Content Distribution Networks

COS 418: *Distributed Systems*
Lecture 19

Kyle Jamieson

[Selected content adapted from M. Freedman, B. Maggs and S. Shenker]

Today

1. **Domain Name System (DNS) primer**
   - A word on DNS security
2. The Web: HTTP, hosting, and caching
3. Content distribution networks (CDNs)

DNS hostname versus IP address

- **DNS host name** (*e.g.* www.cs.princeton.edu)
  - **Mnemonic** name appreciated by humans
  - **Variable length**, full alphabet of characters
  - Provides **little** (if any) information about **location**
- **IP address** (*e.g.* 128.112.136.35)
  - Numerical address appreciated by **routers**
  - **Fixed length**, decimal number
  - **Hierarchical** address space, related to host **location**

Many uses of DNS

- Hostname to IP address translation
  - IP address to hostname translation (***reverse lookup***)
- Host name ***aliasing***: other DNS names for a host
  - ***Alias*** host names point to ***canonical*** hostname
- **Email:** Lookup domain's mail server by domain name

Original design of the DNS

- Per-host file named /etc/hosts
  - Flat namespace: each **line = IP address & DNS name**
  - SRI (Menlo Park, California) kept the master copy
  - Everyone else downloads regularly

- **But, a single server doesn't scale**
  - Traffic implosion (lookups and updates)
  - Single point of failure

- Need a distributed, hierarchical **collection** of servers

DNS: Goals and non-goals

- A wide-area **distributed database**

- Goals:
  - **Scalability**; decentralized maintenance
  - **Robustness**
  - Global scope
    - Names mean the same thing everywhere
  - Distributed updates/queries
  - Good **performance**

- But **don't need** strong **consistency** properties

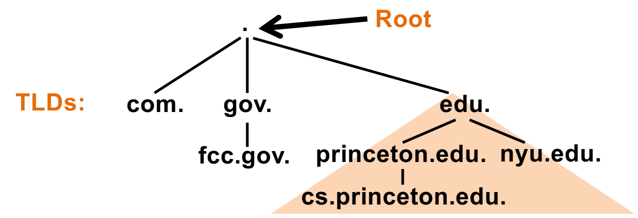
Domain Name System (DNS)

- **Hierarchical name space** divided into contiguous sections called ***zones***
  - Zones are distributed over a collection of DNS servers

- **Hierarchy** of DNS **servers**:
  - ***Root*** servers (identity hardwired into other servers)
  - ***Top-level domain (TLD)*** servers
  - ***Authoritative*** DNS servers

- Performing the translations:
  - ***Local DNS servers*** located near clients
  - ***Resolver*** software running on clients

The DNS namespace is hierarchical



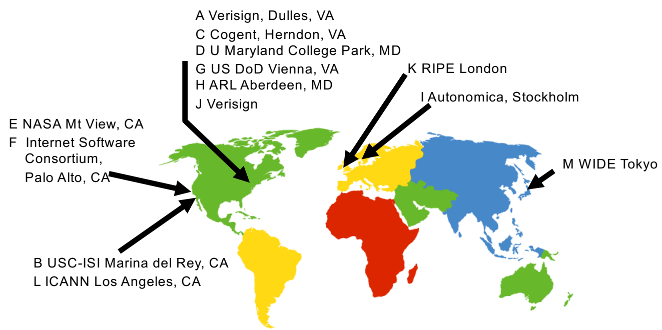
- **Hierarchy of namespace matches hierarchy of servers**

- Set of nameservers answers queries for names within zone

- Nameservers store names and links to other servers in tree

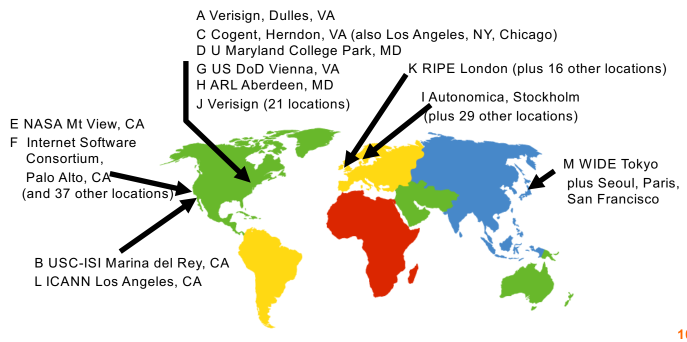
DNS root nameservers

- 13 root servers. *Does this scale?*



DNS root nameservers

- 13 root servers. *Does this scale?*
- Each server is really a **cluster** of servers (some geographically distributed), replicated via **IP anycast**



