in either direction and that $d(v_i, v_j) = d(v_j, v_i)$. For any path $P$ in $G$ from $v_i$ to $v_j$, define

$$D(P) = \sum_{e \in E(P)} d(e),$$

where $D(P) = 0$ if $v_i = v_j$. If $P$ is the unique path from $v_i$ to $v_j$, $D(P)$ may be denoted by $D(v_i, v_j)$ or simply by $D(i, j)$. The proxy location problem is then formulated as

**Proxy Location Problem (PLP):**

Given a network $G$, a weight function $w : V \rightarrow \mathbb{R}^+$, a distance function $d : E \rightarrow \mathbb{R}^+$, and $m(\leq n)$ proxies, the problem is to partition $V$ into $V_1 \cup \cdots \cup V_m$ and to find a set $U = \{u_1, \ldots, u_m\}$ where $u_i \in V_i$ for $1 \leq i \leq m$ such that (i) a proxy is placed at each $u_i$ ( $1 \leq i \leq m$), (ii) the proxy at $u_i$ is to provide services to all requests from nodes in $V_i$, and (iii) the total cost $C(U)$ of accessing documents from proxies at nodes in $U$, where

$$C(U) = \sum_{1 \leq i \leq m} \sum_{v \in V_i} w(v)D(v, u_i).$$

If there are more than one path from $v$ to $u_i$, it is assumed that the shortest path is used unless stated otherwise.

This problem is closely related to the well-known $k$-median problem (where $k = m$) and the related facility location problem in the field of operations research [MF90]. In our case we make the single-assignment property, namely that each client is supplied from the single facility which is closest. Both of these problems are known to be NP-hard for general cost functions. Therefore it is of interest to consider special cases that can be solved optimally in polynomial time.

Linear Topology

We assume that the nodes are connected in a linear chain using bidirectional links. This implies that for any $u \leq v \leq w$, we have $D(u, v) + D(v, w) = D(u, w)$ and $D(u, v) = D(v, u)$. Define $W(i, j)$ for $1 \leq i \leq j \leq n$ to be

$$\sum_{i \leq k \leq j} w(k).$$

As each proxy will service an interval of nodes, the problem is to partitioning the sequence $\{1, 2, \ldots, n\}$. 

file:///C|/WINDOWS/DESKTOP/w9-papers/Performance/proxyLocation.html (3 of 19) [1/5/2000 4:26:03 PM]
into \( m \) disjoint intervals. For \( i < j \), let \([i, j]\) denote the interval consisting of nodes \( i \) through \( j \). Our problem is then equivalent to finding sequence of \( m \) integer breakpoints, \( x_1, x_2, ..., x_m \), where \( 1 \leq x_i < x_{i+1} \leq n \), where \( x_1 = 1 \) and placing a single proxy at an optimum location within each interval \([x_i, x_{i+1} - 1]\). To handle the last interval, let \( x_{m+1} = n + 1 \).

Given an arbitrary interval \([i, j]\), let \( B(i, j, k) \) be the access cost after placing a single proxy at node \( k \) in interval \([i, j]\) (\( i \leq k \leq j \)), \( l(i, j) \) denote the optimum location of a proxy in the interval \([i, j]\), and \( C(i, j, l) \) denote the minimum cost of placing \( l \) proxies in interval \([i, j]\).

We observe that \( l(i, j) \) can trivially be found by checking all values of \( B(i, j, k) \), where \( k \) varies from \( i \) to \( j \). More formally,

\[
l(i, j) = \text{ArgMin}_{1 \leq k \leq j} B(i, j, k),
\]

This would require \( O(n^3) \) time to compute all \( l(i, j) \), \( 1 \leq i < j \leq n \). Below it is shown that all \( l(i, j) \) can be computed in \( O(n^2) \) time.

