SCOM/ SRSI
Caching, Content Delivery Networks

Ana Aguiar

DEEC, FEUP

2018-19

CACHING

- Caching
  - Why
  - Where
  - How
  - HTTP/1.1 headers

Caching

- Store web contents with high number of hits in locations from where they can be more efficiently downloaded
- Cache Hit ≠ Cache Miss
  - On a cache hit, retrieve local copy of the page
  - On a cache miss, retrieve page from original server

Caching

Reduce the need to
- send requests
- send full responses

Web performance improvement
- Better response times

Higher availability
- Lower server load
- Lower network load

(Limited) Disconnected operation

At the cost of possibly stale data

Web Cache Challenges

- How to cache?
- How long to cache?
- Where to cache?

Web Cache Challenges

- How to cache?
  - On-demand
  - Pre-fetch
- How long to cache?
  - Freshness based on expiration timestamp
- Where to cache?
  - At the server, at intermediate nodes, at the client
- Caching requires
  - Large amount of space
  - Fast search methods

Where to Cache?

- At the client
  - Frequently fetched pages must not be retrieved
- Near the client
  - Reduce traffic to distant servers
  - Reduce response time
- At or near the server
  - Reduce load on main server
  - Same cache can serve many clients
    - Higher hit rates

Homework

Consider an institutional network connected to the Internet. Suppose that the average object size is 850,000 bits and that the average request rate from the institution's browsers to the origin servers is 16 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is three seconds on average.

Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use $\Delta/(1 - \Delta b)$, where Δ is the average time required to send an object over the access link and b is the arrival rate of objects to the access link.

a. Find the total average response time.

b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.4. Find the total response time.

J. Kurose, K. Ross. "Computer Networking: a top down approach" Chapter 2, P9

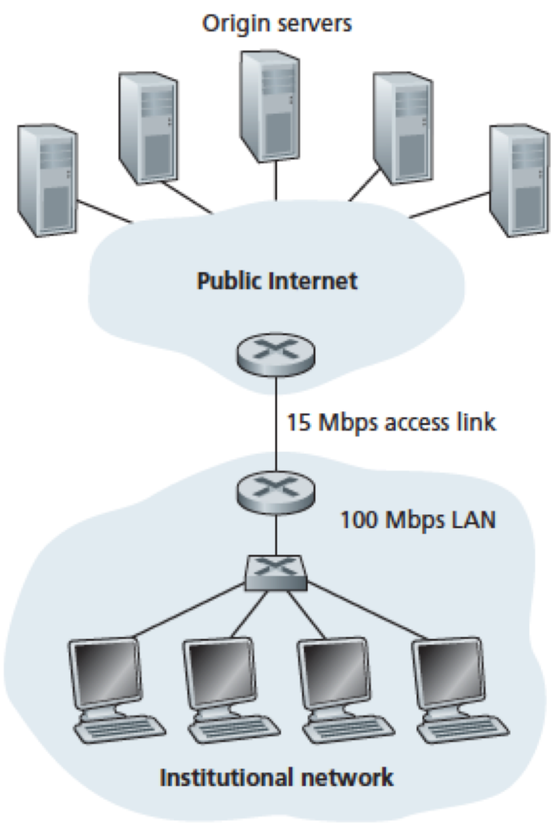


Fig 2.1, Kurose et al "Computer Networks: a Top Down Approach"

WWW Explicit Caching

- Local caching
  - Store visited pages and their elements locally
  - Reduce network accesses
  - Very fast
- Network caching
  - Proxies: machines with whom the local machine communicates instead of the original server
  - Proxies can change headers
- https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html

