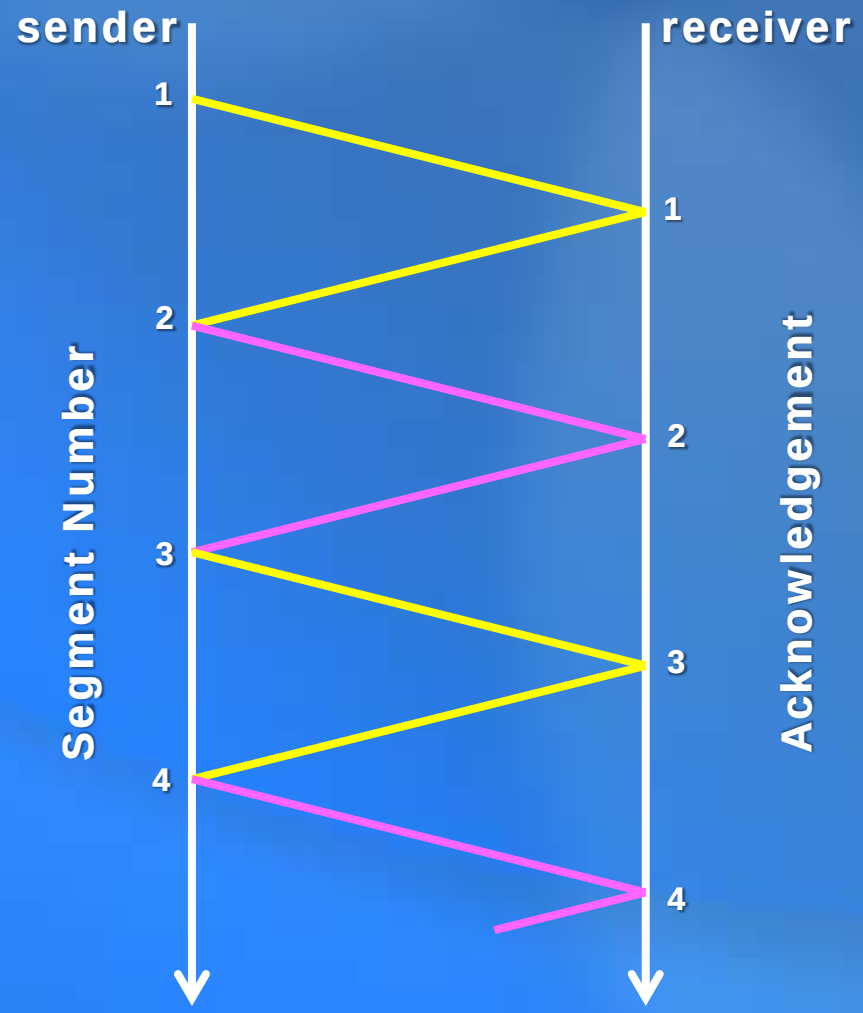
Sliding Window Protocol

Stop-and-wait



The stop-and-wait protocol is relatively simple, but has a low throughput.

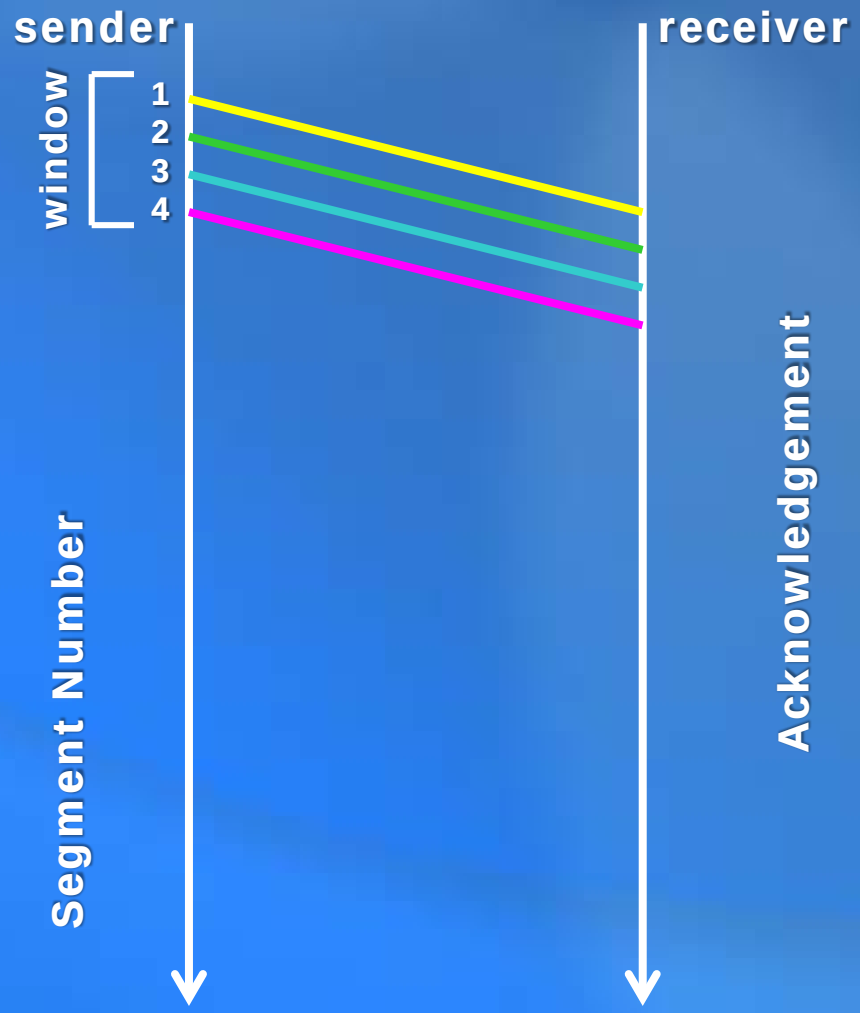
If no losses, $\text{Throughput} = \frac{1}{\text{RTT}}$

where RTT = round-trip time

If loss probability L, $\text{Throughput} = \frac{1 - L}{\text{RTT}}$

Assuming that the time required to transmit a packet is much shorter than RTT, the network is idle most of the time.

