Web Caching

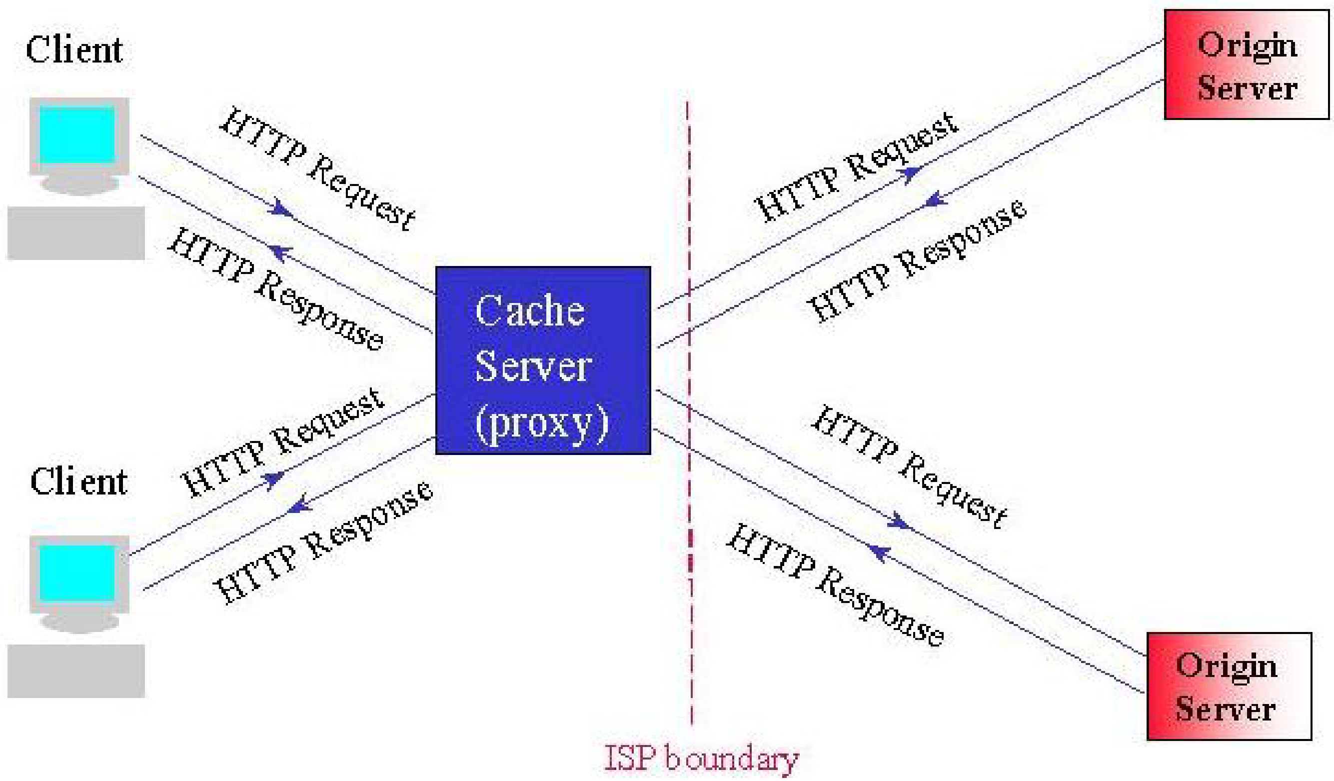
Chapter 11 από:

“Web Protocols and Caching”, B. Krishnamurthy, J. Rexford

Introduction

- The rapid growth of WWW
  - increased network traffic
  - Increased user-perceived latency
- WWW applications in contrast with other internet based applications (e.g. FTP) are very sensitive to delay
- Caching was the first major technique that attempted to reduce user-perceived latency and transmission of redundant traffic
- The traffic pattern showed that: many clients in an organization appeared to visit the *same few* sites
- Caching became a major industry within few years

Cache Server (Proxy Server)

