6.02 Fall 2014
Lecture #23

- Reliable transport
- Sliding-window protocol
- Analysis of sliding-window
Unanswered questions
(about packet-switched networks)

**How do nodes determine routes to every other node?**
Nodes determine routes via either **link-state**, **distance-vector**, or path-vector routing.

**How do nodes route around link failures?**
Routing protocols will **eventually converge**, but experience different problems along the way (routing loops, counting-to-infinity, etc.).

**How do nodes communicate reliably given that the network is best-effort?**
Nodes can use a **stop-and-wait protocol**.
Recap: Stop-and-wait Protocol

At sender:
- Send a packet, keep track of its sequence number
- When an ACK is received for that packet, increment the stored sequence number and repeat
- If an ACK for the outstanding packet hasn’t been received after timeout seconds, retransmit the packet

At receiver:
- Upon receipt of packet k, send an ACK for k
- If k is greater than the last received sequence number, deliver packet to app
Reliable Communication

today’s goal: develop a reliable transport protocol that gets better utilization than the stop-and-wait protocol
Sliding-window Protocol

Window: [4, 5, 6]

**Basic Idea:** Send a new packet whenever an ACK is received, allowing no more than \( W \) outstanding packets at a time.
Sliding-window Protocol
(same protocol, different visualization)

sender

receiver

window

1 2 3 4 5
Sliding-window Protocol

(same protocol, different visualization)
Sliding-window Protocol

(same protocol, different visualization)
Sliding-window Protocol
(same protocol, different visualization)
Sliding-window Protocol: Sender

(sender receiver app)

(on same machine)
Sliding-window Protocol: Sender

(on same machine)

Window: [1,2,3,4,5]

sender  receiver  app

seq 1
seq 2
seq 3
seq 4
seq 5
Sliding-window Protocol: Sender

Window: [1,2,3,4,5]

sender               receiver                app

seq 1
seq 2
seq 3
seq 4
seq 5

(on same machine)
Sliding-window Protocol: Sender

(on same machine)

**sender**

**receiver**

**app**

Window: [2, 3, 4, 5, 6]

- seq 1
- seq 2
- seq 3
- seq 4
- seq 5
- seq 6
Sliding-window Protocol: Sender

(sender) seq$1 receiver app

Window: [2,3,4,5,6]

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6

(on same machine)
Sliding-window Protocol: Sender

Window: \([3,4,5,6,7]\)
Sliding-window Protocol: Sender

Window: \([3,4,5,6,7]\)
Sliding-window Protocol: Sender

(sender) seq$1 seq$2 seq$3 seq$4 seq$5

(receiver) seq$6 seq$7 seq$8

(Window: [4,5,6,7,8])

(app)
Sliding-window Protocol: Sender

(on same machine)

Window: [4, 5, 6, 7, 8]
Sliding-window Protocol: Sender

(on same machine)

sender  receiver  app

Window: [5,6,7,8,9]
Sliding-window Protocol: Sender

Window: [5,6,7,8,9]
Sliding-window Protocol: Sender

Window: [6,7,8,9,10]
Sliding-window Protocol: Sender

Sender: seq 1, seq 2, seq 3, seq 4, seq 5
Receiver: seq 6, X, seq 8, seq 9, seq 10
App: (on same machine)

Window: [6, 7, 8, 9, 10]
Sliding-window Protocol: Sender

Window: [7,8,9,10,11]
## Sliding-window Protocol: Sender

(On same machine)

<table>
<thead>
<tr>
<th>sender</th>
<th>receiver</th>
<th>app</th>
</tr>
</thead>
</table>

Window: $[7, 8, 9, 10, 11]$
Sliding-window Protocol: Sender

Window: [7,8,9,10,11]

sender

receiver

app

 timeouts

seq 1

seq 2

seq 3

seq 4

seq 5

seq 6

seq 7

seq 8

seq 9

seq 10

seq 11

seq 7

(on same machine)
Sliding-window Protocol: Sender

(on same machine)

Window: [7,8,9,10,11]
Sliding-window Protocol: Sender

Window: [7, 9, 10, 11, 12]
window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

Window: $[7, 9, 10, 11, 12]$
window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

Window: [7, 10, 11, 12, 13]
window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

Window: [7,10,11,12,13]

window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

Window: [7, 11, 12, 13, 14]

window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

Window: [7, 12, 13, 14, 15]
window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

