CS 348

Computer Networks

Lec 16
Sliding Window Protocols

Spring 2020 IIT Goa

Course Instructor: Dr. Neha Karanjkar

Note: These slides are adapted from “Computer Networking: A Top-down Approach” by Kurose & Ross, 7th ed

Reliable Data Transfer

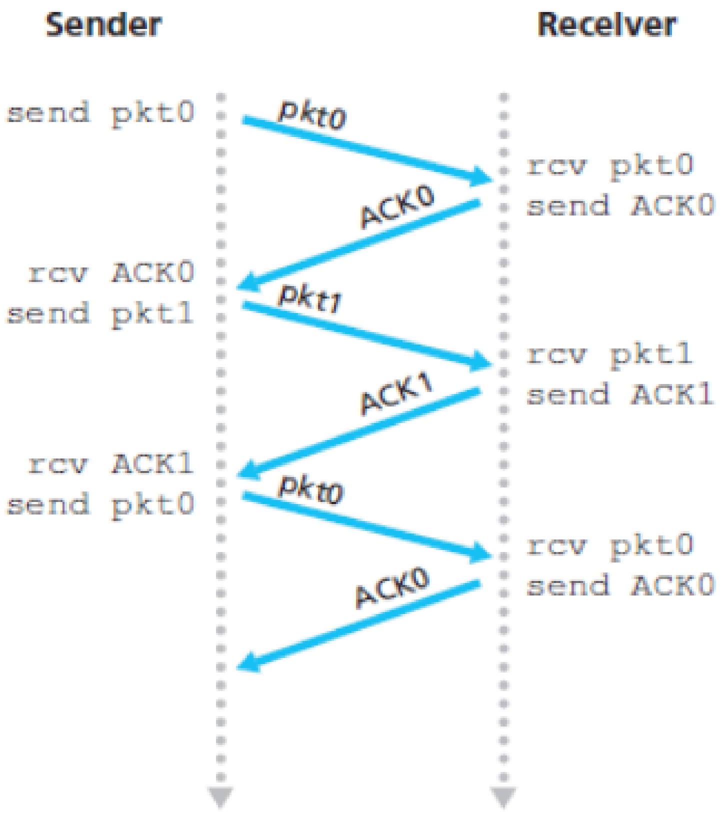
- How can reliable data transfer be possible using the unreliable delivery service provided by the Network layer?

Reliable Data Transfer

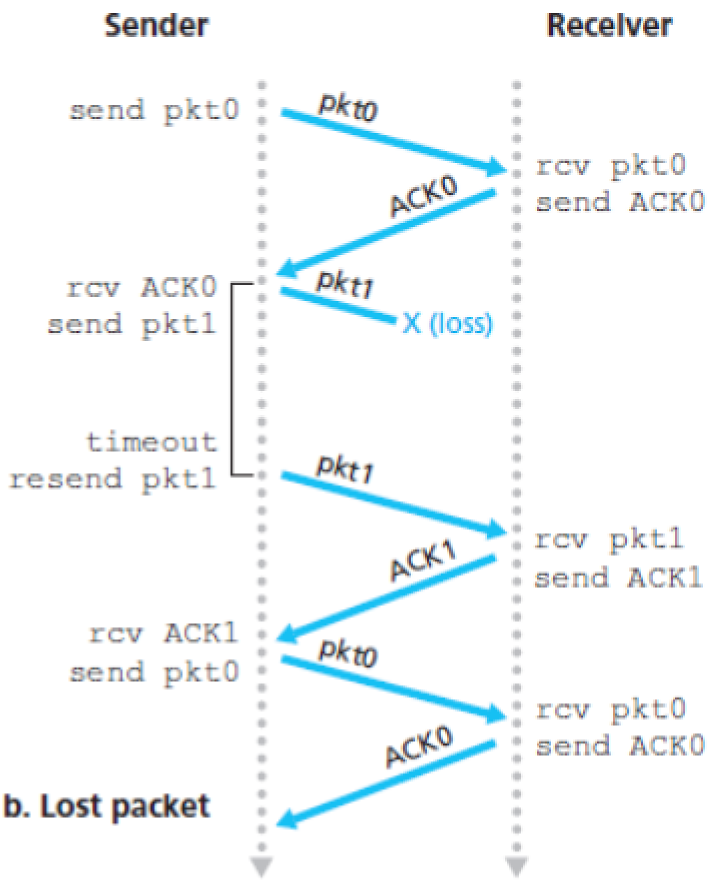
- How can reliable data transfer be possible using the unreliable delivery service provided by the Network layer?
  - Need a mechanism to detect that a packet is corrupted: **Checksums**
  - Receiver sends **acknowledgements**
  - Packets need to be numbered to detect duplicates: **Sequence Num**
  - Resend packets upon a **timeout**: To counter packet loss
- Example: rdt3.0