HTTP/1.1 Caching Headers

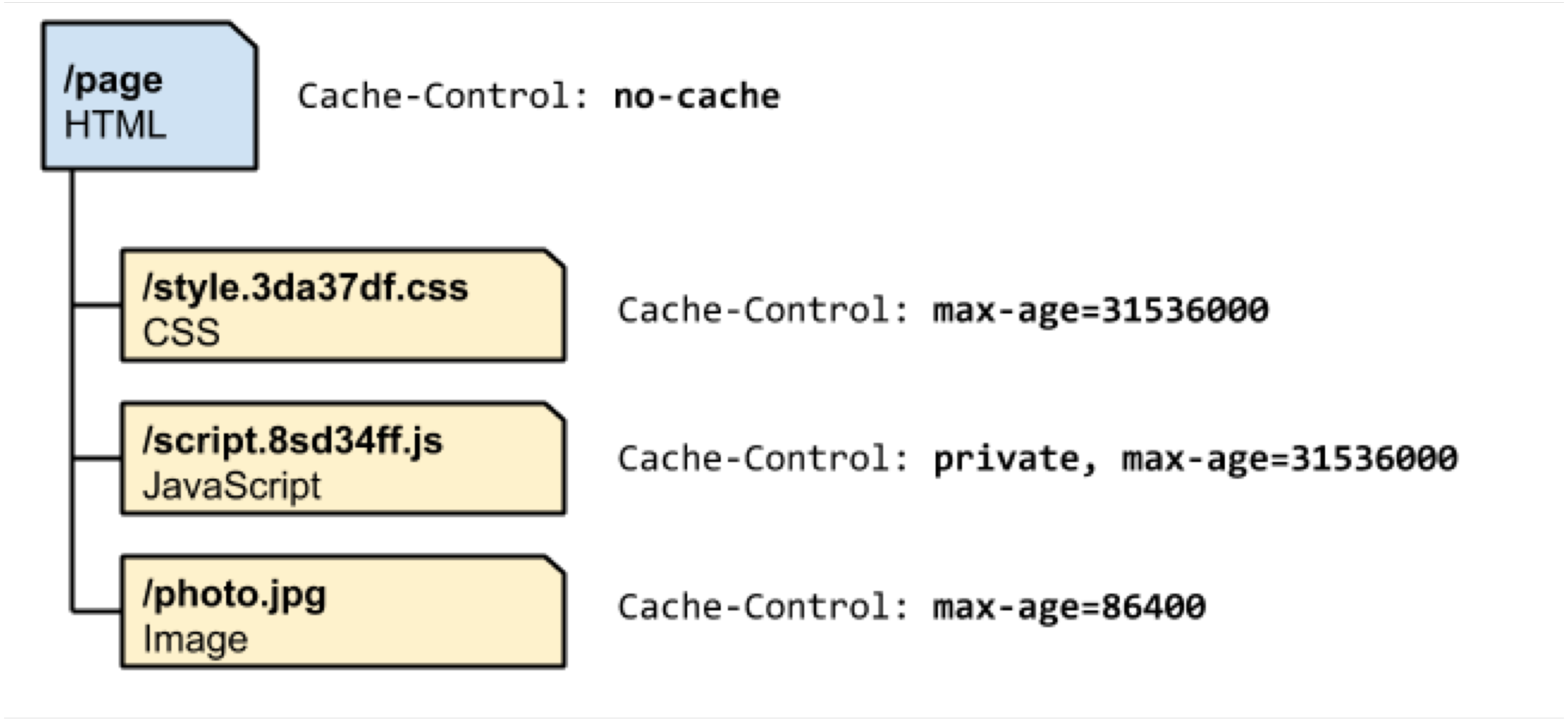
- Content identification
  - etag: token that identifies content
- Cache-Control directives
  - no-cache: require validation
  - no-store: disallow caching at all
  - public: allow caching with authorisation token
  - private: allow caching only in browser
  - max-age: how long is the content considered fresh
  - must-revalidate: always requires validation
- Conditional requests
  - if-none-match
  - if-modified-since

More info: https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching

Follow-Up Questions

- Which elements of a web page can be cached?
- Does it make sense to cache a shopping cart?
  - If yes, where?
- How can you update a cached resource?

