CS514: Intermediate Course in Computer Systems

Lecture 21: March 7, 2003

"More on Web Caching"

We've seen lots of caching

CS514

- DNS caching
- Web caches front-ending web servers
- Caches within computer systems
  - L1 cache, L2 cache
- C. Mohan's compendium of caching
- Akamai cached CDN

Today drill down on web caching

CS514

- Early vision of distributed cache architectures
- Where are caches, and why are they there
  - And where aren't they, and why not?
- HTTP support of caching
- Deployment mechanisms
  - Load balancers for reverse caching
  - Explicit versus transparent proxies (WCCP)
- Cache replacement strategies
- Fancy stuff
  - Cache networks, ICP, pre-fetching, hoarding

Definition of caching

CS514

- Store some fraction of objects closer to the consumer/user of those objects
  - For performance reasons
- Based on idea that if object was recently used, it'll be used again soon
  - Though often attempts to predict what will be used based on other criteria (pre-fetching)
- Loose consistency between cached copy and original
  - Cached object may become "stale"
  - Various ways to deal with this (TTL, explicit invalidation, validation)

Goals of web caching

CS514

- Two primary goals
  - Reduce latency
  - Reduce bandwidth
- Sometimes these two goals conflict

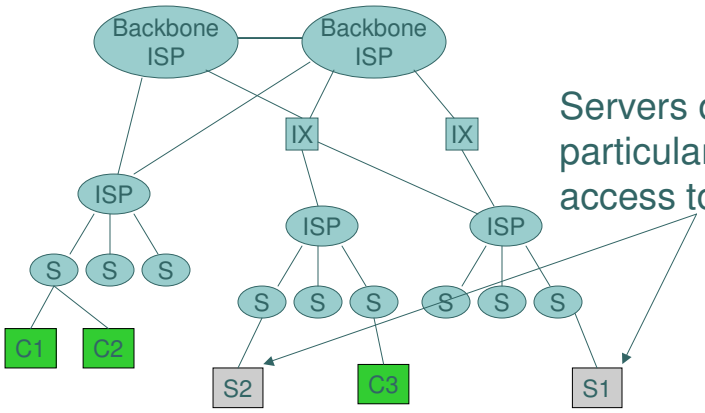
Early days of web caching (circa 1994)

CS514

- There were no mega- websites
  - And there were no huge web server farms
- Internet performance was poor
- Web (or FTP) server performance was poor