**Lemma 1** If \( B(i, j, k + 1) \geq B(i, j, k) \), then \( B(i, j, z + 1) \geq B(i, j, z) \), \( k \leq z \leq j \).

**Proof:**

\[
B(i, j, k + 1) = \sum_{i \leq k' < k + 1} w(k')D(k', k + 1) + \sum_{k + 1 < k' \leq j} w(k')D(k + 1, k')
\]

\[
B(i, j, k) = \sum_{i \leq k' < k} w(k')D(k', k) + \sum_{k < k' \leq j} w(k')D(k, k')
\]

Observe that weight of the node where the proxy-server is placed does not contribute any access cost to the corresponding term. Say \( B(i, j, k + 1) \geq B(i, j, k) \), i.e.,

\[
\sum_{i \leq k' < k + 1} w(k')D(k', k + 1) + \sum_{k + 1 < k' \leq j} w(k')D(k + 1, k') \geq \sum_{i \leq k' < k} w(k')D(k', k) + \sum_{k < k' \leq j} w(k')D(k, k')
\]

\[
\sum_{i \leq k' \leq k} w(k')(D(k', k + 1) - D(k', k)) \geq \sum_{k < k' \leq j} w(k')(D(k, k') - D(k + 1, k'))
\]
\[ \sum_{i \leq k' \leq k} w(k') d(k, k + 1) \geq \sum_{k < k' \leq j} w(k') d(k, k + 1) \]

\[ \sum_{i \leq k' \leq k} w(k') \geq \sum_{k < k' \leq j} w(k') \]

Since \( z \geq k \), and \( \sum_{i \leq k' \leq z} w(k') \geq \sum_{z < k' \leq j} w(k') \)

\[ \sum_{i \leq k' \leq z} w(k') d(z, z + 1) \geq \sum_{z < k' \leq j} w(k') d(z, z + 1) \]

\[ \sum_{i \leq k' \leq z} w(k')(D(k', z + 1) - D(k', z)) \geq \sum_{z < k' \leq j} w(k')(D(z, k') - D(z + 1, k')) \]

\[ B(i, j, z + 1) \geq B(i, j, z), \quad k \leq z < j. \]

End of proof.

Observe that by definition of \( l(i, j) \),

\[ C(i, j, 1) = B(i, j, l(i, j)) \]

**Lemma 2** Given an interval \([i, j]\) of nodes, define the **weighted median** to be the smallest \( k \), \( i \leq k < j \), such that

\[ \sum_{i \leq k' \leq k} w(k') \geq \sum_{k + 1 \leq k' \leq j} w(k') \]

Then, \( l(i, j) \) lies at the weighted median node.

**Proof:**
Suppose that the optimum location of the proxy server is \( k \). Thus, \( B(i, j, k + 1) \geq B(i, j, k) \). Note from the proof of **lemma 1** that this implies that

\[ \sum_{i \leq k' \leq k} w(k') \geq \sum_{k < k' \leq j} w(k') \]

Moreover, \( B(i, j, k - 1) \geq B(i, j, k) \), i.e.,

\[ \sum_{i \leq k' < k-1} w(k')D(k', k - 1) + \sum_{k-1 < k' \leq j} w(k')D(k - 1, k') \geq \sum_{i \leq k' < k} w(k')D(k', k) + \sum_{k < k' \leq j} w(k')D(k, k') \]
Thus, the optimum location of the proxy, i.e., \( l(i, j) \) lies at the weighted median node. End of proof.

Consequently, the following result is obtained.

**Corollary 1** The optimal location \( l(i, j) \) of a single proxy within any interval \([i, j]\), does not depend on the distance function \( d \).

Lemmas 1 and 2 have established that local minima is indeed the global minima for function \( C(i, j, 1) \). Another useful observation here is that optimal proxy position for segment \([i, j]\) is either the same or to the right of the optimal proxy location for interval \([i, j-1]\), i.e., \( l(i, j) \geq l(i, j - 1) \). This leads to a simple algorithm for determining all \( C(i, j, 1) \) and \( l(i, j) \) values in an efficient manner as described in Algorithm 1. Note that the statement \( k++ \) on line 10 of the algorithm, ensures that the total time spent inside the while loop is \( O(n) \) for each value of \( i \); hence, the overall time complexity of Algorithm 1 is \( O(n^2) \).

**Figure:** Linear Topology Proxy Location Algorithm.
Theorem 1. For all $1 \leq i \leq j \leq n$, $C(i,j,1)$ and $l(i,j)$ values can be computed in $O(n^2)$ time.

