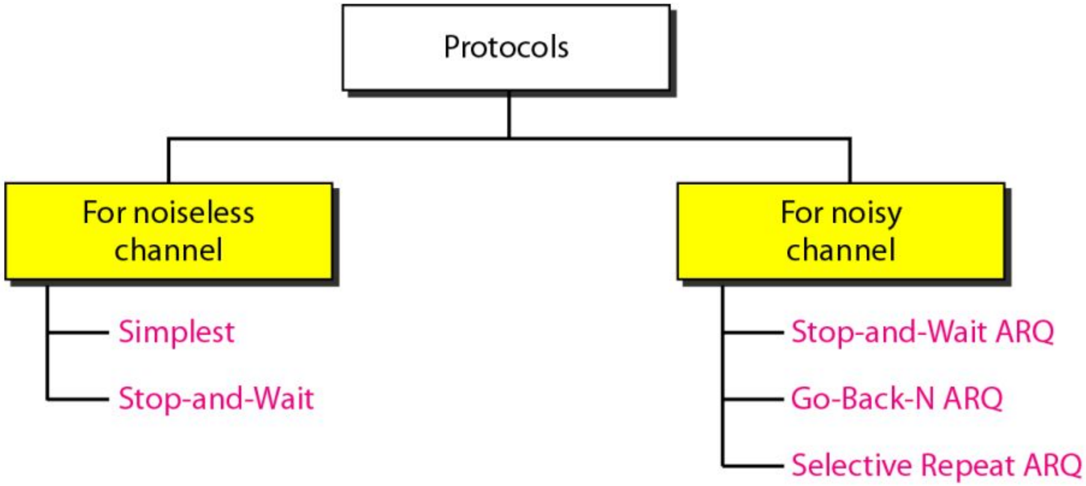
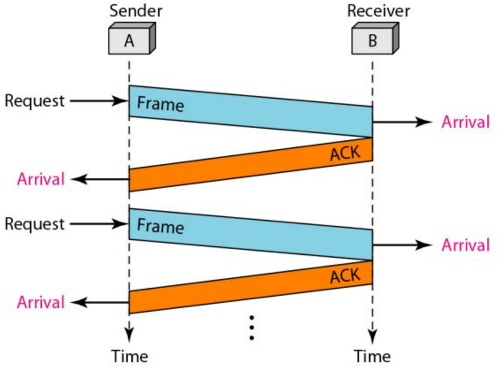
# Error Control in Communication

