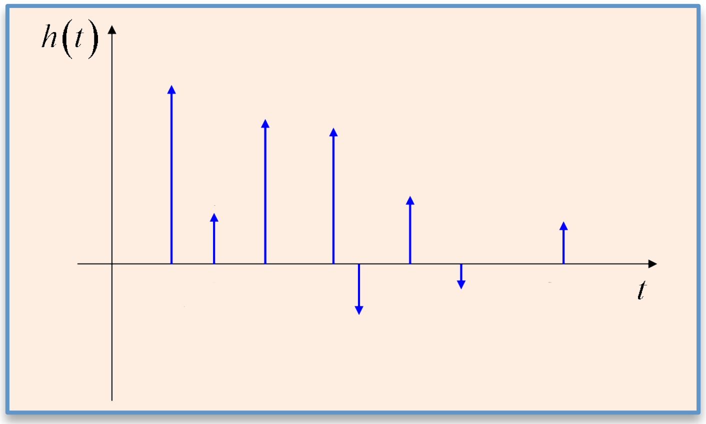
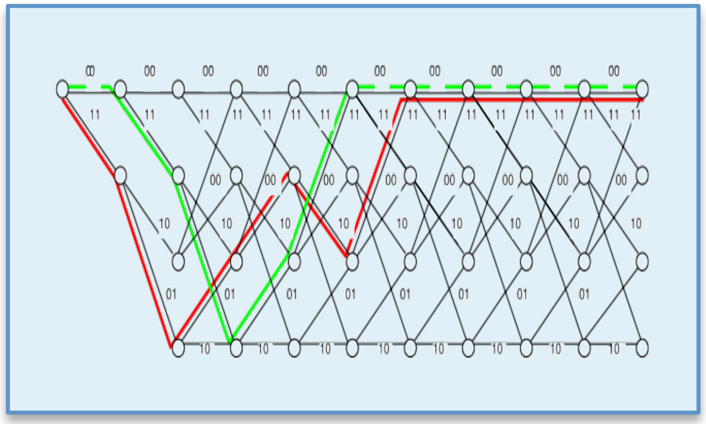
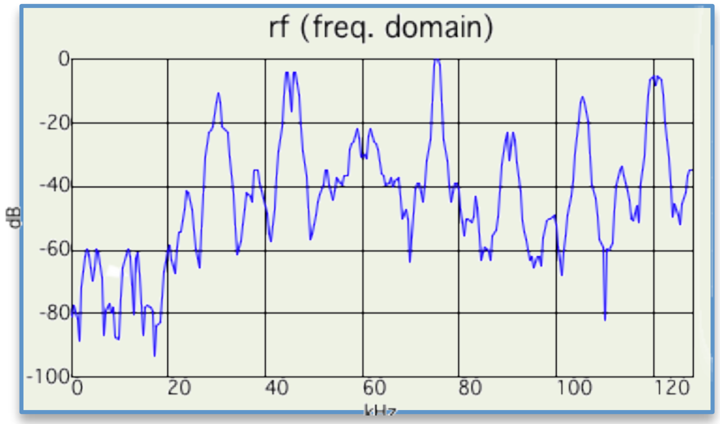
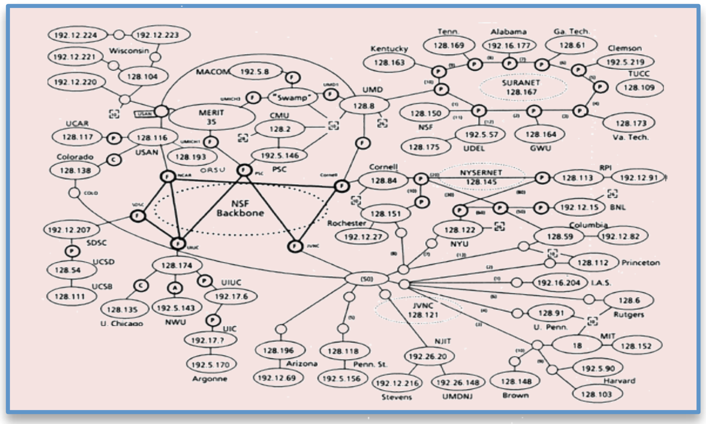








INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

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**

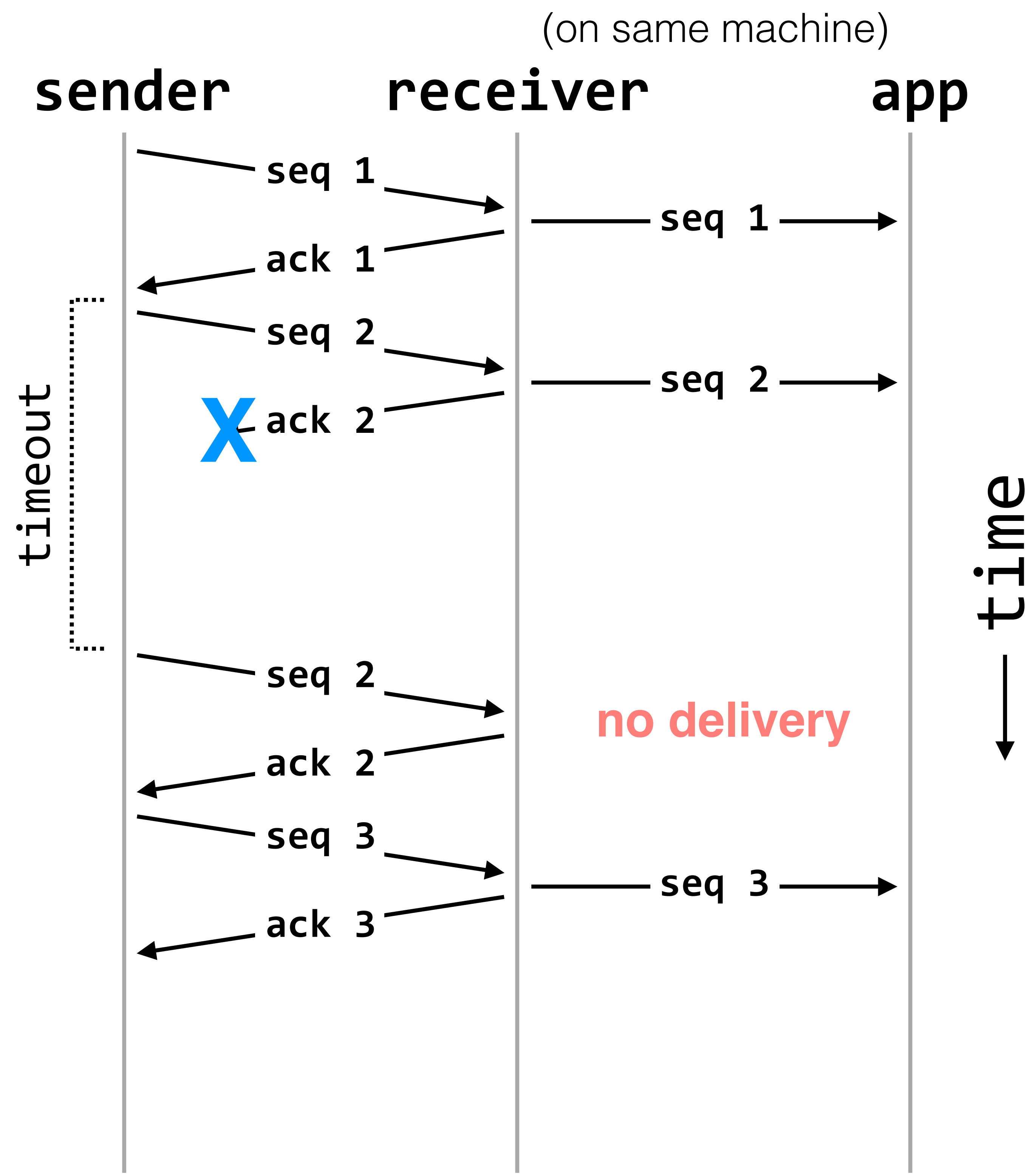
Recap: Stop-and-wait Protocol

At sender:

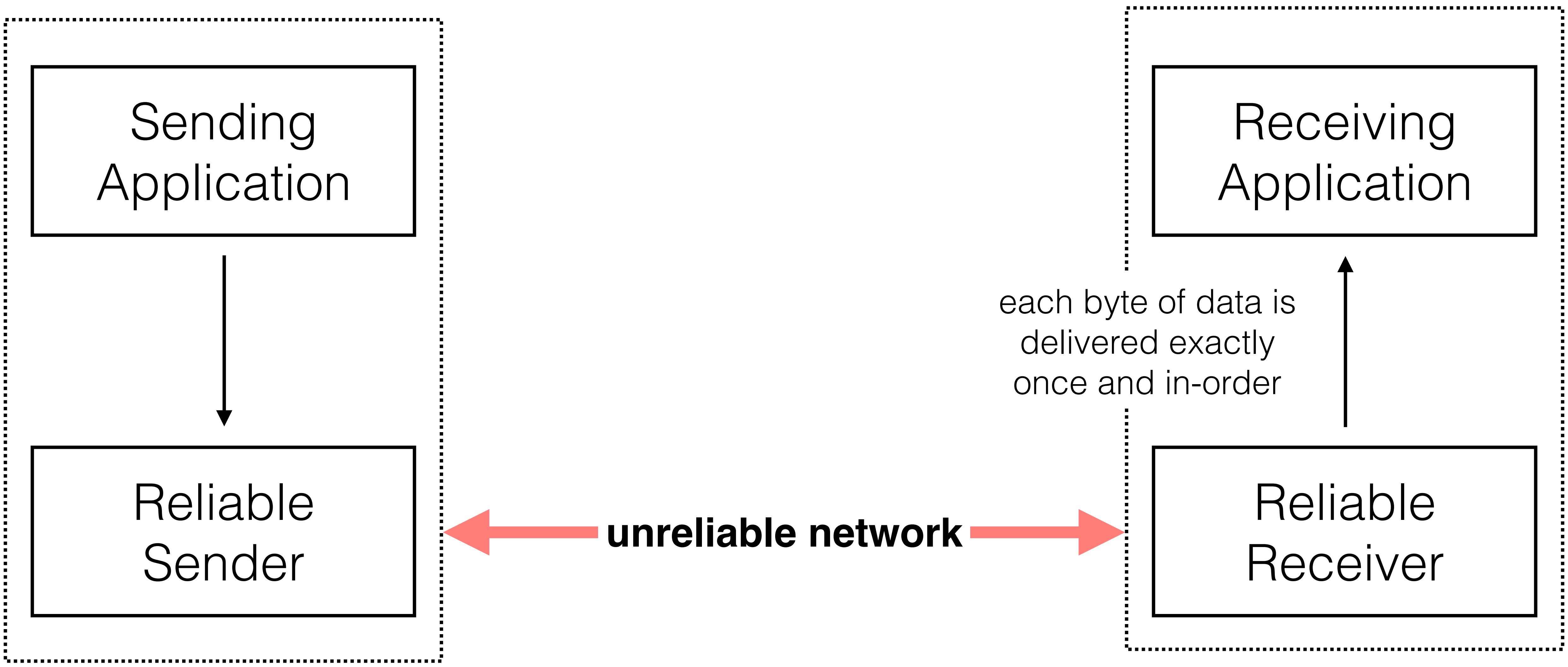
- Send a packet, keep track of its sequence number
- When an ACK is received for that packet, increment the stored sequence number and repeat
- If an ACK for the outstanding packet hasn't been received after `timeout` seconds, re-transmit the packet

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

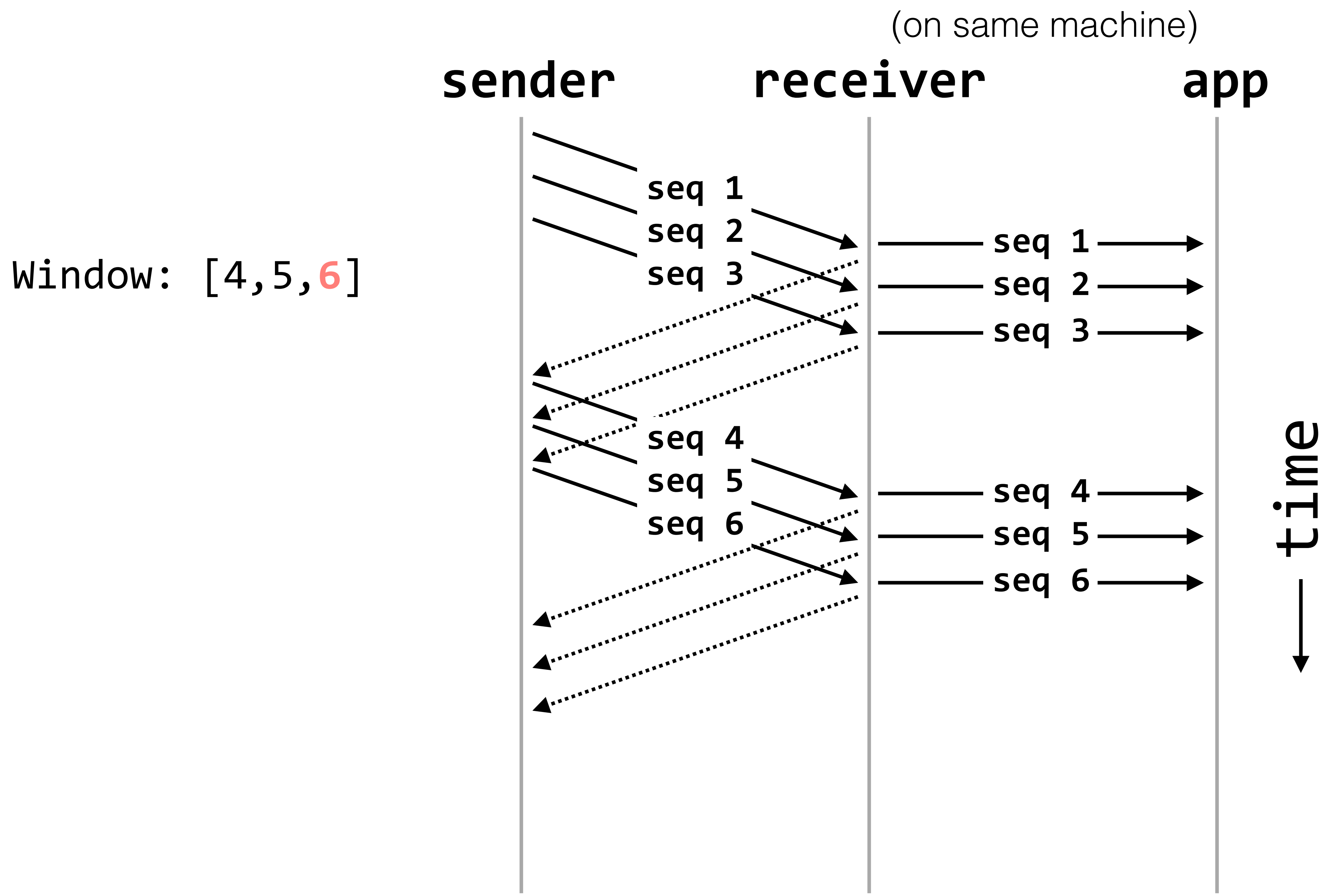


Reliable Communication



today's goal: develop a reliable transport protocol that gets better utilization than the stop-and-wait protocol

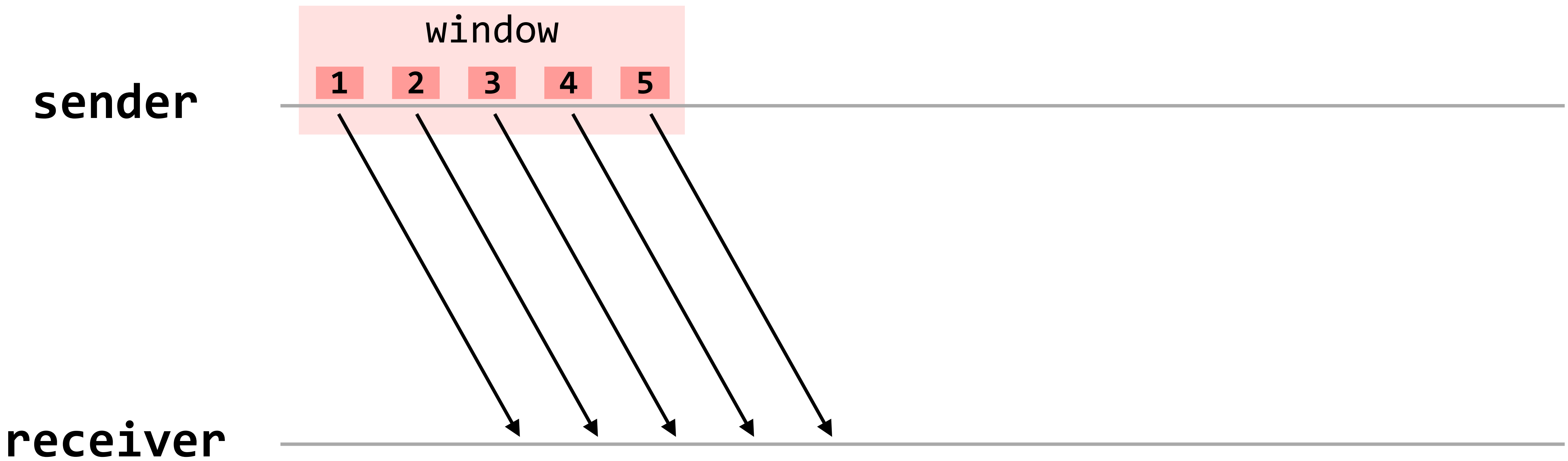
Sliding-window Protocol



basic idea: send a new packet whenever an ACK is received, allowing no more than `W` outstanding packets at a time

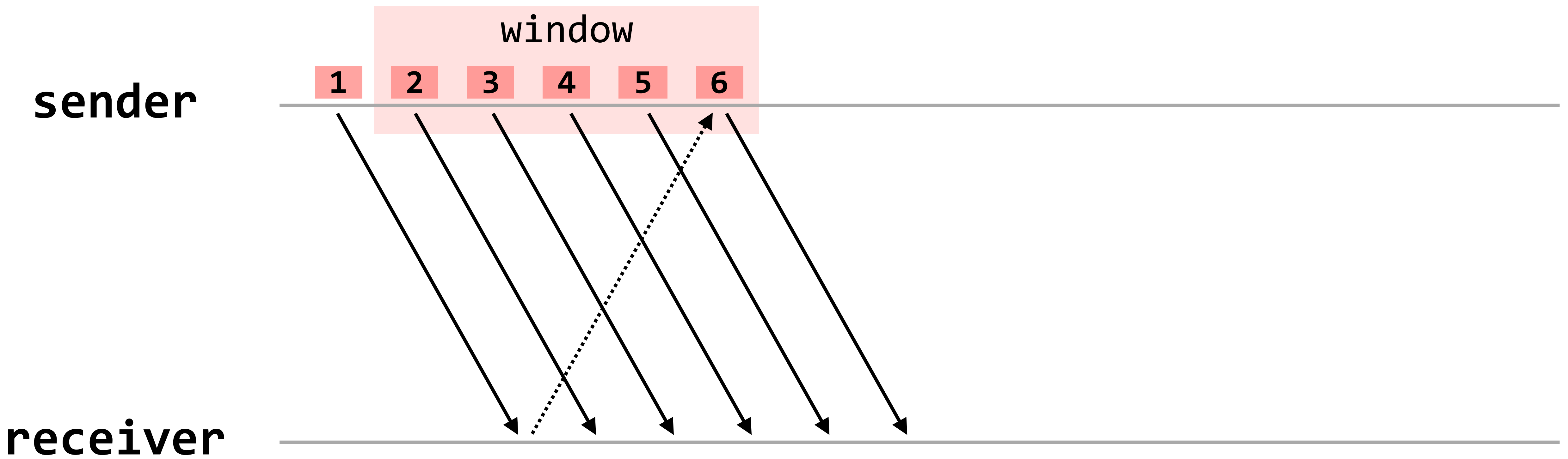
Sliding-window Protocol

(same protocol, different visualization)



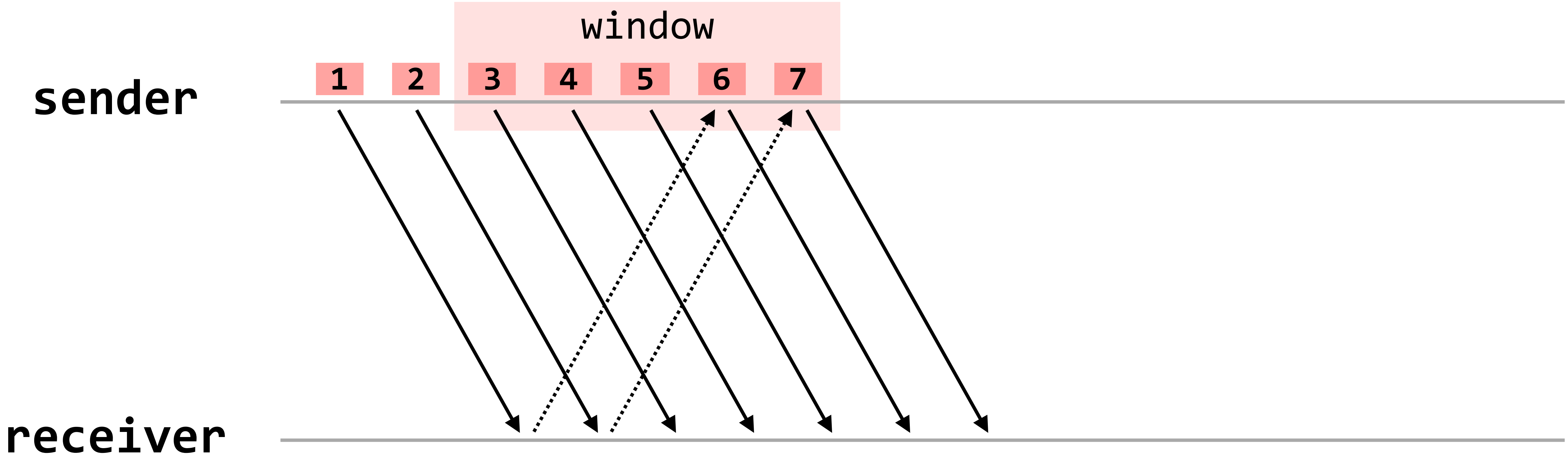
Sliding-window Protocol

(same protocol, different visualization)



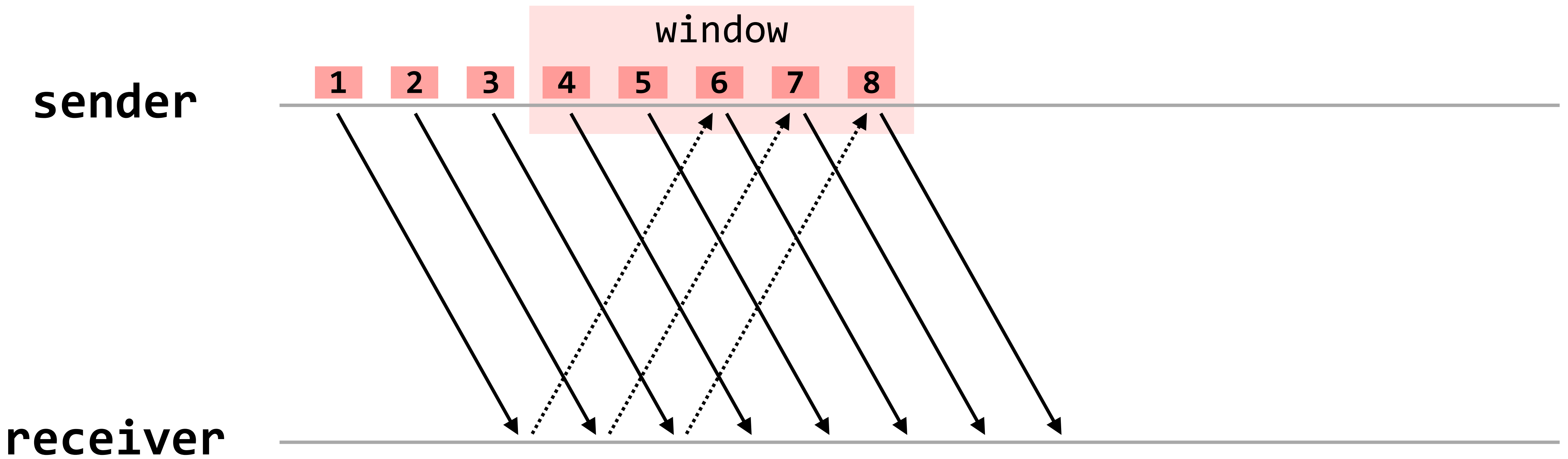
Sliding-window Protocol

(same protocol, different visualization)

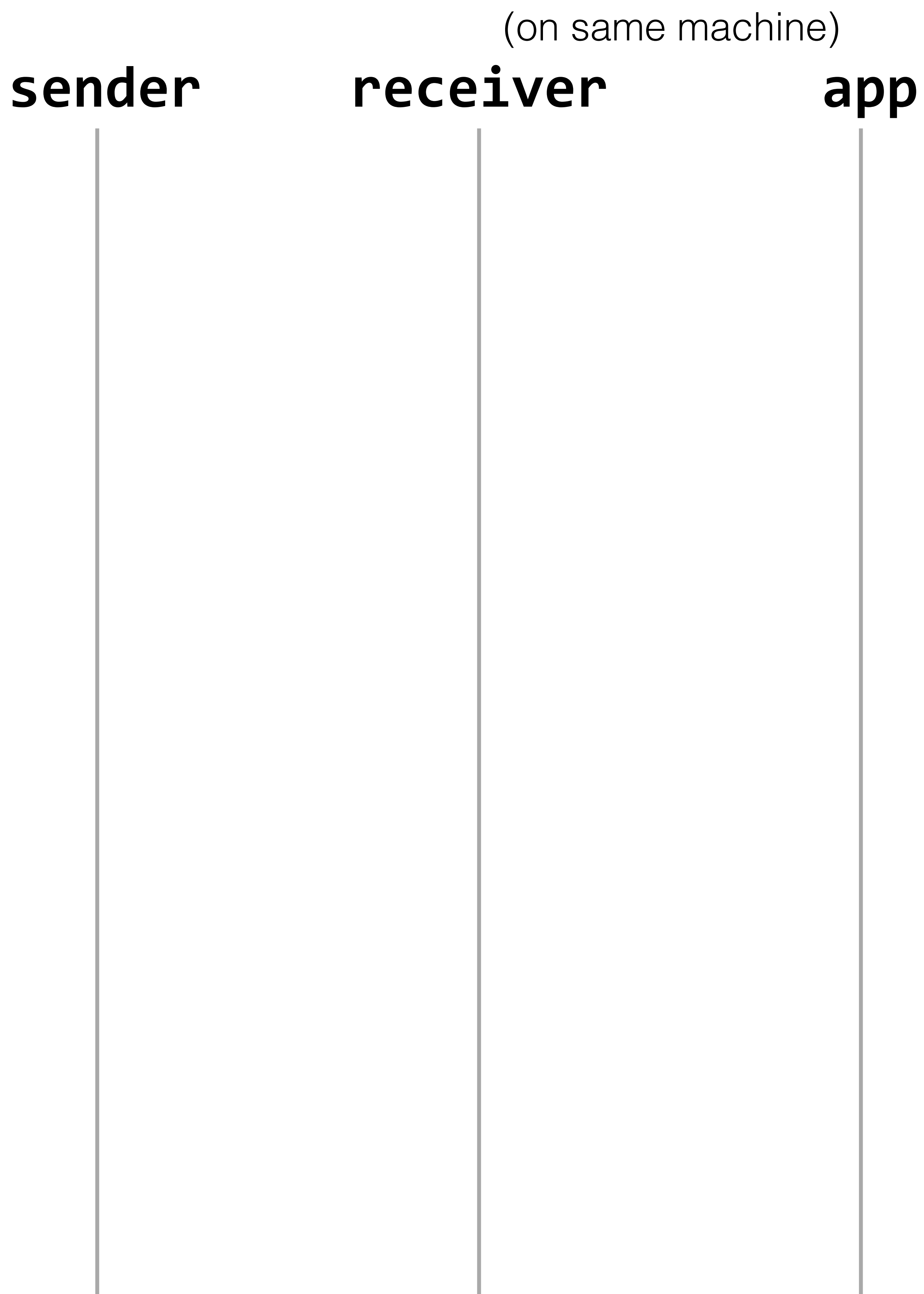


Sliding-window Protocol

(same protocol, different visualization)

