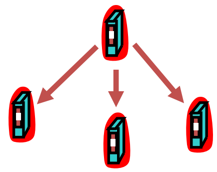


Content Distribution Networks (CDNs)

Mike Freedman

COS 461: Computer Networks

http://www.cs.princeton.edu/courses/archive/spr20/cos461/

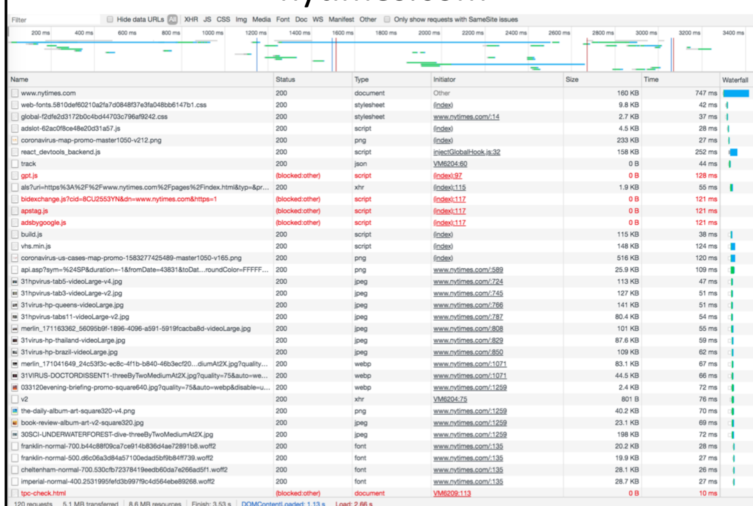
Continuation of Lec 15

HTTP xfer = single object

Web pages = many objects

---

nytimes.com



How to handle many requests?

- Maximize goodput by reusing connections
  - Avoid connection (TCP) setup
  - Avoid TCP slow-start
- Client-server will maintain existing TCP connection for up to K idle seconds

```
GET / HTTP/1.1
Host: www.example.com
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Tue, 27 Mar 2001 03:50:51 GMT
Connection: Keep-Alive
```


Three approaches to multiple requests

Parallel Connections	Persistent Connections	Pipelined Connections
Conn 1:	Conn 1:	Conn 1:
• Request 1	• Request 1	• Request 1
• Response 1	• Response 1	• Request 2
	• Request 2	• Request 3
Conn 2:	• Response 2	• Response 1
• Request 2	• Request 3	• Response 2
• Response 2	• Response 3	• Response 3

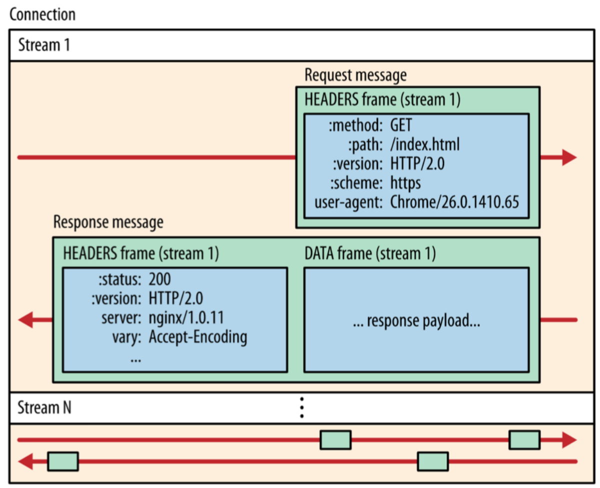
What are challenges with pipelining?

