



15-441 Computer Networking

Lecture 15 – Transport Protocols
Srinu Seshan

Fall 2010
www.cs.cmu.edu/~prs/15-441-F10

Outline



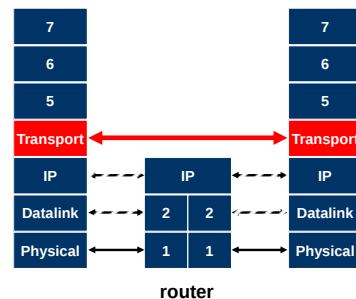
- Transport introduction
- Error recovery & flow control

2

Transport Protocols



- Lowest level end-to-end protocol.
 - Header generated by sender is interpreted only by the destination
 - Routers view transport header as part of the payload



3

Functionality Split



- Network provides best-effort delivery
- End-systems implement many functions
 - Reliability
 - In-order delivery
 - Demultiplexing
 - Message boundaries
 - Connection abstraction
 - Congestion control
 - ...

4

Transport Protocols

- UDP provides just integrity and demux
- TCP adds...
 - Connection-oriented
 - Reliable
 - Ordered
 - Point-to-point
 - Byte-stream
 - Full duplex
 - Flow and congestion controlled

5

UDP: User Datagram Protocol [RFC 768]

- “No frills,” “bare bones” Internet transport protocol
- “Best effort” service, UDP segments may be:
 - Lost
 - Delivered out of order to app
- *Connectionless*:
 - No handshaking between UDP sender, receiver
 - Each UDP segment handled independently of others

Why is there a UDP?

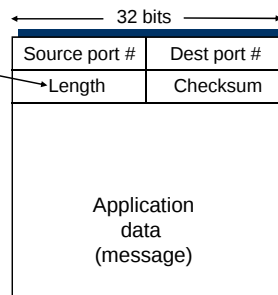
- No connection establishment (which can add delay)
- Simple: no connection state at sender, receiver
- Small header
- No congestion control: UDP can blast away as fast as desired

6

UDP, cont.

- Often used for streaming multimedia apps
 - Loss tolerant
 - Rate sensitive
- Other UDP uses (why?):
 - DNS, SNMP
- Reliable transfer over UDP
 - Must be at application layer
 - Application-specific error recovery

Length, in bytes of UDP segment, including header



UDP segment format

7

UDP Checksum

Goal: detect “errors” (e.g., flipped bits) in transmitted segment – optional use!

Sender:

- Treat segment contents as sequence of 16-bit integers
- Checksum: addition (1's complement sum) of segment contents
- Sender puts checksum value into UDP checksum field

Receiver:

- Compute checksum of received segment
 - Check if computed checksum equals checksum field value:
 - NO - error detected
 - YES - no error detected
- But maybe errors nonetheless?*

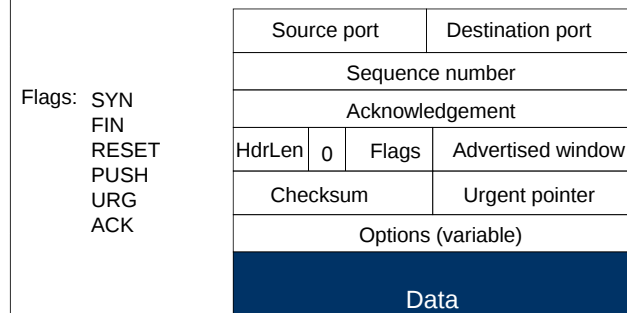
8

High-Level TCP Characteristics

- Protocol implemented entirely at the ends
 - Fate sharing
- Protocol has evolved over time and will continue to do so
 - Nearly impossible to change the header
 - Use options to add information to the header
 - These do change sometimes
 - Change processing at endpoints
 - Backward compatibility is what makes it TCP

