# ECE 435 – Network Engineering Lecture 8

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

12:30pm, Barrows 125

8 February 2024

## Announcements

- HW#3 was due
- HW#4 will be Posted
  Using tools to access DNS info, let me know if you have trouble.
- Working on getting HW#2 graded

## Domain Name System (DNS)

- Hierarchical distributed database
- Maps hostnames to IP addresses
- Why do we need it?
  - Can you remember numbers? Send e-mails to vince@192.168.8.1?
  - What if server moves?
- RFC 1034, 1035 (1987), Supersedes RFC 882, 883 (1983)
- If there's a network problem, it's "always DNS" (or

maybe BGP)

## Ancient History

- In early days NIC.arpa has a "HOSTS.TXT" file you downloaded occasionally with all known machines. Didn't really scale.
- Trivia, called SRI (stanford research) on phone to get Elizabeth "Jake" Feinler during business hours and she'd manually add you to list.
- /etc/hosts is a relic of this, usually checked first
- On Linux this is configured via `/etc/nsswitch.conf`

## Domain Names

- Which ones can you name? .com/.org/.gov/.edu/.net/.mil
- Country codes (.us/.uk/.ie etc)
- Huge expansion in the last few years (.horse)
- Owner of a domain can subdivide, i.e. eece.maine.edu
- How do you buy them? Used to be fairly expensive and only for two years at a time from a single registrar. Not so much anymore.
- whois will show you info on who owns (less details than old days)

## Name Rules

- Can have 127 levels, each 1-63 chars.
- Usually total name cannot exceed 253 chars.
- LDH (letters,digits,hyphens, cannot start with hyphen, not all numbers)
- Case-insensitive
- International names: “punycode”. Trouble, why? Foreign letters that look like ASCII ones.
- punycode – snowman example `http://xn--n3h.net/`
- First commercial name 15 March 1985 symbolics.com

example.com set aside (why be careful with your example names?)

- Shortest? g.cn. Various one-letter domains (like x.org) but they were later reserved.
- Typosquatting, domain squatting, copyrighted names, etc.

## DNS Server

- Listens on port 53, usually UDP
  (Special case if $>$ 512 bytes: use TCP)
- A simple request might look something like:
  a bunch of flags specifying options
  `google.com type: A class: IN`
- A simpler response will restate the question then have the response: `google.com type: A class: IN: addr 1.2.3.4`
- Note it's a binary protocol, not chatty ascii text

## Zone Records

- 5-tuple, NAME TTL CLASS TYPE VALUE
  - TTL (how long to cache)
  - Class (usually IN for internet)

    Mostly reserved, with two obsolete networks chaos, hesiod
  - Type and RDATA (resource data)
  - Common types
    - SOA – start of authority (parameters) primary source, e-mail of admin, etc

- A – IPv4 address of host (32bit int) `linux.deater.net 86400 IN A 1.2.3.4` can have multiple and be cycled through round-robin
- AAAA – IPv6
- MX – Mail exchange (can have multiple, specify priority)
- NS – name sever (name server for this domain)
- CNAME – Canonical name, allows aliases
- PTR – alias for IP, for reverse lookup `4.3.2.1.in-addr.arpa`
- HINFO – cpu and OS type (text) (uncommon)

  - TXT – raw ASCII text
  - SRV – new – sort of generic version of MX
  - SPF – which machines can send e-mails (avoid spam)
- Can you store other things in records? Text adventure? File transfer? Tunneling (iodine?)

## DNS client

- Name resolver, translate from ASCII (still?) name to IP addr

## DNS Lookup Example

- DNS recursor, gets requests from application (like web browser) and does the lookup
- root nameserver, master index
- top level domain (tld) nameserver, has lookups for things like .com
- authoritative nameserver, final authority on a name

## Three types of requests

1. Recursive, ask recursive resolver, want the address or an error

2. Iterative, if server doesn't have the info, will return a referral to server one further down domain space

3. Non-recursive. Respond from server directly either because it's authoritative, or else cached

## Eight steps in DNS lookup

1. User/app makes DNS request, it travels to remote resolver
2. Resolver queries a root DNS server
3. Root resolver responds with address of a TLD nameserver. For example.com, would return the .com registry namserver
4. Resolver then makes request to the TLD nameserver

5. TLD nameserver responds with address of the domain's nameserver
6. Resolver makes request to the domain's nameserver
7. Address of domain is returned
8. DNS resolver then passes back to original application

## Root DNS Servers

- 13 root servers
- mostly in US
- Single-letter server names, limitation of number that can fit in single 512B UDP packet
- a.root-servers.net through m.root-servers.net
- One at University of Maryland

