CSE 486/586 Distributed Systems Web Content Distribution---1 DNS & CDN

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Last Time

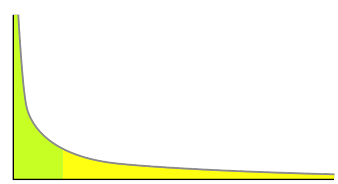
- RPC invoke semantics
  - At least once
  - At most once

Understanding Your Workload

- Engineering principle
  - Make the common case fast, and rare cases correct
  - (From Patterson & Hennessy books)
  - This principle cuts through generations of systems.
- Example?
  - CPU Cache
- Knowing common cases == understanding your workload
  - E.g., read dominated? Write dominated? Mixed?

Content Distribution Problem

- Power law (Zipf distribution)
  - Models a lot of natural phenomena
  - Social graphs, media popularity, wealth distribution, etc.
  - Happens in the Web too.

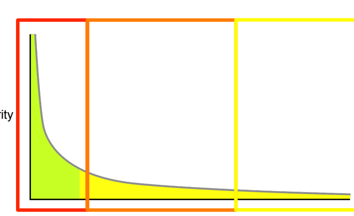


Content Distribution Workload

- What are the most frequent things you do on Facebook?
  - Read/write wall posts/comments/likes
  - View/upload photos
  - Very different in their characteristics
- Read/write wall posts/comments/likes
  - Mix of reads and writes so more care is necessary in terms of consistency
  - But small in size so probably less performance sensitive
- Photos
  - Write-once, read-many so less care is necessary in terms of consistency
  - But large in size so more performance sensitive

Facebook's Photo Distribution Problem

- "Hot" vs. "very warm" vs. "warm" photos
  - Hot: Popular, a lot of views
  - Very warm: Somewhat popular, still a lot of views
  - Warm: Unpopular, but still a lot of views in aggregate



“Hot” Photos

- How would you serve these photos?
- Caching should work well.
  - Many views for popular photos
- Where should you cache?
  - Close to users
- What’s commonly used these days?
  - CDN
  - CDN mostly relies on DNS, so we’ll look at DNS then CDN.
- (Very warm and warm: next two lectures)

CSE 486/586 Administrivia

- Nothing much!

Domain Name System (DNS)

Proposed in 1983 by Paul Mockapetris

Separating Names and IP Addresses

- Names are easier (for us!) to remember
  - www.cnn.com vs. 64.236.16.20
- IP addresses can change underneath
  - Move www.cnn.com to 173.15.201.39
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

Two Kinds of Identifiers

- Host name (e.g., www.cnn.com)
  - Mnemonic name appreciated *by humans*
  - Provides little (if any) information about location
  - Hierarchical, variable # of alpha-numeric characters
- IP address (e.g., 64.236.16.20)
  - Numerical address appreciated *by routers*
  - Related to host’s current location in the topology
  - Hierarchical name space of 32 bits

Hierarchical Assignment Processes

- Host name: www.cse.buffalo.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host
- IP addresses: 128.205.32.58
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP

Overview: Domain Name System

- A client-server architecture
  - The server-side is still distributed for scalability.
  - But the servers are still a hierarchy of clients and servers
- Computer science concepts underlying DNS
  - Indirection: names in place of addresses
  - Hierarchy: in names, addresses, and servers
  - Caching: of mappings from names to/from addresses
- DNS software components
  - DNS resolvers
  - DNS servers
- DNS queries
  - Iterative queries
  - Recursive queries
- DNS caching based on time-to-live (TTL)

Strawman Solution #1: Local File

- Original name to address mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: moving from a machine per domain to machine per user
  - Many more downloads
  - Many more updates

Strawman Solution #2: Central Server

- Central server
  - One place where all mappings are stored
  - All queries go to the central server
- Many practical problems
  - Single point of failure
  - High traffic volume
  - Distant centralized database
  - Single point of update
  - Does not scale

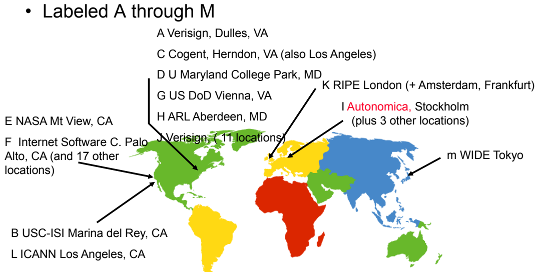
Need a distributed, hierarchical collection of servers