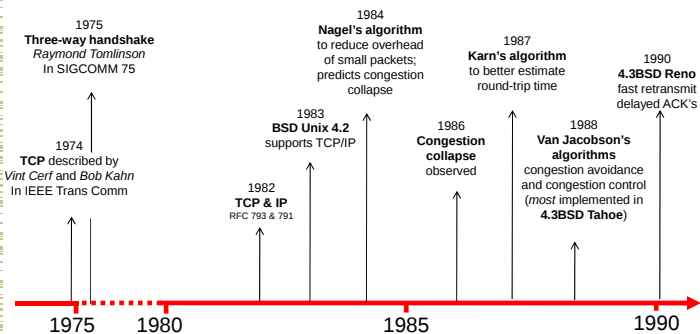
9

TCP Header



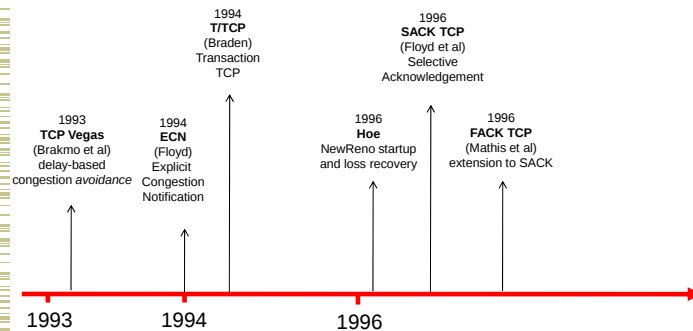
10

Evolution of TCP



11

TCP Through the 1990s



12

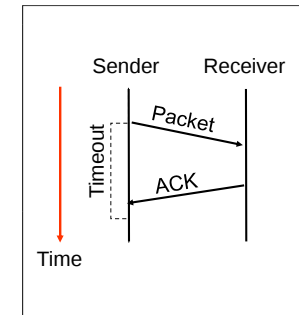
Outline

- Transport introduction
- Error recovery & flow control

13

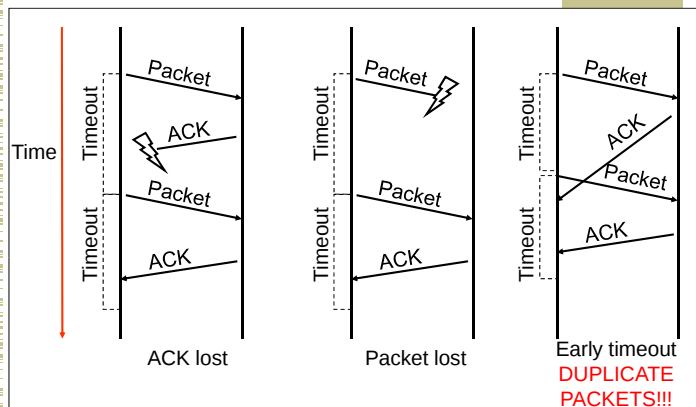
Stop and Wait

- ARQ
 - Receiver sends acknowledgement (ACK) when it receives packet
 - Sender waits for ACK and timeouts if it does not arrive within some time period
- Simplest ARQ protocol
- Send a packet, stop and wait until ACK arrives



14

Recovering from Error



15

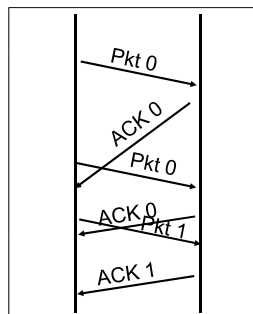
Problems with Stop and Wait

- How to recognize a duplicate
- Performance
 - Can only send one packet per round trip

16

How to Recognize Resends?

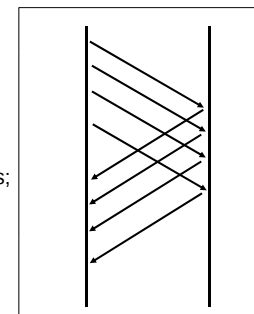
- Use sequence numbers
 - both packets and acks
- Sequence # in packet is finite
 - How big should it be?
 - For stop and wait?
- One bit – won't send seq #1 until received ACK for seq #0



17

How to Keep the Pipe Full?

- Send multiple packets without waiting for first to be acked
 - Number of pkts in flight = window
- Reliable, unordered delivery
 - Several parallel stop & waits
 - Send new packet after each ack
 - Sender keeps list of unack'ed packets; resends after timeout
 - Receiver same as stop & wait
- How large a window is needed?
 - Suppose 10Mbps link, 4ms delay, 500byte pkts
 - 1? 10? 20?
 - Round trip delay * bandwidth = capacity of pipe