Stop and Wait Protocols

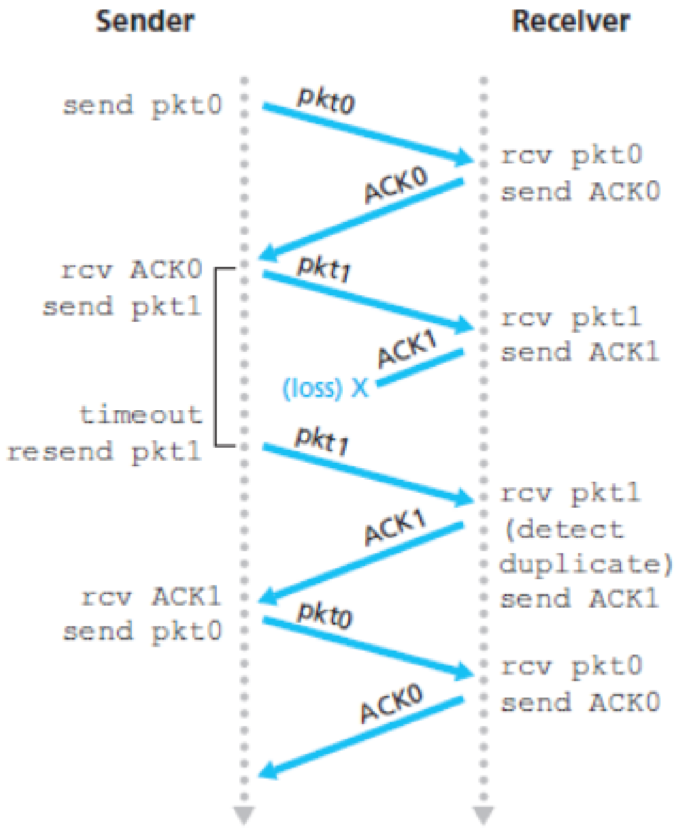
- Protocols such as rdt3.0 are known as stop-and-wait protocols
- Also known as alternating bit protocols.



Stop and Wait Protocols (rdt3.0)

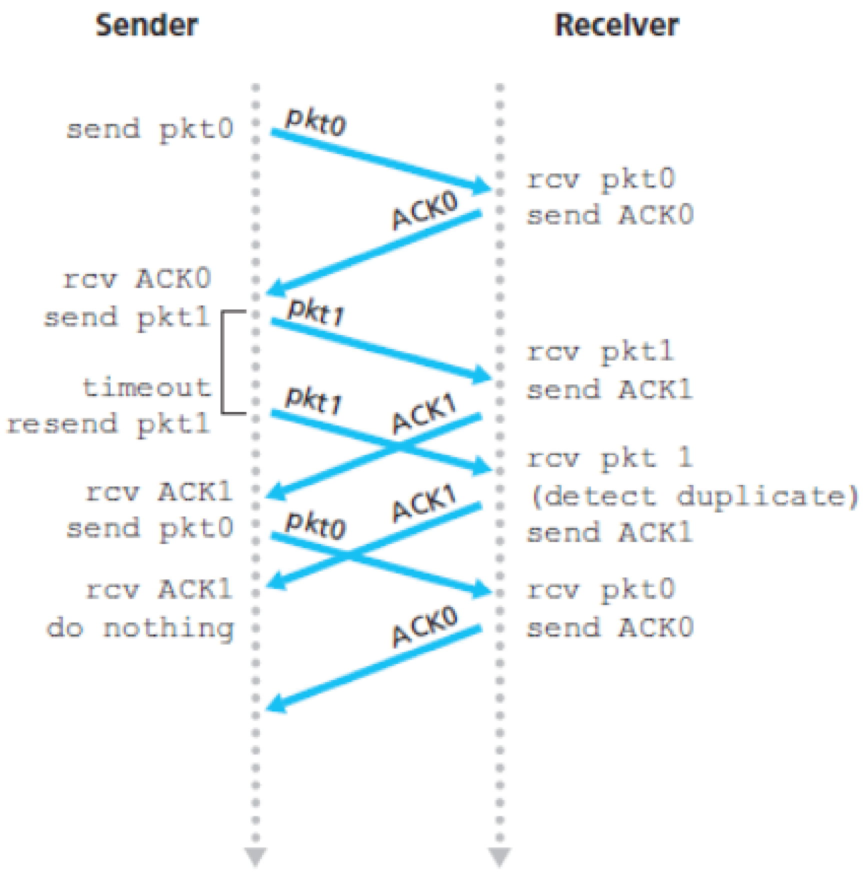


b. Lost packet



c. Lost ACK

Stop and Wait Protocols (rdt3.0)

