CONTENT-DISTRIBUTION NETWORKS

Module 5
Fall 2020

George Porter

UC San Diego

ATTRIBUTION

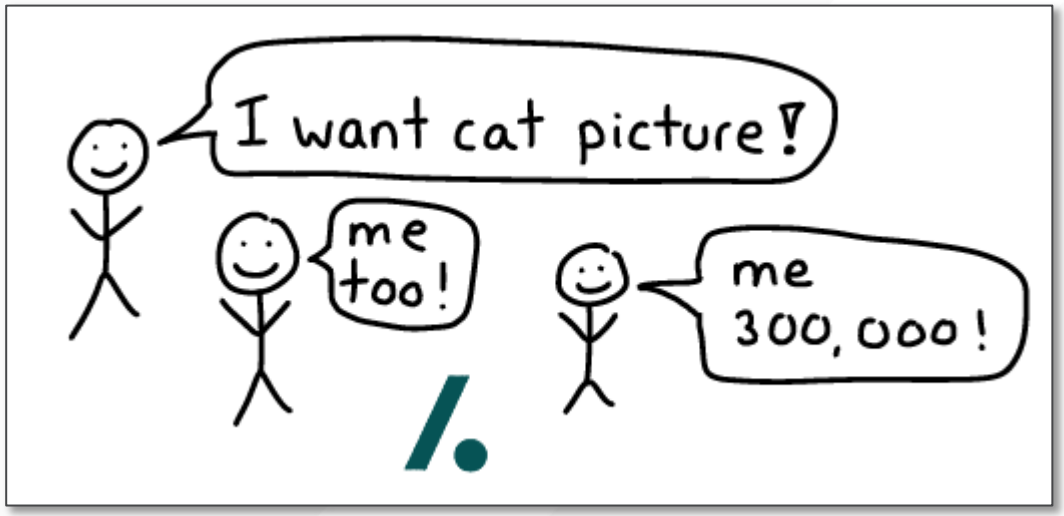
- These slides incorporate material from:
  - Kyle Jamieson, Princeton University (also under a CC BY-NC-SA 3.0 Creative Commons license)
  - David Choffnes, Northeastern University

UC San Diego

OUTLINE

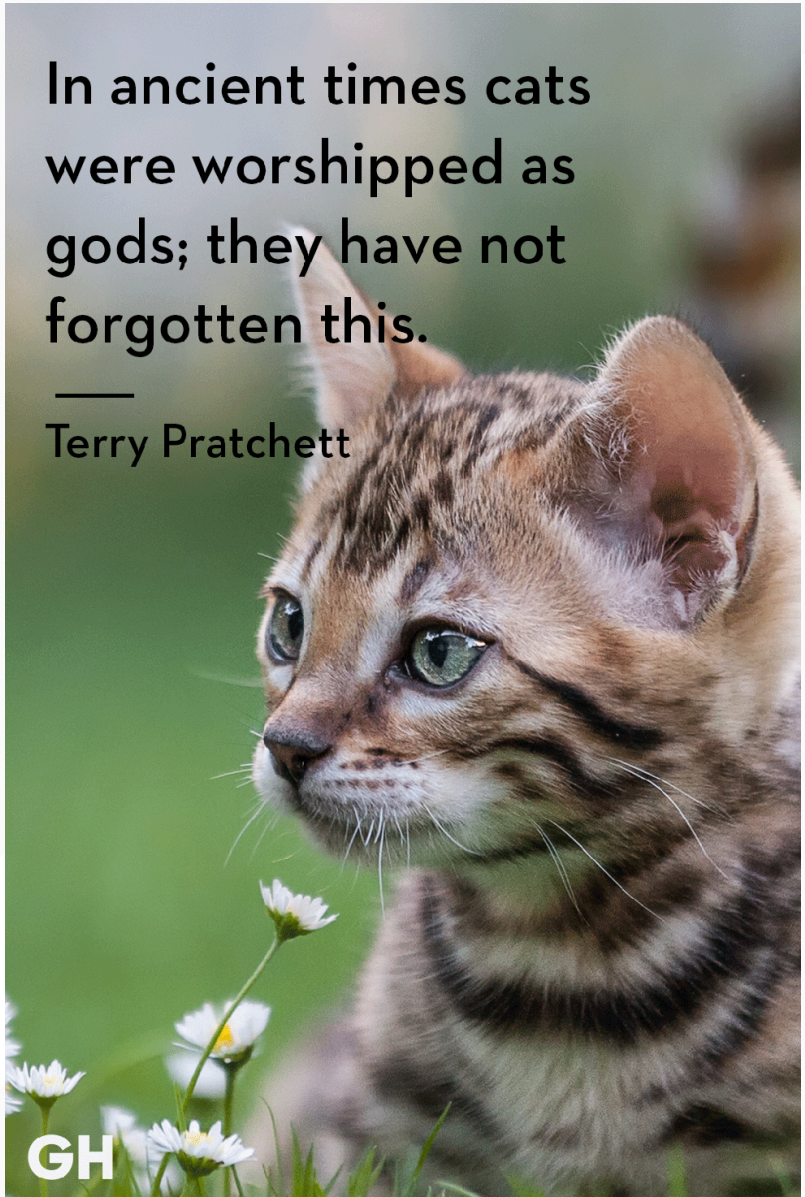
1. Web caching
2. Content-distribution networks
   - Featuring Akamai