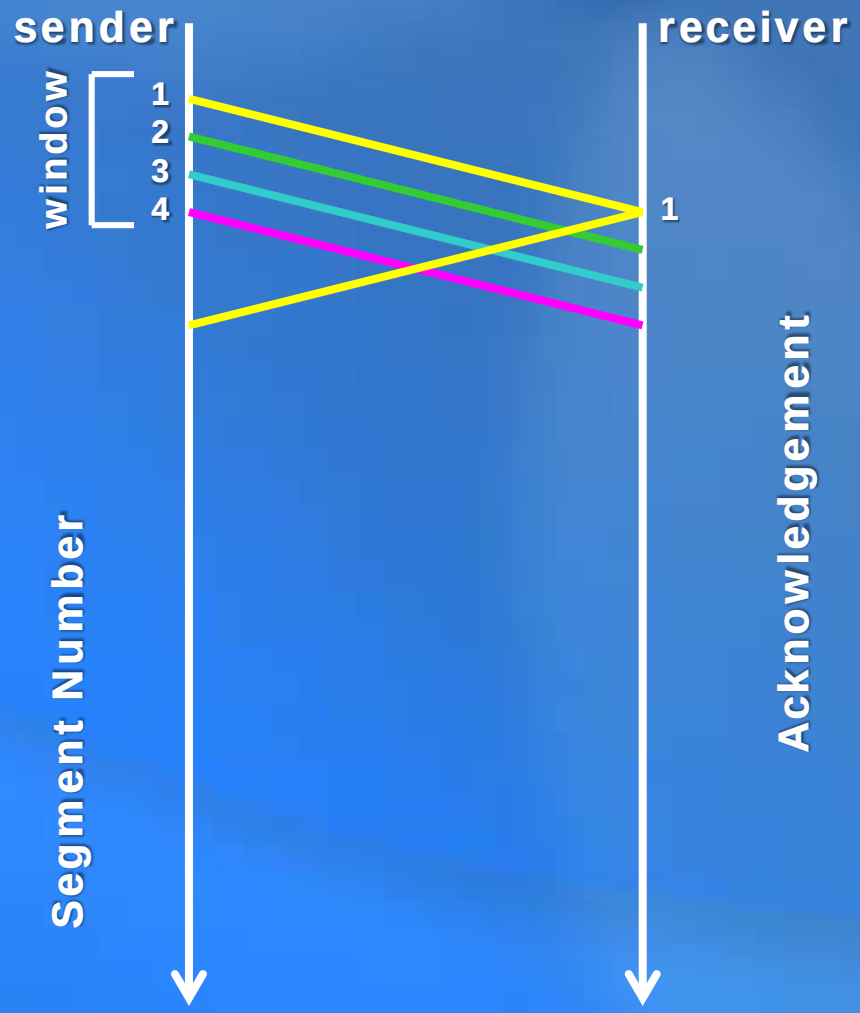
Sliding Window (no losses)



Sender allows for W unacknowledged packets in the network at the same time.

W = window size
(W = 4 in this example)

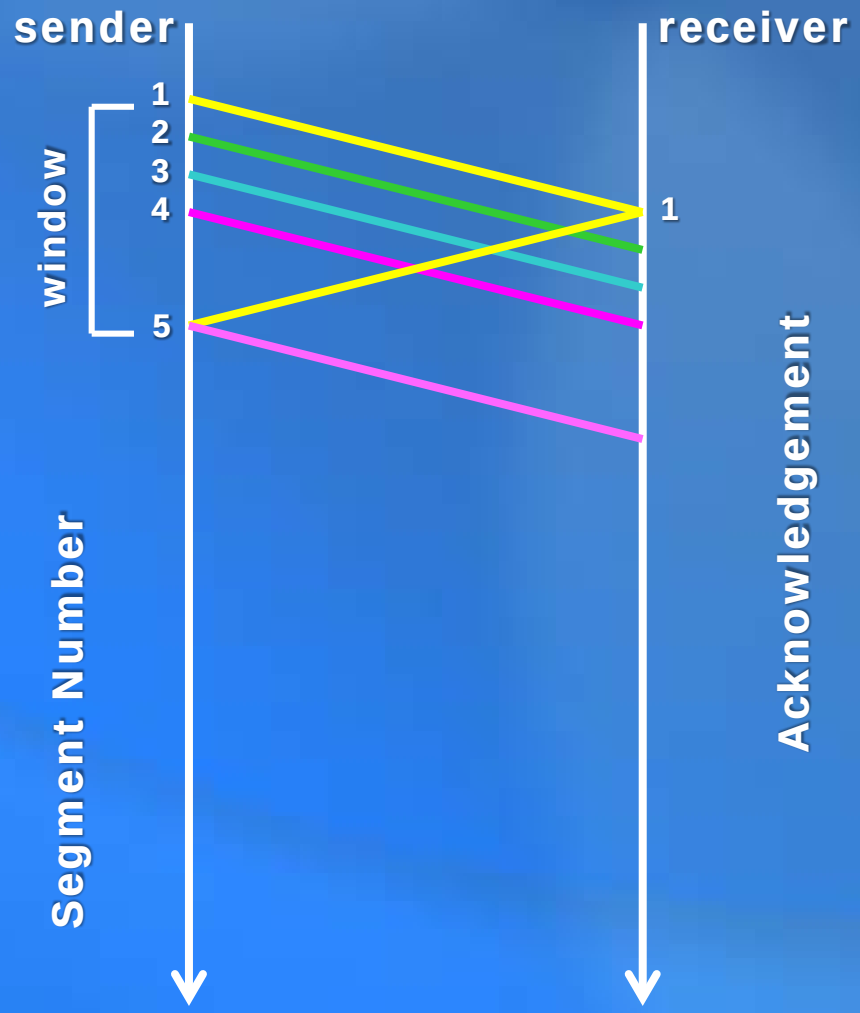
Sliding Window (no losses)



Sender allows for W unacknowledged packets in the network at the same time.