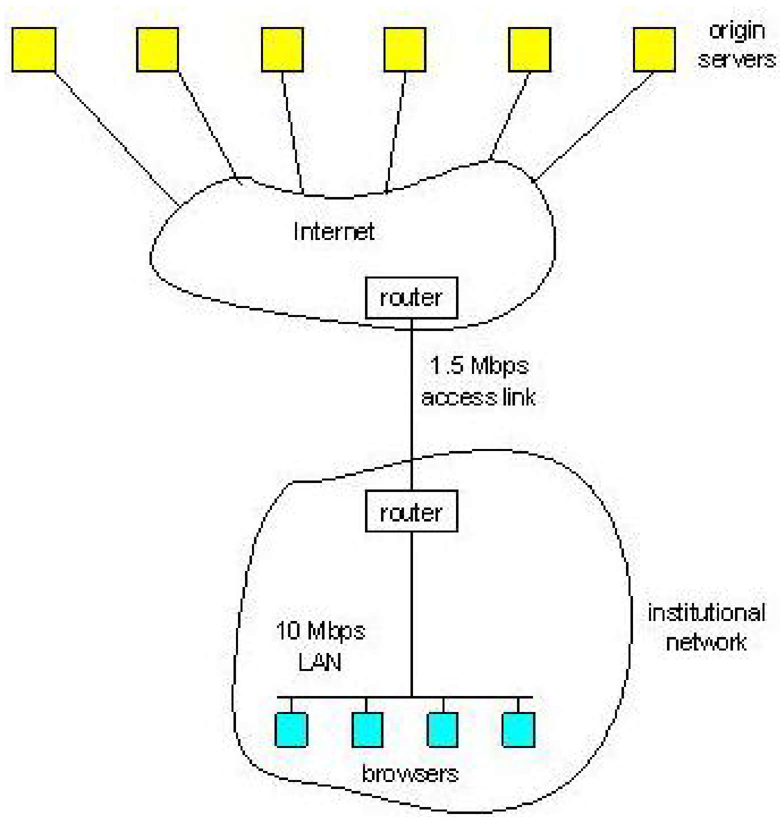


- The Cache Server is both a server and a client

Why Caching in the Network

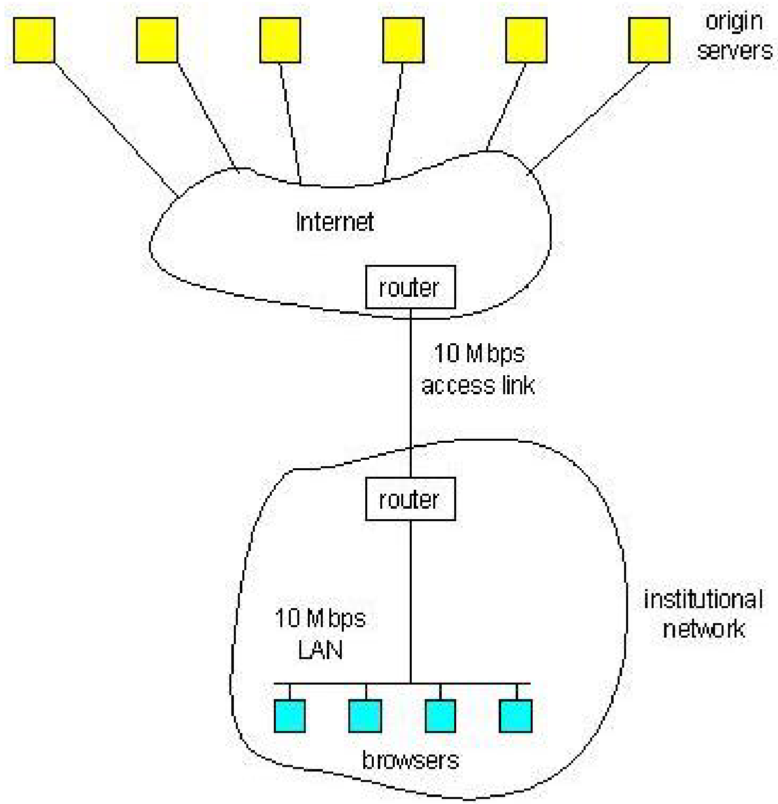
- Reduce access latency
  - By avoiding slow (or congested) links between client and origin server
- Reduce bandwidth consumption
  - By reducing the access to the backbone network
- Server load balancing
  - Spread load of overloaded origin server to the caches
  - Inexpensive server
- Improved data availability
  - When the server or a link to the server is down

Example: No Cache, Insufficient Access Rate



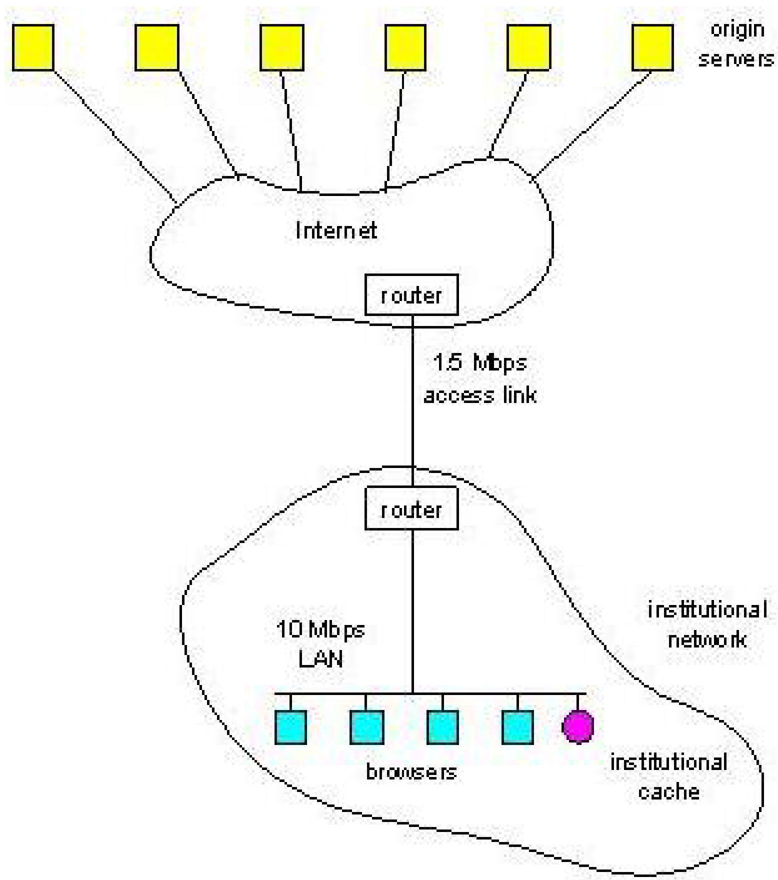
- Assumptions
  - Object Size :100Kbits
  - Request rate: 15 requests/sec.
  - Server – router delay: 2 sec.
- Consequences
  - LAN utilization: 15%
  - Access Link utilization: 100%
  - Total Delay :
  - Internet delay + access delay + LAN = 2sec + minutes + milliseconds
- Conclusion
  - Must increase access link

