Content Distribution Networks

COS 418: Distributed Systems
Lecture 24
Kyle Jamieson

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

1. Domain Name System (DNS) primer
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

- Root
- TLDs: com, gov, edu
- fcc.gov, princeton.edu, nyu.edu, cs.princeton.edu

- 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?*

  - A Verisign, Dulles, VA
  - C Cogent, Herndon, VA
  - D U Maryland College Park, MD
  - E NASA Mt View, CA
  - F Internet Software Consortium, Palo Alto, CA
  - G US DoD Vienna, VA
  - H ARL Aberdeen, MD
  - J Verisign
  - K RIPE London
  - I Autonomica, Stockholm
  - M WIDE Tokyo
  - B USC-ISI Marina del Rey, CA
  - L ICANN Los Angeles, CA

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

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

DNS in operation

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

  Recursive: Nameserver responds with answer or error
  
  Client www.princeton.edu?
  
  Answer: www.princeton.edu A 140.180.223.42

  Iterative: Nameserver may respond with a referral
  
  Client Referral: .edu NS a.edu-servers.net.

A recursive DNS lookup

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)
**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

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”

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/1.0 webpage fetch: Timeline

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

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 ✗
Web caching

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

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
  - 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%)

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

Today

1. Domain Name System (DNS) primer

2. The Web: HTTP, hosting, and caching

3. Content distribution networks (CDNs)
  - Akamai case study

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

1. DNS lookup: cache.cnn.com
2. GET: index.html
3. AlIAS: g.akamai.net
4. End user
How Akamai Uses DNS

cnn.com (content provider)

DNS TLD server

Akamai global DNS server

Akamai cluster

Akamai regional DNS server

Nearby Akamai cluster

Address 1.2.3.4

End user

GET /foo.jpg
Host: cache.cnn.com
How Akamai Works: Cache Hit

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>End user requests a page from cnn.com</td>
</tr>
<tr>
<td>2</td>
<td>dns tld server resolves cnn.com to an Akamai global DNS server</td>
</tr>
<tr>
<td>3</td>
<td>Akamai global DNS server returns the address of the closest Akamai regional DNS server</td>
</tr>
<tr>
<td>4</td>
<td>Akamai regional DNS server returns the address of the closest Akamai cluster</td>
</tr>
<tr>
<td>5</td>
<td>Nearby Akamai cluster returns the address of the server to serve the request</td>
</tr>
<tr>
<td>6</td>
<td>Server in the cluster serves the request</td>
</tr>
</tbody>
</table>

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 day
  - 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

Friday precept:
Extended office hours