W = window size
(W = 4 in this example)

Sliding Window (no losses)

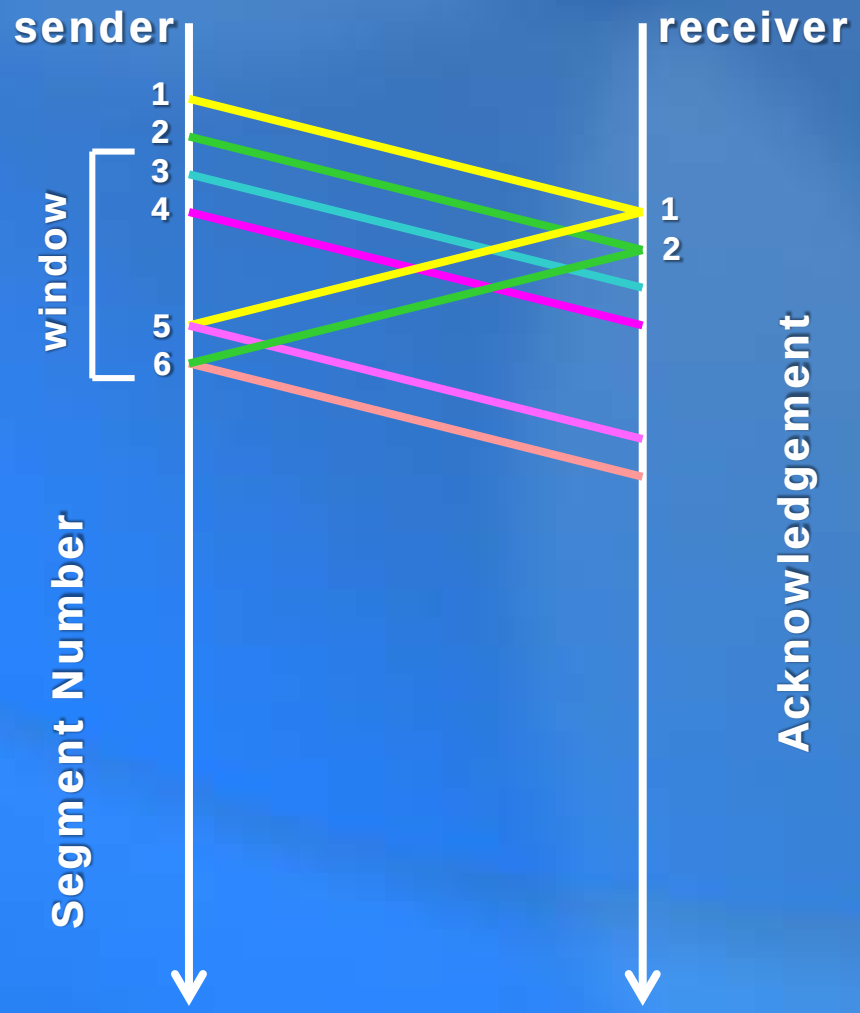


Sender allows for W unacknowledged packets in the network at the same time.

W = window size
(W = 4 in this example)

Sender advances window by one for each acknowledged packet, i.e. the window slides.

Sliding Window (no losses)

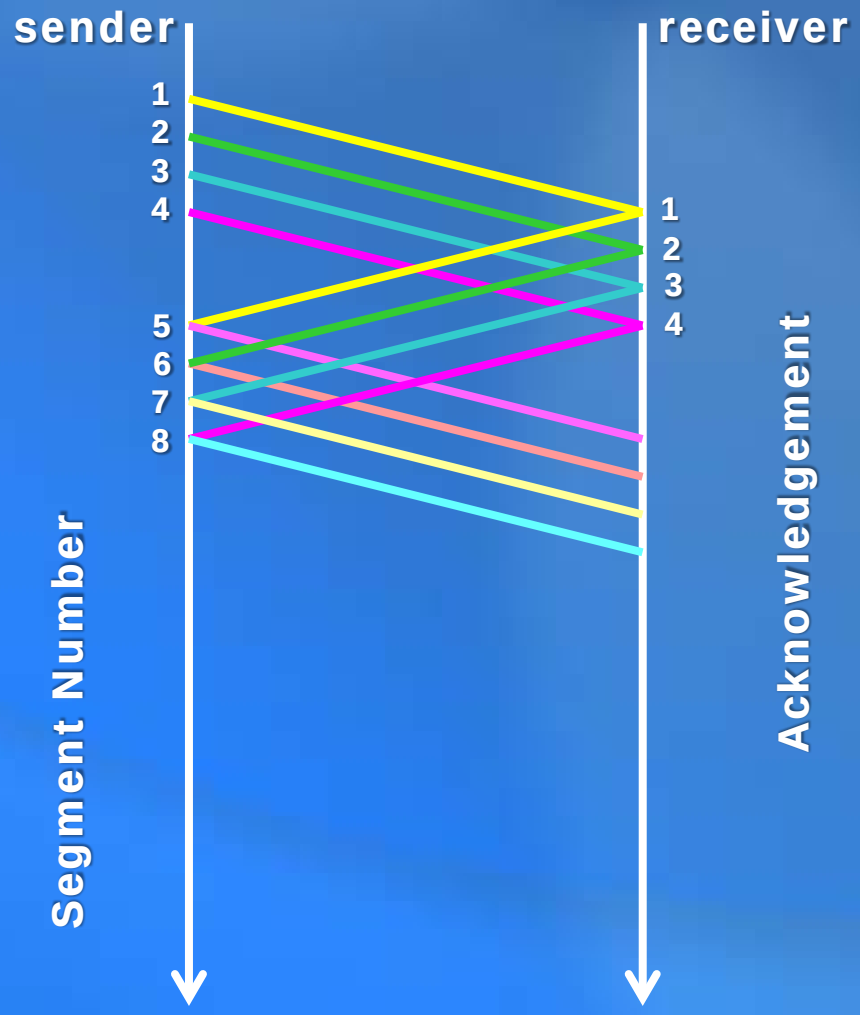


Sender allows for W unacknowledged packets in the network at the same time.

W = window size
(W = 4 in this example)

Sender advances window by one for each acknowledged packet, i.e. the window slides.

