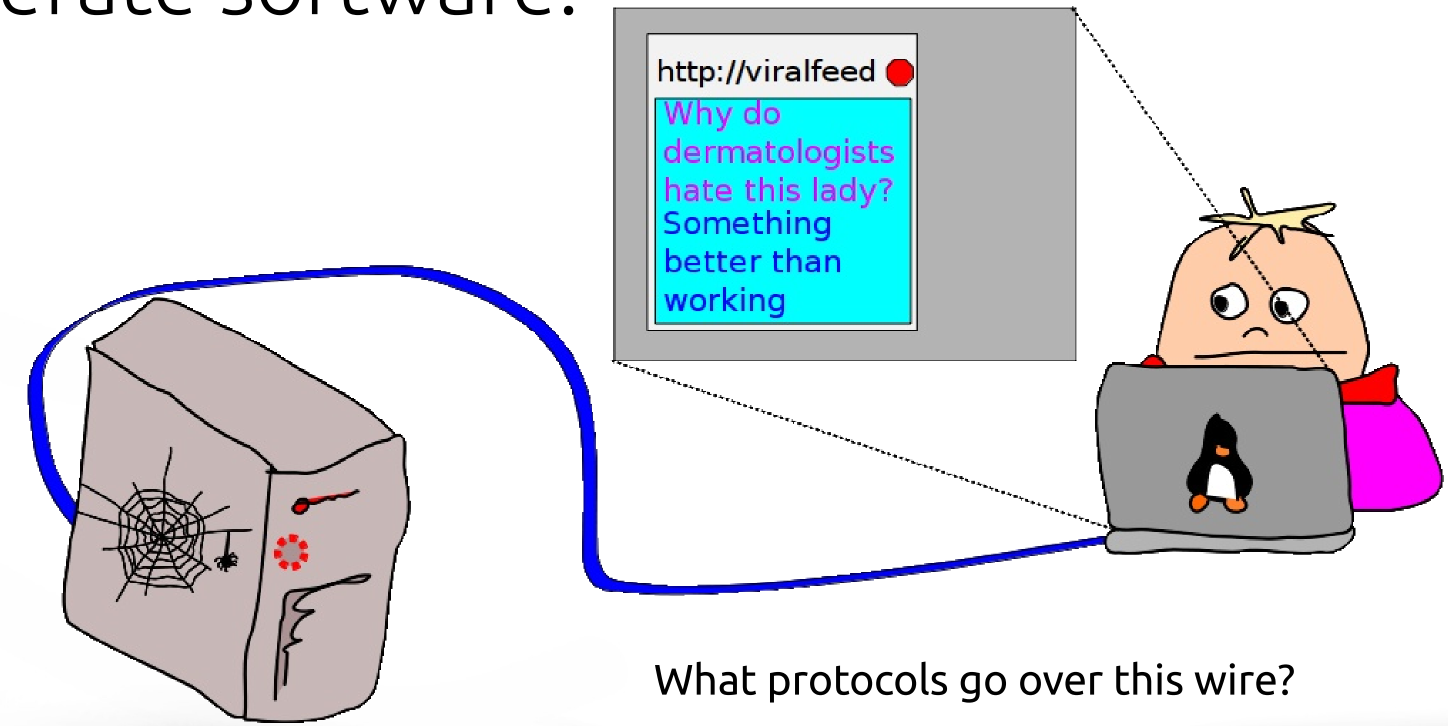
CMPUT 410: The Internet and Stuff

Abram Hindle
abram.hindle@ualberta.ca
Department of Computing Science
University of Alberta
http://softwareprocess.es/

