14. How the Internet Works (Part 2) & How the Web Works (Part 1)

Blase Ur and David Cash
(Some slides borrowed from Ben Zhao)
February 11th, 2022
CMSC 23200 / 33250
Protocols at different layers

L7  Application
    - SMTP
    - HTTP
    - DNS
    - NTP

L4  Transport
    - TCP
    - UDP

L3  Network
    - IP
    - Ethernet
    - FDDI
    - PPP

L2  Data link
    - optical
    - copper
    - radio
    - PSTN

L1  Physical
Goal: Get ALL of the data to its destination
Solution (Protocol): TCP at the transport layer
TCP (Transmission Control Protocol)

- Multiplexes between services
- Multi-packet connections
- Handles loss, duplication, & out-of-order delivery — all received data ACKnowledged
- Flow control — sender doesn’t overwhelm recipient
- Congestion control — sender doesn’t overwhelm network
TCP header

<table>
<thead>
<tr>
<th>bit 0</th>
<th>source port</th>
<th>destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>acknowledgement number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d-offset</td>
<td>receive window size (bytes)</td>
</tr>
<tr>
<td></td>
<td>TCP checksum</td>
<td>urgent pointer</td>
</tr>
</tbody>
</table>

**Data (TCP payload)**

*(If d-offset > 5, the data is preceded by an *options* field)*

**Flag bits in word 4**

- F(IN): finish
- S(YN): synchronize
- R(ST): reset
- P(SH): push
- A(CK): acknowledge
- U(RG): urgent
- o(ther)
- r(eserved)

**IP data (IP datagram payload)**

<table>
<thead>
<tr>
<th>IP header</th>
<th>TCP/UDP header</th>
<th>TCP/UDP payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Common TCP (Default) Ports

- 22: SSH
- 25: SMTP
- 53: DNS
- 67, 68: DHCP
- 80: HTTP
- 143: IMAP
- 443: HTTPS
- Ports 49152-65535 are used by client programs
TCP connections

- Explicit connection setup & teardown

- Explicit control flags (e.g., SYN, ACK, FIN, RST)

- Sequence numbers — reliability & ordering

Source: Wikimedia commons
TCP Sequence Numbers

- Bytes in a TCP sequence are numbered (and acked)
Protocols at different layers

L7  Application  
    |       |
    |       |
    |       |
    |   SMTP |
    |       |
    |       |
    |       |
    |   HTTP |
    |       |
    |       |
    |   DNS  |
    |       |
    |       |
    |   NTP  |

L4  Transport  
    |       |
    |       |
    |       |
    | TCP   |
    |       |
    |       |
    |   UDP |

L3  Network  
    |       |
    |       |
    |       |
    | IP    |
    |       |
    |       |

L2  Data link  
    |       |
    |       |
    |       |
    | Ethernet |
    |       |
    |       |
    | FDDI   |
    |       |
    | PPP    |

L1  Physical  
    |       |
    |       |
    |       |
    | optical |
    |       |
    |       |
    | copper |
    |       |
    | radio  |
    |       |
    | PSTN  |
Layer Encapsulation: Protocol Headers

User A

HTTP request/response
TCP header
IP header
Ethernet header

User B
Goal: Be addressable in ways humans can remember on the Internet
Solution: Domain Names
DNS (Domain Name System)

• Host addresses: e.g., 128.135.11.239
  – a number used by protocols
  – conforms to network structure (the “where”)

• Host names: e.g., super.cs.uchicago.edu
  – usable by humans
  – conforms to organizational structure (the “who”)

• Domain Name System (DNS) is how we map from one to other
  – a directory service for hosts on the Internet
  – See nslookup
Hierarchical Namespace

- “Top Level Domains” are at the top
- Domains are subtrees
  - E.g: .edu, uchicago.edu, cs.uchicago.edu
- Name is leaf-to-root path
  - linux.cs.uchicago.edu
- Name collisions trivially avoided!
  - each domain’s responsibility
Hierarchical Administration

- A zone corresponds to an administrative authority responsible for a contiguous portion of hierarchy
- E.g.: UChicago controls law.uchicago.edu and *.cs.uchicago.edu while CS controls *.cs.uchicago.edu
Political Environment For Domains