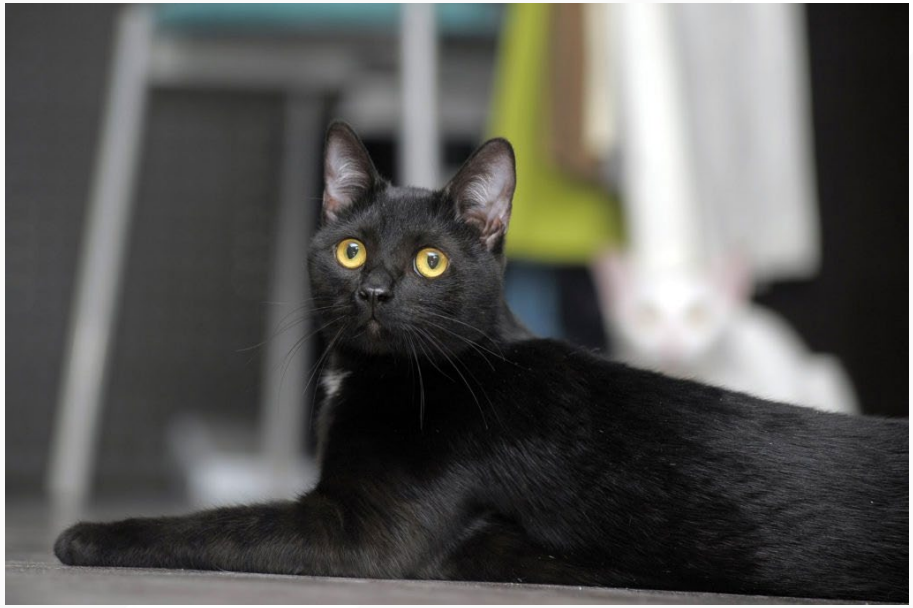
INTRO TO WEB REPLICATION



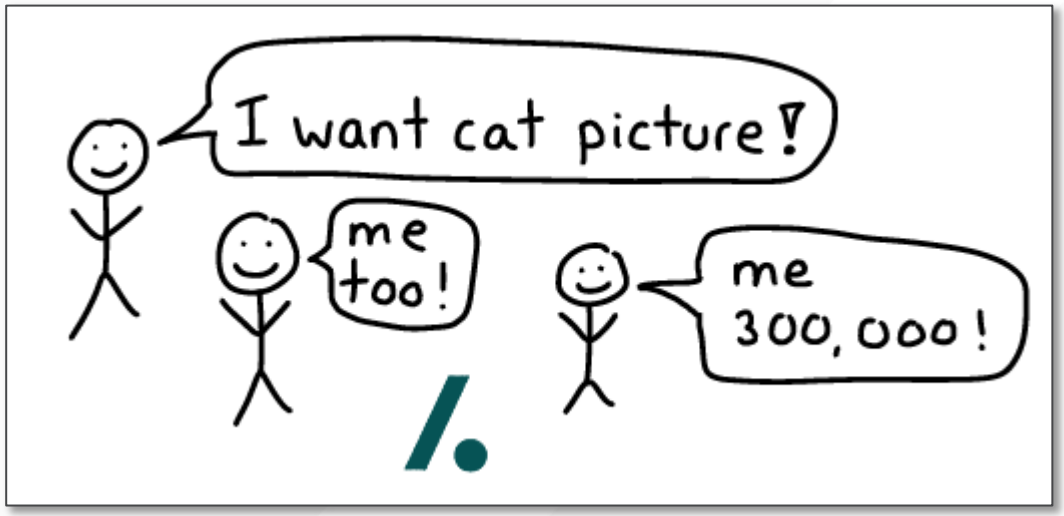
Images by Julia Evans

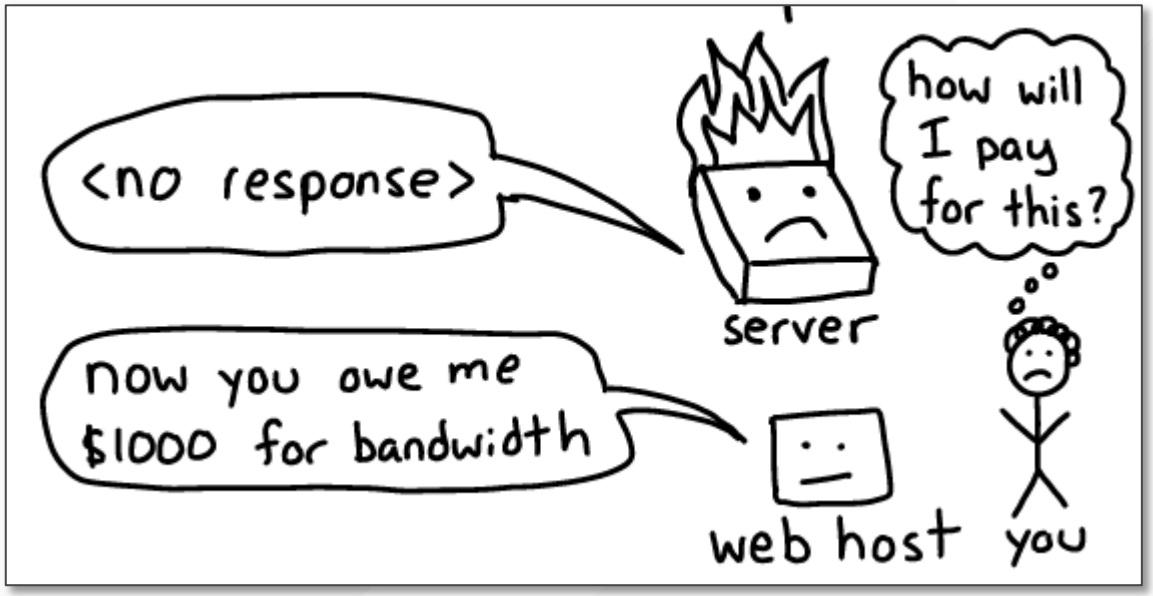
EXAMPLES OF CAT PICTURES FROM THE INTERNET





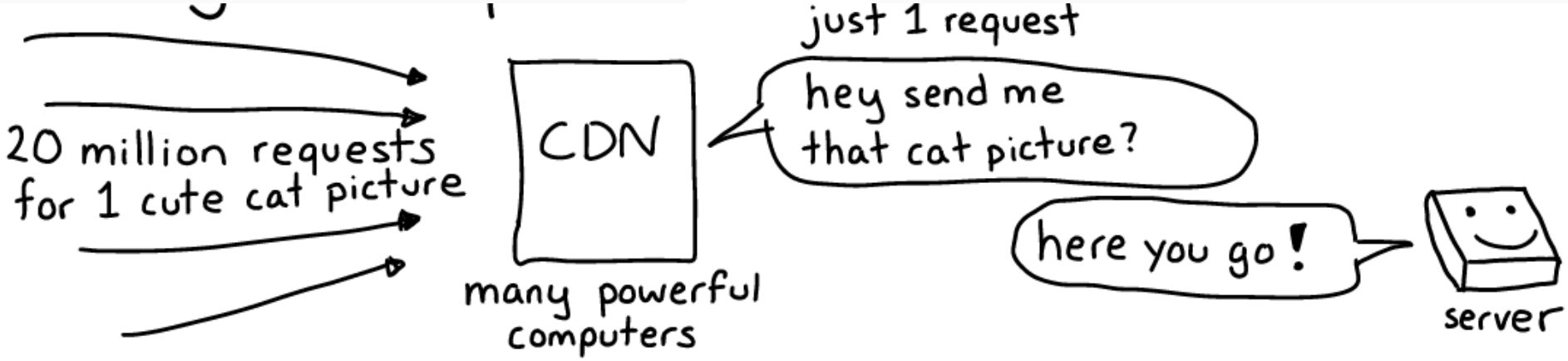
INTRO TO WEB REPLICATION





Images by Julia Evans

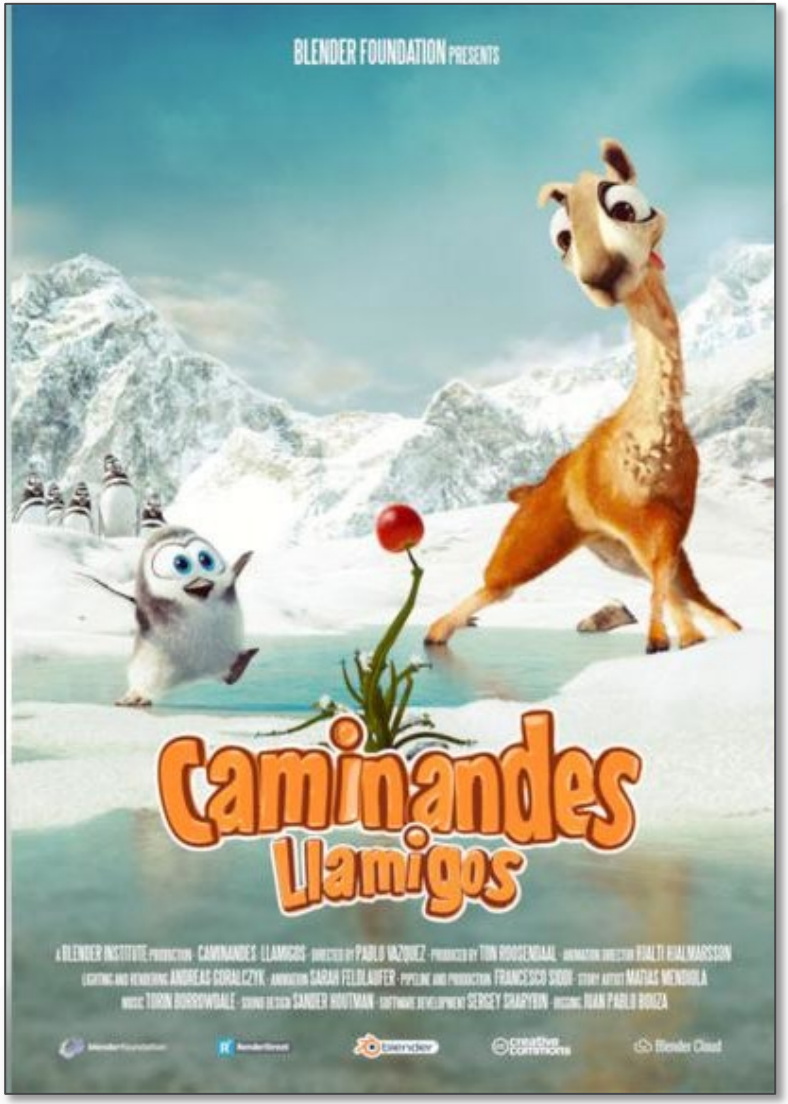
4 COMPONENTS TO CONTENT DISTRIBUTION NETWORKS



1. **Proxies**: How to get web content from a server different than the original one?
2. **Caching**: OK, but then what if the original server updates/changes the content?
3. **Load balancing**: How do I choose which proxy/cache?
4. **Availability**: What if some of the proxies or caches fail?

Images by Julia Evans

STREAMING VIDEO… TO MILLIONS OF USERS?



- Scenario:
  - Use MPEG-DASH to stream video files via HTTP to millions of users
  - How does that *scale*?