Sliding Window (no losses)

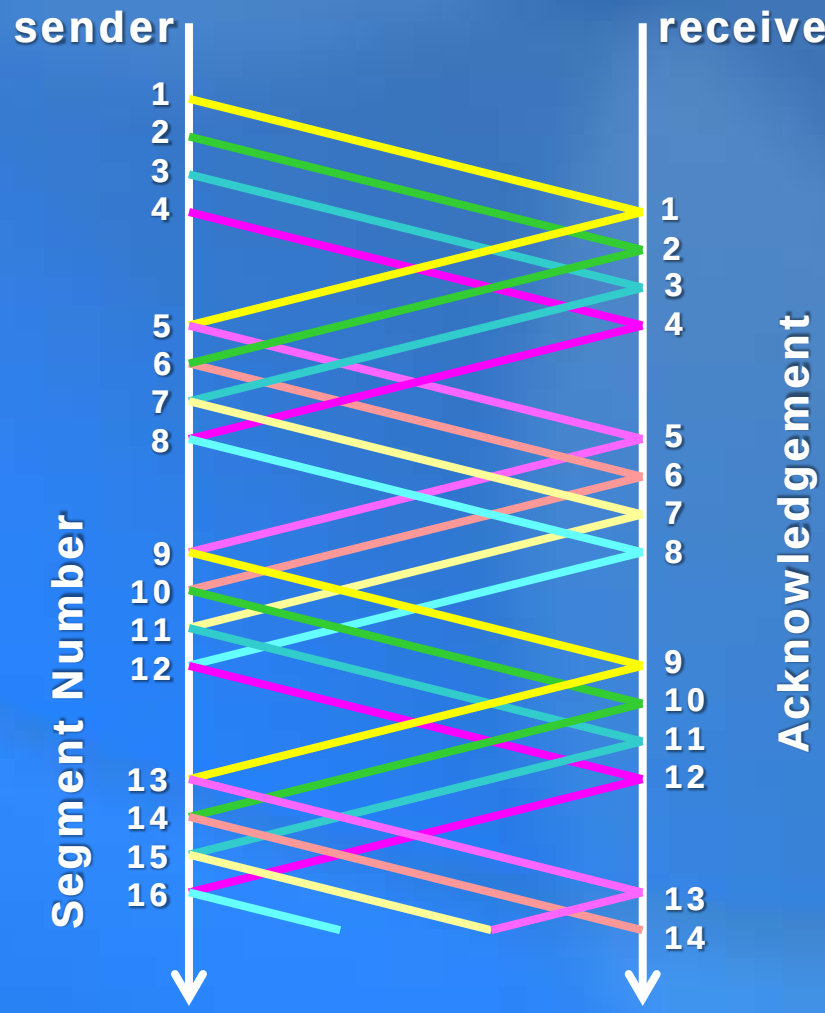


Sender allows for W unacknowledged packets in the network at the same time.

W = window size
(W = 4 in this example)

Sender advances window by one for each acknowledged packet, i.e. the window slides.

Sliding Window (no losses)



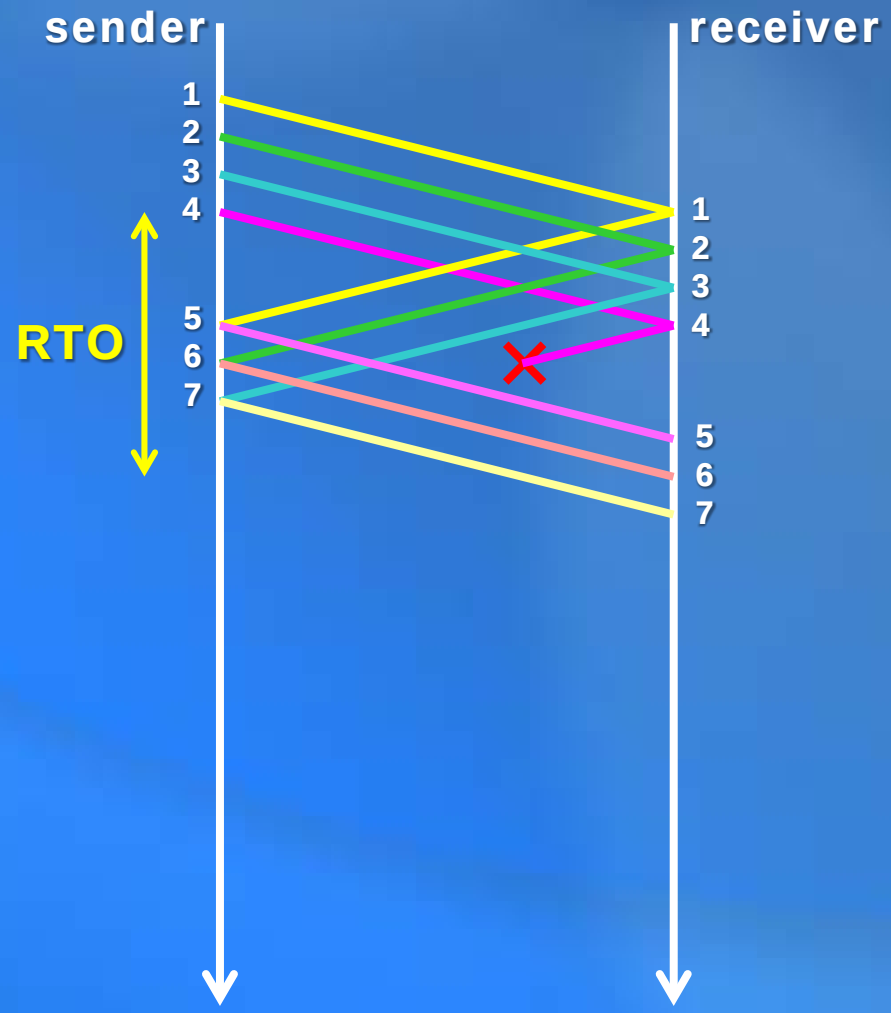
Sender allows for W unacknowledged packets in the network at the same time.

W = window size
(W = 4 in this example)

Sender advances window by one for each acknowledged packet, i.e. the window slides.

This practice of overlapping multiple transmissions is known as **pipelining**.