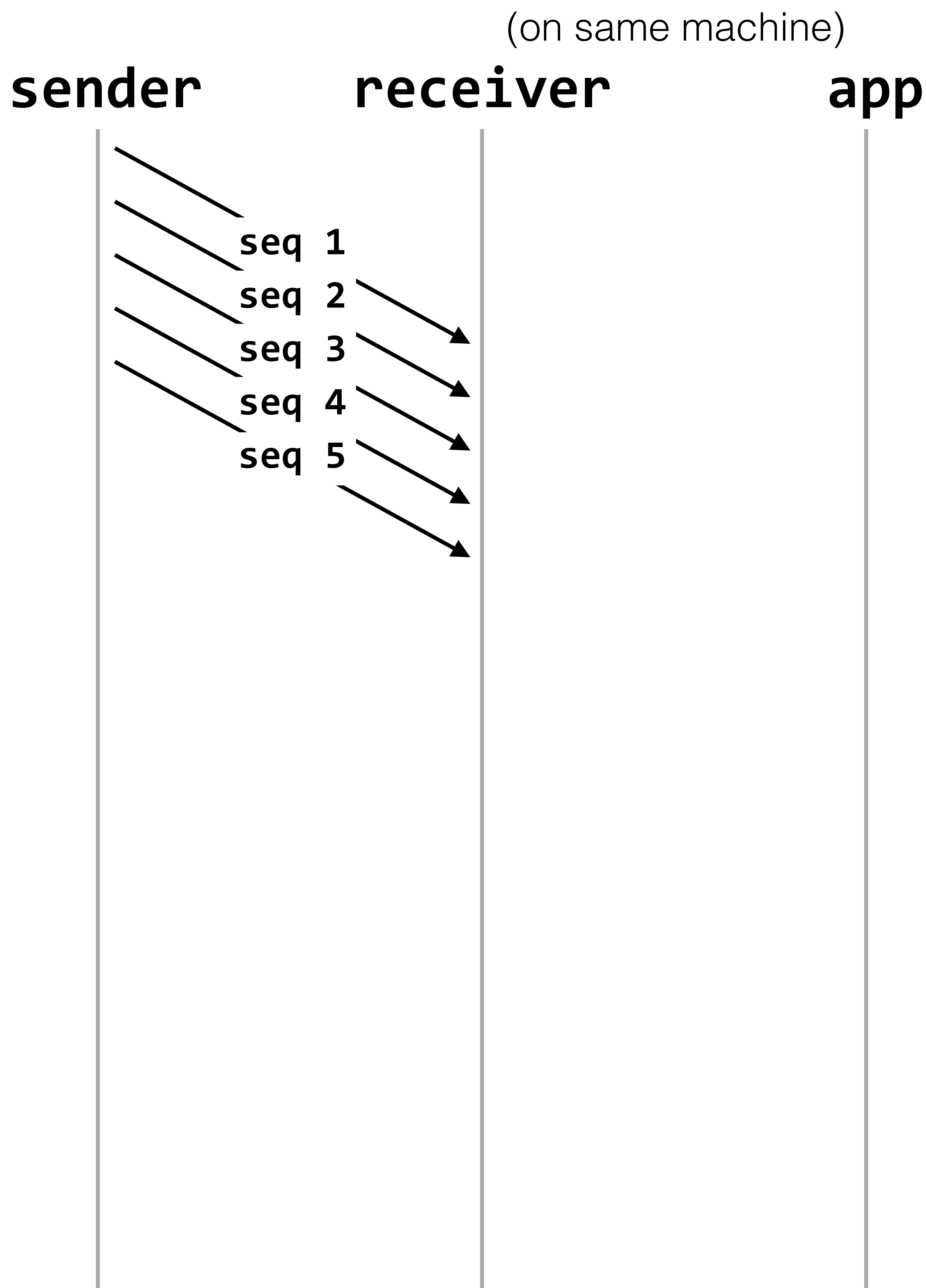


Sliding-window Protocol: Sender



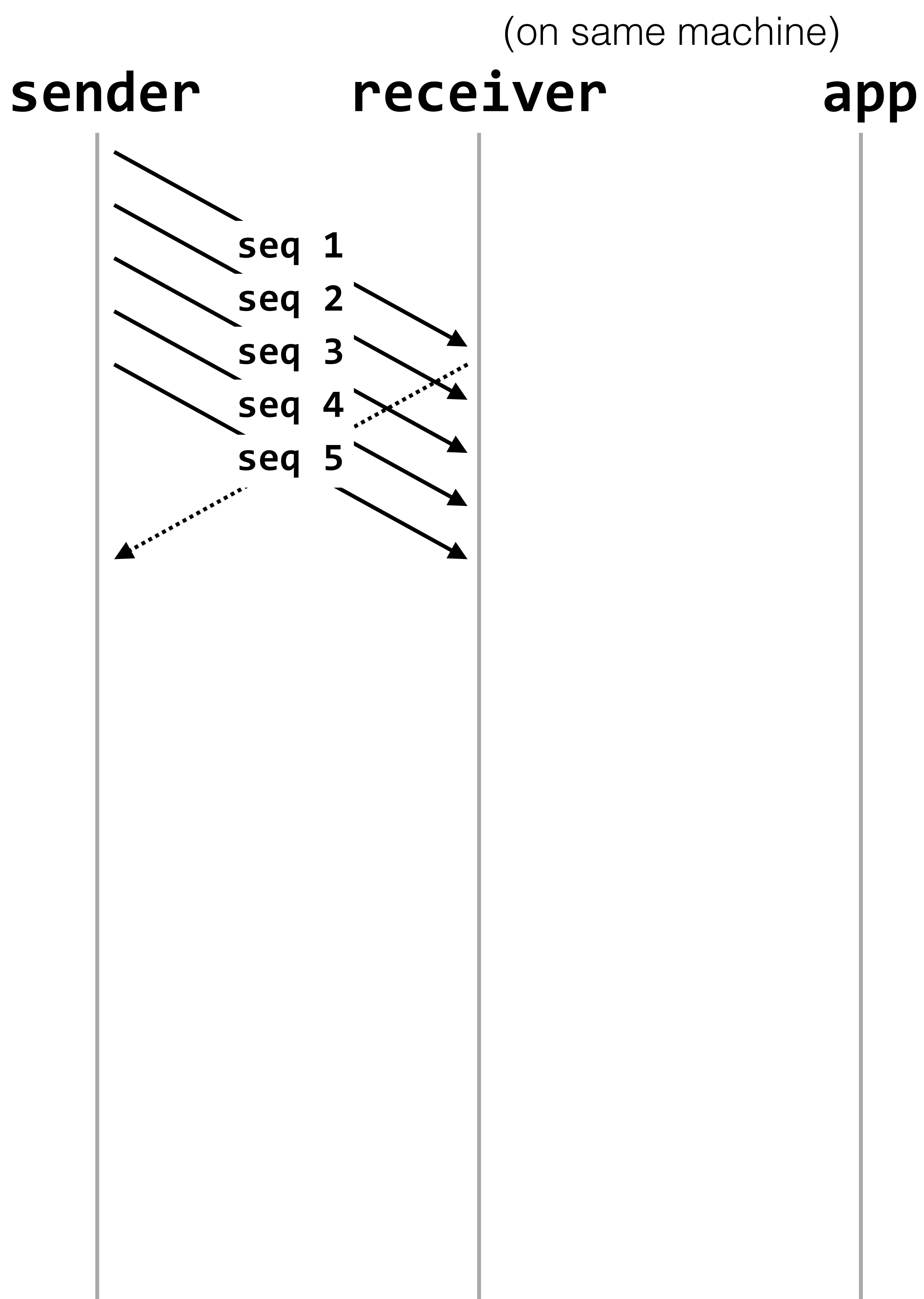
Sliding-window Protocol: Sender

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



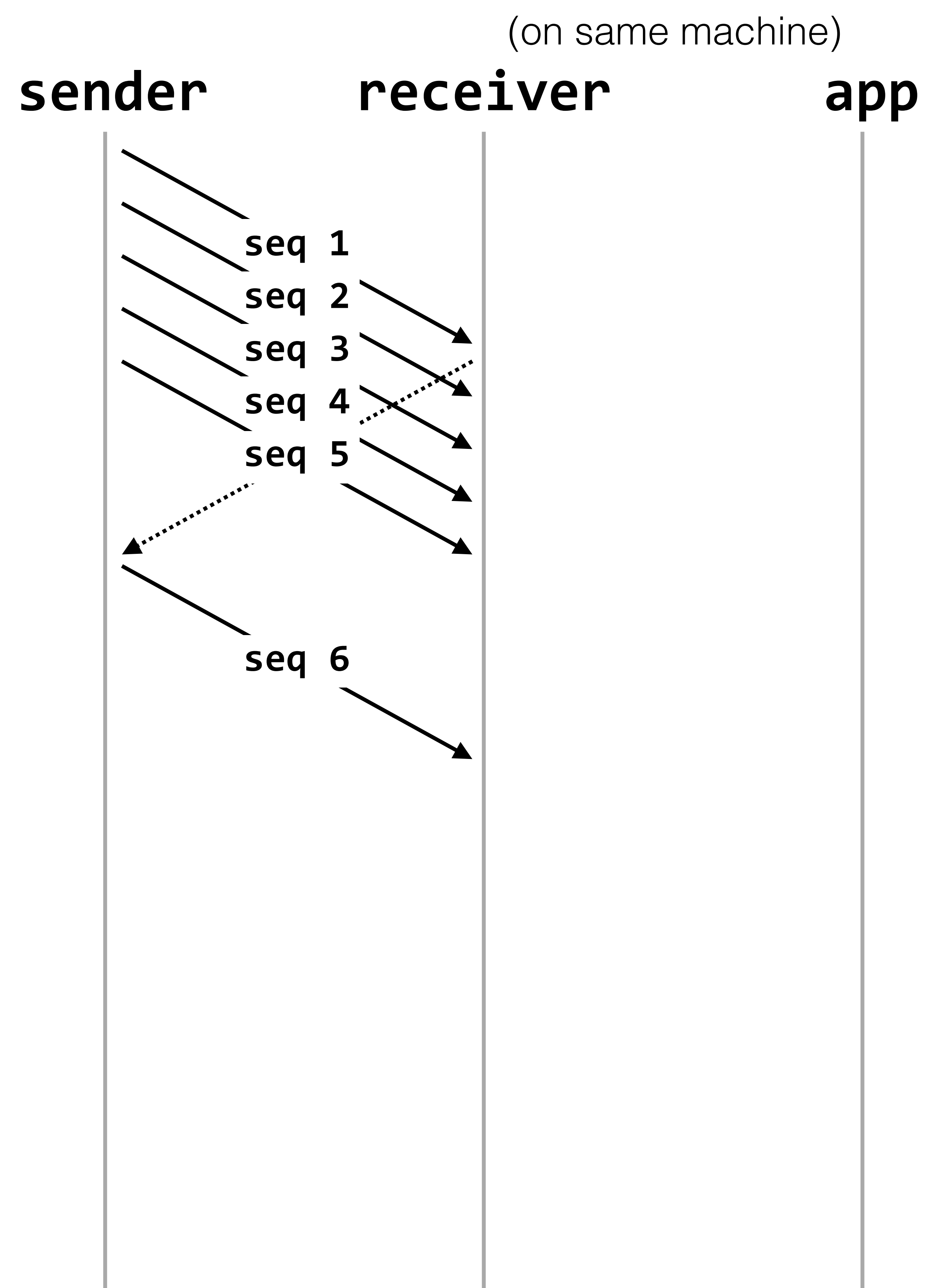
Sliding-window Protocol: Sender

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



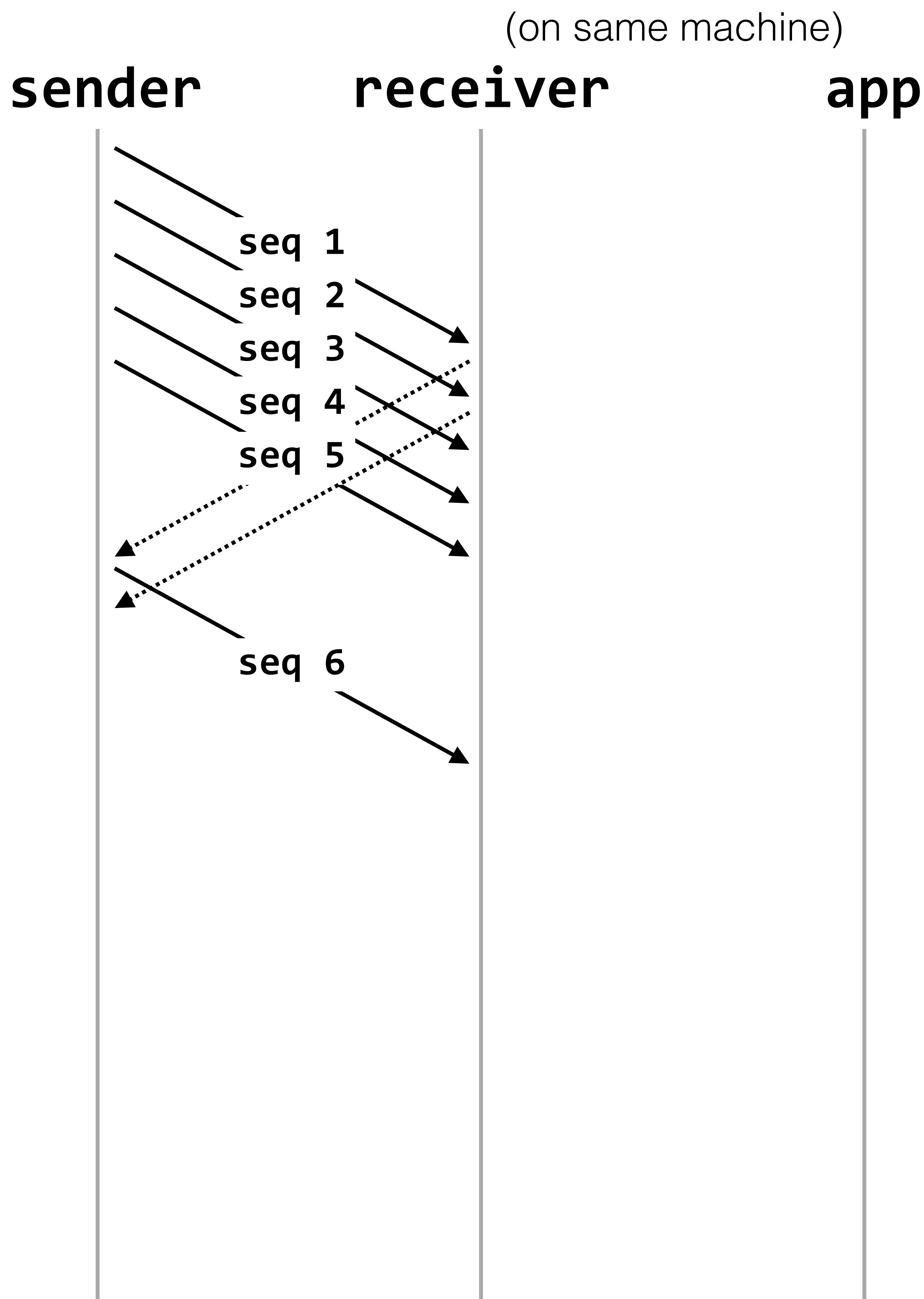
Sliding-window Protocol: Sender

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



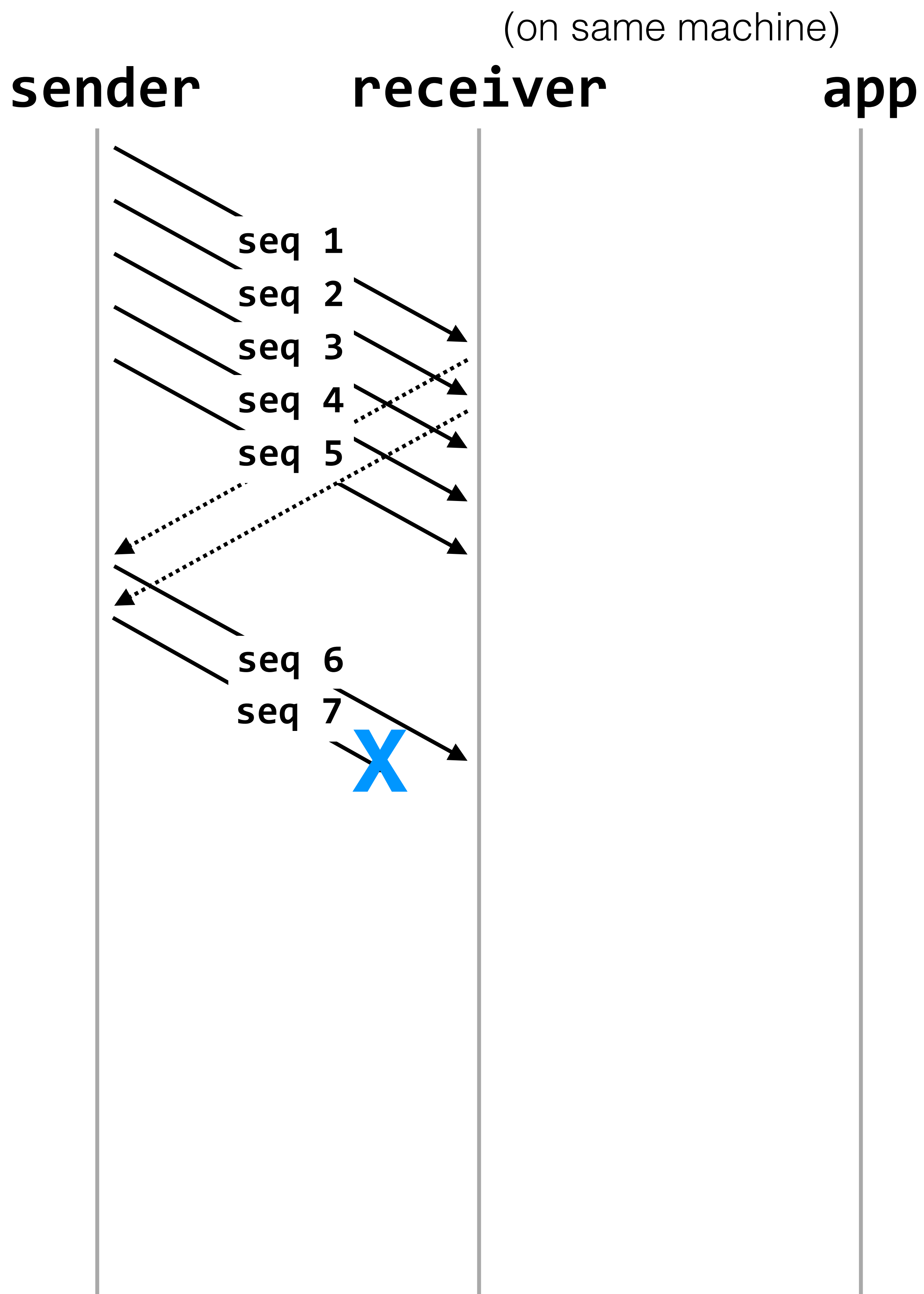
Sliding-window Protocol: Sender

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



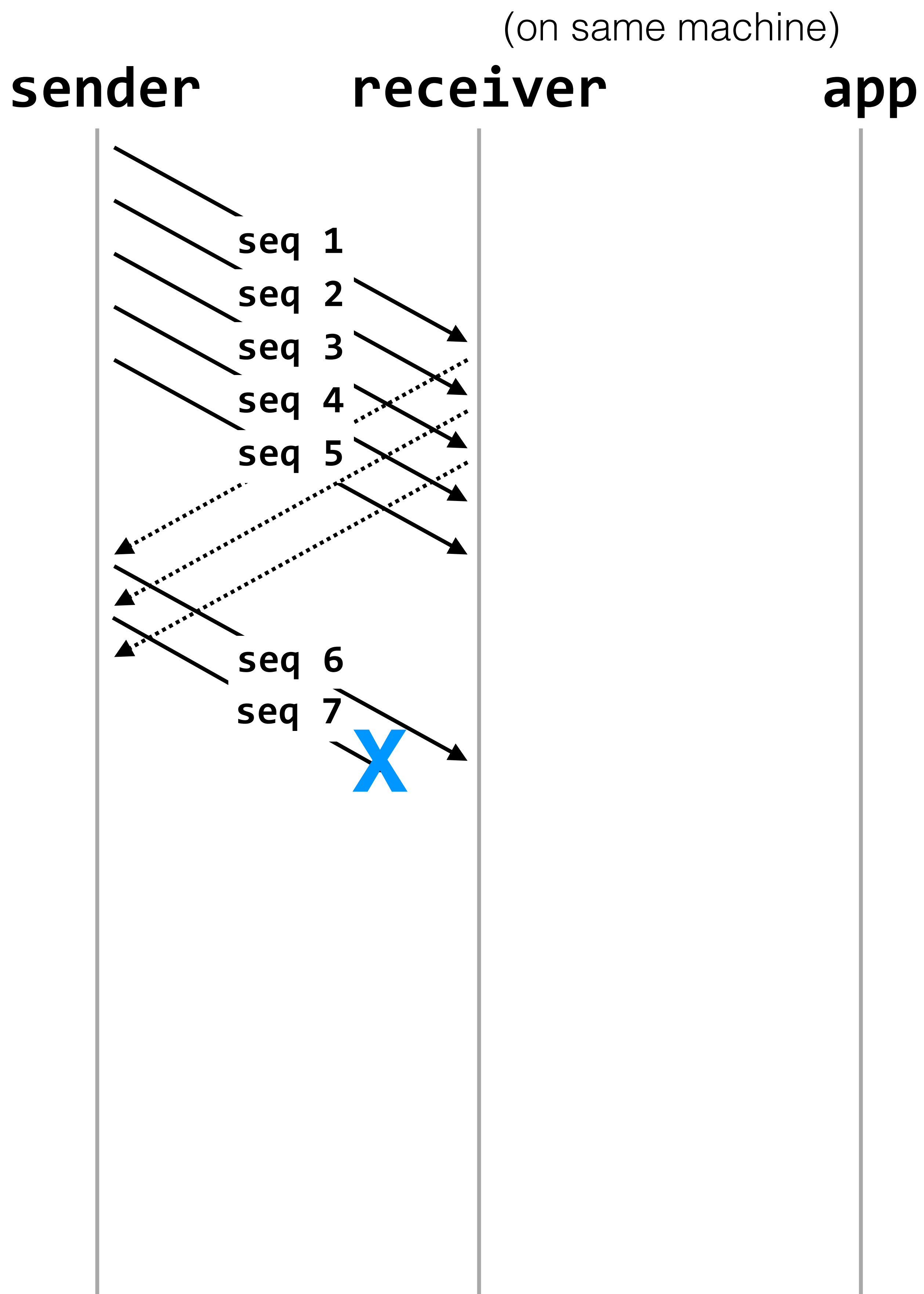
Sliding-window Protocol: Sender

Window: [3,4,5,6,7]



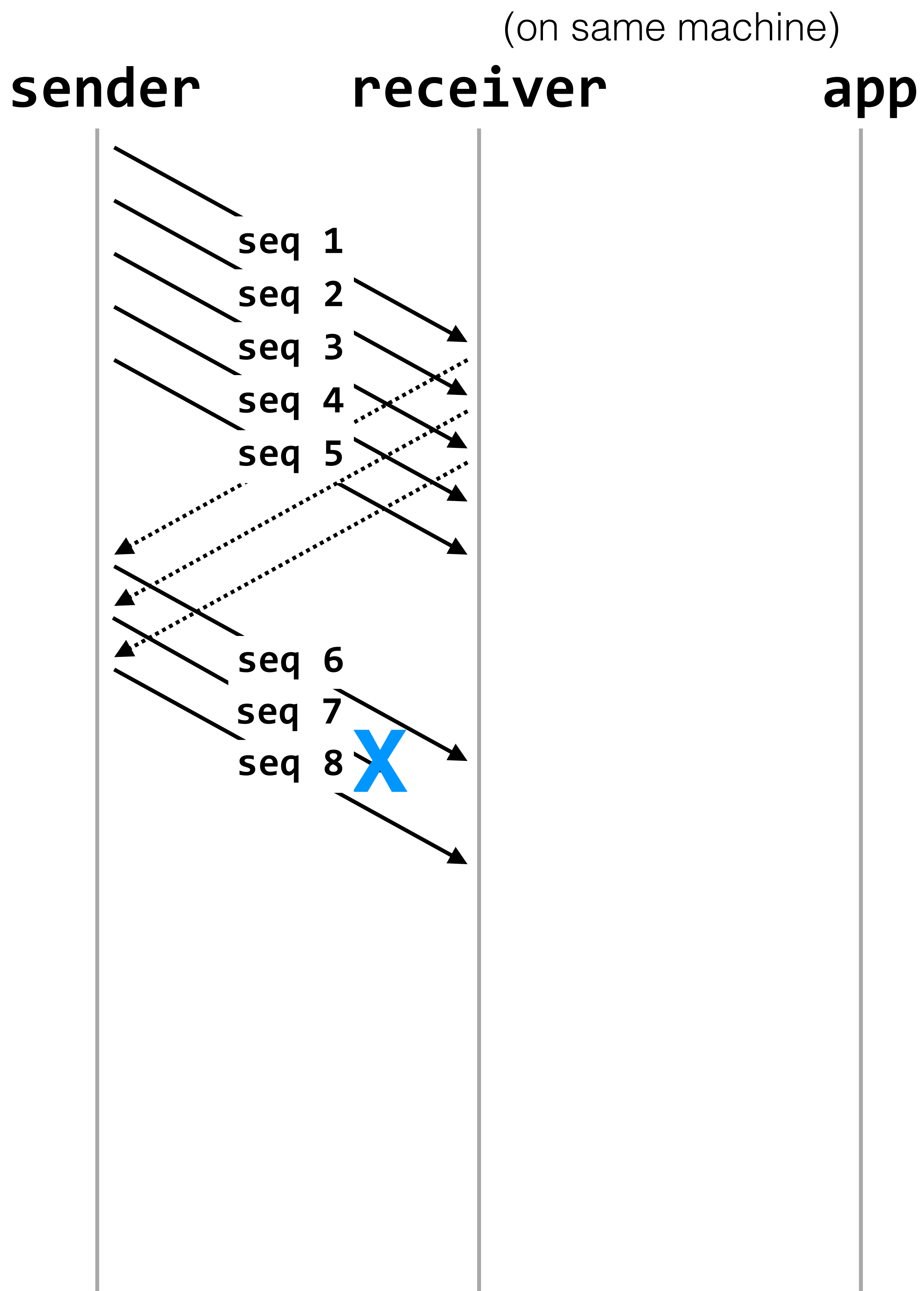
Sliding-window Protocol: Sender

Window: [3,4,5,6,7]