WEB CACHING

- Many clients transf
  - Generates **redunda**
  - Also, clients may ex

Origin s

ISP-1

SP-2

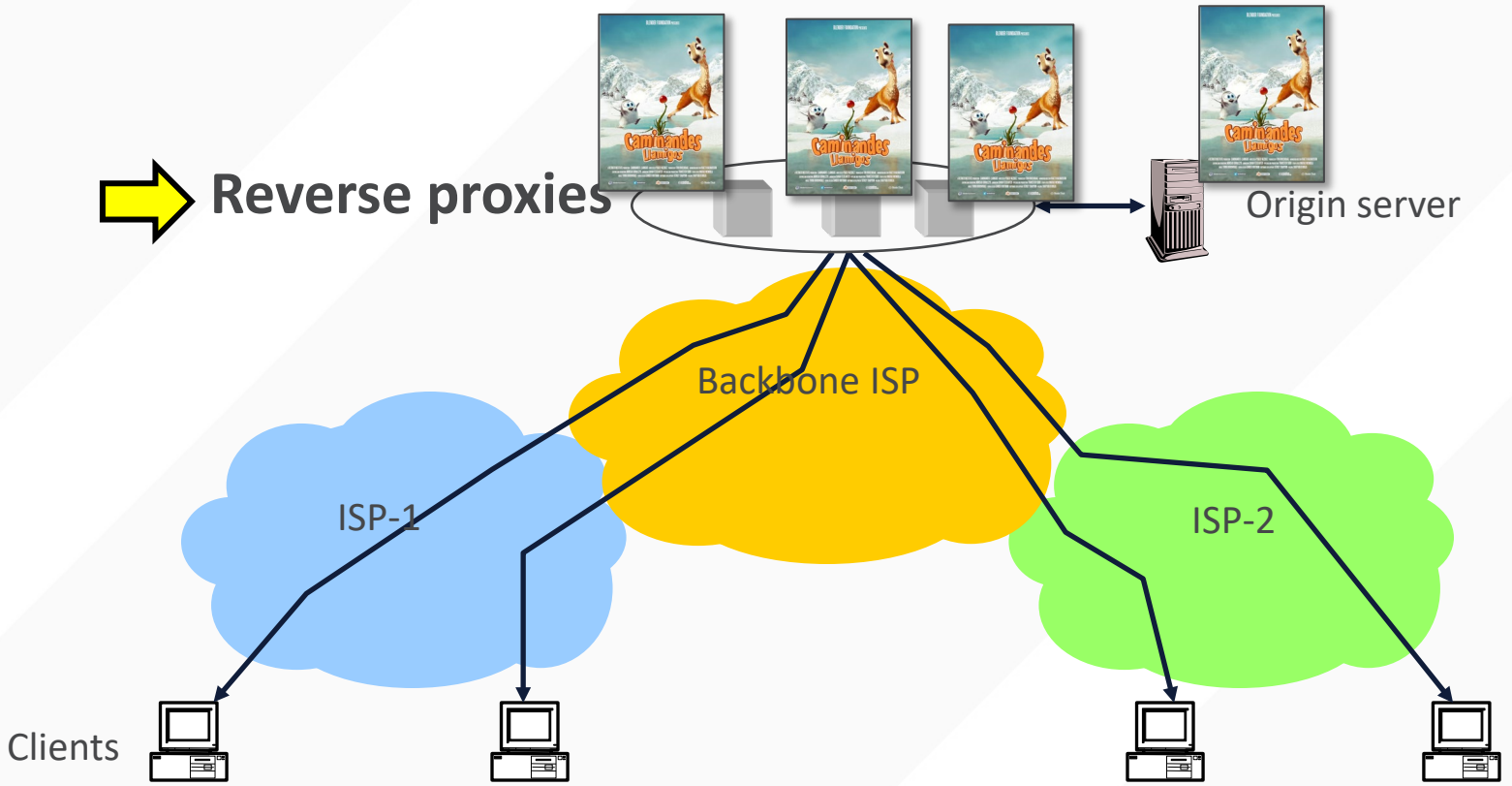
Clients

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

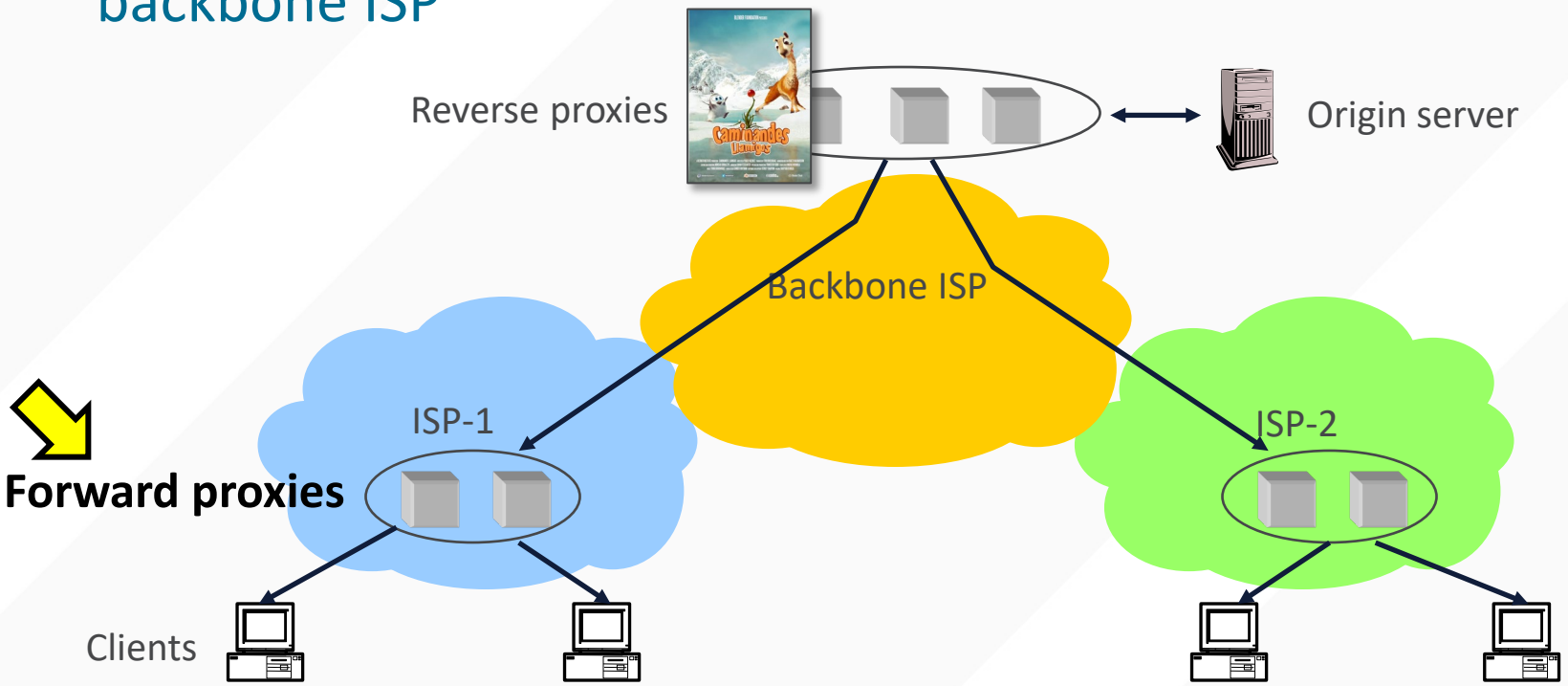
CACHING WITH REVERSE PROXIES

- Cache data close to origin server → decrease server load
  - 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%)**

GETTING CURL TO USE A WEB PROXY

- curl -v -x webproxy.ucsd.edu:3128 -o /dev/null https://cseweb.ucsd.edu/~gmporter/index.html

PROXY CACHES

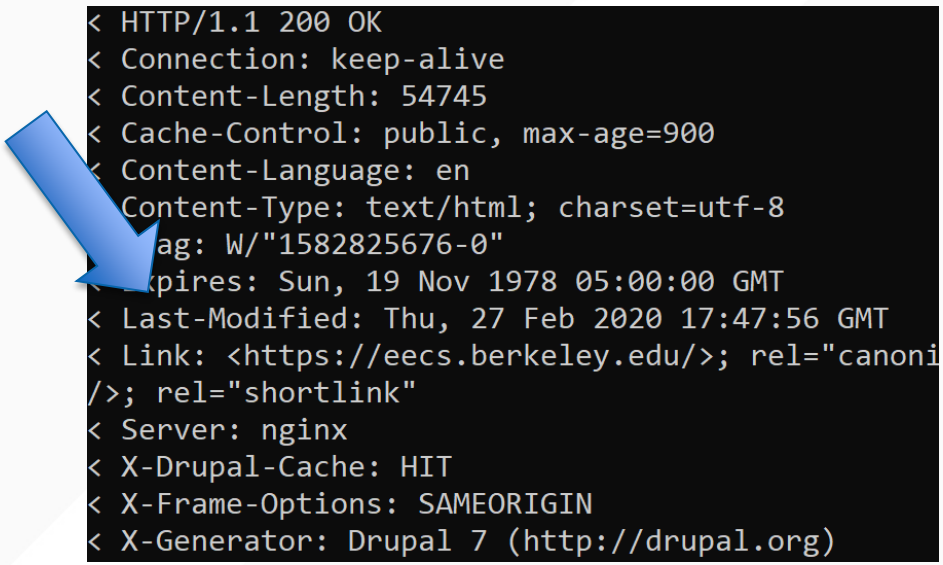
- Let’s have the proxy also cache copies of documents
  - Reduce latency
  - Reduce bandwidth to origin server
  - Share document across local users

- But how does the proxy “know” if the original document has been updated?

APPROACH 1: IF-MODIFIED-SINCE (TIME-BASED)

```
curl --http1.1 -o /dev/null -v https://cs.berkeley.edu
```

- Response header:

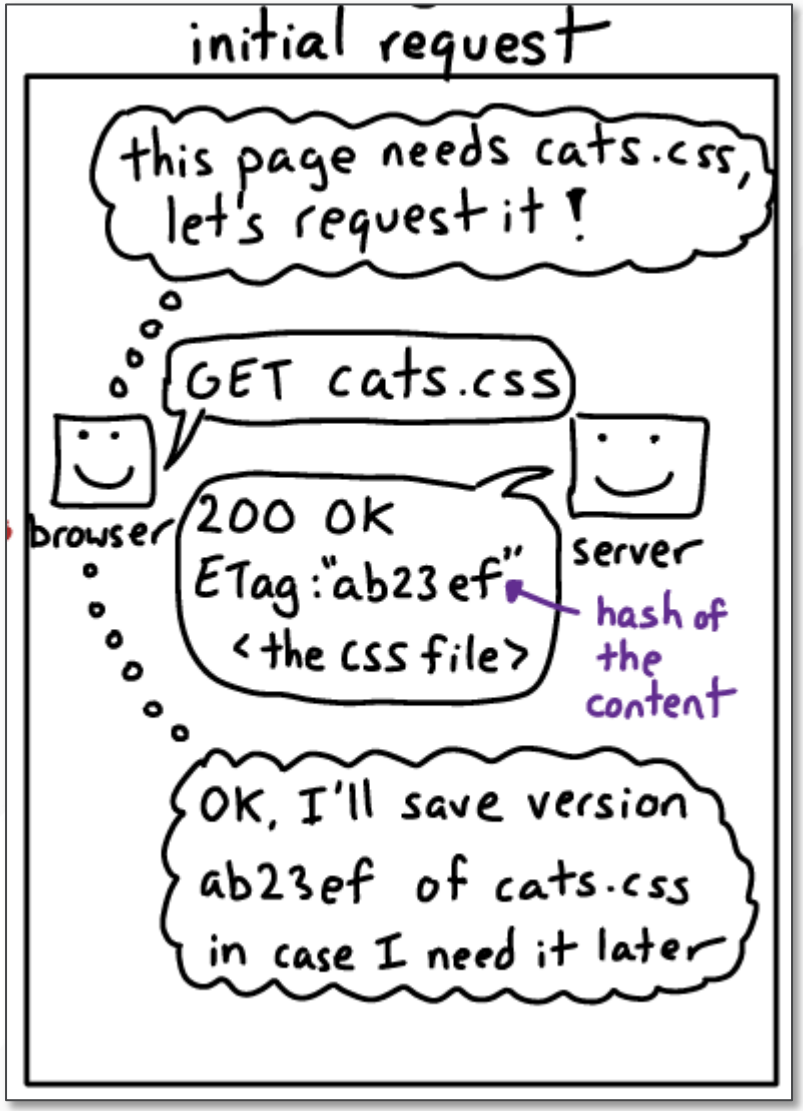


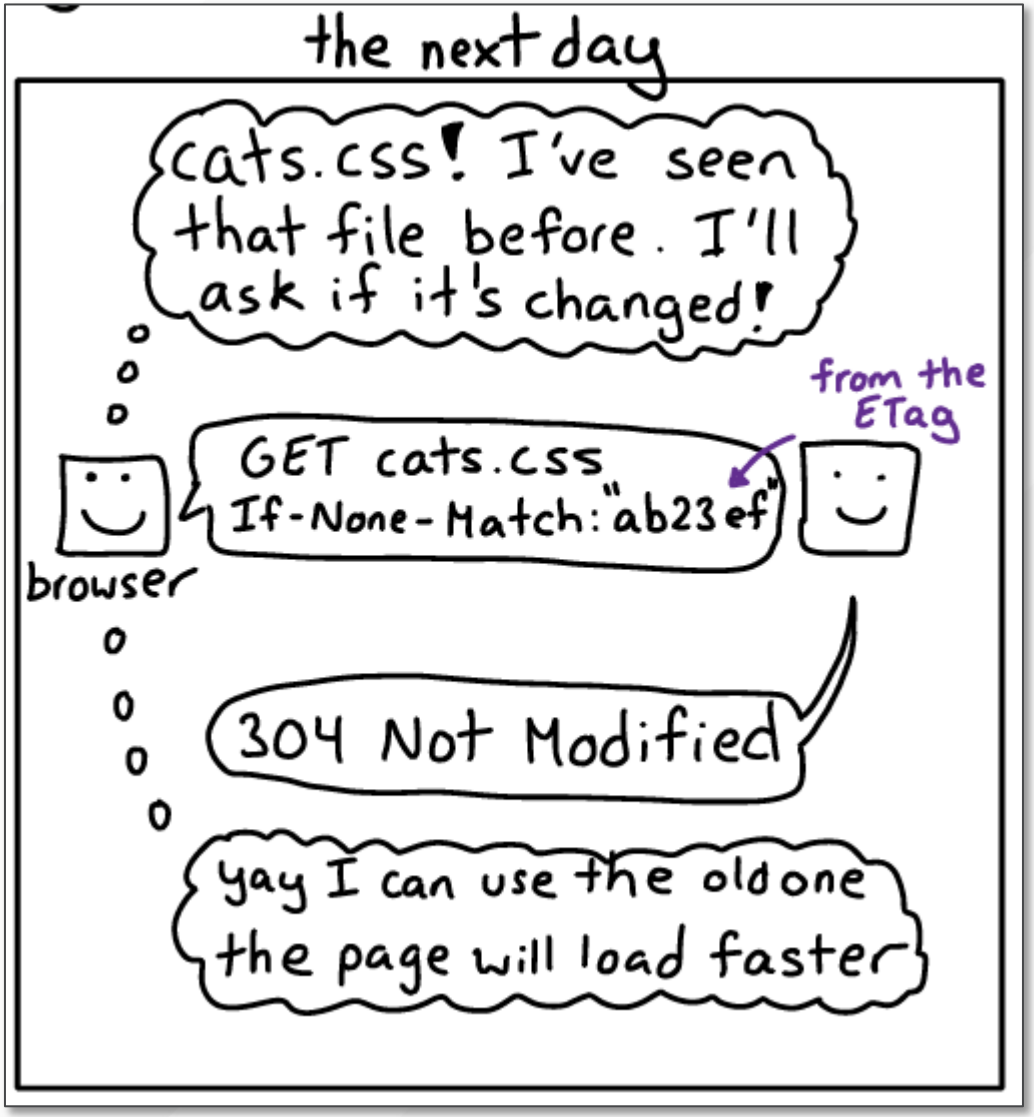
- Request header:
  - If-Modified-Since: <date>
    - Cache hit: 304 Not Modified
    - Cache miss: 200 OK

- Similar to our discussion of NFS (Networked File System) time-based cache verification policy

Question: If finding out it the browser's local cached copy is state or not requires a GET request, what is the benefit of returning 304 sometimes vs 200 every time?

APPROACH 2: ETAG/IF-NONE-MATCH (HASH-BASED)





LET'S TRY IT

- Curl has a "-H" option to specify our own headers
- What happens if we pass in the previously seen Etag?