Window: [12, 13, 14, 15, 16]
window doesn’t have to be contiguous!
Sliding-window Protocol: Sender

- Transmit a packet if $\text{len(un-ACKed list)} < W$
- Upon transmission of packet $k$, keep track of $k$ in the un-ACKed list, and the time that $k$ was sent
- When an ACK for packet $k$ is received, remove $k$ from the un-ACKed list.
- Periodically check the un-ACKed list to see if any packets were sent more than timeout seconds ago. If so, re-transmit.
Sliding-window Protocol: Receiver

(sender) (receiver) (app)

(on same machine)
Sliding-window Protocol: Receiver

(sender) seq$1 -> receiver seq$2 -> app seq$3 -> seq$4 -> seq$5
Sliding-window Protocol: Receiver

(on same machine)

sender  receiver  app

seq 1
seq 2
seq 3
seq 4
seq 5

seq 1
Sliding-window Protocol: Receiver

(sender) \[\text{seq}^1\text{sender}\] \[\text{seq}\] \[\text{receiver}\] \[\text{app}\]

(on same machine)

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6

arrow from seq 1
Sliding-window Protocol: Receiver

(on same machine)

sender  receiver  app

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6

seq 1
seq 2
Sliding-window Protocol: Receiver

(on same machine)

sender  receiver  app

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6
seq 7

X

seq 1
seq 2
Sliding-window Protocol: Receiver

(sender) seq$1 sender seq$2 sender seq$3 seq$4 seq$5

(receiver) seq$6 receiver seq$7

(app) (on same machine) seq 1 seq 2 seq 3
Sliding-window Protocol: Receiver

(on same machine)

sender -> receiver -> app

seq 1 -> seq 1
seq 2 -> seq 2
seq 3 -> seq 3
seq 4 -> seq 4
seq 5 -> seq 5
seq 6 -> seq 6
seq 7 -> seq 7
seq 8 -> seq 8

X
Sliding-window Protocol: Receiver

(send on same machine)
Sliding-window Protocol: Receiver

(sender) seq$1 sender  seq$2 sender  seq$3 sender  seq$4 sender  seq$5

(receiver) seq$6 receiver  seq$7 receiver  seq$8 receiver  seq$9

(app)  seq$1 app  seq$2 app  seq$3 app  seq$4

(on same machine)
Sliding-window Protocol: Receiver

(on same machine)

sender     receiver     app

- seq 1
- seq 2
- seq 3
- seq 4
- seq 5
- seq 6
- seq 7
- seq 8
- seq 9

- seq 1
- seq 2
- seq 3
- seq 4
- seq 5

X
Sliding-window Protocol: Receiver

(seq 1, seq 2, seq 3, seq 4, seq 5) -> (seq 1, seq 2, seq 3, seq 4, seq 5)

(seq 6, seq 7, seq 8, seq 9, seq 10) -> (seq 6, seq 7, seq 8, seq 9, seq 10)
Sliding-window Protocol: Receiver

(sender) seq$1 seq$2 seq$3 seq$4 seq$5
(receiver) seq$1 seq$2 seq$3 seq$4 seq$5
(app) seq$6 seq$7 seq$8 X seq$9 seq$10

(on same machine)
Sliding-window Protocol: Receiver

(sender) seq$1 sender seq$2 sender seq$3 sender seq$4 sender seq$5

(receiver) seq$2 receiver seq$3 receiver seq$4 receiver X

(app) seq$6 app seq$5 app seq$4 app seq$3

(seq 1) app seq 1
(seq 2) app seq 2
(seq 3) app seq 3
(seq 4) app seq 4
(seq 5) app seq 5
(seq 6) app seq 6
(seq 7) app seq 7
(seq 8) app seq 8
(seq 9) app seq 9
(seq 10) app seq 10
(seq 11) app seq 11

(on same machine)
Sliding-window Protocol: Receiver

(seq$1) sender → receiver → app (on same machine)

(sender)

seq 1 → seq 2 → seq 3 → seq 4 → seq 5 →

(seq$2) seq 6 → seq 7 → seq 8 →

(seq$3) seq 9 → seq 10 →

(seq$4) seq 11

(seq$5)

(seq$6)

