Administrivia

- Later today: Look for message about IP grading
  - Meeting slots first week after break (and during break)
- TCP: Draft of assignment out today
  - Read it over before break, start when we get back
- Summer/UTA hiring: Expect a message from me today/tomorrow
Today

Light overview of the transport layer and TCP

– Why we need TCP
– What components are involved
– What you will do in the project
Transport Layer

- Transport protocols sit on top of network layer
- Problem solved: communication among processes
  - Application-level multiplexing (“ports”)
  - Error detection, reliability, etc.
From Lec 2: OSI Reference Model

Link-Layer Protocol

End host

Application
Presentation
Session
Transport
Network
Data link
Physical

End host

Application
Presentation
Session
Transport
Network
Data link
Physical

One or more nodes within the network
From Lec 2: OSI Reference Model

- **Network Protocol**
- **Link-Layer Protocol**

One or more nodes within the network
From Lec 2: OSI Reference Model

End host

Application Protocol

Application
Presentation
Session
Transport
Network
Data link
Physical

Transport Protocol

Network Protocol

Network
Data link
Physical

Link-Layer Protocol

Data link
Physical

One or more nodes within the network
User Datagram Protocol
• Unreliable datagram service
• Adds multiplexing (via ports) and nothing else
• Checksum is pretty useless
Next Problem: Reliability

We talked briefly about link-layer reliability:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropped Packets</td>
<td>Acknowledgments + Timeout</td>
</tr>
<tr>
<td>Duplicate Packets</td>
<td>Sequence Numbers</td>
</tr>
<tr>
<td>Packets out of order</td>
<td>Receiver Window</td>
</tr>
<tr>
<td>Maximizing throughput</td>
<td>Sliding Window (Pipelining)</td>
</tr>
</tbody>
</table>
Next Problem: Reliability

We talked briefly about link-layer reliability:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropped Packets</td>
<td>Acknowledgments + Timeout</td>
</tr>
<tr>
<td>Duplicate Packets</td>
<td>Sequence Numbers</td>
</tr>
<tr>
<td>Packets out of order</td>
<td>Receiver Window</td>
</tr>
<tr>
<td>Maximizing throughput</td>
<td>Sliding Window (Pipelining)</td>
</tr>
</tbody>
</table>

- Single link: things were easy... 😊
Transport Layer Reliability

• Extra difficulties
  – Multiple hosts
  – Multiple hops
  – Multiple potential paths
Transport Layer Reliability

• Extra difficulties
  – Multiple hosts
  – Multiple hops
  – Multiple potential paths

• What does this mean?
  – Multiple opportunities for failure
  – Hosts have different resources
  – Varying RTTs
Extra Difficulties (cont.)

• Out of order packets
  – Not only because of drops/retransmissions
  – Can get very old packets (up to 120s), must not get confused
Extra Difficulties (cont.)

• Out of order packets
  – Not only because of drops/retransmissions
  – Can get very old packets (up to 120s), must not get confused

• Unknown resources at other end
  – Must be able to discover receiver buffer: flow control

⇒ Tell Sender to stop/limit data
Extra Difficulties (cont.)

• Out of order packets
  – Not only because of drops/retransmissions
  – Can get very old packets (up to 120s), must not get confused

• Unknown resources at other end
  – Must be able to discover receiver buffer: flow control

• Unknown resources in the network
  – Should not overload the network
  – But should use as much as safely possible to maximize throughput

⇒ "Fairness in use of scarce BW"
TCP – Transmission Control Protocol

A

APP

SEND DATA TO TCP

TCP STACK

SEND BUFFER

SEND SEGMENTS

SEGS

#1

#2

#3

#4

TCP STACK

(RKERNEL)

(YOUR NODE)

SEND ACKS

B

APP

PROVIDES TO HOST ON RCV,
TCP – Transmission Control Protocol
TCP – Transmission Control Protocol

- Service model: “reliable, connection oriented, full duplex ordered byte stream”
TCP – Transmission Control Protocol

- Service model: “reliable, connection oriented, full duplex ordered byte stream”
- Flow control: If one end stops reading, writes at other eventually stop/fail
TCP – Transmission Control Protocol

- **Service model**: “reliable, connection oriented, full duplex ordered byte stream”
- **Flow control**: If one end stops reading, writes at other eventually stop/fail
- **Congestion control**: Keeps sender from overloading the network
TCP

• Specification
  – RFC 793 (1981), RFC 1222 (1989, some corrections), RFC 5681 (2009, congestion control), ...