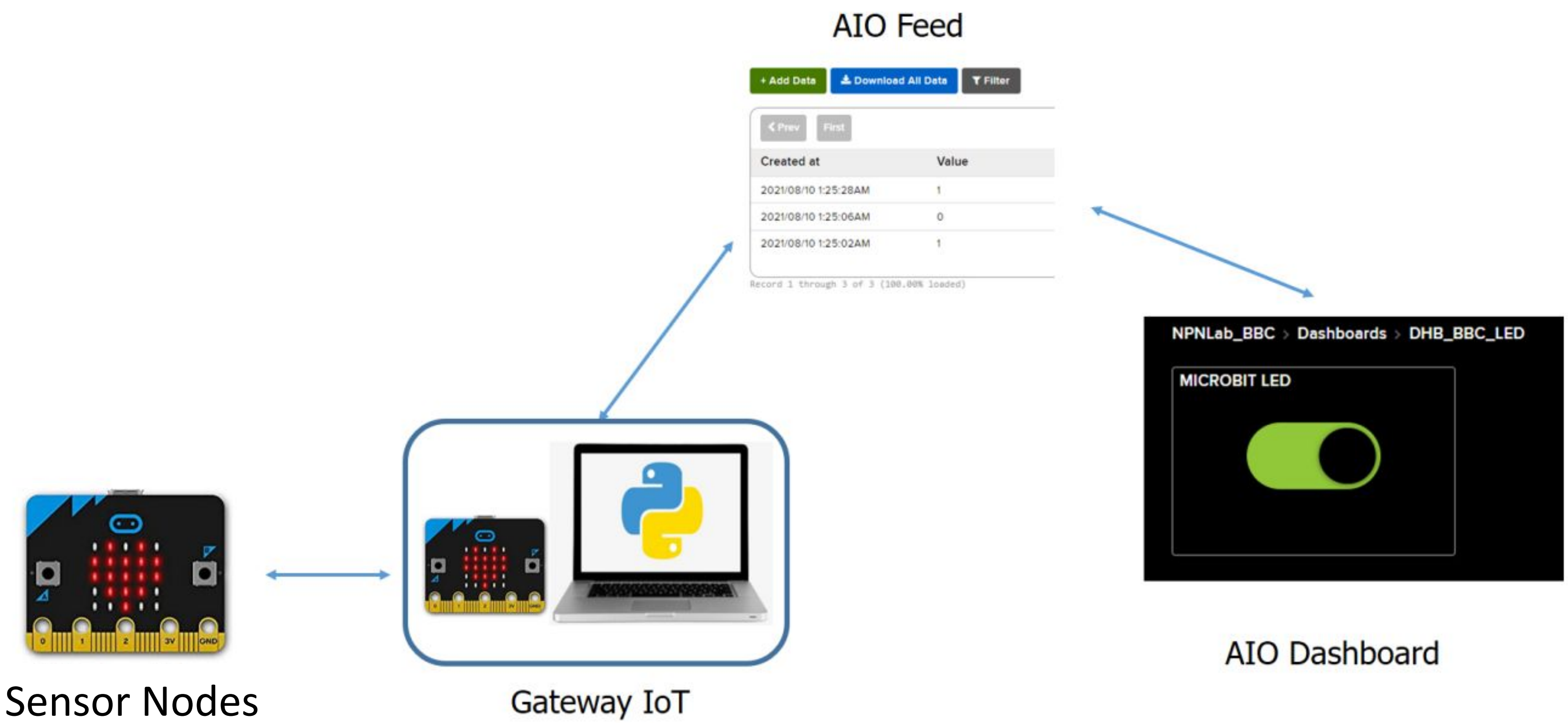



DCE

DEPT. OF COMPUTER ENGINEERING

# System Architecture

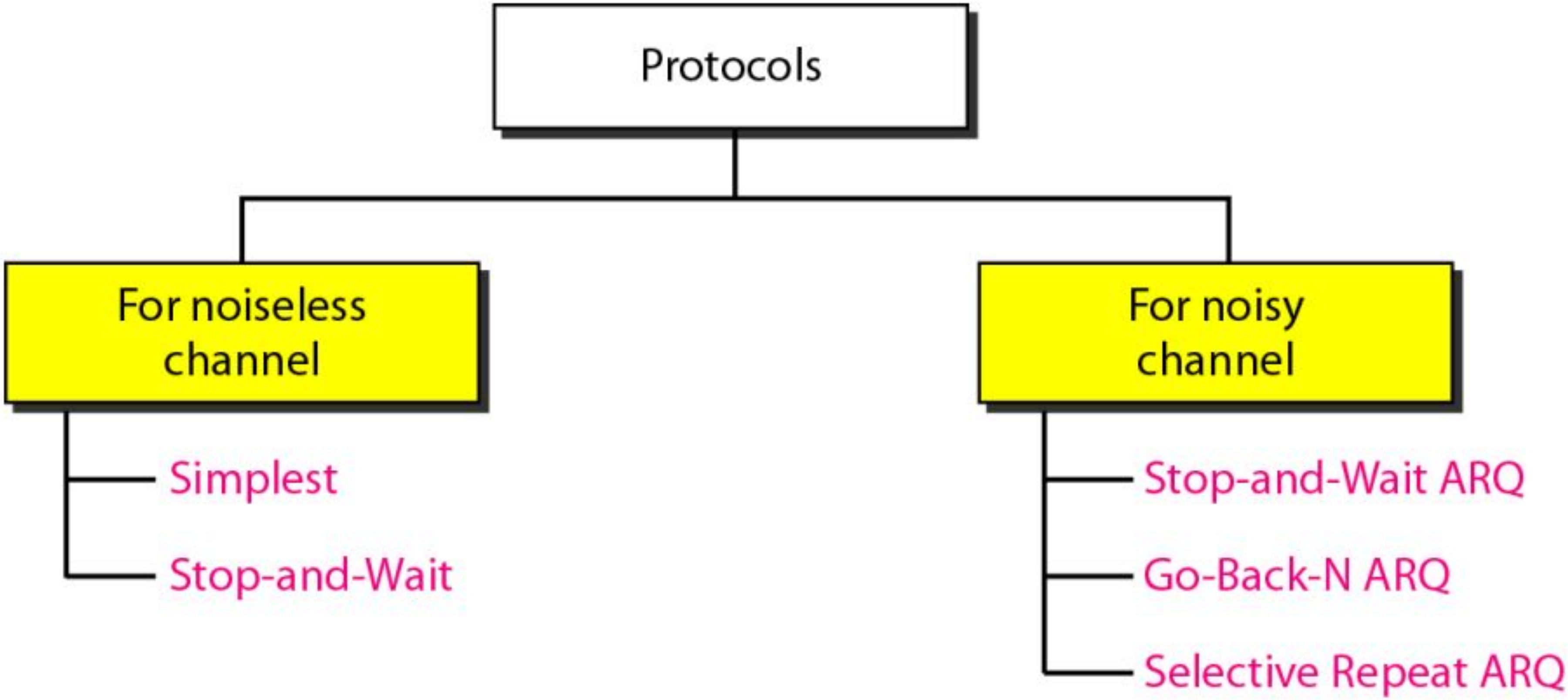


# Communication Protocol

- The combination of **framing**, **flow control**, and **error control** to achieve the delivery of data from one node to another.

- The protocols are normally **implemented in software** by using one of the common programming languages.

# Classification of Protocols



# NOISELESS CHANNELS

- No frames are lost, duplicated, or corrupted
- **Simplest Protocol** – has no flow or error control
- **Stop-and-Wait Protocol** – sender sends one frame, stops until it receives agree from receiver and then sends the next frame

