CSE 461: Introduction to Computer Communication Networks

Chunjong Park
Reliable Data Transfer

- A sends a packet to B
- Ideally, the packet should arrive at B
- But A does not know whether B receives it
- How could B tell A that the packet is arrived at B?
Reliable Data Transfer: ACK

- A sends a packet to B
- The packet arrives at B
- B tells A that it receives the packet
- A sends out the next packet
Reliable Data Transfer: Packet loss

- But what if a packet or an ACK is lost?
- A can’t wait for an ACK forever.
Reliable Data Transfer: Timeout

- A only waits for a certain period of time
- When timeout, A resends the packet
Premature Timeout

Host A

Pkt

ACK

timeout

Host B

Pkt

ACK

ACK for which pkt?

Host A

timeout

Pkt

ACK

Host B

Pkt

ACK

Pkt
Sequence #

Host A

<table>
<thead>
<tr>
<th>Pkt0</th>
<th>ACK0</th>
<th>Pkt0</th>
<th>ACK0</th>
</tr>
</thead>
</table>

Host B

<table>
<thead>
<tr>
<th>Pkt0</th>
<th>ACK0</th>
</tr>
</thead>
</table>

Timeout

ACK0 for Pkt0 is already received. Ignored

Host A

<table>
<thead>
<tr>
<th>Pkt1</th>
<th>ACK0</th>
</tr>
</thead>
</table>

Host B

<table>
<thead>
<tr>
<th>Pkt0</th>
<th>ACK0</th>
</tr>
</thead>
</table>

Sending Pkt1
Stop-and-wait

- A sender sends only a single packet before it receives the corresponding ACK
- Only needs 0/1 for sequence number
  - Just needs to distinguish two consecutive pkts
- Physical link is underutilized!
Stop-and-wait

- $R = 1$ Gbps link, $RTT = 15$ ms prop. delay, $L = 8000$ bit packet

\[
U_{sender} = \frac{L / R}{RTT + L / R} = \frac{0.008}{30.008} = 0.00027
\]

Borrowed from Kurose’s slides
Pipelined protocols (Sliding Window)

pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

- range of sequence numbers must be increased
- buffering at sender and/or receiver

two generic forms of pipelined protocols: go-Back-N, selective repeat

Borrowed from Kurose’s slides
Sliding Window: Sender

Borrowed from Kurose’s slides
Sliding Window: Increased Utilization

- \( R = 1 \text{ Gbps link, } RTT = 15 \text{ ms prop. delay, } L = 8000 \text{ bit packet} \)

\[
U_{\text{sender}} = \frac{3L / R}{RTT + L / R} = \frac{0.0024}{30.008} = 0.00081
\]

3-packet pipelining increases utilization by a factor of 3!

Borrowed from Kurose’s slides
Sliding Window: Selective Repeat

Borrowed from Kurose's slides
Selective Repeat

**sender**

data from above:
- if next available seq # in window, send pkt

timeout(n):
- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:
- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

**receiver**

pkt n in [rcvbase,rcvbase+N-1]
- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]
- ACK(n)

otherwise:
- ignore

Borrowed from Kurose’s slides
Selective Repeat in Action

**sender window (N=4)**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

**sender**

- send pkt0
- send pkt1
- send pkt2
- send pkt3 (wait)
- rcv ack0, send pkt4
- rcv ack1, send pkt5
- record ack3 arrived
- pkt 2 timeout
- send pkt2
- record ack4 arrived
- record ack5 arrived

**receiver**

- receive pkt0, send ack0
- receive pkt1, send ack1
- receive pkt3, buffer, send ack3
- receive pkt4, buffer, send ack4
- receive pkt5, buffer, send ack5
- rcv pkt2; deliver pkt2, pkt3, pkt4, pkt5; send ack2
- record ack4 arrived
- record ack5 arrived

**Q:** what happens when ack2 arrives?
Selective Repeat in Action

**sender window (N=4)**

<table>
<thead>
<tr>
<th>Sender Window</th>
<th>Sender Event</th>
<th>Receiver Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>send pkt0</td>
<td>receive pkt0, send ack0</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>send pkt1</td>
<td>receive pkt1, send ack1</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>send pkt2</td>
<td>receive pkt3, buffer, send ack3</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>(wait)</td>
<td>pkt 2 timeout</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>send pkt3</td>
<td>receive pkt4, buffer, send ack4</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>record ack0 arrived</td>
<td>receive pkt5, buffer, send ack4</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>rcv ack0, send pkt4</td>
<td>record ack4 arrived</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>rcv ack1, send pkt5</td>
<td>record ack5 arrived</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8</td>
<td>send pkt2</td>
<td>rcv pkt2; deliver pkt2, pkt3, pkt4, pkt5; send ack2</td>
</tr>
</tbody>
</table>

**Record:**
- ack3 arrived
- ack4 arrived
- ack5 arrived

**Packet:**
- Send pkt6,7,8,9 (wait)