TLD and Authoritative Servers

- ***Top-level domain (TLD)*** servers
  - Responsible for com, org, net, edu, etc, and all top-level country domains: uk, fr, ca, jp
  - *Network Solutions* maintains servers for com TLD
  - *Educause* non-profit for edu TLD
- ***Authoritative*** DNS servers
  - An organization's DNS servers, providing authoritative information for that organization
  - May be maintained by organization itself, or ISP

Local name servers

- Do not strictly belong to hierarchy
- Each ISP (or company, or university) has one
  - Also called ***default*** or ***caching*** name server
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
  - Does work for the client

DNS resource records

- DNS is a distributed database storing **resource records**
- Resource record includes: (**name**, type, **value**, time-to-live)

Type = **A** (address)
- **name** = hostname
- **value** is IP address

Type = **CNAME**
- **name** = alias for some "canonical" (real) name
- **value** is canonical name

Type = **NS** (name server)
- **name** = domain (e.g. princeton.edu)
- **value** is hostname of authoritative name server for this domain

Type = **MX** (mail exchange)
- **name** = domain
- **value** is name of mail server for that domain

DNS in operation

- Most queries and responses are UDP datagrams
  – Two types of queries:

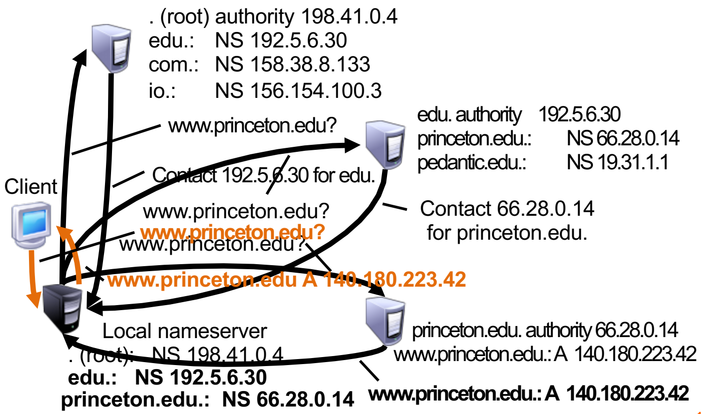
- ***Recursive***: Nameserver responds with answer or error

Answer: www.princeton.edu A 140.180.223.42

- ***Iterative***: Nameserver may respond with a referral

Referral: .edu NS a.edu-servers.net.

A recursive DNS lookup



Recursive versus iterative queries

Recursive query
- Less burden on entity initiating the query
- **More burden on nameserver** (has to return an answer to the query)
- Most root and TLD servers won't answer (shed load)
  – Local name server answers recursive query

Iterative query
- **More burden on query initiator**
- Less burden on nameserver (simply refers the query to another server)

(authoritative for org.)

```
$ dig @a.root-servers.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57494
;; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.freebsd.org.      IN A

;; AUTHORITY SECTION:
org.          172800 IN  NS  b0.org.afilias-nst.org.
org.          172800 IN  NS  d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
b0.org.afilias-nst.org.   172800 IN  A  199.19.54.1
d0.org.afilias-nst.org.   172800 IN  A  199.19.57.1
```

Glue records

[Output edited for clarity]


```
$ dig @199.19.54.1 www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39912
;; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
;www.freebsd.org.      IN A

;; AUTHORITY SECTION:
freebsd.org.    86400  IN NS  ns1.isc-sns.net.
freebsd.org.    86400  IN NS  ns2.isc-sns.com.
freebsd.org.    86400  IN NS  ns3.isc-sns.info.
```

[Output edited for clarity]