Handling Packet Loss at Sender

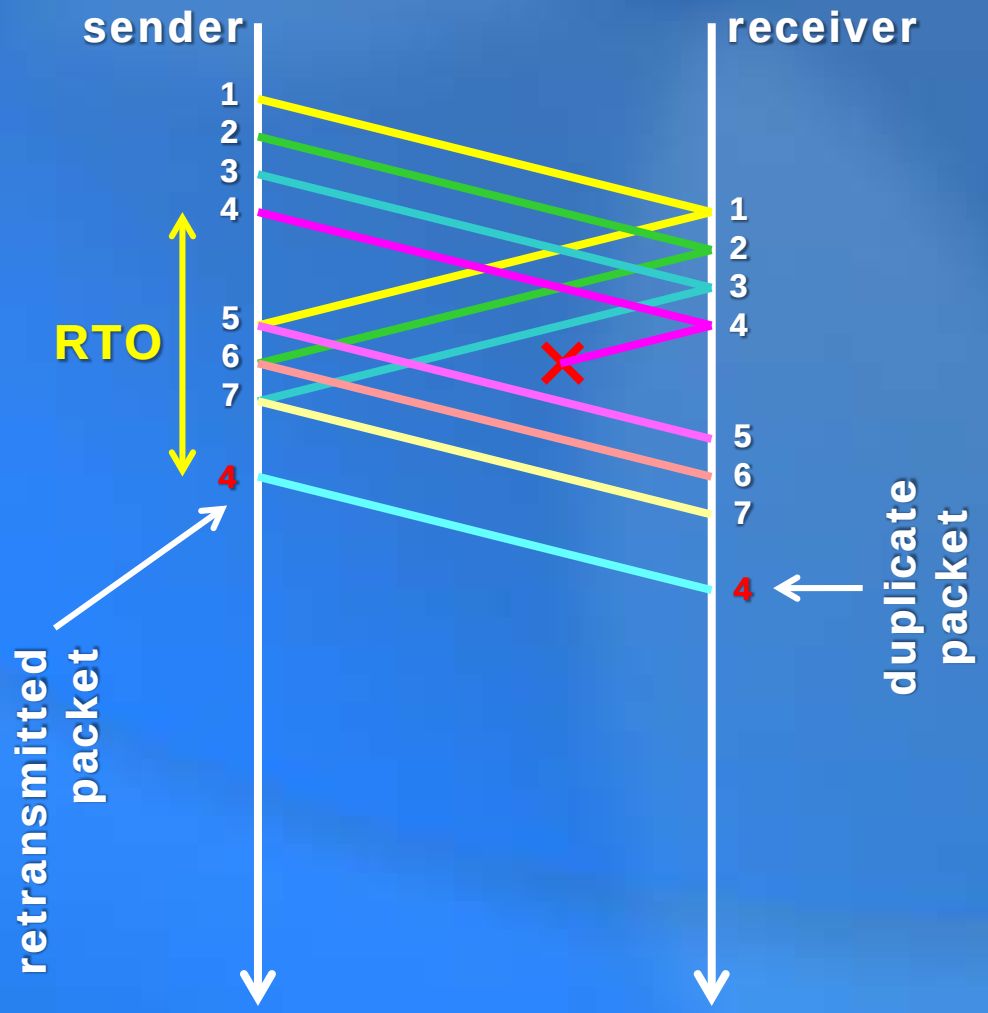


Sender keeps a list of unacknowledged packets and when they were sent.

It periodically checks to see if any packets in the list were sent a “long time” ago (longer than RTO, the retransmission timeout).

If so, it retransmits the packet, and updates the transmit time of the packet.

Handling Packet Loss at Sender

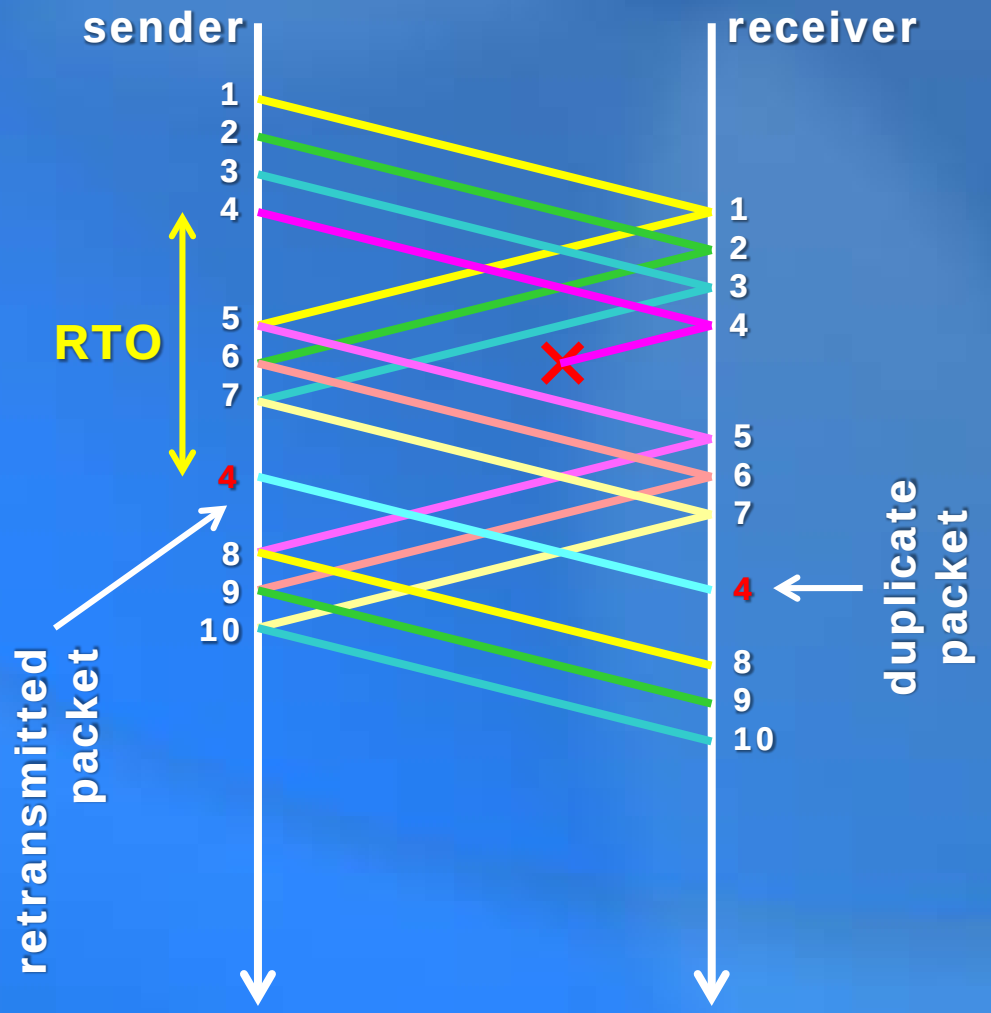


Sender keeps a list of unacknowledged packets and when they were sent.