Example: No Cache, Sufficient Access Rate



- Increased Access Link: 10 Mbps
  - LAN utilization: 15%
  - Access Link utilization: 15%
  - Total Delay :
  - Internet delay + access delay + LAN = 2sec + milliseconds + milliseconds
- Conclusion
  - Reduced delay
  - Higher fee for access link

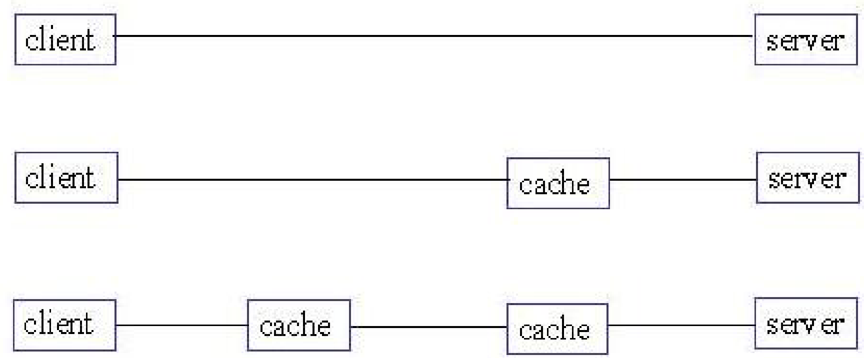
Example: Access Rate 1.5Mbps + Cache



- Cache Hit Rate: 40%
  - Access Link utilization: 60%
  - Delay on Cache Miss
  - Internet delay + access delay + LAN = 2sec + tens of milliseconds + millisec.
  - Delay on Cache Hit
  - LAN delay = milliseconds
  - Expected Delay
  - 0.4*Delay on Hit + 0.6*Delay on Miss $\approx$ 1.2 sec
- Conclusion
  - Lowest average delay
  - Lower fee for access link
  - Less loaded Origin Server

Cache Chaining

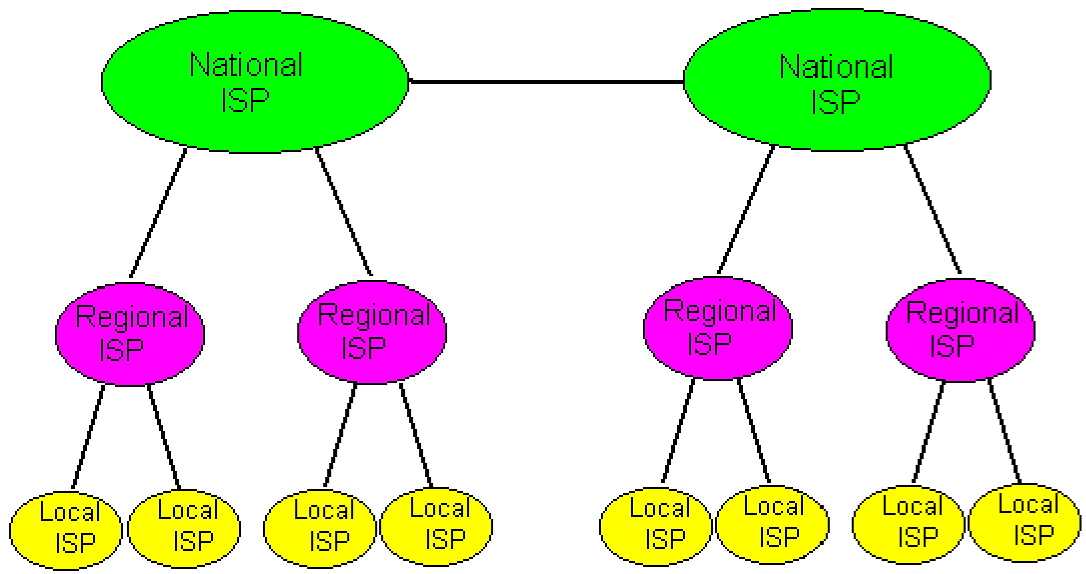
- No Cache
- One Cache
- Chain of Caches



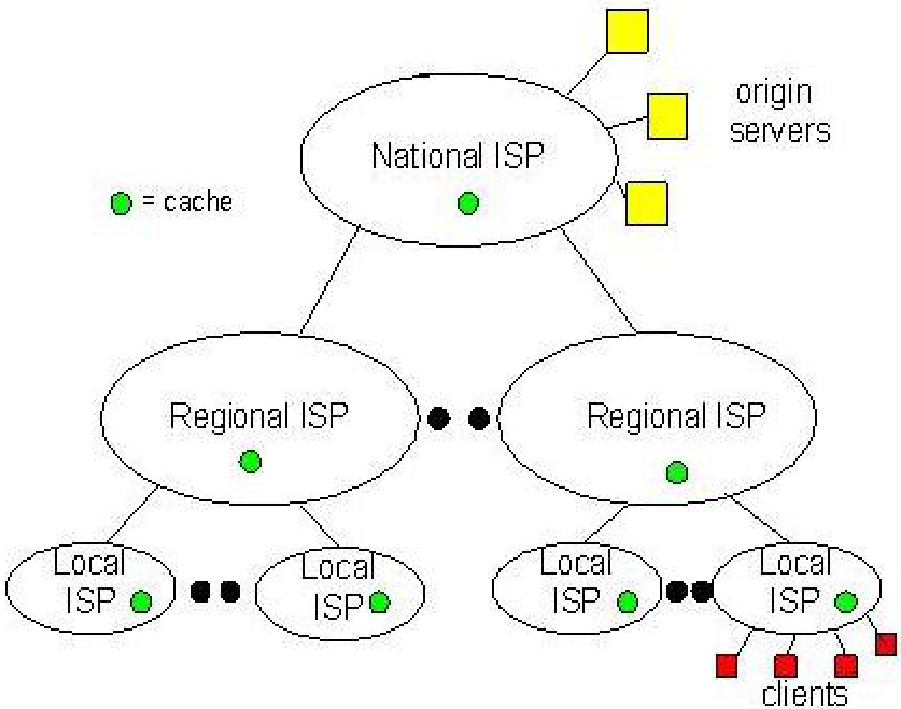
- Communication along chain can be over HTTP
- Cache Hierarchies use Chaining

