Internet Protocols
Fall 2004

Lecture 20

Web Caching & CDNs

Andreas Terzis

Outline

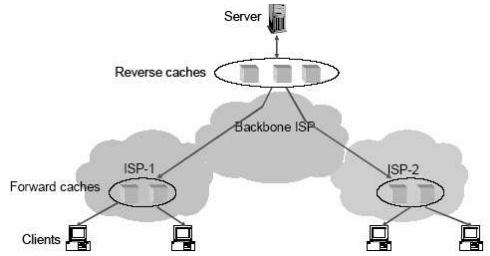
- HTTP Caching
  - Protocol Mechanisms
  - Reverse and Forward Caching
- CDNs
  - Akamai
  - URL Routing

Why web caching?

- Client-server architecture is inherently not scalable
  - Proxies: a level of indirection
- Reduce client response time
  - Direct and indirect effect
  - Less load on the server:
    - Server does not have to over-provision for *slashdot* effect
- Reduce network bandwidth usage
  - Wide area vs. local area use
  - These two objectives are often in conflict
    - May do exhaustive local search to avoid using wide area bandwidth
    - Prefetching uses extra bandwidth to reduce client response time
- New: *DDoS Protection*
  - dissipate attack over massive resources
  - multiplicatively raise level of resources needed to attack

Cache Placement

- Reverse Caches
  - Cache documents close to the server -> decrease server load
  - Done by content providers
- Forward Caches
  - Cache documents close to the clients -> Reduce network traffic and decrease latency
  - Done by ISPs or corporations



Client3

HTTP/1.0 Caching

- Exploit locality of reference
- A modifier to the GET request:
  - If-modified-since – return a "not modified" response if resource was not modified since specified time
- A response header:
  - Expires – specify to the client for how long it is safe to cache the resource
- A request directive:
  - No-cache – ignore all caches and get resource directly from server
- These features can be best taken advantage of with HTTP proxies
  - Locality of reference increases if many clients share a proxy

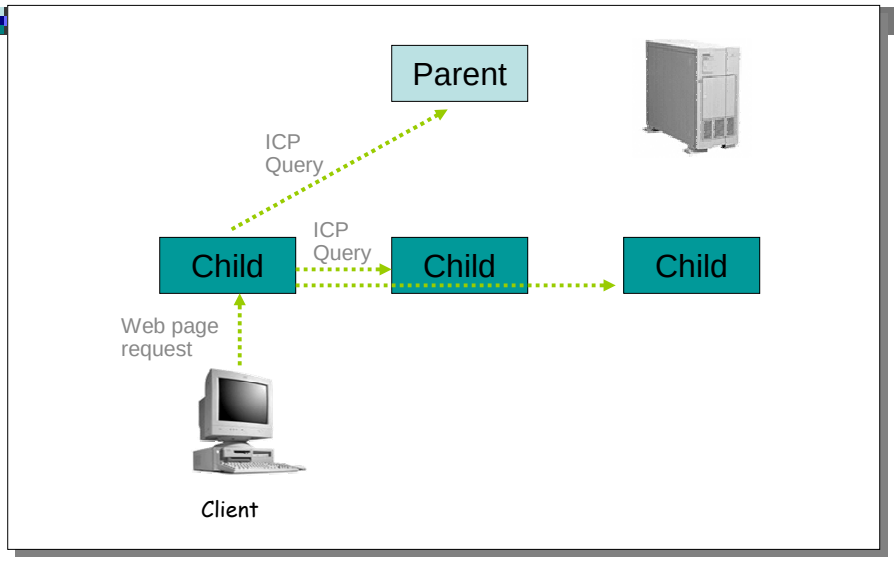
Cache Hierarchies

- Use hierarchy to scale a proxy
  - Why?
    - Larger population = higher hit rate (less compulsory misses)
    - Larger effective cache size
  - Why is population for single proxy limited?
    - Performance, administration, policy, etc.
- NLANR cache hierarchy
  - Most popular
  - 9 top level caches
  - Internet Cache Protocol based (ICP)
  - Squid/Harvest proxy
- How to locate content?