HTTP/1.1 Caching Design



/page
HTML
Cache-Control: no-cache
/style.3da37df.css
CSS
Cache-Control: max-age=31536000
/script.8sd34ff.js
JavaScript
Cache-Control: private, max-age=31536000
/photo.jpg
Image
Cache-Control: max-age=86400

https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching

COOPERATIVE CACHES

Cooperative Caching

- Single cache
  - Single point of failure
  - Limited capacity: storage, processing
- Cooperative caches
  - Look for page on cooperating caches on miss
- How to locate page on cooperating caches?
  - Directory: expensive to keep up-to-date; new point of failure; costly do keep up to date
  - Broadcast: causes traffic, latency waiting for all replies on a miss, can cause replication of cached contents

Hash Functions

- Algorithms that map large sets onto limited sets
  - Results are called hash value or simply hash
  - Deterministic: a value always produces the same hash
  - Uniform: hash values are evenly distributed
  - Fast to compute
  - Examples: modulo function, Bloom filter
- Impact of regular hash functions on caching
  - Cause large number of pages to move when number of caches changes
  - Different views of the caches caused by asynchronous information propagation causes content replication and load imbalance

Cooperative Caches with Consistent Hashing

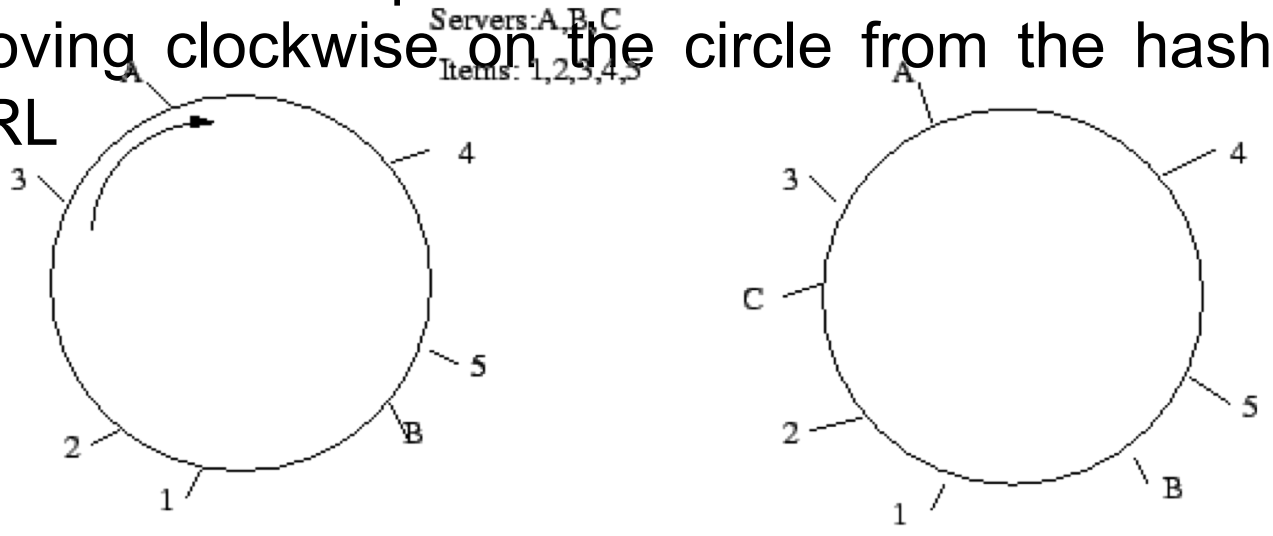
- Unicast is sufficient to get object or discover a miss
- Faster in discovering a miss
- No maintenance or query overhead
- No single point of failure
- No need for an intermediary cache
- No redundant copies reduce miss rate

Consistent Hashing

– Servers and pages (URLs) mapped onto a same identifier space, for example a circle

– Rule to place contents on servers:

- URL contents placed on first server encountered moving clockwise on the circle from the hash of the URL



A. Aguiar

Consistent Hashing

- Implementation
  - Store hash of all cache servers in a binary tree
  - Server for a key can be found with one search
  - Complexity: O(log n)
    - What does this mean?

