ECE 435 – Network Engineering
Lecture 12

Vince Weaver

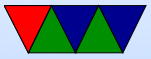
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

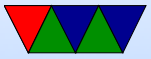
20 February 2026

Announcements

- HW#4 will be posted (e-mail, DNS)

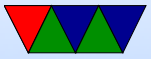


DNS Continued



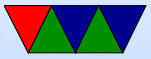
Root DNS Servers

- Traditionally were 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
- These days clusters of machines around world mirrored using anycast, see `https://root-servers.org/`
- There was one (D) at University of Maryland when I was there



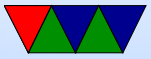
Recursive Query

- The recursive lookup might have to start with the root server if it doesn't know the domain (but not if it's cached)
- It will then ask the .edu server for .maine.edu
- Then if it doesn't know .eece.maine.edu it might have to ask another
- Ideally things are cached all along the way



DNS Lookup – Caching

- Results are cached
- Length of time in (TTL) filed
  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
- Applications can cache lookups
  Things like browsers don't want to keep re-looking up
- The Operating System can cache lookups (stub resolver)

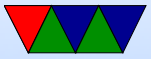


Three types of requests

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

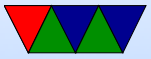
2. Iterative, if server doesn't have the info, will return a referral to server one further down domain space

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



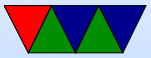
How do you know what DNS server to use?

- 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



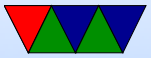
DNS Query with dig – dnsutils

```
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

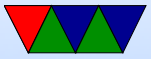


DNS Packet format

- Packet format

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

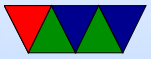
- Flags
  - QR – request (0) or response (1)
  - OpCode – QUERY, IQUERY, STATUS, NOTIFY,



UPDATE

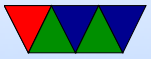
- AA – Authoritative Answer (1) or cache (0)
- Truncated – (1) means too big for UDP
- RD – Recursion Desired
- RA – Recursion Available
- Z – zeros (reserved)
- AD – Authenticated Data (DNSSEC)
- CD – Checking Disabled (DNSSEC)
- RCODE – Error Code

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



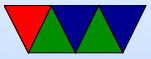
Decoding DNS packet

- Would be nice to decode a DNS packet from a network dump
- Turns out decoding a DNS packet is *really* tricky
- Original protocol was relatively straightforward but modern real-life there's a lot going on



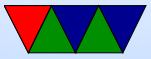
Zone Transfers

- Zone transfers – copying zone list between machines



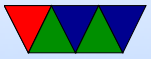
Name Server Software

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



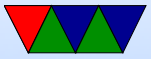
DNS Security – DNSSEC

- RFC 3833
- Digitally sign response
- Can provide things like public keys
- Backwards compatible
- Slow uptake
- Article on DNSSEC `https://www.potaroo.net/ispcol/2024-05/dnssec-fin.html`



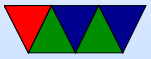
DNS Security – Cache Poisoning Attack

- 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



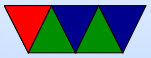
DNS Security – 0x20 Encoding

- 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



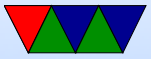
DNS Security – Amplification Attack

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



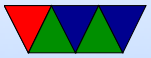
DNSSEC Security – Crashing Attack

- https://www.theregister.com/2024/02/13/dnssec_vulnerability_internet/
- 1/3 of DNS servers handle DNSSEC?
- Sending specially crafted encrypted data can take hours to decode, essentially DOSing the DNS server



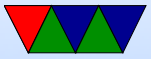
DNS Privacy

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



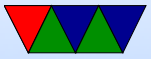
https DNS Tunneling

- Some browsers want to tunnel DNS over https g
- Bypass your ISP's DNS servers and use ones from your browser
- Is this more or less secure?
- Are there privacy implications?
- Why might your ISP/company not like this?



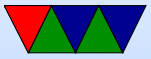
DNS in the news

- https://blog.cloudflare.com/cname-a-record-order-dns-standards/



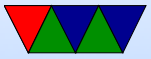
The Transport Layer

	OSI	TCP/IP
7	Application	Application
6	Presentation	
5	Session	
4	Transport	Transport
3	Network	Internet
2	Data Link	Host-to-network
1	Physical	Host-to-network

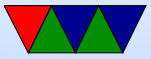


The Transport Layer

- Responsible for reliable point-to-point data transport independent of whatever lies beneath.
- Sender: receives data from application, breaks up to "segments", adds port number, goes to net layer
- Receiver: re-assembles "segments", passes data to application listening on port
- Provide process-to-process connectivity, and per-segment error control and per-flow reliability, as well as rate control

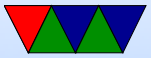


- Can be more reliable than underlying network
- Most common interface “socket” API from homeworks.



Some Transport Layer Protocols

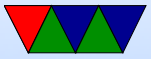
- TCP (Transmission Control Protocol)
  - connection oriented / stateful / per-flow reliability and rate control
- UDP (User Datagram Protocol)
  - stateless / connectionless
- SCTP (stream control transmission protocol)
  - messages like UDP, reliable like TCP
- QUIC
  - running reliable connection over UDP



The Transport Layer

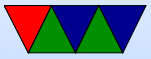
- Terminology: application = process, data-transfer-unit is a segment, traffic is a flow
- addressing – each process needs a unique ID. For TCP/UDP, this is the "port" number (16-bit)
- Rate control
  - Flow control – between source and destination
  - Congestion control – between source and network

    None in link layer because only one hop?



Can be done by sender or network

- Real time requirements – things like video and audio need extra info such as timestamp, loss rate, etc. So hard to do with raw TCP/UDP



Started on UDP, see next lecture for those notes