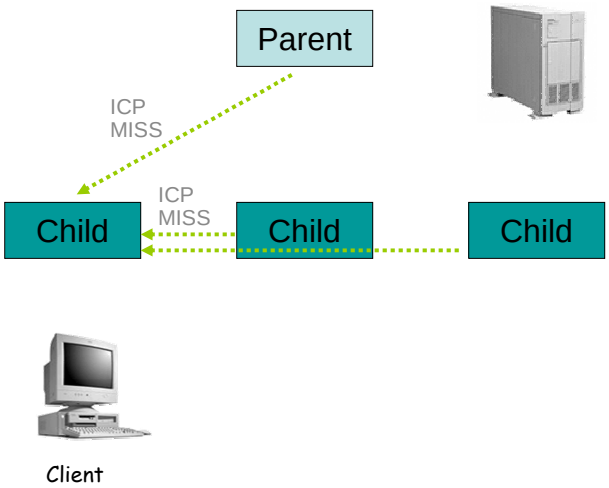
ICP (Internet cache protocol)

- Simple protocol to query another cache for content
- Uses UDP – why?
- ICP message contents
  - Type – query, hit, hit_obj, miss
  - Other – identifier, URL, version, sender address
  - Special message types used with UDP echo port
    - Used to probe server or "dumb cache"
- Query and then wait till time-out (2 sec)
- Transfers between caches still done using HTTP