Consistent Hashing

- Consistent hashing provides
  - Little impact on each server upon change in number of servers (addition/ deletion)
  - Small amount of different servers holding same contents
  - Balanced amount of pages assigned to each server across all views
- Even when different views exist
  - Because hash function randomly distributed caches and hashes in the identifier space

A Glimpse of Gains

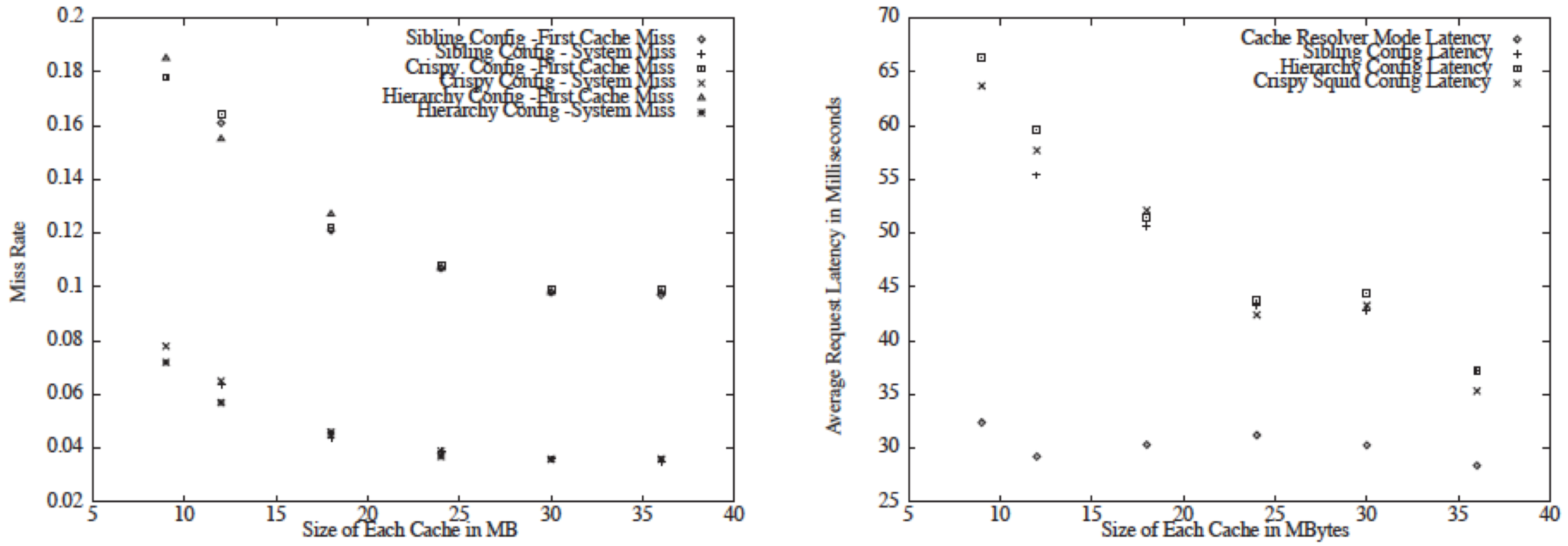


Fig. 3. Miss rates and latencies of three additional configurations.

David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins, and Yoav Yerushalmi. 1999. **Web caching with consistent hashing**

Resources

- **Mandatory reading**: David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins, and Yoav Yerushalmi. 1999. **Web caching with consistent hashing**. In Proceedings of the eighth international conference on World Wide Web (WWW '99), Philip H. Enslow, Jr. (Ed.). Elsevier North-Holland, Inc., New York, NY, USA, 1203-1213.
- **Additional resource:** David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. 1997. **Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web**. In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC '97). ACM, New York, NY, USA, 654-663. DOI=http://dx.doi.org/10.1145/258533.258660

