Internet and Intranet Protocols and Applications

Lecture 8a: WWW Proxy Servers and Cookies

March, 2004

Arthur Goldberg
Computer Science Department
New York University
artg@cs.nyu.edu
Terminology

• **Origin Server**
  - The Web server that hosts the resource

• **Proxy Server**
  - Intermediate server that accepts requests from clients and forwards them to (towards) origin servers, to other proxy servers, or services request from its cache.
  - Acts as server to requesting client, and as client to origin server
Web Caches (proxy server)

**Goal:** satisfy client request without involving origin server

- configure browser: Web accesses via web cache
- client sends all http requests to web cache
  - if object at web cache, web cache immediately returns object in http response
  - else requests object from origin server, then returns http response to client
Figure from RFC 2616

The figure above shows three intermediaries (A, B, and C) between the user agent (UA) and origin server (O).
Why Web Caching?

Assume: cache is “close” to client
• smaller response time
• decrease traffic to distant servers
  – link out of institutional/local ISP network is often a bottleneck
Web Caching Summary

- Web proxy Servers store copies of documents retrieved from origin servers
- Advantages
  - **Improve performance** (latency reduction, bandwidth conservation)
  - **Advanced access control** (intermediate requester in firewalled DMZ, authentication & authorization)
  - **Advanced filtering** (e.g. detect espionage!)
  - **Logging and auditing**
- Disadvantages
  - Recognizing and avoiding stale (out of date) data
Proxy Server: Basic Operation

• Accept connection request from client
  – establishes new Socket client_sock

• Read HTTP request

• Parse HTTP request
  – reject invalid requests with appropriate response code
  – Request is REQUIRED to be in absoluteURI form
    • (see RFC 2396)

• Connect to (towards) requested server
  – establishes new socket serv_sock

• Send original HTTP request to server
  – or to next proxy on path to server
Proxy Server: Basic Operation (continued)

• Read response from Server
  – If time-out server connection, then issue
  – 504 Gateway Timeout

• Copy object in response to cache, if allowed
• Send response to client
• If Connection: close header received, close client connection (client_sock)
• What about server connection (serv_sock)?
HTTP 1.1 Cache Control Directives

• Control an object’s “cacheability”
  – RFC 2616 Section 14.9

• Examples
  – `Cache-Control:` general header is used to specify directives that MUST be obeyed by ALL proxy servers handling the request or response.
  – Directives used in Requests
    - `no-cache` an end-to-end revalidation should be preformed
    - `no-store` sensitive information: do not store any part of request or response on disk
    - `max-age=<delta-seconds>` max age acceptable to client
Cache Control Directives (cont.)

• Directives used in responses
  
  \textit{public} response is cacheable by any cache (proxy or client)
  
  \textit{private} response is cacheable by client only
  
  \textit{no-cache} cache MUST NOT use the response to satisfy a subsequent request (for example, dynamic pages)
  
  \textit{no-store} response may not be written to disk
**Proxy Server: Using Cached Objects**

Parse HTTP request
look for the URL in the cache
if (object is found) then
  if (fresh) then
    send response to client with the object
  else
    // validation, see below
  end_if
end_if
Proxy Server: Validate a Stale Object

Proxy:
Forward ‘conditional’ request towards server with an “If-Modified-Since” header with the object’s modification date

Server:
If (the object has been modified since the If-Modified-Since date) then
return the object in the response
Else (the object has not been modified)
return a 304 (Not Modified) response

Proxy:
If( receives 304) then send object from cache
If( receives new object) then cache and send it,

Avoids
- transmitting the full response if the object is current
- an extra round trip if the object is stale
Proxy Server: Validate a Stale Object, cont.