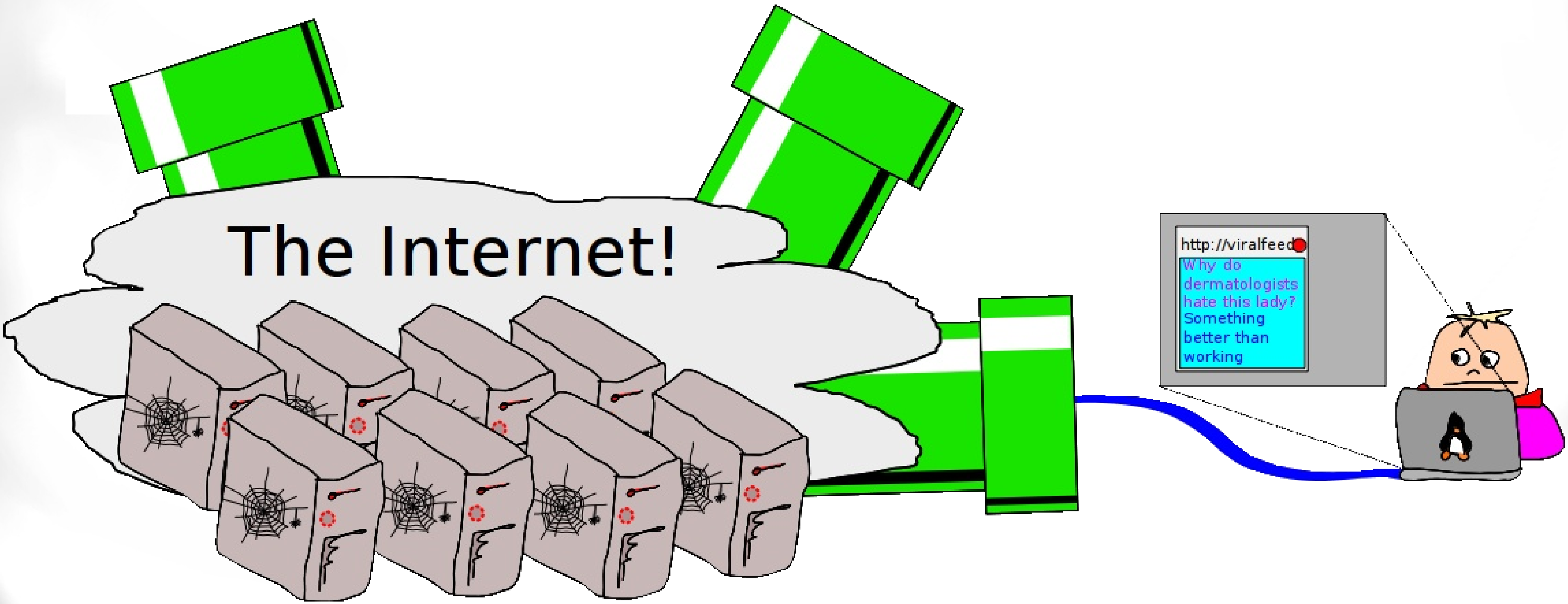
The Simple View of a Web Client and Web Server

- We use the web to request, search, navigate, and share information.
- Furthermore we use the web to access and operate software.



What protocols go over this wire?

But really it is more like this...



What protocols go over these wire?

Ethernet

- 1 Ethernet frame is 64 to 1568 bytes (each field is shown in bytes/octets)

Preamble	SFD	Destination	Source	Length	Payload	CRC
7	1	6	6	2	46-1500	4

- I'm going by the Ethernet spec and not anything fancy, IEEE Std 802.3TM-2012 (Revision of IEEE Std 802.3-2008)
- CRC means each frame has some error detection.
- SFD – Start Frame Delimiter
- Originally used over radio, now used over wires.
- This is not gigabit

Why are ethernet frames important to the Web?

Why are ethernet frames important to the Web?

- Know the minimum packet sizes that can be sent.
- Know the potential waste in a transmission
- Know sizes that aren't fragmented or split
- Know when you'll incur latency due to split packets/frames
- Keep your message sizes smaller than 1.5kb to ensure you stay inside of packets.
- Sending 1 byte, incurs many headers.
- Most people are connected to the internet with something like ethernet and their MTU is 1500 or less.