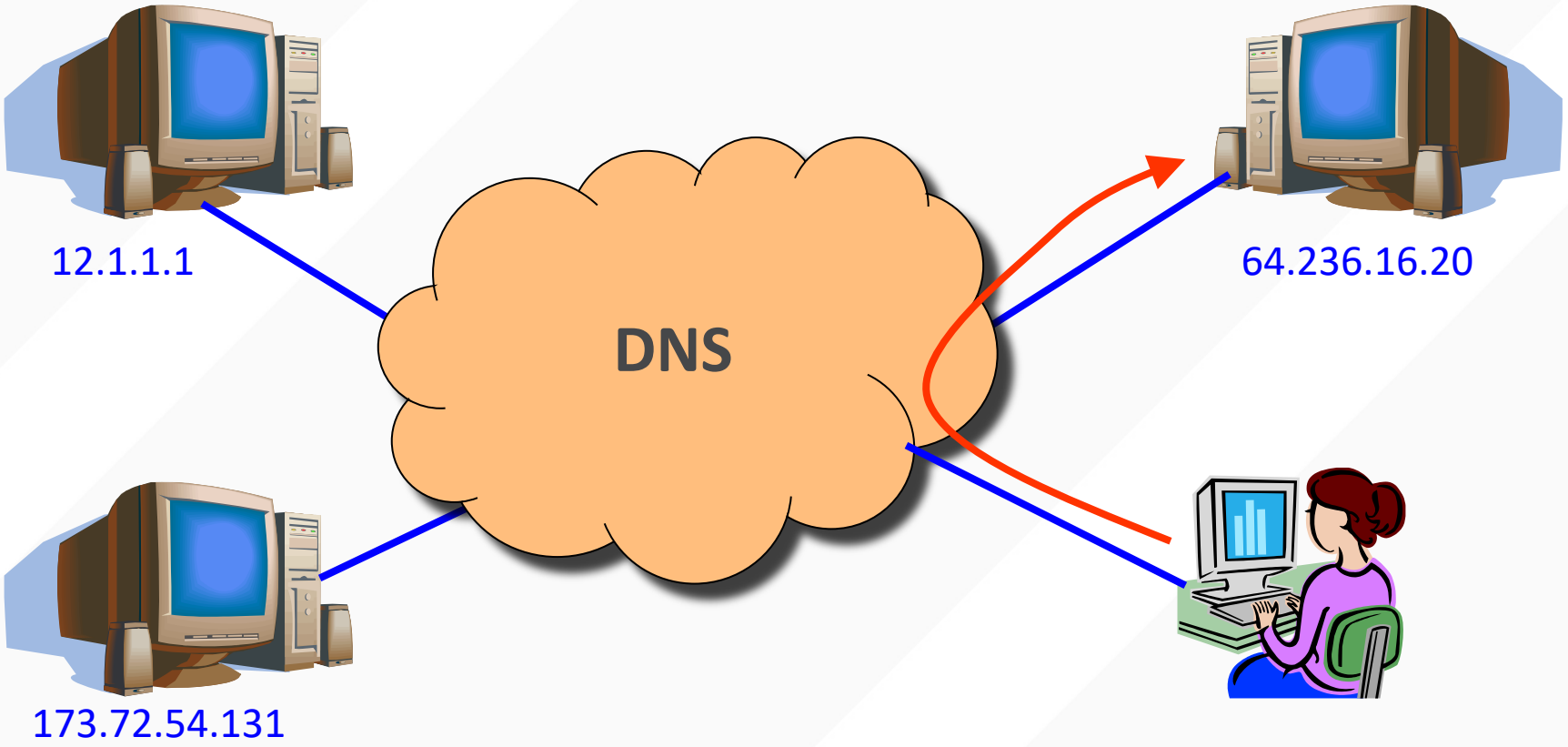
1. curl --http1.1 -v -o /dev/null https://cs.Berkeley.edu
2. curl --http1.1 -v -o /dev/null -H "If-None-Match: \"<etag>\"" https://cs.Berkeley.edu

REVERSE PROXIES: APPROACH 1 (STATIC)

- **Problem: Overloaded** popular web site
  - **Replicate** the site across multiple machines
    - Reverse proxies
- 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 name (www1, www2, www3, etc)
  - Some Web page lists replicas (*e.g.*, by name, location), asks clients to click link to pick

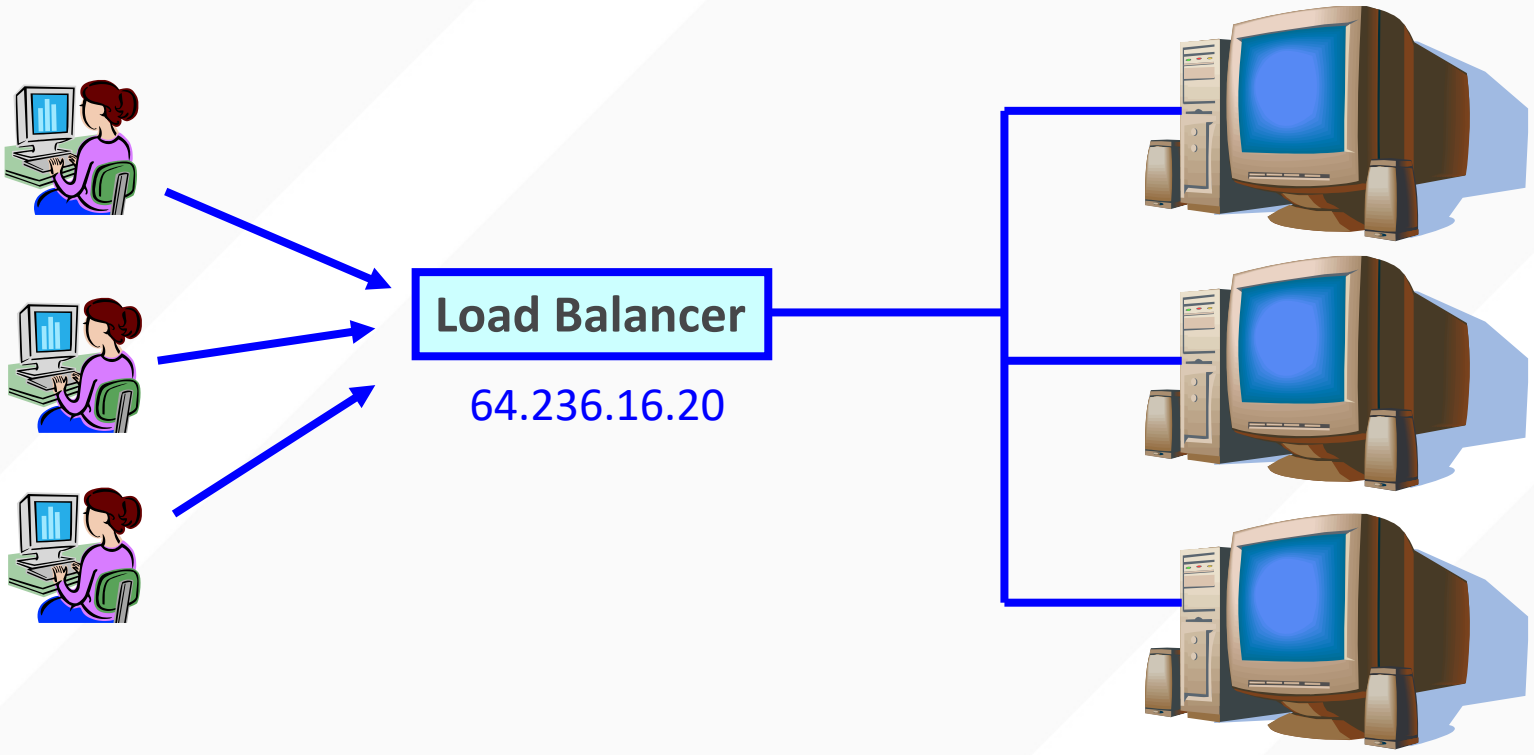
REVERSE PROXIES: APPROACH 2 (DNS)

