Chapter 3
Transport Layer

TCP retransmit scenario

Host A
Seq=92, 8 bytes of data
ACK=100

Host B
Seq=120, 15 bytes of data

What does it mean if the sender receives multiple ACKs for the same packet?

An earlier packet is lost!

Fast retransmit after sender receipt of triple duplicate ACK

TCP fast retransmit

- time-out period often relatively long:
  - long delay before resending lost packet
- detect lost segments via duplicate ACKs.
  - sender often sends many segments back-to-back
  - if segment is lost, there will likely be many duplicate ACKs.

Chapter 3 outline

3.1 transport-layer services
3.2 multiplexing and demultiplexing
3.3 connectionless transport: UDP
3.4 principles of reliable data transfer
3.5 connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
3.6 principles of congestion control
3.7 TCP congestion control

TCP: retransmission scenarios

Host A
Seq=92, 8 bytes of data
ACK=100

Host B
Seq=100, 20 bytes of data
ACK=120

TCP flow control

- receiver controls sender, so sender won’t overflow receiver’s buffer by transmitting too much, too fast
TCP flow control

- receiver “advertises” free buffer space by including rwnd value in TCP header of receiver-to-sender segments
  - RcvBuffer size set via socket options (typical default is 4096 bytes, see setsockopt() and socket(7))
  - many operating systems autoadjust RcvBuffer
- sender limits amount of unacked (“in-flight”) data to receiver’s rwnd value
- guarantees receive buffer will not overflow

Chapter 3 outline

3.1 transport-layer services
3.2 multiplexing and demultiplexing
3.3 connectionless transport: UDP
3.4 principles of reliable data transfer
3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management
3.6 principles of congestion control
3.7 TCP congestion control

Connection Management

before exchanging data, sender/receiver “handshake”:
- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters

Agreeing to establish a connection

2-way handshake:

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters

TCP 3-way handshake

- variable delays
- retransmitted messages (e.g., req_conn(x)) due to message loss
- message reordering
- can’t “see” other side

Agreeing to establish a connection

2-way handshake failure scenarios:
TCP 3-way handshake: FSM

- Socket serverSocket = ServerSocket.accept();
- SYN(seq=x)
- create new socket for communication back to client
- SYN rcvd
- SYN sent
- ACK(seq=x+1)
- ESTAB

- ServerSocket
- SYN(seq=x)
- SYNACK(seq=y, ACKnum=x+1)
- create new socket for communication back to client
- SYNACK(seq=y, ACKnum=x+1)
- ACK(ACKnum=y+1)

TCP: closing a connection

- client, server each close their side of connection
  - send TCP segment with FIN bit = 1
- respond to received FIN with ACK
  - on receiving FIN, ACK can be combined with own FIN
- simultaneous FIN exchanges can be handled

TCP: closing a connection

- Client state
- ESTAB
- FIN_WAIT_1
- FIN_WAIT_2
- TIMED_WAIT
- CLOSED
- server state
- ESTAB
- CLOSE_WAIT
- LAST_ACK
- CLOSED
- clientSocket.close()