IPV4

- Ethernet is not routeable
- We need to communicate over large distances
- We need to communicate to many computers
- IP was a compromise to address computers
- Stateless
- Backbone of the internet
- But we're almost out of IP addresses

IPV4 Header

IPv4 Header Format

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><th>0</th><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Length</td></tr>
<tr><th>4</th><th>32</th><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><th>8</th><th>64</th><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><th>12</th><th>96</th><td colspan="32">Source IP Address</td></tr>
<tr><th>16</th><th>128</th><td colspan="32">Destination IP Address</td></tr>
<tr><th>20</th><th>160</th><td colspan="32">Options (if IHL > 5)</td></tr>
</table>

IPV6

- Like IPV4
- More address space
- TCP can fit over top of it.
- So can HTTP
- IPV6 matters because it means your regular expressions for IP addresses as hostnames aren't enough to be IPV6 compatible!

IPV6 Header

Fixed header format

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="8">Traffic Class</td><td colspan="20">Flow Label</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Payload Length</td><td colspan="8">Next Header</td><td colspan="8">Hop Limit</td></tr>
<tr><td>8</td><td>64</td><td colspan="32" rowspan="4">Source Address</td></tr>
<tr><td>12</td><td>96</td></tr>
<tr><td>16</td><td>128</td></tr>
<tr><td>20</td><td>160</td></tr>
<tr><td>24</td><td>192</td><td colspan="32" rowspan="4">Destination Address</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
<tr><td>36</td><td>288</td></tr>
</table>

UDP

- User Datagram Protocol
- User means that user-space applications can use it.
- Provides checksums – some integrity
- Provides port numbers
- Stateless
- Lossy
- No Connections
- No guarantees

UDP

- This header sits on top of IP

UDP Header

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Length</td><td colspan="16">Checksum</td></tr>
</table>

- The data comes after it. It'll be the IP data size minus the UDP header size.
- Checksum is unfortunately optional but includes
- Data and this header and the IP header

DNS

- Domain Name Service
- Allows us to bind a name to another name, IP, or set of IPs.
- **A** records point to an IP
- **CNAME** records point to another name
- Works on IPV6 and IPV4
- Use dig or nslookup to check names

Example Name Record

A 75.119.223.206
NS ns1.dreamhost.com.
NS ns2.dreamhost.com.
NS ns3.dreamhost.com.
anti A 75.119.223.206
anti MX 0 mx1.sub4.homie.mail.dreamhost.com.
anti MX 0 mx2.sub4.homie.mail.dreamhost.com.
_domainkey.anti TXT o=~; r=postmaster@anti.aliz.es
anti.aliz.es._domainkey.anti TXT k=rsa; t=y; p=CENSORED
ftp.anti A 75.119.223.206
lists.anti A 66.33.216.120
openid A 75.119.223.206
ftp.openid A 75.119.223.206
ssh.openid A 75.119.223.206
www.openid A 75.119.223.206

Accessing DNS records

```
hindle1@st-francis:~$ dig aliz.es

; <<>> DiG 9.8.1-P1 <<>> aliz.es
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13485
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL

;; QUESTION SECTION:
;aliz.es.         IN  A

;; ANSWER SECTION:
aliz.es.      14400  IN  A   75.119.223.206

;; AUTHORITY SECTION:
aliz.es.      86255  IN  NS  ns2.dreamhost.com.
aliz.es.      86255  IN  NS  ns1.dreamhost.com.
aliz.es.      86255  IN  NS  ns3.dreamhost.com.

;; ADDITIONAL SECTION:
ns1.dreamhost.com.   147856 IN  A   66.33.206.206
ns2.dreamhost.com.   147856 IN  A   208.96.10.221
ns3.dreamhost.com.   147856 IN  A   66.33.216.216
```