(seq$7)

timeout

(seq$8)
Sliding-window Protocol: Receiver

(sender on same machine)

seq$1

seq$2

seq$3

seq$4

seq$5

seq$6

seq$7

seq$8

seq$9

seq$10

seq$11

timeout

seq$1

seq$2

seq$3

seq$4

seq$5

seq$6

seq$7
Sliding-window Protocol: Receiver

(on same machine)

sender  receiver  app

seq 1  seq 1
seq 2  seq 2
seq 3  seq 3
seq 4  seq 4
seq 5  seq 5

seq 6
seq 7
seq 8
seq 9
seq 10

seq 11
seq 7

receiver buffer: [8]

timeout
Sliding-window Protocol: Receiver

(on same machine)

sender  receiver  app

timeout

receiver buffer:

[8]
Sliding-window Protocol: Receiver

(on same machine)

sender       receiver       app

seq$1

seq$2

seq$3

seq$4

seq$5

seq$2

seq$3

seq$4

seq$6

seq$7

X

seq$9

seq$10

seq$5

seq$11

seq$12

seq$8

seq$1

seq$2

seq$3

seq$4

seq$5

seq$6

receiver buffer: [8,9]
Sliding-window Protocol: Receiver

(On same machine)

**sender**

- seq 1
- seq 2
- seq 3
- seq 4
- seq 5
- seq 6
- seq 7
- seq 8
- seq 9
- seq 10
- seq 11
- seq 12
- seq 13

**receiver**

- seq 1
- seq 2
- seq 3
- seq 4
- seq 5
- seq 6
- seq 7

**app**

- seq 1
- seq 2
- seq 3
- seq 4
- seq 5
- seq 6

**receiver buffer:**

[8,9]
Sliding-window Protocol: Receiver

(on same machine)

sender -> receiver -> app

receiver buffer: [8,9,10]
Sliding-window Protocol: Receiver

(sender) seq$1 seq$2 seq$3 seq$4 seq$5
(receiver) seq$2 seq$3 seq$4
(app) seq$6 seq$7

receiver buffer: [8,9,10]

timeout

(seq 1) -> (seq 2) -> (seq 3) -> (seq 4) -> (seq 5)
(seq 6) -> (seq 7) -> (seq 8) -> (seq 9) -> (seq 10)
(seq 11) -> (seq 12) -> (seq 13) -> (seq 14)
Sliding-window Protocol: Receiver

(on same machine)

sender

seq 1
seq 2
seq 3
seq 4
seq 5

seq 6
seq 7
seq 8
seq 9
seq 10
seq 11
seq 12
seq 13
seq 14

receiver

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6

receiver buffer: [8,9,10,11]

app

X

timeout
Sliding-window Protocol: Receiver

(On same machine)

sender | receiver | app

seq 1
seq 2
seq 3
seq 4
seq 5

seq 6
seq 7
seq 8
seq 9
seq 10

seq 11
seq 12
seq 13
seq 14

Receiver buffer: []

seq $1$
seq $2$
seq $3$
seq $4$
seq $5$

seq $6$
seq $7$
seq $8$

seq $9$
seq $10$
seq $11$
seq $12$
seq $13$
seq $14$

Receiver buffer:
[]
Sliding-window Protocol: Receiver

- Send an ACK for every received packet
- Save delivered packets — ignoring duplicates — in a local buffer
- Keep track of the next packet the application expects. After each reception, deliver as many in-order packets as possible.
timeout is slightly larger than rtt, as it should be.

(refer to)

The graph shows the relationship between the DATA or ACK sequence number and time. The y-axis represents the sequence number, and the x-axis represents time in milliseconds. The graph includes points labeled 'DATA' and 'ACK' indicating retransmissions. The window size and retransmissions ACKs for retransmitted packets are also indicated on the graph.
Window Size vs. Throughput

Throughput

Window Size (W)

$B \times RTT_{\text{min}}$
Max queue size: 100 packets
Packet size: 1000 bytes
ACK size: 40 bytes
Initial window size: 10 packets

1. Double W
2. Halve the propagation times
3. Double bottleneck link rate
Netflix takes up 9.5% of \textit{upstream} traffic on the North American Internet

ACK packets make Netflix an upload monster during peak viewing hours.

by Jon Brodkin - Nov 20 2014, 7:00am EST

• **Sliding-window protocol**
  Uses sequence numbers, acknowledgements, and timeouts to ensure exactly-once delivery; allows $W$ packets on the wire at once to improve utilization

• **Setting the window size**
  $W$ should be at or slightly above (depending on loss) the bandwidth-delay product of the network; this keeps the network utilized without building excessive queues