Early web caching vision

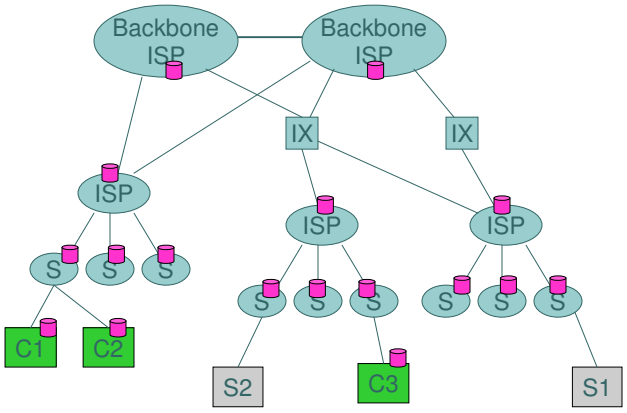
CS514



Servers don't have particularly fat access to internet

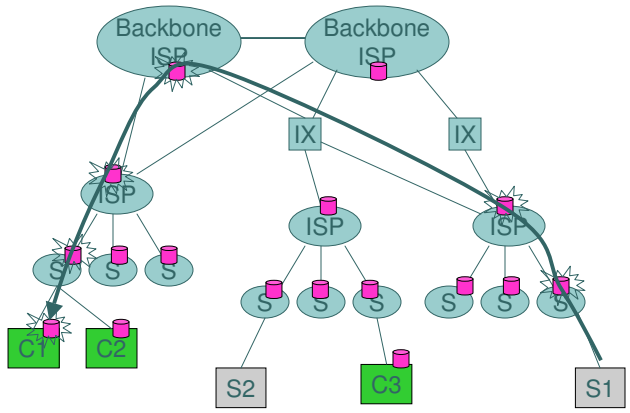
Put caches everywhere

CS514



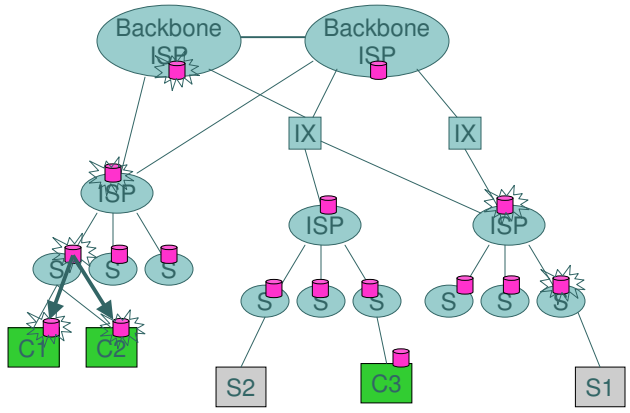
First access populates all caches in path

CS514



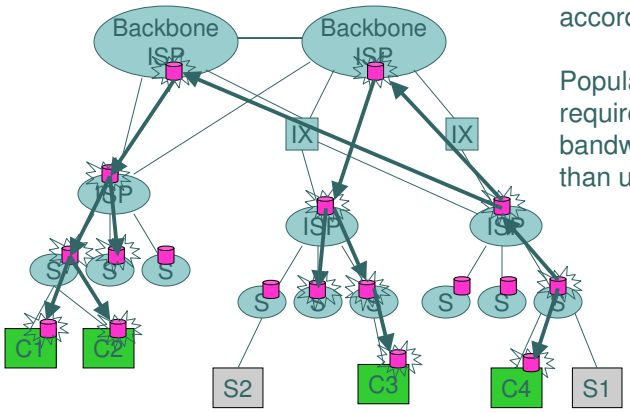
Localized demand served from caches near clients

CS514



Broader demand served from caches near clients and server

CS514



Caches populated according to demand

Popular servers wouldn't require all that much more bandwidth and capacity than unpopular servers

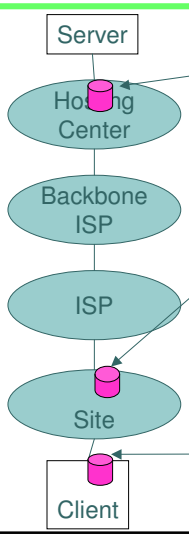
Early grand cooperative vision never materialized

CS514

- Caches deployed at the edges…
  - Near servers and in or near clients
- But not really in the middle
  - Little or no economic motivation for middle to add caches
  - Little evidence that caching in the middle helps
    - Though Akamai may be a counterexample
  - Indeed, caching in middle can hurt performance
    - Cache misses increase latency
    - Caches can be bottlenecks, failure points

Where are web caches today?

CS514



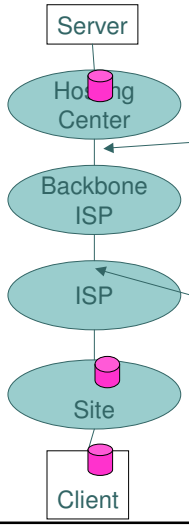
"Reverse proxy" cache: Front-end web servers to reduce server load
- System specialization
- Also surge protection

"Proxy" cache: Between site (enterprise) and ISP
- Offload ISP access link
- Gives site control over internet usage

"Local" cache: On browser
- Reduce latency

Where *aren't* web caches today?

CS514



Not at backbone ISP.
- Because backbone ISP sells bandwidth to hosting center, has no motivation to offload this link

Are caches here??? ISP may have motivation (pays backbone ISP for bandwidth), but may not wish to deal with cache problems
- Hard to debug…
- Possibly more common for residential ISPs???