(authoritative for freebsd.org.)

```
$ dig @ns1.isc-sns.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17037
;; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.freebsd.org.      IN A

;; ANSWER SECTION:
www.freebsd.org.    3600   IN A   69.147.83.33

;; AUTHORITY SECTION:
freebsd.org.    3600   IN NS  ns2.isc-sns.com.
freebsd.org.    3600   IN NS  ns1.isc-sns.net.
freebsd.org.    3600   IN NS  ns3.isc-sns.info.

;; ADDITIONAL SECTION:
ns1.isc-sns.net.    3600   IN A   72.52.71.1
ns2.isc-sns.com.    3600   IN A   38.103.2.1
ns3.isc-sns.info.   3600   IN A   63.243.194.1
```

[Output edited for clarity]

DNS caching

- Performing all these queries takes time
  - And all this **before actual communication** takes place
- Caching can **greatly reduce overhead**
  - The top-level servers very rarely change
    - Popular sites visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - All DNS servers **cache responses to queries**
  - Responses include a time-to-live (TTL) field
    - Server deletes cached entry after TTL expires

Plays a key role in **CDN** (Akamai) **load balancing**

Today

1. Domain Name System (DNS) primer
   – **A word on DNS security**
2. The Web: HTTP, hosting, and caching
3. Content distribution networks (CDNs)

A word on DNS security

- Implications of subverting DNS:

1. Redirect victim's web traffic to rogue servers
2. Redirect victim's email to rogue email servers (MX records in DNS)

- Does Secure Sockets Layer (SSL) provide protection?
  – **Yes**—user will get "wrong certificate" if SSL enabled
  – **No**—SSL not enabled or user ignores warnings
  – **No**—how is SSL trust established? **Often, by email!**

Security Problem #1: Coffee shop

- As you sip your latte and surf the Web, how does your laptop find google.com?
- **Answer:** it asks the **local DNS nameserver**
  – Which is run by the coffee shop or their contractor
  – And can return to you any answer they please
- How can you know you're getting correct data?
  – Today, you can't. (Though HTTPS site helps.)
  – One day, hopefully: DNSSEC extensions to DNS

Security Problem #2: Cache poisoning

- You receive request to resolve **www.foobar.com & reply:**

```
;; QUESTION SECTION:
;www.foobar.com.          IN    A

;; ANSWER SECTION:
www.foobar.com.    300    IN    A    212.44.9.144

;; AUTHORITY SECTION:
foobar.com.        600    IN    NS   dns1.foobar.com.
foobar.com.        600    IN    NS   google.com.

;; ADDITIONAL SECTION:
google.com.          5    IN    A    212.44.9.155
```

Evidence disappears five sec. later!

A foobar.com machine, *not* google.com

DNS cache poisoning (cont'd)

- Okay, but how do you get the victim to **look up** www.foobar.com in the first place?
- Perhaps you connect to their mail server and send
  - HELO www.foobar.com
  - Which their mail server then looks up to see if it corresponds to your source address (anti-spam measure)
- Perhaps you send many **spam or phishing emails** containing a link to www.foobar.com

Mitigation: Bailiwick checking

- Local nameserver **ignores** any RR not **in or under same zone** as question
  - Widely deployed since *ca.* 1997
- But, **other attacks are possible** (*e.g.* Kaminsky poisoning)

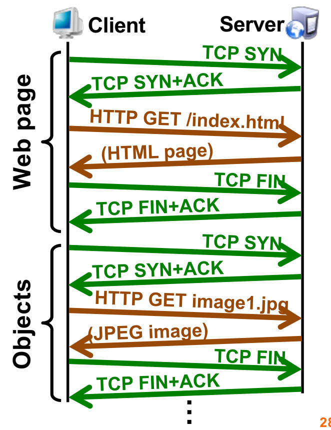
```
;; QUESTION SECTION:
;www.foobar.com.            IN    A

;; ANSWER SECTION:
www.foobar.com.    300      IN    A    212.44.9.144

;; AUTHORITY SECTION:
foobar.com.        600      IN    NS   dns1.foobar.com.
foobar.com.        600      IN    NS   google.com.

;; ADDITIONAL SECTION:
google.com.          5      IN    A       212.44.9.155
```