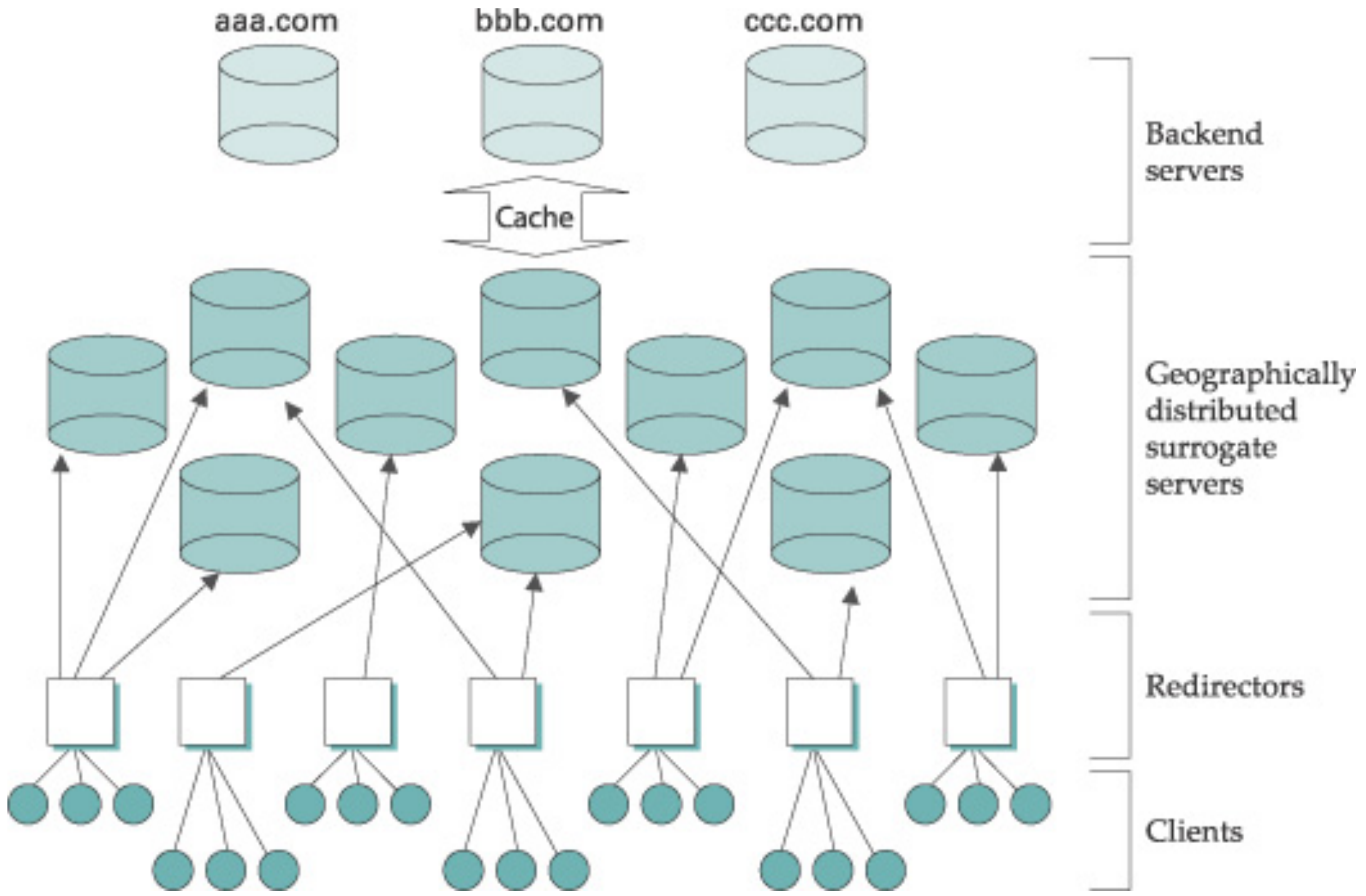
CONTENT DELIVERY NETWORKS

Content Delivery Network

•Overlay networks that use distributed algorithms on multiple servers to deliver contents with low latency (low miss rate and round trip time)

- Proactively replicate some content
- Replicate content that does not change frequently
- Retrieve content on a miss
- Re-direct clients to appropriate servers
  - Provide the best response time to the client
  - Achieve the highest possible system throughput = number of processed requests per second

CDN Architecture



CDN Challenges

- How to replicate content?
- Where to replicate content?
- How to find replicated content?
- How to choose among replicas?
- How to direct clients towards replica?

$\left.\begin{array}{l} \text{How to replicate content?} \\ \text{Where to replicate content?} \\ \text{How to find replicated content?} \end{array}\right]$ DHT, e. g. Consistent hashing

CDN Redirection Decision

- Conflicting goals!