Using the result in Theorem 1, a simple $O(n^2m)$ dynamic programming algorithm can be obtained as follows. Observe that $C(1,j,k)$ i.e., minimum cost for placing $k$ proxies in the interval $[1,j]$ can be computed by recurring over the last breaking point, i.e., the subsegment covered by the last proxy server. More formally,

$$\min_{1 \leq i \leq j \leq n} C(1,j',k-1) + C(j'+1,j,1)$$

In other words, we break the interval $[1,j]$ into the two intervals $[1,j']$ and $[j'+1,j]$. We place one proxy in $[j'+1,j]$ at cost $C(j'+1,j,1)$ and $k-1$ proxies in $[1,j']$. The resulting solution with Dynamic Programming will have $O(nm)$ entries (for all possible values of $j$ and $k$); with $O(n)$ time spent for calculating each entry using Equation (3). Thus the running time of this approach is $O(n^2m)$. 

---

**Linear Topology Proxy Location**

**Input:** A linear graph $G$ and a weight vector $w(i)$, $1 \leq i \leq n$.

**Output:** $l(i,j)$ and $C(i,j)$ for all pairs $1 \leq i \leq j \leq n$.

1. for $1 \leq i \leq n-1$ do {
2. $k' = i$, $LeftCost = 0$, $RightCost = 0$;
3. $LeftWt = 0$, $RightWt = 0$;
4. for $i \leq j \leq n$
5. while $(LeftWt + w(k')) < RightWt$ {
6. $LeftWt = LeftWt + w(k')$;
7. $LeftCost = LeftCost + LeftWt \cdot D(k', k' + 1)$;
8. $RightCost = RightCost - RightWt \cdot D(k', k' + 1)$;
9. $k'++$;
10. $RightWt = RightWt - w(k')$;
11. }
12. $C(i,j) = LeftCost + RightCost$;
13. $l(i,j) = k'$;
14. if ($j < n$) then {
15. $RightWt = RightWt + w(j + 1)$;
16. $RightCost = RightCost + w(j + 1) \cdot D(k', j + 1)$;
17. }
18. }
19. }
Improved Efficient Algorithm

The algorithm described in the previous section can be further improved to yield a more efficient algorithm. First observe that by changing the segment \([i, j - 1]\) to \([i, j]\), the optimal location of each proxy, either moves to right or remains the same. The same also holds for each break point.

Let's assume for the time being that \(m\) is a power of 2. Later this assumption will also be removed.

Observe that placement of \(m\) proxies introduces \(m - 1\) links as break-points on the paths that are not being used because nodes on the ends of that link are near to separate proxies (by single assignment property, if distance is the same, one proxy is picked in some manner). Thus if \(m\) is a power of 2, \(m - 1\) is odd, and so there is a breakpoint (numbered \(\frac{m}{2}\) from either side) that can be termed the middle breaking point. Note that there are \(2^{p-1}\) proxies on either side of that breaking point.

Let's denote by \(f(i, j, 2^p)\) the optimum location of the middle breaking point. Also we have,

\[
C(i, j, 2^p) = \min_{i \leq j' < j} (C(i, j', 2^{p-1}) + C(j' + 1, j, 2^{p-1}))
\]  

(3)

It is noteworthy that for the first time, both the first and the third parameter of \(C(i, j, l)\) are not 1. Thus any solution with dynamic programming will have at least \(O(n^2)\) entries (unlike \(O(nm)\) as earlier). Observe that

\[
f(i + 1, j + 1, 2^p) \geq f(i, j + 1, 2^p) \geq f(i, j, 2^p)
\]  

(4)

**Figure:** Improved Linear Proxy Location Algorithm.
Based on the result in Equation (5), we now describe Algorithm 2 which computes an optimal solution in $O(n^2 \log m)$ time. In Algorithm 2, the entries of the Dynamic Programming table are computed in diagonal. More precisely, for any two $f(i, j, 2^p)$ and $f(i', j', 2^p)$ with $j - i < j' - i'$, $f(i, j, 2^p)$ and $C(i, j, 2^p)$ are computed before $f(i', j', 2^p)$ and $C(i', j', 2^p)$. For example, entries $f(9, 10, 2)$ and $C(9, 10, 2)$ will be computed before $f(1, 3, 2)$ and $C(1, 3, 2)$. This is controlled by variable $y$ in the algorithm. Observe that the value of UpperBound in an iteration is same as the value of the variable LowerBound in the previous iteration. By using Equation (5), note that total time spent inside the while loop for each iteration of $y$ is bounded by

$$f(n - y + 1, n, 2^p) - f(1, y, 2^p)$$

i.e., total time spent in the while loop for each value of $y$ is $O(n)$. Besides that, Algorithm 2 has 3 nested loops which
run for log \( m, n \), and \( n \) iterations respectively. Thus the time complexity of the algorithm is \( O(n^2 \log m) \).