Today

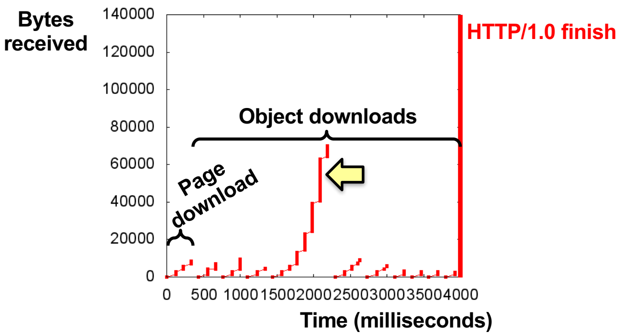
1. Domain Name System (DNS) primer
2. **The Web: HTTP, hosting, and caching**
3. Content distribution networks (CDNs)

Anatomy of an HTTP/1.0 web page fetch

- Web page = HTML file + embedded images/objects
- ***Stop-and-wait*** at the granularity of objects:
  - Close then open new TCP connection for **each object**
    - Incurs a **TCP round-trip-time delay** each time
  - Each TCP connection may stay in "slow start"



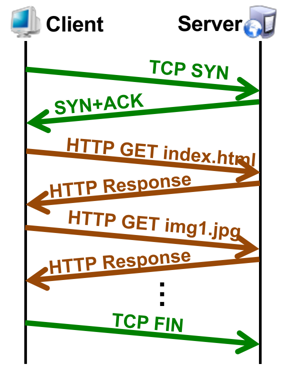
HTTP/1.0 webpage fetch: Timeline



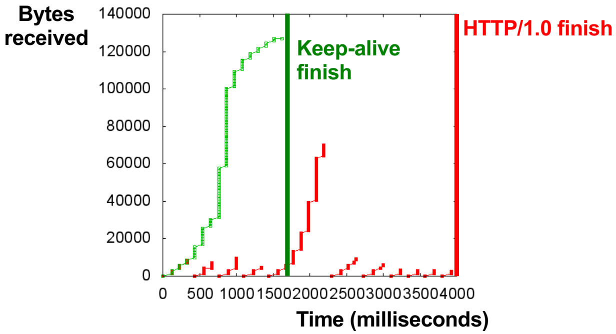
- Fetch 8.5 Kbyte page with 10 objects, most < 10 Kbyte

Letting the TCP connection persist

- Known as ***HTTP keepalive***
- **Still stop-and-wait** at the granularity of objects, at the application layer
  - HTTP response fully received before next HTTP GET dispatched
    - ≥ 1 RTT per object



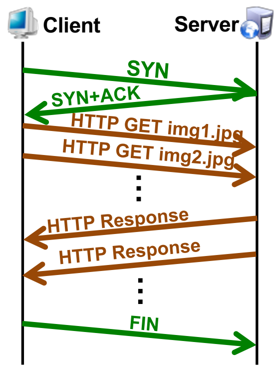
HTTP Keepalive avoids TCP slow starts



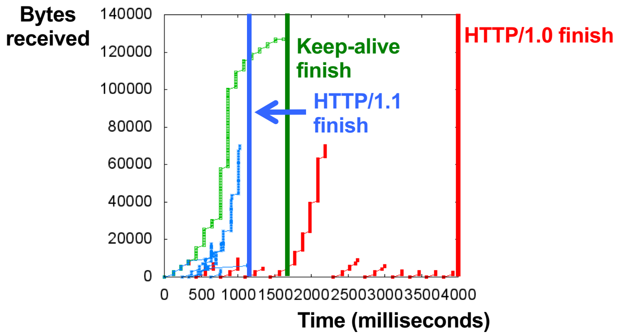
Incur **one slow start, but stop-and-wait** to issue next request

Pipelining within HTTP

- **Idea: Pipeline** HTTP GETs and their responses
- Main benefits:
  1. **Amortizes the RTT** across multiple objects retrieved
  2. **Reduces overhead** of HTTP requests, packing multiple requests into one packet
- Implemented in HTTP/1.1



Pipelined HTTP requests overlap RTTs



- Many **HTTP requests** and **TCP connections** at once
- **Overlaps RTTs of all requests**