• Internet Corporation for Assigned Names and Numbers (ICANN) is a non-profit that controls the assignment of both IP addresses and domain names.
DNS Root Servers


- A Verisign, Dulles, VA
- C Cogent, Herndon, VA
- D U Maryland College Park, MD
- 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
- E NASA Mt View, CA
- F Internet Software Consortium Palo Alto, CA
DNS Root Servers

- All replicated via anycast

A Verisign, Dulles, VA
C Cogent, Herndon, VA (also Los Angeles, NY, Chicago)
D U Maryland College Park, MD
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign (21 locations)
K RIPE London (plus 16 other locations)
I Autonomica, Stockholm (plus 29 other locations)
M WIDE Tokyo plus Seoul, Paris, San Francisco

E NASA Mt View, CA
F Internet Software Consortium, Palo Alto, CA (and 37 other locations)
B USC-ISI Marina del Rey, CA
L ICANN Los Angeles, CA
DNS Records

- DNS servers store Resource Records (RRs)
  - RR is (name, value, type, TTL)
- Type = A: (→ Address)
  - name = hostname
  - value = IP address
- Type = NS: (→ Name Server)
  - name = domain
  - value = name of dns server for domain
- Type = MX: (→ Mail eXchanger)
  - name = domain in email address
  - value = name(s) of mail server(s)
Inserting Resource Records into DNS

- Example: you want “blaseur.com”
- Register blaseur.com at registrar (e.g., GoDaddy)
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts into the .com TLD server who your name servers are
- Store resource records in your server
  - e.g., type A record for www.blaseur.com
  - e.g., type MX record for blaseur.com
DNS client (me.cs.uchicago.edu)

local DNS server (mydns.uchicago.edu)

root servers

.edu servers

.nyu.edu servers
DNS client
(me.cs.uchicago.edu)

DNS server
root servers
.edu servers

local DNS server
(mydns.uchicago.edu)

www.nyu.edu?

DNS client
(me.cs.uchicago.edu)

.root servers
.edu servers

nyu.edu servers
DNS client (me.cs.uchicago.edu)

DNS server (mydns.uchicago.edu)

.edu servers

NYU.edu servers

root DNS server
recursive DNS query

root DNS server

DNS server
(mydns.uchicago.edu)

DNS client
(me.cs.uchicago.edu)

.edu servers

.nyu.edu servers
DNS client (me.cs.uchicago.edu)

DNS server (mydns.uchicago.edu)

.root DNS server

.edu servers

.nyu.edu servers
DNS FAQs

• Do you have to follow that recursive process every time?
  – No (DNS queries are cached)

• Is DNS “secure” / “private”?
  – No

• Have people tried to make DNS secure
  – Yes. See, e.g., DNSSEC, which aims to provide integrity by signing DNS records. These efforts are ongoing!
Now, let’s see how the web works!
Your interface to the web

• Your web browser contacts a web server
A 10,000 Foot View of Technologies

- Where things run:

  - HTML / CSS
  - JavaScript (Angular/React)
  - Browser Extensions
  - Django (Python) / CGI (Perl) / PHP / Node.js / Ruby on Rails
  - Databases (MySQL)
The Anatomy of a URL

- https://www.uchicago.edu/fun/funthings.htm?query=music&year=2022#topsection
The Anatomy of a URL

- **Protocol**: https
- **Hostname**: www.uchicago.edu
  - .edu is the top level domain (TLD)
- **Path**: /fun/funthings.html
- **Parameters**: (key=value pairs, & delimited)
- **Named anchor**: #topsection

Some technologies (e.g., Django) parse the path differently (e.g., parameters are there)
The Anatomy of a Webpage

• view-source:https://www.cs.uchicago.edu/

• HTML (hypertext markup language)
  – Formatting of a page
  – All sorts of formatting: `<div><p>Hi</p></div>`
  – Links: `<a href="blaseur.com">Click here</a>`
  – Pictures: `<img src="unicorn.jpg"`>
  – Forms

• HTML 5 introduced many media elements
The Anatomy of a Webpage