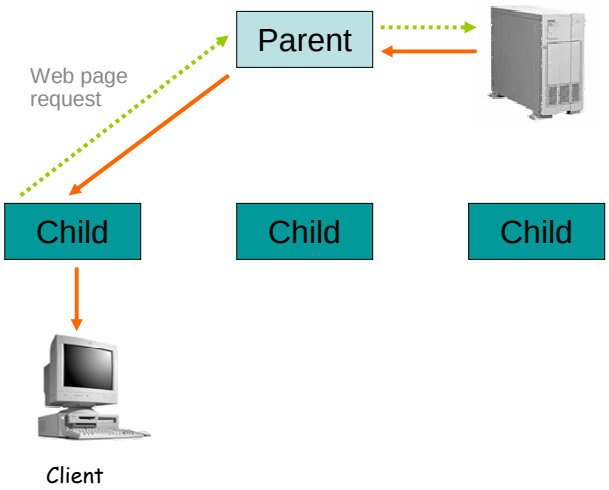
Squid



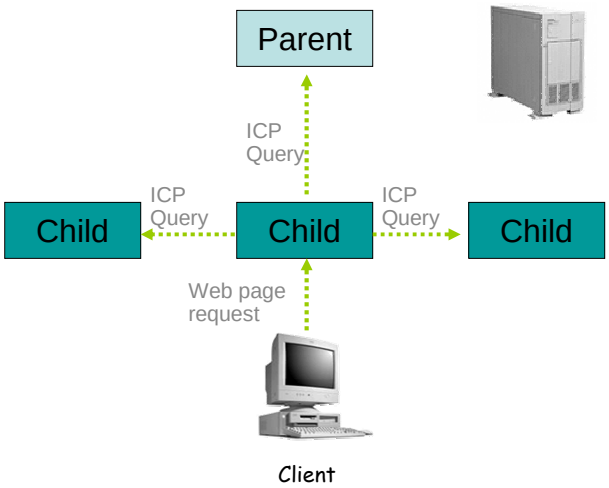
Squid


Parent
ICP
MISS
ICP
MISS
Child
Child
Child
Client

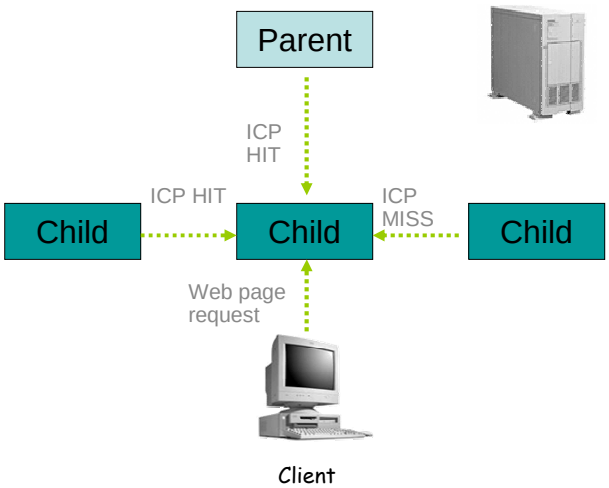
Squid


Parent
Web page
request
Child
Child
Child
Client

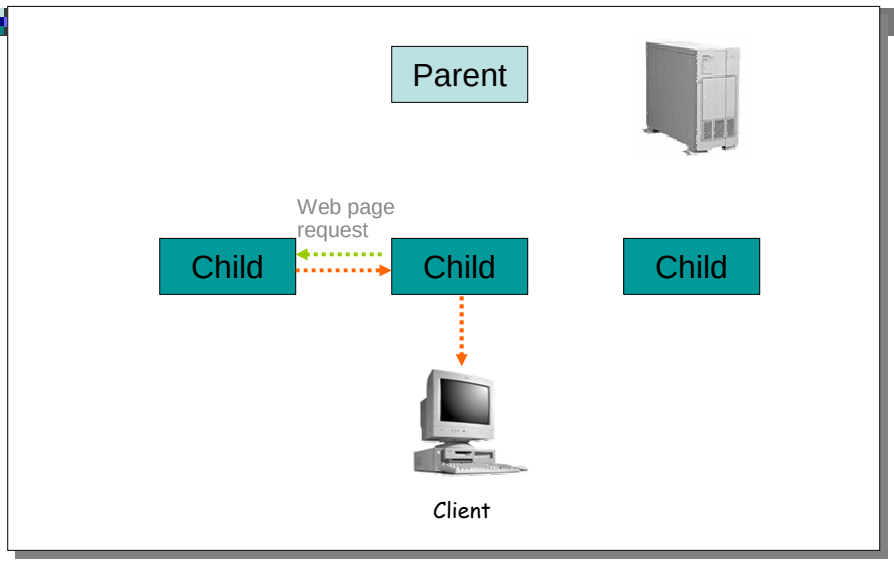
Squid


Parent
ICP
Query
Child
ICP
Query
Child
ICP
Query
Child
Web page
request
Client

Squid


Parent
ICP
HIT
Child
ICP HIT
Child
ICP
MISS
Child
Web page
request
Client

Squid



Optimal Cache Mesh Behavior

- Ideally, want the cache mesh to behave as a single cache with equivalent capacity and processing capability
- ICP: many copies of popular objects created – capacity wasted
- More than one hop needed for searching object
- Locate content – *how?*

Hinting

- Have proxies store content as well as metadata about contents of other proxies (hints)
  - Minimizes number of hops through mesh
  - Size of hint cache is a concern – size of key vs. size of document
- Having hints can help consistency
  - Makes it possible to push updated documents or invalidations to other caches
- How to keep hints up-to-date?
  - Not critical – incorrect hint results in extra lookups, not incorrect behavior
  - Can batch updates to peers

Summary Cache

- Primary innovation – use of compact representation of cache contents
  - Typical cache has 80 GB of space and 8KB objects → 10 M objects
  - Using 16byte MD5 → 160 MB per peer
  - Solution: Bloom filters
- Delayed propagation of hints
  - Waits until threshold %age of cached documents are not in summary
  - Perhaps should have looked at %age of false hits?

Bloom Filters

- Proxy contents summarize as a M bit value
- Each page stored contributes k hash values in range [1..M]
  - Bits corresponding to the k hashes set in summary
- Check for page = if all k hash bits corresponding to a page are set in summary, it is likely that proxy has summary
- Tradeoff → false positives
  - Larger M reduces false positives
  - What should M be? 8-16 * number of pages seems to work well
  - What about k? Is related to (M/number of pages) → 4 works for above M

Problems with caching

- Over 50% of all HTTP objects are uncacheable.
- Sources:
  - Dynamic data → stock prices, frequently updated content
  - CGI scripts → results based on passed parameters
  - SSL → encrypted data is not cacheable
    - Most web clients don't handle mixed pages well →many generic objects transferred with SSL
  - Cookies → results may be based on passed data
  - Hit metering → owner wants to measure # of hits for revenue, etc, so, cache busting

CDN

- Replicate content on many servers
- Challenges
  - How to replicate content
  - Where to replicate content
  - How to find replicated content
  - How to choose among known replicas
  - How to direct clients towards replica
    - DNS, HTTP 304 response, anycast, etc.
- Akamai

Server Selection

- Service is replicated in many places in network
- How to direct clients to a particular server?
  - As part of routing → anycast, cluster load balancing
  - As part of application → HTTP redirect
  - As part of naming → DNS
- Which server?
  - Lowest load → to balance load on servers
  - Best performance → to improve client performance
    - Based on Geography? RTT? Throughput? Load?
  - Any alive node → to provide fault tolerance

Routing Based

- Anycast
  - Give service a single IP address
  - Each node implementing service advertises route to address
  - Packets get routed from client to "closest" service node
    - Closest is defined by routing metrics
    - May not mirror performance/application needs
  - What about the stability of routes?

Routing Based

- Cluster load balancing
  - Router in front of cluster of nodes directs packets to server
  - Can only look at global address (L3 switching)
  - Often want to do this on a connection by connection basis – why?
    - Forces router to keep per connection state
    - L4 switching – transport headers, port numbers
  - How to choose server
    - Easiest to decide based on arrival of first packet in exchange
    - Primarily based on local load
    - Can be based on later packets (e.g. HTTP Get request) but makes system more complex

L-7 switching

- Interpret requests, content-aware switches
- Have to do the initial hand-shake
- Different proxies for different content-types
- Load balancing vs locality
- Locality means all requests (even to a popular object) serviced by a single proxy
- Caching alleviates the above problem. Why?