Today

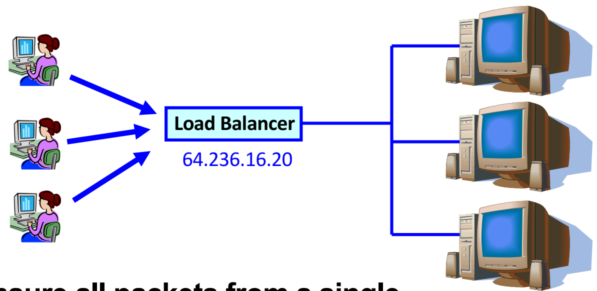
1. Domain Name System (DNS) primer
2. The Web: HTTP, **hosting, and caching**
   – **Handling heavy loads**
3. Content distribution networks (CDNs)

Hosting: Multiple machines per site

- **Problem:** **Overloaded** popular web site
  – **Replicate** the site across multiple machines
    - Helps to handle the load
- Want to direct client to a particular replica. Why?
  – **Balance load** across server replicas
- **Solution #1:** Manual selection by clients
  – Each replica has its own site name
  – Some Web page lists replicas (*e.g.*, by name, location), asks clients to click link to pick

Hosting: Load-balancer approach

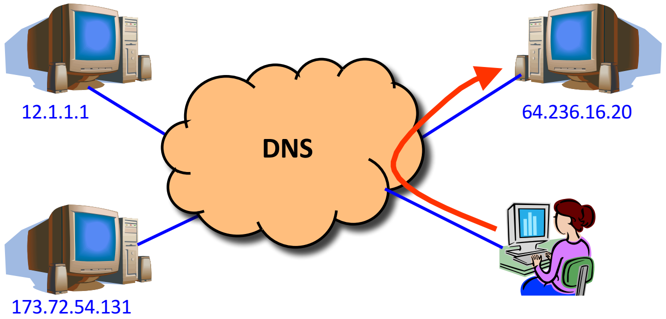
- **Solution #2:** Single IP address, multiple machines
  – Run multiple machines behind a single IP address



  – **Ensure all packets from a single TCP connection go to the same replica**

Hosting: DNS redirection approach

- **Solution #3:** Multiple IP addresses, multiple machines
  - Same DNS name but different IP for each replica
    - DNS server returns IP addresses “round robin”



37

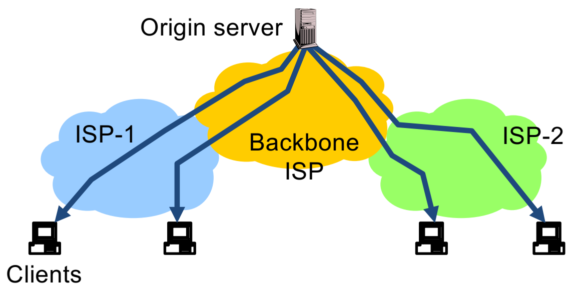
Hosting: Summary

- Load-balancer approach
  - No geographical diversity ✘
  - TCP connection issue ✘
  - Does not reduce network traffic ✘
- DNS redirection
  - No TCP connection issues ✔
  - Simple round-robin server selection
    - May be less responsive ✘
  - Does not reduce network traffic ✘

38

Web caching

- Many clients transfer the **same information**
  - Generates **redundant** server and network load
  - Also, clients may experience high **latency**



39

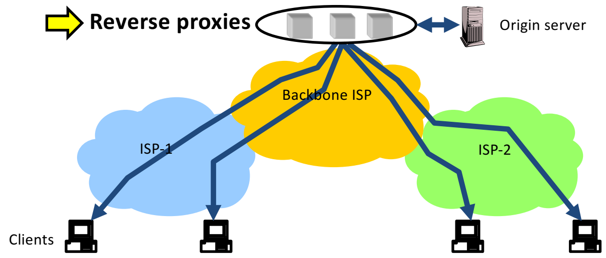
Why web caching?

