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 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?)

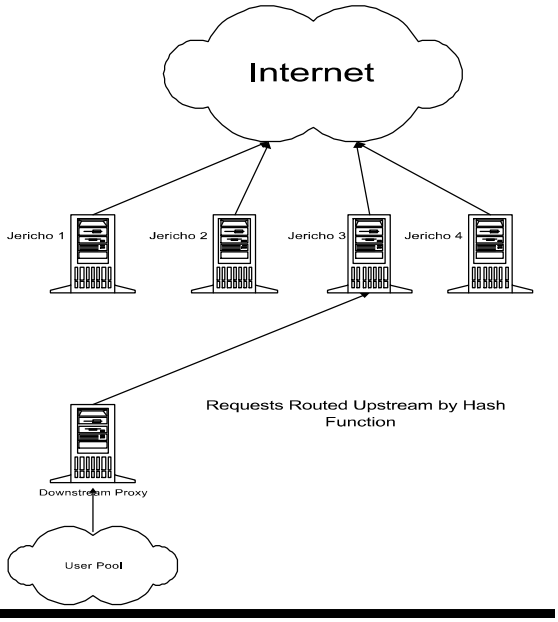
Cache Array Routing Protocol (CARP)

- 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

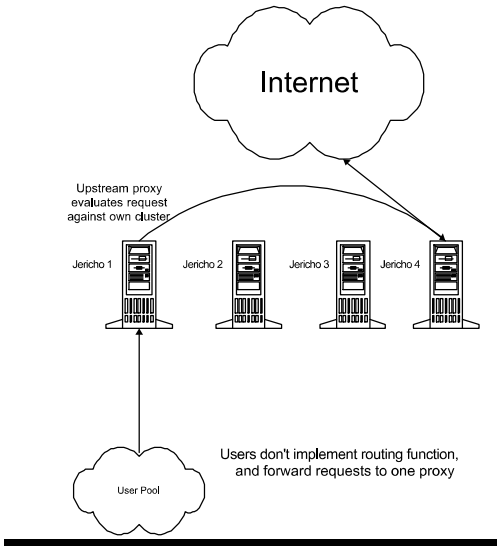
- Given an array of Proxy servers
- Assume array membership is tracked using a membership list
- A hash value H_s is computed for the name of each proxy server in list (only when list changes)
- A hash value H_u is computed for the name of each requested URL
- For each request, a combined hash value $H_c = F(H_s, H_u)$ 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

Proxy	Hash	www.microsoft.com	www.yahoo.com	www.msn.com	www.ibm.com
		19	14	5	2
Jericho1	13	5	6	10	4
Jericho2	8	9	2	7	5
Jericho3	5	7	4	3	10
Jericho4	28	4	7	8	1

Note the distribution of URL across servers

CARP: adding a new server

Proxy	Hash	www.microsoft.com	www.yahoo.com	www.msn.com	www.ibm.com
		19	14	5	2
Jericho1	13	5	6	10	4
Jericho2	8	9	2	7	5
Jericho3	5	7	4	3	10
Jericho4	28	4	7	8	1
Jericho5	14	2	9	4	6

A 5th server is added and effects only 1/5 of the existing mappings