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



REVERSE PROXIES: APPROACH 3 (LOAD BALANCER)

- 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**

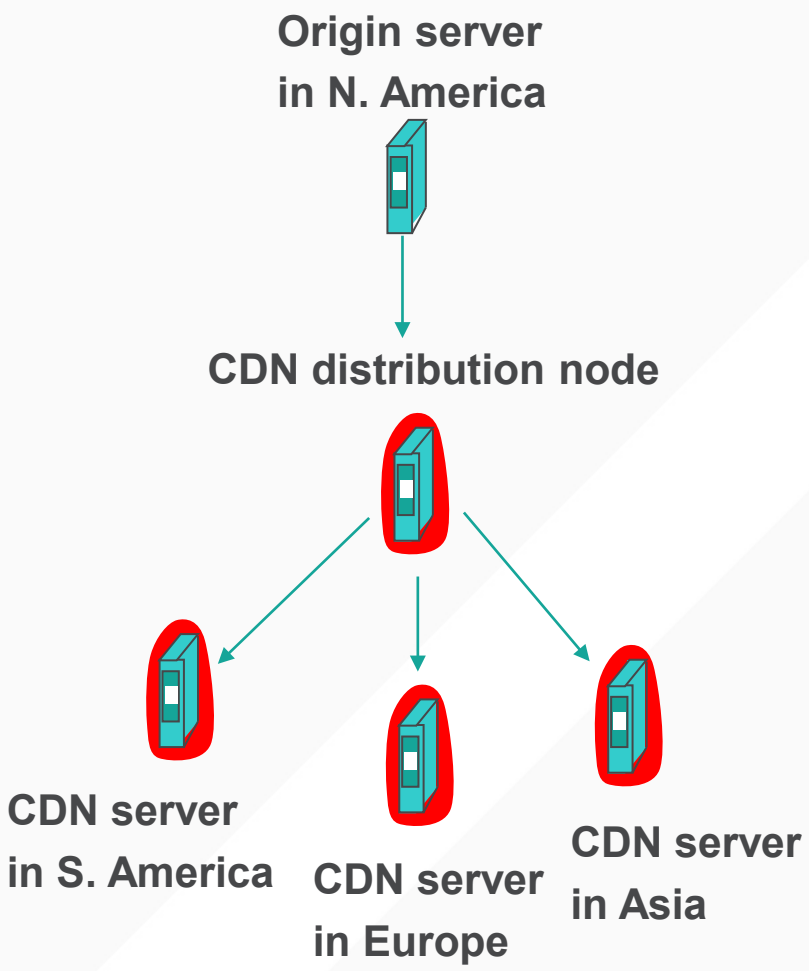
LOAD BALANCING VIA NETWORK ADDRESS TRANSLATION

OUTLINE

1. Web caching
2. Content-distribution networks
   - Featuring Akamai

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

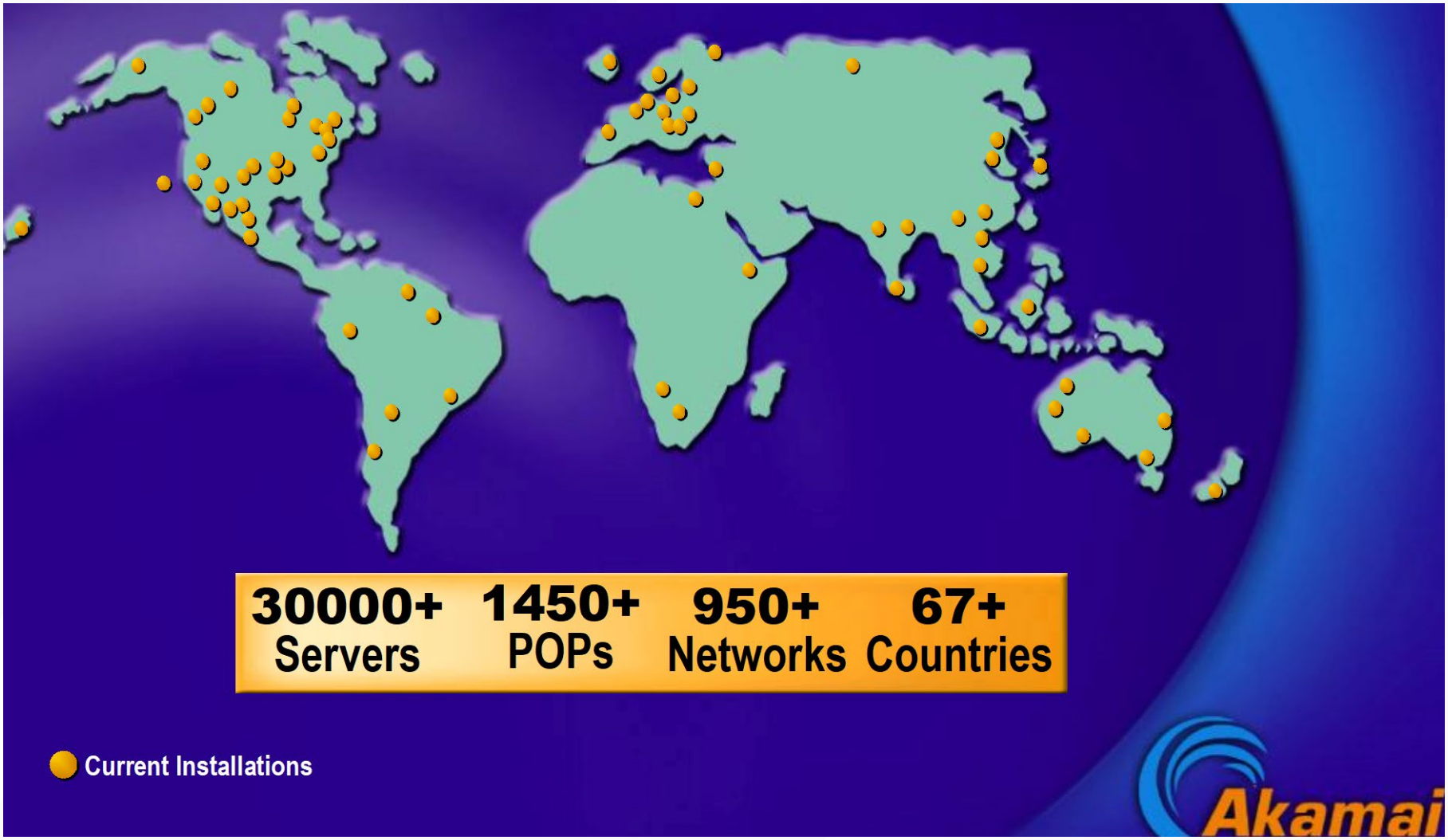
- **Live** server
  - For availability
- Lowest **load**
  - To balance load across the servers
- **Closest**
  - Nearest geographically, or in round-trip time
- Best **performance**
  - Throughput, latency, reliability…