Application Based

- HTTP supports simple way to indicate that Web page has moved
- Server gets Get request from client
  - Decides which server is best suited for particular client and object
  - Returns HTTP redirect to that server
- Can make informed application specific decision
- May introduce additional overhead → multiple connection setup, name lookups, etc.
- While good solution in general HTTP Redirect has some design flaws – especially with current browsers?

Naming Based

- Client does name lookup for service
- Name server chooses appropriate server address
- What information can it base decision on?
  - Server load/location → must be collected
  - Name service client
    - Typically the local name server for client
- Round-robin
  - Randomly choose replica
  - Avoid hot-spots
- [Semi-]static metrics
  - Geography
  - Route metrics
  - How well would these work?

How Akamai Works

- Clients fetch html document from primary server
  - E.g. fetch index.html from cnn.com
- URLs for replicated content are replaced in html (Akamaization)
  - E.g. <img src="http://cnn.com/af/x.gif"> replaced with <img src="http://a73.g.akamaitech.net/7/23/cnn.com/af/x.gif">
- Client is forced to resolve aXYZ.g.akamaitech.net hostname

How Akamai Works

- How is content replicated?
- Akamai only replicates static content
  - Serves about 7% of the Internet traffic !
- Modified name contains original file
- Akamai server is asked for content
  - First checks local cache
  - If not in cache, requests file from primary server and caches file

How Akamai Works

- Root server gives NS record for akamai.net
- Akamai.net name server returns NS record for g.akamaitech.net
  - Name server chosen to be in region of client's name server
  - TTL is large
- G.akamaitech.net nameserver choses server in region
  - Should try to chose server that has file in cache - How to choose?
  - Uses aXYZ name and consistent hash
  - TTL is small

Hashing

- Advantages
  - Let the CDN nodes are numbered 1..m
  - Client uses a ***good*** hash function to map a URL to 1..m
  - Say hash (url) = *x*, so, client fetches content from node *x*
  - No duplication – not being fault tolerant.
  - One hop access
  - Any problems?
    - What happens if a node goes down?
    - What happens if a node comes back up?
    - What if different nodes have different views?

Robust hashing

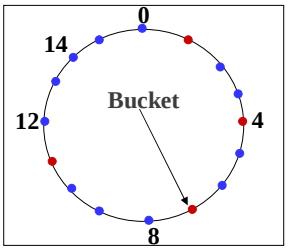
- Let 90 documents, node 1..9, node 10 which was dead is alive again
- % of documents in the wrong node?
  - 10, 19-20, 28-30, 37-40, 46-50, 55-60, 64-70, 73-80, 82-90
  - *Disruption coefficient* = $\frac{1}{2}$
  - Unacceptable, use consistent hashing – idea behind Akamai!

Consistent Hash

- "view" = subset of all hash buckets that are visible
- Desired features
  - Balanced – in any one view, load is equal across buckets
  - Smoothness – little impact on hash bucket contents when buckets are added/removed
  - Spread – small set of hash buckets that may hold an object regardless of views
  - Load – across all views # of objects assigned to hash bucket is small

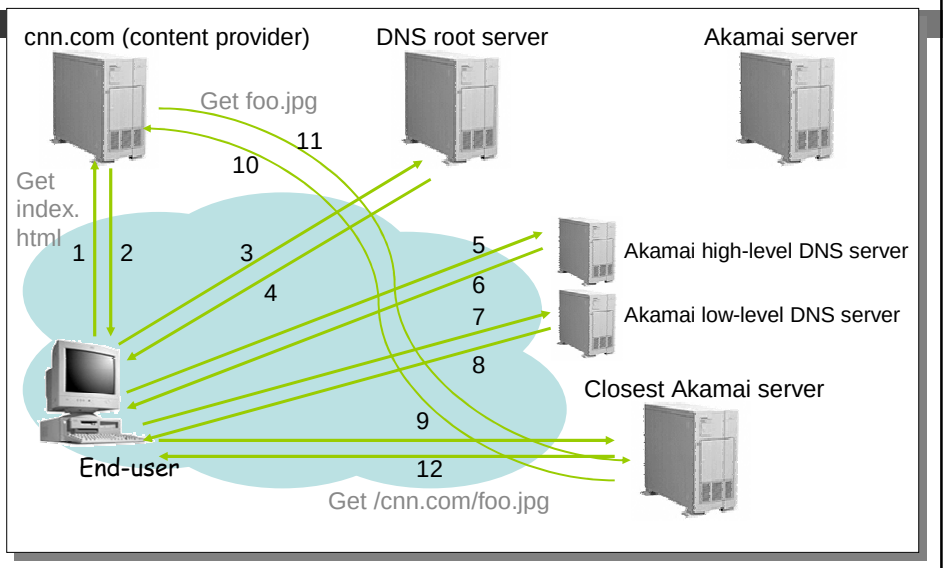
Consistent Hash – Example

- Construction
  - Assign each of C hash buckets to random points on mod 2^n circle, where, hash key size = *n.*
  - Map object to random position on circle
  - Hash of object = closest clockwise bucket