Dig a CNAME

```
hindle1@st-francis:~$ dig beams.softwareprocess.es

; <<>> DiG 9.8.1-P1 <<>> beams.softwareprocess.es
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38853
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;beams.softwareprocess.es. IN   A

;; ANSWER SECTION:
beams.softwareprocess.es. 14400 IN   CNAME    ghs.google.com.
ghs.google.com.       77681    IN   CNAME    ghs.l.google.com.
ghs.l.google.com. 300  IN   A    74.125.25.121

;; AUTHORITY SECTION:
google.com.      11573    IN   NS   ns4.google.com.
google.com.      11573    IN   NS   ns1.google.com.
google.com.      11573    IN   NS   ns3.google.com.
google.com.      11573    IN   NS   ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.       220411   IN   A    216.239.32.10
ns2.google.com.       218363   IN   A    216.239.34.10
ns3.google.com.       218363   IN   A    216.239.36.10
ns4.google.com.       218363   IN   A    216.239.38.10
```

TCP

- Connections
- 3 packet handshake
- Acknowledgement of receiving packets
- In order
- Used by most internet applications
- Used by HTTP

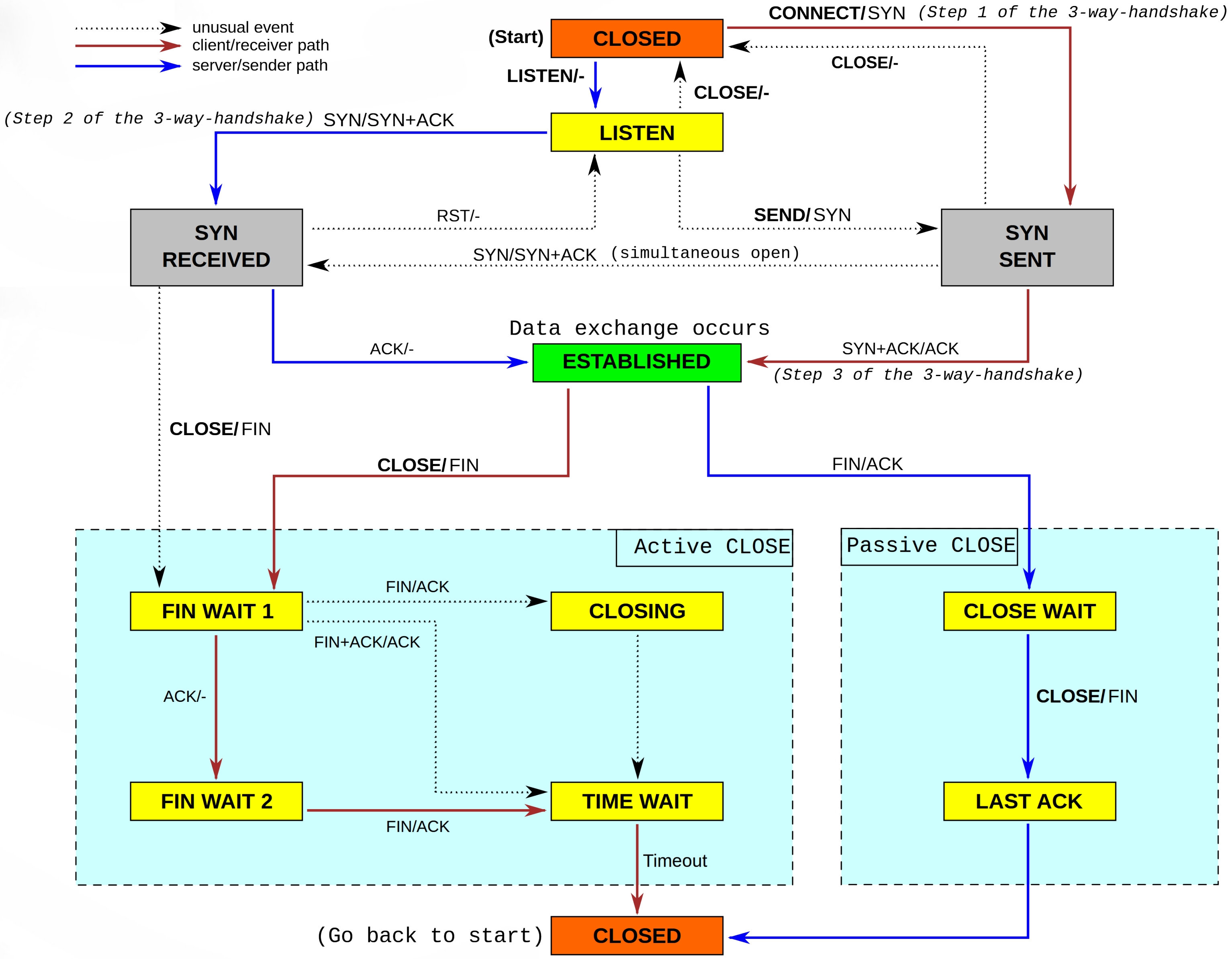
TCP Header

TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

- Pay attention to the size of the header

TCP Connections



Firewalls

- Usually prevent hosts from communicating on certain ports, or hosting services.
- HTTP and firewalls means that webclients are unlikely to be webservers as well. That communication must be initiated by clients rather than webservices.
- IETF seems unaware of their existence but at least HTTP gets through.

Scenario: Get http://slashdot.org

- Context
  - I am at home on a Friday evening. It is 10pm and I haven't been outside all day.
  - I need to read slashdot because I'm bored
  - I have a cable modem internet connection from Shaw.
  - I've connected to the cable modem with CAT5 cables and ethernet.