# Simplest Protocol

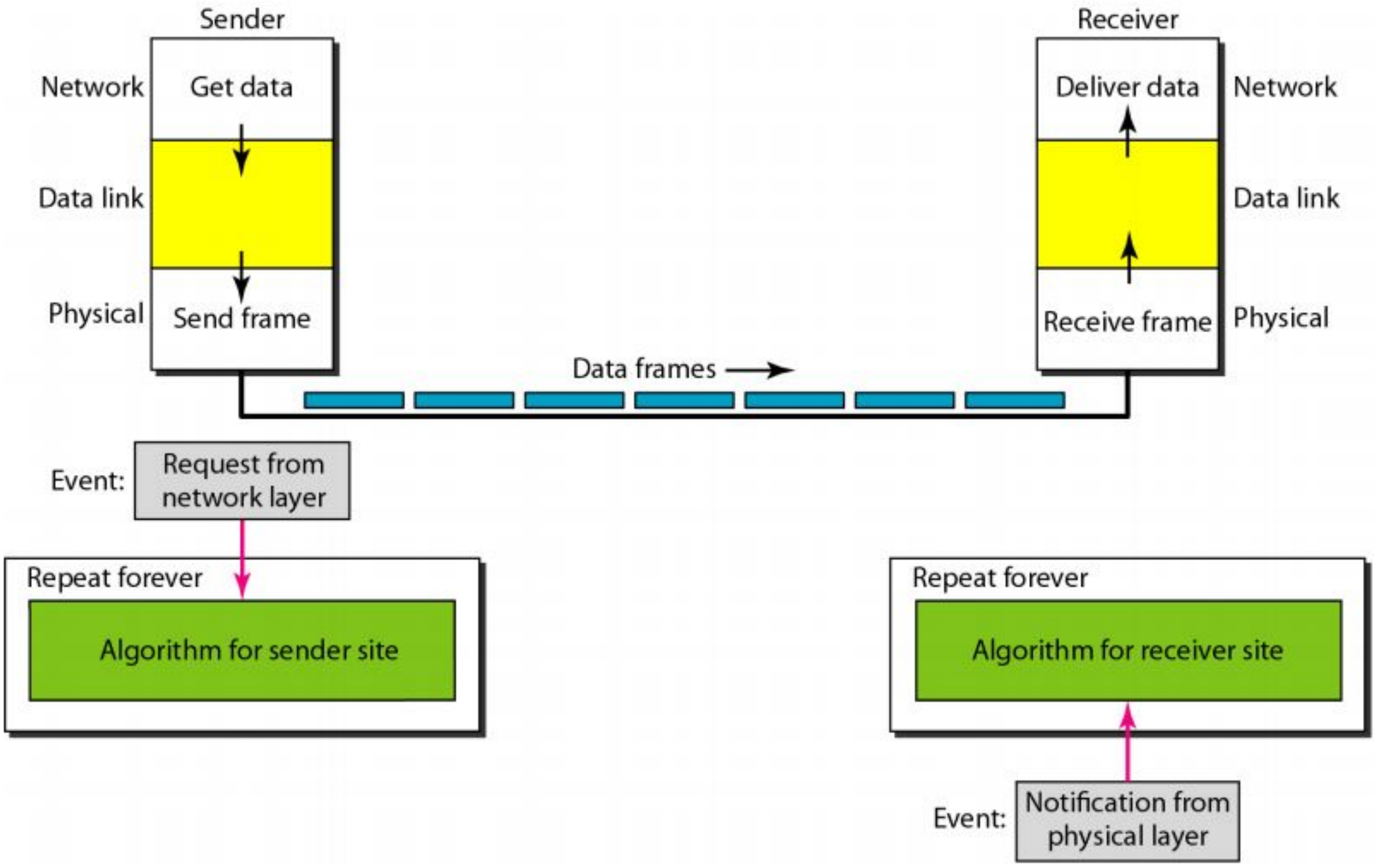
- **Unidirectional protocol:** data frames are traveling in only **one direction-from the sender to receiver**.

- The receiver can immediately handle any frame it receives with a **processing time that is small enough** to be negligible.

- The data link layer of the receiver immediately removes the header from the frame and hands the data packet to network layer, which can also accept the packet immediately.

# Simplest Protocol Design

message

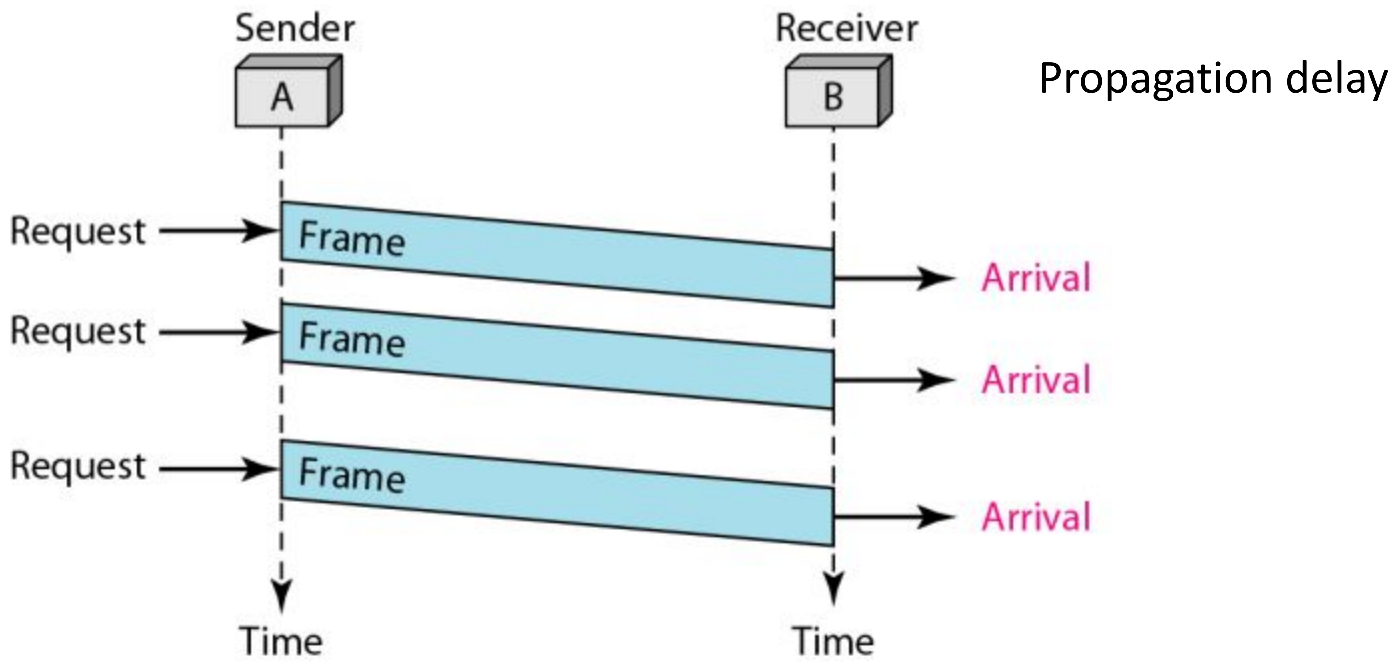


# Implementation

```
while(true){
    WaitForEvent();
    if(Event(RequestToSend)){
        GetData();
        MakeFrame();
        SendFrame();
    }
}
```

```
while(true){
    WaitForEvent();
    if(Event(ArrivalNotification)){
        ReceiveFrame();
        ExtractData();
        DeliverData();
    }
}
```