Now, we will generalize the number of proxies to be \( 2^q \cdot 1 < m < 2^q \). This will remove the previous restriction on \( m \). Let's consider a new linear network \( G' \) with \( n' \) nodes obtained from \( G \) such that \( V' = \{ v_i | n + 1 \leq i \leq n + 2^q - m \} \). For each node \( v' \in V' - V \), define \( w(v') \) to be \( \infty \), and assume \( m' = 2^q \) proxies to be located. Observe that each of dummy node must be a proxy location, since the weight of the demand is very large. Second observation is that \( n' = n + 2^q - m \) and \( m' = 2^q \). From \( 2^q \cdot 1 < m < 2 \cdot 2^q \cdot 1 < 2m \), which implies that \( m' = 2^q = O(m) \). Since \( m < n \), \( n' = O(n) \), and the following result is then established.

\[ \textbf{Theorem 2} \quad C(i, j, m) \text{ and } f(i, j, m) \text{ values, for all } 1 \leq i \leq j \leq n, \text{ can be computed in } O(n^2 \log m) \text{ time.} \]

\section*{Ring Topology}

Next we consider the proxy location problem with a ring topology under the bidirectional model. A ring consists of \( n \) nodes, 1,..., \( n \) joined in a loop (so that nodes \( n \) and 1 are adjacent). As in the linear case, we will work by partitioning the ring into \( m \) linear intervals, \([x_i, x_{i+1}]\), and will place a proxy within each one. We assume that the resulting sequence of breakpoints will be given in clockwise order.

To solve the problem on the ring topology, we can break the ring at an arbitrary node \( i \), and solve the problem on the resulting linear network, using the algorithm 2 described in the previous section. Let \( x_1 = i \) be the first node of the linear intervals after the initial break of the ring. Define \( X_{\text{opt}}(x_1) = \{x_1, x_2, \ldots, x_m\} \) be the sequence of optimal breakpoints for \( F_i \) after breaking the ring at node \( x_1 \). This will produce \( m \) intervals \([x_1, x_2 - 1], [x_2, x_3 - 1], \ldots, [x_m, x_1 - 1]\), where \([x_i, x_{i+1} - 1]\) includes node \( x_i \) and \( x_{i+1} - 1 \). Define \( C(X_{\text{opt}}(x_1)) = C(x_1, x_1 - 1, m) \) to be optimal solution to linear topology with \([x_1, x_2 - 1], [x_2, x_3 - 1], \ldots, [x_m, x_1 - 1]\) break intervals.

Observe that if the optimal ring solution breaks the ring at node \( x_1 \), then \( X_{\text{opt}}(x_1) \) is optimal for the ring. Applying this observation will result in an algorithm whose running time is greater by a factor of \( n \) (by trying each possible breaking point). We will develop an optimal algorithm whose running time is greater only by a factor of \( \frac{n}{m} \) (which is typically much smaller than \( n \)). The algorithm is based on the following property about the structure of optimal breakpoints \( X_{\text{opt}}(x_1) \).

\[ \textbf{Lemma 3} \quad \text{(Interleaving Lemma)} \]

Let \( x_1 \) be an arbitrary node in the ring, and \( X_{\text{opt}}(x_1) \) be the optimum breakpoint sequence after breaking the ring at \( x_1 \). Let \( y_1 \) be an arbitrary node in \( x_1 < y_1 < x_2 \), and \( Y_{\text{opt}}(y_1) \) be the optimum breakpoint sequence after breaking the ring at \( y_1 \), where \( X_{\text{opt}}(x_1) = \{x_1, x_2, \ldots, x_m\} \), and \( Y_{\text{opt}}(y_1) = \{y_1, y_2, \ldots, y_m\} \). We claim that \( x_i \leq y_i < x_{i+1} \) for each \( i, 1 \leq i \leq m \) where \( x_{m+1} = x_1 \).

\[ \textbf{Proof:} \]

Suppose to the contrary that \( X_{\text{opt}}(x_1) \) and \( Y_{\text{opt}}(y_1) \) do not interleave. Then there must be some interval \([x_i, x_{i+1} - 1]\) that contains no element of \( Y_{\text{opt}}(y_1) \), and because both sequences contains the same number of breakpoints, there must be some closest interval \([x_j, x_{j+1} - 1]\) that contains two elements of \( Y_{\text{opt}}(y_1) \).