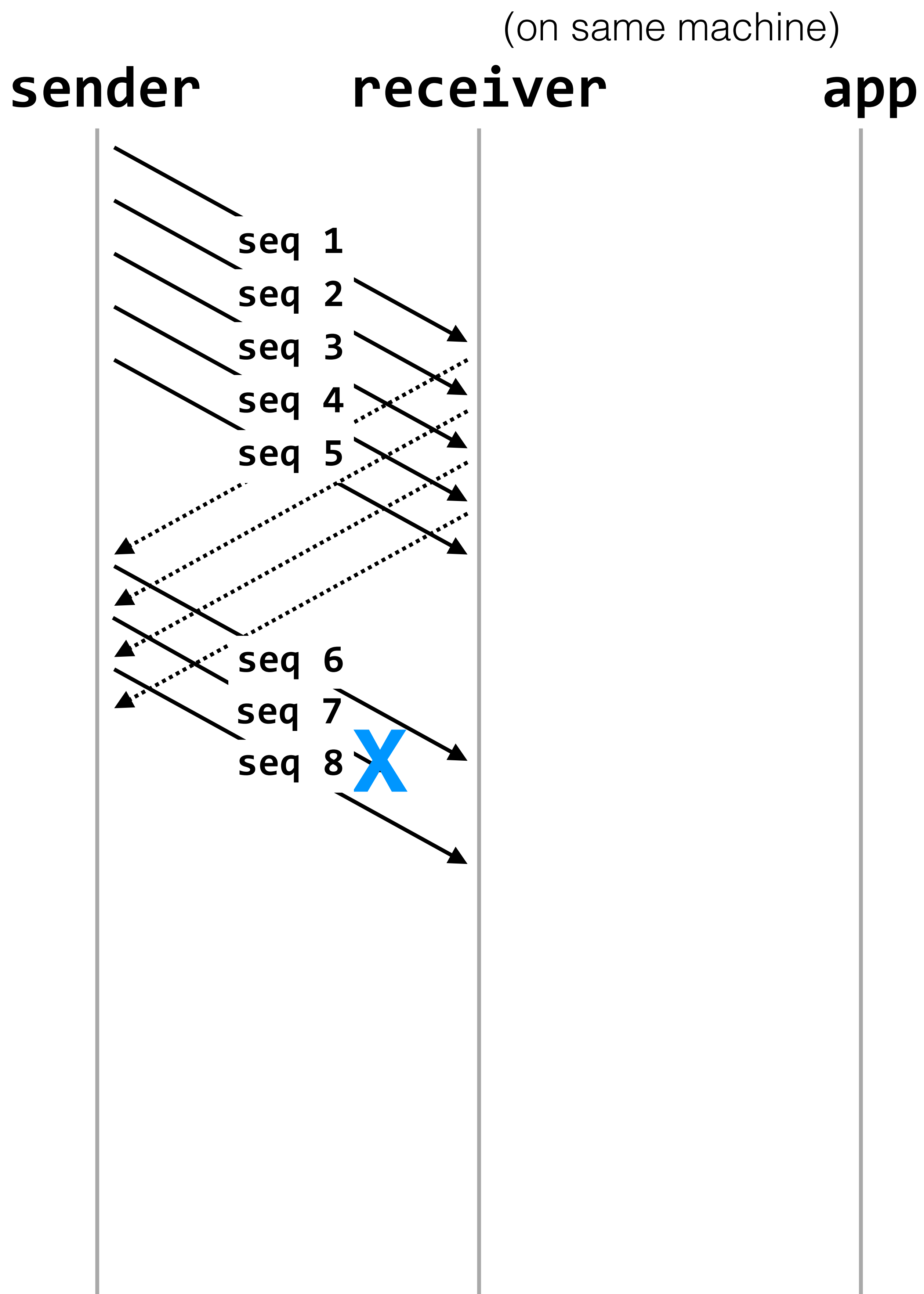
Sliding-window Protocol: Sender

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



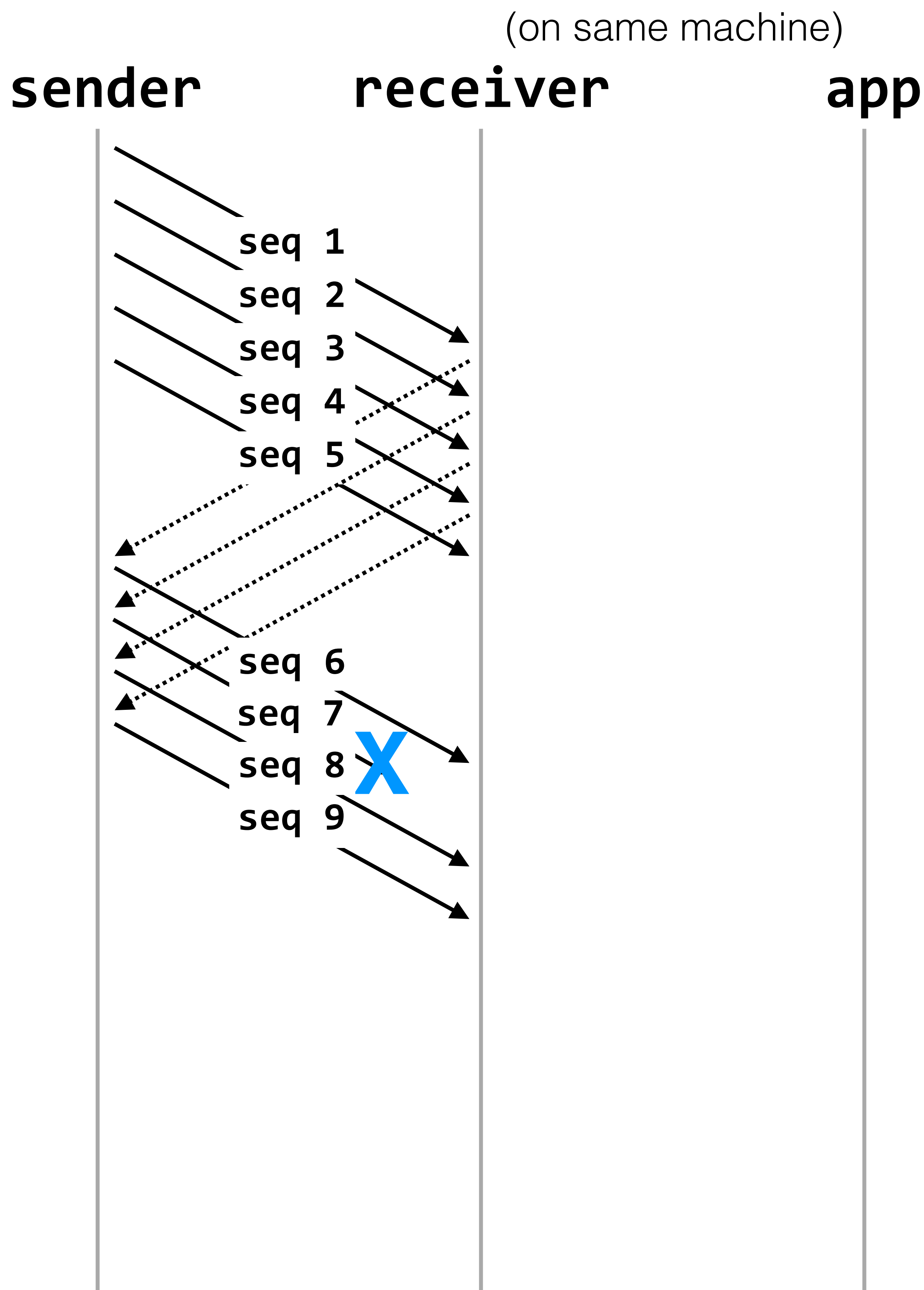
Sliding-window Protocol: Sender

`Window: [4,5,6,7,8]`



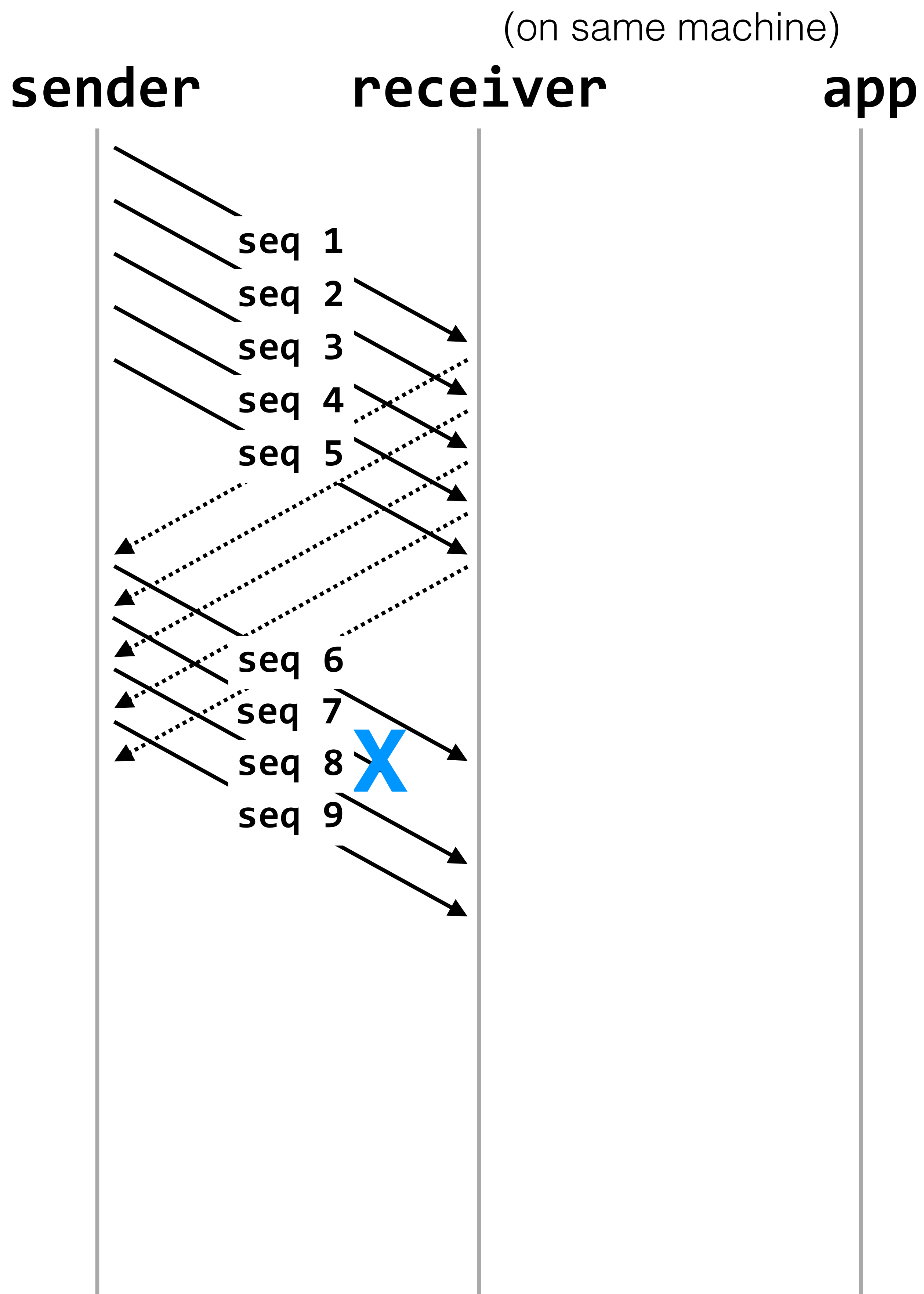
Sliding-window Protocol: Sender

`Window: [5,6,7,8,9]`



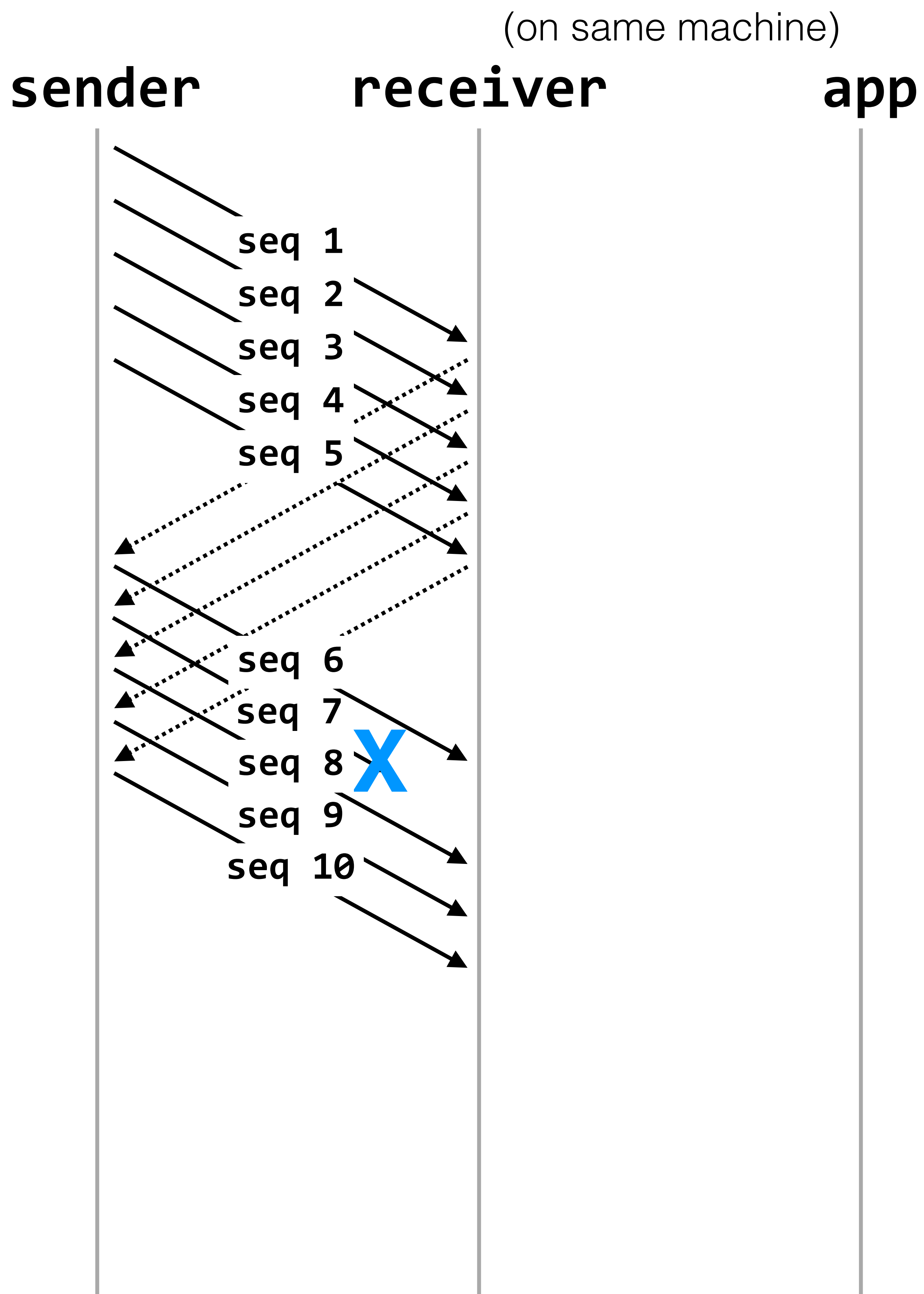
Sliding-window Protocol: Sender

Window: [5,6,7,8,9]



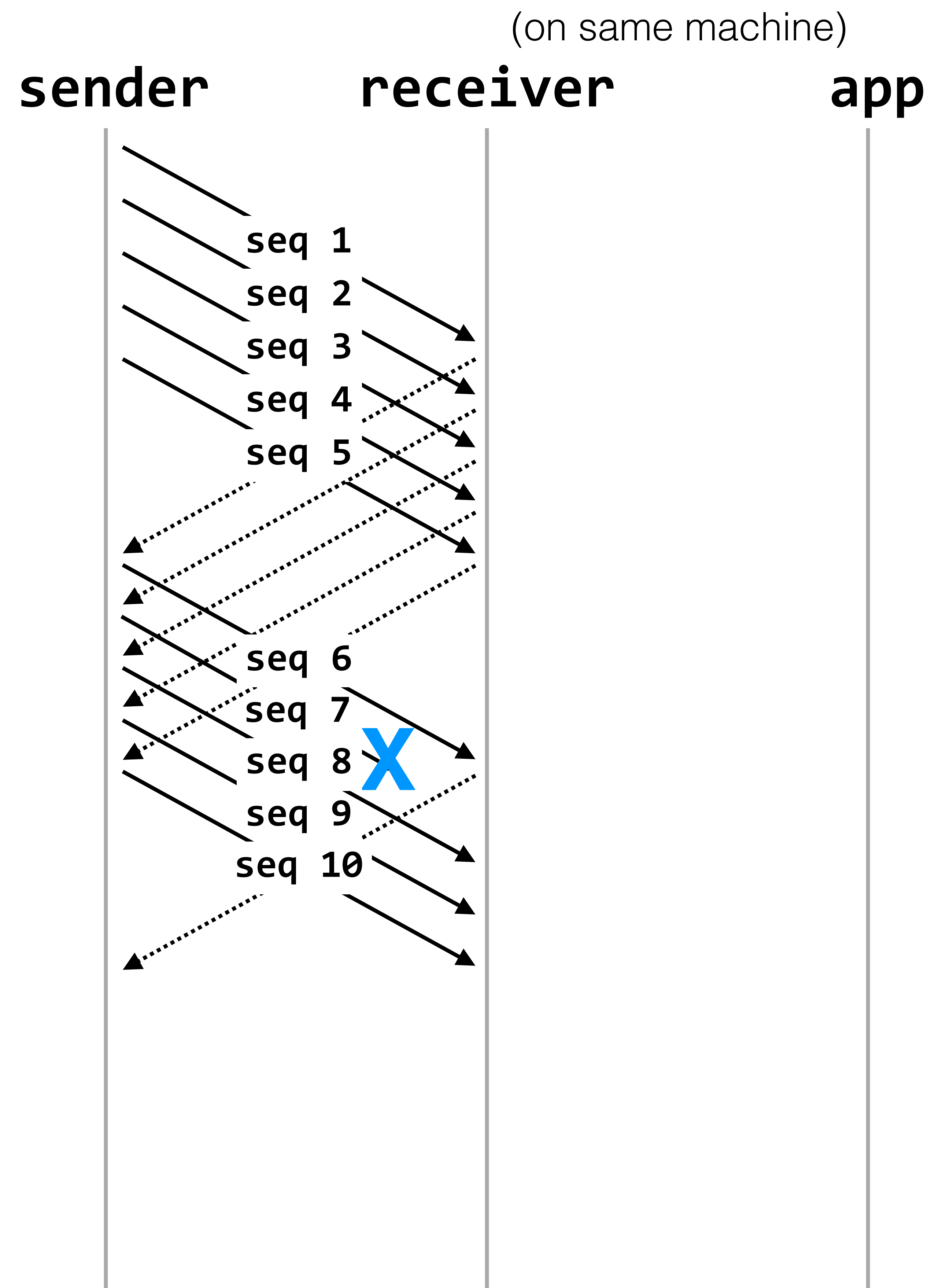
Sliding-window Protocol: Sender

Window: [6,7,8,9,10]



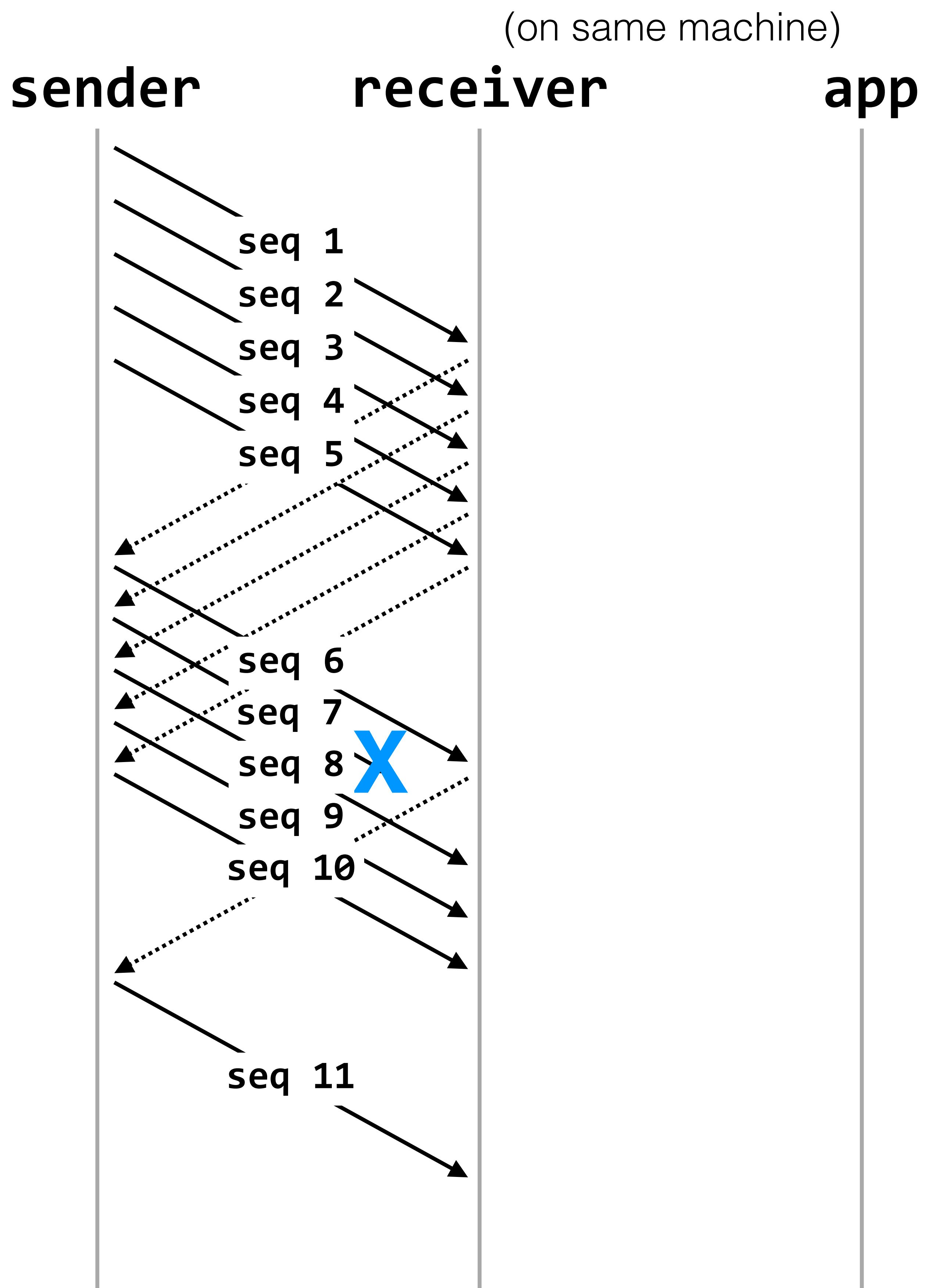
Sliding-window Protocol: Sender

Window: [6,7,8,9,10]



