

# *Introduction to Networks and the Internet*

## **CMPE 80N**

*Winter 2004*

*Lecture 20*



## **Announcements**

- *Fourth quiz on Monday, March 1<sup>st</sup>.*
- *HTML summary posted on the Web page.*
- *Quiz review session today by Kiran in BE 354I from 4:45-5:45.*
  - *Practice quiz.*
- *In order to schedule a second review session, we will wait to hear from you first.*



## **More announcements**

- *Library presentation on Wed. March 3<sup>rd</sup>.*
- *“Internet behind the Web” video on March 10<sup>th</sup>.*
- *5<sup>th</sup>. Quiz Friday, March 12<sup>th</sup>.*
- *Final exam, Thu March 18<sup>th</sup>.*



## **The Transport Layer**

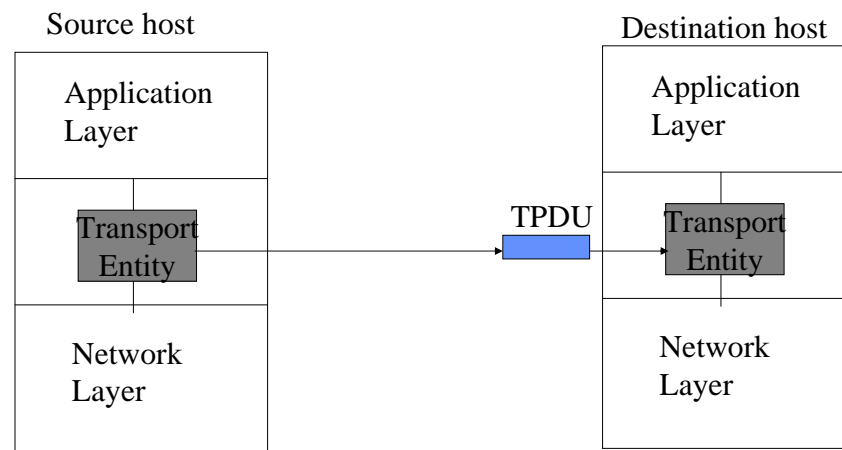


## The Transport Layer

- So far, all functions performed on a hop-by-hop basis.
- Transport layer: first end-to-end layer.
  - Communication from source to destination host.
  - Only hosts run transport-level protocols.
  - Under user's control as opposed to network layer which is controlled/owned by network provider.



## The Transport Layer



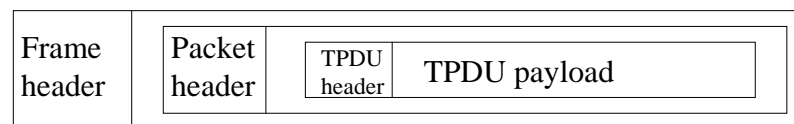
## Types of Transport Services

- Provided to the application layer.
- Connection-less versus connection-oriented.
- Connection-less service:
  - No logical connections, no flow or error control.
- Connection-oriented:
  - Based on logical connections: connection setup, data transfer, connection teardown.
  - Flow and error control.
  - Reliability and in-order delivery.



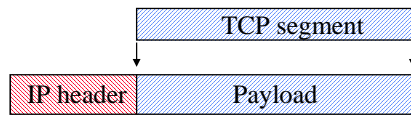
## TPDU

- Transport protocol data unit.
- Messages sent between transport entities.
- TPDU's contained in network-layer packets, which in turn are contained in DLL frames.