Let \( i \) be the smallest index such that for any \( i' < i \), \( x_i \leq y_{i'} < x_{i+1} \), but \( y_i \geq x_{i+1} \). We then observe that there exists
Let $C(X_{opt}(x_1))$ denote the cost of $X_{opt}(x_1)$ and $C(Y_{opt}(y_1))$ denote the cost of $Y_{opt}(y_1)$. We will derive a contradiction by constructing two new sequences $W_{opt}(x_1)$ and $Z_{opt}(y_1)$ whose associated costs are $C(W_{opt}(x_1))$ and $C(Z_{opt}(y_1))$ such that

$$C(W_{opt}(x_1)) + C(Z_{opt}(y_1)) \leq C(X_{opt}(x_1)) + C(Y_{opt}(y_1)).$$

This implies that either $C(W_{opt}(x_1)) \leq C(X_{opt}(x_1))$ or $C(Z_{opt}(y_1)) \leq C(Y_{opt}(y_1))$, which is a contradiction to the assumption that both $X_{opt}(x_1)$ and $Y_{opt}(y_1)$ are optimal sequences. Define the sequences $W$ and $Z$ by swapping their respective subsequences in $X_{opt}$ and $Y_{opt}$ so that $w_i$ and $z_i$ interleave. Intuition here is by spreading the two breakpoints (which violate the interleaving property) apart in $X$ and $Y$, we will get the interleaving result. That is constructing $W(w_1)$ and $Z(z_1)$ as follows: $w_q = x_q$ for $1 \leq q \leq i$; $w_q = y_q$ for $i + 1 \leq q \leq k$; $w_q = x_q$ for $j + 1 \leq i \leq m$. $z_q = y_q$ for $1 \leq q \leq i - 1$; $z_q = x_q + 1$ for $i \leq q \leq j - 1$; $z_q = y_q + 1$ for $j \leq i \leq m$.

$$X = \{x_1, \ldots, x_{i - 1}, x_i, x_{i + 1}, \ldots, x_j, x_{j + 1}, \ldots, x_m\}$$

$$Y = \{y_1, \ldots, y_{i - 1}, y_i, y_{i + 1}, \ldots, y_j, y_{j + 1}, \ldots, y_m\}$$

$$W = \{x_1, \ldots, x_{i - 1}, x_i, y_i, y_{i + 1}, \ldots, y_k, x_{j + 1}, \ldots, x_m\}$$

$$Z = \{y_1, \ldots, y_{i - 1}, x_i + 1, x_i + 2, \ldots, x_j, y_{k + 1}, \ldots, y_m\}$$

Observe that $W$ and $Z$ interleave within the interval of interest. Notice that cost for the following intervals cancelled, since they have the same interval.

$$x_q = w_q, \quad q = 1, 2, \ldots, i, j + 1, j + 2, \ldots, m$$

$$y_q = z_q, \quad q = 1, 2, \ldots, i - 1, k + 1, j + 1, \ldots, m$$

$$x_q = z_{q - 1}, \quad q = i + 1, i + 2, \ldots, j$$

$$y_q = w_{q + 1}, \quad q = i, i + 1, \ldots, j - 1, k$$

$$\delta = (C(x_i, x_{i + 1}) + C(y_{i - 1}, y_i) - C(w_i, w_{i + 1}) - C(z_{i - 1}, z_i)) + (C(x_j, x_{j + 1}) + C(y_k, y_{k + 1}) - C(w_j, w_{j + 1}) - C(z_{j - 1}, z_j)).$$

To complete the proof, we will prove the following claim.

**Claim 1** For any $1 \leq x \leq u \leq y \leq v \leq n$,

$$C(x, y) + C(u, v) \leq C(x, v) + C(u, y).$$
Proof of Claim:
Let \( a = l(x, y), \ b = l(u, y), \ c = l(x, v) \) and \( d = l(u, v) \). Because we have \( x \leq u \leq y \leq v \), and these points are weighted medians over these endpoints, it follows that \( a \leq b, \ c \leq d \). The order of \( b \) and \( c \) may vary. By definition of the cost function we have

\[
\begin{align*}
C(x, y) &= B(x, y, a) \\
&= \sum_{x \leq i < a} w(i)D(i, a) + \sum_{a < i \leq y} w(i)D(a, i) \\
C(u, y) &= B(u, y, b) \\
&= \sum_{u \leq i < b} w(i)D(i, b) + \sum_{b < i \leq y} w(i)D(b, i) \\
C(x, v) &= B(x, v, c) \\
&= \sum_{x \leq i < c} w(i)D(i, c) + \sum_{c < i \leq v} w(i)D(c, i) \\
C(u, v) &= B(u, v, d) \\
&= \sum_{u \leq i < d} w(i)D(i, d) + \sum_{d < i \leq v} w(i)D(d, i)
\end{align*}
\]