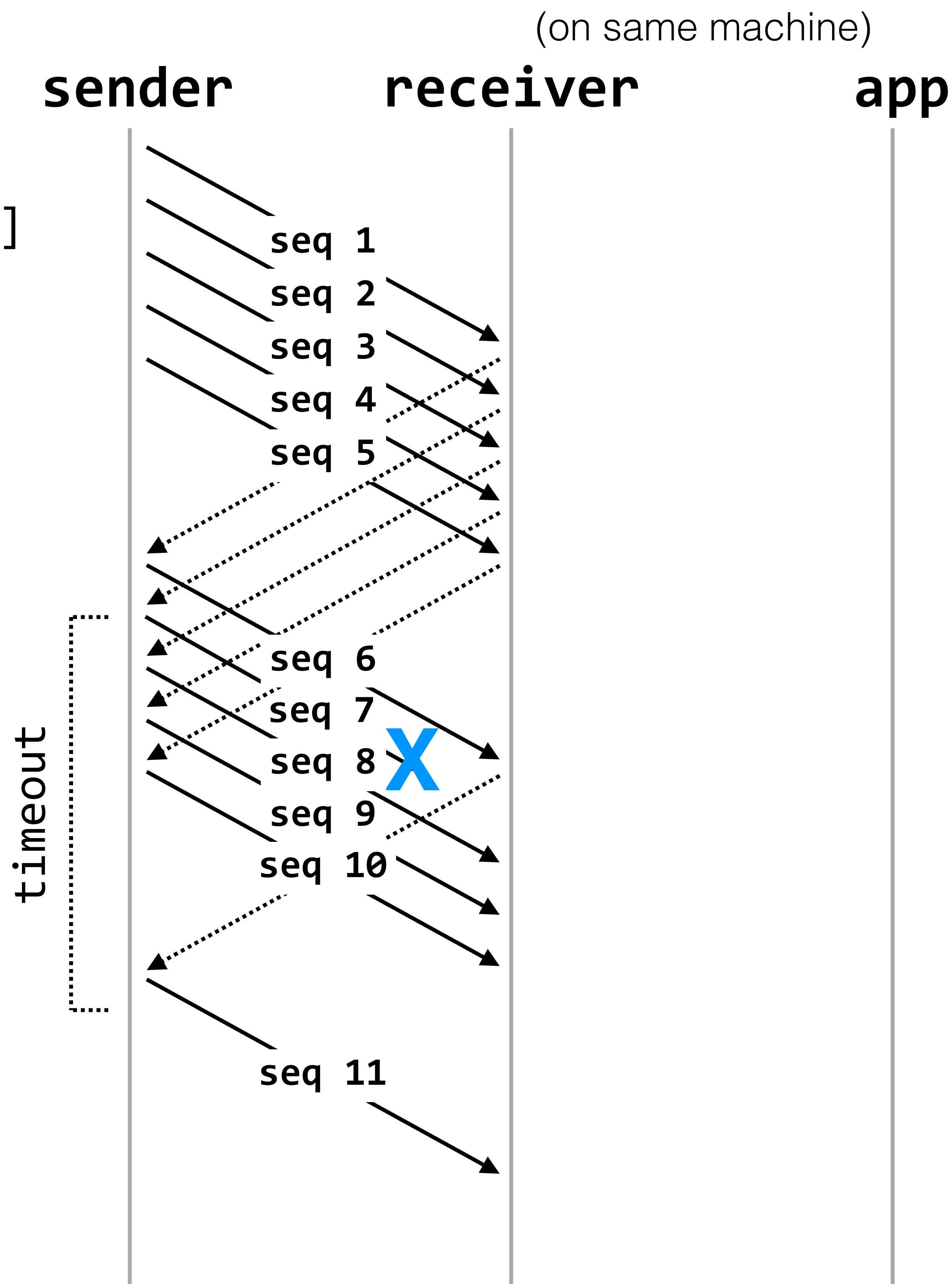
Sliding-window Protocol: Sender

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



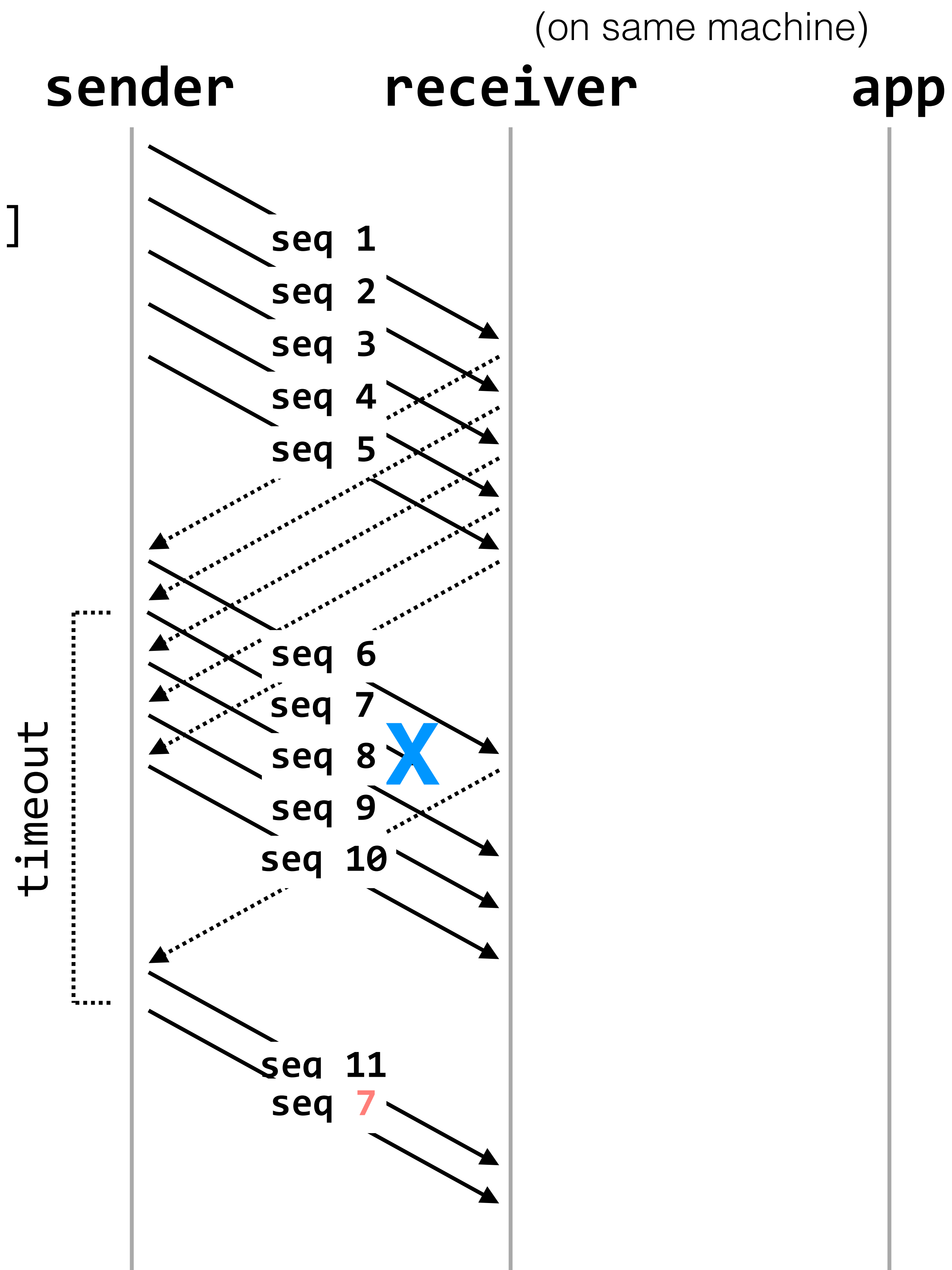
Sliding-window Protocol: Sender

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

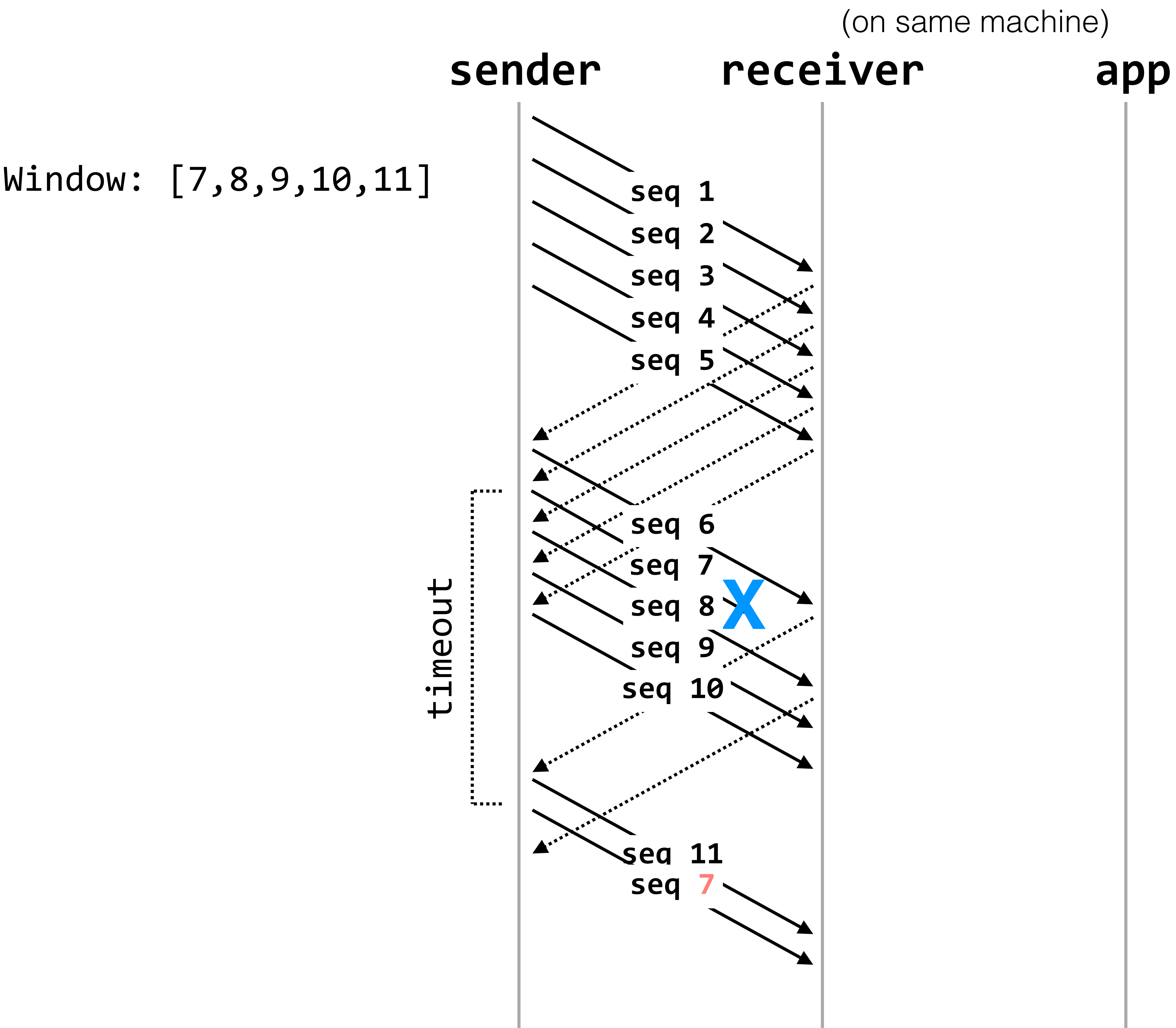


Sliding-window Protocol: Sender

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



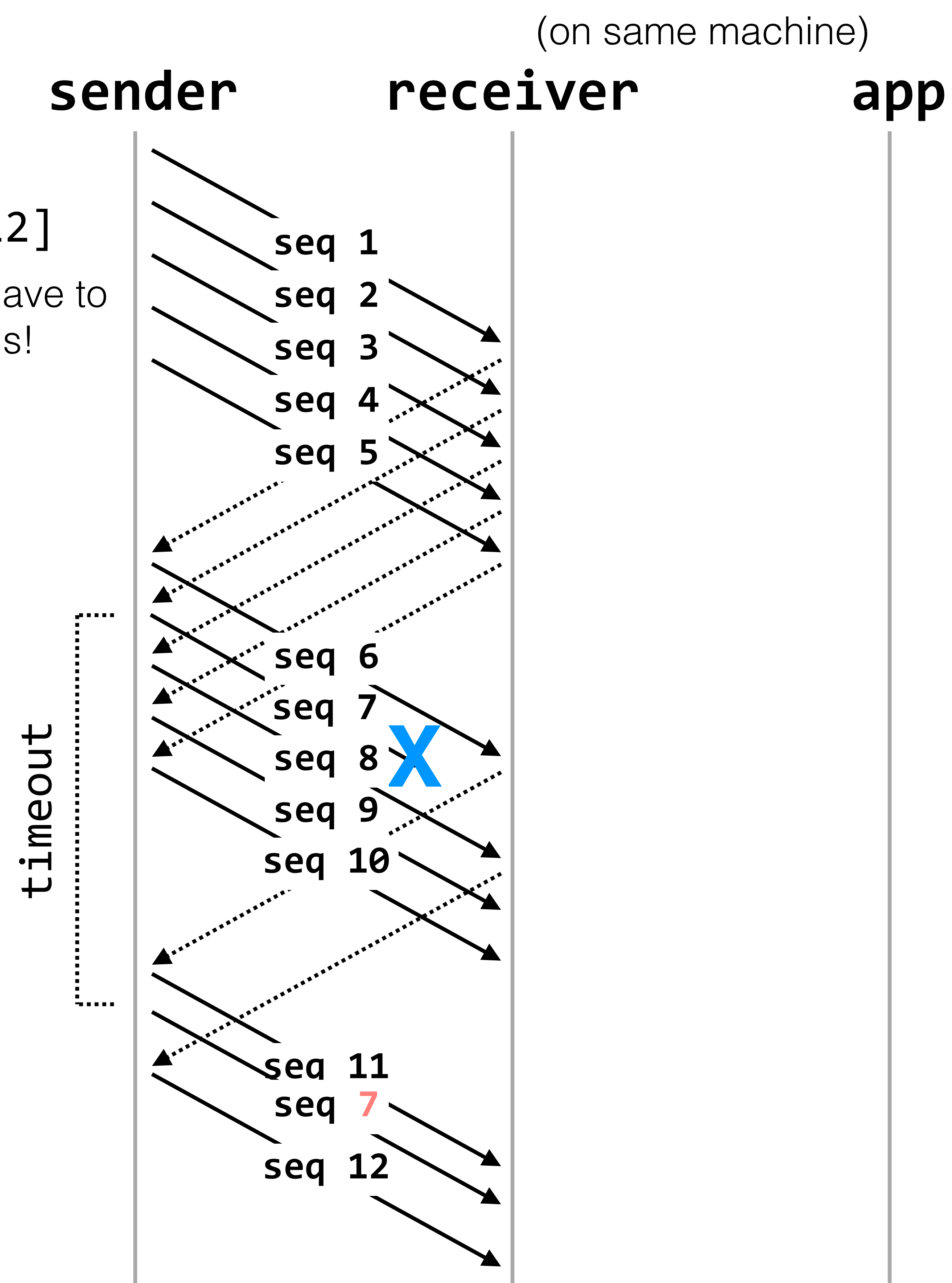
Sliding-window Protocol: Sender



Sliding-window Protocol: Sender

Window: [7,9,10,11,12]

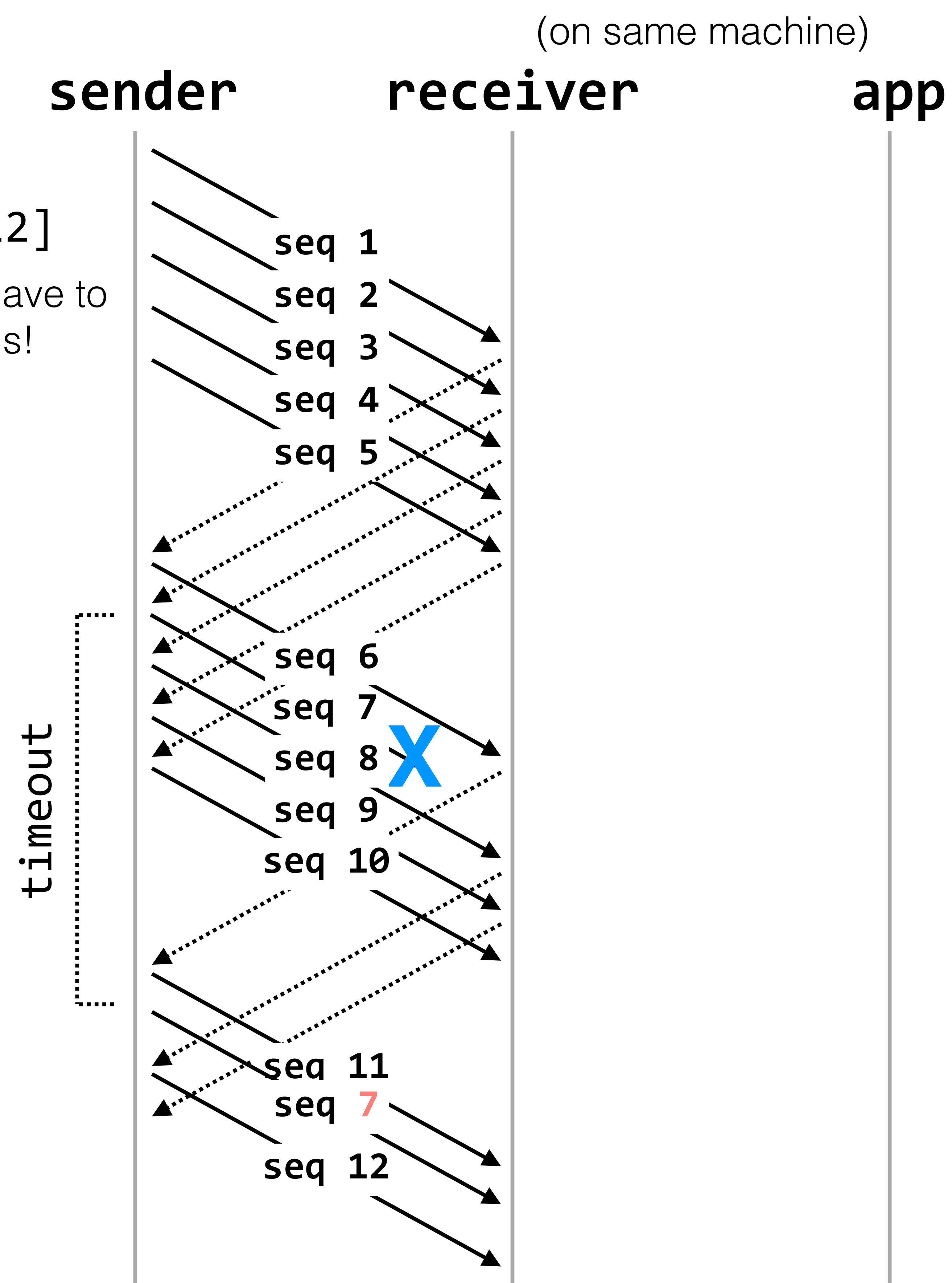
window doesn't have to be contiguous!



Sliding-window Protocol: Sender

Window: [7,9,10,11,12]

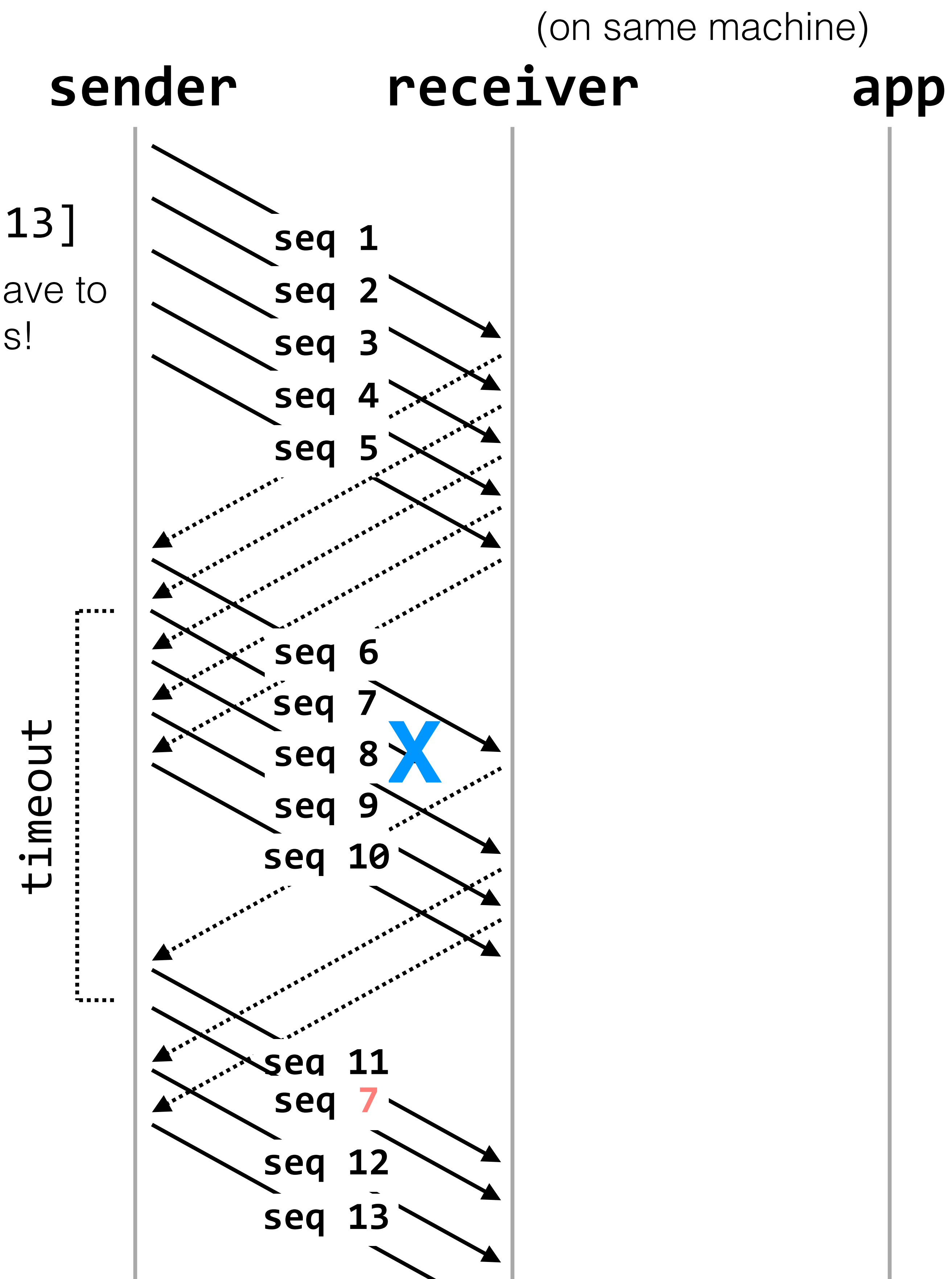
window doesn't have to be contiguous!



Sliding-window Protocol: Sender

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

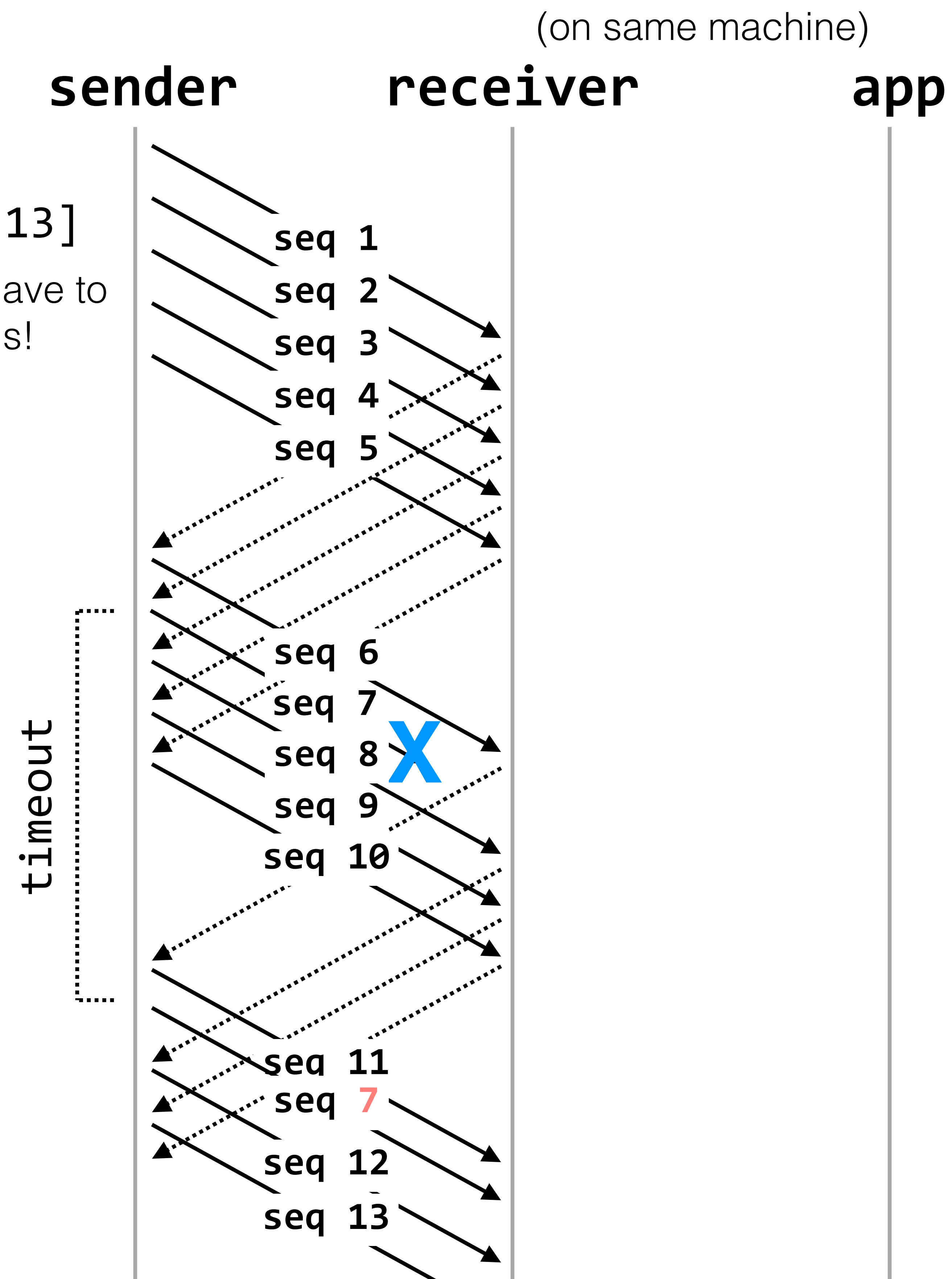
window doesn't have to be contiguous!



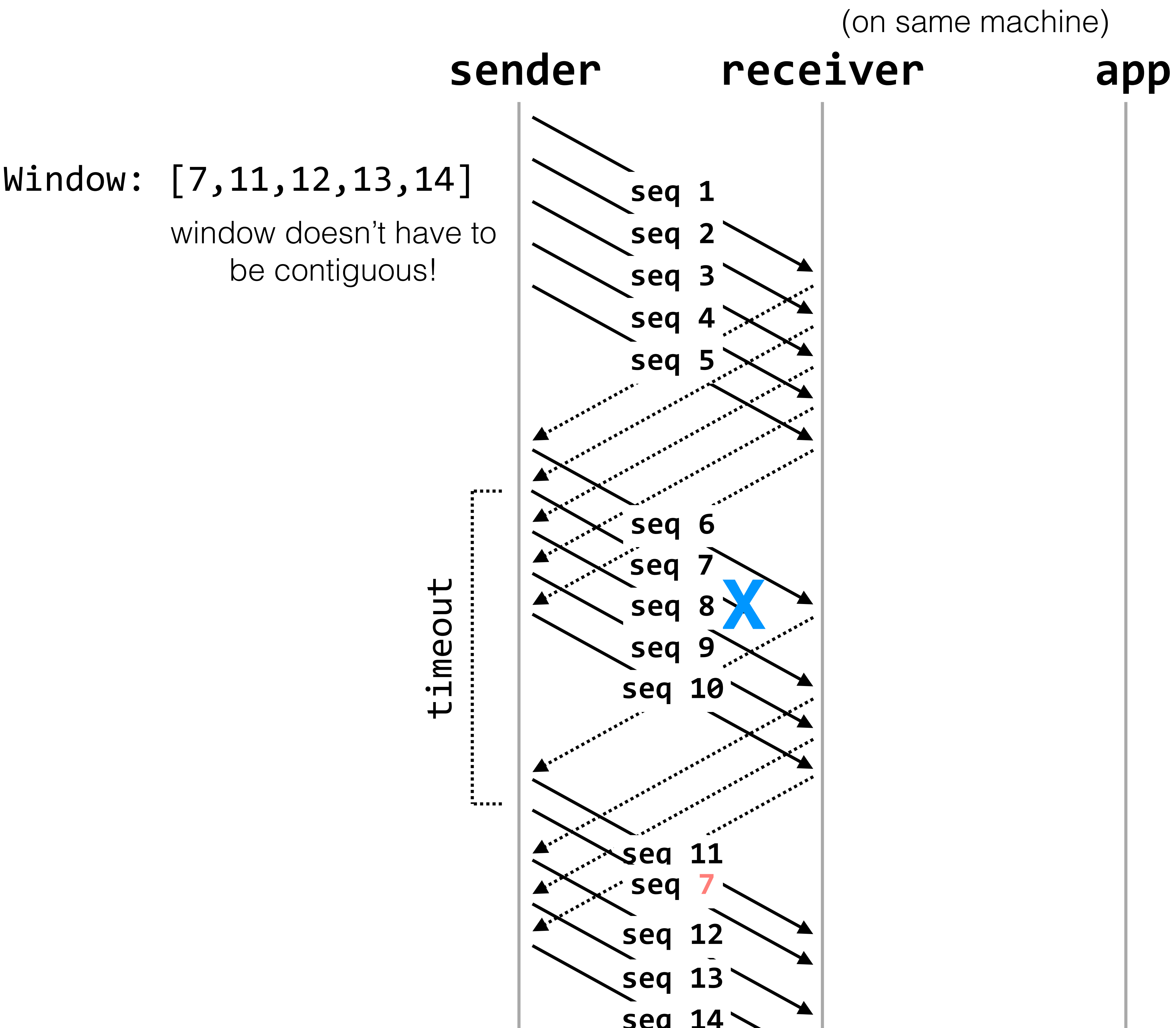
Sliding-window Protocol: Sender

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

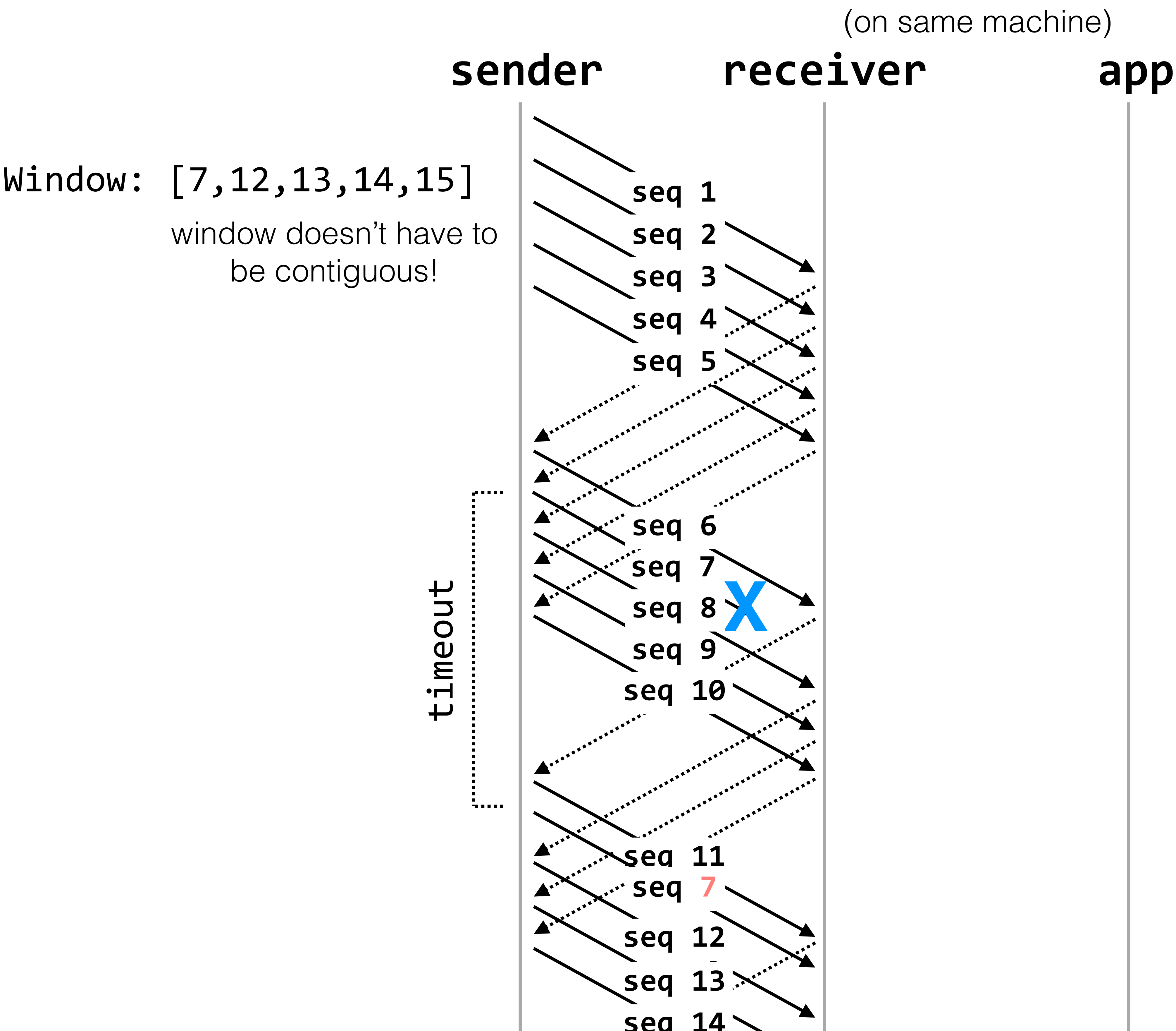
window doesn't have to be contiguous!