Scenario: Get http://slashdot.org

- $ GET http://slashdot.org
  - Perl LWP tells the OS to connect to “slashdot.org” via TCP on TCP Port 80
  - The OS looks up slashdot.org and needs to contact a nameserver
    - OS sends a UDP packet on port 53 to the nameserver configured. It asks for the address of slashdot.org
      - This UDP packet is over IP
        - This IP packet is over ethernet
          - Cable modem accepts this packet and forwards it on.

Scenario: Get http://slashdot.org

- Shaw's DNS server receives my UDP packet and doesn't know slashdot.org so it asks a more authoritative server.
  - Sends DNS request over UDP over IP over ethernet to the switch in its datacenter
  - Gets a response back on the UDP port, response contains an A record listing an IP of slashdot.org
  - Shaw's DNS server makes a DNS response packet and sends it back to me over UDP, over IP, over ethernet, over the internet, over their private network, back through my cable modem, back onto ethernet, back on to ethernet, IP, and UDP back to my home computer.

Scenario: Get http://slashdot.org

- My OS receives the DNS response, records the IP address and then initiates a TCP connection to port 80 of the slashdot.org IP.
  - A TCP SYN packet is sent to the slashdot.org IP at port 80, over IP, over ethernet to the cable modem, through shaw and through the internet to slashdot's datacenter where a copy of the packet appears on some ethernet cable, decoded as an IP, TCP connect SYN packet.
  - Slashdot.org sends a TCP SYN+ACK packet back across IP, across ethernet, over their network and internet back to shaw, over shaw's network, to my cable modem, over ethernet, over IP, back to my computer
  - My computer sends a TCP ACK packet back across all the way to slashdot.org through all the prior layers
  - A connection is established!

