Transport Layer Overview

- Tasks performed by the transport layer
  - Services provided to the application layer
  - Services expected from the network layer
- Multiplexing and demultiplexing
- Error checking - the checksum
- Connection management
- Reliability
Transport Layer Tasks

- The transport layer (TCP) provides reliability over an unreliable network

- What can go wrong?
  - Bit errors
    - original data as well as ACKs
  - Lossy channel (with bit errors)
    - Stop-and-wait v. pipelining
  - Out-of-order packets
  - Noticeable delay

The Actual Transport Layer

- Basic transport layer services:
  - Connection management
  - Reliable data transfer
  - Multiplexing/demultiplexing
  - Some error checking
  - Flow control & Congestion control

- Services not available:
  - delay guarantees
  - bandwidth guarantees
  - security

- Internet transport protocols:
  - UDP: Connectionless transport
  - TCP: Connection-oriented transport & Reliability
Multiplexing/demultiplexing

- Multiplexer
  - Selects input from one of many input lines (processes) and directs the information to a single output line
  - Many sockets to one network connection

- Demultiplexer
  - Direct a single input to one of many possible processes that are running
  - Single network connection to many sockets (processes)

Multiplexing/demultiplexing

- Multiplexing at sender:
  - handle data from multiple sockets
  - add transport header (later used for demultiplexing)

- Demultiplexing at receiver:
  - use header info to deliver received segments to correct socket
Connectionless demultiplexing

UDP socket is bound to the local host port #

recall: when creating datagram to send into a UDP socket, the socket must specify
- destination IP address
- destination port #

when host receives UDP segment:
1) check destination port # in segment header
2) direct UDP segment to socket with that port #

IP datagrams with same dest. port #, but different source IP addresses and/or source port numbers will be directed to same socket at destination
**UDP: User Datagram Protocol**

- UDP is a “best effort” service. Segments may be:
  - lost
  - delivered out of order

**SO why is there a UDP?**

- Better control over what is sent and when
- Simple: no connection state at sender, receiver
- Fast(er):
  - no connection establishment (can add delay)
  - small segment header
  - no congestion control: UDP can blast away as fast as desired

**Connection-oriented demux: example**

Three segments all destined to IP address: B, dest port: 80 are demultiplexed to different sockets
Connection-oriented demux

- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- demux: receiver uses all four values to direct segment to appropriate socket
- server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

TCP Socket & Segment

- **TCP:** Server host has simultaneous TCP sockets, one for each connection:
  - each socket identified by its own 4-tuple
- **TCP segment** includes data, and source & destination port and IP addresses (+ length & checksum)
Error Checking: Checksum

* Practice in HW * - straightforward calculation

**Goal:** detect “errors” (e.g., flipped bits) in transmitted segment

**Sender:**
- treat segment contents as sequence of 16-bit integers
- checksum: 1’s complement of the sum of (16-bit) segment contents
- sender puts checksum value into UDP checksum field

**Receiver:**
- compute checksum of received segment - including the sender’s checksum 16-bit word in the sum
- If receiver’s sum is all ‘1’s then there were no errors (probably)
  - If a bit is 0 then the packet has errors

Internet Checksum Example

- **Note**
  - When adding numbers, a carryout from the most significant bit needs to be added to the result, for 1’s complement
- **Example:** add two 16-bit integers

```
  1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0
  1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
  _____________________________
  1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1

  wraparound 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1

  sum 1 0 1 1 1 0 1 1 1 0 1 1 1 0 0 0
  checksum 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1
```

16
Summary

- Transport layer services
  - Desired services
  - Actual protocol services
  - What can go wrong?
- Multiplexing and demultiplexing
- Connection Management
- Error checking - checksum
  - Transport layer provides end-to-end error checking v. link layer single link error checking