- Motivation for **placing content closer to client**:
  - User gets **better response time**
    - Content providers get happier users
  - Network gets **reduced load**
- Why does caching work? Exploits locality of reference
- How well does caching work?
  - Very well, **up to a limit**
  - Large overlap in content
  - But many unique requests

40

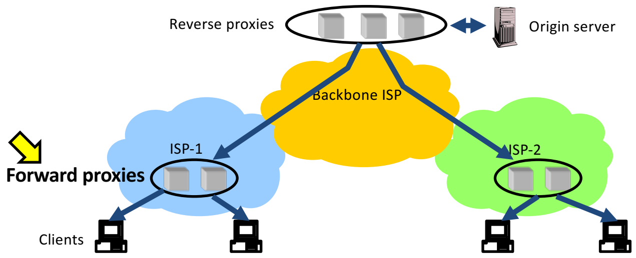
Caching with Reverse Proxies

- Cache data close to origin server → decrease server load
  - Typically done by content providers
  - Client thinks it is talking to the origin server (the server with content)
- Does not work for **dynamic content**



Caching with Forward Proxies

- Cache close to clients → less network traffic, less latency
  - Typically done by ISPs or corporate LANs
  - **Client configured** to send HTTP requests to forward proxy
- Reduces traffic on ISP-1's access link, origin server, and backbone ISP



Caching & Load-Balancing: Outstanding problems

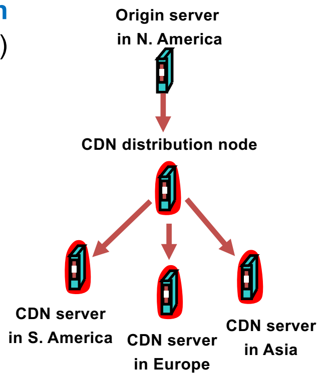
- Problem *ca.* 2002: *How to reliably deliver large amounts of content to users worldwide?*
  - Popular event: **"Flash crowds" overwhelm** (replicated) web server, access link, or back-end database infrastructure
  - More rich content: audio, video, photos
- Web caching: Diversity causes **low cache hit rates (25−40%)**

Today

1. Domain Name System (DNS) primer
2. The Web: HTTP, hosting, and caching
3. **Content distribution networks (CDNs)**
   - Akamai case study

Content Distribution Networks

- **Proactive content replication**
  - Content provider (*e.g.* CNN) pushes content out from its own ***origin server***
- CDN **replicates** the content
  - On many servers spread throughout the Internet
- Updating the replicas
  - Updates **pushed to replicas** when the content changes



Replica selection: Goals

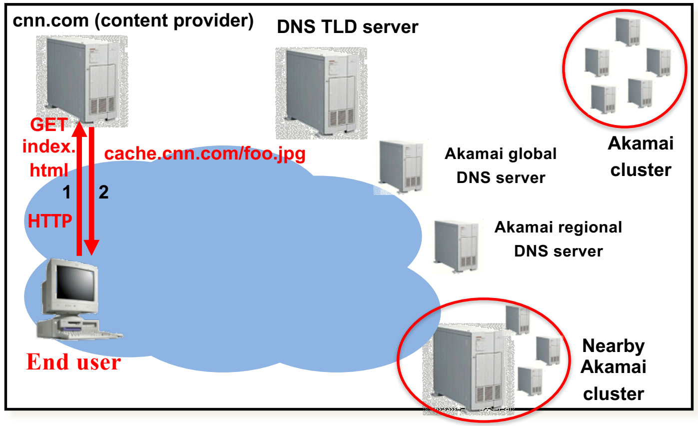
- **Live** server
  - For availability
- Lowest **load**
  - To balance load across the servers
- **Closest**
  - Nearest geographically, or in round-trip time
- Best **performance**
  - Throughput, latency, reliability…

Requires continuous monitoring of liveness, load, and performance

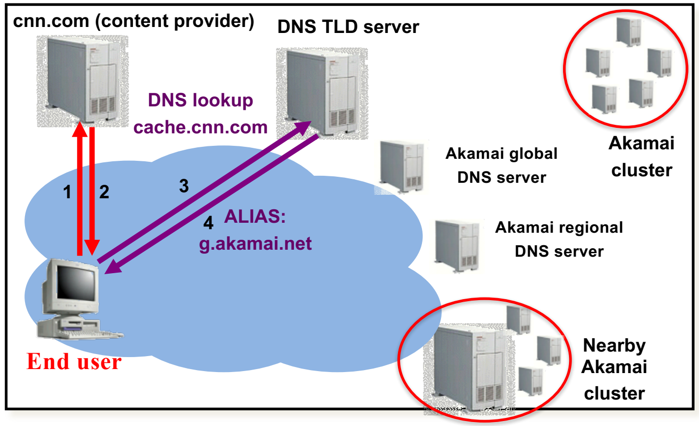
Akamai statistics

- Distributed servers
  - Servers: ~100,000
  - Networks: ~1,000
  - Countries: ~70
- Many customers
  - Apple, BBC, FOX, GM IBM, MTV, NASA, NBC, NFL, NPR, Puma, Red Bull, Rutgers, SAP, …
- Client requests
  - 20+M per second
  - Half in the top 45 networks
  - 20% of all Web traffic worldwide

