Lecture 8b: Proxy Server Load Balancing
March, 2004
Arthur Goldberg
Computer Science Department
New York University
artg@cs.nyu.edu
Load Balancing

• Problem: Single physical Origin or Proxy Server may not be able to handle its load
• Solution: install multiple servers and distribute the requests.
• How do we distribute requests among the servers?
DNS Round Robin

- DNS is configured so multiple IP Addresses correspond to a single host name
  - multiple type “A” records in DNS Database
    A harpo 10.0.0.15
    A harpo 10.0.0.16
    A harpo 10.0.0.17

- Modify the DNS server to round-robin through the IP addresses for each new request

- This way, different clients are pointed to different servers
Problems with DNS Round Robin

• Not optimal for proxy servers
  – cache content is duplicated (why?)
  – multi-tier proxy arrangement won’t work if cookies are used
  – load is not truly balanced
    • assignment is at DNS lookup level, *not* HTTP request level

• Failures are seen by the client (why?)
ICP
Internet Cache Protocol

- Used for querying proxy servers for cached documents
- Typically used by proxy servers to check other proxy server’s cache
- Could be used by clients however
- RFC 2186, 2187
ICP

- ICP request has desired URL in it
- send via UDP to other proxy servers
- Other proxy servers respond “HIT” or “MISS”
- Works better in LANs than Internet (why?)
- Might IP multicast help?
Problems with ICP

• ICP queries generate extra network traffic
• Does not scale well
  – more proxy servers = more querying
• Caches become redundant
Non-redundant Proxy Load Balancing

• Proxy selection based on a hash function
• Hash value is calculated from the URL
• Use resulting hash value to choose proxy
• Use Host name in hash function to ensure request routed to same proxy server (why?)
Cache Array Routing Protocol (CARP)

- Hash-based proxy selection mechanism
- No queries
  - hashing used to select server
- Highly scalable
  - performance improves as size of array increases
  - automatically adjusts to additions/deletions of servers
- Eliminates cache redundancy
- No new protocols!
How CARP Works

• Given an array of Proxy servers
• Assume array membership is tracked using a membership list
• A hash value $H_s$ is computed for the name of each proxy server in list (only when list changes)
• A hash value $H_u$ is computed for the name of each requested URL
• For each request, a combined hash value $H_c = F(H_s, H_u)$ is computed for all servers
• Use highest $H_c$ to select server
CARP: Hierarchical Routing

- One server acts as director using Hash routing.
- Cache hit rate is maximized (why?)
- Single point of failure (use DNS RR?)
CARP: Distributed Routing

- Requests can be sent directly to ANY member of the Array.
- Route request to best score if not me.
- Don’t cache response if redirected
CARP Features

• Assume the membership stays the same
• Then a given URL always maps to the same Proxy (because the hash functions are deterministic)
  – Thus, a given page always resides in the same proxy
  – So caching works
  – And pages are not stored redundantly
• When a membership of size n changes by one, only $1/n$ th of the URLs are remapped
CARP Example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jericho1</td>
<td>13</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Jericho2</td>
<td>8</td>
<td>9</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Jericho3</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Jericho4</td>
<td>28</td>
<td></td>
<td>7</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

Note the distribution of URL across servers
**CARP: adding a new server**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jericho1</td>
<td>13 5</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Jericho2</td>
<td>8 9</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Jericho3</td>
<td>5 7</td>
<td>4</td>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Jericho4</td>
<td>28 4</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Jericho5</td>
<td>14 2</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

A 5th server is added and effects only 1/5 of the existing mappings
The CARP Hash Functions

• Host (server) Hash
  – Computations use 32 bit UNSIGNED integers

\[ H_S = 0; \quad \text{// initially} \]

for each character \( C_i \) in host name

\[ H_S += R(H_S, 19) + C_i \]

// where \( R(x, n) ::= \text{logical left rotate } x \text{ by } n \)

End for

\[ H_S += H_S * 0x62531965 \]

\[ H_S = R(H_S, 21) \]
The CARP Hash Functions

• URL Hash
  – Computations use 32 bit UNSIGNED integers
  $H_U = 0$;  // initial $H_U = 0$
  for each character $C_i$ in URL
    $H_U += R(H_U, 19) + C_i$
  End for
The CARP Hash Functions

- Combining Hash Function
  - Again, all computations are performed using 32-bit unsigned integers

\[
\begin{align*}
H_C &= H_U \oplus H_S \quad \text{// [exclusive OR]} \\
H_C &= H_C \times 0x62531965 \\
H_C &= R(H_C, 21)
\end{align*}
\]
The CARP Membership Table

The format of the table is:

# This information is the **Global Information** given once per table
Proxy Array Information/<Version number>
ArrayEnabled: <0 | 1>
ConfigID: <opaque string>
ArrayName: <opaque string>
ListTTL: <minutes until next check>
<CR>LF>
# The following fields are given for **EACH member** of the Array
<name> <IP addr> <listening port> <table URL> <agent str>
<statetime> <status UP | DOWN> <load factor> <cache size>
CARP Membership Table
Global Information

• Applies to the entire Array
• The **ListTTL** is most important field. It gives time in seconds that this copy of table is valid. After this time, a new copy of the table must be obtained.
• Global fields are separated by `<CR><LF>`
• Global Information is separated from member information by blank line (`<CR><LF>`)
CARP Membership Entries

• Fields in the membership record for a member are separated by spaces
• Member records are separated by <CR><LF>
• Important fields are:
  – Name: the host name of this proxy server
  – IP Addr: IP address for server (if not present, resolve host name)
  – Listening Port: TCP port this proxy is listening on
CARP Membership Entries

- Table URL
  - URL for membership table
- Agent String (informational)
- Statetime
  - how long this member has been in array and current state
- Status (UP | Down)
  - member accept requests, or refused connection on last request
- Load Factor
  - relative amount of total load that server can handle
- Cache Size (informational)