Today’s Lecture

- Mid-Term Exam
- Transport
  - UDP
  - TCP

Mid-Term Exam

- Two pages notes
  - Front / back
  - Typed or hand written
- Exam components
  - Short answer
    - Is CSMA / CD deterministic or non-deterministic?
  - Exercises
    - Compute a netmask / write pseudocode
  - Scenarios
    - X notes that A > B, is that true?

Mid-Term Exam

- 75 minutes
  - Normal class period
- Tips
  - Read exam through before starting
  - Record the answers to key points in your two pages of notes
  - Example code blurbs
  - When in doubt, ASK!

Reliable Byte-Stream (TCP)

Outline
- Connection Establishment/Termination
- Sliding Window Revisited
- Flow Control
- Adaptive Timeout

TCP Segment Format
Segment Format (cont)

- Each connection identified with 4-tuple:
  - (SrcPort, SrcIPAddr, DstPort, DstIPAddr)
- Sliding window + flow control
  - ACK, SequenceNum, AdvertisedWindow
- Flags
  - SYN, FIN, RESET, PUSH, URG, ACK
- Checksum
  - pseudo header + TCP header + data

Connection Establishment

Active participant (client) - Passive participant (server)

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

Connection Termination

First participant - Second participant

- FIN, SequenceNum = x
  - ACK, Acknowledgment = y + 1
  - ACK, Acknowledgment = x + 1, FIN, SequenceNum = y

State Transition Diagram

Sliding Window Revisited

- Sending side
  - LastByteAcked ≤ LastByteSent
  - LastByteSent ≤ LastByteWritten
  - buffer bytes between LastByteAcked and LastByteWritten

- Receiving side
  - LastByteRead ≤ NextByteExpected
  - NextByteExpected ≤ LastByteWritten + 1
  - buffer bytes between LastByteRead and LastByteWritten

Example – Sequence / Ack
Flow Control

- Send buffer size: `MaxSendBuffer`
- Receive buffer size: `MaxRcvBuffer`
- Receiving side
  - `LastByteRcvd - LastByteRead` ≤ `MaxRcvBuffer`
  - `AdvertisedWindow` = `MaxRcvBuffer` - `(NextByteExpected - 1) - LastByteRcvd`
- Sending side
  - `LastByteSent - LastByteAcked` ≤ `AdvertisedWindow`
  - `EffectiveWindow` = `AdvertisedWindow` - `(LastByteSent - LastByteAcked)`
  - `LastByteWritten - LastByteAcked` ≤ `MaxSendBuffer`
- Always send ACK in response to arriving data segment
- Persist when `AdvertisedWindow` = 0

TCP Rate Control - Brief

- Protection Against Wrap Around
  - 32-bit `SequenceNum`
  - `Bandwidth` | `Time Until Wrap Around`
    - Ethernet (10 Mbps) | 6.4 hours
    - T3 (45 Mbps) | 13 minutes
    - FDDI (100 Mbps) | 6 minutes
    - STS-3 (155 Mbps) | 4 minutes
    - STS-12 (622 Mbps) | 55 seconds
    - STS-24 (1.2 Gbps) | 28 seconds

Silly Window Syndrome

- How aggressively does sender exploit open window?
- Receiver-side solutions
  - after advertising zero window, wait for space equal to a maximum segment size (MSS)
  - delayed acknowledgements

Nagle’s Algorithm

- How long does sender delay sending data?
  - too long: hurts interactive applications
  - too short: poor network utilization
  - strategies: timer-based vs self-clocking

- when application produces data to send
  - if both the available data and the window >= MSS
    - send a full segment
  - else if there is unACKed data in flight
    - buffer the new data until an ACK arrives
  - else send all the new data now

Adaptive Retransmission

- Round-Trip Time Estimation:
  - wait at least one RTT before retransmitting
  - importance of accurate RTT estimators:
    - Low RTT -> unneeded retransmissions
    - High RTT -> poor throughput
  - RTT estimator must adapt to change in RTT
  - But not too fast, or too slow!
- problem: if the instantaneously calculated RTT is 10, 20, 5, 12, 3, 5, 6;
  - RTT estimator must adapt to change in RTT
- EstimatedRTT = α * EstimatedRTT + (1 - α) SampleRTT
- recommended value for α: 0.8 - 0.9
- retransmit timer set to β * RTT, where β = 2

Running average
Retransmission Ambiguity

Karn/Partridge Algorithm

- Accounts for retransmission ambiguity
- If a segment has been retransmitted:
  - don’t count RTT sample on ACKs for this segment
  - reuse RTT estimate only after one successful transmission
  - double timeout after each retransmission

Jacobson/Karels Algorithm

- Key observation:
  - using β RTT for timeout doesn’t work
  - at high loads round trip variance is high
- Solution:
  - if D denotes mean variation
  - timeout = RTT + 4D

Jacobson/Karels Algorithm

- New calculations for average RTT
- 
  \[ \text{Diff} = \text{SampleRTT} - \text{EstimatedRTT} \]
  \[ \text{EstimatedRTT} = \text{EstimatedRTT} + (d \times \text{Diff}) \]
  \[ \text{Dev} = \text{Dev} + d \times (|\text{Diff}| - \text{Dev}) \]
  - where \( d \) is a factor between 0 and 1
- Consider variance when setting timeout value
- 
  \[ \text{TimeOut} = m \times \text{EstimatedRTT} + f \times \text{Dev} \]
  - where \( m = 1 \) and \( f = 4 \)

Record Boundaries

- Byte-stream protocol: write 8+2+20 bytes and read 5+5+5+5+5+5 (loop).
- TCP offers two features to insert record boundaries:
  - URG flag
  - push operation

TCP Extensions

- Implemented as header options
- Better way to measure RTT (use actual system clock for sending time and add timestamp to segment).
- 64-bit sequence numbers: 32-bit sequence number in low-order 32 bits, timestamp in high-order 32 bits.
- Shift (scale) advertised window.