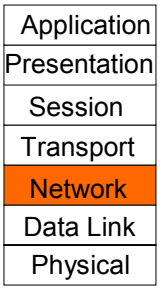
CSE/EE 461 – Module 10

Introduction to the Transport Layer

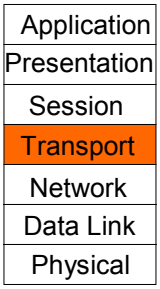
Last Time

- We finished up the Network layer
  - Internetworks (IP)
  - Routing (DV/RIP, LS/OSPF, BGP)
- It was all about routing: how to provide end-to-end delivery of packets.



This Time

- We begin on the Transport layer
- Focus
  - Process-to-process communication
    - Fast?
    - Reliable?
  - Impact on the network
    - Congestion control
- Topics
  - The Transport layer
  - Acknowledgements and retransmissions (ARQ)
  - Sliding windows



M10.3

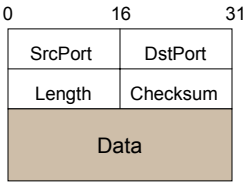
The Transport Layer

- Builds on the services of the Network layer
  - "TCP / IP"
- Communication between processes running on hosts
  - Naming / Addressing
- Stronger guarantees of message delivery make sense
  - This is the first layer that is talking "end-to-end"

M10.4

Internet Transport Protocols

- UDP
  - Datagram abstraction between processes
  - With error detection



- TCP
  - Bytestream abstraction between processes
  - With reliability
  - Plus congestion control (later!)

M10.5

UDP/IP Properties (User Datagram Protocol)

UDP
- Datagram oriented
- Lost packets
- Reordered packets
- Duplicate packets
- Limited size packets

IP
- Datagram oriented
- Lost packets
- Reordered packets
- Duplicate packets
- Limited size packets

M10.6

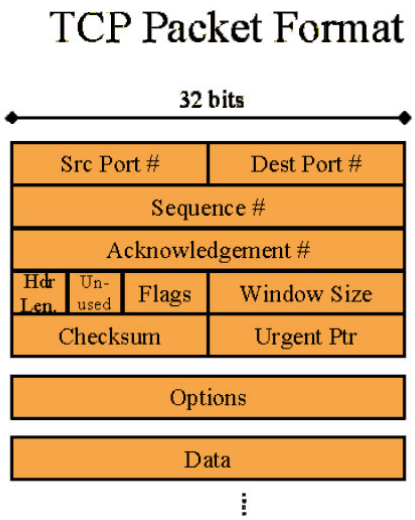
TCP/IP Properties (Transmission Control Protocol)

TCP
- Connection-oriented
- Multiple processes
- Reliable byte-stream delivery
  - In-order delivery
  - Single delivery
  - Arbitrarily long messages
- Synchronization
- Flow control
- Reliable delivery

IP
- Datagram oriented
- Lost packets
- Reordered packets
- Duplicate packets
- Limited size packets

M10.7

TCP Packet Format



16 bit window size gets Cramped with large Bandwidth x delay

16 bits --> 64K
BD ethernet: 122KB
STS24 (1.2Gb/s): 14.8MB

32 bit sequence number must not wrap around faster than the maximum packet lifetime. (120 seconds)
-- 622Mb/s link: 55 seconds

M10.8

TCP End-to-End Properties

- TCP provides a full-duplex connection
  - Each side of a connection can send to the other
- Connection is a stream
  - Packet boundaries may not be visible to application
- Sliding window
  - Endpoints exchange window sizes
  - Packets carry sequence numbers
    - Actually, byte counts in the connection stream
  - Performance
  - Reliability (ARQ)

M10.9

End-to-end Properties

- Performance
  - Sliding Window
    - Try to enable sender to put bandwidth *x* delay product bytes on the wire
- Reliability
  - Lost packets?
    - Sliding window performs flow control
    - Sliding window performs ARQ (Automatic Repeat Request)
  - Duplicate / out-of-order packets?
    - Sliding window receive (re-order) buffer

M10.10

Network Property: Congestion Control

- TCP also implements congestion control
  - High level goal: keep from over-loading the bottleneck network link
  - Immediate goal: find the fastest transmission rate that doesn't overload the bottleneck
- Does it make sense to put congestion control in TCP?
  - Could it be in some other layer?
    - Would it make sense to apply it to UDP?
- Another goal: fairness
  - I'm not slowing down, you slow down…