# Example

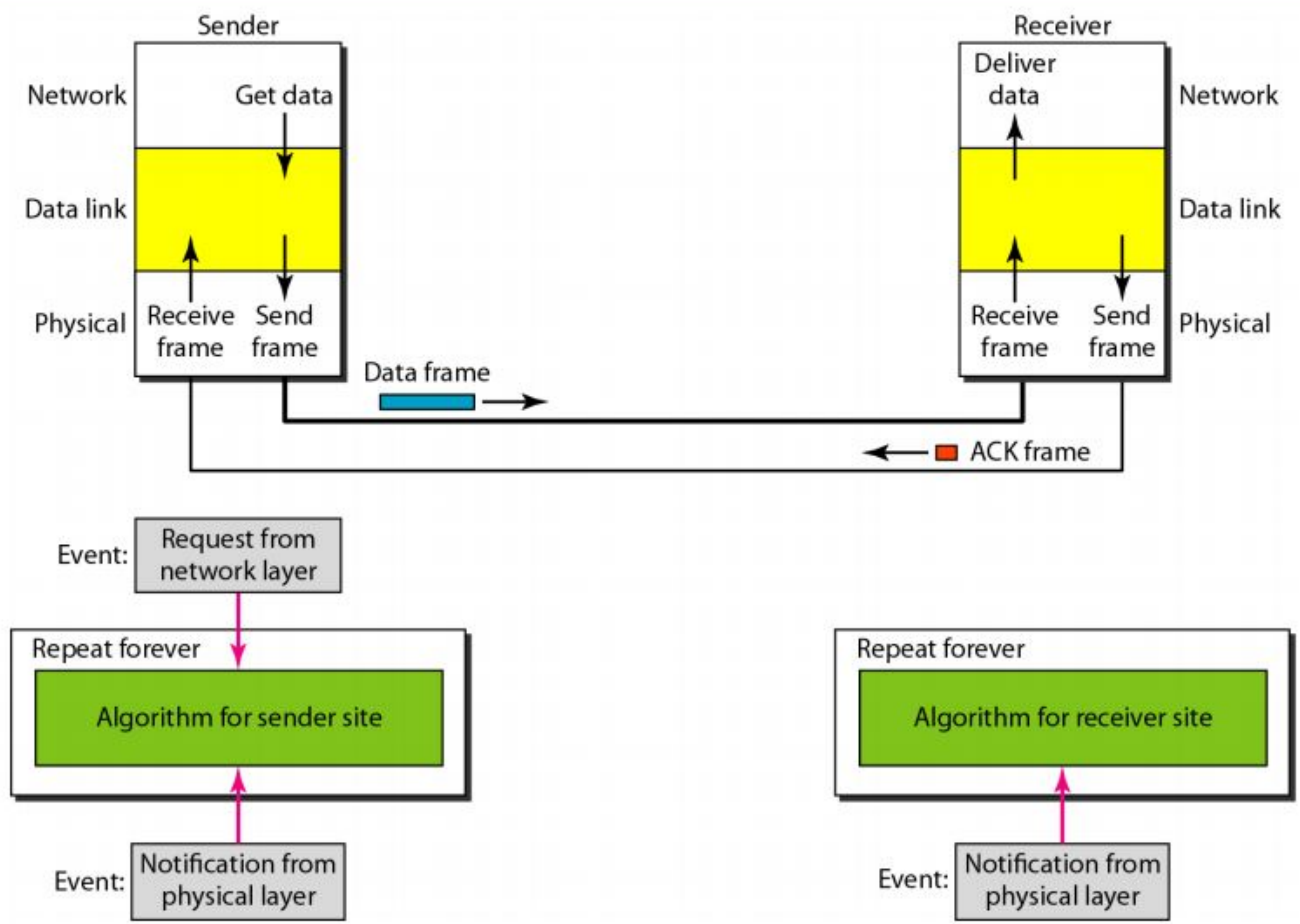



- The sender **sends a sequence of frames** without even thinking about the receiver
- There is no error handler
- There is no synchronization (the receiver processing time is slower than the transmission speed)