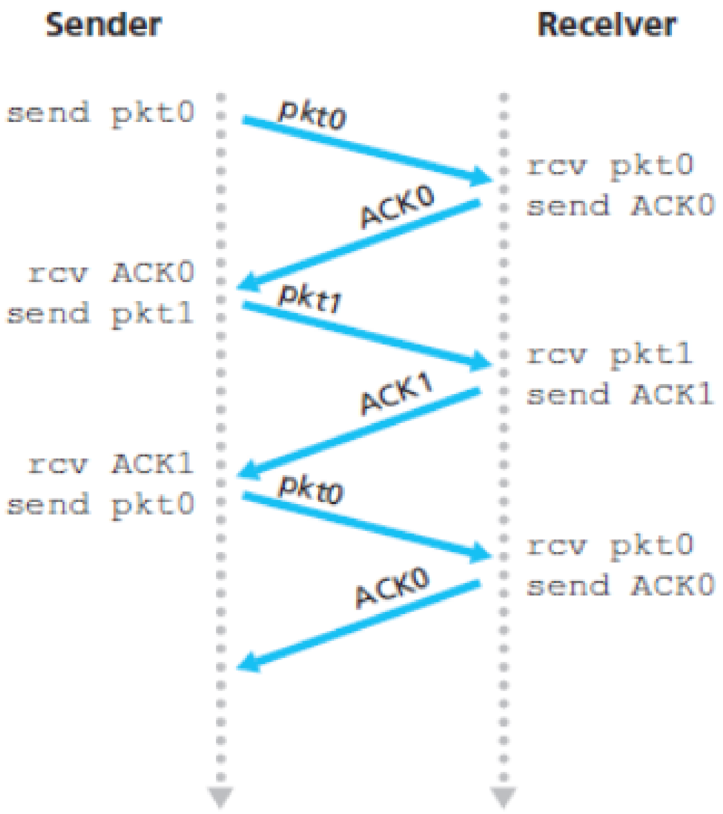


d. Premature timeout

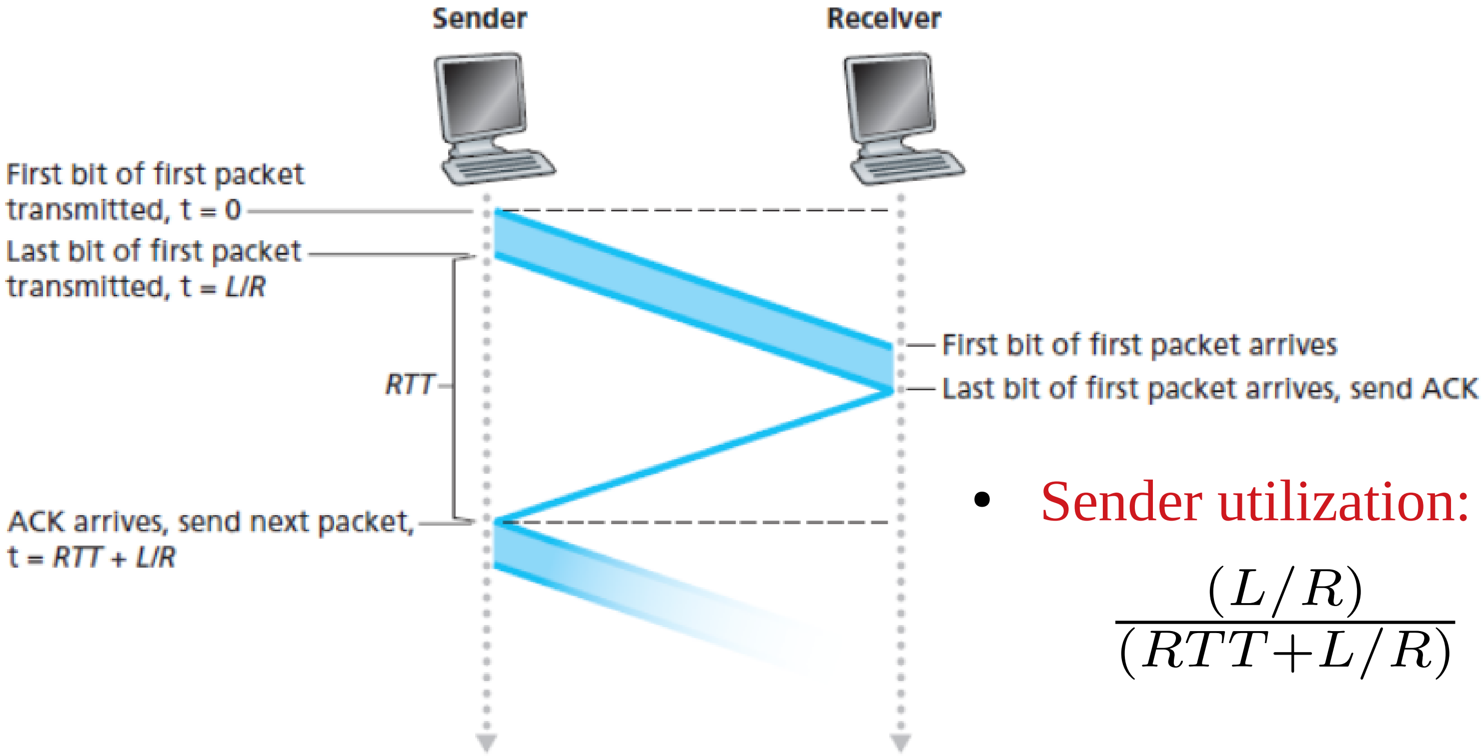
Some issues with rdt3.0

- Doesn't work if it is possible for packets to be re-ordered on the channel.
- **Poor performance.**
  - Need to wait for an ack

    (Round Trip Time) before next packet can be sent.



