Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Networks

Assignment 1: Reliable Transport

Reliable Transport

Implement a **reliable packet stream** (not byte stream!)

… which handles:

- Packet drops
- Packet corruption
- Flow control
- Packet reordering

Fundamental Mechanisms

Error detection

- Corrupt packets must be discarded
- Implemented via checksum

Acknowledgments (ACK)

- Small control packet to confirm the reception of a packet
- When a sender gets an ACK, sender learns that recipient has successfully gotten all packets until (excl.) ackno

Fundamental Mechanisms

Time-outs

- If the sender doesn’t get an ACK after “reasonable” time, it retransmits the original packet

Naive Approach: Stop-and-Wait

Algorithm

- After transmitting one packet, sender waits for an ACK
- If the ACK doesn’t arrive in time, sender retransmits

Disadvantage

- Inefficient use of link’s capacity

Sliding Window Protocol

Objective: better utilization of link bandwidth by sending multiple unacknowledged packets

Send window (SW): set of unacknowledged packets

Receiver window (RW): set of received packets

Requirement: need to keep SW and RW synchronized

Sliding Window Definitions

"Frame" - This is a packet in the assignment, though typically it is bytes

Each frame is indivisible and has a sequence number (seqno)

Sliding Window: Sender

Sliding Window: Sender

Sent and acknowledged

Sliding Window: Sender

Sent not yet acknowledged

Sent and acknowledged

Sliding Window: Sender

Sent not yet acknowledged

Sent and acknowledged

Not sent

SND.UNA: lowest seqno of outstanding frames

SND.UNA: lowest seqno of outstanding frames

The acknowledgement number (ackno)
is the next seqno the receiver expects to get

SND.UNA

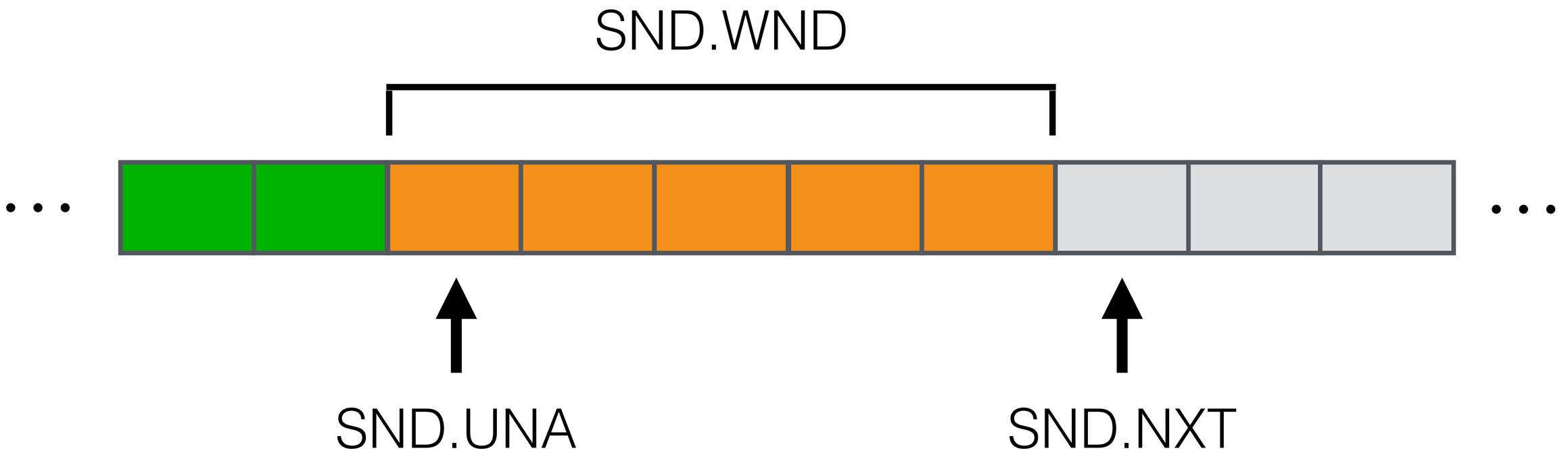
Acknowledgements are cumulative

SND.UNA = max(SND.UNA, ackno)
when an ACK arrives

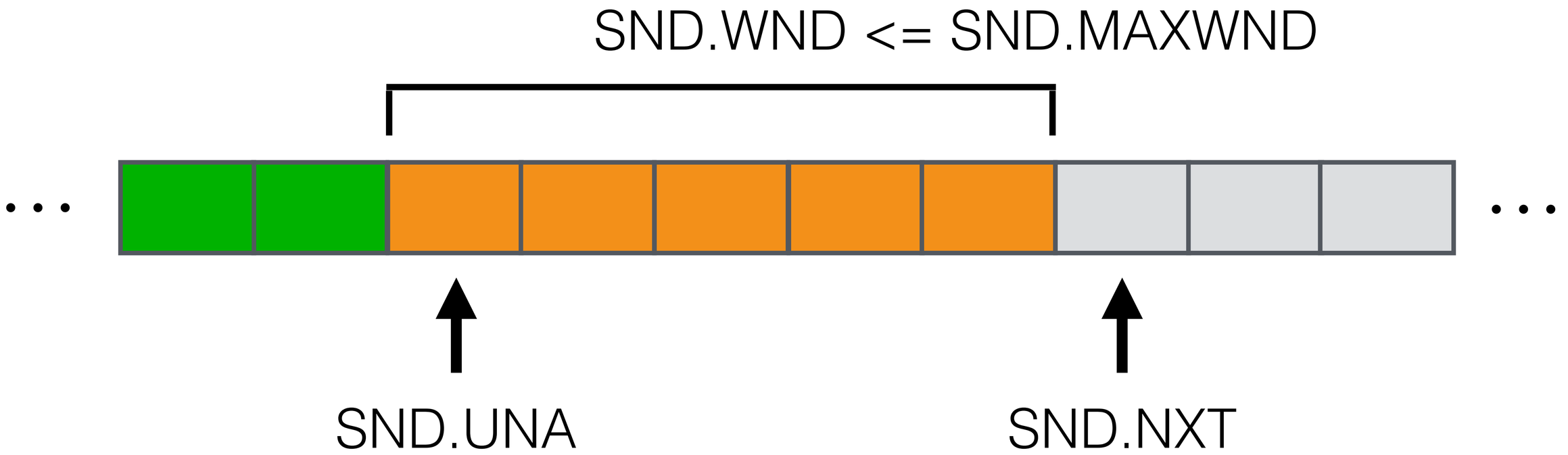
We don’t do selective acknowledgement in this assignment

SND.NXT: seqno of next frame to send out

SND.WND = SND.NXT - SND.UNA



SND.WND <= SND.MAXWND



Timeouts

Associate timeouts with each frame sent
Retransmit if no ACK received before timeout

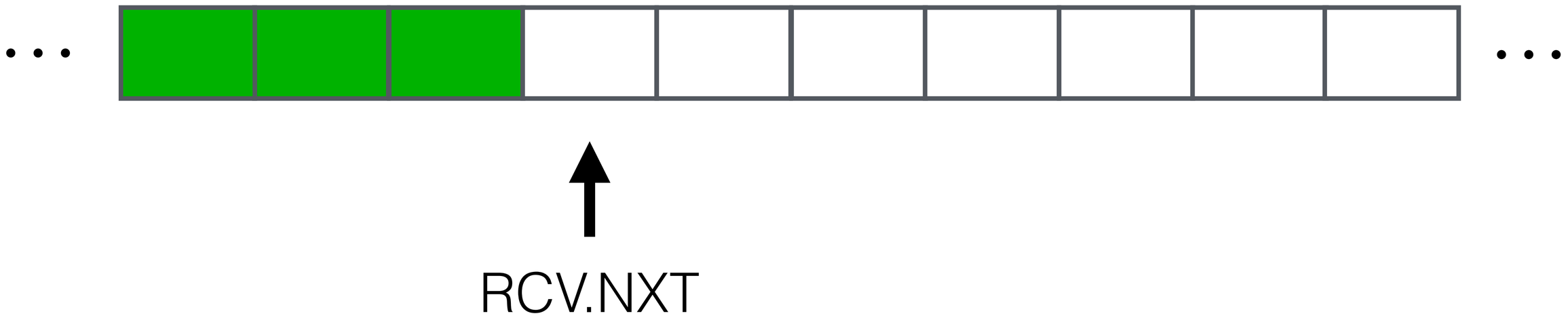
Sender summary

- **State:** SND.UNA, SND.NXT
- SND.WND = SND.NXT - SND.UNA
- Send window is not a constant!
- SND.WND <= SND.MAXWND
- Upon reception of ACK(ackno): SND.UNA = max(SND.UNA, ackno)
- Timeouts resend frames within the SND.WND
- You can only send a next “new” frame with seqno = SND.NXT if SND.NXT < SND.UNA + SND.MAXWND
- After sending frame with seqno = SND.NXT, SND.NXT = SND.NXT + 1