Figure: Submodular graph for \( C(x, y) + C(u, v) \leq C(x, v) + C(u, y) \).
The first goal is to separate out common sums and eliminate them.

Consider the sum for \( C(x, y) \). Since \( a \leq b \leq y \), we can write this as

\[
C(x, y) = \sum_{x \leq i < a} w(i)D(i, a) + \sum_{a < i < b} w(i)D(i, a) + \sum_{b < i \leq y} w(i)D(i, b) + W(b, y)D(b, a)
\]

\[
C(u, v) = \sum_{u \leq i < b} w(i)D(b, i) + W(u, b)D(d, b) + \sum_{b < i < d} w(i)D(d, i) + \sum_{d < i \leq v} w(i)D(i, d)
\]

\[
C(x, v) = \sum_{x \leq i < a} w(i)D(a, i) + \sum_{a < i < b} w(i)D(b, i) + W(x, a)D(b, a) + \sum_{b < i < c} w(i)D(c, i) + W(x, b)D(c, b) + \sum_{c < i < d} w(i)D(i, c) + W(d, v)D(d, c) + \sum_{d < i \leq v} w(i)D(i, d)
\]

We will show that \( \Delta = C(x, v) + C(u, y) - C(x, y) - C(u, v) \geq 0 \). By forming this expression and cancelling common summations we have
\[ \Delta = \sum_{a < i < b} w(i)D(b, i) + \sum_{b < i < c} w(i)D(c, i) + \sum_{c < i < d} w(i)D(i, c) + \]

\[ W(x, a)D(b, c) + W(x, b)D(c, b) + W(d, v)D(d, c) - \sum_{a < i < b} w(i)D(i, a) - \]

\[ W(b, y)D(b, a) - \sum_{b < i < d} w(i)D(d, i) - W(u, b)D(d, b). \]

Let us assume that \( b \leq c \). The other case is similar. First observe that we can break the sum over the interval \([b, d]\) into two sums over the intervals \([b, c]\) and \([c, d]\) yielding

\[ \sum_{b < i < d} w(i)D(d, i) + W(u, b)D(d, b) = \sum_{b < i < c} w(i)D(c, i) + W(u, b)D(c, b) + \]

\[ \sum_{c < i < d} w(i)D(d, i) + W(u, c)D(d, c). \]

We break \( \Delta \) into three parts and will show that each is positive.

\[ \Delta = \Delta_1 + \Delta_2 + \Delta_3 \]

\[ \Delta_1 = \sum_{a < i < b} w(i)D(b, i) + W(x, a)D(b, a) - \sum_{a < i < b} w(i)D(i, a) - W(b, y)D(b, a) \]

\[ \Delta_2 = \sum_{b < i < c} w(i)D(c, i) + W(x, b)D(c, b) - \sum_{b < i < c} w(i)D(c, i) - W(u, b)D(c, b) \]

\[ \Delta_3 = \sum_{c < i < d} w(i)D(i, c) + W(d, v)D(d, c) - \sum_{c < i < d} w(i)D(d, i) - W(u, c)D(d, c). \]

Note that all the terms in the above equations are positive.

For \( \Delta_2 \) observe that the two sums cancel with one another, and that \( W(x, b) > W(u, b) \) because the interval \([x, b]\) contains the interval \([u, b]\). Thus \( \Delta_2 \geq 0 \).