HTTP cache support model

CS514

- Expiration
  - Improve latency and bandwidth
- Validation
  - Improve bandwidth
- Relaxed semantic transparency:
  - Can be controlled by server or client
  - Cache can provide warning

Expiration

CS514

- Improve latency because cache doesn't need to validate non-expired content
- Cache-control: max-age=60
  - Previous caches use Age: header to show that they have aged the entry
- Expires header
  - Expires: Fri, 30 Oct 2002 08:06:55 GMT
  - Problem: How does server know when content will expire?
- *Does not* cause browser to reload

Validation

CS514

- Client can conditionally request validation
  - Entity-tag
    - Provided by server, cached by client
    - Client later sends to server to validate cached entry
  - Last-Modified header
    - Client indicates when it got cached entry, and server doesn't send content if not modified since then

Important HTTP cache support features

CS514

- max-age, min-fresh, max-stale
  - allows client to limit age, freshness (time in cache), or staleness (allow stale response)
- public/private
  - public=cacheable, private=non-cacheable
  - used to override default behavior
- no-cache
  - server forces revalidation
  - client forces end-to-end reload

Important HTTP cache support features

CS514

- no-store
  - server prevents caching altogether
- must-revalidate
  - server can over-ride cache or client allowed staleness
- no-transform
  - server or client can specify no transformation of content

HTTP warning headers

CS514

- Added by cache to warn client/user of possible problem:
  - 110 Response is stale
  - 111 Revalidation failed
  - 112 Disconnected operation
  - 113 Heuristic expiration
  - 199 Miscellaneous warning
    - Contains text string for user or log
  - 214 Transformation applied

Ethereal example

CS514

```
GET /cnn/.element/img/1.0/logo/cnn.gif HTTP/1.1
Accept: */*
Referer: http://www.cnn.com/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-Modified-Since: Mon, 16 Sep 2002 13:13:36 GMT; length=1910
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: i.a.cnn.net
Connection: Keep-Alive

HTTP/1.0 304 Not Modified
Content-Type: image/gif
Last-Modified: Mon, 16 Sep 2002 13:13:36 GMT
Cache-Control: max-age=2870
Date: Thu, 06 Mar 2003 16:55:38 GMT
Surrogate-Server: AkamaiGHost
Connection: keep-alive
```

Explicit versus transparent caches

CS514

- Browsers may be configured with an explicit proxy
  - All HTTP requests go to this proxy, not the origin page
- Original benefit was getting through firewalls
- Benefits for low-speed links (wireless):
  - Maintain really persistent connection
  - No DNS lookups
- But configuring browser is a pain
  - So now routers can force packets to cache without explicit configuration

Transparent caches: WCCP

CS514

- WCCP: early Cisco vision
  - Web Cache Communication Protocol
  - Between steering router and caches
  - Idea was that caches would direct router behavior
    - Which hash buckets to direct to which caches
    - Which caches were overloaded
    - Reject individual queries (i.e. because authentication failed)
  - This vision never took root

Transparent caches

CS514

- Recall load balancer lecture
- Load balancer has various “switch and persist” policies
- Can monitor load and correctness of caches
- No need for explicit coordination between cache and load balancer

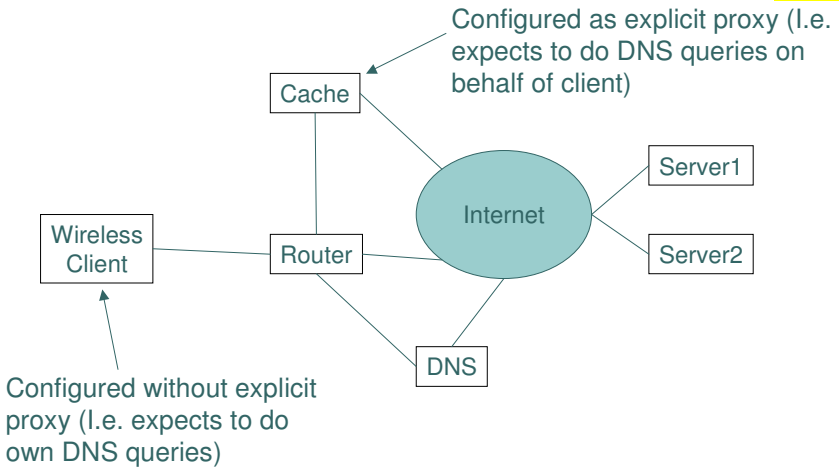
Idea for wireless HTTP: DNS booster

CS514

- From Pablo Rodriguez (while at Tahoe)
- Try to get benefits of explicit proxy without requiring browser configuration
- Idea is to modify DNS answers to emulate explicit proxy behavior at wireless client