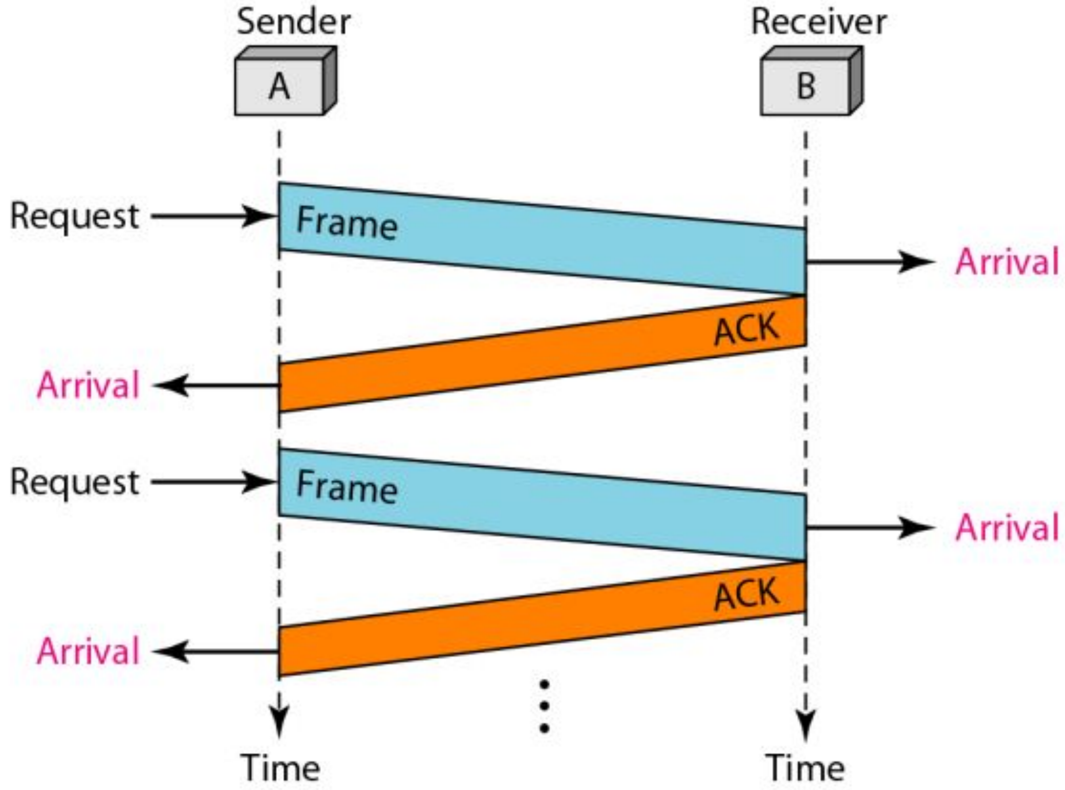
# Stop and Wait Protocol

- If data frames arrive at the receiver site faster than they can be processed, **the frames must be stored** until their use
- Normally, the **receiver does not have enough storage space**, especially if it is receiving data from many sources
- The sender sends one frame, stops until it receives agreement the receiver (okay to go ahead), and then sends the next frame
- **ACK frames** (simple tokens of acknowledgment) travel from the other direction

# Stop and Wait Protocol Design



# Example




- The sender sends one frame and waits for feedback from the receiver before sending the next frame
- Four events at the Sender and two events at the Receiver

# NOISY CHANNELS

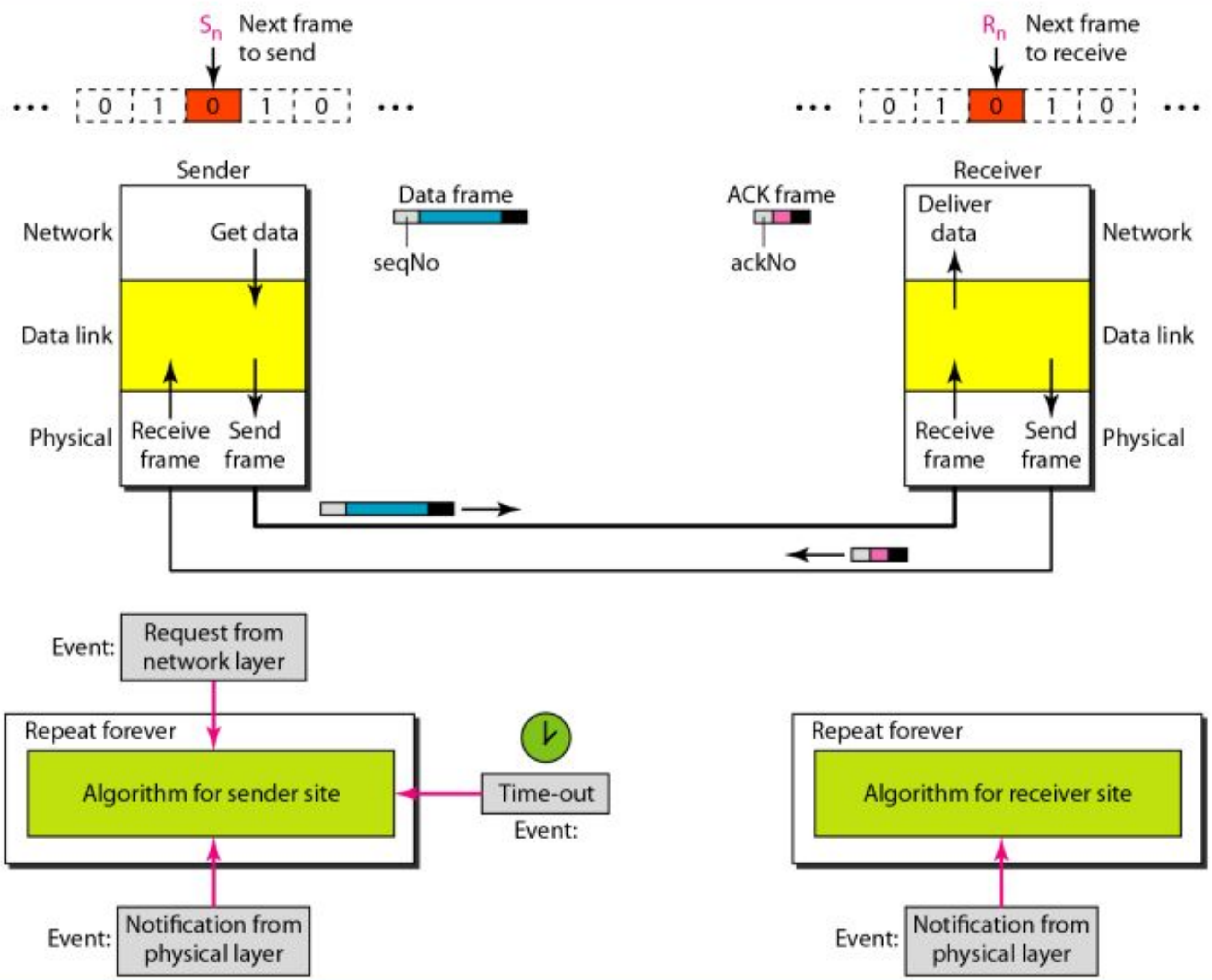
- Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, **noiseless channels are nonexistent.**

- Stop-and-Wait Automatic Repeat Request(ARQ)
- Go-Back-N Automatic Repeat Request
- Selective Repeat Automatic Repeat Request

