Some HTTP Message Headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Source</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Client</td>
<td>The type of documents the client can handle</td>
</tr>
<tr>
<td>Accept-Charset</td>
<td>Client</td>
<td>The character sets are acceptable for the client</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>Client</td>
<td>The document encodings the client can handle</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>Client</td>
<td>The natural language the client can handle</td>
</tr>
<tr>
<td>Authorization</td>
<td>Client</td>
<td>A list of the client’s credentials</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>Server</td>
<td>Security challenge the client should respond to</td>
</tr>
<tr>
<td>Date</td>
<td>Both</td>
<td>Date and time the message was sent</td>
</tr>
<tr>
<td>ETag</td>
<td>Server</td>
<td>The tags associated with the returned document</td>
</tr>
<tr>
<td>Expires</td>
<td>Server</td>
<td>The time for how long the response remains valid</td>
</tr>
<tr>
<td>From</td>
<td>Client</td>
<td>The client’s e-mail address</td>
</tr>
<tr>
<td>Host</td>
<td>Client</td>
<td>The DNS name of the document’s server</td>
</tr>
<tr>
<td>If-Match</td>
<td>Client</td>
<td>The tags the document should have</td>
</tr>
<tr>
<td>If-None-Match</td>
<td>Client</td>
<td>The tags the document should not have</td>
</tr>
<tr>
<td>If-Modified-Since</td>
<td>Client</td>
<td>Tells the server to return a document only if it has been modified since the specified time</td>
</tr>
<tr>
<td>If-Unmodified-Since</td>
<td>Client</td>
<td>Tells the server to return a document only if it has not been modified since the specified time</td>
</tr>
<tr>
<td>Last-Modified</td>
<td>Server</td>
<td>The time the returned document was last modified</td>
</tr>
<tr>
<td>Location</td>
<td>Server</td>
<td>A document reference to which the client should redirect its request</td>
</tr>
<tr>
<td>Referrer</td>
<td>Client</td>
<td>Refers to client’s most recently requested document</td>
</tr>
<tr>
<td>Upgrade</td>
<td>Both</td>
<td>The application protocol the sender wants to switch to</td>
</tr>
<tr>
<td>Warning</td>
<td>Both</td>
<td>Information about the status of the data in the message</td>
</tr>
</tbody>
</table>

Status Code

- The status-code in an HTTP response message is a 3-digit integer; the first digit defines the class of response
  - **1xx: Informational** - Request received, continuing process
  - **2xx: Success** - The request was successfully received, understood, and accepted
  - **3xx: Redirection** - Further action must be taken by the user agent in order to fulfill the request
  - **4xx: Client Error** - The request contains bad syntax or cannot be fulfilled
  - **5xx: Server Error** - The server failed to fulfill an apparently valid request
Example Status Codes

200 OK
   – Request succeeded, requested object included in this message
301 Moved Permanently
   – Requested object has moved, new location specified in Location header
400 Bad Request
   – Request message not understood by server
404 Not Found
   – Requested document not found on this server
503 Service Unavailable
   – The server is currently unable to handle the request due to a temporary overloading or maintenance of the server
505 HTTP Version Not Supported
   – The server does not support the HTTP protocol version that was used in the request message

Client-Side Caching

• Client-side caching reduces the response time perceived by Web users
• Caching at browsers: whenever a document is fetched it is stored in the browser’s cache from where it is loaded the next time
• Caching at proxies: Some organizations use a web proxy to improve web performance
   – The web proxy implements a cache shared by all clients at the organization
   – The proxy accepts requests from local clients and passes them to web servers if the requested documents cannot be found in the cache
• How can a web proxy maintain consistency of cached data?
   – Push vs. pull
Push-Based Approach

- Server keeps track of all proxies that have requested documents; if a document is modified, server notifies each proxy that holds a copy of the document
  - Proxy must inform the server when it purges a document from cache
- Types of notification
  - **Invalidation**: Indicate document has changed
  - **Update**: Send new version of document
- How to choose between invalidation and update?
  - Send updates for popular documents, send invalidations for the rest
- Advantage: Consistency of cached data is maintained
- Disadvantage: Servers are stateful → not resilient to server crashes

Pull-Based Approach

- Proxy is responsible for maintaining cache consistency
- Proxy periodically polls the server to see if a cached document has changed by sending a **conditional GET** HTTP request message
  - The conditional GET request message contains a **If-Modified-Since** header specifying the last modification time of the cached document, denoted by T
  - Server will return the document only if the document has been modified since T
Conditional GET

Client

HTTP request
If-Modified-Since: Fri, 01 Apr 2007 05:00:01 GMT

HTTP response
HTTP/1.1 304 Not Modified

document not modified

Server

HTTP request
If-Modified-Since: Fri, 01 Apr 2007 05:00:01 GMT

HTTP response
HTTP/1.1 200 OK
<data>
document modified

Intelligent Polling

- A proxy can dynamically adjust the polling interval instead of using a fixed polling interval
  - Start with an initial polling interval
  - Increase interval if document has not changed between two successive polls
  - Decrease interval if document has changed between two successive polls
Pull-Based Strategy in Squid Web Proxy

• Assign an expiration time to a cached document
  \[ T_{\text{expire}} = \alpha(T_{\text{cached}} - T_{\text{last_modified}}) + T_{\text{cached}} \ (\alpha=0.2) \]
  – \(T_{\text{last_modified}}\) is the last modification time of the document
  – \(T_{\text{cached}}\) is the time the document was cached
• Until \(T_{\text{expire}}\) the document is considered valid
  – After the expiration time, the proxy sends a conditional GET request to server to get a fresh copy of the document
• Cached document in may be obsolete when \(\alpha>0\)
  – When \(\alpha=0\), cached document is always up-to-date

Pull-Based Approach: Pros and Cons

• Pros
  – Supported by HTTP (conditional GET)
  – Server remains stateless \(\rightarrow\) resilient to server crashes
• Cons
  – Weaker consistency guarantees: proxy may return obsolete documents to clients
  – More sophisticated proxies required
Content Distribution Networks (CDNs)

- A CDN provides an infrastructure of CDN servers for distributing and replicating the web documents of multiple sites across the Internet
  - A CDN provides its customers (i.e., content providers) guarantees on high availability and performance
- Each web page consists of a base HTML page and several embedded documents; the embedded documents are replicated on CDN servers
  - Cost of maintaining consistency is low because embedded documents rarely change
- The best CDN server to answer a client request is selected automatically
  - Best server could be the closest one or the least-loaded one

Akamai CDNs

URL of base document: http://www.foo.com/homepage.html
Consistency in Akamai CDN

• How to enforce the consistency of embedded documents?
  – A hash value of an embedded document is included in its modified URL
  – The hash value changes when the embedded document changes → The hash value changes the name of the embedded document
  – When the client is redirected to a specific CDN server, the server will not find the named document in the cache and thus will fetch the document from the origin server