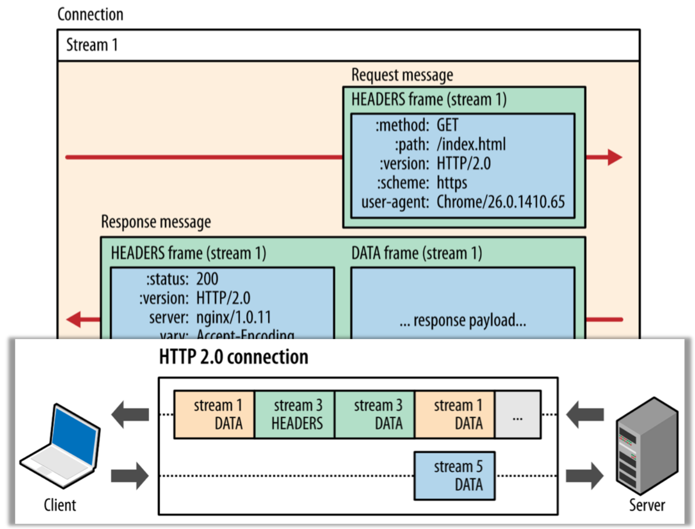
- Head-of-line blocking
  - Small xfers can "block" behind large xfer
- No reordering
  - HTTP response does not "identify" which request it's in response to; obvious in simple request/response
- Can behave *worse* than parallel + persistent
  - Can send expensive query 1 on conn 1, while sending many cheap queries on conn 2

Google's SPDY -> HTTP/2

- Server "push" for content
  - One client request, multiple responses
  - After all, server knows that after parsing HTML, client will immediately request embedded URLs
- Better pipelining and xfer
  - Multiplexing multiple xfers w/o HOL blocking
  - Request prioritization
  - Header compression

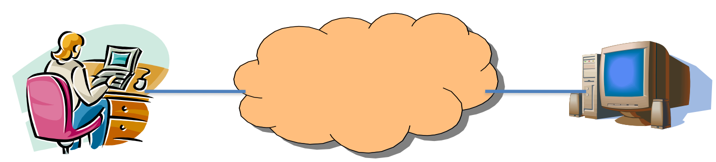
https://developers.google.com/web/fundamentals/performance/http2





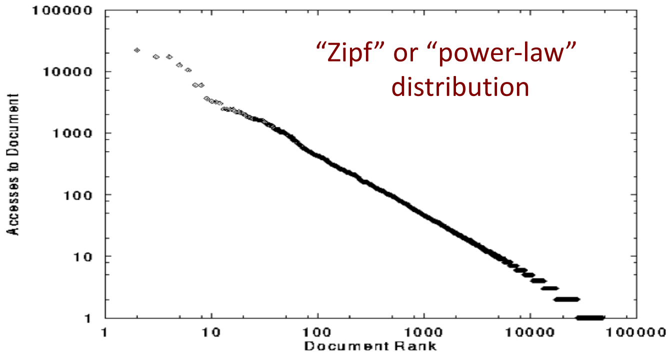
Why Web Caching?

Single Server, Poor Performance



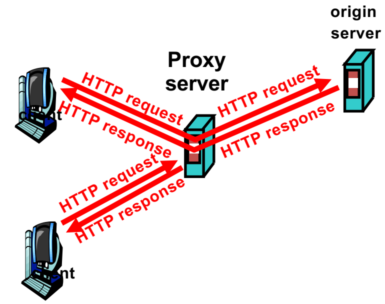
- Single server
  - Single point of failure
  - Easily overloaded
  - Far from most clients
- Popular content
  - Popular site
  - “Flash crowd” (aka “Slashdot effect”)
  - Denial of Service attack

Skewed Popularity of Web Traffic



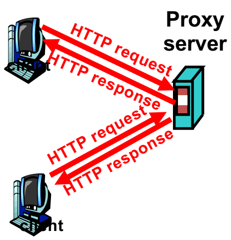
Characteristics of WWW Client-based Traces
Carlos R. Cunha, Azer Bestavros, Mark E. Crovella, BU-CS-95-01