Requires continuous monitoring of liveness, load, and performance

AKAMAI

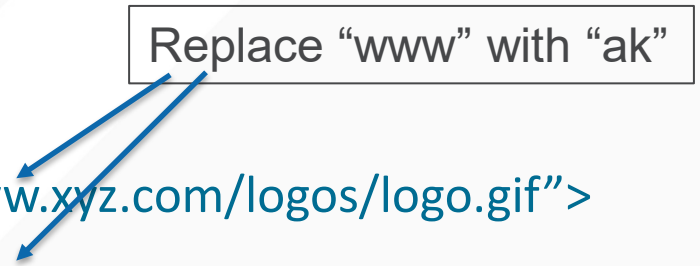
- Deployment
  - 147K+ servers, 1200+ networks, 650+ cities, 92 countries
  - highly hierarchical, caching depends on popularity
  - 4 yr depreciation of servers
  - Many servers inside ISPs, who are thrilled to have them
  - Deployed inside100 new networks in last few years
- Customers
  - 250K+ domains: all top 60 eCommerce sites, all top 30 M&E companies, 9 of 10 top banks, 13 of top 15 auto manufacturers
- Overall stats
  - 5+ terabits/second, 30+ million hits/second, 2+ trillion deliveries/day, 100+ PB/day, 10+ million concurrent streams
  - 15-30% of Web traffic

CIRCA 2007 OR SO

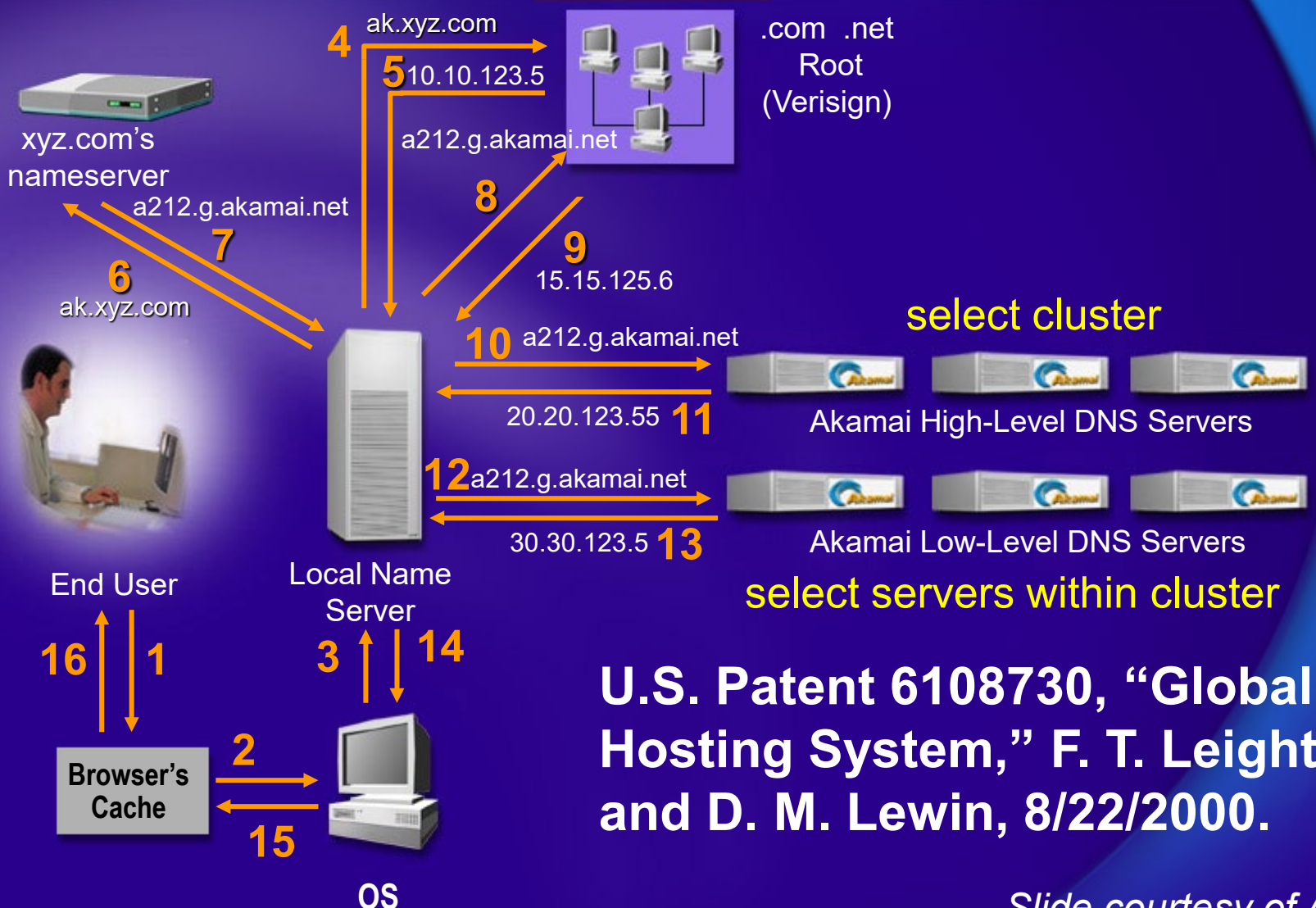


EMBEDDED IMAGE DELIVERY

```
<html>
<head>
<title>Welcome to xyz.com!</title>
</head>
<body>
<img src="http://www.xyz.com/logos/logo.gif">
<img src="http://www.xyz.com/jpgs/background.jpg">
<h1>Welcome to our Web site!</h1>
<a href="page2.html">Click here to enter</a>
</body>
</html>
```



Akamai DNS Resolution



U.S. Patent 6108730, "Global Hosting System," F. T. Leighton and D. M. Lewin, 8/22/2000.

Slide courtesy of Akamai

OPTIMIZING PERFORMANCE: NETWORK

- There are good solutions to server load and content
  - What about network performance?
- Key challenges for network performance
  - Measuring paths is hard
    - Traceroute gives us only the forward path
    - Shortest path != best path
  - RTT estimation is hard
    - Variable network conditions
    - May not represent end-to-end performance
  - No access to client-perceived performance

OPTIMIZING PERFORMANCE: NETWORK

- Example approximation strategies
  - Geographic mapping
    - Hard to map IP to location
    - Internet paths do not take shortest distance
  - Active measurement
    - Ping from all replicas to all routable prefixes
    - 56B * 100 servers * 500k prefixes = 500+MB of traffic per round
  - Passive measurement
    - Send fraction of clients to different servers, observe performance
    - Downside: Some clients get bad performance

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

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**

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)

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

UC San Diego