It periodically checks to see if any packets in the list were sent a “long time” ago (longer than RTO, the retransmission timeout).

If so, it retransmits the packet, and updates the transmit time of the packet.

Handling Packet Loss at Sender

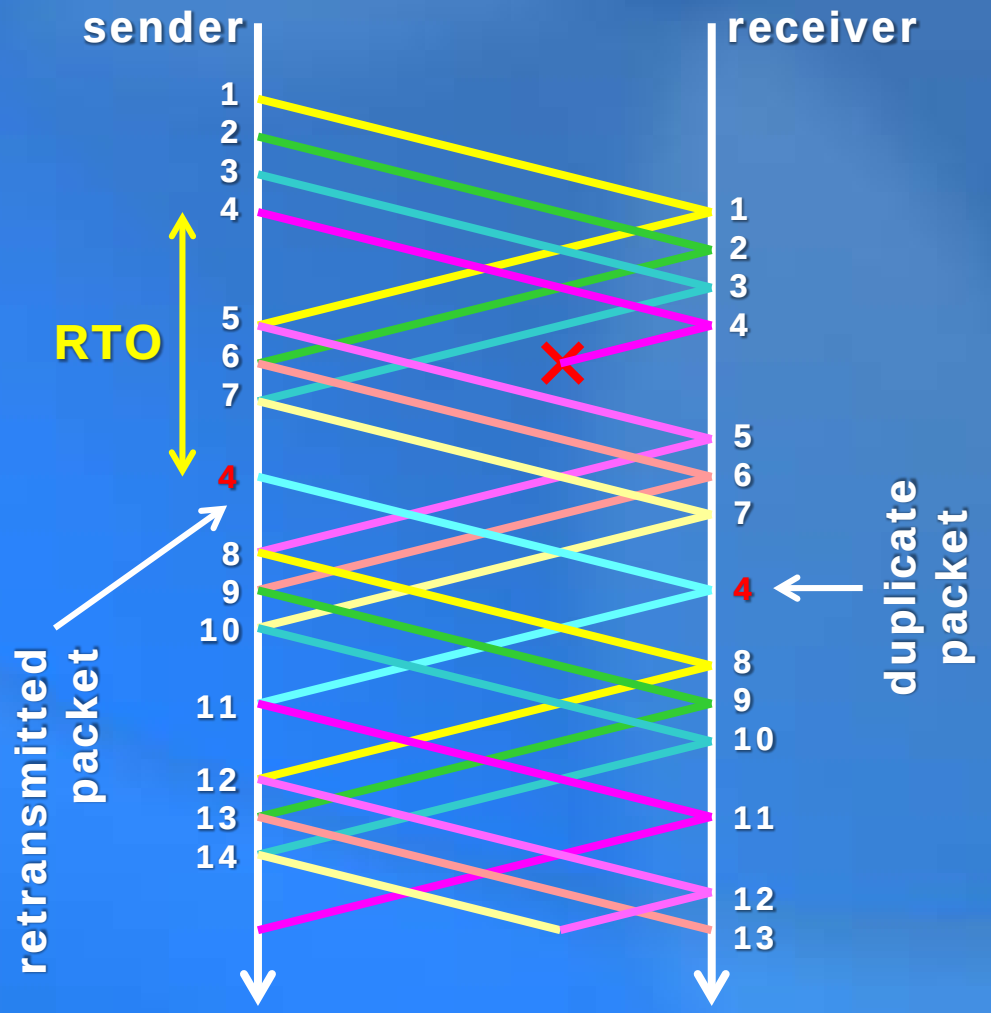


Sender keeps a list of unacknowledged packets and when they were sent.

It periodically checks to see if any packets in the list were sent a “long time” ago (longer than RTO, the retransmission timeout).

If so, it retransmits the packet, and updates the transmit time of the packet.