M10.11

TCP / UDP comparison

TCP	UDP
Reliable	Unreliable
Stream-oriented	Packet-oriented
Connection	Connectionless

M10.12

TCP / UDP comparison

- Stream- vs. packet-oriented
  - Visible packet boundaries can act as "end of record" indicators to application
  - In a stream, if the application wants the notion of "records", it must embed them in the data
    - Example: lines in a text file
  - Since TCP doesn't know about app record boundaries, reading records can be cumbersome
    - Each read() operation returns whatever data happens to have arrived in the stream to this point

M10.13

TCP / UDP comparison

- Connection vs. connectionless
  - UDP: "flexible" (or "you don't know who you're talking with")
    - Incoming data can be from anywhere
    - Outgoing data can go anywhere
    - *(Java API provides a connect() interface – filters packets before returning them to app)*
  - TCP: incoming/outgoing packets are separated into "flows"
    - Provides a nice programming abstraction for many apps
    - How do I open a connection?
    - How do I close one?
    - How do I know when the other side has stopped listening/sending

M10.14

HW4

- Out tonight (probably)
- Option to do it alone or in pairs
- Java (1.4) programming
- "Real networking"
  - Sockets and the like
  - Packets travel over the Internet (not just locally)
    - Planetlab
- Real networking ⇒ concurrency ⇒ threads
- Real networking ⇒ lots of "system calls"
- All humanly possible effort has been made to minimize the frustration factor
  - Still, MUCH less contained potential problems than the last time

M10.15

TCP / UDP / HW4 comparison

TCP	UDP	UW461
Reliable	Unreliable	Reliable
Stream-oriented	Packet-oriented	Packet-oriented
Connection	Connectionless	*Connection-like*
Full-duplex	N-to-1 in, 1-to-N out	*In theory, N-1 and 1-N* *In practice, 1-1 in one direction*

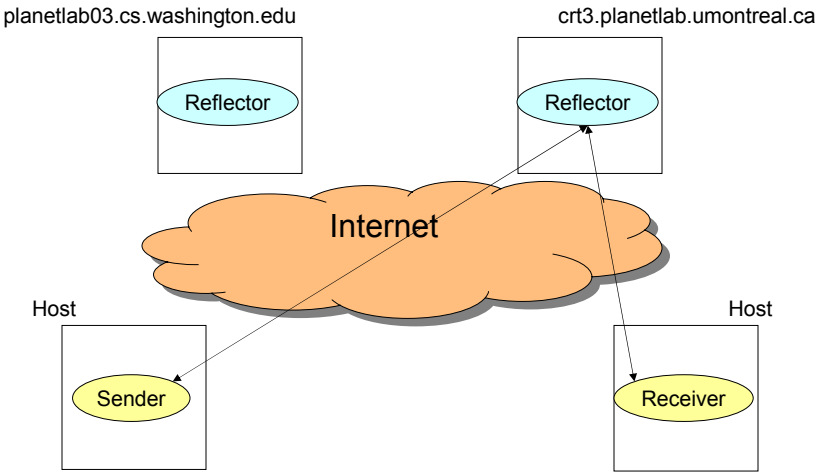
M10.16

UW461 Transport

- Goals:
  - Design and test schemes for achieving high bandwidth, reliable transfers
  - Test those schemes using the Internet as a testbed
  - Keep the programming effort required under control
  - Together those goals led to the somewhat odd properties of this transport ("connection-like")
  - Plausible schemes are not limited to what TCP does
    - Do we care about fairness?
    - Do we care about congestion control?
    - One sample program simply blasts UDP packets at the receiver
      - Plenty fast; not very reliable

M10.17

The Testbed



M10.18

A Transport, Not a File Transfer Application

- The only app your transport has to run correctly does a file transfer, but…
- There is no application-level protocol:
  - Receiver doesn't know how much data to expect
  - Receiver doesn't know the name of the file
  - Receiver doesn't verify to sender that it has managed to write the file to disk
  - Etc…
- None of that is part of this assignment!

M10.19