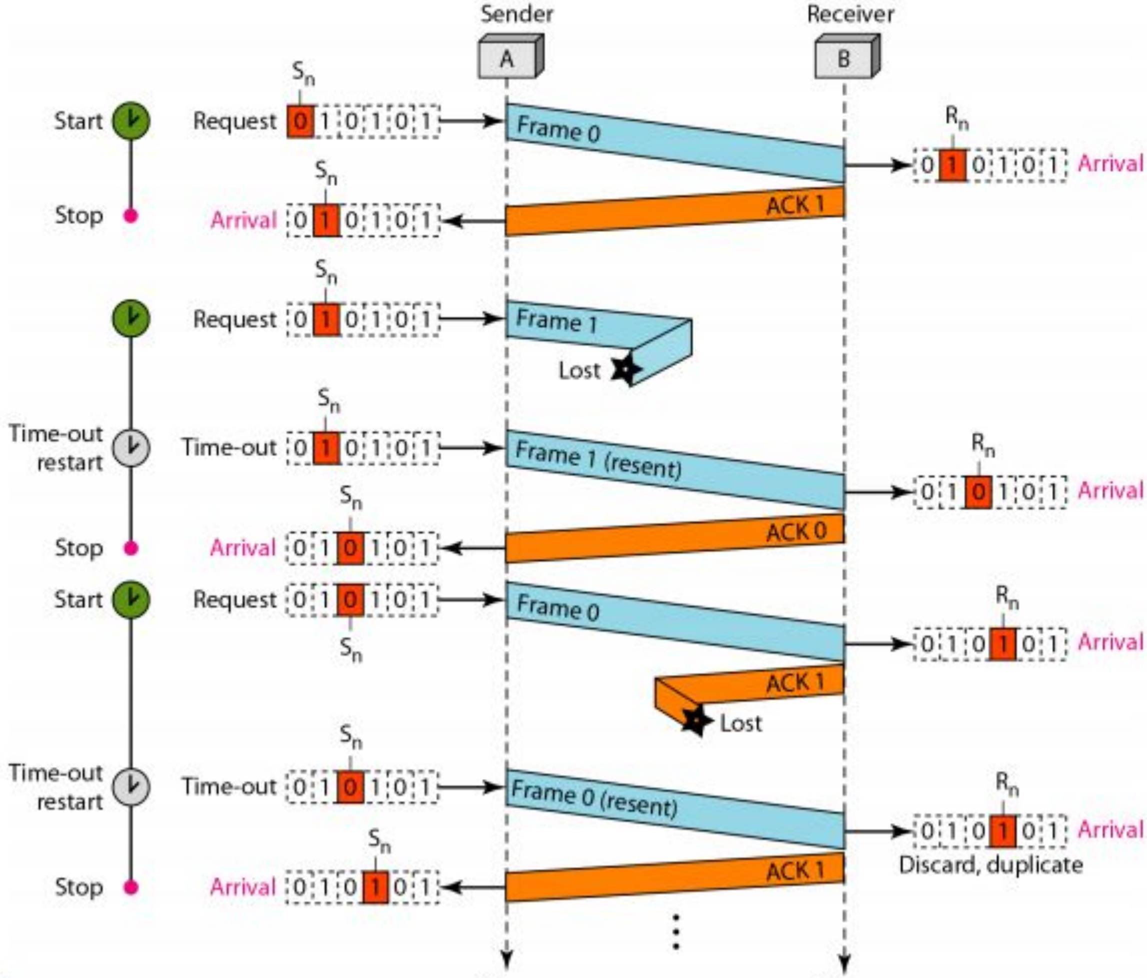
# Stop and Wait ARQ Principles

- A copy of a frame (sent to the receiver) is kept in the buffer
- Retransmitting this frame when the timer expires, meaning that ACK is not received
- **Sequence numbers** are used to index the frames.
- The acknowledgment number always announces the sequence number of the **next frame expected**