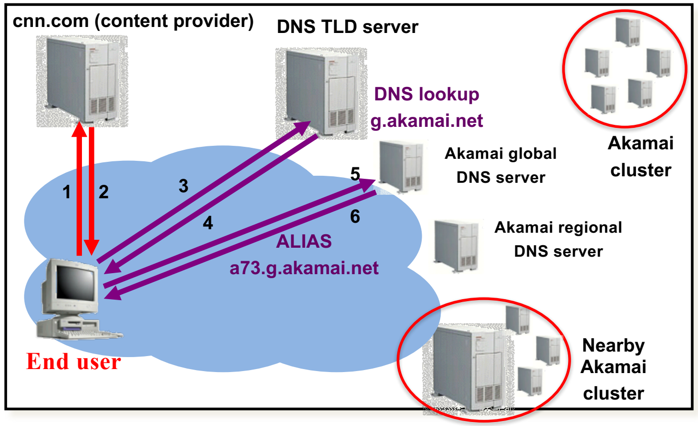
How Akamai Uses DNS



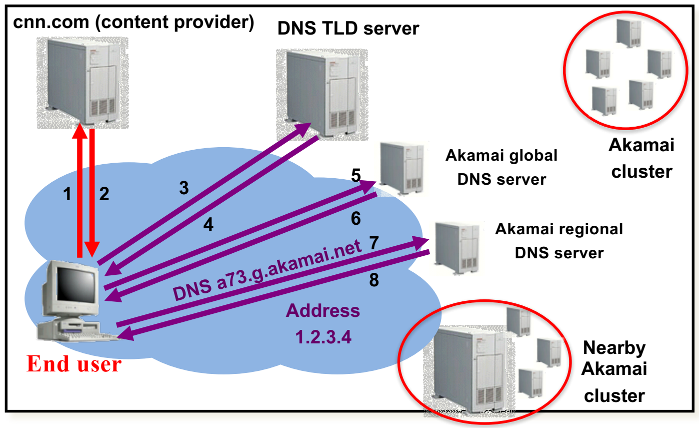
How Akamai Uses DNS



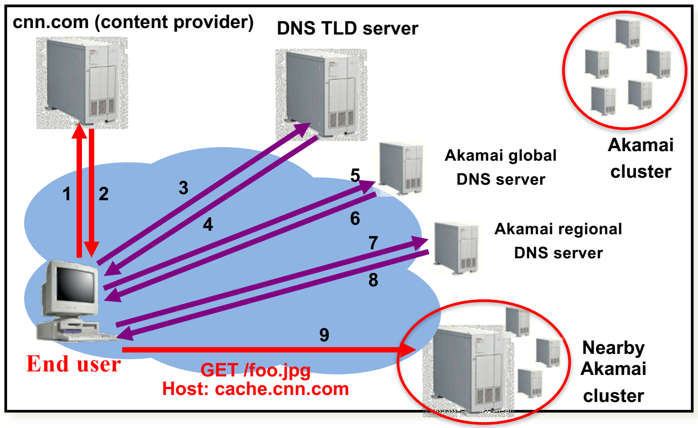
How Akamai Uses DNS



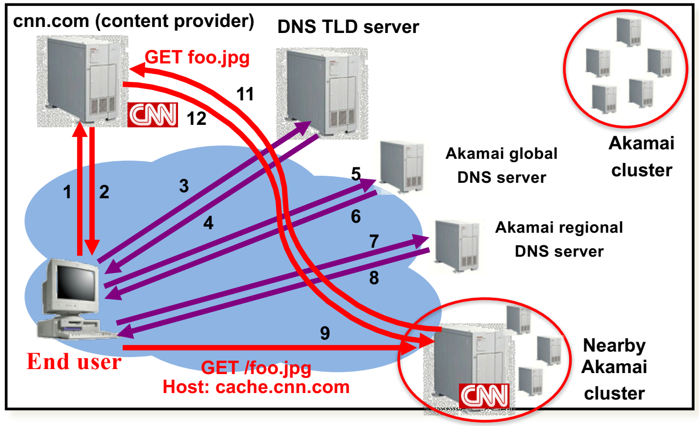
How Akamai Uses DNS



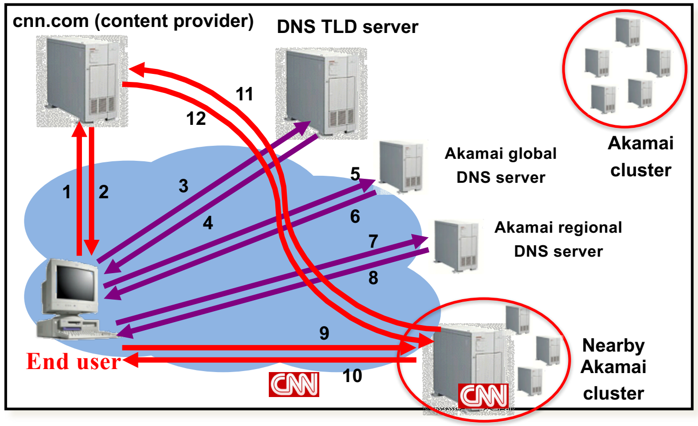
How Akamai Uses DNS



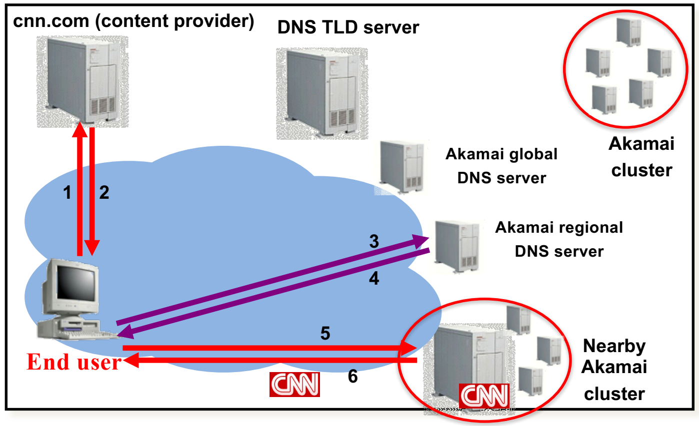
How Akamai Uses DNS



How Akamai Uses DNS



How Akamai Works: Cache Hit



Mapping System

- Equivalence classes of IP addresses
  - IP addresses experiencing similar performance
  - Quantify how well they connect to each other
- **Collect and combine** measurements
  - Ping, traceroute, BGP routes, server logs
    - *e.g.*, over 100 TB of logs per days
  - Network latency, loss, throughput, and connectivity

56

Routing client requests with the map

- Map each **IP class** to a preferred **server cluster**
  - Based on performance, cluster health, etc.
  - Updated roughly every minute
    - **Short, 60-sec DNS TTLs** in Akamai regional DNS accomplish this
- Map client request to a server in the cluster
  - **Load balancer** selects a specific server
  - *e.g.*, to **maximize** the **cache hit rate**

57

Adapting to failures

- Failing **hard drive** on a server
  - Suspends after finishing "in progress" requests
- Failed **server**
  - Another server takes over for the IP address
  - Low-level map updated **quickly** (load balancer)
- Failed **cluster**, or **network path**
  - High-level map updated **quickly** (ping/traceroute)

58

Take-away points: CDNs

- Content distribution is hard
  - Many, diverse, changing objects
  - Clients distributed all over the world
- **Moving content to the client** is key
  - Reduces latency, improves throughput, reliability
- Content distribution solutions **evolved:**
  - Load balancing, reactive caching, to
  - Proactive content distribution networks

59

Friday precept:
How to transition from
Assignment 3 to Assignment 4

Monday topic:
Distributed Wireless Networks: *Roofnet*

60