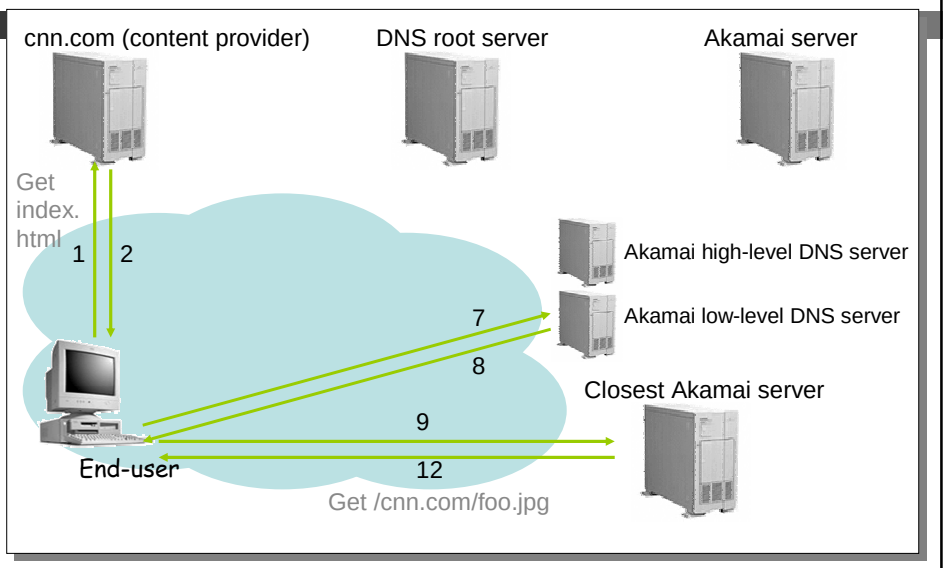


- Smoothness → addition of bucket does not cause much movement between existing buckets
- Spread & Load → small set of buckets that lie near object
- Balance → no bucket is responsible for large number of objects

How Akamai Works

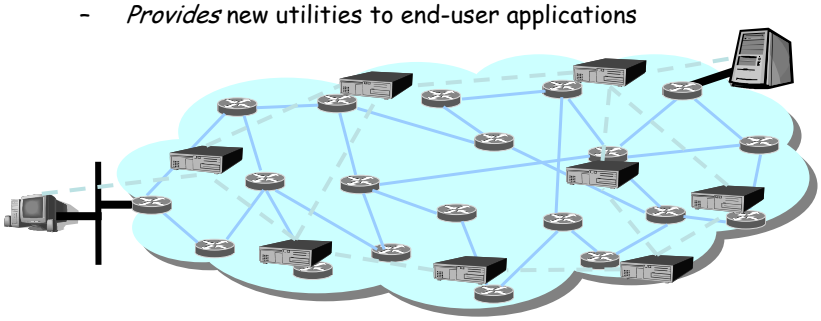


Akamai – Subsequent Requests



Internet services so far

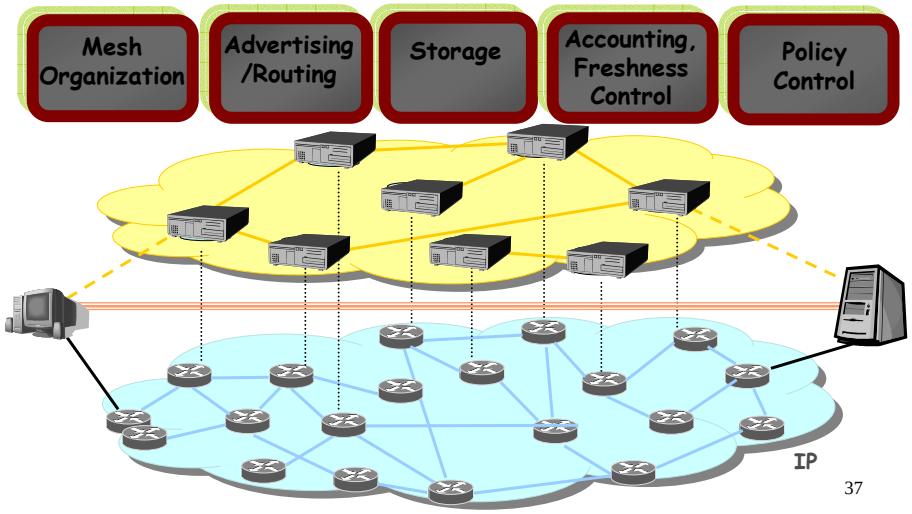
- Internet so far offers reliable deliver of packets
  - Plus DNS and applications such as email and web
- Add a Service Infrastructure at the application-level on top of the Internet IP delivery infrastructure
  - *Shared* among end-user applications
  - *Uses* underlying network resources
  - *Provides* new utilities to end-user applications



Why do we need Application Network?