Handling Packet Loss at Sender

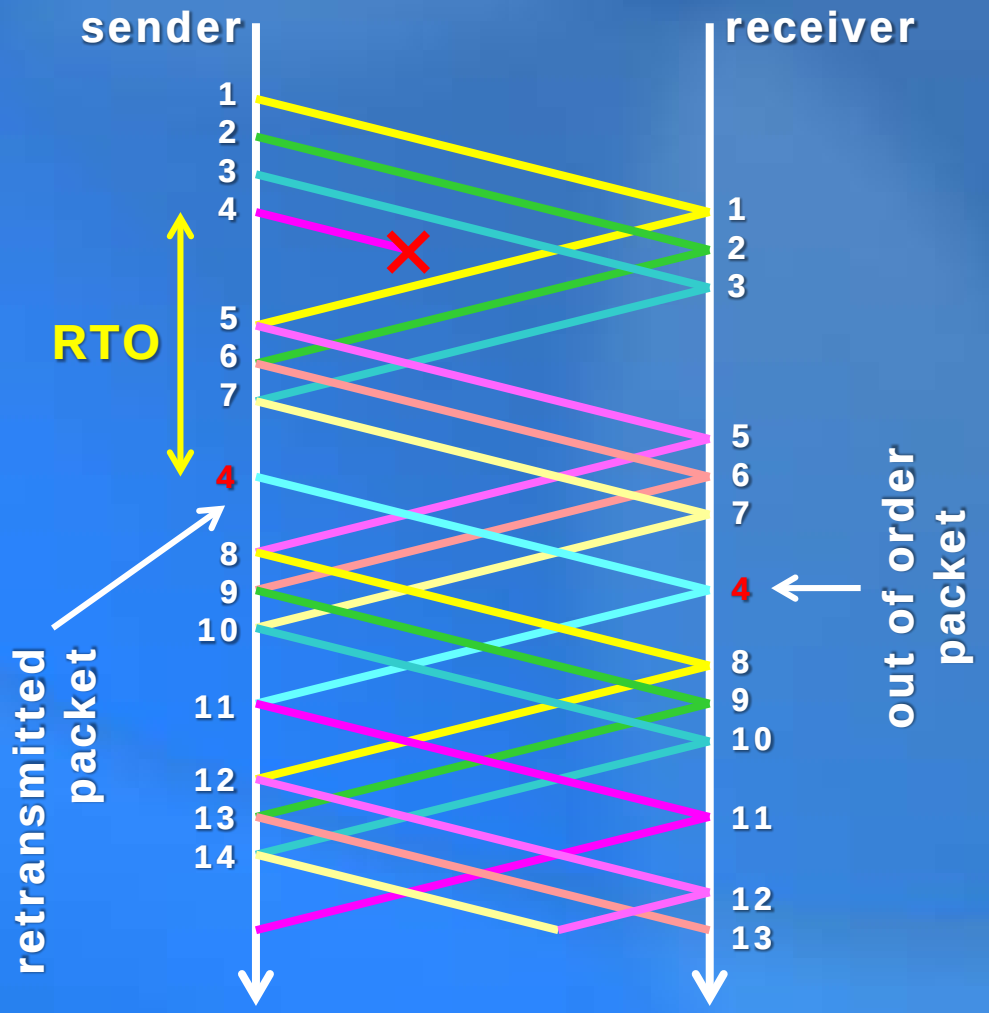


Sender keeps a list of unacknowledged packets and when they were sent.

It periodically checks to see if any packets in the list were sent a “long time” ago (longer than RTO, the retransmission timeout).

If so, it retransmits the packet, and updates the transmit time of the packet.

Handling Packet Loss at Receiver



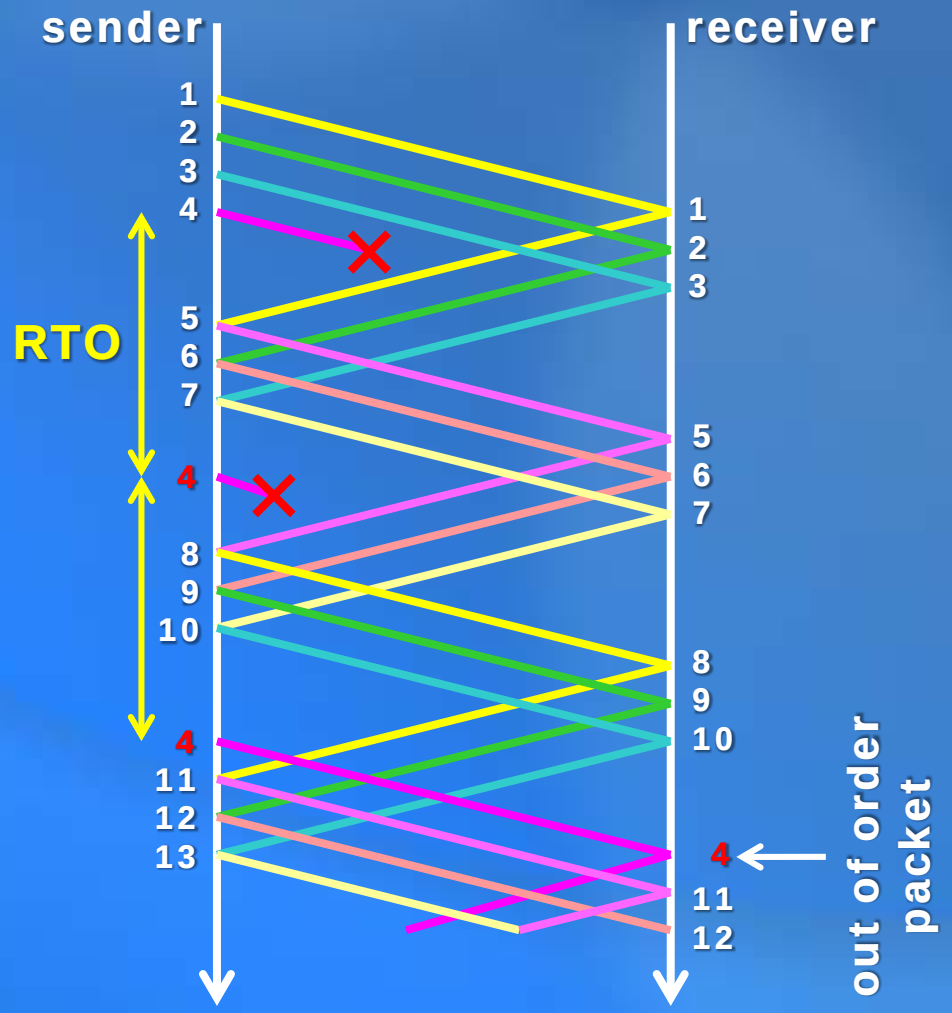
Packet loss may cause packets at the receiver to