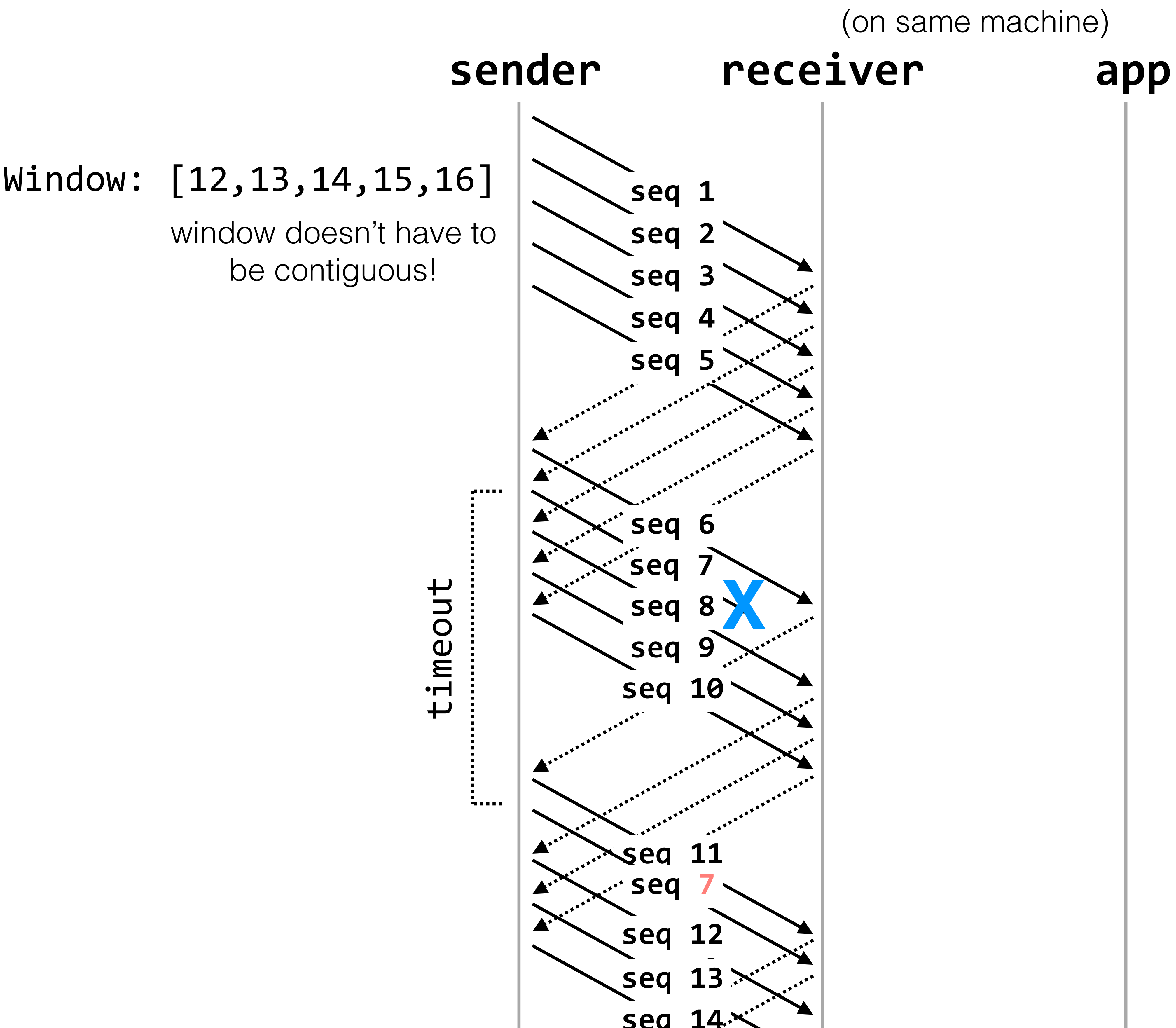
Sliding-window Protocol: Sender



Sliding-window Protocol: Sender



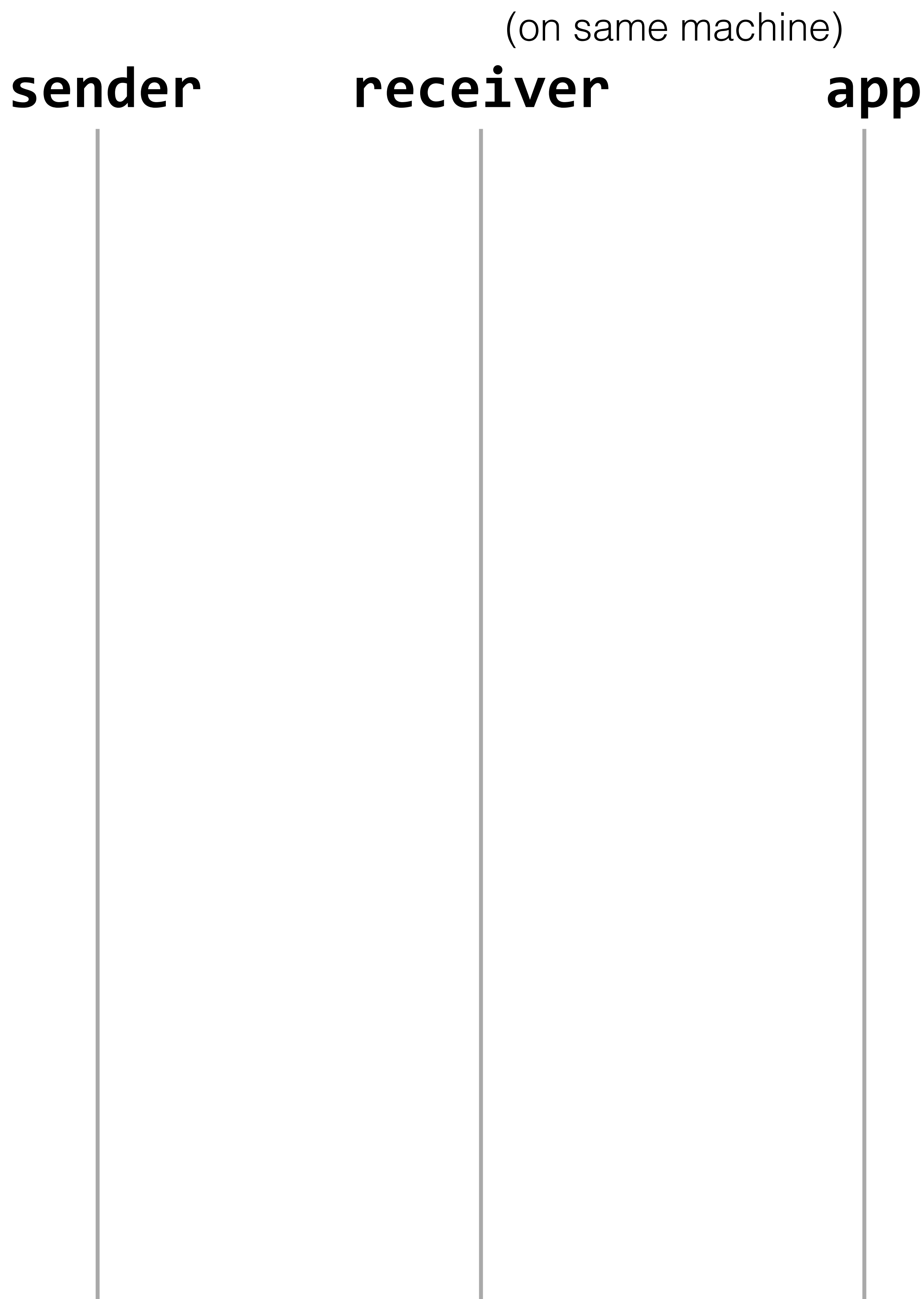
Sliding-window Protocol: Sender



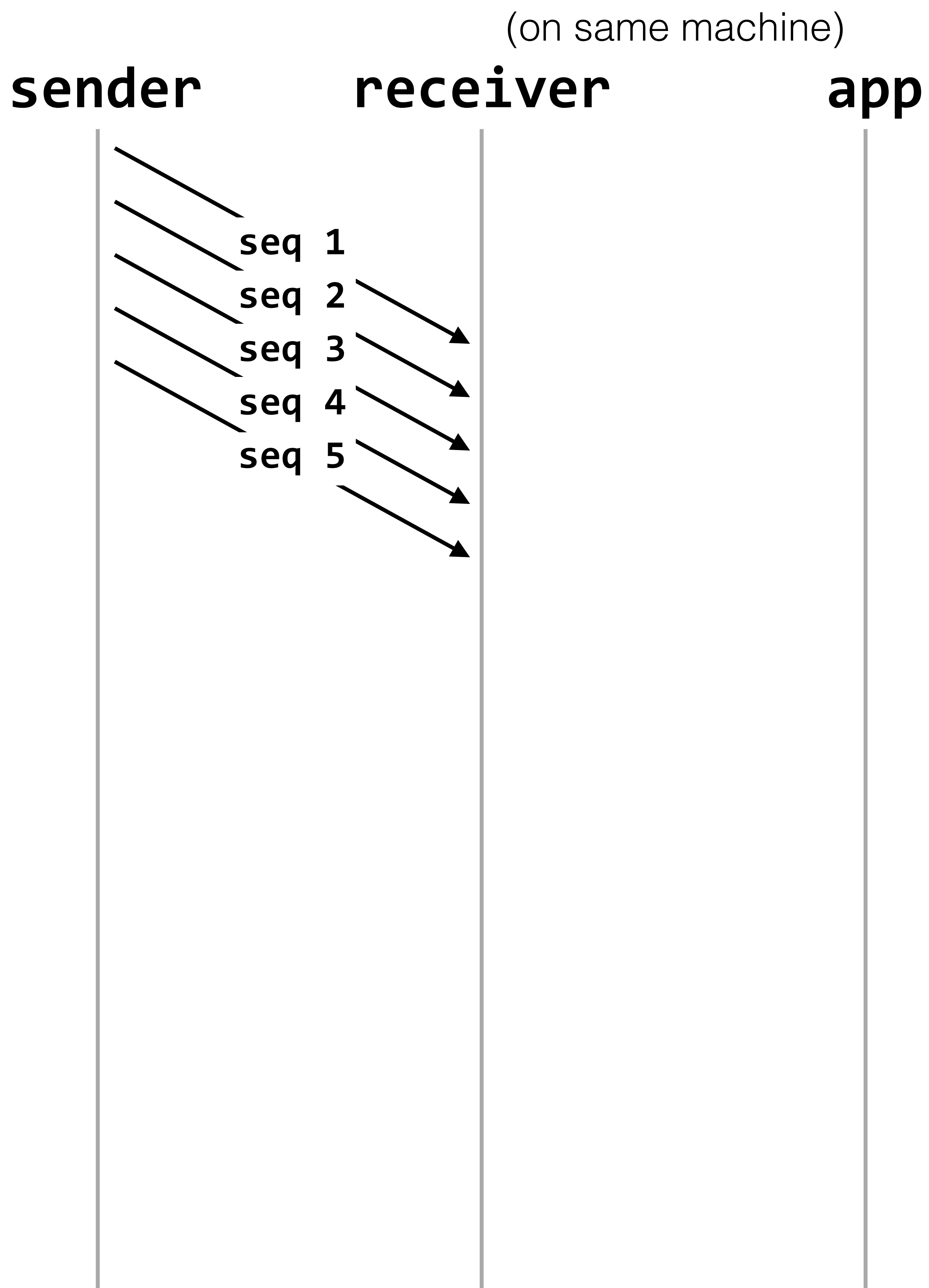
Sliding-window Protocol: Sender

- Transmit a packet if `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.