Hierarchical Caching (1/2)

- The topology of internet is loosely hierarchical
  - National, Regional, local/institutional ISP's



Hierarchical Caching (2/2)



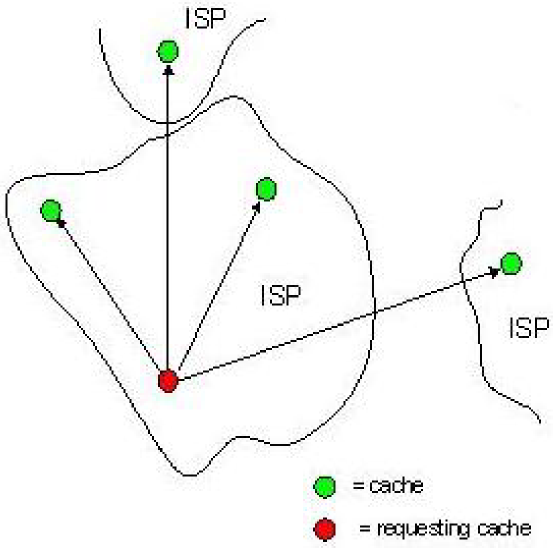
Illustrative Hit Probabilities

- National Cache: 0.25
- Regional Cache: 0.33
- Local Cache: 0.4

- Probability that object is found in institutional cache: 0.4
- Probability that object is found in Regional cache: 0.33 * (1-0.4) = 0.2
- Probability that object is found in Regional cache: 0.25 * (1-0.4-0.2) = 0.1
- Probability that object is found in the cache hierarchy: 0.4 + 0.2 + 0.1 = 0.7

Cooperative Caching

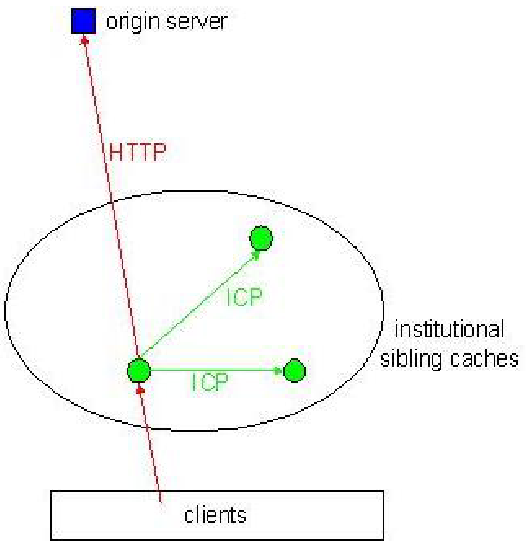
- With pure hierarchical caching, each client and cache, points to at most one other cache
- With Cooperative Caching, a cache can **directly** obtain an object from of many neighboring caches. The neighboring caches can be:
  - In the ISP (sibling caches)
  - Or outside the ISP
- Two popular Cooperative Cache Schemes
  - ICP (Internet Cache Protocol)
  - CARP (Cache Array Routing Protocol)



Overview of ICP

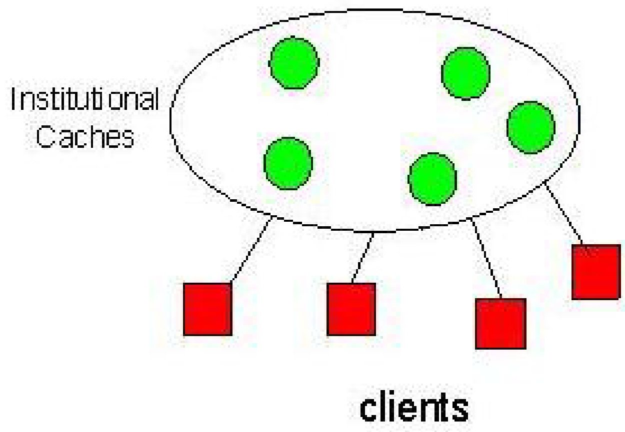
- ICP is primarily used in a cache mesh to locate specific Web objects in neighboring caches.
- When one cache queries another there are three possible outcomes
  - ICP hit message returned
  - ICP miss message returned
  - No response (proxy server down or network connection down)
- When an ICP cache cannot fulfill a request from its own cache then:
  - Queries all neighbors with ICP
  - Obtains the object from first neighbor to respond with a hit
  - Stores a copy of the object and forwards a copy to the requestor
  - If there are no hits, forwards the request to the parent in the hierarchy or to the origin server
- Cache waits up to 2 seconds for response to query

ICP Example



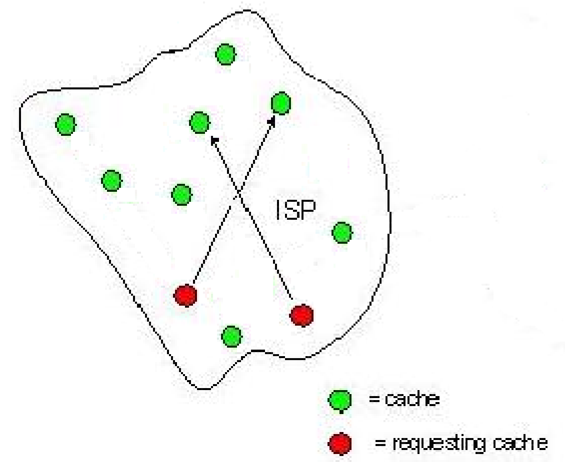
- Client contacts an institutional cache, which does not have the object
- Cache uses ICP to query its siblings
- If a sibling has the object, cache retrieves object from sibling and forwards it to client
- If no sibling responds with a Hit, cache obtains object directly form the origin server