Some issues with rdt3.0

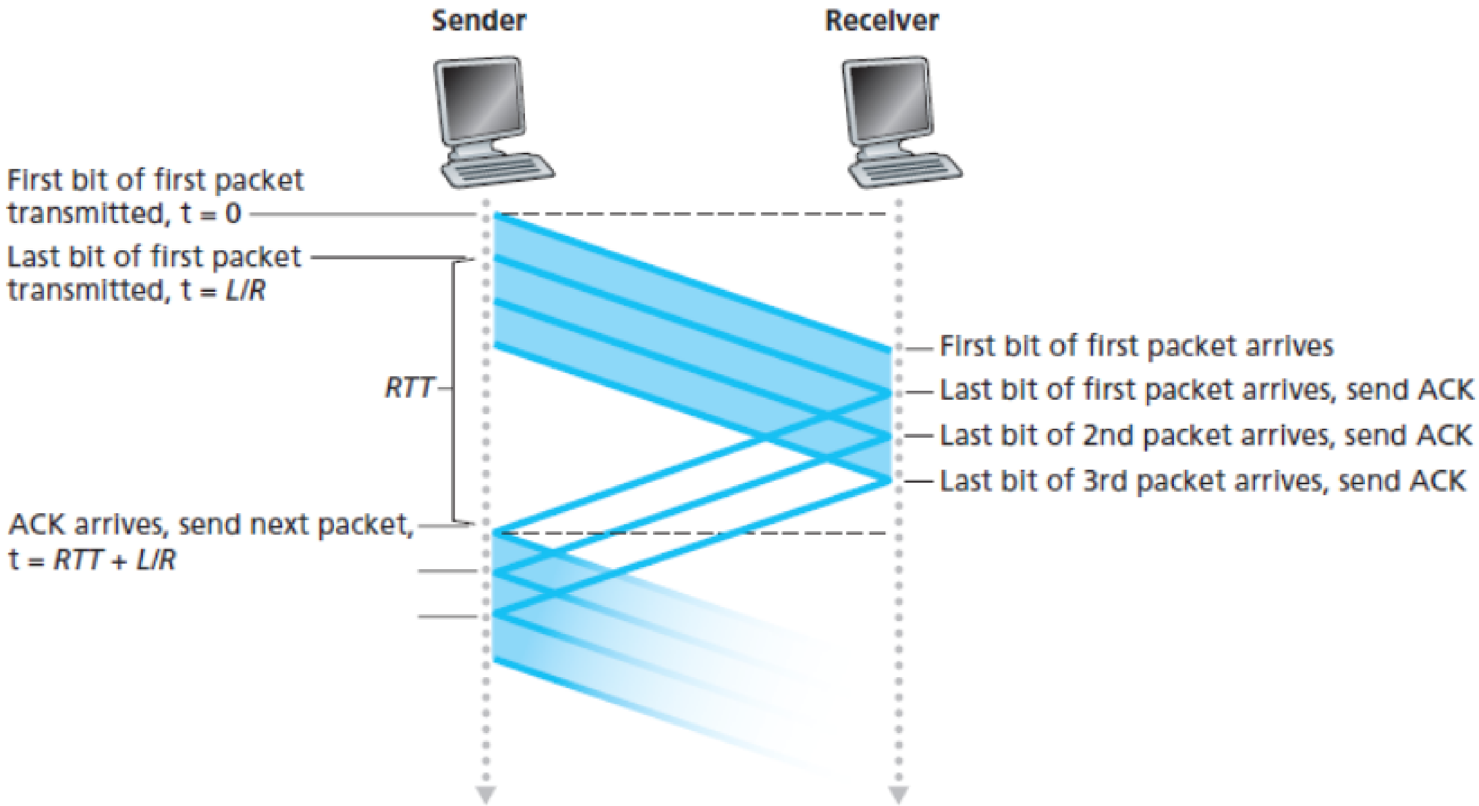


a. Stop-and-wait operation

- Sender utilization:

$$\frac{(L/R)}{(RTT+L/R)}$$

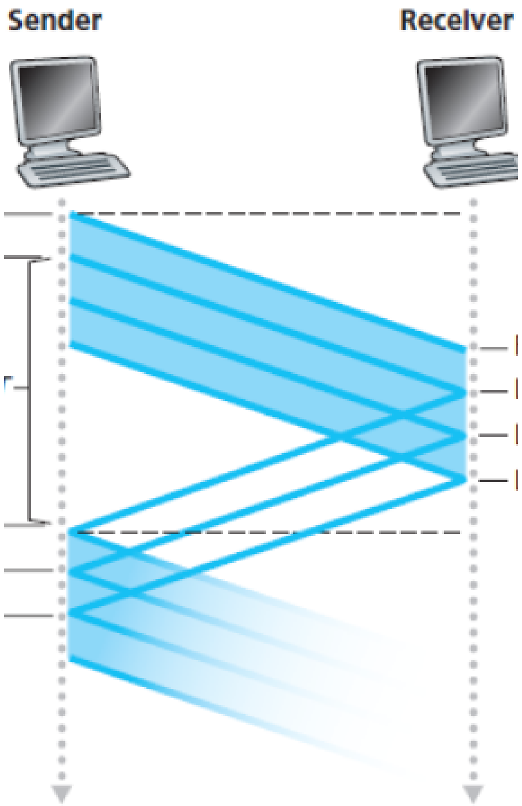
Pipelined Data Transfer



- How can we develop a protocol for this?

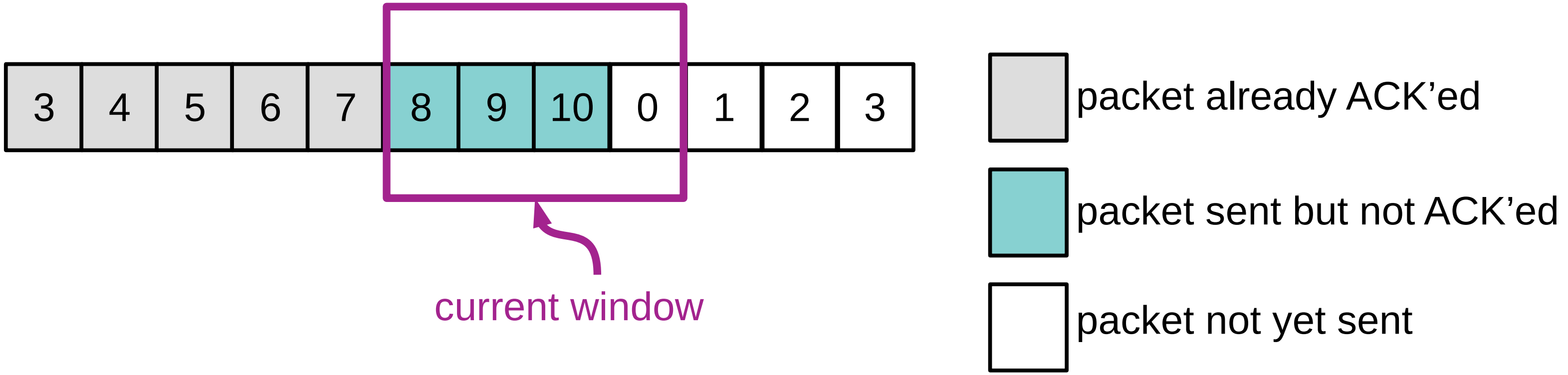
Pipelined Data Transfer

- How can we develop a protocol for this?
  - Larger range of sequence numbers (but finite)
  - Buffering at the sender and receiver