Proxy Caches



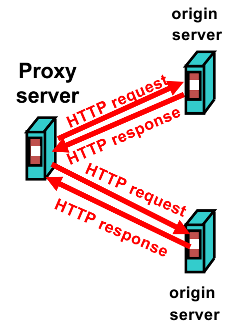
Forward Proxy

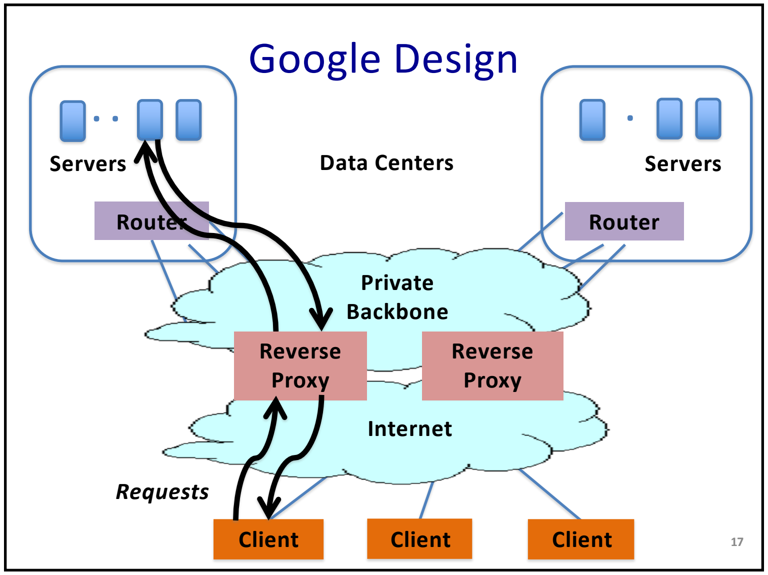
- Cache "close" to the client
  - Under administrative control of client-side AS
- Explicit proxy
  - Requires configuring browser
- Implicit proxy
  - Service provider deploys an "on path" proxy
  - … that intercepts and handles Web requests



Reverse Proxy

- Cache "close" to server
  - Either by proxy run by server or in third-party content distribution network (CDN)
- Directing clients to the proxy
  - Map the site name to the IP address of the proxy





Proxy Caches

(Y) Forward (M) Reverse (C) Both (A) Neither

- Reactively replicates popular content
- Reduces origin server costs
- Reduces client ISP costs
- Intelligent load balancing between origin servers
- Offload form submissions (POSTs) and user auth
- Content reassembly or transcoding on behalf of origin
- Smaller round-trip times to clients
- Maintain persistent connections to avoid TCP setup delay (handshake, slow start)

18

Proxy Caches

(Y) Forward (M) Reverse (C) Both (A) Neither

- Reactively replicates popular content **C**
- Reduces origin server costs **C**
- Reduces client ISP costs **Y**
- Intelligent load balancing between origin servers **M**
- Offload form submissions (POSTs) and user auth **A**
- Content reassembly or transcoding on behalf of origin **C**
- Smaller round-trip times to clients **C**
- Maintain persistent connections to avoid TCP setup delay (handshake, slow start) **C**

19

Modern HTTP Video-on-Demand

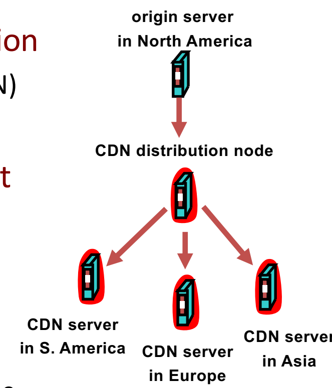
- Download “content manifest” from origin server
- List of video segments belonging to video
  - Each segment 1-2 seconds in length
  - Client can know time offset associated with each
  - Standard naming for different video resolutions and formats: e.g., 320dpi, 720dpi, 1040dpi, …
- Client downloads video segment (at certain resolution) using standard HTTP request.
  - HTTP request can be satisfied by cache: it’s a static object
- Client observes download time vs. segment duration, increases/decreases resolution if appropriate

20

Content Distribution Networks

Content Distribution Network

- Proactive content replication
  - Content provider (e.g., CNN) contracts with a CDN
- CDN replicates the content
  - On many servers spread throughout the Internet
- Updating the replicas
  - Reactive by TTL or updates pushed to replicas when the content changes