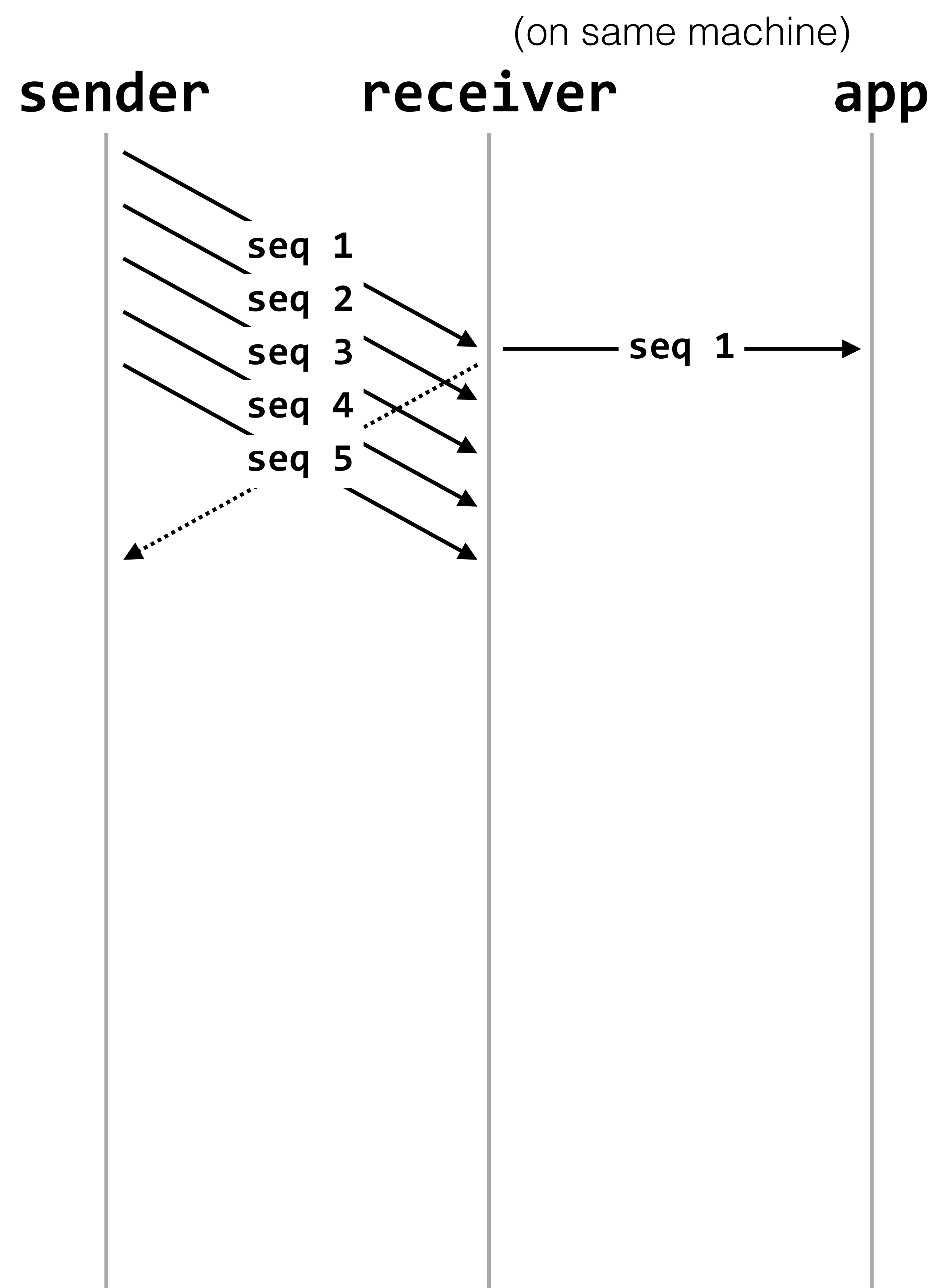
Sliding-window Protocol: Receiver



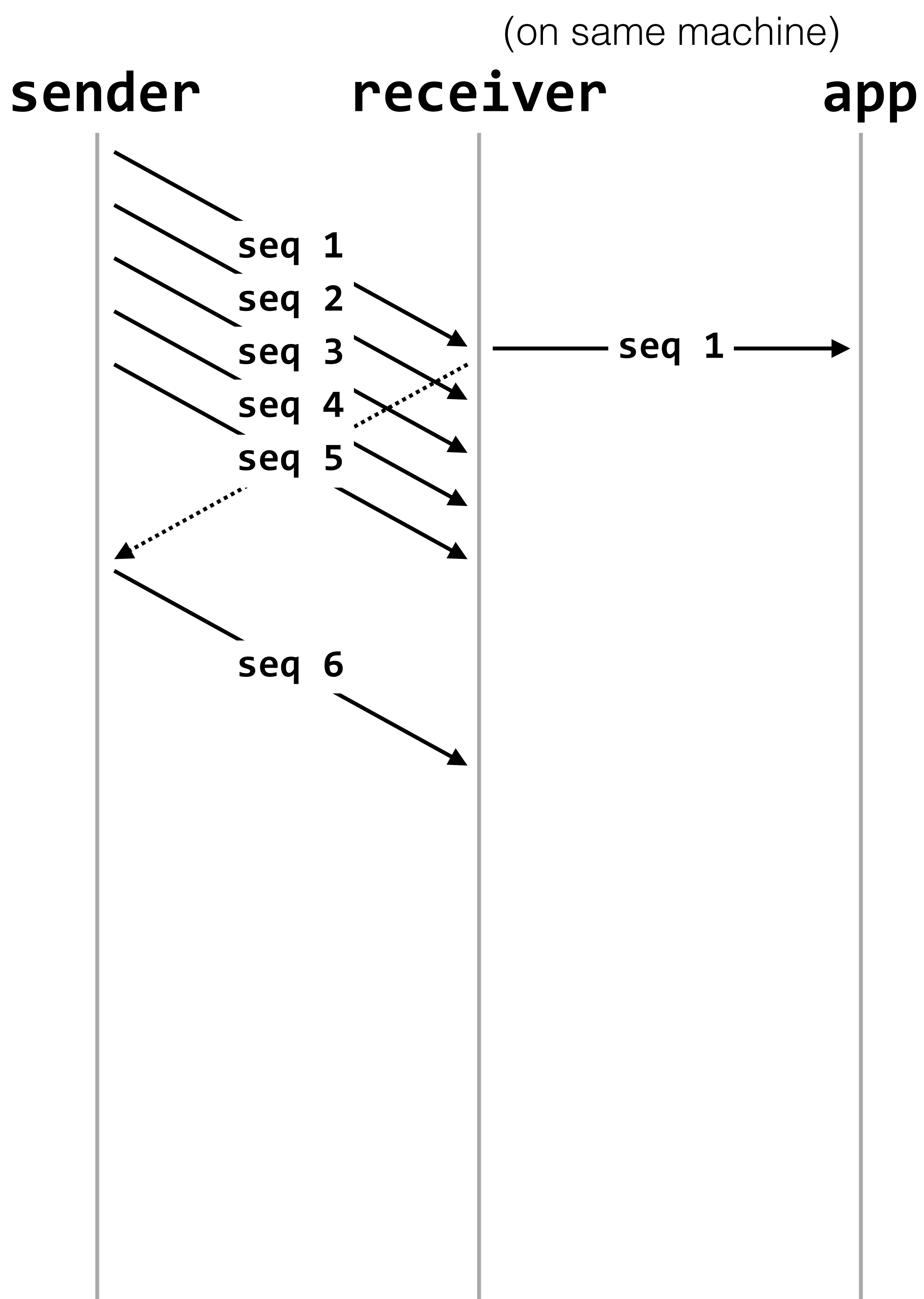
Sliding-window Protocol: Receiver



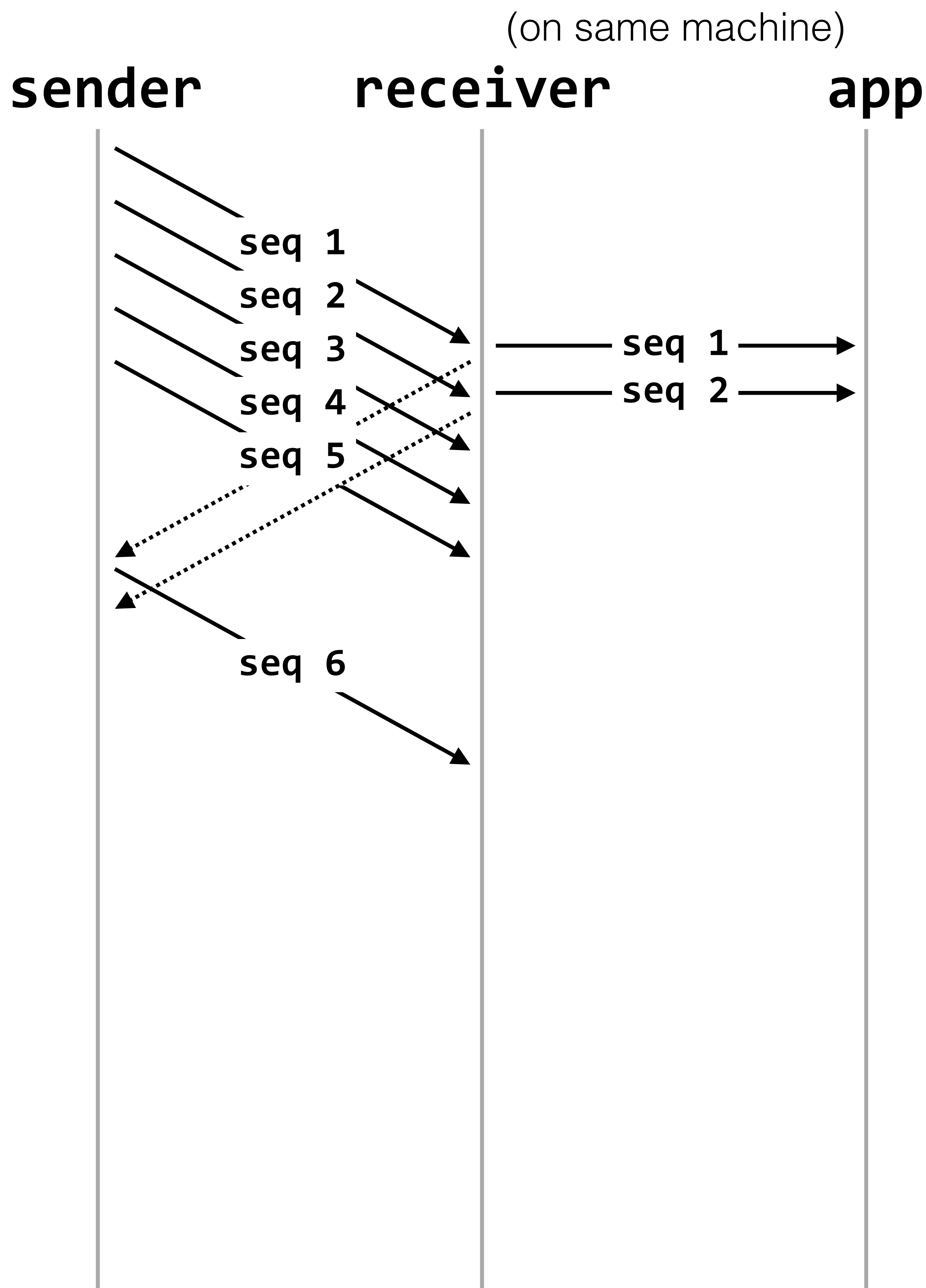
Sliding-window Protocol: Receiver



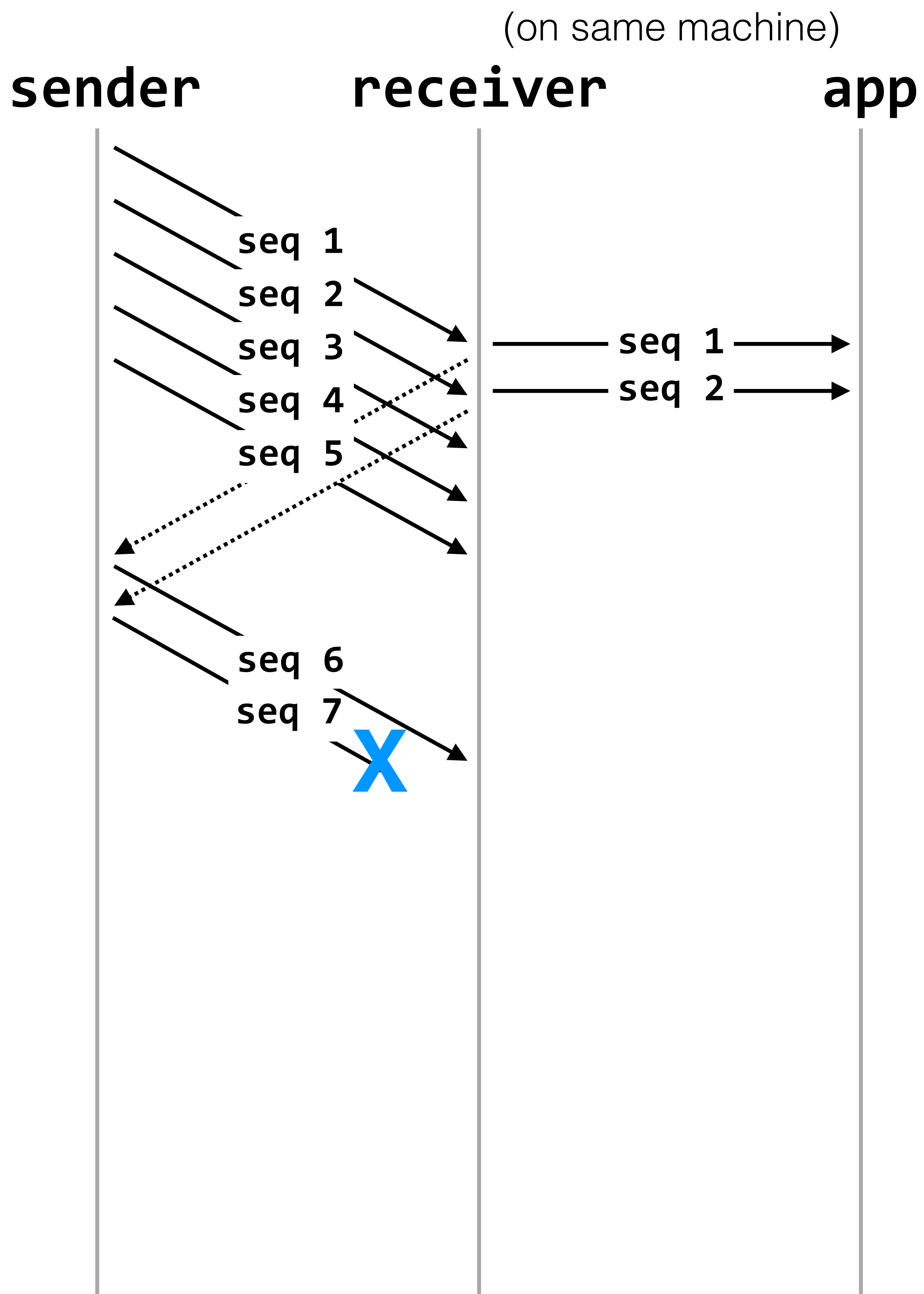
Sliding-window Protocol: Receiver



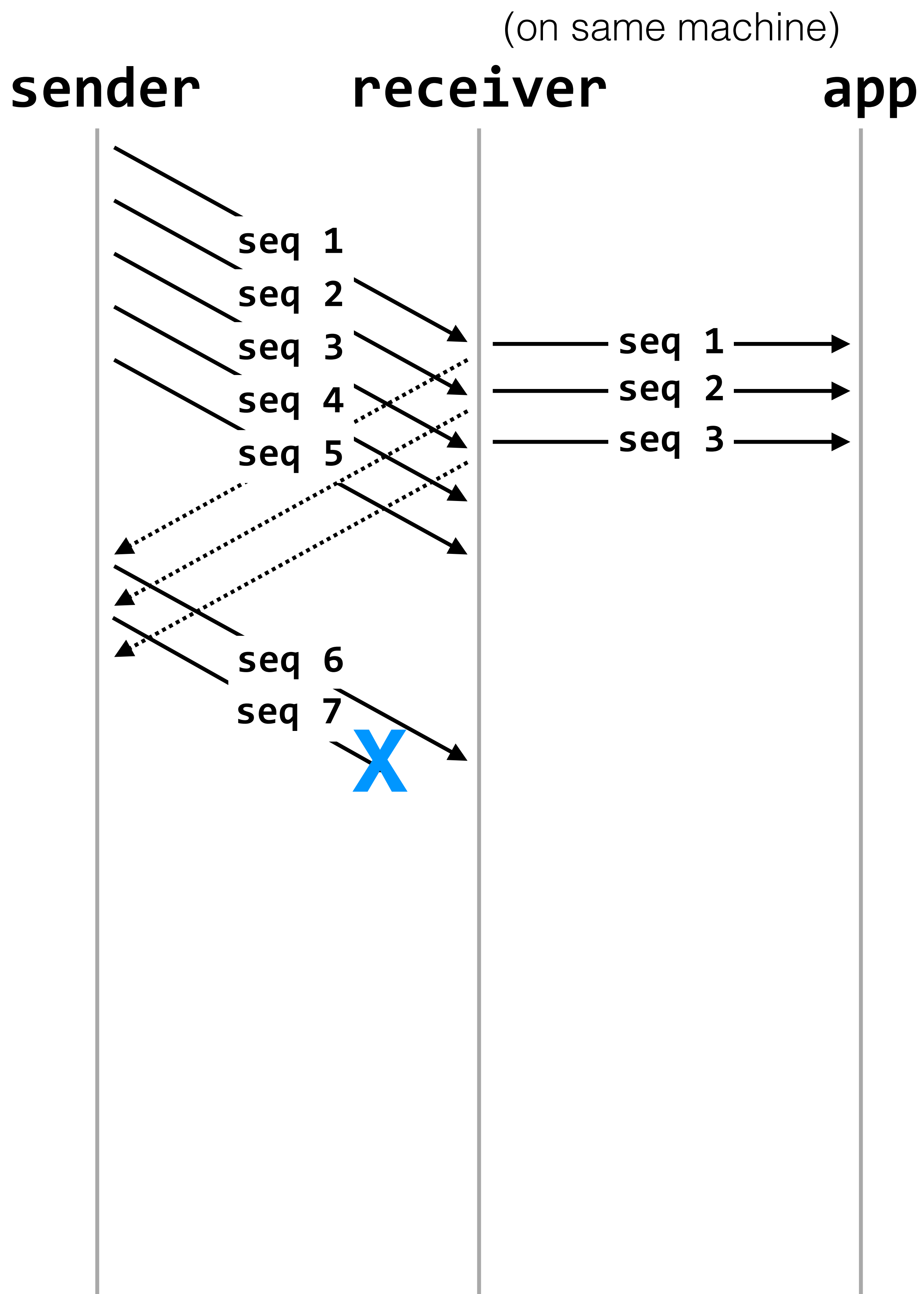
Sliding-window Protocol: Receiver



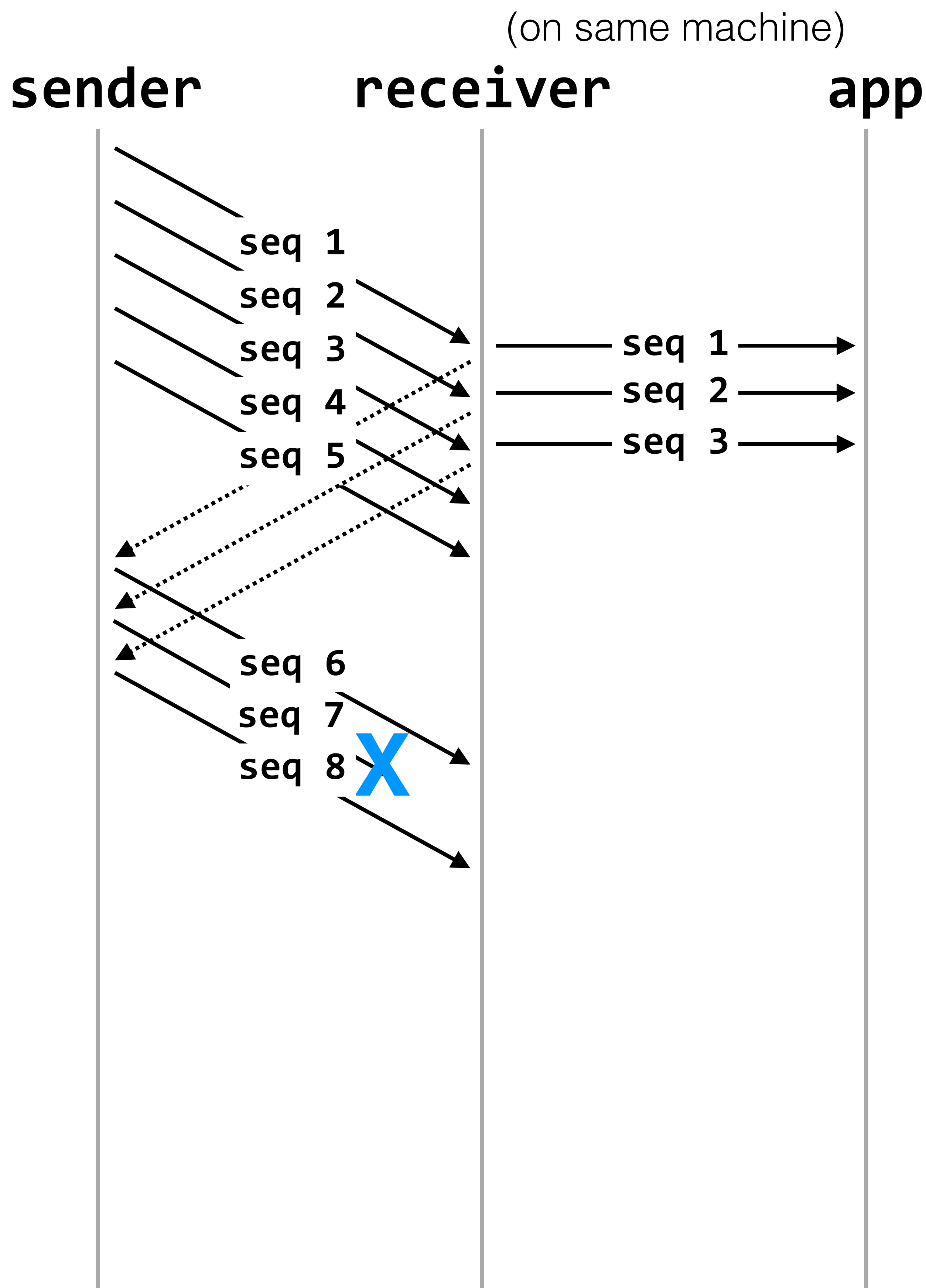
Sliding-window Protocol: Receiver



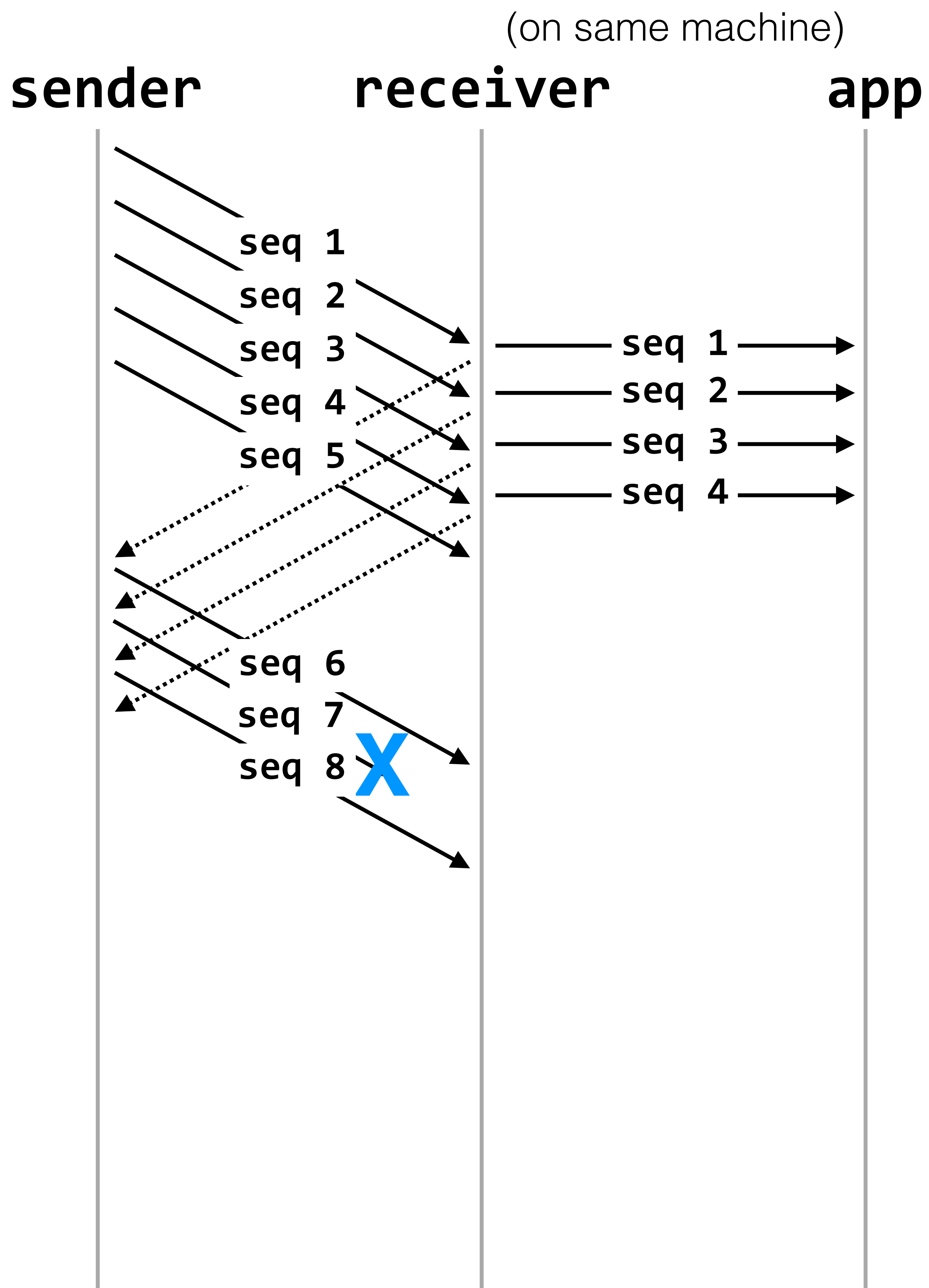
Sliding-window Protocol: Receiver



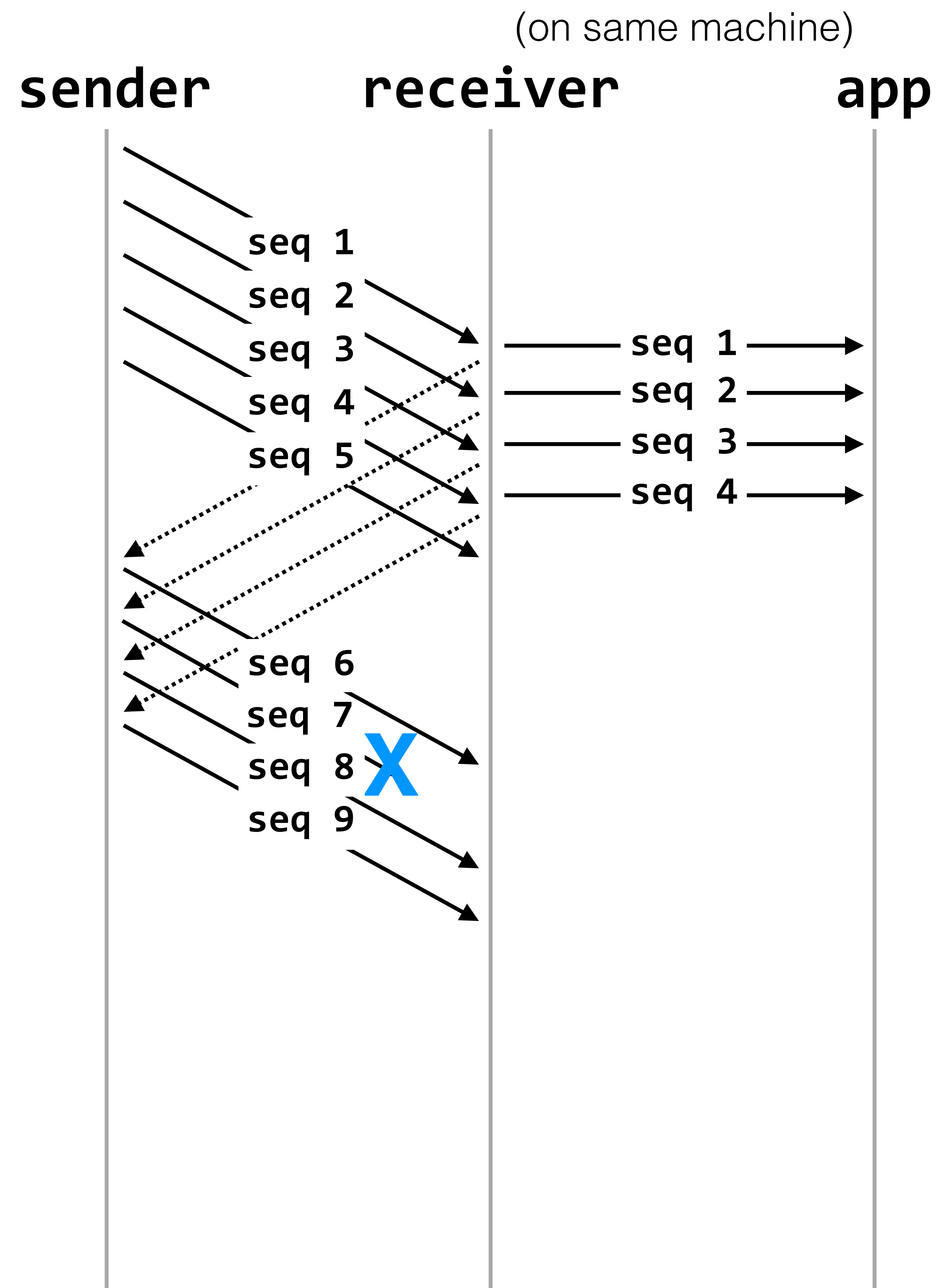
Sliding-window Protocol: Receiver



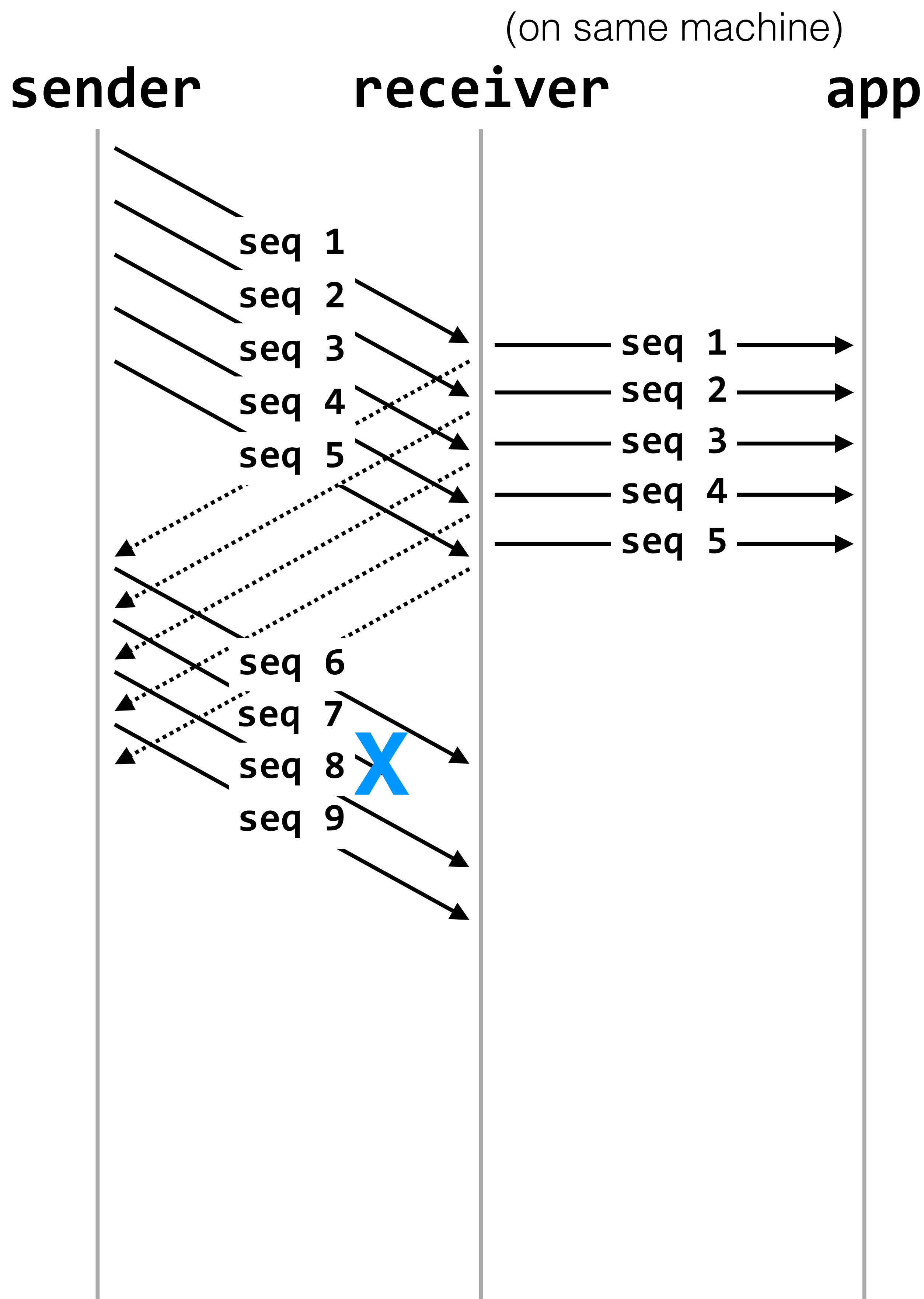
Sliding-window Protocol: Receiver



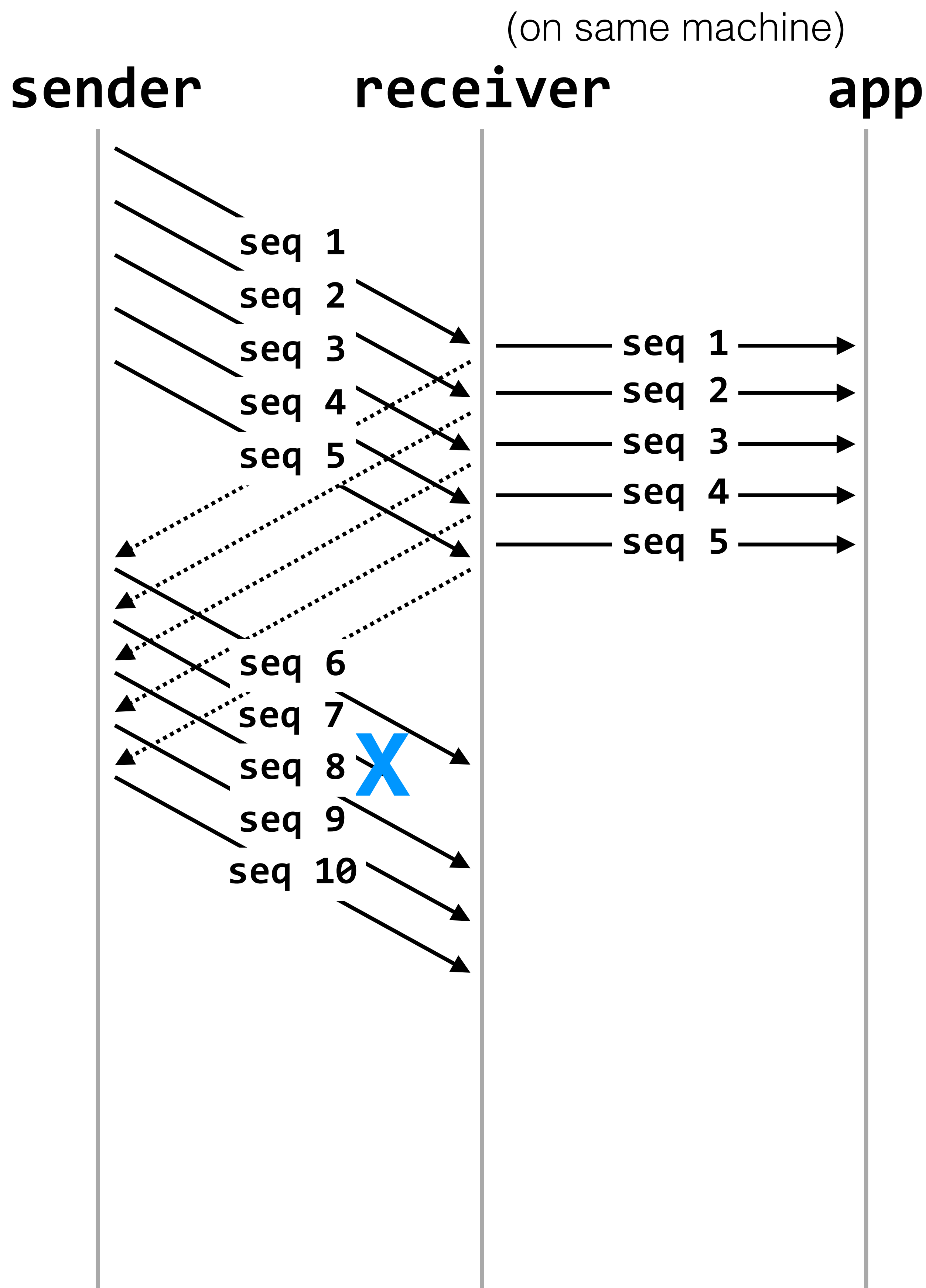
Sliding-window Protocol: Receiver



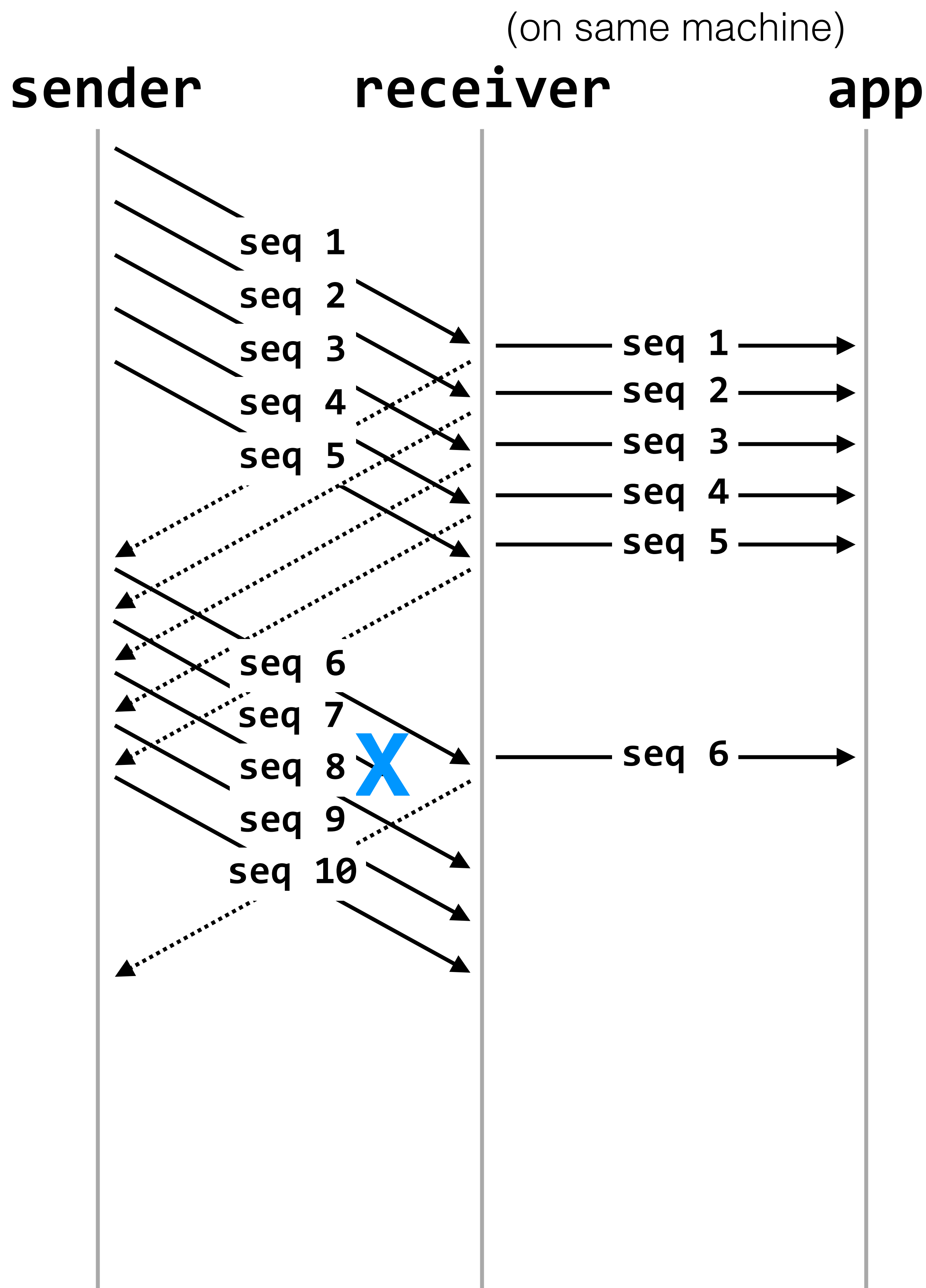
Sliding-window Protocol: Receiver



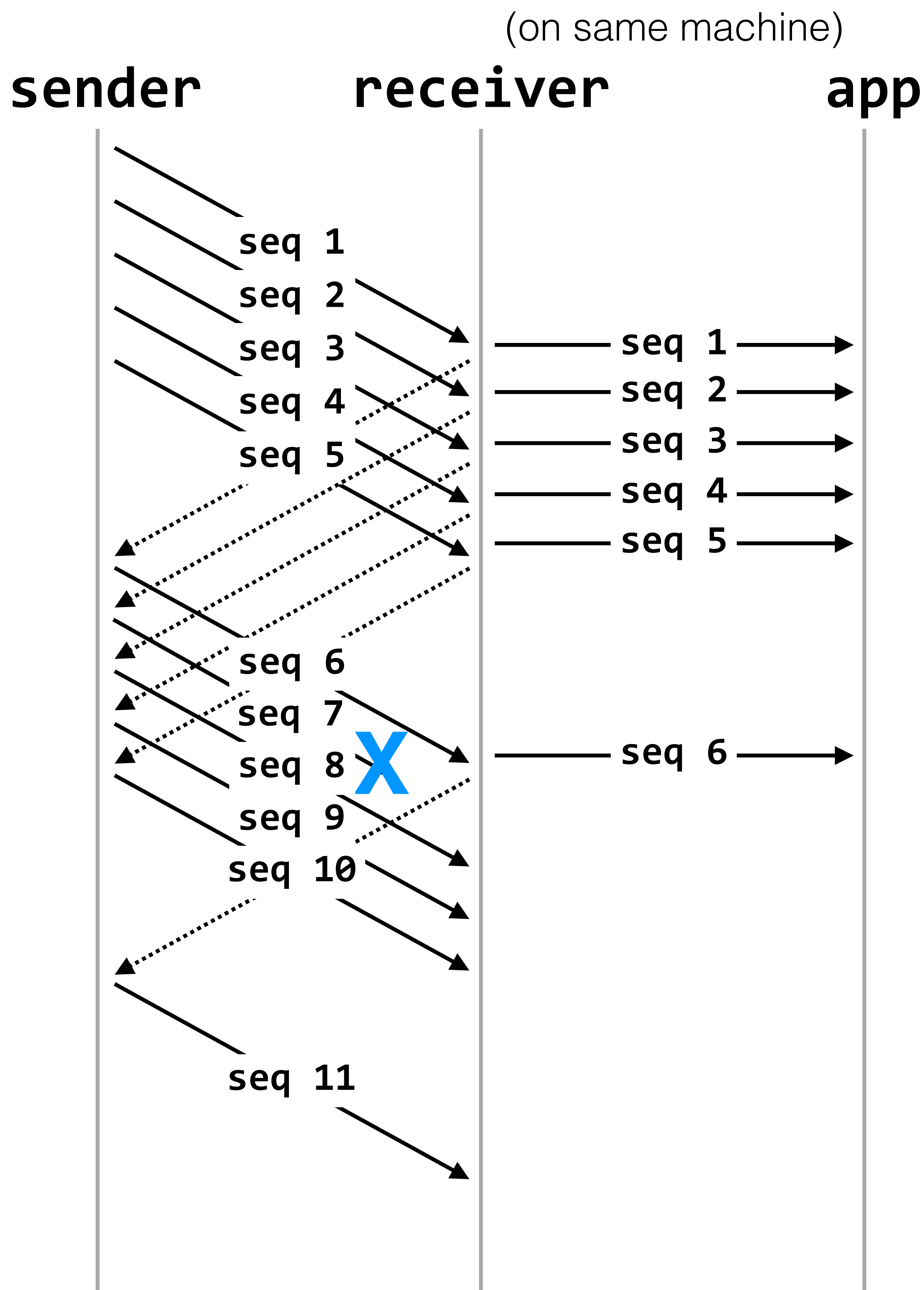