- appear out of order
- be duplicated

To deal with this, the receiver must

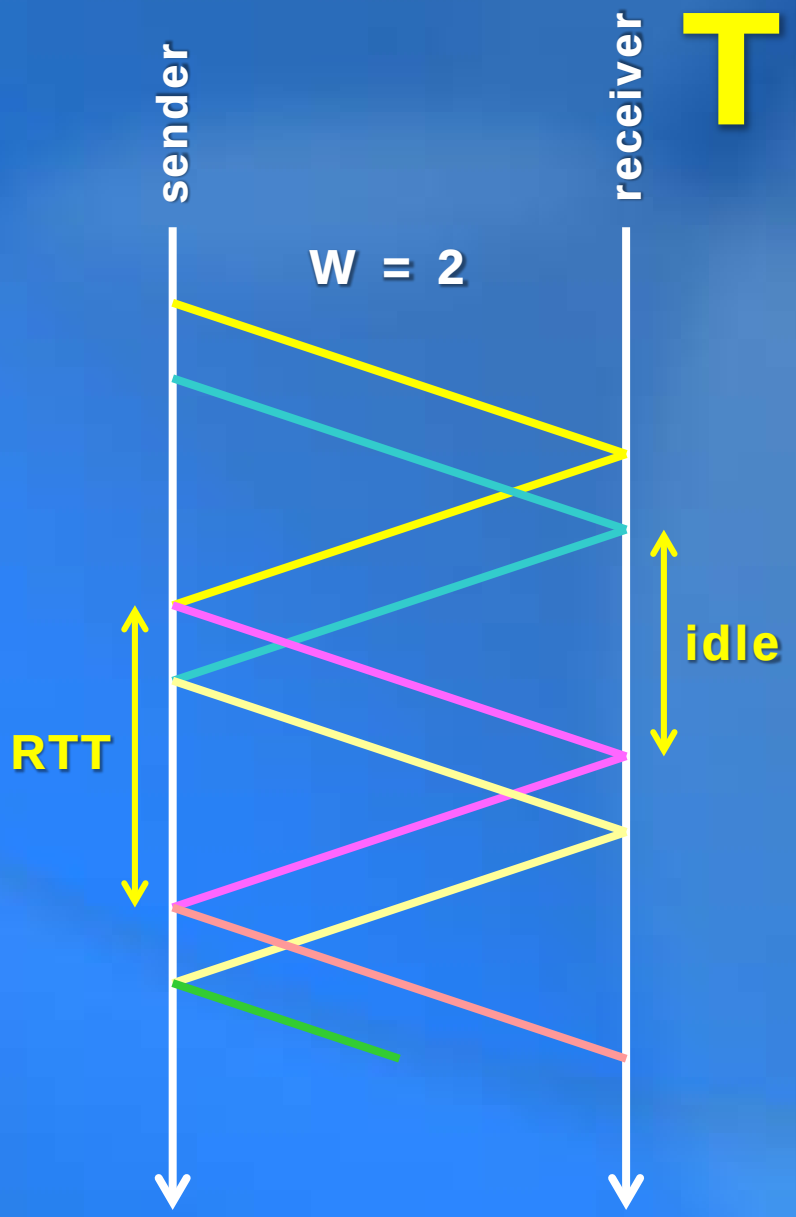
- Save received packets in a local buffer
- Discard duplicates
- Keep track of the next packet application expects.
- After each reception, deliver as many in-order packets as possible.

Size of Receiver Buffer



Since retransmissions may also be lost, there is no limit to the number of packets that must be stored at the receiver.

Throughput



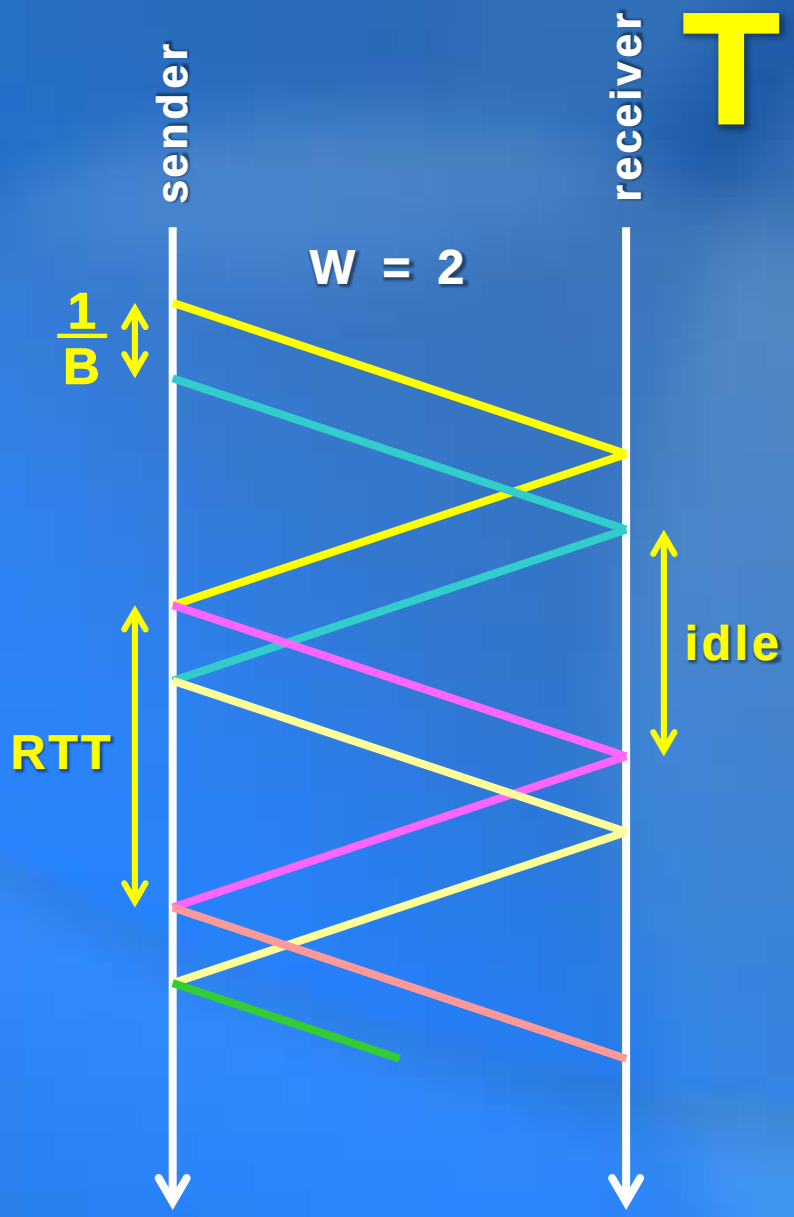
Assume no packet loss and constant RTT (round trip time).

Since, the sliding window protocol delivers W packets every RTT:

$$\text{Throughput} = \frac{W}{RTT}$$

To maximize throughput, we must minimize the idle time (increase W)

Throughput



However, we cannot increase W arbitrarily, since the time required to transmit a packet is limited by

B = rate of slowest link (packets/sec)

The maximum window size is limited by

$$W_{max} \times \frac{1}{B} = RTT$$

i.e. $W_{max} = B \cdot RTT$ packets

Thus, the maximum throughput is

$$\text{Throughput} = \frac{W}{RTT} \leq \frac{B \cdot RTT}{RTT} = B$$