Scenario: Get http://slashdot.org

- Now that my home computer is connected with slashdot.org over TCP I can send data packets across that TCP connection.
- Perl eventually runs send(ourSlashdotConnection, “GET / HTTP/1.0\r\nHost: slashdot.org\r\n\r\n”);
  - This causes a TCP data packet on the slashdot connection to be made, shuffled off to IP and ethernet, across to cable modem and back all the way to slashdot.org

Scenario: Get http://slashdot.org

- Slashdot's webserver is waiting on the connection and it is reading bytes from the connection. After my packet is delivered to the webserver (over TCP, over IP, over ethernet, over the datacenter network, over the internet, ...) "GET / HTTP/1.0\r\nHost: slashdot.org\r\n\r\n"
  - Slashdot.org's webserver's TCP layers send a TCP ACK packet back to my IP address acknowledging the receipt of the packet that contained the GET request I sent.
  - Slashdot.org's webserver sends an HTTP response which is over 40kb in size broken up across 29 packets. All these packets needs to be acknowledged by my home computer.

So what does it look like?

- 1 UDP DNS Request for slashdot.org
- 1 UDP DNS Response from my nameserver for slashdot.org of 1.2.3.4
- 1 TCP SYN for 1.2.3.4 on port 80
- 1 TCP SYN+ACK from 1.2.3.4 port 80
- 1 TCP ACK to 1.2.3.4 on port 80
- 1 TCP data packet with the GET request to 1.2.3.4
- 1 TCP ACK from 1.2.3.4
- 1 TCP data packet from 1.2.3.4
- 1 TCP ACK to 1.2.3.4
- … 26 data & ACKs later
- 1 TCP data packet from 1.2.3.4
- 1 TCP ACK to 1.2.3.4
- 1 TCP FIN close from 1.2.3.4
- 1 TCP FIN+ACK to 1.2.3.4
- 1 TCP ACK from 1.2.3.4

~ 2 UDP Packets
~ 60 TCP Packets
~ 62 Ethernet packets on my end

The TCP packets are probably copied at least 10 times across 10 or more links.

So my 1 request of 50kb in size could cost the entire network more than 500kb in traffic.

How did we get routed to slashdot?

```
hindle1@piggy:~$ sudo traceroute slashdot.org
traceroute to slashdot.org (216.34.181.45), 30 hops max, 60 byte packets
 1  192.168.0.1 (192.168.0.1)  0.171 ms
 2  * * *
 3                     .ed.shawcable.net (64.59.184.245)  33.812 ms
 4  rc3sc-tge0-0-0-10.wp.shawcable.net (66.163.74.226)  44.058 ms
 5  rc2so-tge0-4-0-1.cg.shawcable.net (66.163.77.98)  77.525 ms
 6  ix-3-3-2-0.tcore1.CT8-Chicago.as6453.net (66.110.14.13)  74.733 ms
 7  64.86.78.10 (64.86.78.10)  70.375 ms
 8  hr1-te-9-0-0.elkgrovech3.savvis.net (204.70.196.14)  74.230 ms
 9  das5-v3032.ch3.savvis.net (64.37.207.158)  71.660 ms
10  64.27.160.194 (64.27.160.194)  83.311 ms
11  slashdot.org (216.34.181.45)  73.920 ms
```

Resources

- Wikipedia
  - http://en.wikipedia.org/wiki/Transmission_Control_Protocol
  - http://en.wikipedia.org/wiki/Transmission_Control_Protocol
  - http://en.wikipedia.org/wiki/Ethernet
  - http://en.wikipedia.org/wiki/802.11
  - http://en.wikipedia.org/wiki/Domain_Name_System
- Ethernet
  - http://standards.ieee.org/about/get/802/802.3.html
  - http://www.osnews.com/story/21132/Metcalfe_on_Ethernet_s_History/
  - http://timeline.ethernethistory.com/
  - http://theinstitute.ieee.org/technology-focus/technology-history/ethernet-turns
  -

Resources

- DNS
  - Domain Names
    - http://tools.ietf.org/html/rfc1035
    - http://tools.ietf.org/html/rfc1123
    - http://tools.ietf.org/html/rfc2181
  - Paul Vixie on DNS
    - http://queue.acm.org/detail.cfm?id=1242499
  - Tools
    - On Unix: nslookup and dig and whois and pwhois
    - http://network-tools.com/nslook/