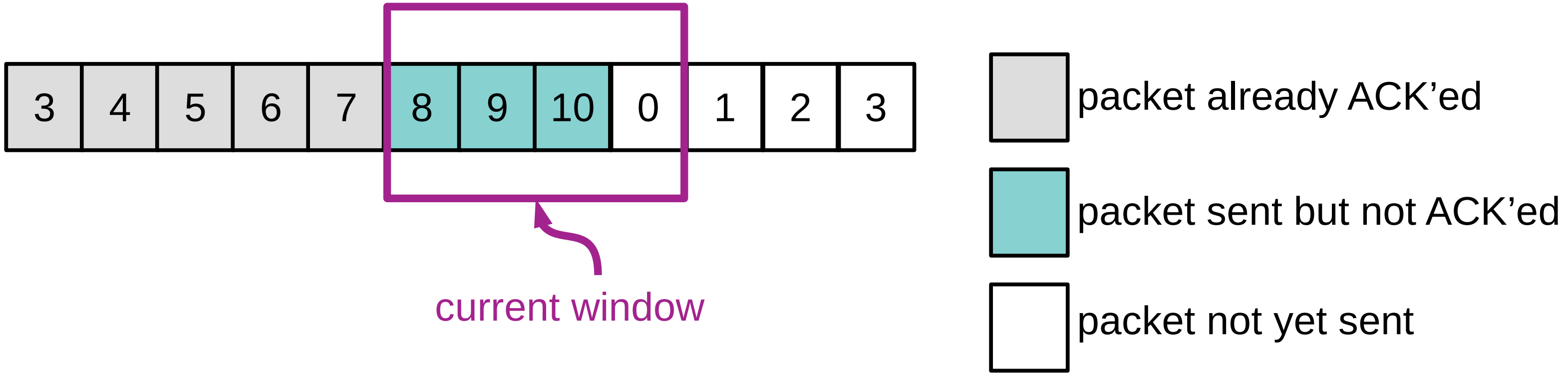
Sliding Window Protocols

- **Window Size (N)**: The maximum number of packets that can be sent without waiting for an acknowledgement
- **Current Window:** The packets that have been sent or can be sent without waiting for an ACK.
  - “**Slide**” the window to the right after receiving acknowledgement for the *oldest* unacknowledged packet



Sliding Window Protocols

- **Two Approaches:**
  - Go-Back-N (GBN)
  - Selective Repeat (SR)



Go-Back-N

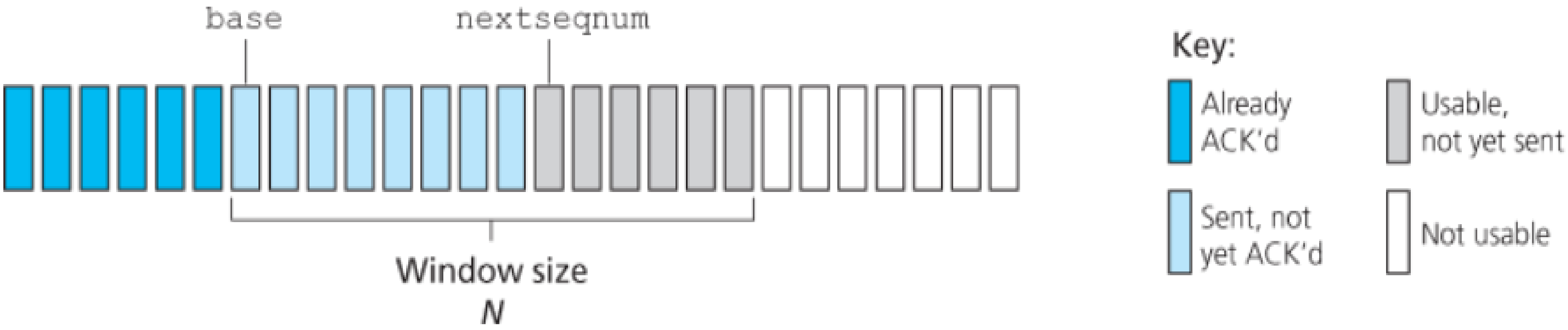


Figure 3.19 Sender's view of sequence numbers in Go-Back-N

Go-Back-N

- **Sender has a Window of size N**
- **Receiver**
  - simply discards all packets **except** the packet with the next expected sequence number
  - sends a **cumulative** ACK(*n*) implying all packets upto *n* were successfully received **in order**.
- **Sender:**
  - Maintains a single timer for the oldest unacknowledged packet (*base*)
    - If ACK(*base*) is received, slide window to the right
    - Else if timeout occurs, **re-transmit** all packets in interval [*base*, *nextseqnum-1*] , hence the name Go-Back-N

Go-Back-N

- Interactive applet:

  https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/go-back-n-protocol/index.html

