Web Cache Consistency

“Requirements of performance, availability, and disconnected operation require us to relax the goal of semantic transparency.”
- HTTP 1.1 specification

Any caching/replication framework must take steps to ensure that the cache does not deliver old copies of modified objects.

Issues for cache consistency in the Web:
- large number of clients/proxies
- most static objects don’t change very often
- weaker consistency requirements
  Stale information might be OK, as long as it is “not too stale”.

Validation vs. Invalidation

Validation
- Proxy periodically polls server for updates to cached objects
- How often to poll? (“freshness date”)
- Sync vs. async

Invalidation
- Server informs proxy if cached object is updated

Validation vs. Invalidation: The Tradeoffs

What are the tradeoffs?
- Scale
- Consistency quality
- Performance and poll overhead
  Fast hit vs. slow hit
  Does popularity correlate with update rate?

Validation “works” today!
  GET-IF-MODIFIED-SINCE

How to set the TTLs or expires headers?

Design of a scalable invalidation architecture for the Web is a difficult challenge.

Cache Expiration and Validation

HTTP 1.0 cache control
- Origin server may add a “freshness date” (Expires) response header.
  ...or the cache could determine expiration time (TTL) heuristically.
- Proxy must revalidate cache entry if it has expired.
  Last-Modified and If-Modified-Since

Consistency: Variations on a Theme

- Pipeline validations and Piggyback Cache Validations
  [Krishnamurthy and Wills]
  Opportunistically “prefetch” validations.
  Enough traffic to benefit?
- Coarse granularity: volumes
  Cluster objects in volumes to reduce the number of validations when update rates are low.
- Delta encoding [Mogul et al 1997]: fine-grained updates
  Optimistic delta: reduce latency of a consistency miss by sending a stale copy from cache, followed by the delta.
  Nice hack for cookied content.
HTTP 1.1

Specification effort started in W3C, finished in IETF... much later.
A number of research works influenced the specification.

Performance
- persistent connections with pipelining
- range requests, incremental update, deltas

Caching
- cache control headers
- negotiation of content attributes and encodings
- content attributes vs. transport attributes
- transport encodings for transmission through proxies

Trailer header and trailer headers

Expiration and Validation in HTTP 1.1

HTTP 1.1 cache control allows origin server to:
- use relative instead of absolute expiration times (max-age);
- issue opaque validators (ETag) instead of timestamps;
- Origin server may specify which of several cached entries to use.

Other 1.1 Cache Control Features

- Client may specify that no caching is to occur.
  - private or no-store
- Vary headers allow server to specify that certain request headers must also match if the proxy does not have any cached response valid.
  - language, character set, etc.
- Server may specify that a response is not cacheable.
  - Pragma: no-cache header since HTTP 1.0
- Client may explicitly request the proxy to validate the response.
- Vary header
- Proxy may/shall/must tell client the age of a cached response.
- Age header
- Proxy may/shall/must tell client that it could not validate a non-fresh cached response with the origin server.
- Warning header

The Role of the Content Developer

- Use expiration dates where known
- Limit the scope of cookies
- If using cookies for personalization, use cache control headers to disable caching on the personalized objects
  - What if you forget?
    - Decompose dynamic pages into cacheable and uncacheable components.
      - Templates [Douglas97]
      - Edge-side includes (Akamai)
      - Base instance [WebExpress]

Cookies

HTTP cookies (RFC2109) have brought us a better Web.
- S optionally includes arbitrary state as a cookie in a response.
- Cookie is opaque to C, but C saves the cookie.
- C sends the saved cookie in future requests to S, and possibly to other servers as well.
- Allows stateful servers for sessions, personalized content, etc.

But: cookies raise privacy and security issues.
- What did S put in that cookie? Can anyone else see it? How much space does it take up on my disk that I paid sooo much for?
- Cookies may allow third parties who are friends of S1,..., SN to observe C’s movements among S1,..., SN.
- Unverifiable transactions, e.g., DoubleClick and other ad services.

Unverifiable Transactions

- Users may not know that they are interacting with DoubleClick.
  - Amazon and MyCFO trust DoubleClick, but client is ignorant.
- The user visits pages at many sites that reference DoubleClick.
  - DoubleClick’s cookie allows it to associate all the requests from a given user.
  - If the browser sends Referer headers, DoubleClick may gather information about all the sites the user visits that reference DoubleClick.
WCDP

Sara Sprenkle led a discussion of WCDP, a protocol for server-driven consistency from IBM.
Slides for this portion of the class may be found at:
http://www.cs.duke.edu/~sprenkle/wcdp.ppt
It is important to understand the context of the server-driven approach, its role in CDNs, the opportunity to use invalidation, and how WCDP addresses the scalability concerns.