- Minimising reponse time favours servers in the network proximity
- Overall system throughput improves if load is balanced across servers
- Thoughput and response time improve if the server already has the desired contents

How to Choose Among Caches?

- Round robin or random
  - Good for load balance, but not for response time
  - Ignores proximity and locality

- Solution: add an additional level or indirection
  - Clusters of caches geographically spread
  - Resolve for proximity first
  - Then resolve for load balance and locality

How to redirect?

- DNS redirection
  - DNS server answers with the IP address of the target
- Embedded link re-writing
  - At the level of links or page elements
- Combination of the previous
- IP anycast
- HTTP Re-direct
  - Original server can be overloaded by re-direct
  - Additional round-trip time
  - Alternatively, redirector (proxy) close to the client

Using DNS as Re-Director

- Root DNS server returns CDN’s name server
- High level DNS server chooses local level DNS server “near” client
- Local level DNS server returns IP of a content server in a local cluster

- All further GET from that client will be directly forwarded to the cache chosen by the local DNS

Example

- Clients GET main page from primary server
  - E.g. GET www.cnn.com
- URLs for replicated content are substituted by “virtual cache” addresses
  - 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 must lookup a73.g.akamaitech.net
  - First resolution resolves g.akamaitech.net for proximity
  - Second resolution resolves a73.g.akamaitech.net inside the local cluster
    - a73 is the hash of the desired URL
- Server is asked for content
  - If miss occurs, server retrieves it from primary server and caches it

Using DNS as Re-Director

- All intelligence can be inside the CDN network
- CDN can optimise algorithms without changes outside their network
- Re-directors must monitor load and livelihood of cluster participants, inside each regional cluster