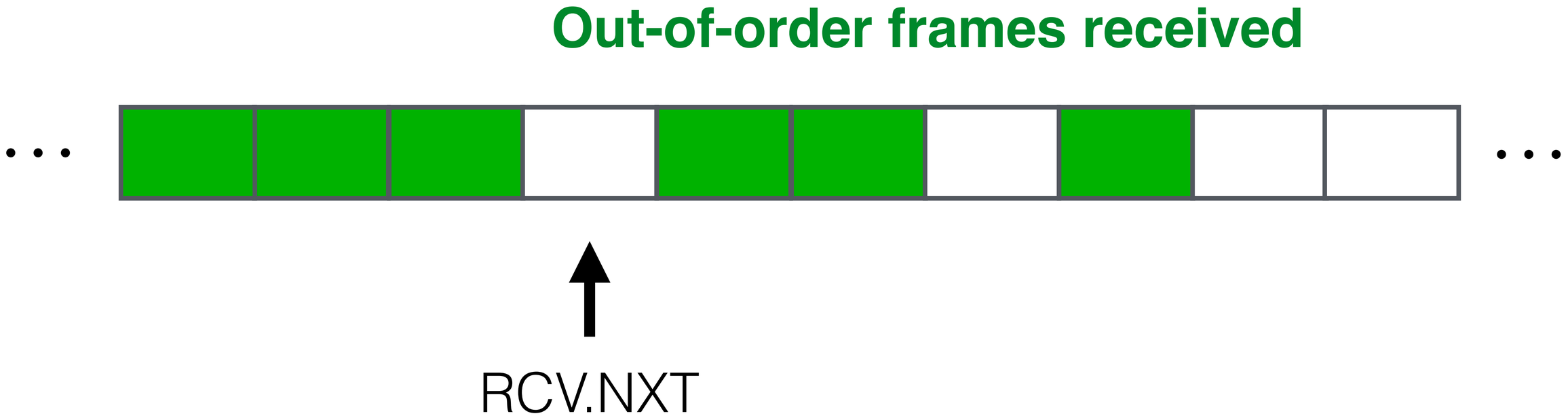
Sliding Window: Receiver

RCV.NXT: next seqno expected

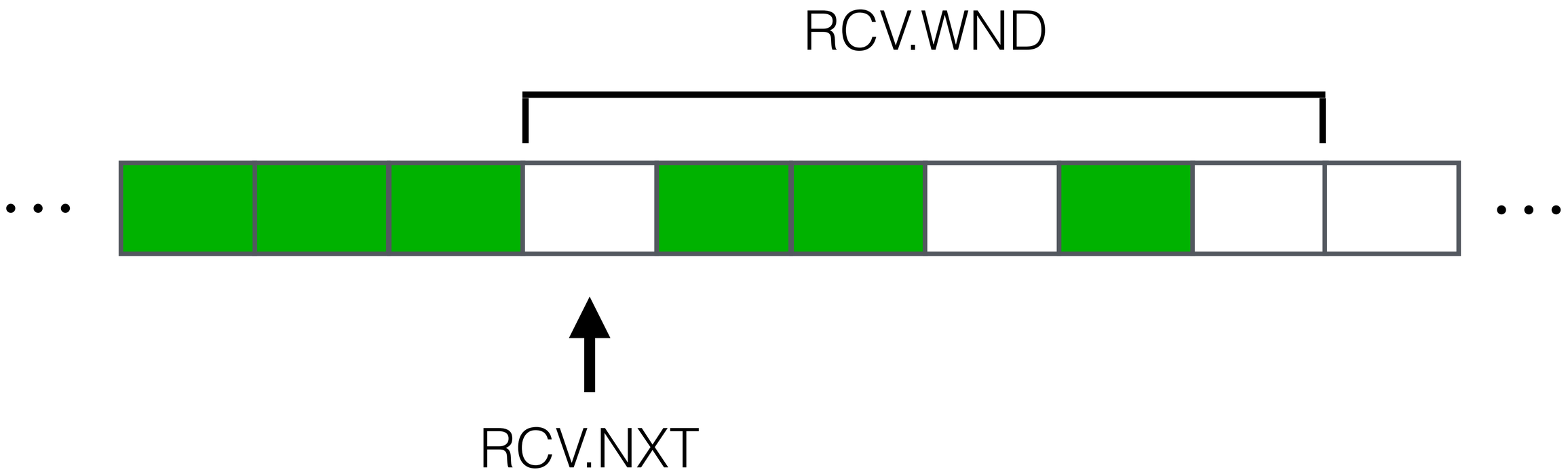
Received everything till here



RCV.NXT: next seqno expected



RCV.WND: max. # of out-of-order frames it will accept



Receiver summary

- **State:** RCV.NXT
- RCV.WND = RCV.MAXWND

Receiver summary: upon receiving frame

If seqno >= RCV.NXT + RCV.WND

    Drop frame

Else

    Store in the buffer if not already there

If seqno == RCV.NXT:

    Set RCV.NXT to the highest seqno consecutively stored in the buffer + 1

    Release data [seqno, RCV.NXT - 1] to application layer

Send back ACK with cumulative ackno = RCV.NXT