CcompSci 356 Computer Network Architecture
Lecture 25: Final review

Xiaowei Yang
xwy@cs.duke.edu
What we have learned

• Key components of the Internet

• What happens when data is sent from one host to another
What the Internet looks like

Ethernet, CSMA/CD
Bridges, Switches, Spanning Tree

Bandwidth x Delay
TCP Performance

RIP, OSFP
Distance Vector
Link-State

IP Blocks, CIDR, Subnets
Longest Prefix Match, Fragmentation, MTU

The Internet

TCP Performance

Modulation
Coding
FDMA, TDMA
Functions/Concepts at different layers

- Application protocols vs Applications
- Reliable transport, multiplexing
- Forwarding, Routing, Addressing
- Encoding, framing, error detection, reliability
- Media sharing, switching
- Bandwidth, latency, throughput, delay-bandwidth product
Physical properties of a link

- CPU
- Network adapter
- I/O bus
- Memory
- Cache

To network:
- From network:

Node A  Adaptador  Bits  Adaptador  Node B

Frames
Bandwidth

- Bandwidth of a link refers to the number of bits it can transmit in a unit time
  - A second of time as distance
  - Each bit as a pulse of width

- Fast link
- Slow link
Latency to transmit a packet

• Has four components
  – Link propagation delay
  – Transmission/serialization latency
  – Queuing delay
  – Processing delay (often ignored)
Round trip time (RTT)

- Time to send a packet and receive an acknowledgement
How to determine the “optimal” sliding window size

• Discuss midterm problem 3
• What does “keep the pipe full” mean?
Mechanisms at Different layers

• Link layer
  – Encoding
    • NRZ, NRZI, Manchester, 4B/5B
  – Framing
    • Byte-oriented, bit-oriented, time-based
    • Bit stuffing
  – Error detection
    • Parity, checksum, CRC
  – Reliability
    • FEC, sliding window
Link layer continued

• Multi-access link
  – Ethernet
    • Collision Sense Multiple Access/Collision Detection (CSMA/CD)
  – WIFI
    • Carrier-sense multiple access with collision avoidance (CSMA/CA)
    • Cannot send and receive at the same time
    • Must send when channel is idle
    • RTS/CTS
Link layer continued

• Virtual circuit switching
  – ATM

• Datagram switching
  – Ethernet learning bridges
    • Spanning tree algorithm

• Source routing
The network layer

- The Internet Protocol
- Classless Interdomain Routing (CIDR)
  - Addressing format
  - Subnet, network prefix
- Forwarding
  - Longest prefix matching
The network layer continued

• Routing
  – Distance vector
  – Link state
  – BGP
• Auxiliary functions
  – ARP, ICMP, DHCP, NAT, IP Tunnel
• Multicast
• QoS
The transport layer

• UDP
  – Datagram, connectionless, multiplexing multiple applications

• TCP
  – Reliable, byte stream
TCP

- Connection establishment
- Reliability
  - Sliding window
  - Loss recovery
    - Time out, duplicate acks, selective ACKs
- Flow control
- Congestion control and avoidance
TCP congestion control

1. Probing for the available bandwidth
   - slow start (cwnd < ssthresh)

2. Avoid overloading the network
   - congestion avoidance (cwnd >= ssthresh)
Slow Start

• **Initial value:**  Set `cwnd = 1 MSS`
  • Modern TCP implementation may set initial `cwnd` to 2

• **When receiving an ACK, `cwnd += 1 MSS`**
  • If an ACK acknowledges two segments, `cwnd` is still increased by only 1 segment.
  • Even if ACK acknowledges a segment that is smaller than MSS bytes long, `cwnd` is increased by 1.
  • Question: how can you accelerate your TCP download?
Congestion Avoidance

• If $cwnd \geq ssthresh$ then each time an ACK is received, increment $cwnd$ as follows:
  • $cwnd += MSS \times (MSS / cwnd)$ ($cwnd$ measured in bytes)

• So $cwnd$ is increased by one MSS only if all $cwnd$/$MSS$ segments have been acknowledged.
Slow Start

• Initial value: Set $cwnd = 1$ MSS
  • Modern TCP implementation may set initial $cwnd$ to 2

• When receiving an ACK, $cwnd += 1$ MSS
  • If an ACK acknowledges two segments, $cwnd$ is still increased by only 1 segment.
  • Even if ACK acknowledges a segment that is smaller than MSS bytes long, $cwnd$ is increased by 1.
  • Question: how can you accelerate your TCP download?
Congestion Avoidance

• If $cwnd \geq ssthresh$ then each time an ACK is received, increment $cwnd$ as follows:
  • $cwnd += MSS \times (MSS / cwnd)$ ($cwnd$ measured in bytes)

• So $cwnd$ is increased by one MSS only if all $cwnd$/MSS segments have been acknowledged.
TCP congestion control modeling

\[ x_i(t+1) = \begin{cases} 
  a_i + b_i x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase}, \\
  a_D + b_D x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.}
\end{cases} \]

- Four sample types of controls
- AIAD, AIMD, MIAD, MIMD
Phase plot
Application layer

• Domain Name System (DNS)
  – Problem 5 of homework

• Socket interface

• Application protocols vs applications
  – Email

• Security primitives
Figure out the DNS server hierarchy

- `dig +norecurse @a.root-servers.net NS www.cnn.com`
- https://ns1.com/articles/using-dig-trace
- `dig +trace www.cnn.com`
- `dig +trace turner-tls.map.fastly.net`. 
Other examples