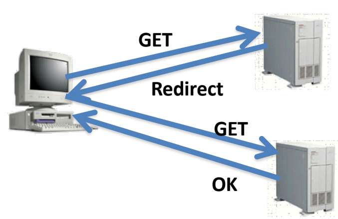
Server Selection Policy

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

Requires continuous monitoring of liveness, load, and performance

Server Selection Mechanism

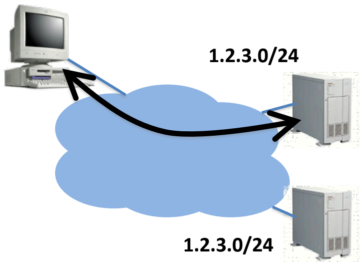
- Application
  - HTTP redirection



- Advantages
  - Fine-grain control
  - Selection based on client IP address
- Disadvantages
  - Extra round-trips for TCP connection to server
  - Overhead on the server

Server Selection Mechanism

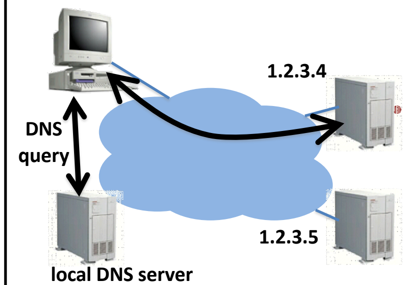
- Routing
  - Anycast routing



- Advantages
  - No extra round trips
  - Route to nearby server
- Disadvantages
  - Does not consider network or server load
  - Different packets may go to different servers
  - Used only for simple request-response apps

Server Selection Mechanism

- Naming
  - DNS-based server selection



- Advantages
  - Avoid TCP set-up delay
  - DNS caching reduces overhead
  - Relatively fine control
- Disadvantage
  - Based on IP address of local DNS server
  - “Hidden load” effect
  - DNS TTL limits adaptation

How Akamai Works

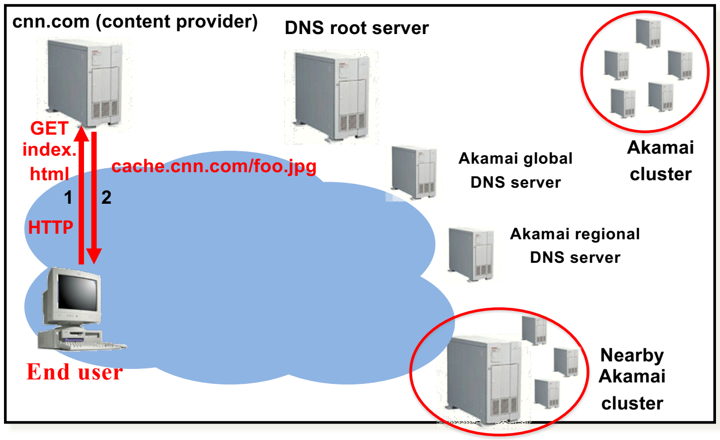
Akamai Statistics

- Distributed servers
  - Servers: ~275,000
  - Networks: 1,500
  - Countries: 136
- Many customers
  - 50% of Fortune Global 500
- Network
  - Up to 50 Tbps daily
  - 2019 Cricket World Cup: 25.3M concurrent viewers
  - 85% Internet is one network hop from Akamai servers