## Example: TCP

- **TCP segments travel in IP datagrams**



- *Internet routers only look at IP header to forward datagrams*



## Transport Protocol Addressing

- *Address of the transport-level entity.*
- *Several transport-level entities may be running on single machine.*
- *Source-destination address pair not enough to uniquely identify transport entity.*
- *Port number: uniquely identifies transport entity.*



## The Internet Transport Protocols: TCP and UDP

- *UDP: user datagram protocol (RFC 768).*
  - *Connection-less protocol.*
- *TCP: transmission control protocol (RFCs 793, 1122, 1323).*
  - *Connection-oriented protocol.*



## TCP

- *Reliable end-to-end communication.*
- *TCP transport entity:*
  - *Interfaces to the IP layer.*
  - *Manages TCP streams.*
    - *Accepts user data, breaks it down and sends it as separate IP datagrams.*
    - *At receiver, reconstructs original byte stream from IP datagrams.*



## Features of TCP

- **Connection oriented:** An application requests a “connection” to destination and uses connection to transfer data
  - IP does not use “connections” - each datagram is sent independently!
- **Point-to-point:** A TCP connection has two endpoints (no broadcast/multicast)
- **Reliability:** TCP guarantees that data will be delivered without loss, duplication or transmission errors



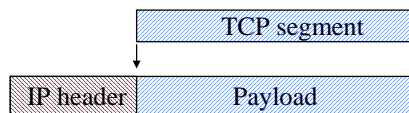
## Features of TCP (cont'd)

- **Full duplex:** Endpoints can exchange data in both directions simultaneously
- **Reliable connection startup:** TCP guarantees reliable, synchronized startup between endpoints (using “three-way handshake”)
- **Graceful connection shutdown:** TCP guarantees delivery of all data after endpoint shutdown



## Delivering TCP Segments

- **TCP segments travel in IP datagrams.**

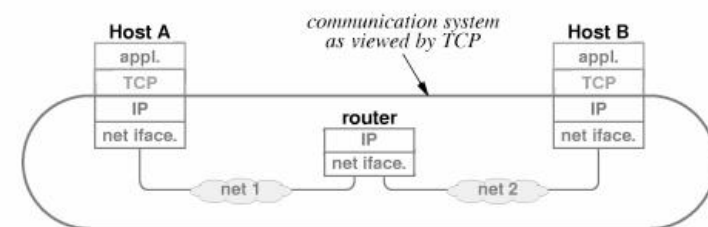


- **Internet routers only look at IP header to forward datagrams.**



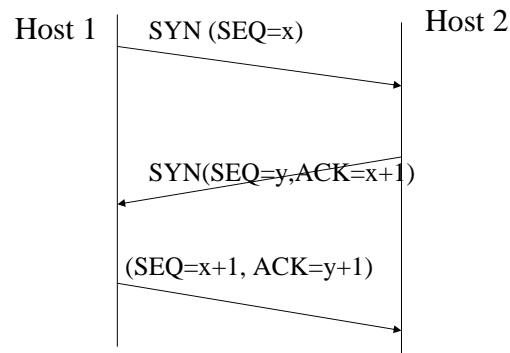
## Delivering TCP

- **TCP at destination interprets TCP messages**



## TCP Connection Setup

- 3-way handshake.



## TCP and Reliable Delivery

- TCP provides reliable delivery, recovering from:
  - Lost packets
  - Duplicate packets
  - Delayed packets
  - Corrupted data
  - Source/destination mismatches
  - Congestion



## TCP Reliability

- Reliable delivery.
  - Acknowledgements..
  - Timeouts and retransmissions.
- Ordered delivery.
  - Sequence numbers.



## Lost Packets

- Recipient sends **acknowledgment** control message (**ACK**) to sender to verify successful receipt of data
  - ACKs usually are carried onboard other TCP packets.
  - However, even if an application has nothing to transmit, it must transmit acknowledgment packets for each packet it receives.
- Thus, for each packet sent, a host expects to receive an acknowledgment, which ensures that the packet did not get lost.
  - What if the packet or the acknowledgment get lost?