Drawbacks of ICP



- ICP message overhead: whenever there is a miss, each sibling must process an ICP request and response message
- Replication of objects: popular objects get replicated in all the caches

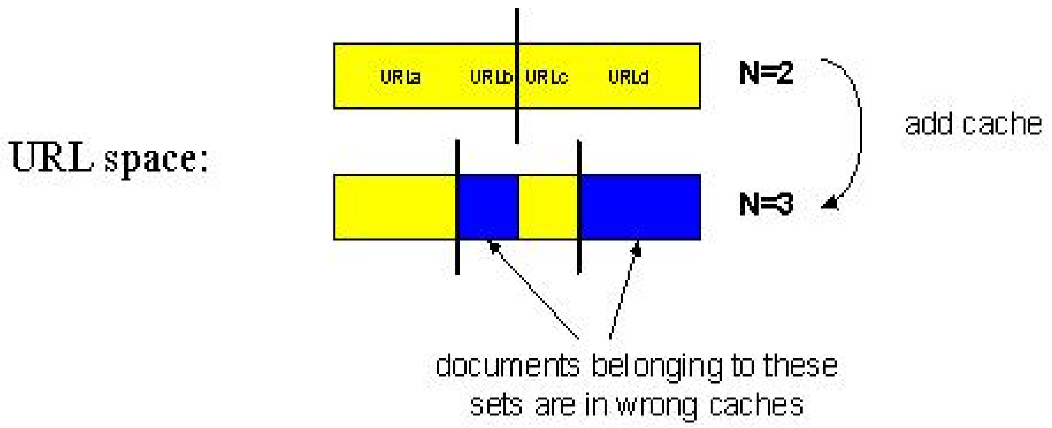
Hash Routing



- Choose a hash function h() which maps URLs to a hash space
  - For example:
    - Let hash space be {1,…,60}
    - Let h() be the sum of the ASCII representation of the characters in the URL, modulo 60
- Partition hash space: one set for each sibling
  - Client hashes URL, determines set to which hashed URL belongs, and sends request to corresponding sibling
    - Example: N=2 Caches, set for cache 1= {1,…,30}, set for cache 2 = {31,…,60}. If h(URLa) = 35, then client sends HTTP request to cache 2
  - If sibling does not have the object, it obtains object from origin server, stores a copy, and forwards a copy to the client.
- Each object resides in at most one sibling
- Client is immediately directed to the correct sibling

Hash Routing Drawbacks

- When a cache is added or removed:
  - The correspondence of a URL to particular cache has to change
  - A cached object can reside in the wrong sibling cache.



- Some caches become more loaded than others

CARP (Cache Array Routing Protocol)

- Alternative to ICP
- Uses a variant of Hash routing
- All queries done over HTTP
  - No new appliaction-layer protocol such as ICP
  - Can take advantage of HTTP/1.1 rich set of headers

Other Cache related protocols

- Cache Digest Protocol
  - Extension provided to ICP
  - Basic idea: permit the exchange of a terse description (*digest*) of a cache's contents

- Web Cache Coordination Protocol (WCCP)
  - WCCP specifies interactions between one or more routers (or Layer 3 switches) and one or more web-caches.
  - The purpose of the interaction is to establish and maintain the transparent redirection of selected types of traffic flowing through a group of routers.
  - The selected traffic is redirected to a group of web-caches with the aim of optimizing resource usage and lowering response times.

Caching Challenges

- Cache Consistency
  - Caches often must guess whether a stored object is stale or fresh
- Hit Counts
  - Caches can cause hit count calculations to fail
- Access Control
  - How do you make sure that the seller of the document gets paid?
  - Legal and security restrictions
- Cache Storage Management
  - Caches must achieve high hit probabilities

Cache Consistency

- A cache may have to ensure that the cached response is still valid before returning it to a client requesting the resource
- Target: Minimize the amount of work required to verify consistency
- Strong Consistency:
  - The caching proxy sends a revalidation request each time a hit occurs
- Weak Consistency:
  - The proxy uses a heuristic to decide whether the cached response is still fresh, without consulting the origin server
  - lease-based heuristic, time-base heuristic

Weak Consistency

- Leased-based approach
  - The cache agrees to store a response for a fixed amount of time (lease - period) without revalidating
  - The server promises to notify the cache if a cached resource changes within the leased period
  - If the lease period expires, the cache can revalidate the resource and decide to renew the lease
- Time to Live approach
  - Responses have expiration time (TTL) associated with them.
  - When the interval passes, the responses are considered stale
  - During TTL period, the cache does not revalidate the response, saving bandwidth at the risk of staleness

Example (1/2)

- Conditional GET:

Cache server obtains object from origin server:

1) Request from cache to origin server:

GET /~ross/index.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg

2) Response from origin server to cache:

HTTP/1.0 200 OK
Date: Wed, 12 Aug 1998 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: **Mon, 22 Jun 1998 09:23:24**
Content-Type: text/html

data data data data data

Later, cache uses the "conditional GET" to determine whether the cached object is stale.

