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

Connection Termination
- First participant
  - FIN, SequenceNum = x
  - ACK, Acknowledgment = x+1
- Second participant
  - ACK, Acknowledgment = y+1
  - FIN, SequenceNum = y
  - ACK, Acknowledgment = x+1
State Transition Diagram

Sliding Window Revisited

Example – Sequence / Ack
Flow Control

- Send buffer size: \texttt{MaxSendBuffer}
- Receive buffer size: \texttt{MaxRcvBuffer}
- Receiving side
  - \(\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer}\)
  - \(\text{AdvertisedWindow} = \text{MaxRcvBuffer} - (\text{NextByteExpected} - 1) - \text{LastByteRead}\)
- Sending side
  - \(\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}\)
  - \(\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})\)
  - \(\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}\)
- Block sender if \((\text{LastByteWritten} - \text{LastByteAcked}) + \gamma > \text{MaxSenderBuffer}\)
- Always send ACK in response to arriving data segment
- Persist when \(\text{AdvertisedWindow} = 0\)

TCP Rate Control - Brief

Protection Against Wrap Around

- 32-bit \texttt{SequenceNum}

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Time Until Wrap Around</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 (1.5 Mbps)</td>
<td>6.4 hours</td>
</tr>
<tr>
<td>Ethernet (10 Mbps)</td>
<td>57 minutes</td>
</tr>
<tr>
<td>T3 (45 Mbps)</td>
<td>13 minutes</td>
</tr>
<tr>
<td>FDDI (100 Mbps)</td>
<td>6 minutes</td>
</tr>
<tr>
<td>STS-1 (155 Mbps)</td>
<td>4 minutes</td>
</tr>
<tr>
<td>STS-12 (622 Mbps)</td>
<td>55 seconds</td>
</tr>
<tr>
<td>STS-24 (1.2 Gbps)</td>
<td>28 seconds</td>
</tr>
</tbody>
</table>
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

```plaintext
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 -> unnecessary 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; what RTT should we use for calculations?
    - EstimatedRTT = α * EstimatedRTT + (1 - α) SampleRTT
    - recommended value for α: 0.8 – 0.9
    - retransmit timer set to β RTT, where β = 2

---
**Retransmission Ambiguity**

![Diagram showing retransmission ambiguity](image)

**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 + 4\( D \)
Jacobson/Karels Algorithm

- New Calculations for average RTT
- Diff = SampleRTT - EstimatedRTT
- EstimatedRTT = EstimatedRTT + (d * Diff)
- Dev = Dev + d * (|Diff| - Dev)
  - where d is a factor between 0 and 1
- Consider variance when setting timeout value
- TimeOut = m * EstimatedRTT + f * 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.