Transport Layer

Jasleen Kaur

Fall 2016

What Issues Will We Focus On?

- Basic functionality:
  - Provide a “process-to-process” communication channel
    - Versus the host-to-host abstraction provided by the lower layers

- Wish-list from such a channel:
  - Guarantees message delivery
  - Guarantees in-order delivery
  - Guarantees no duplicate messages
  - Supports arbitrarily large messages
  - Guarantees bounded delay
    - Helps support synchronization between the sender and the receiver
  - Allows receiver to control data flow from sender
  - Supports multiple application processes on each host
  - Security

How to provide these services on top of IP’s best-effort service?
User Datagram Protocol (UDP)

- Simplest possible service: multiplexing/demultiplexing
  - Just extend from host-to-host to process-to-process communication
  - Simply add a level of de-multiplexing
    - Since there may be several processes running on a host

- How to identify processes?
  - Process IDs
    - OS-dependent (will work only in networks where everyone runs same OS)
  - Ports/mailboxes:
    - Abstract locator for sending messages to and receiving messages from
    - Allows multiple channels to be established in same process
    - Host-local scope: host ID + port number uniquely identify a channel

- How to learn of destination port number?
  - Servers use well-known port numbers (or port-mapper service)

User Datagram Protocol: Segment Format

- 16-bit port identifiers used
  - Considered enough for host-local scope

- Checksum is optional
  - Same algo as IP checksum (sum of 16-bit words)
  - Computed on the “pseudo-header” + UDPheader + Data
    - Pseudo-header: (protocol, srcIP, destIP, UDPlength) fields
    - Includes fields already included in the IP checksum
Transmission Control Protocol (TCP)

- **Connection-oriented, full-duplex, byte-stream service**
  - Application writes bytes
  - TCP sends segments
  - Application read bytes

- **Service offered:**
  - Reliable delivery
  - In-order delivery
  - Receiver-limited flow control
    - An end-to-end issue
  - Congestion control
    - Host-to-network interaction

TCP: Segment Format

- Ports + IP addresses uniquely identify a TCP connection (what about UDP?)
- Each byte has a sequence number (why?)
- ACK, AdvWin carry info about data sent in opposite direction
- Header length in units of 32 bit words
- Flags used for control information (SYN, FIN, ACK, RESET, URG, PUSH)
**TCP: Connection Establishment**

- **Three-way Handshake:**
  - Two sides agree on starting sequence numbers to use
    - Why not start from 0?
  - SequenceNum = Next sequence expected

`Active participant (client)`

`Passive participant (server)`

- `SYN`, `SequenceNum = x`
- `SYN + ACK`, `SequenceNum = y`, `Acknowledgment = x + 1`
- `ACK`, `Acknowledgment = y + 1`

---

**TCP: State Transition Diagram**

- Why incur 1-RTT setup delay?
- Why explicit teardown, rather than simply timing out?