3) Request from cache to origin server - the "conditional GET":

GET /~ross/index.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
If-modified-since: **Mon, 22 Jun 1998 09:23:24**

4) Response from origin server to cache:

HTTP/1.0 304 Not Modified
Date: Wed, 12 Aug 1998 15:48:58
Server: Apache/1.3.0 (Unix)

(empty entity body)

Example (2/2)

- A cache can provide strong consistency by using the conditional GET for every request
  - Wasteful
- Update Heuristic (TTL-based)
  - Hold document for a time proportional to the known lifetime of object:

    Known-lifetime = date – last modified

    Documents which have not changed for a long time are unlikely to change
  - 50% rule: example, receive response

```
HTTP/1.0 200 OK
Date: Mon, 22 Jun 1998 09:23:24
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 8 Jun 1998 09:23:24
Content-Type: text/html
```

  - Hold document for seven days, until 29, June 1998 09:23:24

Hit Counts

- Not all servers are interested in having cached content delivered by proxies
- A Web site, hosting advertisements needs to determine the number of hits to the page hosting the advertisement
  - Advertisers want to know the exposure of their advertisements
  - Billing to advertisements is based on the hit counts
  - Departments within a company receive funding for their web-servers based on their hits
- Caching prevents sites from getting accurate access counts
- Sites often employ “cache busting” (=mark cacheable objects as non-cacheable)
- Solution: hit metering (unsuccessful)
- Advertising sites are encouraged to mark only HTML files as non-cacheable

Caching and Authentication

- With HTTP/1.1 it is possible to cache information that is for sale
- Whenever origin server distributes objects, it includes in the response the header `Cache-control: proxy-revalidate`
- Cache servers, cache the object and tag it with “proxy-revalidate”
- Later when a request for object arrives at the cache, cache sends a conditional GET to origin server.

Cache Storage Management

- Once the cache is full, objects must be removed to make room to cache new responses
- Typical issues taken into account by replacement algorithms:
  - Cost of fetching an object
  - Cost of storing an object
  - The number of accesses to the object in the past
  - The probability of the object to be accessed in the near future
  - Expiration Time
- One Level Algorithms: use only one metric
- Two Level Algorithms: Use a combination with primary and secondary metrics

Cache Replacement Algorithms (1/2)

- Least Recently Used (LRU): remove objects that have not been accessed for a long time.
  - Examples:
    - $Value_{LRU}$ = days_since_last_ access/ N
    - Remove objects with lowest $Value_{LRU}$ when disk space is exhausted
  - *Weighted by retrieval time*
    - record retrieval time of each object: t_r
    - $Value = Value_{LRU} * \log(t_r + 1)$
  - *Weighted by object size*
    - $Value = Value_{LRU} * \log(\text{Object Size} + 1)$

Cache Replacement Algorithms (2/2)

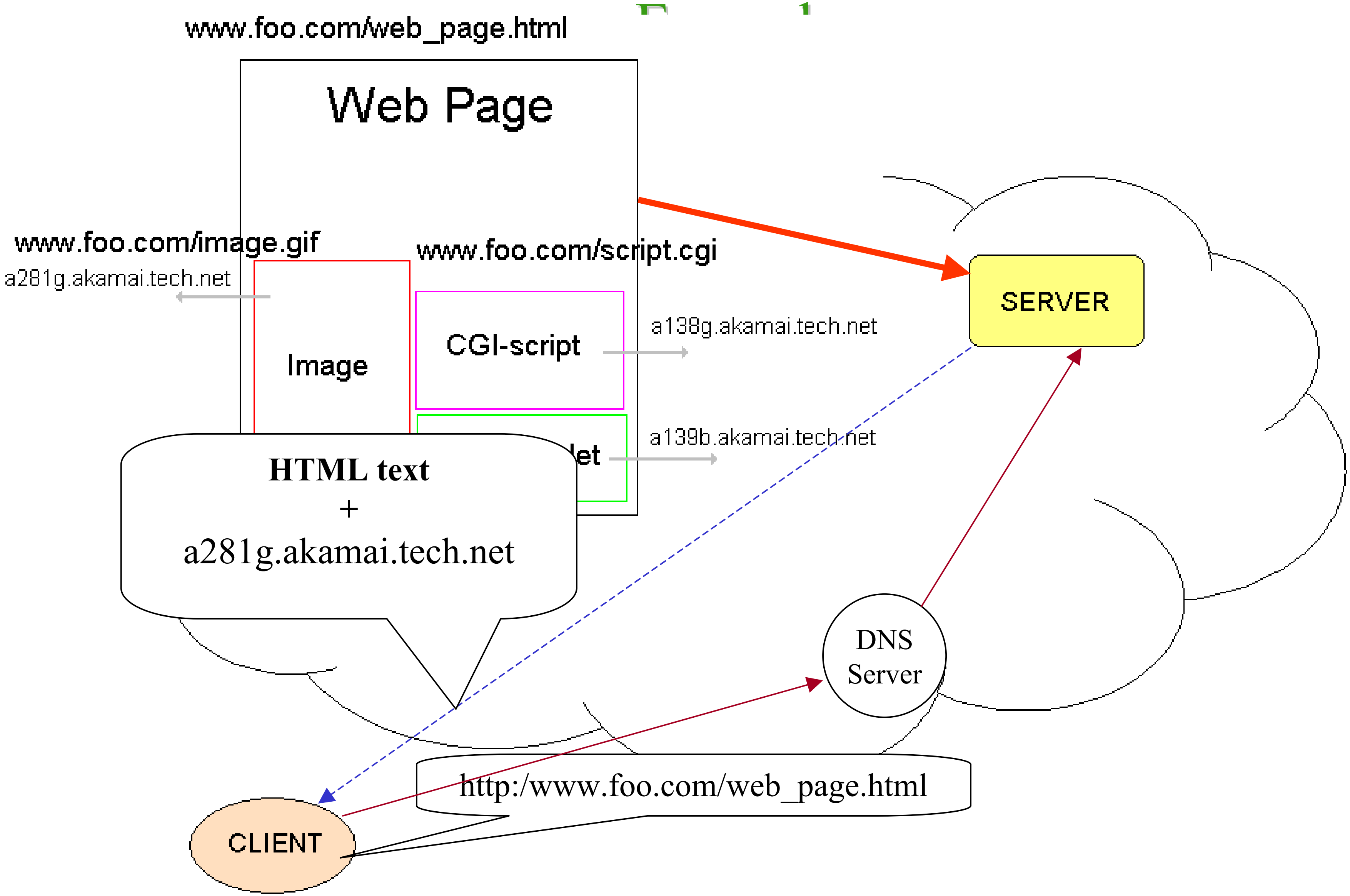
- CLIMB: Similar with LRU. On each access the object climbs one position in the access list (instead of moving to the top (LRU))
- Least Frequently Used (LFU): evicts the object which is accessed least frequently
- SIZE: evicts the largest object
- Lowest Latency First: tries to minimize average latency by removing the object with the lowest download time first.
- Hyper-G: combines LFU, LRU and SIZE.
- Greedy Dual Size: uses a *utility* value and evicts the objects that has the lowest utility. Utility takes into account, the cost of bringing the object from the source, its size and an age factor