# Stop and Wait ARQ Protocol



# Example



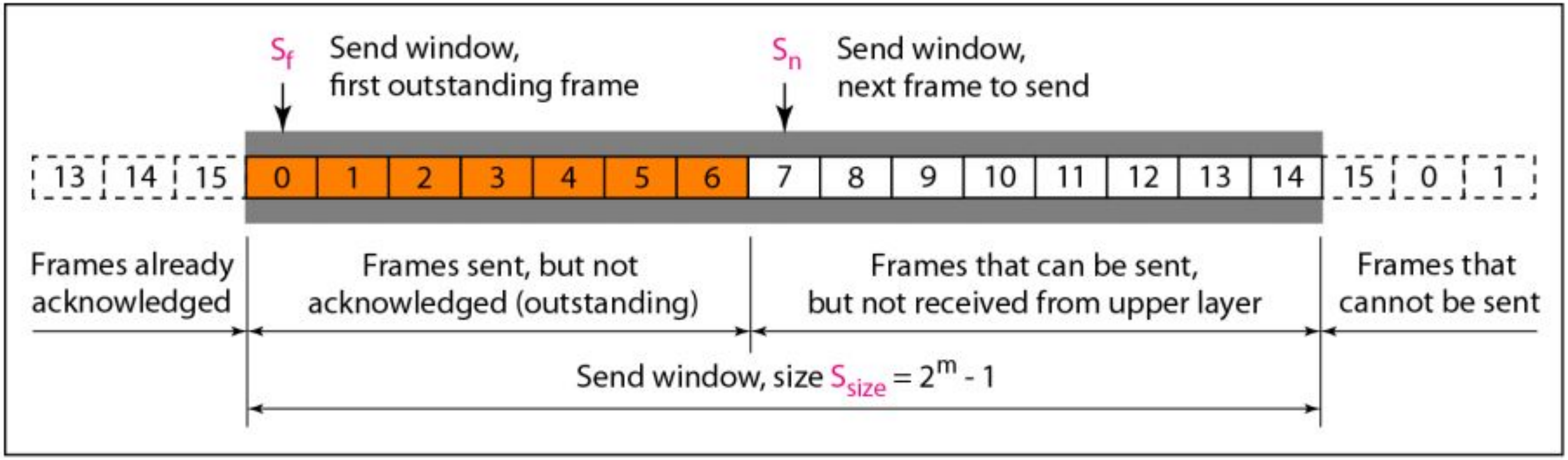

- **Frame is lost**
- **ACK is lost**
  - Duplicate reception at the receiver

# Go-Back-N Automatic Repeat Request

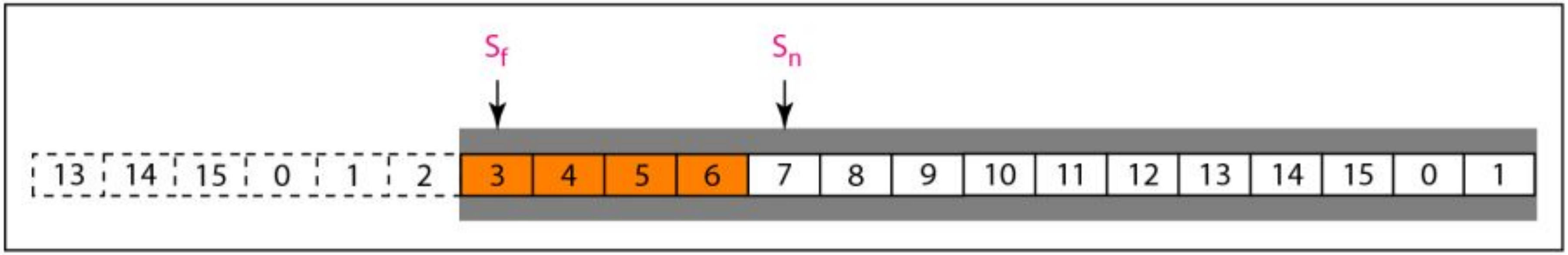
- **Multiple frames must be in transition** while waiting for acknowledgment to maximize the efficiency

- Protocol principles:
  - Several frames are sent before receiving ACKs
  - A copy of these frames are kept until the ACKs arrive

# Send Windows for Go Back N (m=4)




a. Send window before sliding



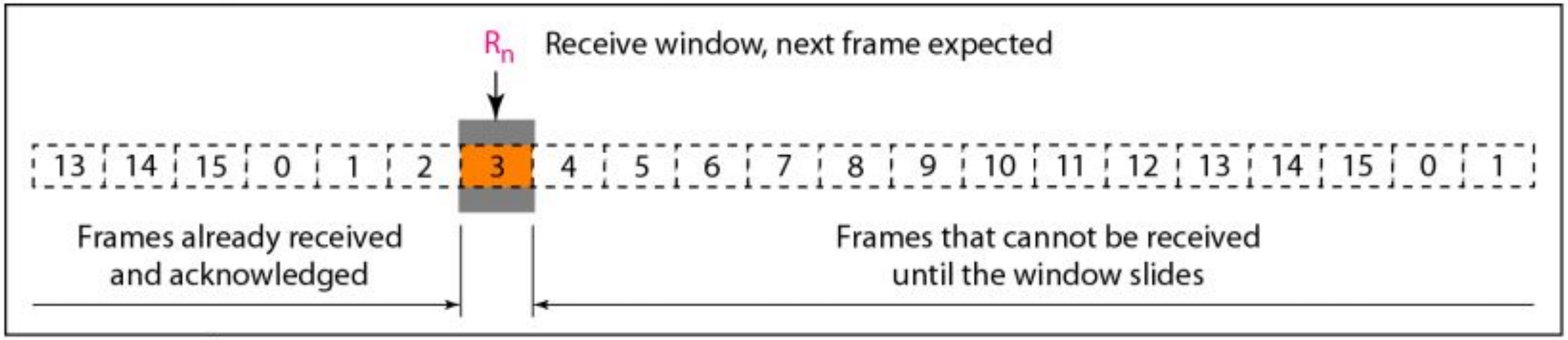

b. Send window after sliding

# Definitions

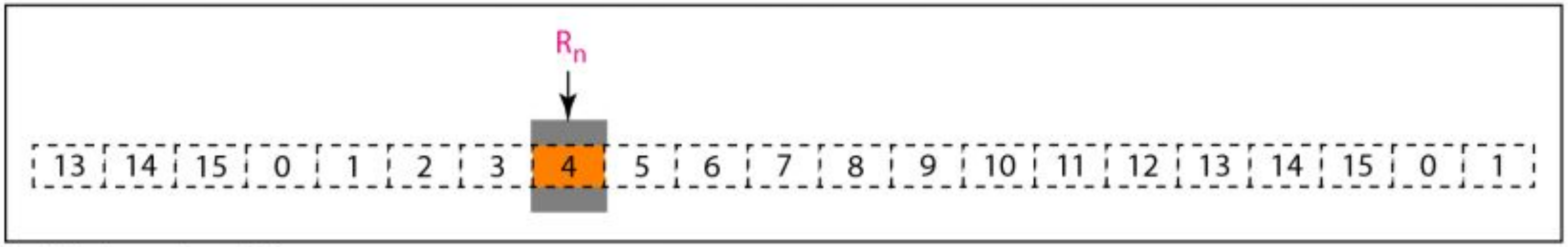
- $S_f$: the sequence number of the first (oldest) outstanding frame
- $S_n$: the sequence number that will be assigned to the next frame to be sent.
- $S_{size}$: the size of the window, which is fixed in our protocol.

- $S_f$ can **slide one or more slots** when a valid ACK arrives.