Sliding-window Protocol: Receiver



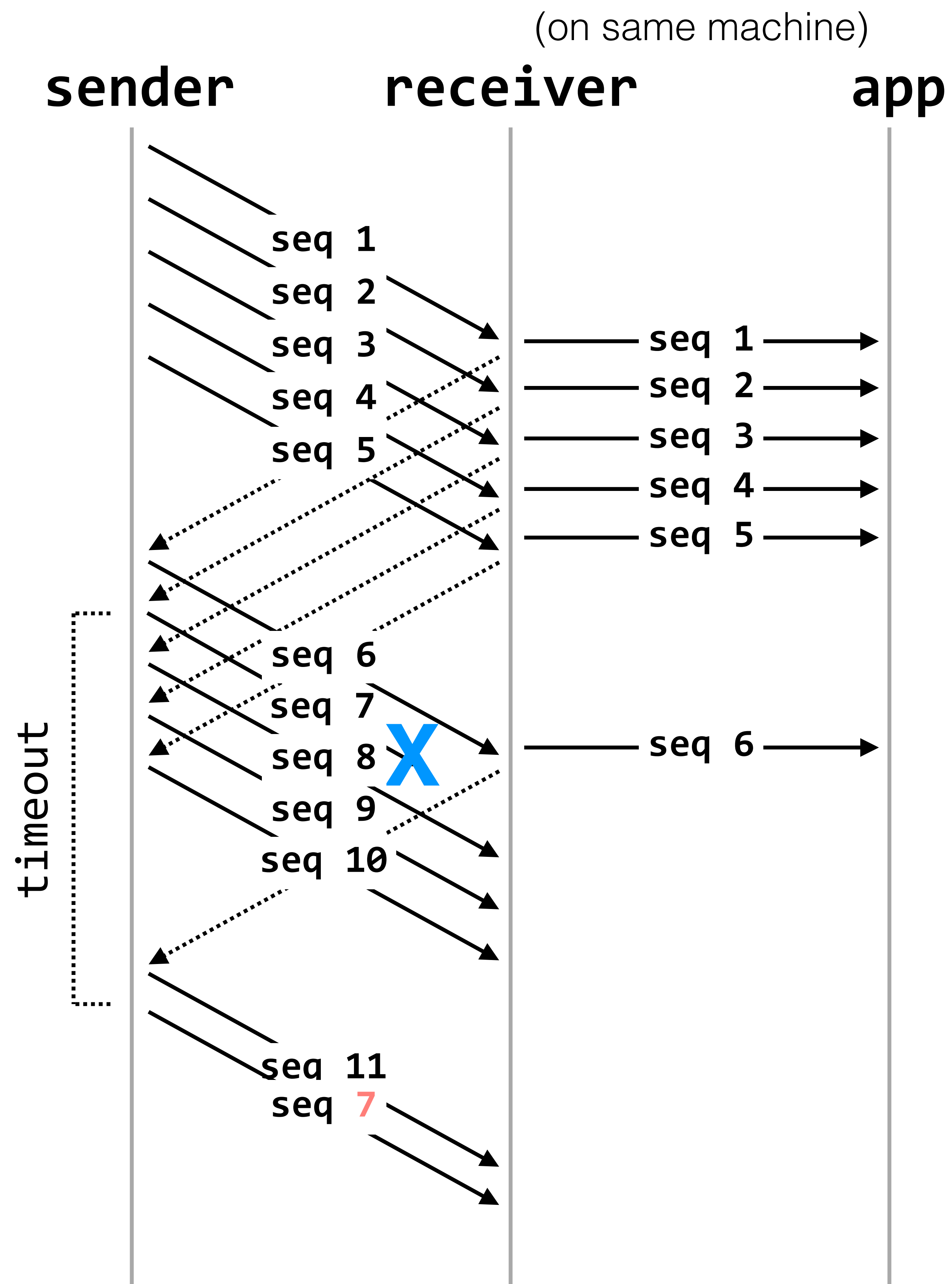
Sliding-window Protocol: Receiver



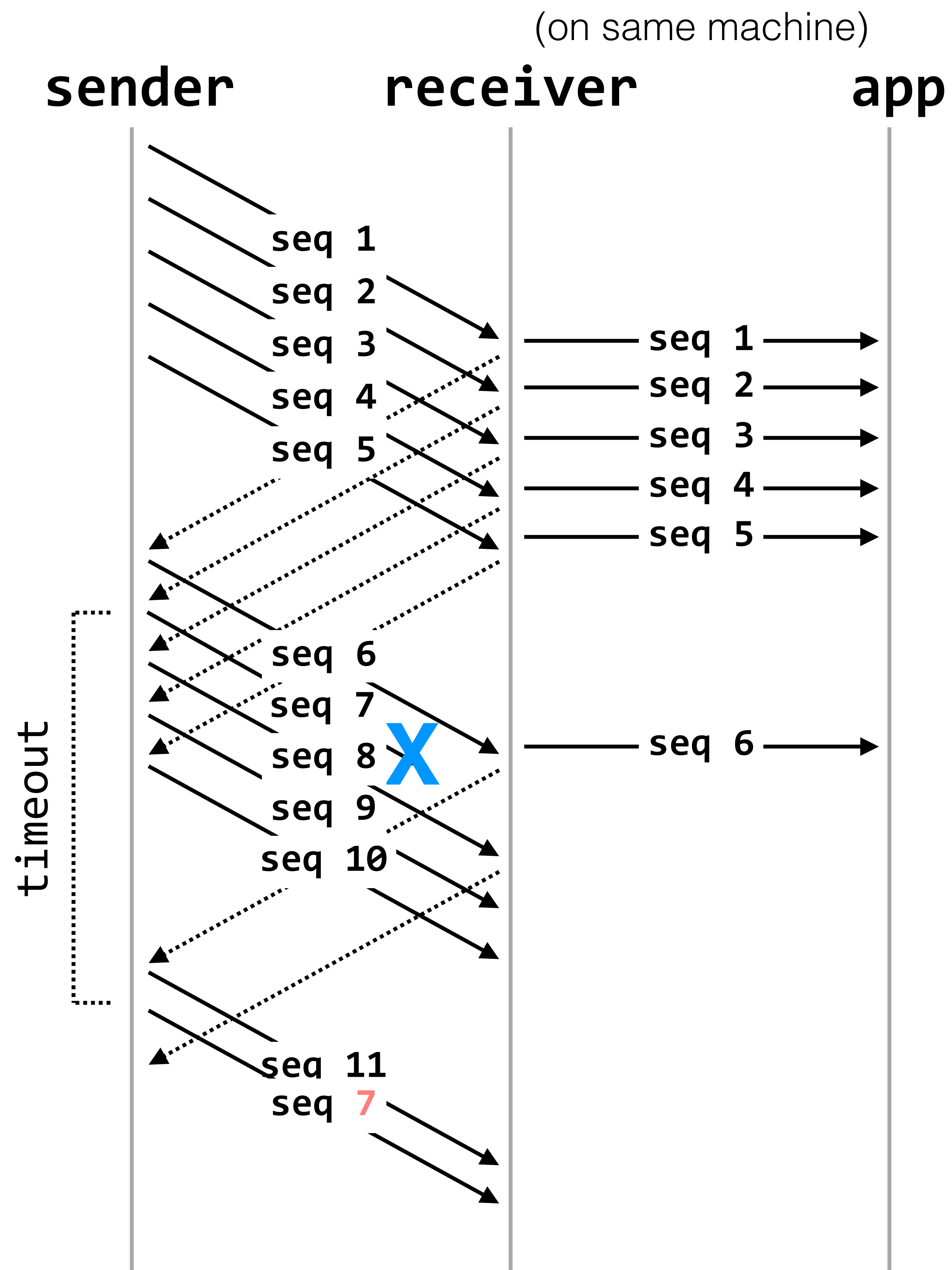
Sliding-window Protocol: Receiver



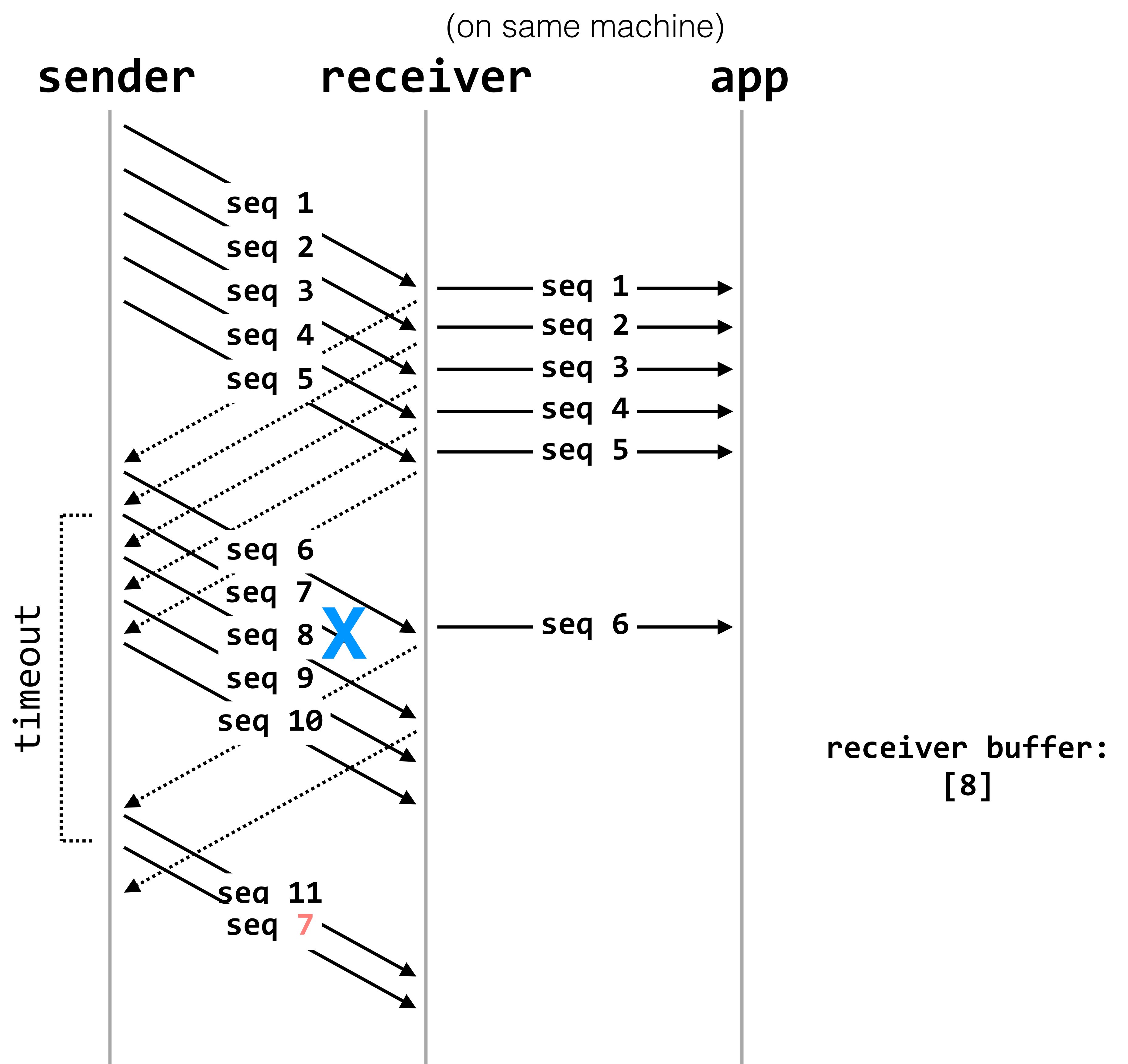
Sliding-window Protocol: Receiver



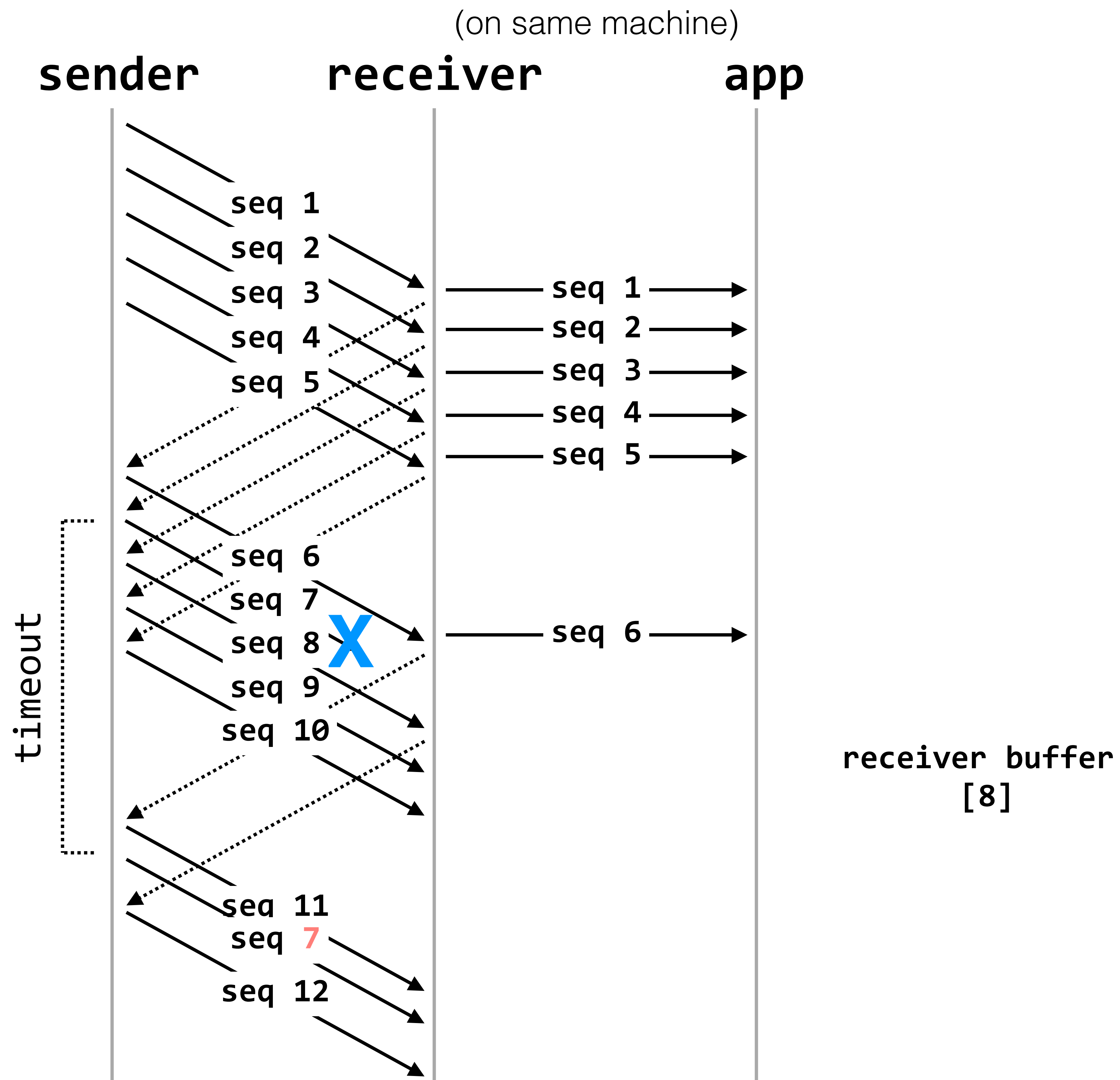
Sliding-window Protocol: Receiver



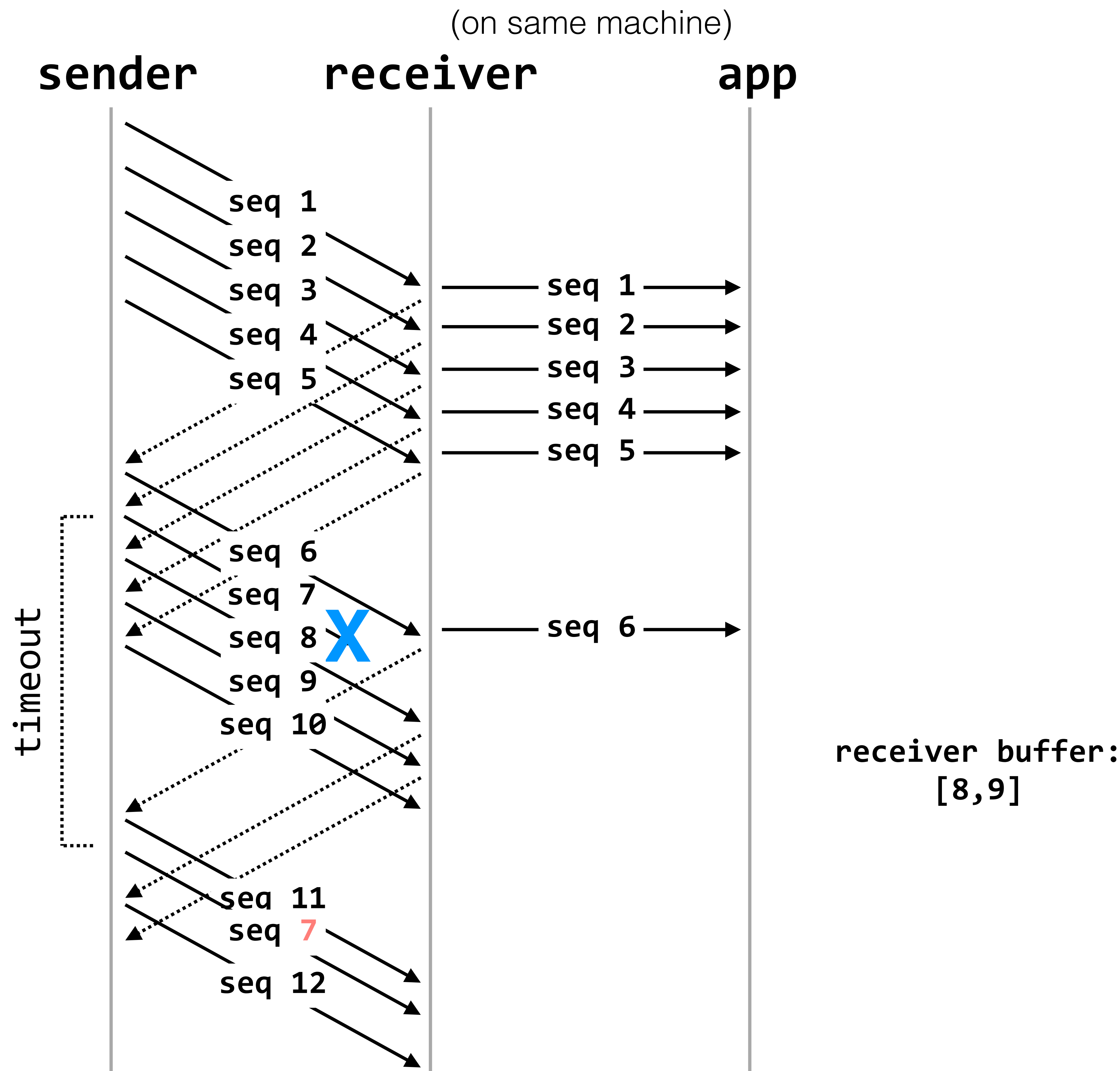
Sliding-window Protocol: Receiver



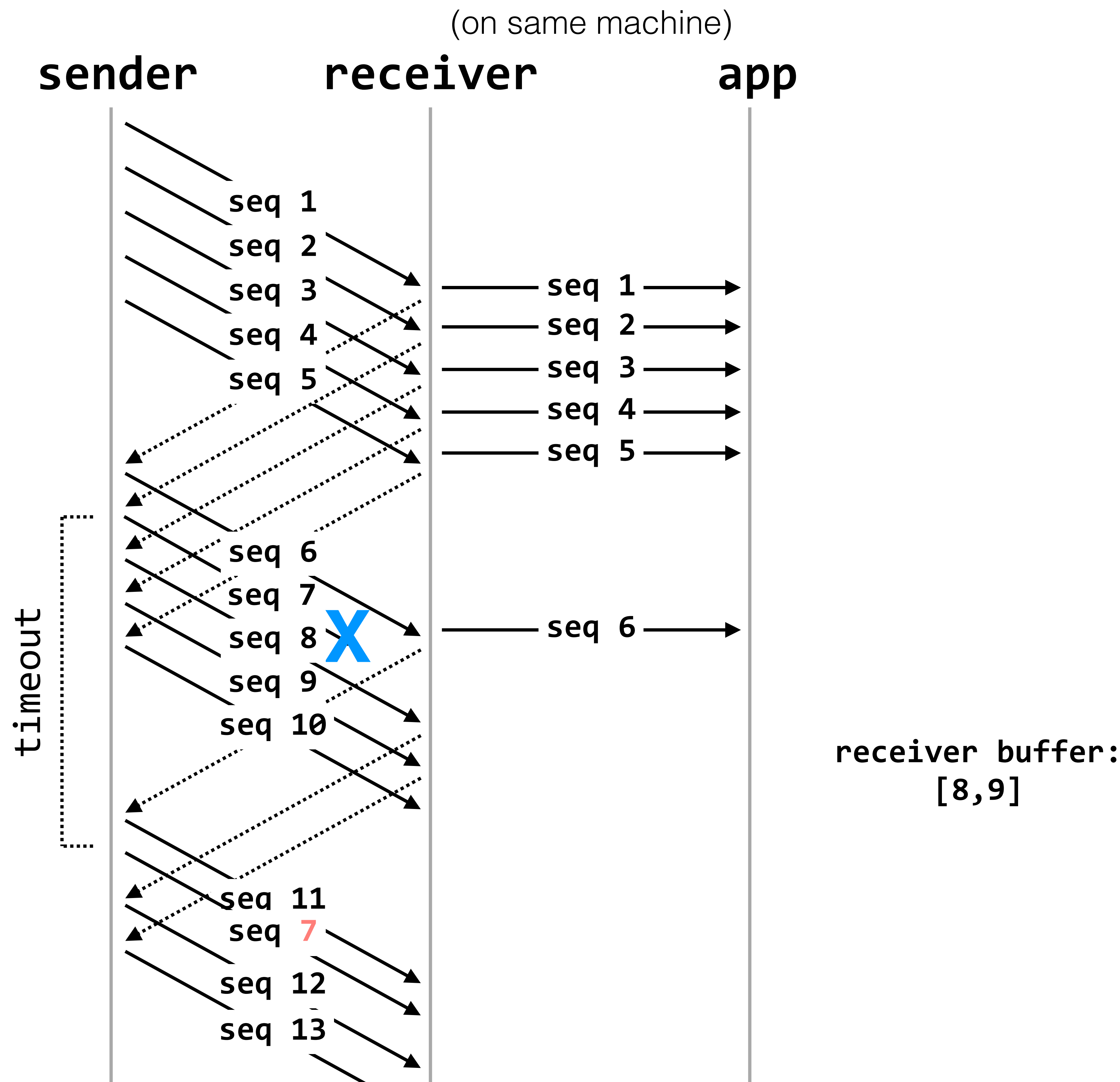
Sliding-window Protocol: Receiver



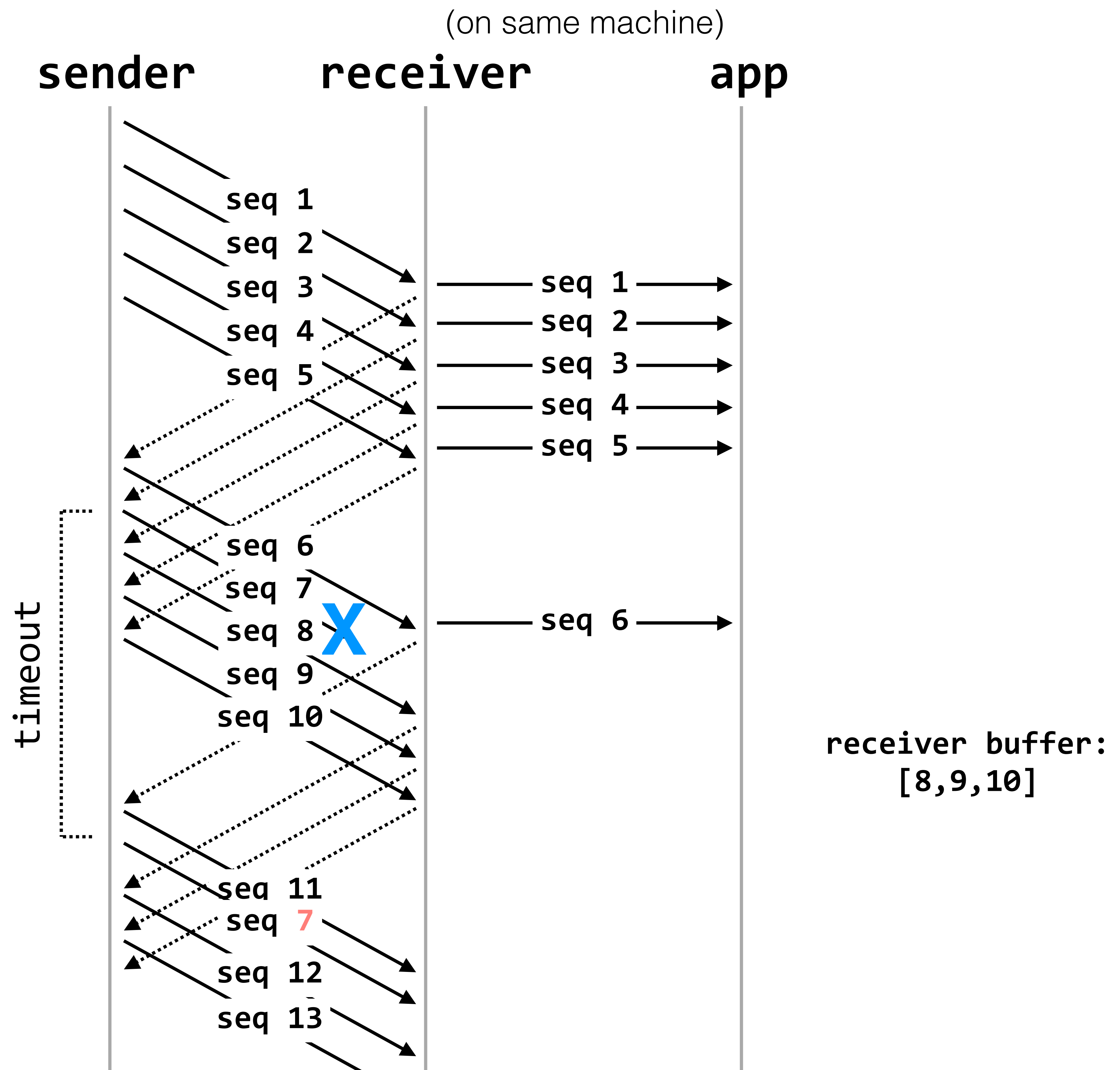
Sliding-window Protocol: Receiver



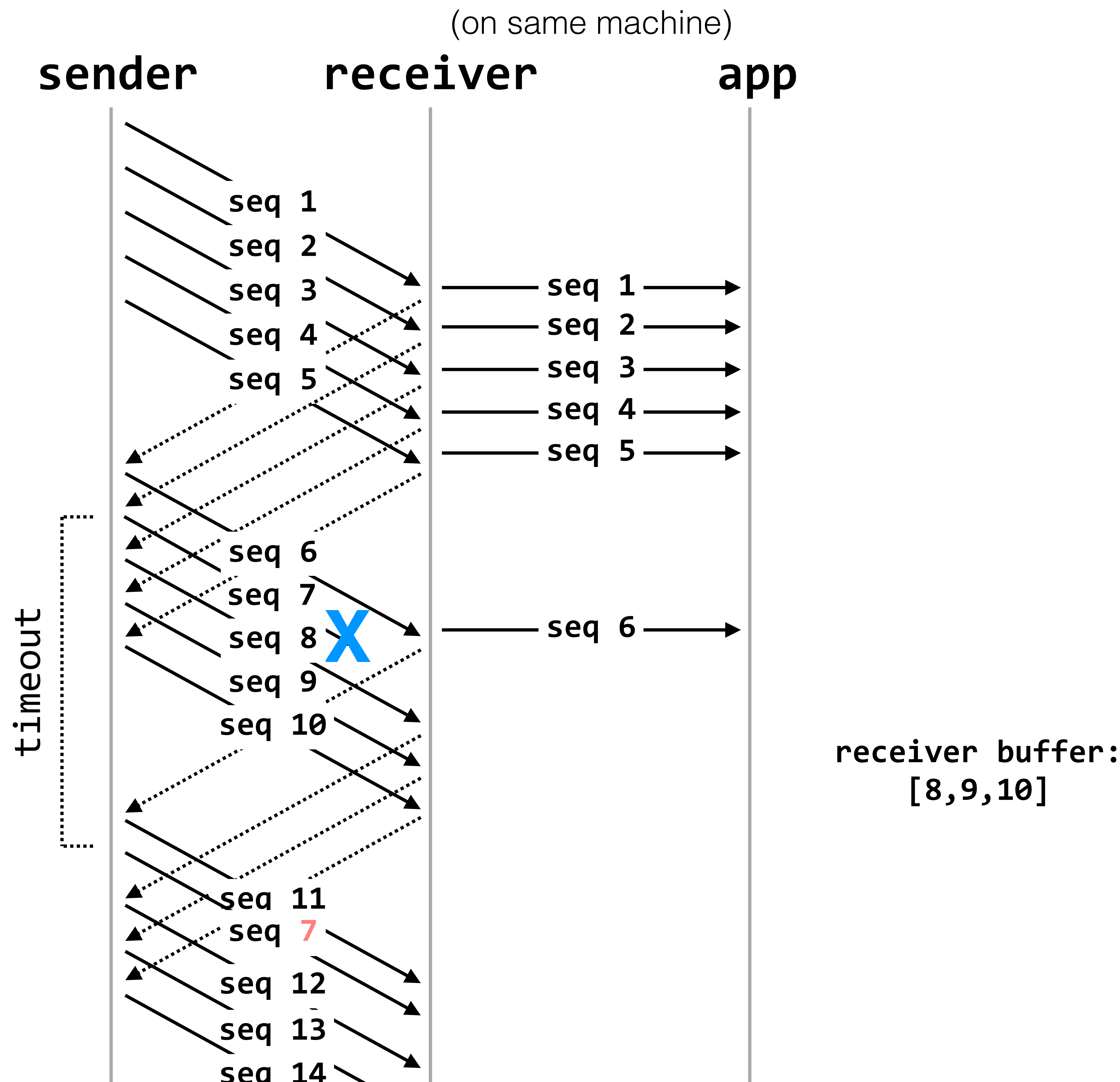
Sliding-window Protocol: Receiver



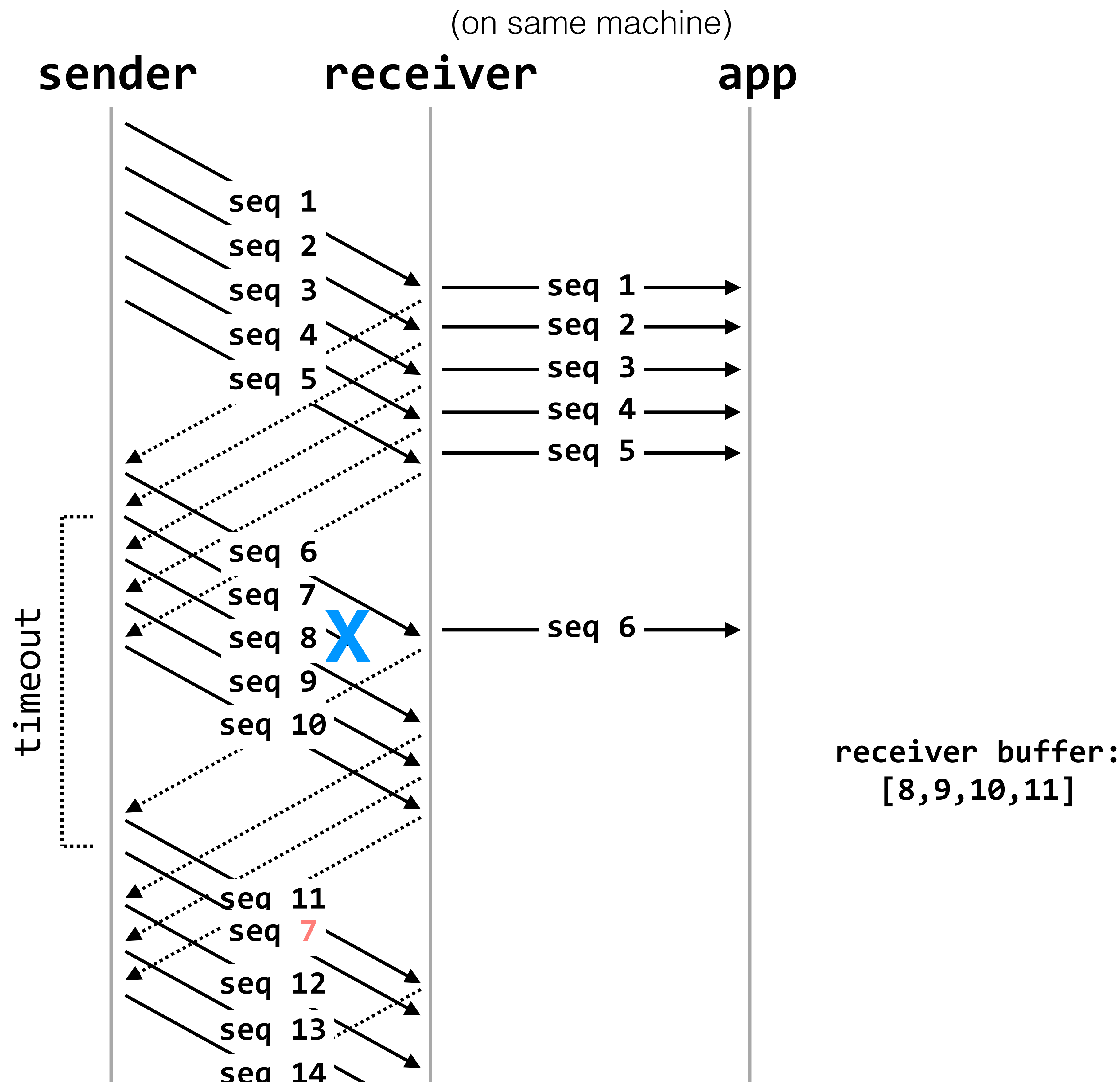
Sliding-window Protocol: Receiver



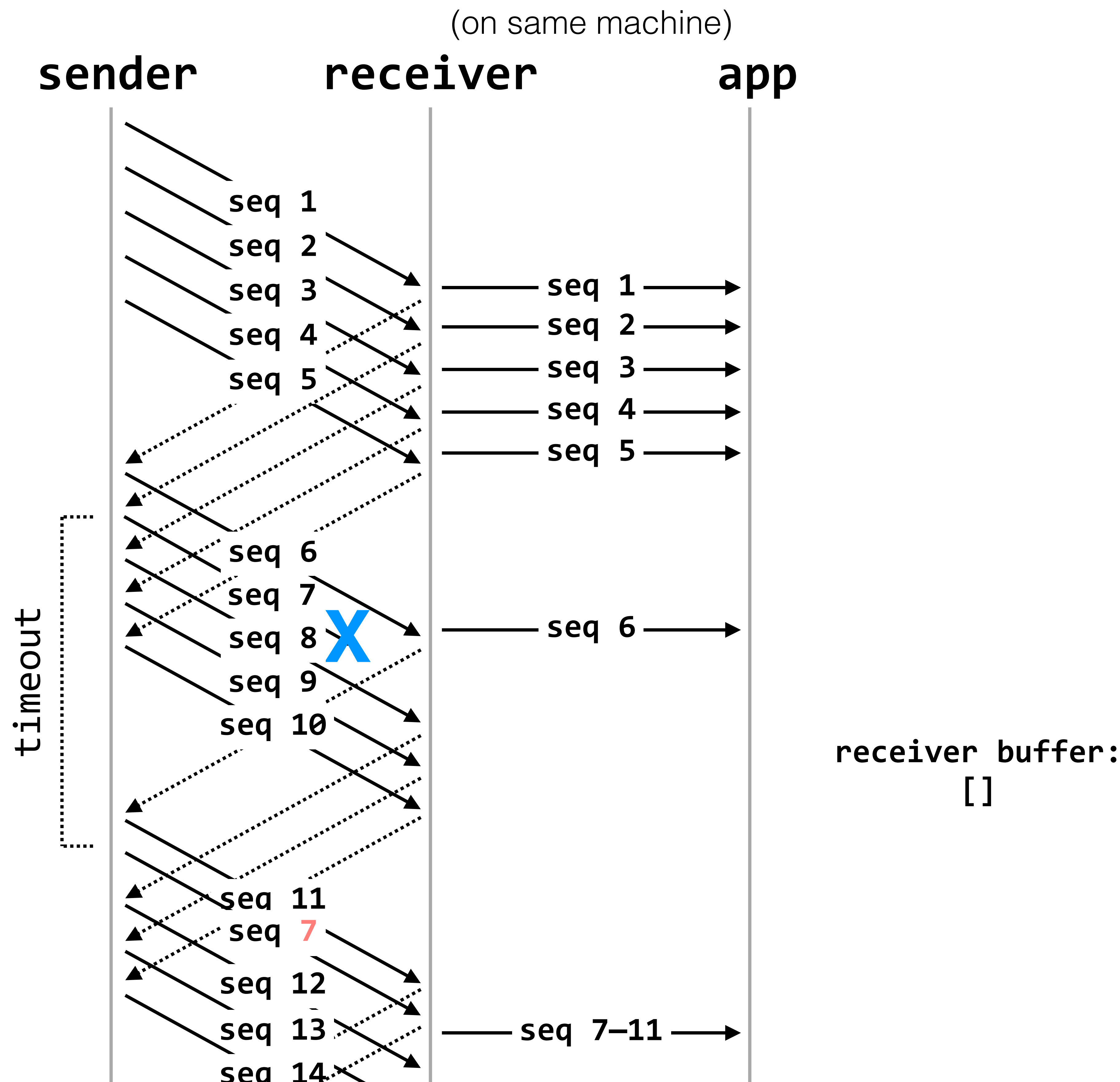
Sliding-window Protocol: Receiver



Sliding-window Protocol: Receiver

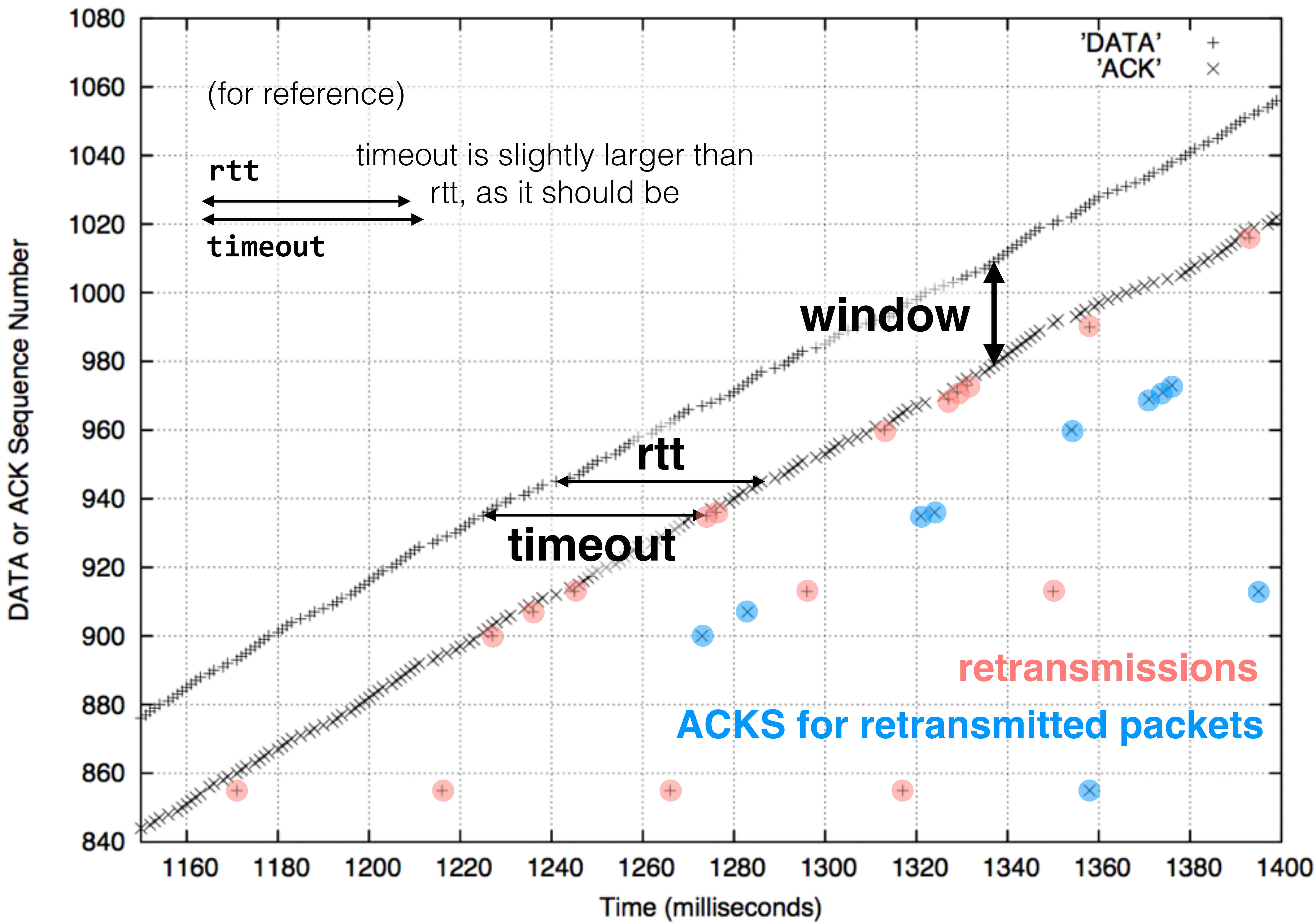


Sliding-window Protocol: Receiver



