CACHE MANAGEMENT

More Definitions

- **Cache**
  - A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cachable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, except for the client.

  *Click Here to see Netscape's Cache*

- **First-hand**
  - A response is first-hand if it comes directly and without unnecessary delay from the origin server, perhaps via one or more proxies. A response is also first-hand if its validity has just been checked directly with the origin server.

- **Explicit expiration time**
  - The time at which the origin server intends that an entity should no longer be returned by a cache without further validation.

- **Heuristic expiration time**
  - An expiration time assigned by a cache when no explicit expiration time is available.

- **Age**
  - The age of a response is the time since it was sent by, or successfully validated with, the origin server.

- **Freshness lifetime**
  - The length of time between the generation of a response and its expiration time.

- **Fresh**
  - A response is fresh if its age has not yet exceeded its freshness lifetime.

- **Stale**
  - A response is stale if its age has passed its freshness lifetime. semantically transparent.

- **Validator**
  - A protocol element (e.g., an entity tag or a Last-Modified time) that is used to find out whether a cache entry is an equivalent copy of an entity.
Caching in HTTP

- The objective of HTTP caching is to improve the performance by reducing network traffic by caching responses.
- Specific Goals:
  - Reduces/eliminates send/request entire cycles.
  - Reduces/eliminates sending full responses.
- Control Models:
  - "expiration" model
  - "validation" model

Control Mechanisms

- Default Cache Control Algorithm:
  - A set of algorithms which work with server specified:
    - Expiration times
    - Validators
    - Algorithms remove ambiguity.
- Explicit Cache Control Directives:
  - A server or client can send explicit directives to HTTP caches. There directives overrides default algorithms.
  - If, there is conflict between directives, the most restrictive one supercedes.

Expiration Models

- Server-Specified Expiration
  - Origin Server specifies an explicit expiration date.
  - A cache can return a fresh response without contacting the server by looking at the expiration date.
  - It completely eliminates cache to server roundtrip communication.
  - Server can specify an expiration time by:
    - Expire Header
    - Expires: Thu, 01 Dec 1998 16:00:00 GMT
    - max-age in Cache-Control Header
    - Cache Control: max-age=3600
  - A cache can use a heuristic model to determine expiration date. Example 20% (Last modified minus creation date)

General Idea of using Age Value

- HTTP/1.1 uses the Age response-header to help convey age information between caches.
- The Age header value is the sender's estimate of the amount of time since the response was generated at the origin server.
- In the case of a cached response that has been revalidated with the origin server, the Age value is based on the time of revalidation, not the original response.
- Age value is the sum of the time that the response has been resident in each of the caches along the path from the origin server, plus the amount of time spent in transit.

Two Methods

- A response's age can be calculated in two entirely independent ways:
  - 1. now minus date_value, if the local clock is reasonably well synchronized to the origin server's clock. If the result is negative, the result is replaced by zero.
  - 2. age_value, if all of the caches along the response path implement HTTP/1.1.
- Given that we have two independent ways to compute the age of a response when it is received, we can combine these as:
  \[
  \text{CORRECTED\_RECVD\_AGE} = \max(\text{NOW} - \text{DATE\_VALUE}, \text{AGE\_VALUE})
  \]
Computing Current Age

**RESPONSE IS FRESH = if**

\[
\text{FRESHNESS\_LIFE\_TIME > \text{CURRENT\_AGE}}
\]

**APPARENT\_AGE = max(RESPONSE\_TIME - DATE\_VALUE, 0)**

**CORRECTED\_RECEIVED\_AGE = max(APPARENT\_AGE, AGE\_VALUE)**

**CORRECTED\_LOCAL\_AGE = CORRECTED\_RECEIVED\_AGE - RESPONSE\_DELAY**

**RESIDENT\_TIME = \text{NOW} - RESPONSE\_TIME**

**CURRENT\_AGE = CORRECTED\_LOCAL\_AGE + RESIDENT\_TIME**

\[
\text{or} \quad \text{CURRENT\_AGE} = \text{max}(0, \text{RESPONSE\_TIME - DATE\_VALUE} + \text{NOW} - \text{REQUEST\_TIME})
\]

---

**Validation Models**

- **When a cache has a stale entry it first has to check with the origin server (or possibly an intermediate cache) with a fresh response) to see if its cached entry is still usable.**
- **We do not want to pay the overhead of:**
  - retransmitting the full response, if the cached entry is good
  - or an extra round trip, if the cached entry is invalid
- **How can we do that?**
  - The HTTP/1.1 protocol supports the use of conditional methods with cache validators.

**Example:** Last Modified date based Validation

- **Last-modified Dates**
  - The Last-Modified entity-header field value is often used as a cache validator. In simple terms, a cache entry is considered to be valid if the entity has not been modified since the Last-Modified value.

  **HTTP/1.1 200 OK**
  **Server:** NCSA/1.3
  **Mime_version:** 1.0
  **Age:** 3600
  **Last Modified:** Thu, 01 Dec 1998 16:00:00 GMT
  **Date:** Tue, 15 Nov 1998 08:12:31 GMT
  **Content-type:** text/html
  **Content-length:** 2000
  **<HTML>…</HTML>**

  **What is wrong here?**
  - Server makes them equal.

**Strong vs. Weak Validators**

**Example: Conditional Validation**

- **Entity Tag Cache Validators**
  - The ETag entity-header field value, an entity tag, provides for an "opaque" cache validator. This may allow more reliable validation in situations where it is inconvenient to store modification dates, where the one-second resolution of HTTP date values is not sufficient, or where the origin server wishes to avoid certain paradoxes that may arise from the use of modification dates.

**Explicit Cache Control Directives**

- **The Cache-Control general-header field is used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain.**
- **These directives typically override the default caching algorithms.**
- **Cache directives are unidirectional in that the presence of a directive in a request does not imply that the same directive should be given in the response.**

  **Major Classes**
  - Restriction, what is cacheable (by OS)
  - Restriction, what may be stored (by OS)
  - Modification, based on Expiration Model (by OS+UA)
  - Control over Cache Revalidation (by UA)
  - Control over Entity Transformation
Explicit Cache Control Directives

**Request Directives:**
- no-cache
- no-store
- max-age = delta-seconds
- max-stale = delta-seconds
- min-fresh = delta-seconds
- only-if-cached
- cache-extension

**Response Directives:**
- public
- private
- no-cache
- no-store
- no-transform
- must-revalidate
- proxy-revalidate
- max-age = delta-seconds
- cache-extension

---

**Cache Location**

- **Client**
  - # clients: one, few, most
- **Server**
  - # servers: all, most, few
- **Internet**
  - This node

---

**Cache Replacement Policies**

- It is a delicate balance between the cache and performance.
- Following are some attributes which can be used for cache document replacement:
  - SIZE document size (bytes)
  - ETIME time doc entered cache
  - NREF number references to doc
  - ATIME time doc last accessed
**A Typical Performance Result:**
Primary Key Comparison
(Cache Size = 10% of max needed)

---

**Reference Material**

- Primary source RFC-2068.
  - Specially read section:
    - Section 13
    - Section 14.9

- Books:
    - Sections 6 & 7.