Domain Name System (DNS)

- Properties of DNS
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- Performing the translations
  - Local DNS servers
  - Resolver software

DNS Root Servers

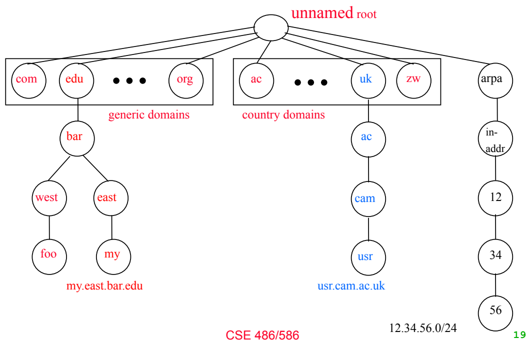
- 13 root servers (see http://www.root-servers.org/)
- Labeled A through M



TLD and Authoritative DNS Servers

- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, fr, ca, jp)
  - Typically managed professionally
    - » Network Solutions maintains servers for "com"
    - » Educause maintains servers for "edu"
- Authoritative DNS servers
  - Provide public records for hosts at an organization
  - For the organization's servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider

Distributed Hierarchical Database

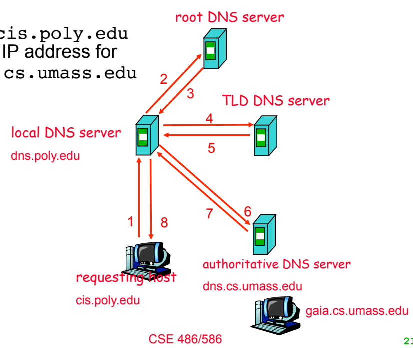


Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do *gethostbyname()* to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional *gethostbyaddr()* to translate into name

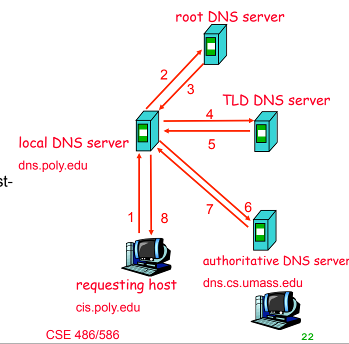
Example

Host at `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`



Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs



DNS Caching

- Performing all these queries take time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes the cached entry after TTL expires

Negative Caching

- Remember things that don't work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - … so the failure takes less time the next time around

DNS Resource Records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

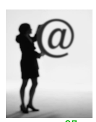
- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=CNAME
  - name is alias for some "canonical" (the real) name: www.ibm.com is really srveast.backup2.ibm.com
  - value is canonical name
- Type=MX
  - value is name of mailserver associated with name

Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don't care which server responds

Inserting Resource Records into DNS

- Example: just created startup "FooBar"
- Register foobar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - » (foobar.com, dns1.foobar.com, NS)
    - » (dns1.foobar.com, 212.212.212.1, A)
- Put in authoritative server dns1.foobar.com
  - Type A record for www.foobar.com
  - Type MX record for foobar.com
- Play with "dig" on UNIX




```
$ dig nytimes.com ANY
; QUESTION SECTION:
;nytimes.com.                 IN      ANY

;; ANSWER SECTION:
nytimes.com.          267     IN      MX      100
  NYTIMES.COM.S7A1.PSMTP.com.
nytimes.com.          267     IN      MX      200
  NYTIMES.COM.S7A2.PSMTP.com.
nytimes.com.          267     IN      A       199.239.137.200
nytimes.com.          267     IN      A       199.239.136.200
nytimes.com.          267     IN      TXT     "v=spf1 mx ptr
  ip4:199.239.138.0/24 include:alerts.wallst.com include:authsmtp.com
  ~all"
nytimes.com.          267     IN      SOA     ns1t.nytimes.com.
  root.ns1t.nytimes.com. 2009070102 1800 3600 604800 3600
nytimes.com.          267     IN      NS      nydns2.about.com.
nytimes.com.          267     IN      NS      ns1t.nytimes.com.
nytimes.com.          267     IN      NS      nydns1.about.com.

;; AUTHORITY SECTION:
nytimes.com.          267     IN      NS      nydns1.about.com.
nytimes.com.          267     IN      NS      ns1t.nytimes.com.
nytimes.com.          267     IN      NS      nydns2.about.com.

;; ADDITIONAL SECTION:
nydns1.about.com.     86207   IN      A       207.241.145.24
```



```
$ dig nytimes.com +norec @a.root-servers.net

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53675
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14

;; QUESTION SECTION:
;nytimes.com.                 IN      A

;; AUTHORITY SECTION:
com.                  172800  IN      NS      K.GTLD-SERVERS.NET.
com.                  172800  IN      NS      E.GTLD-SERVERS.NET.
com.                  172800  IN      NS      D.GTLD-SERVERS.NET.
com.                  172800  IN      NS      I.GTLD-SERVERS.NET.
com.                  172800  IN      NS      C.GTLD-SERVERS.NET.

;; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET.   172800  IN      A       192.5.6.30
A.GTLD-SERVERS.NET.   172800  IN      AAAA    2001:503:a83e::2:30
B.GTLD-SERVERS.NET.   172800  IN      A       192.33.14.30
B.GTLD-SERVERS.NET.   172800  IN      AAAA    2001:503:231d::2:30
```



```
$ dig nytimes.com +norec @k.gtld-servers.net

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38385
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;nytimes.com.                 IN      A

;; AUTHORITY SECTION:
nytimes.com.          172800  IN      NS      ns1t.nytimes.com.
nytimes.com.          172800  IN      NS      nydns1.about.com.
nytimes.com.          172800  IN      NS      nydns2.about.com.

;; ADDITIONAL SECTION:
ns1t.nytimes.com.  172800  IN  A  199.239.137.15
nydns1.about.com.  172800  IN  A  207.241.145.24
nydns2.about.com.  172800  IN  A  207.241.145.25

;; Query time: 103 msec
;; SERVER: 192.52.178.30#53(192.52.178.30)
```