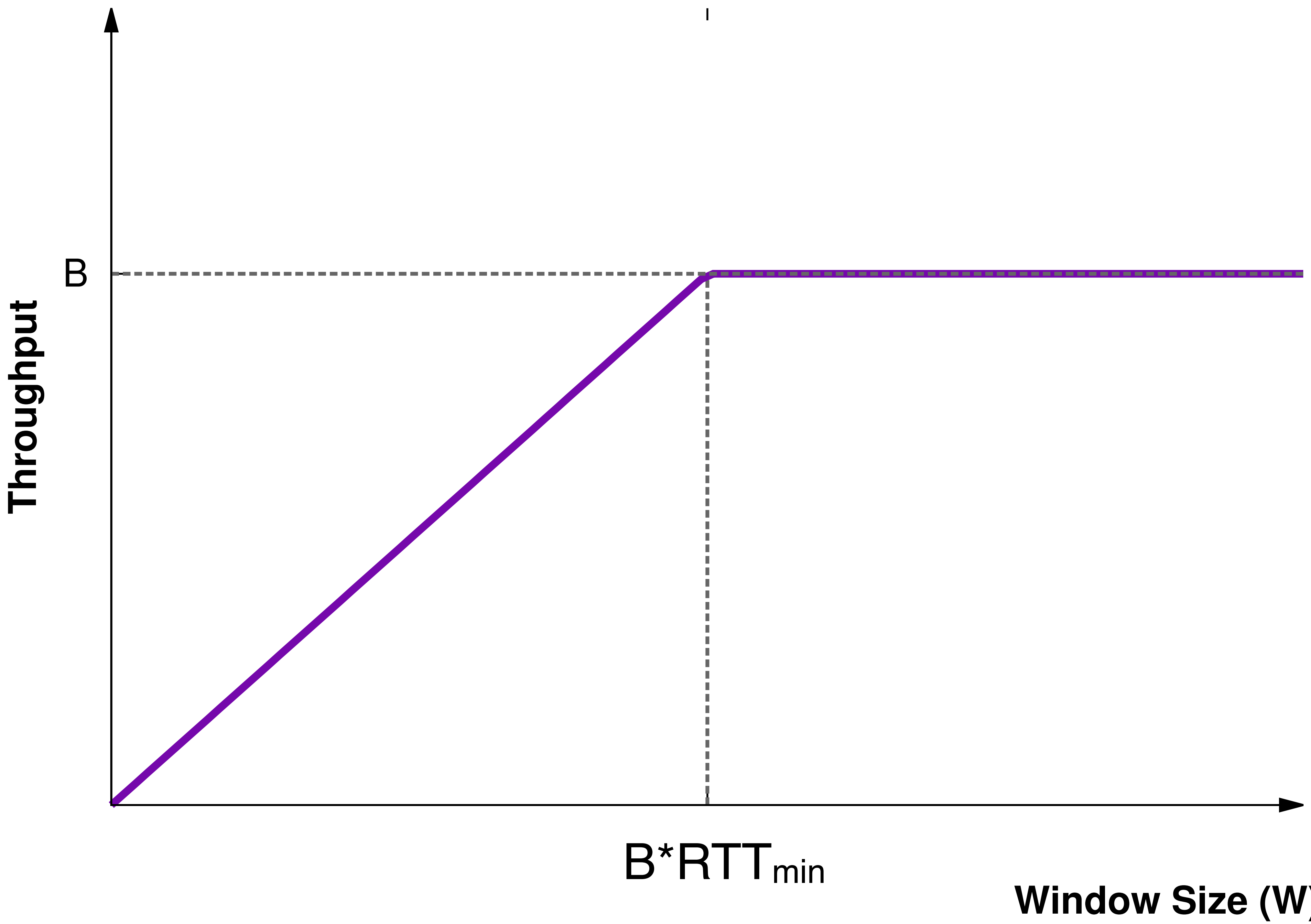
Sliding-window Protocol: Receiver

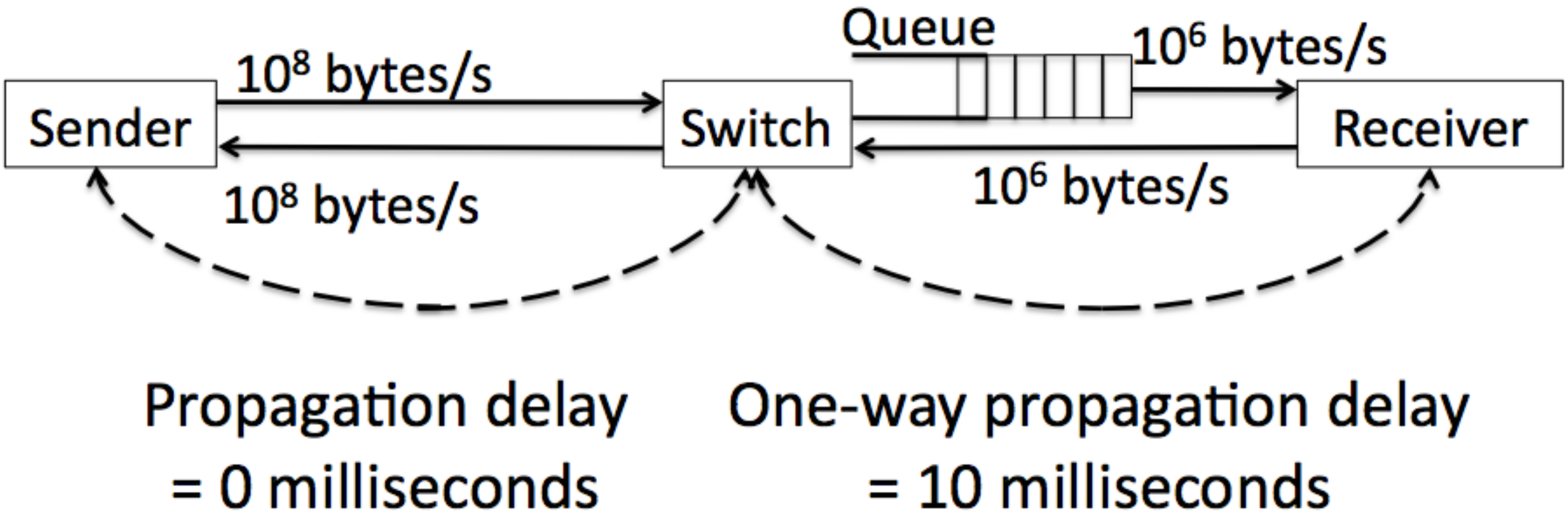
- Send an ACK for every received packet
- Save delivered packets — ignoring duplicates — in a local buffer
- Keep track of the next packet the application expects. After each reception, deliver as many in-order packets as possible.



'DATA'
'ACK'
(for reference)
rtt
timeout
timeout is slightly larger than rtt, as it should be
window
rtt
timeout
retransmissions
ACKS for retransmitted packets
DATA or ACK Sequence Number
Time (milliseconds)

Window Size vs. Throughput





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 *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

Share Tweet 168

http://arstechnica.com/information-technology/2014/11/netflix-takes-up-9-5-of-upstream-traffic-on-the-north-american-internet/

- **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