Caching versus Replication

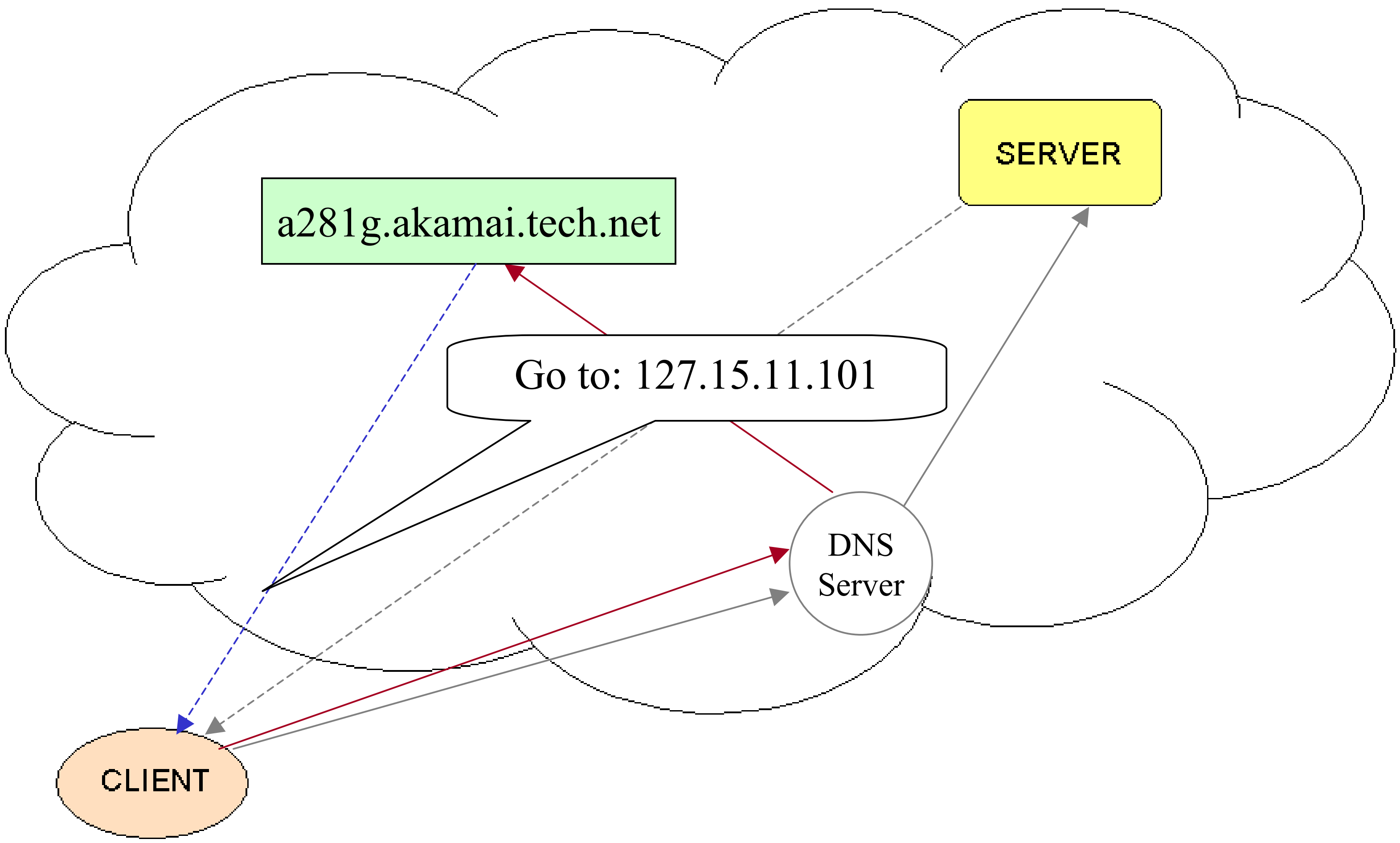
- The same target : Move content closer to the user
- In replication the contents of an origin server are copied to mirror sites
- Advantages:
  - All objects may be available at all times, instead of just the popular objects cached at the proxies
  - The list of replicas is known beforehand; therefore it is possible to multicast the data to the replicas
- Disadvantages:
  - Waste of resources
  - Not scalable
- Reaching a mirror site can be accomplished:
  - Explicitly
  - Redirection via HTTP headers
  - Redirection via DNS lookup