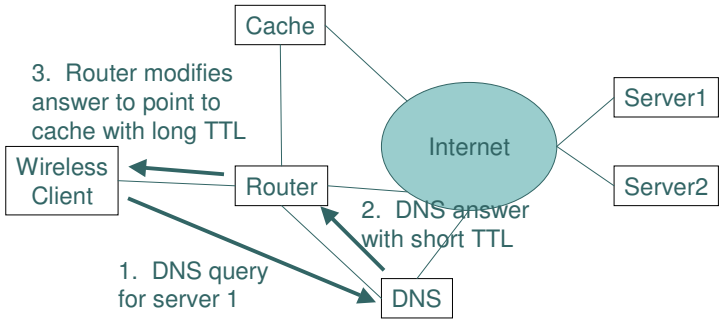
Wireless DNS booster

CS514



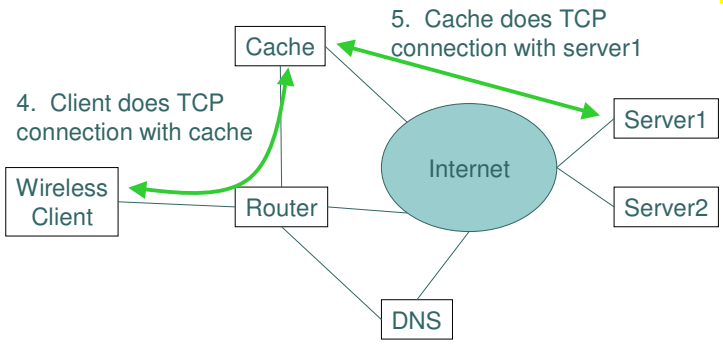
Wireless DNS booster

CS514



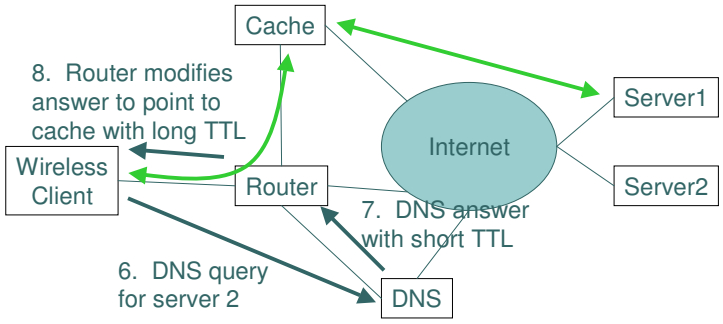
Wireless DNS booster

CS514



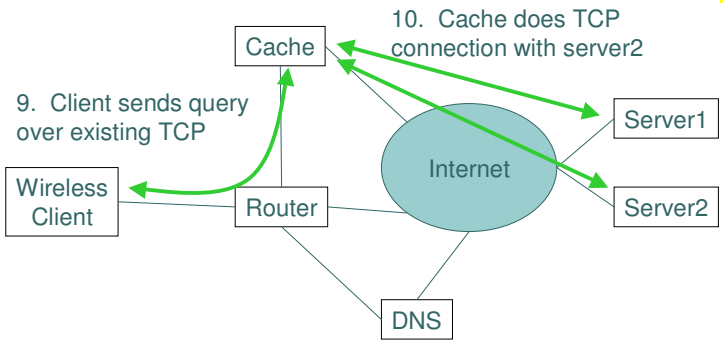
Wireless DNS booster

CS514



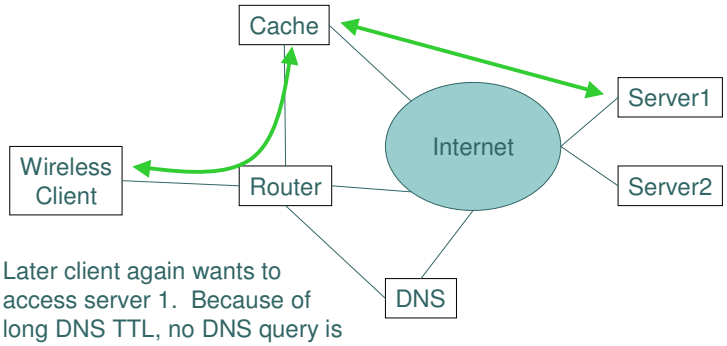
Wireless DNS booster

CS514



Wireless DNS booster

CS514



Later client again wants to access server 1. Because of long DNS TTL, no DNS query is necessary. Client does TCP to cache (if one not already up), and cache does DNS query.

Cache replacement strategies

CS514

- When new object added to cache, some old object must be removed
- How to select object to remove?
- Two basic approaches:
  - Remove least used object (least recently or least frequently)
  - Remove largest object
    - Large object can make room for lots of small objects
- Lots of variations and hybrids

Best replacement strategy depends on goal

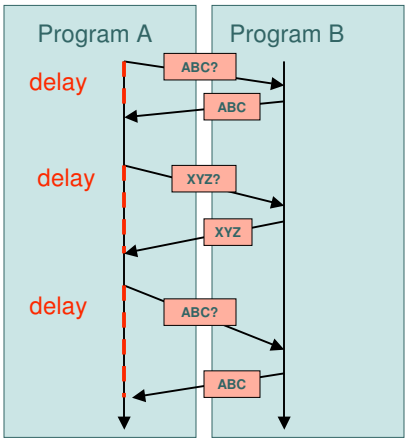
CS514

- If goal is mainly latency, then remove large objects
- If goal is mainly to reduce bandwidth (I.e. site cache), then remove least used
- Strategies very dependent on "workload"
  - Jia Wang survey: no strategy performed best over all workloads

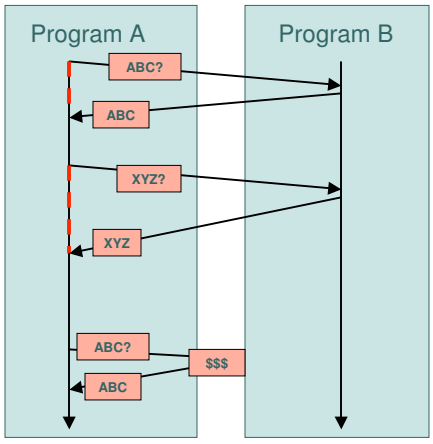
Ken's Caching picture (lecture 5)

CS514

- Avoid



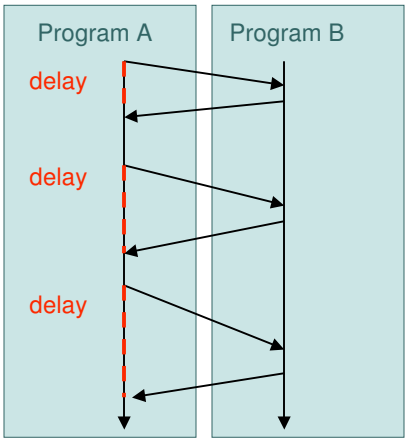
- Better



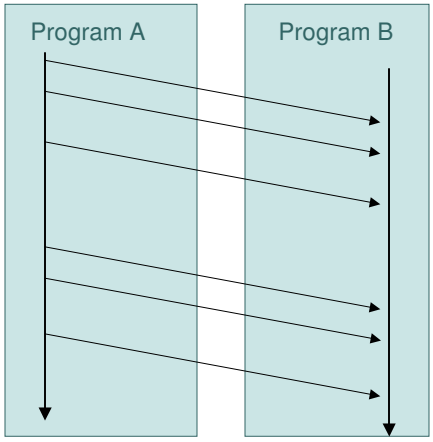
Ken's latency picture (lecture 5)

CS514

- Avoid



- Better (within limits)