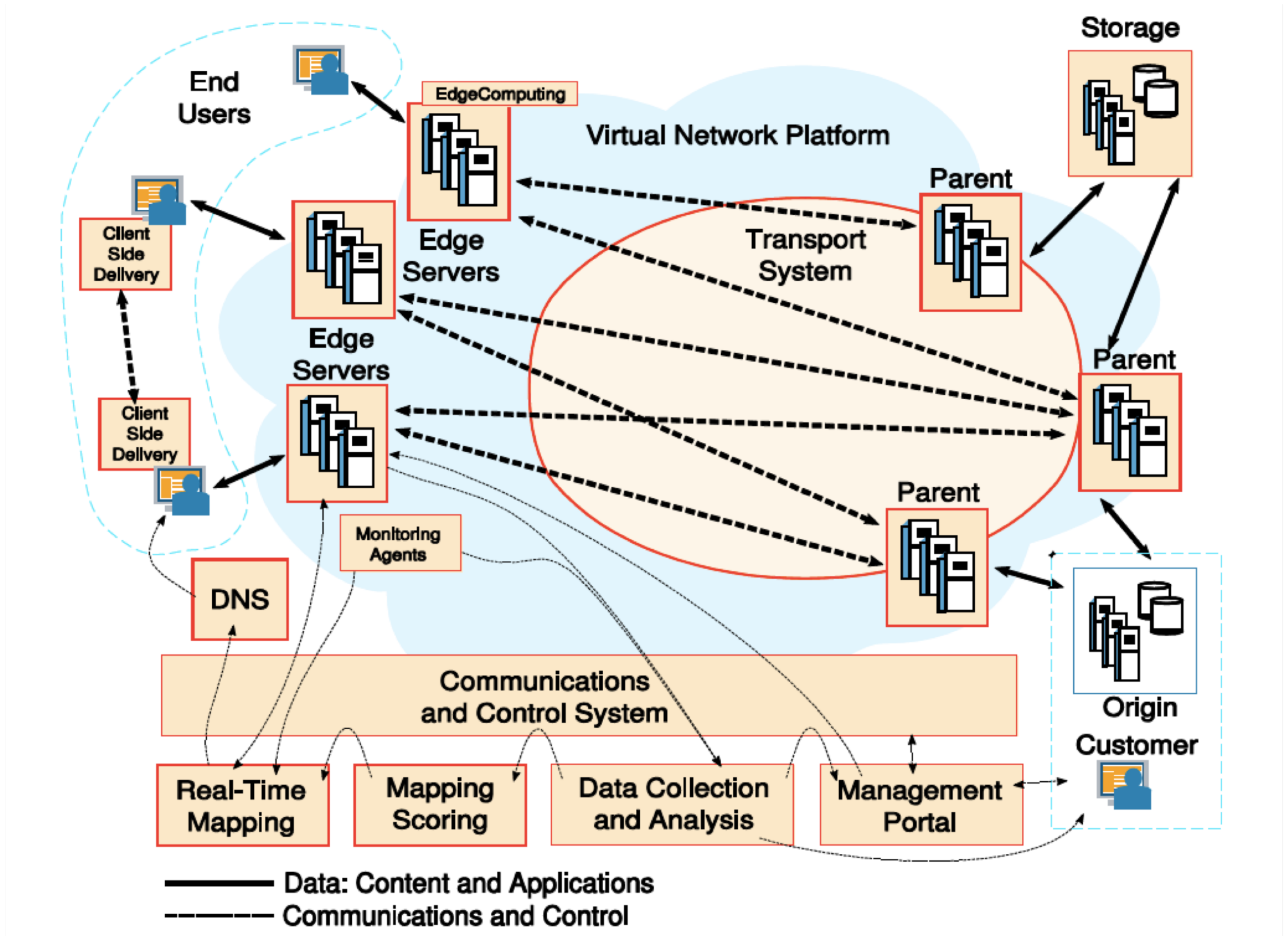
Load Balancing

- Goal: distribute load evenly among servers
- Algorithm:
  - Sort server list according to load
  - Search first server below threshold load
- As load increases, this scheme starts creating copies of the URL in less loaded servers
- Even less popular pages will be replicated, if they lie on busy servers

Akamai

- Result of a challenge of Tim Berners-Lee (WWW founder) to Tom Leighton (Applied Mathematics, Parallel Algorithms and Architectures), MIT professors
  - They developed consistent hashing and applied it to speeding up the web
- Founded in 1998 by Tom Leighton, Danny Levine (graduate student) and Jonathan Seelig (MIT Sloan Business School)
- First business plan was application to the MIT 50k Enterpreneurship Competition
- Today > 7000 employees, worth 8.7B USD on Nasdaq
- "240,000 servers in over 130 countries […]"
- 85% of Internet users within a single hop […]"

Anatomy of a CDN (Akamai)



Ramesh K. Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. “**Overlay Networks: An Akamai Perspective**“. In Advanced Content Delivery, Streaming, and Cloud Services, Ed. Pathan, Sitaraman, and Robinson, John Wiley & Sons, 2014

Overlay enables Optimisation

- Different behaviours for different content
  - Static web contents
  - Web applications and dynamic web contents
  - Streaming
- Protocol enhancements reduce latency
  - Path optimisation using monitoring data
  - Reduce packet loss, e.g. by using FEC
  - Proprietary transport protocol
    - Persistent connections
    - Optimal TCP window setting
    - Reduced timeouts due to accurate information on latency

Resources

- David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins, and Yoav Yerushalmi. 1999. **Web caching with consistent hashing**. In Proceedings of the eighth international conference on World Wide Web (WWW '99), Philip H. Enslow, Jr. (Ed.). Elsevier North-Holland, Inc., New York, NY, USA, 1203-1213.
- Ramesh K. Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. "**Overlay Networks: An Akamai Perspective**". In Advanced Content Delivery, Streaming, and Cloud Services, Ed. Pathan, Sitaraman, and Robinson, John Wiley & Sons, 2014.
- Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. "**The Akamai Network: A Platform for High-Performance Internet Applications**". ACM SIGOPS Operating Systems Review, Vol. 44, No.3, July 2010.
- Michael J. Freedman, Eric Freudenthal, and David Mazières. "**Democratizing Content Publication with Coral**". In Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04) San Francisco, CA, March 2004.

NEXT: PEER TO PEER NETWORKS