Content Distribution

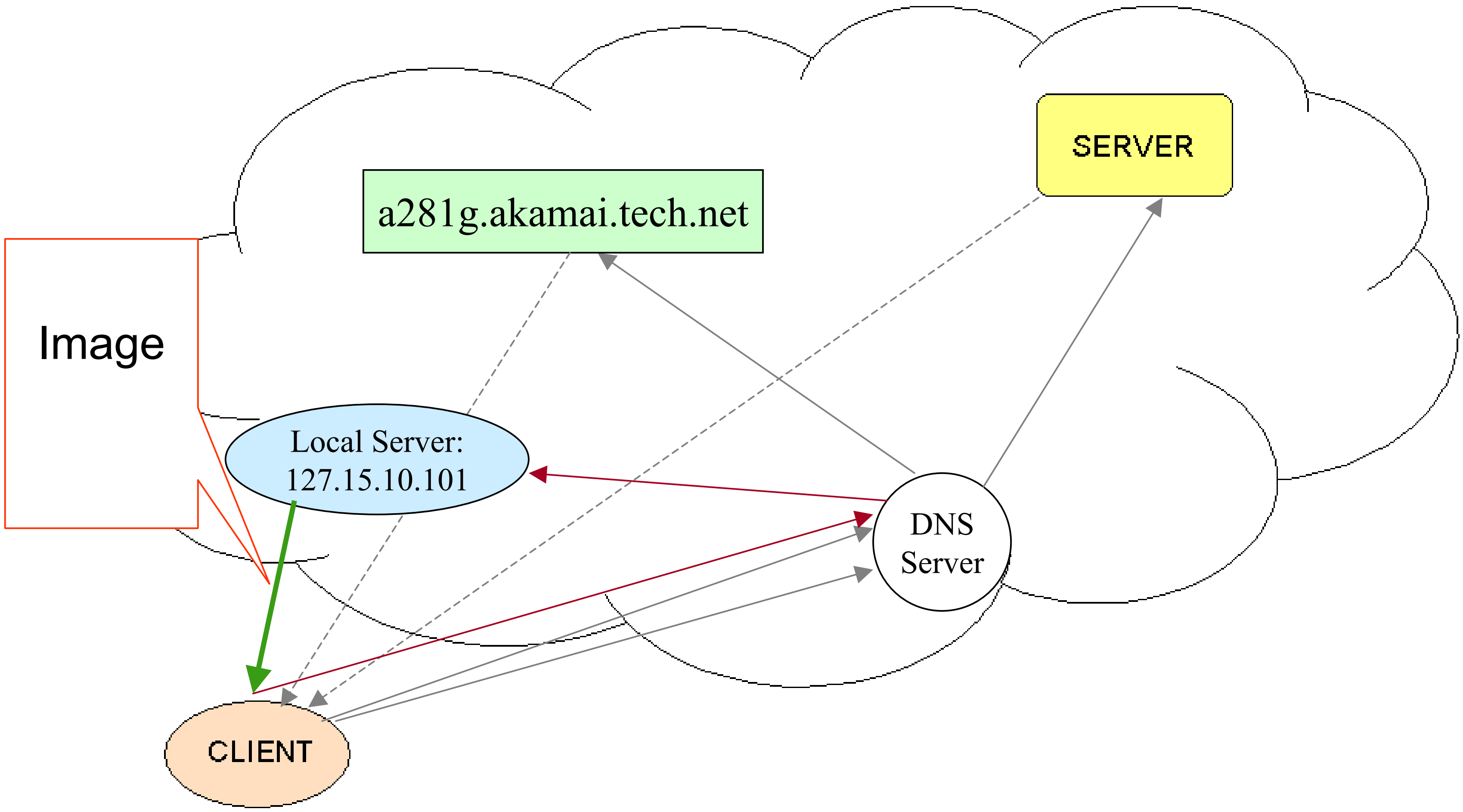
- Basic Idea: offload server by serving some or all contents via a set of replicas
- Web – documents are partitioned into *base* and *embedded* components
  - Base component: container document (e.g. HTML source file)
  - Embedded components: images, scripts etc. (stored in *content distribution servers*)
- The content distribution servers are replicated at different locations in the world
- The origin server only decides which content must be mirrored; Mirroring task is the responsibility of the CDN


www.foo.com/web_page.html
Web Page
www.foo.com/image.gif
a281g.akamai.tech.net
Image
www.foo.com/script.cgi
CGI-script
a138g.akamai.tech.net
a139b.akamai.tech.net
SERVER
HTML text
+
a281g.akamai.tech.net
DNS
Server
http:/www.foo.com/web_page.html
CLIENT

Example


SERVER
a281g.akamai.tech.net
Go to: 127.15.11.101
DNS
Server
CLIENT

Example



SERVER
a281g.akamai.tech.net
Image
Local Server:
127.15.10.101
DNS
Server
CLIENT

Content Adaptation

- Offload work from origin server by moving some of the work traditionally done by servers closer to the clients
- *Content Adaptation*: converting resources to different formats, translations, other “expensive” operations
- Example:
  - Consider an object available in a particular format: video + accompanying audio in English language
  - Suppose few users behind a proxy are interested in getting a still image collection in Dutch language
  - It is possible that the server could maintain different versions in both language and content format
  - It would be desirable for a (content adaptation) proxy to perform such operations