- Validation
  - More general than ‘creation date’ and ‘If-Modified-Mince’
- Includes
  - Expires date
  - Age allowed
  - Heuristic expiration times
- Quite complex—see 13.2 of 2616
Request Loops

• ... to avoid request loops, a proxy MUST be able to recognize all of its server names, including any aliases, local variations, and the numeric IP address.
Cache Architectures

• Components of a Web proxy cache
  – Network communications
  – Storage mechanism for storing the cache data
  – Mapping mechanism to establish relationship between URLs and their cached copies
  – Format for cached object content and its metadata
Cache Architecture: mapping

• Direct mapping
  – e.g., map URL to a file system path
  – direct mappings are reversible

• Hash mapping
  – compute some unique ID
  – could be file name or index to table
  – not reversible
Existing Mapping Mechanisms

• Directly mapping URLs to filesystem
  – Original CERN httpd used a tree map
  – Easy to implement, but not a good performer
    • long pathnames = long inode search
    • garbage collection requires complete traversal of tree
Existing Mapping Mechanisms

- Hashing URLs
  - Netscape Proxy server
- Object location (on disk) based on MD5 hash
  - very fast
  - good distribution of different object types (image, text) across cache
  - disadvantage: cannot compute URLs from hash
Alternative Cache Protocols

• On-Demand
  – document does not exist in cache unless it has been requested (at least once) by some client

• On-Command, or Pre-Fetch
  – proxy server automatically retrieves documents (or even entire web sites!) at regular intervals
General Purpose Proxy Servers

• **Transparency**
  – users get same response whether connection was direct or to proxy
    (a non-transparent proxy modifies content in some way)

• **Use is client controlled**
  – client programs (e.g., browsers) can be configured to use (or not use) proxy servers.

• **Origin Server is unaware of proxy server**
  – Origin Server does not have to process request from proxy differently than from other client

• **Example protocols**
  – Ftp, ssh, socks, telnet, SMTP

• **Typical Location**
  – Firewalled DMZ
Other Intermediate Systems

• Firewall
  – General term for hardware, software, or combination used to protect internal network from intruders.
  – Uses packet filtering to enforce generic security policies
  – Uses application level proxy servers to enforce protocol-specific policies

• Packet filter
  – Control based on something in packet headers (e.g., IP addresses or port numbers)

• Application level proxy
  – Control based on knowledge of application level protocol (e.g., SMTP headers or HTTP methods)
HTTP State Management: Cookies

- We said earlier that HTTP is a stateless protocol.
- We also said that stateful protocols can provide improved performance. This feature is usually established by the idea of a “session” between client and server.
- So, cookies enable HTTP sessions.
Cookies

• Cookie protocol - RFC 2109
• A cookie is a token given to a client by a server
  – Server sends Set-cookie: <cookie> header in a response
  – Client associates <cookie> with the server that sent it
• The *cookie* a sequence of name/value pairs
• Each cookie has a unique name
Cookie Fields

• NAME=VALUE
  - REQUIRED. The name of the "cookie" is NAME, and its value is VALUE.

• Domain=value
  - OPTIONAL. The Domain attribute specifies the domain for which the cookie is valid.

• Path=value OPTIONAL.
  - The Path attribute specifies the subset of URLs on the origin server to which this cookie applies.

• Secure
  - OPTIONAL. The Secure attribute directs the user agent to use only secure means to contact the origin server whenever it sends back this cookie.

• Version=value
  - REQUIRED. The Version attribute identifies the version of the state management specification to which the cookie conforms.
# Example Cookies

<table>
<thead>
<tr>
<th>Origin server domain name</th>
<th>Path</th>
<th>Secure</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>.intellicast.com</td>
<td>/</td>
<td>FALSE</td>
<td>RMID</td>
<td>426cdd9b3ed8ad10</td>
</tr>
<tr>
<td>smallbusiness.yahoo.com</td>
<td>/webhosting/</td>
<td>FALSE</td>
<td>bmcPromoCode</td>
<td>1</td>
</tr>
<tr>
<td>12.46.120.19</td>
<td>/</td>
<td>FALSE</td>
<td>HASBRO_ID</td>
<td>66.108.222.128-70824240.29582435:863790A4DC04E40567FF2D0CF4145723</td>
</tr>
</tbody>
</table>
Client-server Interaction: Cookies

- server sends “cookie” to client in response message
  
  `Set-cookie: 1678453`

- client presents cookie in later requests
  
  `Cookie: 1678453`

- server matches presented-cookie with server-stored info
  - authentication
  - remembering user preferences, previous choices