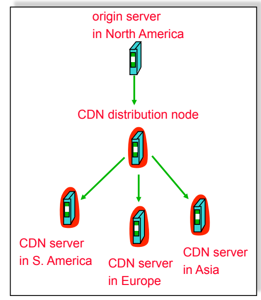
```
$ dig nytimes.com ANY +norec @ns1t.nytimes.com

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39107
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;nytimes.com.                IN      ANY

;; ANSWER SECTION:
nytimes.com.   300 IN   SOA   ns1t.nytimes.com.
      root.ns1t.nytimes.com. 2009070102 1800 3600 604800 3600
nytimes.com.   300 IN   MX    200 NYTIMES.COM.S7A2.PSMTP.com.
nytimes.com.   300 IN   MX    100 NYTIMES.COM.S7A1.PSMTP.com.
nytimes.com.   300 IN   NS    ns1t.nytimes.com.
nytimes.com.   300 IN   NS    nydns1.about.com.
nytimes.com.   300 IN   NS    nydns2.about.com.
nytimes.com.   300 IN   A     199.239.137.245
nytimes.com.   300 IN   A     199.239.136.200
nytimes.com.   300 IN   A     199.239.136.245
nytimes.com.   300 IN   TXT   "v=spf1 mx ptr ip4:199.239.138.0/24
```


Content Distribution Networks (CDNs)

- Content providers are CDN customers

Content replication

- CDN company installs thousands of servers throughout Internet
  - In large datacenters
  - Or, close to users
- CDN replicates customers' content
- When provider updates content, CDN updates servers



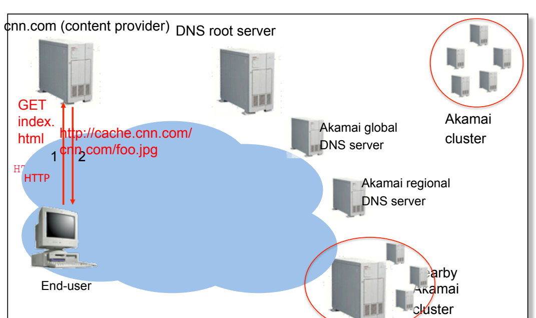
Content Distribution Networks

- Replicate content on many servers
- Challenges
  - How to replicate content
  - Where to replicate content
  - How to find replicated content
  - How to choose among replicas
  - How to direct clients towards a replica