Go-Back-N

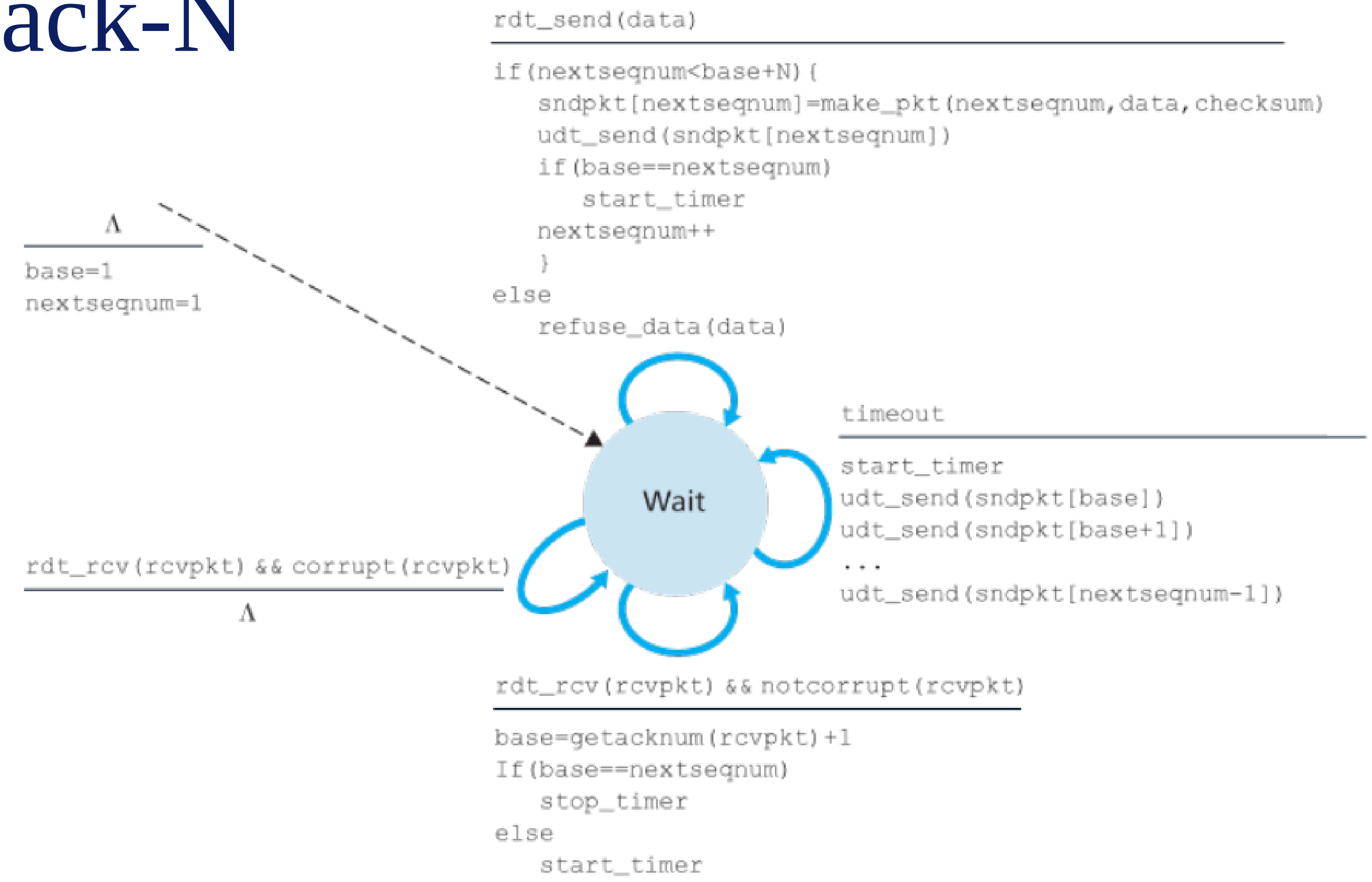


Figure 3.20 Extended FSM description of the GBN sender

Go-Back-N

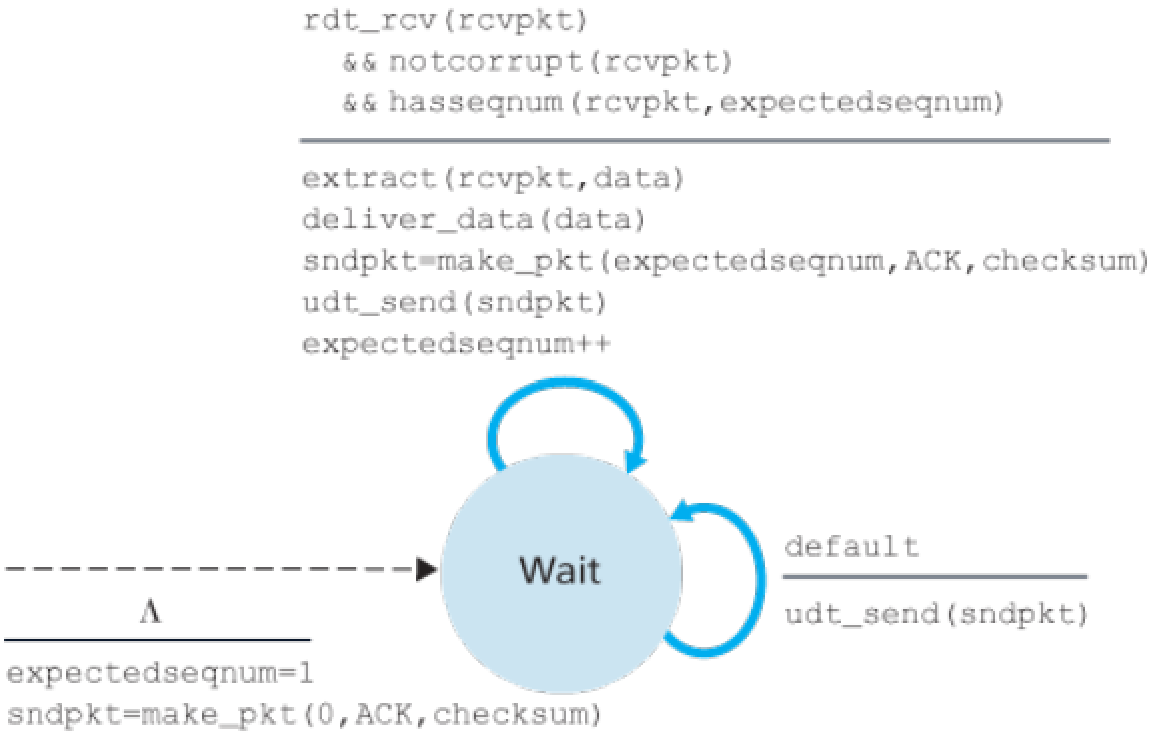


Figure 3.21 Extended FSM description of the GBN receiver

Issues With Go-Back-N

- **Why restart the timer** upon receiving duplicate ACKs of older packets? Seems to simply delay the re-transmission of a lost packet...

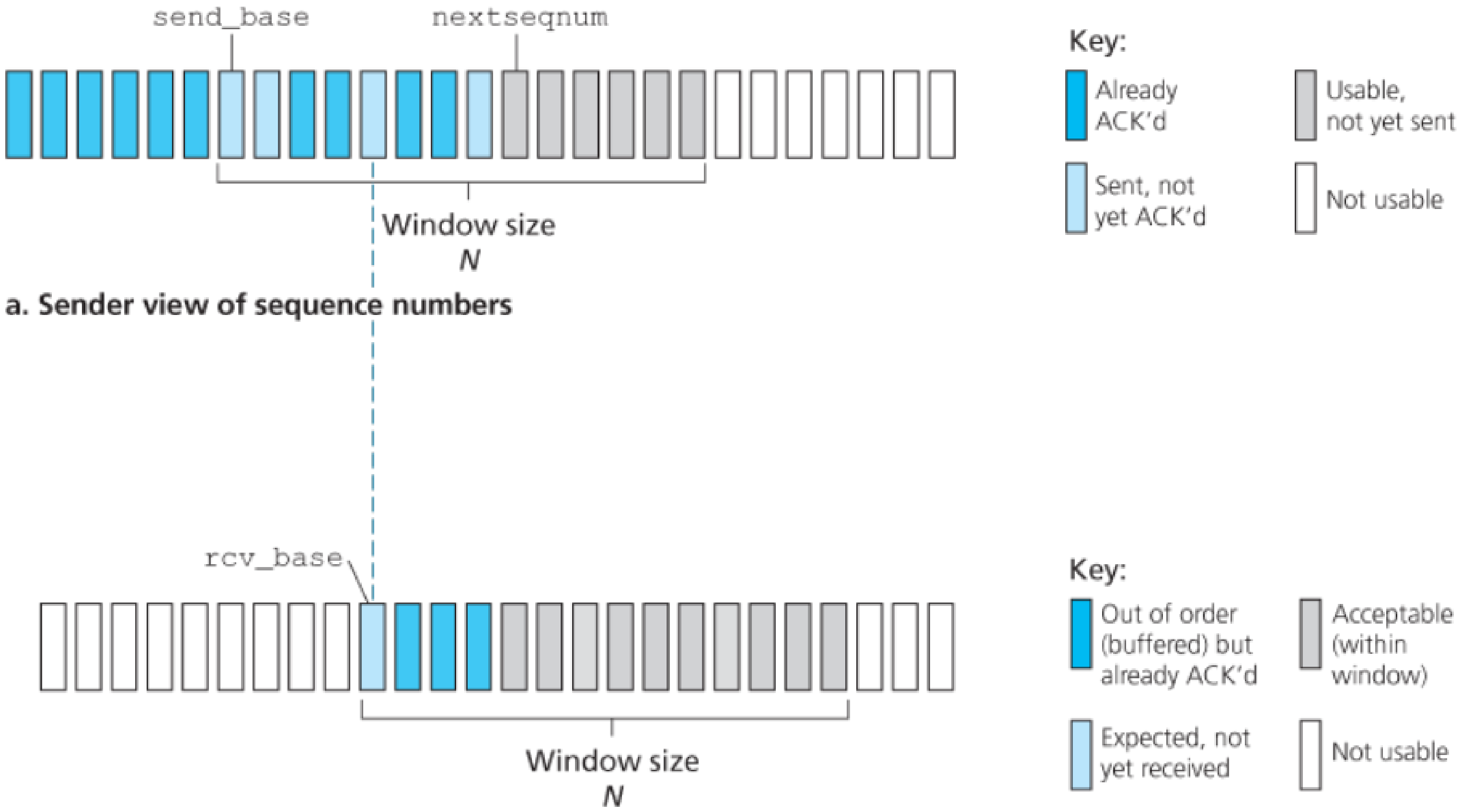
- A single packet error causes GBN to retransmit a large number of packets which fill up the pipeline. **Why discard packets received out of order? Seems wasteful...**
  - Can the receiver buffer packets received out of order?
  - Can the sender selectively retransmit only those packets that were lost/unacked... ?

--> This is essenstially what the **Selective Repeat** (SR) protocol does.