## Caching

- Applications can cache lookups
  Things like browsers don't want to keep re-looking up
- The Operating System can cache lookups (stub resolver)
- Often the NS entries are caches and can be used directly, or the TLD is cached to avoid loading down root server

## Another DNS Lookup Example

- Basically: application calls a library (resolver) with the hostname.
  gethostbyname() in socket examples
  This sends UDP to local DNS server, which figures out the address (possibly recursively) and returns the address to caller.
- Details
  - First check `/etc/nsswitch.conf` which might say to check `/etc/hosts` and maybe NIS/LDAP first

- Query via UDP local nameserver (`/etc/resolv.conf`)
- If the local is the official nameserver, get *authoritative response* (from responsible zone)
  the alternative is a cached response
- If local DNS server doesn't know about it, it has to ask up the chain.
- If not known, query "root" server. So if looking up weaver-lab.eece.maine.edu will ask root, which will direct to .edu DNS server
- 13 root servers, mostly in US
  Single-letter server names, limitation of number that

can fit in single 512B UDP packet

- ○ Recursive query It will not likely know but will know about maine.edu, so ask that one, which will ask eece.maine.edu, etc, and then passed back
- ○ Result is then cached along the way (TTL) caching up to 68 years (or none at all). Why low values? Why can that be bad?
- ○ Caching also means usually the root server does not have to respond to each request
- ○ As the response is passed back through it will be cached along way.

## How do you know what DNS server to use?

- Usually your ISP would tell you
- These days set up so DHCP will set it up for you
- Companies offer “easy to remember” ones you can use, google 8.8.8.8 and cloudflare 1.1.1.1

# DNS Query with dig – dnsutils

```
dig weaver-lab.eece.maine.edu

; <<>> DiG 9.11.3-2-Debian <<>> weaver-lab.eece.maine.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1268
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;weaver-lab.eece.maine.edu. IN A

;; ANSWER SECTION:
weaver-lab.eece.maine.edu. 3599 IN A 130.111.218.24

;; Query time: 79 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Sep 25 14:12:42 EDT 2018
;; MSG SIZE  rcvd: 70
```

## Reverse DNS request

- Given IP address, how can you find the name?
- Linux can use the "host" command.
- For IPv4, there is special in-addr.arpa domain
- To look up 1.2.3.4, lookup 4.3.2.1.in-addr.arpa
- It will iterate down. This gets trickier now with non-contiguous IP allocations.
- Similar thing for IPv6 using ip6.arpa

## DNS Packet format

- Packet format

| 16-bits | 16-bits |
|---|---|
| ID | Control Flags |
| Query Count | Answer Count |
| Authority Count | Additional Count |
| . . . | |
| . . . | |

- Flags
  - QR – request (0) or response (1)
  - OpCode – QUERY, IQUERY, STATUS, NOTIFY,

UPDATE

- ◦ AA – Authoritative Answer (1) or cache (0)
- ◦ Truncated – (1) means too big for UDP
- ◦ RD – Recursion Desired
- ◦ RA – Recursion Available
- ◦ Z – zeros (reserved)
- ◦ AD – Authenticated Data (DNSSEC)
- ◦ CD – Checking Disabled (DNSSEC)
- ◦ RCODE – Error Code

- Counts say how many of each included
- Then the actual requests

## Zone Transfers

- Zone transfers – copying zone list between machines

## Name Server Software

- Can you set up your own?
- BIND/named
- dig / nslookup tools

## DNS Security – DNSSEC

- RFC 3833
- Digitally sign response
- Can provide things like public keys
- Backwards compatible
- Slow uptake

## DNS Security – Cache Poisoning Attack

- Make request that causes a recursive lookup
- Immediately flood server with spoofed response packets with inaccurate info
- This is easier as only 16-bit IDs in protocol so easy to try all of them
- Can get inaccurate info into cache and will hand out bad info

## DNS Security – 0x20 Encoding

- Used to avoid cache poisoning attack
- DNS lookups case independent
- So server can toggle case in random way, eXaMPlE.cOm
- (Toggling case is equivalent to flipping bit 0x20)
- Poison attacks would have to not only guess 16-bit value but also the pattern of flipped bits which is much harder

## DNS Security – Amplification Attack

- Send requests to DNS server with spoofed return address
- UDP makes this easy
- Do this with enough servers, can be DDoS

## DNS Tunneling

- Tunnel DNS over https? More secure? Privacy implications?
  Some browsers doing this?

## DNS Privacy

- Can people spy on your web-browsing through DNS?
- Can a web-browser tunnel DNS over https?
- 1.1.1.1 and 8.8.8.8 name servers?