- Unifying architecture for
  - Enhancing existing applications
    - Increase the performance
    - Increase the reliability/availability
    - Increase the scalability
  - Enabling new network utilities
    - Application level multicast
    - Global file systems
    - Ubiquitous access to information
      - Extranets, wireless devices

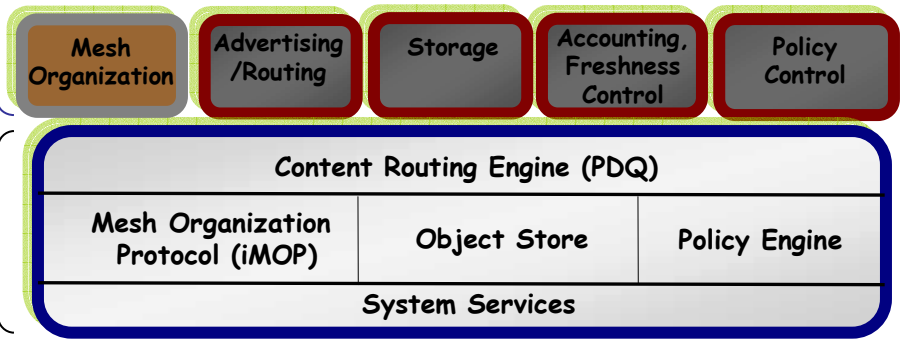
Application Network Primitives



37

Application Network Node

- *Application Network Primitives*



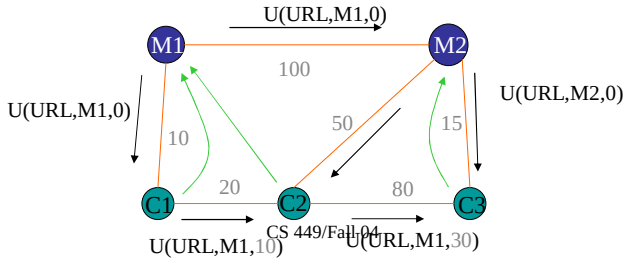
- *Core System: Application Network Node*

PDQ Problem Statement

- PDQ stands at the core of the AN
- Issues related to PDQ
  - Create and maintain connections among AN nodes
  - Propagate information about application resources
  - Forward client requests based on resource names
  - Distribute freshness control signals
  - Collect accounting information

CDN Support

- CDN Service: Replicate content to various nodes in the AN → Content Mirrors
- Mirrors advertise their capacity to serve content by sending UPDATE messages
- UPDATE message contains the URL prefix mirrored, the mirror's address and the distance to the mirror
- Nodes choose the closest mirror according to metric

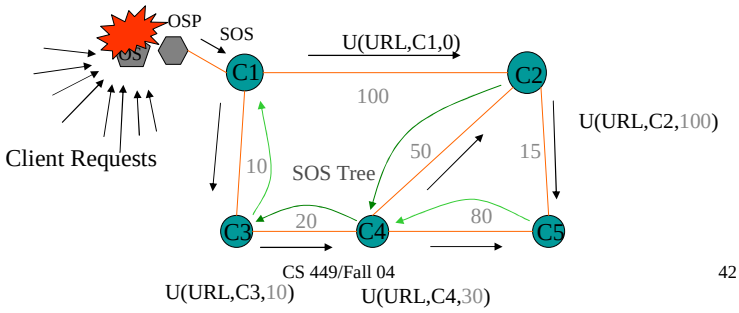


CDN Support (II)

- Multiple mirrors for the same site form a *mirror tree*
  - Only root of the mirror tree has to fetch pages from OS
  - Accounting information aggregates up the mirror tree
- Mirror tree members and MirrorHead are configured
- Mirror Head sends special UPDATE
  - Members of mirror tree choose parent using the same algorithm used to choose mirror

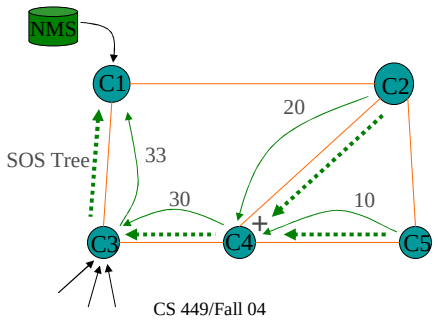
Hot Spot Insurance (HSI)