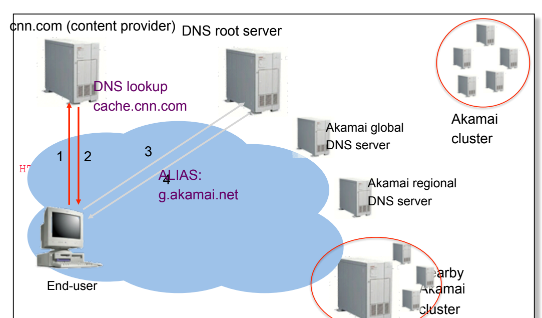
Server Selection

- Which server?
  - Lowest load: to balance load on servers
  - Best performance: to improve client performance
    - » Based on what? Location? RTT? Throughput? Load?
  - Any alive node: to provide fault tolerance
- How to direct clients to a particular server?
  - As part of routing: anycast, cluster load balancer
  - As part of application: HTTP redirect
  - As part of naming: DNS

How Akamai Works

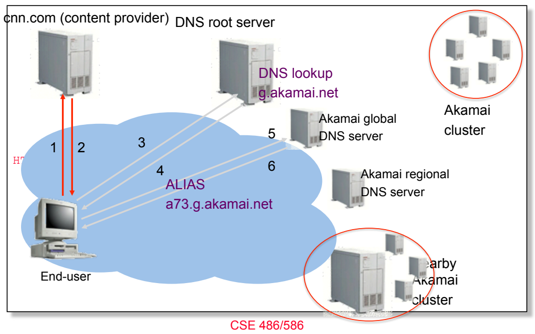


How Akamai Works



37

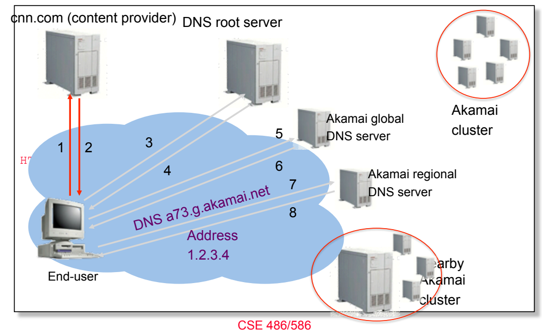
How Akamai Works



CSE 486/586

38

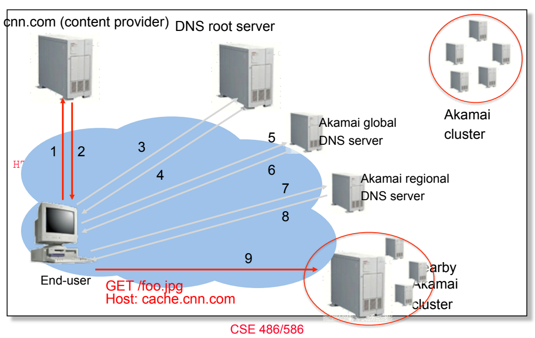
How Akamai Works



CSE 486/586

39

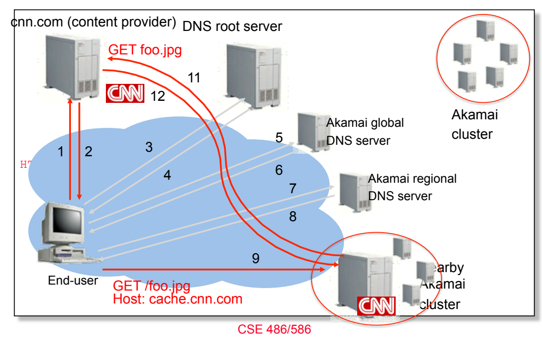
How Akamai Works



CSE 486/586

40

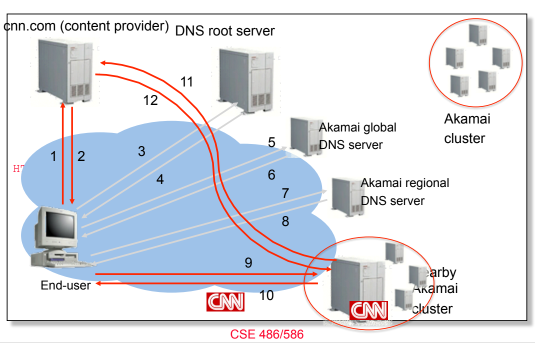
How Akamai Works



CSE 486/586

41

How Akamai Works



CSE 486/586

Summary

- DNS as an example client-server architecture
- Why?
  - Names are easier (for us!) to remember
  - IP addresses can change underneath
  - Name could map to multiple IP addresses
  - Map to different addresses in different places
  - Multiple names for the same address
- Properties of DNS
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers, top-level domain (TLD) servers, authoritative DNS servers

CSE 486/586 42

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC), Michael Freedman (Princeton), and Jennifer Rexford (Princeton).