

The Web Caching Architecture

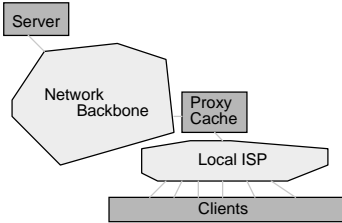
Web Caching

Reducing Load on the Internet

- Transport layer mechanisms
 - TCP congestion control (Tahoe, Reno)
 - TCP congestion avoidance (Vegas)
- Application layer mechanisms
 - Web caching:
 - Popular content is stored in the vicinity of clients
 - Client requests don't have to cross the Internet backbone to access popular remote sites

Web Caching

- Caching improves perceived performance

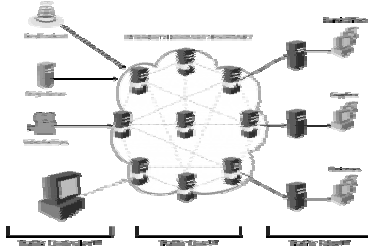


Cache Providers

- The market for cache appliances is on the rise:
 - CacheFlow Inc.
 - CISCO Cache Engine
 - Inctomi Content Networking
 - Network Appliance
 - SkyCache
 - Squid Cache (free)

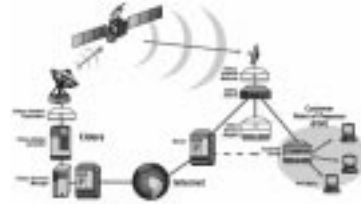
Example: Inktomi Content Networking Architecture

- Caching + application layer multicast



Example: Cidera SkyCache

- Broadcasts rich content via satellite to more than 600 sites around the Internet
- Bypasses congestion on the ground



Example: Akamai

- Client retrieves main page from origin site
- Embedded links are changed to point to a nearby Akamai server
- Client retrieves embedded objects from Akamai



Types of Web Caching

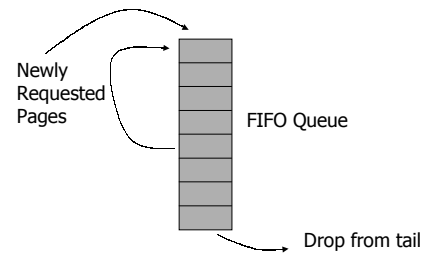
- The cache retains popular URLs
- Types of caching:
 - Client caching
 - browser caches URLs for future access
 - Proxy caching
 - proxy server caches most requested URLs
 - Server-side caching
 - server-side cache reduces load on server
 - Content distribution
 - Server implements a node in an application-layer (multicast) content distribution network

Web Caching Policies

- A URL is cached when it is requested
- Cache must replace content when full
- Cache replacement policies
 - LRU (replace least recently used page)
 - LFU (replace least frequently used page)
- Other considerations:
 - page size, client importance, cache distance from server

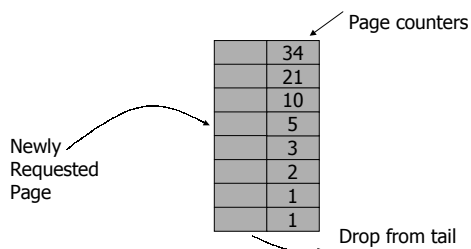
Cache Replacement

- Least Recently Used



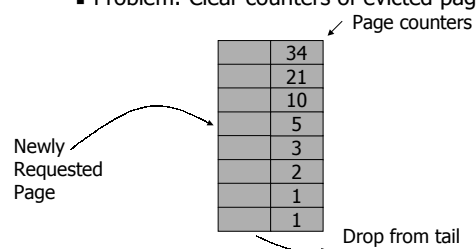
Cache Replacement

- Least Frequently Used



Cache Replacement

- Least Frequently Used
 - Problem: Clear counters of evicted pages?



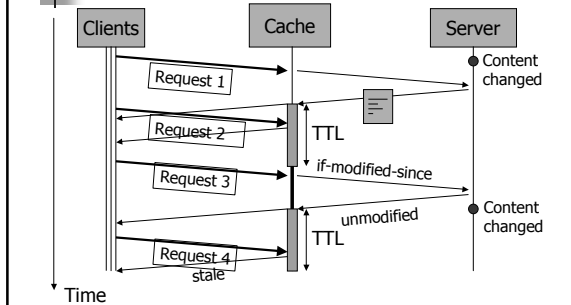
Cache Consistency Management

- Problem with caching
 - Cached pages may get modified on server
 - Clients accessing cache will get stale copy
- Possible solutions
 - Finite "time-to-live"
 - Invalidation/Update
 - Contracts

Time-to-live & Conditional GET

- Server specifies a time-to-live for requested page
- Proxy caches page and serves subsequent clients until time-to-live expires
- Proxy sends if-modified-since query to server on subsequent request(s)
- Server sends page if it was modified (presently used method in Squid)