## Lost Packets (cont'd)

- **Retransmission timer**
  - When a data segment is sent, a timer is started
  - If the segment is acknowledged before the timer expires, the timer is stopped and reset
  - Otherwise, the segment is retransmitted (and the timer is reset and started again)
- **The choice of the timeout is critical!**
  - If timeout is too long: overall throughput may be reduced (always waiting for acknowledgments)
  - If timeout is too short: too many packets get retransmitted (may increase network congestion)

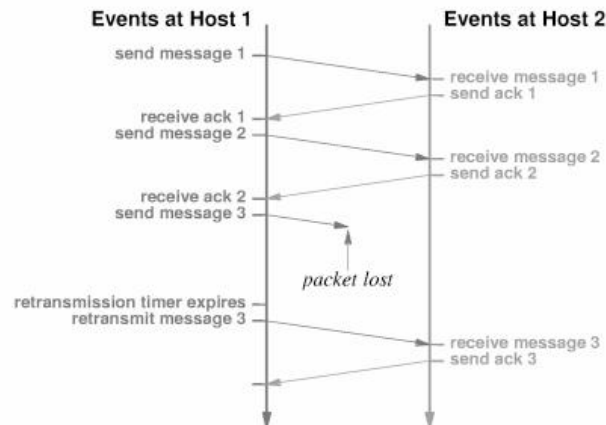


## Lost Packets (cont'd)

- **IMPORTANT:** packet retransmission (especially if it has to be carried out on an end-to-end basis) **significantly increases latency (delay)**
  - For real-time video or audio transmission, **delay** is a more important performance issue than **error rate**
  - Thus, in many cases it is preferable to forget the error and simply work with the received data stream



## Lost Packets - Example



## TCP Transmission

- *Sender process initiates connection.*
- *Once connection established, TCP can start sending data.*
- *Sender writes bytes to TCP stream.*
- *TCP sender breaks byte stream into segments.*
  - *Each byte assigned sequence number.*
  - *Segment sent and timer started.*



## TCP Transmission (cont'd)

- If timer expires, retransmit segment.
  - After retransmitting segment for maximum number of times, assumes connection is dead and closes it.



## Flow Control

- **Flow control** is necessary so that source doesn't transmit too fast for given receiver.
  - E.g., a fast server trying to send 1Gb/s data to a small PC.
  - Without some form of control, some data will get lost.
- **Requires feedback from receiver.**
  - So sender realizes it is sending too fast.

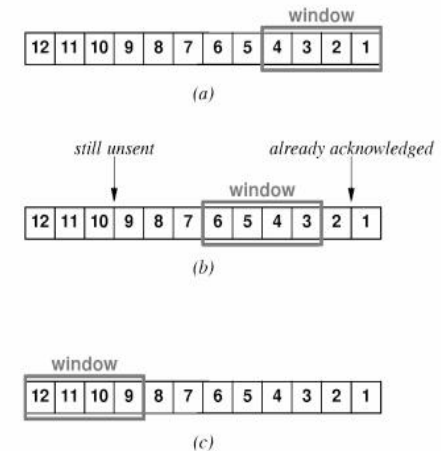


## Sliding Window

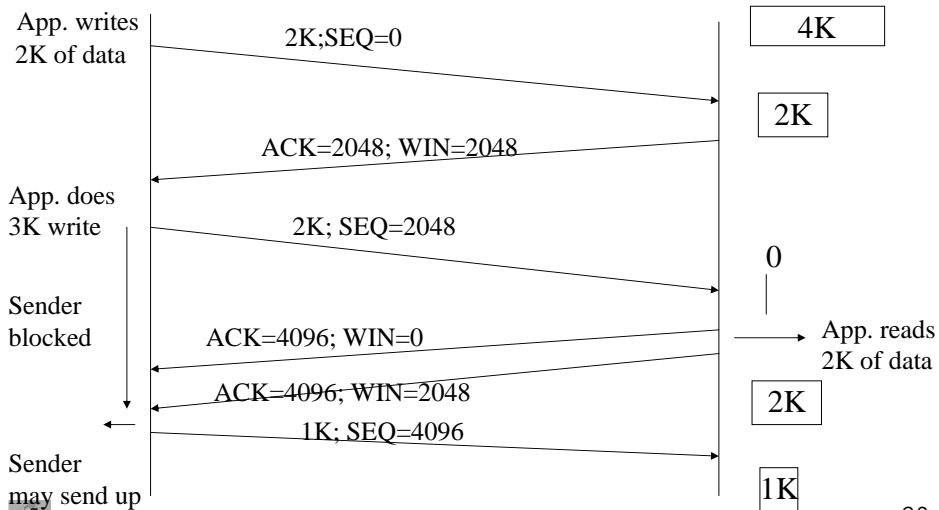
- TCP uses ACKs and sliding window mechanism for flow control.