- OSP detects when OS becomes overloaded and sends request to forward reachability information (SOS)
- Nodes forward SOS message to all neighbors after updating metric
- SOS tree is created



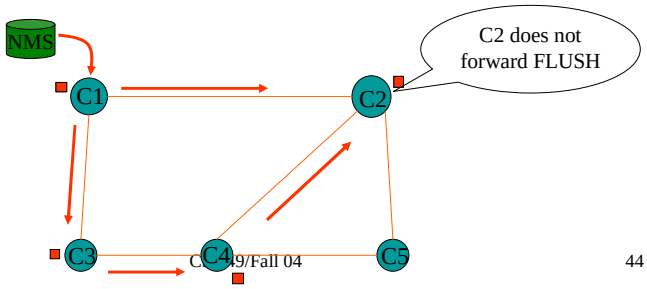
Collection of Accounting Info

- Hit Metering information travels upstream the (mirror) tree
- Each node aggregates information from children before forwarding



Content Freshness

- Content freshness is implemented by means of FLUSH messages
- FLUSH messages carry the URL to be removed
- FLUSH messages are flooded through the AN
  - Looping is controlled by TTL
  - Placeholder entries (that eventually time out)



Failure Modes

- Node fails
- Link goes down
- Route expires

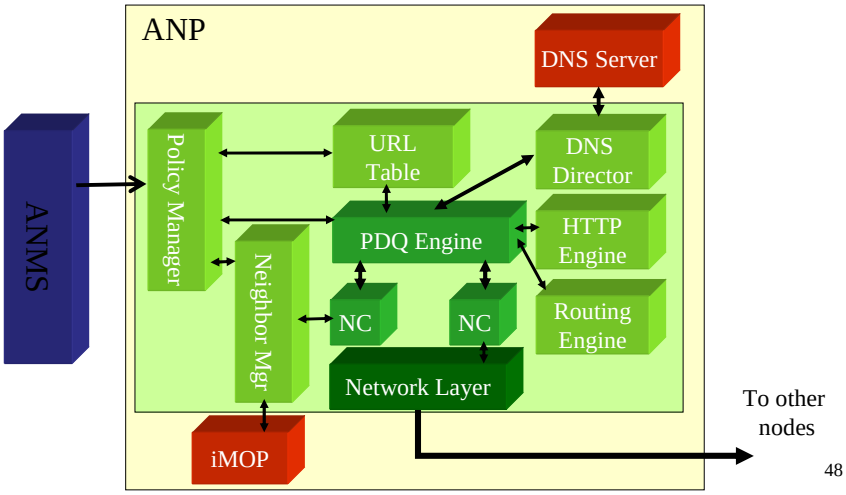
Node Failure

- Peer detects node failure
  - TCP connection gets torn down
  - KEEPALIVEs are not received
- Routes learned from that peer invalidated
- Infinity metric sent to all other peers
- Over the next refresh interval neighbors are going to announce alternate routes to the same destinations
- Link removal is equivalent to node failure from peer's standpoint

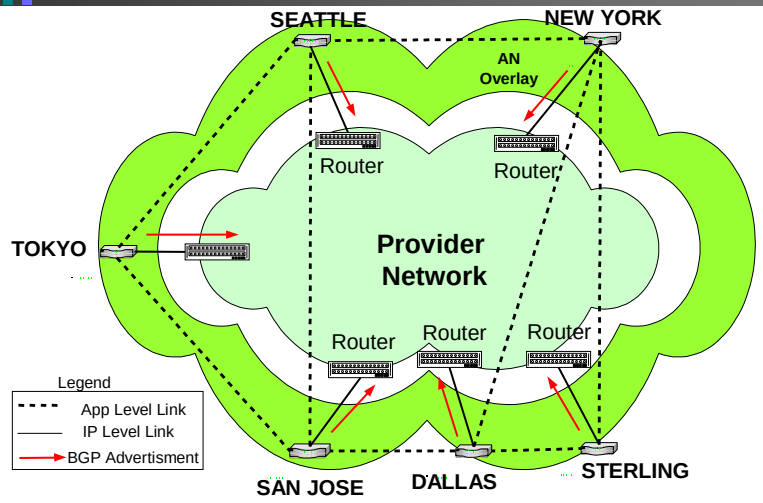
Route Expiration

- "Soft State" Model → all routes periodically refreshed
- When a node does not hear an update for three refresh periods it considers the route expired
  - Route is not used but is not deleted
  - Route is deleted in the next refresh period
- Route stays "locked" for a short period of time to avoid routing loops