https://www.akamai.com/us/en/about/facts-figures.jsp

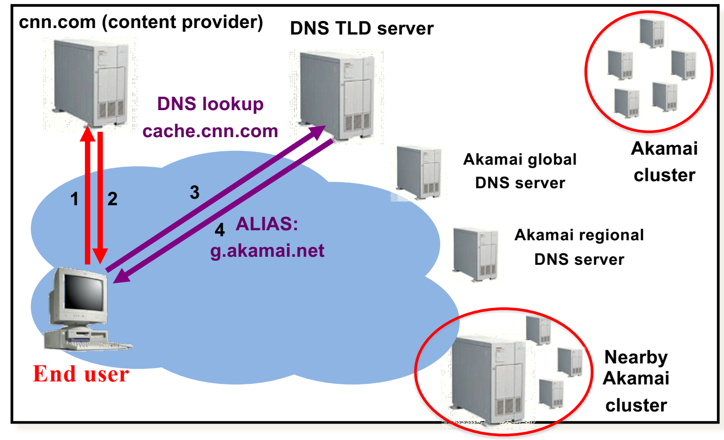
29

How Akamai Uses DNS



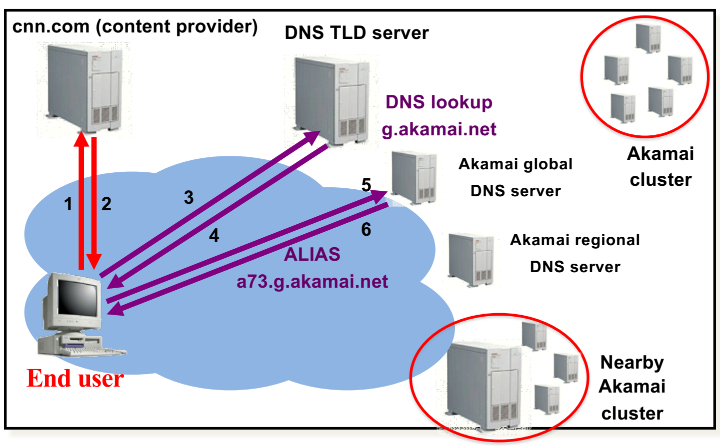
30

How Akamai Uses DNS



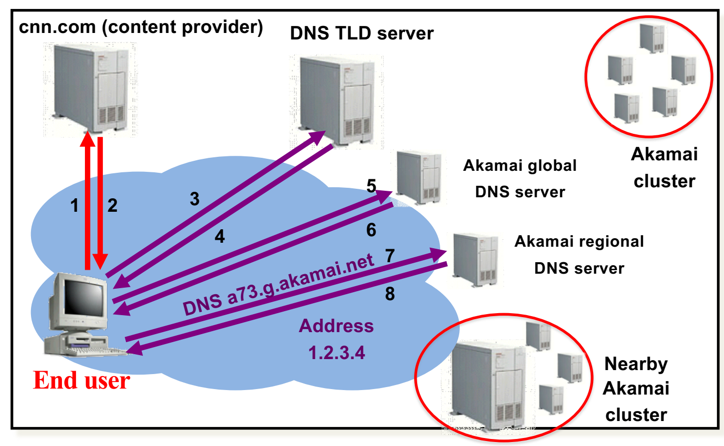
31

How Akamai Uses DNS



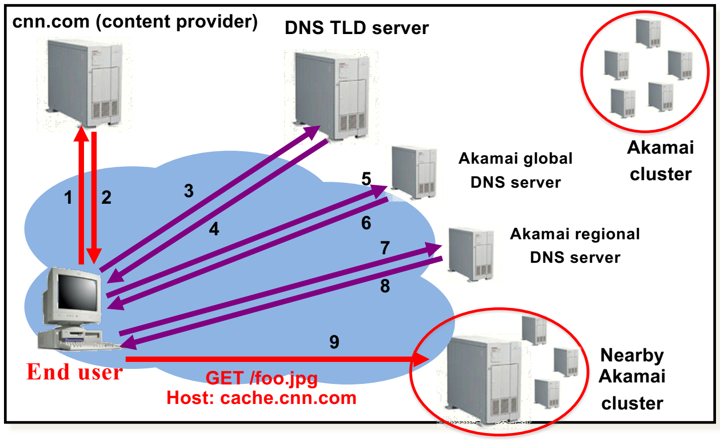
32

How Akamai Uses DNS



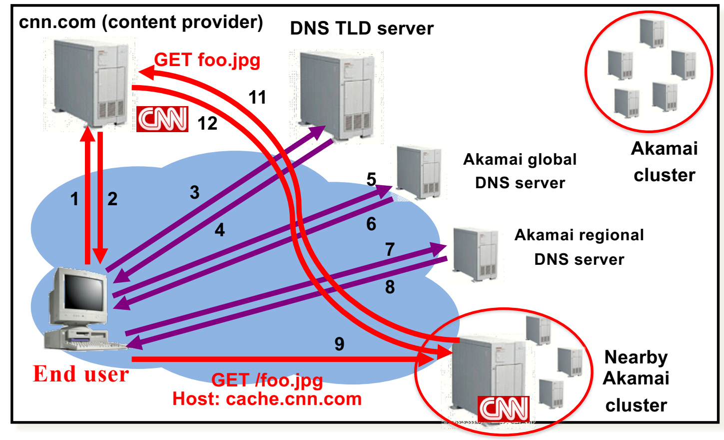
33

How Akamai Uses DNS



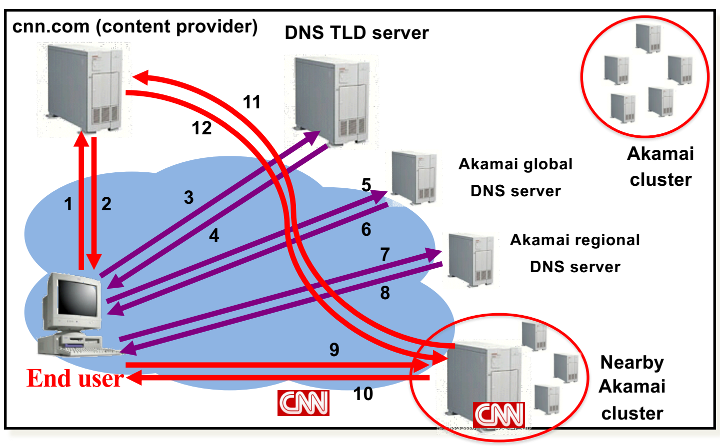
34

How Akamai Uses DNS



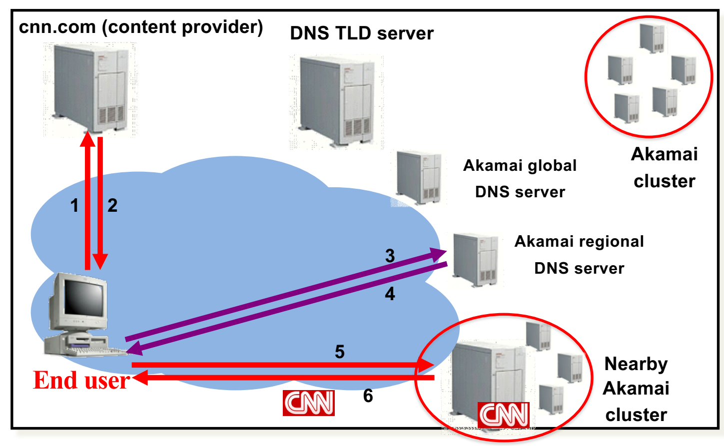
35

How Akamai Uses DNS



36

How Akamai Works: Cache Hit



Mapping System

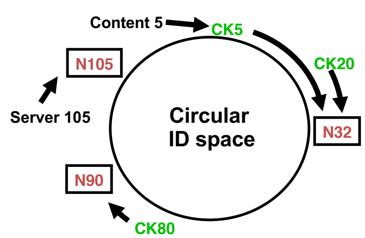
- Equivalence classes of IP addresses
  - IP addresses experiencing similar performance
  - Quantify how well they connect to each other
- Collect and combine measurements
  - Ping, traceroute, BGP routes, server logs
    - E.g., over 100 TB of logs per days
  - Network latency, loss, and connectivity

Routing Client Requests within 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

Selecting server inside cluster

- "Consistent hashing"
  - content_key = hash(URL) mod N
  - node_key = hash(server ID) mod N
  - Content belongs to server's node_key is "closest" to URL's content_key



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
- Failed cluster or network path
  - High-level map updated quickly
- Failed path to customer's origin server
  - Route packets through an intermediate node

Akamai Transport Optimizations

- Bad Internet routes
  - Overlay routing through an intermediate server
- Packet loss
  - Sending redundant data over multiple paths
- TCP connection set-up/teardown
  - Pools of persistent connections
- TCP congestion window and round-trip time
  - Estimates based on network latency measurements

Akamai Application Optimizations

- Slow download of embedded objects
  - Prefetch when HTML page is requested
- Large objects
  - Content compression
- Slow applications
  - Moving applications to edge servers
  - E.g., content aggregation and transformation
  - E.g., static databases (e.g., product catalogs)

Conclusion

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