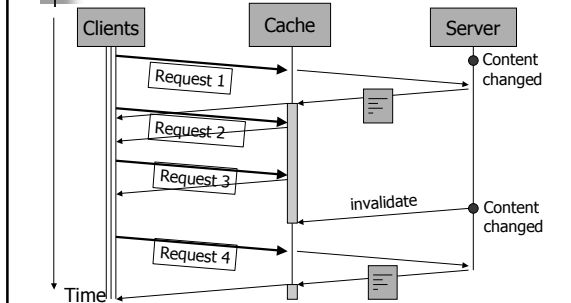
Example: TTL



Cache Invalidation

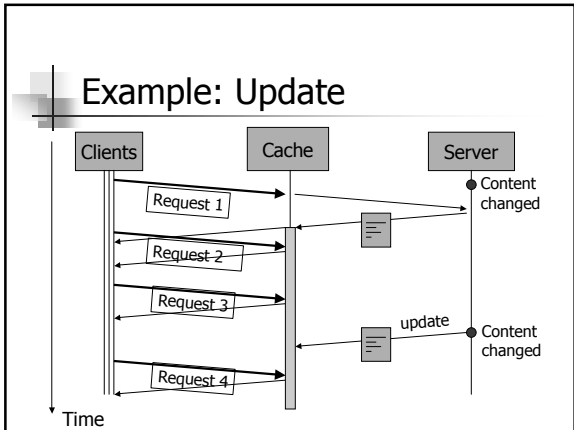
- Server remembers where its pages are cached
- When a page is modified, server notifies the caches which have a copy of this page
- Caches mark page as stale
- Subsequent client requests will fetch fresh copy from server

Example: Invalidation

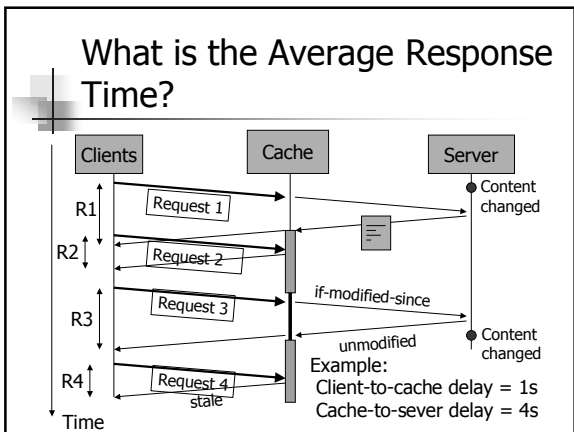
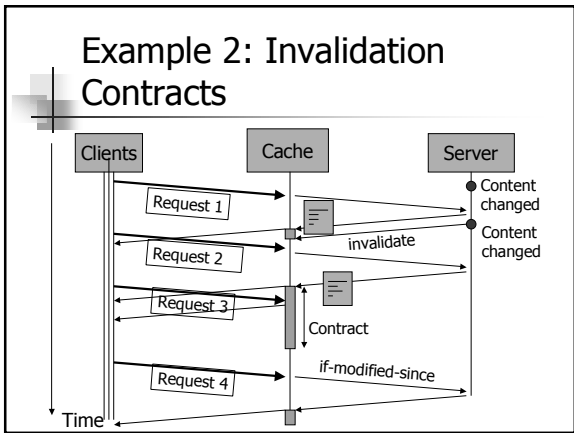
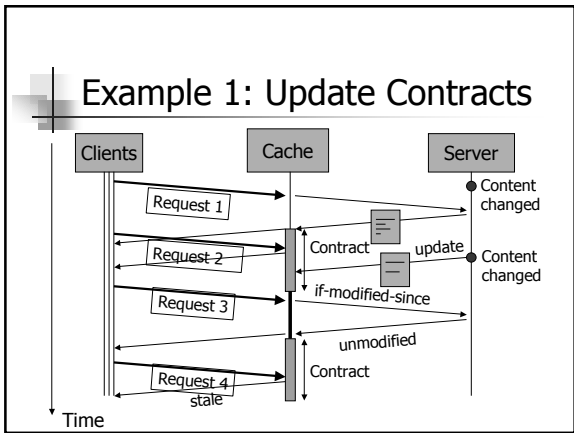


Cache Update

- Server remembers where its pages are cached
- When a page is modified, server notifies the caches and gives them an updated version of the page (usually used together with application layer multicast)
- Caches always assume that they have the latest content



- ### Invalidation/Update Contracts
- Invalidation/update traffic can get very large
 - To limit this traffic server gives invalidation or update contracts to caches
 - Contracts have expiration time
 - When content changes server notifies only those caches whose contract has not expired



- ### What do I need to know?
- How to compute cache response time?
 - For a particular request arrival scenario and caching scheme, how to tell:
 - Which requests are going to get served from the cache? Which requests will go to the origin server?
 - Which requests will cause the cache to send the server an if-modified-since? Which will cause the cache to send a get?
 - Which requests will get served a stale copy of data?

Caching versus Content Distribution

- Caching: client demand driven
 - Content is moved in response to client requests (or in accordance with contracts that occur due to client requests)
- Content distribution: supply driven
 - Web site owners (content providers) make agreements with content distribution agents to replicate their content across the Internet backbone (irrespectively of client request patterns)

Business Model

- Why caching?
 - Local ISPs pay backbone providers for traffic the ISPs generate on the backbone provider's network
 - ISP use caching to reduce traffic across the backbone and minimize their fees to backbone provider
 - Side effect: Users see good performance

Business Model

- Why content distribution?
 - Important businesses (e.g., e-commerce portals) want their web sites to be available to users at all times regardless of Internet load conditions (to increase revenue)
 - Solution: pay a content distribution network to distribute the web site around the Internet backbone so it is closer to prospective clients

Challenges in Content Distribution Networks

- Where to replicate the content?
 - What is the minimum number of replicas that guarantees that all clients will see acceptable performance?
- How to propagate updates to content copies?
 - An application-layer multicast problem
- How to route client requests to content replicas?

Example

Where to place content so that it is accessible from any node within 5 sec?