For \( \Delta_1 \), and observe that the second sum can be bounded above by \( W(a, b)D(b, a) \) to get

\[ \Delta_1 \geq \sum_{a < i < b} w(i)D(b, i) + W(x, a)D(b, a) - W(x, a)D(b, a) - W(a, b)D(b, a) - W(b, y)D(b, a) \]

\[ - W(a, b)D(b, a) - W(b, y)D(b, a). \]
\[ \Delta_3 \geq \sum_{i} w(i)D(i, c) + W(d, v)D(d, c) - W(c, d)D(d, c) - W(u, c)D(d, c) + \sum_{i} w(i)D(b, a) \]

Since \( w(i) \geq \sum_{i} w(i) \), we have \( W(x, a) \geq W(a, y) \) implying that \( \Delta_1 \geq 0 \). Similarly, observe that the second sum can be bounded above by \( (W(c, d) - w(d))D(d, c) \) for \( \Delta_3 \), and we get

\[ \Delta_3 \geq \sum_{i} w(i)D(i, c) + W(d, v)D(d, c) - W(c, d)D(d, c) - W(u, c)D(d, c) + \sum_{i} w(i)D(b, a) \]

Since \( \sum_{i} w(i) \leq \sum_{d} W(d, v) = W(d, v) \) implying that \( \Delta_3 \geq 0 \). Thus \( \Delta \geq 0 \), completing the proof of Claim 1.

By applying the Claim 1 to the first summand (where \( x = y_{j-1} \), \( u = x_j \), \( y = x_{j+1} \), and \( v = y_{j} \)) and then to the second summand (where \( x = x_{j} \), \( u = y_{k} \), \( y = y_{k+1} \), and \( v = x_{j+1} \)), it follows that both summands are nonnegative, implying that \( \delta \geq 0 \), as desired. This completes our proof for Lemma 3.

Using the result of Lemma 3, an optimal algorithm for a ring can be developed such that we pick an arbitrary node \( x_1 \) to cut the ring and compute optimal solution using the \( \text{Linear}_{\text{opt}} \) Algorithm in linear network. This will produce a sequence \( m \) of intervals. By the Interleaving Lemma 3 above, the optimum placement must put breakpoint within each of these intervals. By the pigeonhole principle, the smallest interval has at most \( \left\lfloor \frac{n}{m} \right\rfloor \) possible brake locations. Try making each of these breaks, and invoke the \( \text{Linear}_{\text{opt}} \) Algorithm for the linear topology on each of the resulting linear networks. Return the best of these. The following results is now obtained.

**Theorem 3** The PLP can be solved in \( O\left( \frac{n^3 \log m}{m} \right) \) time for ring networks.
Unidirectional Model

In this section we will consider the proxy location problem in the unidirectional model as considered in [LDGS98], for both chains and rings. In [LDGS98], it was shown that if the network is a linear chain with unidirectional flow, then an optimal solution can be computed in $O(n^2m)$ time. We improve their results by showing that this can be done in $O(n^2 \log m)$ time. We will use the same techniques that we developed in Section 4.

Unlike the bidirectional model, it is assumed that the original Web server is located at node $x_1$ (which is always a proxy) and requests for documents from node $x_i$ proceed to smaller numbered nodes, until the first proxy is encountered. This makes the problem simpler to solve, since proxy locations and breakpoints are identical. The major differences between unidirectional and bidirectional models are the computation of the cost function and finding the proxy location. The computation of $l(i,j)$ and $f(i,j)$ is trivial, because the proxy is always at $i$. For bidirectional cost, $C(i,j)$ involves two costs, namely the cost for the nodes that are on the left hand side of the proxy and the nodes on the right side of the proxy. The cost function $C(i,j)$ in unidirectional is simpler since only right side of the cost involved. Here is the cost function for the unidirectional model

$$C(i,j,1) = B(i,j,i) = \sum_{i \leq k' \leq j} w(k')D(i,k')$$

The resulting modifications to the bidirectional algorithms are straightforward, and are omitted here.

For the ring topology, the cost function in the unidirectional case is also simpler, since the proxy will always be placed at the beginning of the $m$ linear intervals. The main differences arise in the proof of Claim 1. Since $x \leq u \leq y \leq v$, and intervals are overlapping for $(x,y),(u,v),(x,v)$, and $(u,y)$, all the cost from node $u$ to node $v$ cancels except the cost from $u$ to $v$. And since the $W(x,u) < W(x,v)$, we have $C(x,y) + C(u,v) \leq C(x,v) + C(u,y)$ as desired.

Therefore, the following results are established.

Theorem 4  The PLP can be solved in $O(n^2 \log m)$ time for unidirectional linear networks.

Theorem 5  The PLP can be solved in $O(\frac{n^3 \log m}{m})$ time for unidirectional ring networks.