• Was born coupled with IP, later factored out

• End-to-end protocol
  – Minimal assumptions on the network
  – All mechanisms run on the end points

• What if you had link-layer reliability instead?
Why not provide X on the network layer?

X = Reliability, security, message ordering…

• Cost
  – These functionalities are not free: don’t burden those who don’t need them
Why not provide X on the network layer?

X = Reliability, security, message ordering...

• Cost
  – These functionalities are not free: don’t burden those who don’t need them

• Conflicting
  – Timeliness and in-order delivery, for example
Why not provide X on the network layer?

X = Reliability, security, message ordering…

• Cost
  – These functionalities are not free: don’t burden those who don’t need them

• Conflicting
  – Timeliness and in-order delivery, for example

• Insufficient
  – Example: reliability
End-to-end argument

• Functions placed at lower levels of a system may be redundant or of little value
  – They may need to be performed at a higher layer anyway
• But they may be justified for performance reasons
End-to-end argument

- Functions placed at lower levels of a system may be redundant or of little value
  - They may need to be performed at a higher layer anyway
- But they may be justified for performance reasons
  - Or just because they provide most of what is needed
  - Example: retransmissions
- Takeaway: weigh the costs and benefits at each layer
TCP Header

WHERE THIS PACKET IS IN STREAM
LAST SEQUENCE RECEIVED
HOW MUCH SPACE THE RECEIVER HAS FOR DATA

FLAGS
Header Fields

- **Ports**: multiplexing
- **Sequence number**
  - Correspond to bytes, not packets!
- **Acknowledgment Number**
  - Next expected sequence number
- **Window**: willing to receive
  - Lets receiver limit SWS (even to 0) for flow control
- **Data Offset**: # of 4 byte (header + option bytes)
- **Flags, Checksum, Urgent Pointer**
Header Flags

- **URG**: whether there is urgent data
- **ACK**: ack no. valid (all but first segment)
- **PSH**: push data to the application immediately
- **RST**: reset connection
- **SYN**: synchronize, establishes connection
- **FIN**: close connection
ESTABLISHING A TCP CONNECTION

CLIENT

CONNECT()  
SYN-SENT

SYN  
SEG = X

SYN + ACK  
SEG = Y  
ACK = X + 1

ACK  
ACK = Y + 1

ESTABLISHED

SERVER

(listening on some port)

SYN-REVD

3-WAY HANDSHAKE

- SETS UP CONNECTION
- AGREE ON SEQ NUMBERS

ACCEPT() RETURNS HERE
Establishing a Connection

- Three-way handshake
  - Two sides agree on respective initial sequence nums
- If no one is listening on port: server sends RST
- If server is overloaded: ignore SYN
- If no SYN-ACK: retry, timeout
Connection Termination

- FIN bit says no more data to send
  - Caused by close or shutdown
  - Both sides must send FIN to close a connection

- Typical close
Summary of TCP States
Summary of TCP States

Connection Establishment

- Passive open
- Active open/SYN
- Close
- Send/SYN
- SYN/SYN + ACK
- SYN/SYN + ACK
- SYN + ACK/ACK
- ACK
- FIN/ACK
- Close/FIN
- Close/FIN
- TIME_WAIT
- FIN/ACK
- ACK
- FIN/ACK
- ACK + FIN/ACK
Summary of TCP States

Active close:
Can still receive
Summary of TCP States

**Passive close:** Can still send!

**Active close:** Can still receive

- **CLOSE_WAIT**
  - Close/FIN
  - FIN/ACK
  - Timeout after two segment lifetimes

- **LAST_ACK**
  - ACK

- **CLOSING**
  - ACK

- **FIN_WAIT_2**
  - ACK
  - ACK + FIN/ACK
  - FIN/ACK

- **FIN_WAIT_1**
  - Close/FIN
  - FIN/ACK

- **ESTABLISHED**
  - Close/FIN
  - FIN/ACK

- **SYN_SENT**
  - SYN + ACK/ACK

- **SYN_RCVD**
  - SYN/SYN + ACK
  - SYN/SYN + ACK

- **LISTEN**
  - Close
  - Send/SYN

- **CLOSED**
  - Active open/SYN
  - Passive open

- **Connection Establishment**
Next class

- Sending data over TCP