Selective Repeat (SR) Protocol

- **Sender:**
  - Maintains a timer for each packet sent and currently unacknowledged
  - Selectively re-transmits only those packets that were possibly lost/corrupted as indicated by no ACK received before timeout
- **Receiver:**
  - Sends individual acknowledgements for each packet that is correctly received, even those received out of order.
  - Buffers packets received out of order until the missing packets are received, and sends a batch of packets in sequence to the application layer.

Selective Repeat (SR) Protocol



a. Sender view of sequence numbers

b. Receiver view of sequence numbers

Selective Repeat (SR) Protocol

- Interactive applet:

  https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html

Selective Repeat (SR) Protocol

- Some questions about the nitty-gritty:
  - Do the sender's and the receiver's windows always move in sync?
  - Why should the receiver send ACKs for packets received with sequence numbers *below* the current window base?
  - Can the receiver mistake a re-transmitted packet for a fresh (new) packet?
    - Example, Seq numbers: 0,1,2,3,0,1,2,3,0,1,2,3..., Window size=3

Selective Repeat (SR) Protocol

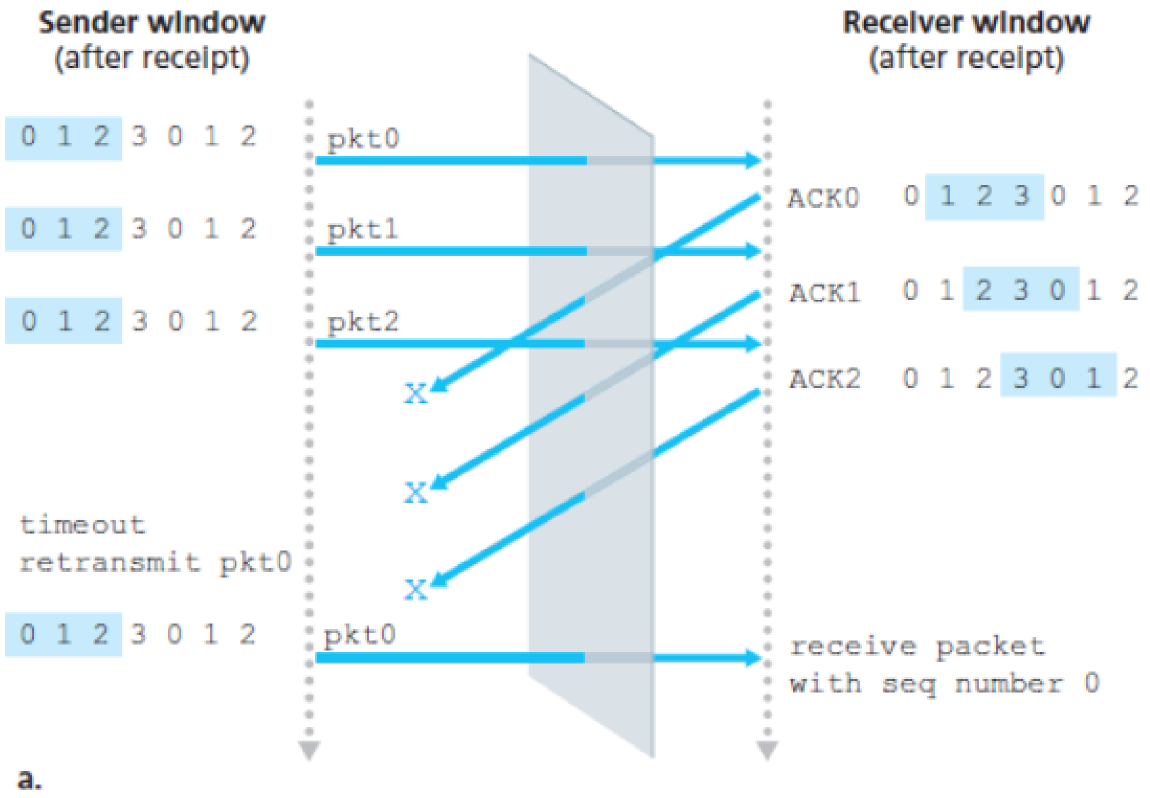


Figure 3.27 SR receiver dilemma with too-large windows: A new packet or a retransmission?

Selective Repeat (SR) Protocol

- Can the receiver mistake a re-transmitted packet for a fresh (new) packet?

  YES!

  - How to mitigate this?
  - Choose: *Window size <= size of the sequence number space* / 2
  - Eg. If sequence numbers range from 0 to 9,

    Window size should be <=5

Summary

- Components of a Reliable Data Transfer Protocol
  - Checksums
  - Acknowledgements
  - Sequence Numbers
  - Timers/Timeouts
  - Windows, Buffering and Pipelining
- Caveat:
  - Neither Go-Back-N nor SR can work if we assume that packets can get re-ordered. Why? How is this mitigated in the real world?
    - Do not re-use sequence number until a certain time elapses.

Exercise

- **[LAB] Design, simulate and test the GBN and SR protocols**

  Using Python+SimPy

References and Reading Assignment

- **Kurose and Ross 6th ed:** Section 3.4

So far...

- Structure and Physical components of the Internet
- Design of the Internet: Layering and Encapsulation
- The Applications Layer:
  - Sockets Interface
  - The Web and HTTP
  - DNS
- **The Transport Layer: how it works**
  - **Basic services, UDP**
  - **Principles of Reliable Data Transfer (rdt 3.0 etc)**
  - **Pipelined data transfer (Sliding window protocols)**
  - **TCP details**
  - **Congestion and Flow control**