Software Architecture



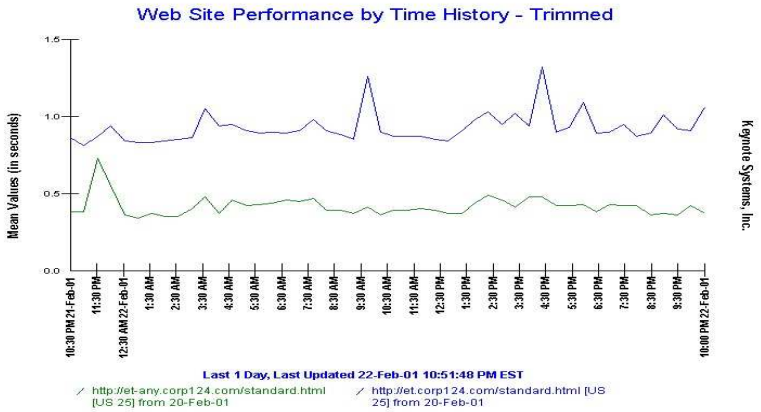
Deployed Topology



Measurement Methodology

- Placed web page on Origin Server
- Placed same web page on AN of 5 nodes, two commercial CDNs
- Used Keynote service
  - Downloads page from 15 US locations every 30 min both from OS and from AN
  - Provides average delay across all locations
  - Organized per geographic location

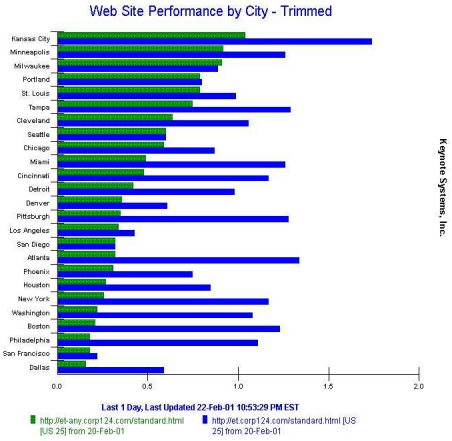
Delay compared to OS



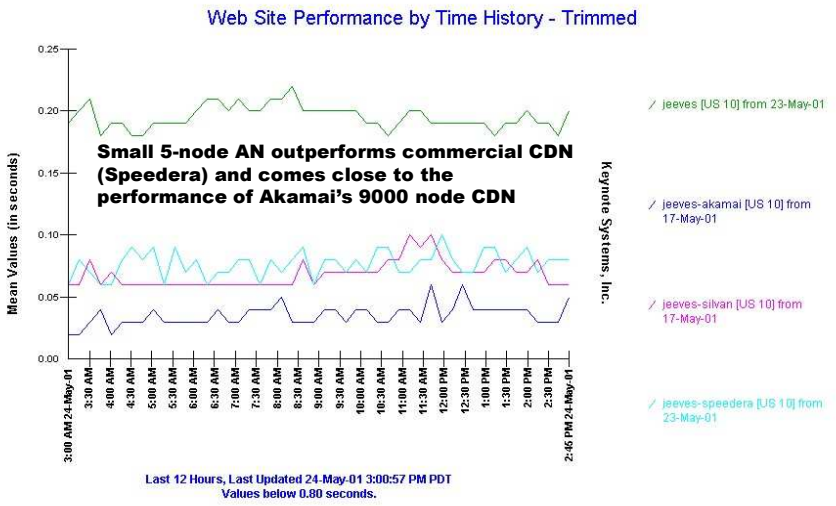
- 50% improvement compared to OS

Delay sorted per city

- Up to 6X Improvement with only 5 nodes in the U.S.
- Results from U.S. agents alone.
- Improvement from global agents expected to be larger.

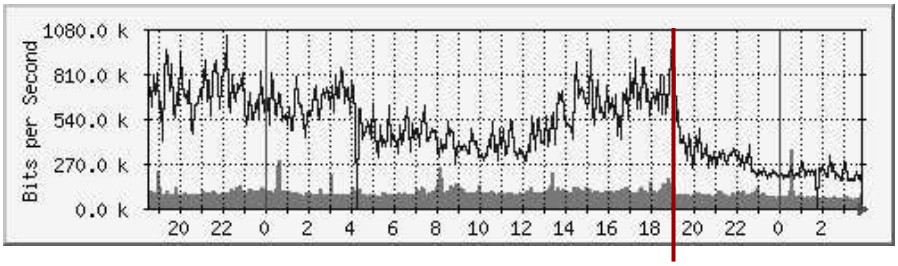


Delay Compared to commercial CDNs



Bandwidth reduction at OS

- Web-site bandwidth before and after use of AN
  - The web server serves a number of websites
- Dramatic bandwidth reduction at the web server after AN is enabled



Resiliency results

- ANP closest to client was purposely shutdown
- Client retrieved content from next best node
- Switched to best node after node became available again
- Spikes are due to noise in measurements