CSCI-1680
DNS
Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti
Administrivia

- **TCP milestone II**: sign up for a meeting this week *(announcement soon)* (BY FRI)

- **TCP gearup III**: tentative, but probably this Thursday 5-7pm

- **HW3**: due tonight—it’s short!

We’re working through our grading backlog, should have progress soon
Connecting to a server: the story so far

POV: You want to connect to some website

connecting (5.6.7.8, 80)

Is this how users interact with the network? No!
Might have multiple IPs per service
Less error-prone (user don’t want to type/remember names)
IP addresses can be reassigned
Users don’t know IPs
Client applications don’t know IPs of server
IPs depend on where you are located on the network
What we have

IP addresses
• Used by routers to forward packets
• Fixed length, binary numbers
• Assigned based on where host is on the network
• Usually refers to one host

Examples
• 5.6.7.8
• 212.58.224.138
• 2620:6e:6000:900:c1d:c9f7:8a1c:2f48

=> Need a new abstraction for “stuff” we are trying to access
What we want: a new abstraction for names

- connect("website.com", 80)
- connect(5.6.7.8, 80)

DECOUPLING NAME OF HOST OR SERVICE FROM THE IP
What we want: a new abstraction for names

Want:
- names
- Human-readable
- Variable length
- Don’t need to care about where destination is/what server it is

=> Can refer to a service, not just a host
What does this mean?

**DNS**

cs.brown.edu => 128.148.32.110

**Why?**
- Names are easier to remember
- Addresses can change underneath
- Useful Multiplexing/sharing

---

ONE NAME => MULTIPLE IPS

MULTIPLE NAME => ONE IP.
Another Change in Layers...

- Remember ARP
  - ARP: maps IP addresses to MAC addresses

ARP: WHO HAS 1.2.3.4?
⇒ AA:BB:CC:DD:EE:FF

---

DNS: NAME: "WHO HAS GOOGLE.COM?"

NETWORK LAYER INFO: 1.2.3.4

Answer
The original way: one file: `hosts.txt`
- Flat namespace
- Central administrator kept master copy (for the Internet)
- To add a host, emailed admin
- Downloaded file regularly
<table>
<thead>
<tr>
<th>HOST NAME</th>
<th>HOST ADDRESS</th>
<th>SPONSOR</th>
<th>LIAISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>10.2.0.54</td>
<td>VDM</td>
<td>ARPA</td>
</tr>
<tr>
<td>CPUtype:</td>
<td>PDP-11/70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCAT-T1P</td>
<td>10.2.0.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPUtype:</td>
<td>H-316</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AEROSPACE</td>
<td>10.2.0.65</td>
<td>AFSC</td>
<td></td>
</tr>
<tr>
<td>CPUtype:</td>
<td>VAX-11/780</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFGL</td>
<td>10.1.0.66</td>
<td>AFSC</td>
<td></td>
</tr>
<tr>
<td>CPUtype:</td>
<td>PDP-11/50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFGL-TAC</td>
<td>10.2.0.66</td>
<td>AFSC</td>
<td></td>
</tr>
<tr>
<td>CPUtype:</td>
<td>C/30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scalable (Address <-> Name) Mappings

Original way: one file: hosts.txt
- Flat namespace
- Central administrator kept master copy (for the Internet)
- To add a host, emailed admin
- Downloaded file regularly

Is this feasible today? Lol no.
Domain Name System (DNS)
• Originally proposed by RFC882, RFC883 (1983)

• Distributed protocol to translate hostnames -> IP addresses
  – Human-readable names
  – Delegated control
  – Load-balancing/content delivery
  – So much more…

=> Distributed key-value store, before it was cool…
High-level DNS goals

Scalability: need to be able to have a huge number of “records”
- Lots of queries for names
- Lots of updates (though updates << queries)

Distributed control: need to let people/organizations etc control their own names

Redundancy/fault tolerance
- Need to have redundant way to do lookups, provide name records

Some properties about the system that make this possible
- Loose consistency: when changing records, not a huge problem if it takes a while to propagate (several minutes)
- Read-mostly database: can do lots of caching for records all over the world
The good news

Compared to other distributed systems, some properties that make these goals easier to achieve…

1. Read-mostly database
   Lookups MUCH more frequent than updates

2. Loose consistency
   When adding a machine, not end of the world if it takes minutes or hours to propagate

Can use lots and lots of caching

- Once you’ve lookup up a hostname, remember
- Don’t have to look again in the near future
How it works

Hierarchical namespace broken into zones

cslab1a.cs.brown.edu

- Hostname (one node/service)
- Subdomain
- Top-level domain (TLD)
- Name of some org/entity
- Managed by "registrars"
Types of DNS servers

- “Authoritative” servers: servers that have records for some domain (servers that “own” the records for cs.brown.edu)

- Resolver: you (or another DNS server) queries it to look up names, tries to get closer to authoritative server
  
  =>$> in most cases you interact with, will find authoritative server
How it works

- Hierarchical namespace broken into zones
  - root (.), edu., brown.edu., cs.brown.edu.,
  - Zones separately administered => delegation
  - Parent zone tells you how to find servers for subdomains
- Each zone served from multiple replicated servers
- Lots and lots of caching
“Types” of DNS servers

- Top Level Domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, br, tv, in, ly)
  - Special domains (e.g., arpa)
  - Corporate domains (...)