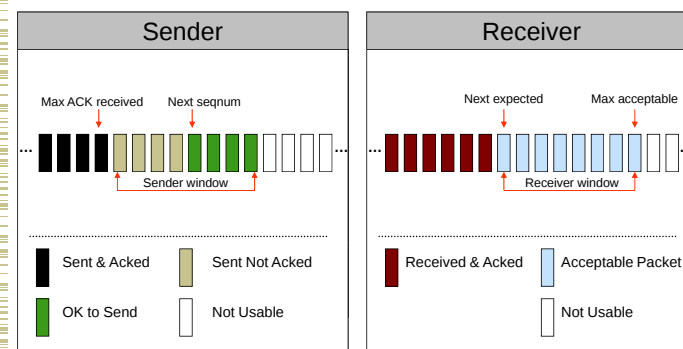
18

Sliding Window

- Reliable, ordered delivery
- Receiver has to hold onto a packet until all prior packets have arrived
 - Why might this be difficult for just parallel stop & wait?
 - Sender must prevent buffer overflow at receiver
- Circular buffer at sender and receiver
 - Packets in transit $\leq$ buffer size
 - Advance when sender and receiver agree packets at beginning have been received

19

Sender/Receiver State



20

Sequence Numbers



- How large do sequence numbers need to be?
 - Must be able to detect wrap-around
 - Depends on sender/receiver window size
- E.g.
 - Max seq = 7, send win=rcv win=7
 - If pkts 0..6 are sent successfully and all acks lost
 - Receiver expects 7,0..5, sender retransmits old 0..6!!!
- Max sequence must be $\geq$ send window + rcv window

21

Window Sliding – Common Case



- On reception of new ACK (i.e. ACK for something that was not acked earlier)
 - Increase sequence of max ACK received
 - Send next packet
- On reception of new in-order data packet (next expected)
 - Hand packet to application
 - Send **cumulative ACK** – acknowledges reception of all packets up to sequence number
 - Increase sequence of max acceptable packet

22

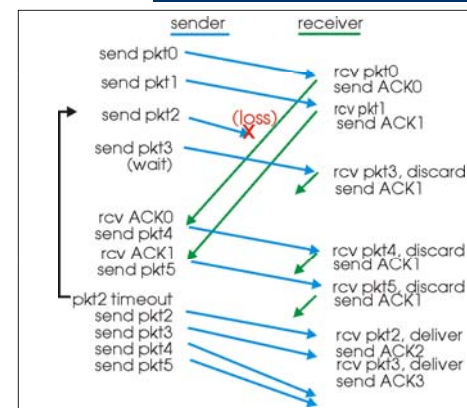
Loss Recovery



- On reception of out-of-order packet
 - Send nothing (wait for source to timeout)
 - Cumulative ACK (helps source identify loss)
- Timeout (Go-Back-N recovery)
 - Set timer upon transmission of packet
 - Retransmit all unacknowledged packets
- Performance during loss recovery
 - No longer have an entire window in transit
 - Can have much more clever loss recovery

23

Go-Back-N in Action



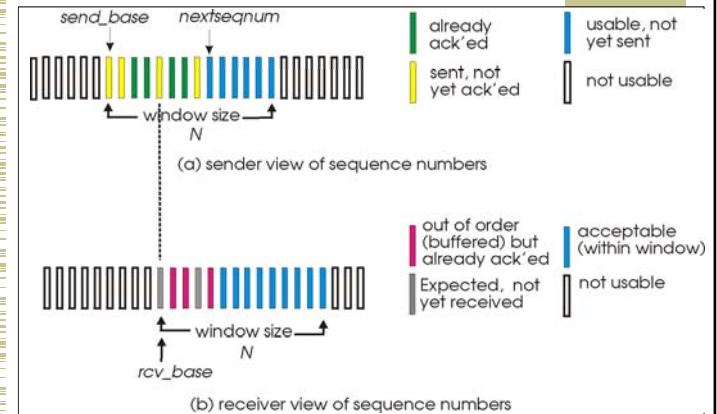
24

Selective Repeat

- Receiver *individually* acknowledges all correctly received pkts
 - Buffers packets, as needed, for eventual in-order delivery to upper layer
- Sender only resends packets for which ACK not received
 - Sender timer for each unACKed packet
- Sender window
 - N consecutive seq #'s
 - Again limits seq #'s of sent, unACKed packets

25

Selective Repeat: Sender, Receiver Windows



26

Important Lessons

- Transport service
 - UDP → mostly just IP service
 - TCP → congestion controlled, reliable, byte stream
- Types of ARQ protocols
 - Stop-and-wait → slow, simple
 - Go-back-n → can keep link utilized (except w/ losses)
 - Selective repeat → efficient loss recovery
- Sliding window flow control
 - Addresses buffering issues and keeps link utilized

27

Next Lecture

- Congestion control
- TCP Reliability

28

Ponder This...



- A bus station is where a bus stops.
- A train station is where a train stops.
- A work station is where...
- Maybe that explains why it is so hard getting project 2 done ☺ ouch