# CSE and EE 461
# Lecture 23

Autumn Quarter 1997



## Transport Layer

- Services provided to the transport layer from the network layer
  - virtual circuit service (potentially with quality of service options)
  - datagram service
    - connectionless
    - lossy
    - non-fifo
  - network service given and cannot control it



## Services Provided by the Transport Layer

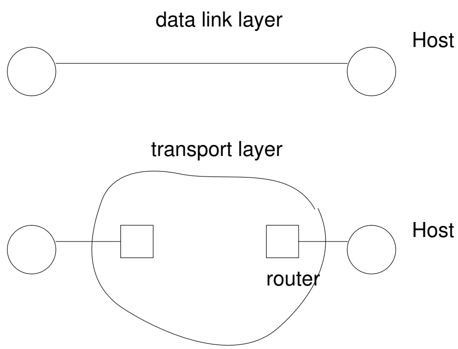
- Level of service - Byte stream
  - reliable (TCP is reliable, UDP is not)
  - timely (Satisfy delay constraints or not)
- Transport Layer Addressing - TSAP
  - Addresses typically found within the operating system of a host. Thus, a TSAP address would be hierarchical
    <country><network><host><port>
  - Example is Unix "sockets." Uses IP address plus port number.
- In internetworking the transport layer does not have to be universal. There may be several options depending on services desired.



## Transport Layer Protocols

- Connection management
  - multiplexing many connections
  - making sure a packet from an old connection does not get in the stream of a new connection.
- Dividing long messages into network packets
- Error control
  - reassemble messages correctly upon arrival - packets can arrive out of order.
  - Setting the timeout period properly
- Flow control
  - both sender and receiver have limited buffer space for packets
- Deal with network failure

## Data Link vs. Transport






## Connection Control

- Similar to opening and closing a file, but complications arise.
- The attempt to open does not elicit a response

  - is the open packet lost or is the receiver down?

  - Early timeouts should not cause two openings to happen.
- Can closing be mutually agreed upon? Theoretically no, but for practical purposes sort of.
- How do you reopen when party has crashed and the other has not.
- How do you choose sequence numbers for reliable transmission.

  - some sequence numbers might not be safe to use because they were recently used in a previous connection.



## Flow Control

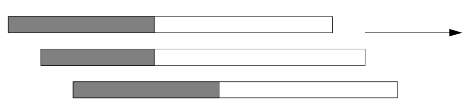
- For reliable transmission

  - Use a sliding window like protocol with selective repeat.

  - With a byte stream protocol an acknowledgment does not necessarily mean that the sender should send more data. The received data may not have been consumed by the application yet.
- For timely transmission

  Assume the sender transmits at a relatively steady rate of x bytes per second. We want the receiver to deliver the data at the same rate with some delay, but with possible gaps.

receiver's buffer space is a window sliding at x bytes per second

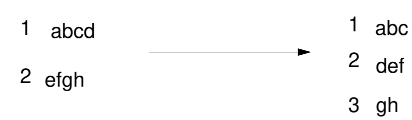




## TCP Service Model

- Connection oriented, reliable, process to process, byte stream protocol
- Using basic primitives like socket, listen, accept, connect, bind

  - a local socket is established that can be one end of a connection.
  - a duplex connection is established between the two end points
  - the connection is identified by a (socket1, socket2) pair.
- In TCP a socket address is an IP address and a Port Number
- Once the connection is established each processes can use primitives like send and receive to send bytes to the other.
- Bytes sent from one socket to the other are guaranteed to get there in the correct order, but are not guaranteed to arrive in the groups sent.



## TCP Segments

- TCP calls its packets "segments" which contain a header and some number of bytes, possible zero bytes. There is a maximum size segment agreed upon by two ends, but never exceeds about 65,000 bytes so it will fit in an IP packet.
- Segment header

  - source and destination port (2 bytes each)
  - header length in 4 byte units (3 bits)
  - sequence number - every byte has a sequence number (4 bytes)
  - ack number - ack is for the next byte expected (4 bytes)
  - window size - how many bytes of storage available (2 bytes)
  - checksum - checksums the header plus data (2 bytes)
  - urgent pointer - offset from current data that is urgent (2 bytes)
  - options (variable length)
  - control bits -
    URG, ACK, PSH, RST, SYN, FIN



## TCP Connection Management

- Server

  - Listen - set up a queue for potential clients trying to connect
  - Accept - blocks waiting for a connection
- Client

  - Connect - try to connect to a server
- Uses a three-way handshake

  1. client sends segment declaring its starting sequence number
  2. server sends segment acknowledging the client's segment and declaring its starting sequence number
  3. client acknowledges the server's segment
- Client and server know what starting sequence number is safe to use
- SYN and ACK bits are used in these control segments
- These segments count as one byte each!



## TCP Transmission Policy

- Basic Policy

  - Receiver tells sender what sequence number is next and how many bytes it is willing to accept. Could be zero bytes.
  - Sender - tells the sequence number of the starting byte and how many bytes
- Gaps in data - can use either go-back-N or selective repeat, the latter is preferred. Don't react too quickly to gaps, they might fill with out of order arrivals. Every segment sent has a timeout associated with it.
- Piggyback if necessary
- When sender has been told not to send more, it can later ask if the receiver is ready for more.
- Slow sender problem - group bytes together when possible
- Slow receiver problem - don't just ask for one byte, ask for many



## Congestion Control in TCP

- Principle: If all TCP sites cooperate to control their input into the network then congestion can be controlled.
- Slow start

  - Congestion window and threshold are maintained
  - Threshold starts large
  - Window starts small and doubles in size if all the segments sent with the current window size are successfully acknowledged until the threshold is reached.
  - When the threshold is reached the window only increases by one maximum size segment.
  - If segments timeout then the threshold is cut in half and window starts small again.

## Timeouts in TCP

- Retransmission timer - when to resend if ack has not arrived

  - must be variable not fixed because round-trip delay can vary

  - RTT = a RTT + (1-a) M   commonly used value is a = 7/8.  RTT is the current estimate of round trip delay. M is the most recent round trip delay.

  - timeout period could be say 2 RTT or perhaps RTT + 4 D where D is an estimate of the current mean variation of the delay.
- Persistence timer - a packet to the sender may tell it to stop transmitting. The packet from the receiver telling it to restart again might get lost so it needs a timer.
- Keepalive timer - when a connection has been idle for a long time a packet may be needed to check to see if the other party is still there.