## TCP Sliding Window



## TCP Flow Control: Example



## Congestion

- Network with 1 Mb/s lines and 1000 computers, half of which are trying to transfer files at 100 Kb/s to the other half.
  - The total offered traffic exceeds what the network can handle (**congestion**).
- **Congestion collapse:**
  - When congestion occurs, packets get dropped.
  - Due to packet loss, packets get retransmitted.
  - Congestions gets worse and worse!



## Congestion Control

- Why do it at the transport layer?
  - Real fix to congestion is to slow down sender.
- Use law of “conservation of packets”.
  - Keep number of packets in the network constant.
  - Don’t inject new packet until old one leaves.
- Congestion indicator: packet loss.



## TCP and Congestion Control

- Interprets packet loss as an indicator of congestion
  - When it senses packet loss, it slows down the rate of packet transmission
  - When packets are received correctly, sends packets faster
    - Still within the limits of the sliding window



## TCP Congestion Control

- Like, flow control, also window based.
  - Sender keeps congestion window (cwin).
  - Each sender keeps 2 windows: receiver's advertised window and congestion window.
  - Number of bytes that may be sent is  $\min(\text{advertised window}, \text{cwin})$ .



## TCP Congestion Control (cont'd)

- Slow start [Jacobson 1988]:
  - Connection's congestion window starts at 1 segment.
  - If segment ACKed before time out,  $\text{cwin} = \text{cwin} + 1$ .
  - As ACKs come in, current cwin is increased by 1.
  - Exponential increase.



## TCP Congestion Control (cont'd)

- Congestion Avoidance:
  - Third parameter: threshold.
  - Initially set to 64KB.
  - If timeout,  $\text{threshold} = \text{cwin}/2$  and  $\text{cwin} = 1$ .
  - Re-enters slow-start until  $\text{cwin} = \text{threshold}$ .
  - Then, cwin grows linearly until it reaches receiver's advertised window.



## TCP Retransmission Timer

- When segment sent, retransmission timer starts.
  - If segment ACKed, timer stops.
  - If time out, segment retransmitted and timer starts again.



## TCP Segment Header

Source port		Destination port		
Sequence number				
Acknowledgment number				
Header length	U	A	P R S F	Window size
Checksum		Urgent pointer		
Options (0 or more 32-bit words)				
Data				



## UDP

- Provides connection-less, unreliable service.
  - No delivery guarantees.
  - No ordering guarantees.
  - No duplicate detection.
- Low overhead.
  - No connection establishment/teardown.
- Suitable for short-lived connections.
  - Example: client-server applications.



## UDP Segment Format

0	15	31
Source port		Destination port
Length		Checksum
Data		

Source and destination ports: identify the end points.

Length: 8-byte header+ data.

Checksum: optional; if not used, set to zero.



## TCP and UDP

- **TCP** provides end-to-end communication. It takes care of **reliable, error-free transfer** of data, and **in-sequence delivery**
- **UDP** has **less overhead** compared to TCP, but **does not guarantee transfers**
  - TCP is preferred to transfer files
  - UDP is preferred to transfer audio/video streams
    - In real-time streaming, we cannot afford the delay consequent to packet retransmission
- Both protocols support **multiplexing**, i.e. they allow several distinct streams of data between two hosts