- Authoritative DNS servers
  - Provides public records for hosts at an organization
  - Can be maintained locally or by a service provider

- Recursive resolvers
  - Big public servers, or local to a network
  - Lots of caching
How a DNS query works: Iterative Version

1. **Host** asks **Local Resolver**
2. **Resolver** starts **Recursive** query from **Root**
3. **Intermediate Nameservers** don't have answer but **respond** with next server that knows more
4. **Found server** with **Authoritative Answer**

Host: ggfiiio.nu
Q: cslab1a.cs.brown.edu
NOTE: Resolver can **cache** any of these responses for later.

1. *Host* asks **Local Resolver**
2. **Resolver** starts **Recursive** query from **Root**
3. **Intermediate Nameservers** don't have answer but **respond** with next server that knows more
4. **Found server** with **Authoritative Answer**!
More commonly, hosts perform recursive queries to larger DNS servers, which do the typical iteration process (from the previous page) on the client's behalf.

Why? All resolvers cache responses—a larger resolver is more likely to have these entries in its cache. If the resolver has a valid answer for any of the steps, it can skip it! (For example, if the nameserver for .edu is cached but cs.brown.edu is not, the local resolver can skip steps 2-3.)
Resolver operation

- Apps make **recursive** queries to local DNS server (1)
  - Ask server to get answer for you
- Server makes **iterative** queries to remote servers (2,4,6)
  - Ask servers who to ask next
  - Cache results aggressively

**DNS software architecture**

- Two types of query:
  - Recursive
  - Non-Recursive
- Apps make recursive queries to local DNS server (1)
- Local server queries remote servers non-recursively (2, 4, 6)
  - Aggressively caches result
  - E.g., only contact root on first query
  - Ending .umass.edu
DNS Caching

• Recursive queries are expensive
• Caching greatly reduces overhead
  – Top level servers very rarely change
  – Popular sites visited often
  – Local DNS server caches information from many users
• How long do you store a cached response?
  – Original server tells you: TTL entry
  – Server deletes entry after TTL expires

WHEN TTL EXPIRES
DELETE CACHE ENTRY
Where is the root server?

- Located in New York
- How do we make the root scale?

Verisign, New York, NY
DNS Root Servers

- 13 Root Servers (www.root-servers.org)
  - Labeled A through M (e.g, A.ROOT-SERVERS.NET)
- Does this scale?
DNS Root Servers

- 13 Root Servers (www.root-servers.org)
  - Labeled A through M (e.g., A.ROOT-SERVERS.NET)
- Remember anycast?
DNS Root Servers: Today

From: www.root-servers.org
DNS Example

$ dig cs.brown.edu @10.1.1.10
; <<>> Dig 9.10.6 <<>> cs.brown.edu @10.1.1.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8536
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
;; QUESTION SECTION:
;cs.brown.edu. IN A

;; ANSWER SECTION:
;cs.brown.edu. 1800 IN A 128.148.32.12

;; Query time: 69 msec
;; SERVER: 10.1.1.10#53(10.1.1.10)
;; WHEN: Tue Apr 19 09:03:39 EDT 2022
;; MSG SIZE rcvd: 57
$ dig cs.brown.edu @10.1.1.10
; <<>> DiG 9.10.6 <<>> cs.brown.edu @10.1.1.10
;; global options: +cmd
;; Got answer:
;; HEADER<<
;; opcode: QUERY, status: NOERROR, id: 8536
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
;; QUESTION SECTION:
;cs.brown.edu. IN A

;; ANSWER SECTION:

;; Query time: 69 msec
;; SERVER: 10.1.1.10#53(10.1.1.10)
;; WHEN: Tue Apr 19 09:03:39 EDT 2022
;; MSG SIZE rcvd: 57
% dig +norec cs.brown.edu @j.root-servers.net

; <<>> DiG 9.10.6 <<>> +norec cs.brown.edu @j.root-servers.net
;; global options: +cmd
;; Got answer:
;; ->>>HEADER<<- opcode: QUERY, status: NOERROR, id: 61618
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;cs.brown.edu. IN A

;; AUTHORITY SECTION:
edu. 172800 IN NS a.edu-servers.net.
edu. 172800 IN NS b.edu-servers.net.
edu. 172800 IN NS l.edu-servers.net.
edu. 172800 IN NS m.edu-servers.net.

;; ADDITIONAL SECTION:
a.edu-servers.net. 172800 IN A 192.5.6.30
b.edu-servers.net. 172800 IN A 192.33.14.30
c.edu-servers.net. 172800 IN A 192.26.92.30
d.edu-servers.net. 172800 IN A 192.31.80.30
e.edu-servers.net. 172800 IN A 192.12.94.30

When server doesn’t know all info...

No answers, but lists other servers to try.