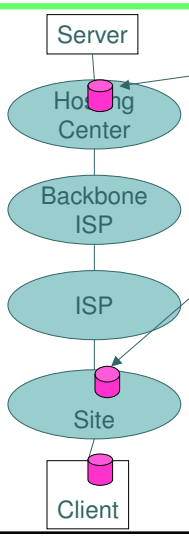
AT&T study: caching connections useful too

CS514

- Cache maintains persistent connection with server even after client disconnects
  - If another client connects soon, can reuse connection without new TCP handshake
- Shown to be at least as effective as data caching
- Study also showed that aborted connections can increase bandwidth usage
  - Client aborts connection, but cache continues to download, or already downloaded content

Connection caches

CS514



Valuable here because offload frequent TCP setups from server

Valuable here because ISP access link is the bottleneck link.
- But requires sufficient locality of reference

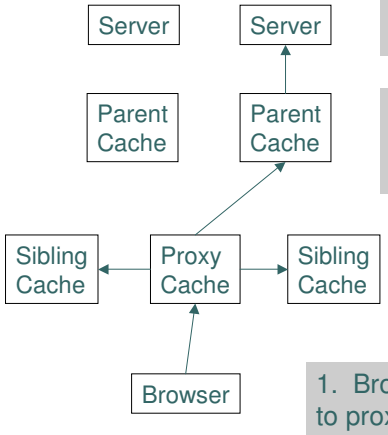
Prefetching

CS514

- Browser (or proxy) tries to predict what content the user is likely to request next
  - Various strategies
- Can help latency
  - (40% - 60%) for high bandwidth clients
- But can increase network traffic and burstiness
  - Doubling of network traffic seen
- Potentially more useful for mobile (disconnected) users (hoarding)
  - If not paying by the byte…

Cache hierarchies (Internet Cache Protocol, ICP)

CS514



4. If miss, parent cache asks server

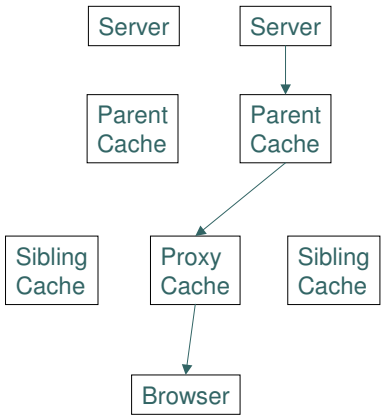
3. If siblings miss, proxy cache asks parent nearest to server

2. If miss, proxy cache may ask sibling caches (possibly with multicast)

1. Browser makes query to proxy cache

Cache hierarchies (Internet Cache Protocol, ICP)

CS514



Content cached on the way back

It seemed like a good idea at the time, but…
- Lots of overhead
- Bottleneck and misses at the top
- Latency at each cache
- Hard to select parent cache near server

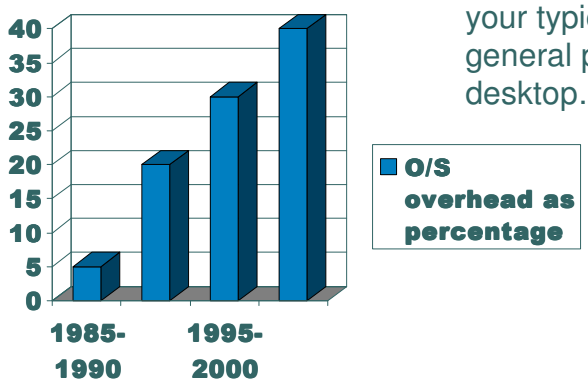
Distributed caching

CS514

- All caches are siblings
- Select sibling cache likely to have content
  - Typically by hashing URL
- If sibling doesn't have it, query to origin
- P2P version: Use browser caches as sibling caches!
  - This keeps popping up as a "good idea"
  - But remember Ken's O/S latency picture

O/S latency: the most expensive overhead on LAN communication!

CS514



And even worse for your typical windows general purpose desktop…

Akamai versus caching hierarchies

CS514

- Akamai limited only to cacheable content
- Akamai channels certain content through certain servers
  - Increases hit rate
- Akamai smart about load balancing
  - …and maybe sort-of smart about nearness to client…