- Sample final problem 4
- Midterm problem 2
Looking forward

• Graduate networking class
  – Datacenter networking
  – Future Internet architectures
What to expect in the final

• Networking knowledge
  – Understanding

• Application of networking knowledge
Course evaluation

• Please do it if you have not!
An Example
A simple TCP/IP Example

• A user on host argon.tcpip-lab.edu (“Argon”) makes web access to URL


• What actually happens in the network?
HTTP Request and HTTP response

- Web server runs an HTTP server program
- HTTP client Web browser runs an HTTP client program
- sends an HTTP request to HTTP server
- HTTP server responds with HTTP response
GET /example.html HTTP/1.1
Accept: image/gif, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: 192.168.123.144
Connection: Keep-Alive
HTTP Response

HTTP/1.1 200 OK
Date: Sat, 25 May 2002 21:10:32 GMT
Server: Apache/1.3.19 (Unix)
Last-Modified: Sat, 25 May 2002 20:51:33 GMT
ETag: "56497-51-3ceff955"
Accept-Ranges: bytes
Content-Length: 81
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<HTML>
<BODY>
<H1>Internet Lab</H1>
Click <a href="http://www.tcpip-lab.net/index.html">here</a> for the Internet Lab webpage.
</BODY>
</HTML>

• How does the HTTP request get from Argon to Neon?
From HTTP to TCP

- To send request, HTTP client program establishes an TCP connection to the HTTP server Neon.
- The HTTP server at Neon has a TCP server running
Resolving hostnames and port numbers

• Since TCP does not work with hostnames and also would not know how to find the HTTP server program at Neon, two things must happen:
  
  1. The name “neon.tcpi-lab.edu” must be translated into a 32-bit IP address.
  
  2. The HTTP server at Neon must be identified by a 16-bit port number.
Translating a hostname into an IP address

- The translation of the hostname *neon.tcpip-lab.edu* into an IP address is done via a database lookup
  - `gethostbyname(host)`
- The distributed database used is called the *Domain Name System (DNS)*
- All machines on the Internet have an IP address:
  
<table>
<thead>
<tr>
<th>IP Address</th>
<th>hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.143.136.15</td>
<td>argon.tcpip-lab.edu</td>
</tr>
<tr>
<td>128.143.71.21</td>
<td>neon.tcpip-lab.edu</td>
</tr>
</tbody>
</table>
Finding the port number

- **Note:** Most services on the Internet are reachable via well-known ports. E.g. All HTTP servers on the Internet can be reached at port number “80”.
- **So:** Argon simply knows the port number of the HTTP server at a remote machine.
- On most Unix systems, the well-known ports are listed in a file with name `/etc/services`. The well-known port numbers of some of the most popular services are:

<table>
<thead>
<tr>
<th>Service</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>21</td>
</tr>
<tr>
<td>telnet</td>
<td>23</td>
</tr>
<tr>
<td>smtp</td>
<td>25</td>
</tr>
<tr>
<td>finger</td>
<td>79</td>
</tr>
<tr>
<td>http</td>
<td>80</td>
</tr>
<tr>
<td>nntp</td>
<td>119</td>
</tr>
</tbody>
</table>
The HTTP client at argon.tcpip-lab.edu requests the TCP client to establish a connection to port 80 of the machine with address 128.143.71.21.

connect(s, (struct sockaddr*)&sin, sizeof(sin))

The HTTP client at argon.tcpip-lab.edu requests the TCP client to establish a connection to port 80 of the machine with address 128.141.71.21.
The TCP client at Argon sends a request to establish a connection to port 80 at Neon.

This is done by asking its local IP module to send an IP datagram to 128.143.71.21.

(The data portion of the IP datagram contains the request to open a connection)
Sending the IP datagram to the default router

- *Argon* sends the IP datagram to its default router
- The default gateway is an IP router
- The default gateway for *Argon* is *Router137.tcpip-lab.edu* (128.143.137.1).
Invoking the device driver

The IP module at Argon tells its Ethernet device driver to send an Ethernet frame to address 00:e0:f9:23:a8:20

- The IP module at Argon tells its Ethernet device driver to send an Ethernet frame to address 00:e0:f9:23:a8:20
- Ethernet address of the default router is found out via ARP
The route from *Argon* to *Neon*

- Note that the router has a different name for each of its interfaces.
Sending an Ethernet frame

- The Ethernet device driver of *Argon* sends the Ethernet frame to the Ethernet network interface card (NIC)
- The NIC sends the frame onto the wire
Forwarding the IP datagram

- The IP router receives the Ethernet frame at interface 128.143.137.1
  1. recovers the IP datagram
  2. determines that the IP datagram should be forwarded to the interface with name 128.143.71.1
- The IP router determines that it can deliver the IP datagram directly
The IP protocol at \textit{Router71}, tells its Ethernet device driver to send an \textit{Ethernet frame} to address \texttt{00:20:af:03:98:28}

- The IP protocol at \textit{Router71}, tells its Ethernet device driver to send an \textit{Ethernet frame} to address \texttt{00:20:af:03:98:28}
Sending another Ethernet frame

• The Ethernet device driver of Router71 sends the Ethernet frame to the Ethernet NIC, which transmits the frame onto the wire.
Data has arrived at Neon

- *Neon* receives the Ethernet frame
- The payload of the Ethernet frame is an IP datagram which is passed to the IP protocol.
- The payload of the IP datagram is a TCP segment, which is passed to the TCP server