Conclusions

In this paper, we have studied efficient algorithms for determining the optimal locations of Web proxies in the Internet. Our results both improve and extend the results of [LDGS98], which considered only the unidirectional model and the linear network topology. We have studied two models, unidirectional and bidirectional in both the linear array and ring topology. We have presented an $O(n^2 \log m)$ time algorithm which computes optimum proxy locations for a linear network, in both models. We also showed an $O(n^3(\log m)/m)$ time algorithm for both models in the ring topology.

The algorithms that we presented can easily be modified for the problem of optimizing the performance of distributed directory servers. The question is where to replicate the directory service for enterprise systems on the Internet so that services can be closer to the clients. For a large enterprise system, the directory service could easily grow to millions of entries. Examples of the directory service that can be stored in distributed directory servers include: user information, access control, security certificates, applications and network device configuration information.

One interesting direction of future work is the performance analysis of distributed Web proxy locations in the...
presence of faults. In particular, I will consider the problem that one or more proxies fail. One trivial solution is to recompute an optimal solution for the remaining number of available proxies. However, this could involve moving each of the remaining proxies to a new location, which is clearly undesirable. A much more reasonable strategy would be to move only a constant number of proxies for each failure. However, this will necessarily imply that as faults continue, our solutions will be suboptimal.

Bibliography

ACSW96
A hierachical internet object cache.
In USENIX Proceedings, 1996.

AYHI96
D. Andresen, T. Yang, V. Holmedahl, and O. Ibarra.
Sweb: Towards a scalable www server on multicomputers.

Din96
Partl Dingle.
Web cache coherence.
In Fifth International World Wide Web Conference, 1996.

GGMP95
Michael Garland, Sebastian Grassia, Robert Monroe, and Siddhartha Puri.
Implementing distributed server groups for the world wide web.

Ira97
Sandy Irani.
Page replacement with multi-size pages and applications to web caching.

JBM99
M. Luby J. Byers and M. Mitzenmacher.
Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads.
In IEEE Infocomm '99, March 1999.

KMT96
Dias Kish, Mukherjee, and Renu Tewari.
A scalable and highly available web server.
In COMPCON'96, pages 85-92, 1996.

LAISD99
Design and performance of a server accelerator.
In IEEE Infocom '99, March 1999.

LC98
Chengjie Liu and Pei Cao.
Maintaining strong cache consistency in the world-wide web.

LDGS98
Bo Li, Xin Deng, Mordecai J. Golin, and Kazem Sohraby.
In The 8th IFIP Conference on High Performance Networking (HPN'98), 1998.
LGI+99
Bo Li, Mordecai J. Golin, Giuseppe F. Italiano, Xin Deng, and Kazem Sohraby.
On the optimal placement of web proxies in the internet.

LM97
Steve Lewontin and Elizabeth Martin.
Client side load balancing for the web.

MDZ99
A. Myers, P Dinda, and H Zhang.
Performance characteristics of mirror servers on the internet.
In IEEE InfoCom ’99, March 1999.

MF90
P. B. Mirchandani and R. L. Francis.
Discrete location theory.

MLB96
Radhika Malpani, Jacob Lorch, and David Berger.
Making world wide web caching servers cooperate.

NRY97
Data distribution algorithms for load balanced fault-tolerant web access.

SBSV98
M Sayal, Y Breitbart, P Scheuermann, and R Vingralek.
Selection algorithms for replicated web servers.

SNCC96
Rahul Simha, B. Narahari, H-A Choi, and Li-Chuan Chen.
File allocation for a parallel webserver.
In IEEE Int. Conf. High Performance Computing, December 1996.

SWAF96
C. Standridge S. Williams, M. Abrams, G Abdulla, and E. Fox.
Removal policies in network caches for world-wide web documents.
In Proceedings of ACM SIGCOMM, August 1996.

UT99
of University and Texas.
Center for research in electronic commerce.
In The Internet Economy Indicators, June 1999.
Efficient Optimal Algorithms for Locating Web Proxies in Linear and Ring Networks

This document was generated using the LaTeX HTML translator Version 98.1p1 release (March 2nd, 1998)


The command line arguments were:
latex2html -split 0 -no_math -html_version 3.2,math proxyLocation.tex.

The translation was initiated by Hyeong-Ah Choi on 1999-11-22

Footnotes

... Chen

The MITRE Corporation, Enterprise-Service Wide Department

Hyeong-Ah Choi
1999-11-22