# Receive Windows for Go Back N



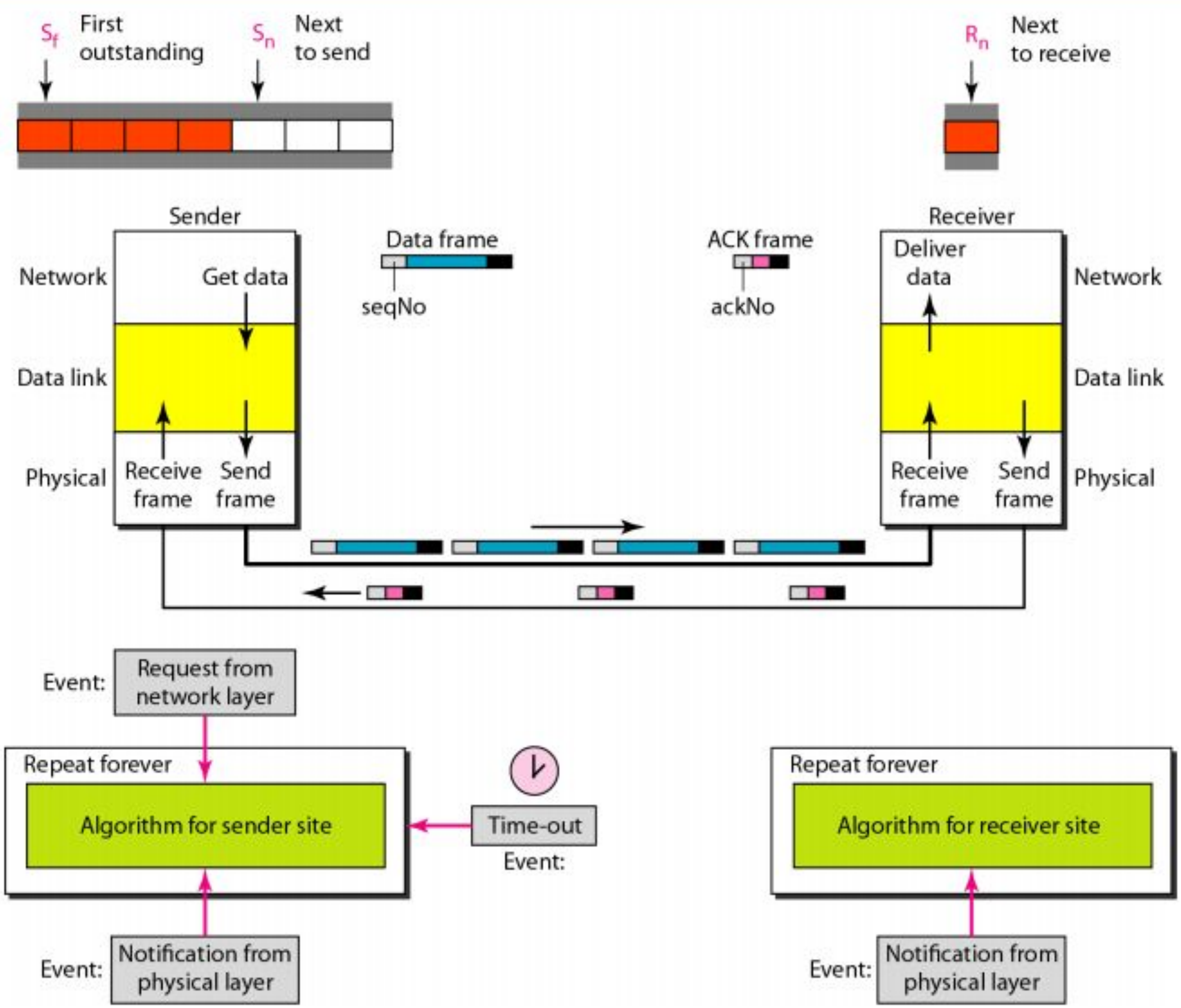

a. Receive window




b. Window after sliding

- The window **slides one slot** when a correct frame has arrived;

# Go Back N Design



# Selective Repeat Automatic Repeat Request

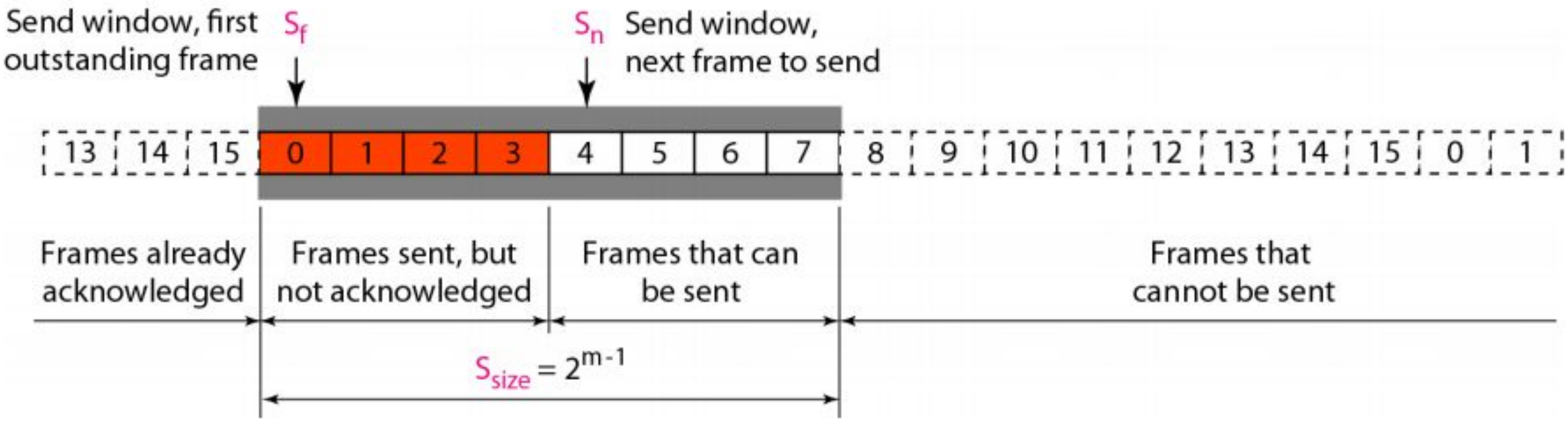
- In a noisy link a frame has a higher probability of damage, which means the **resending of multiple frames.** This resending uses up the bandwidth and slows down the transmission
- Selective Repeat ARQ: **does not resend N frames** when just **one frame is damaged**
- It is **more efficient** for noisy links, but the processing at the receiver is **more complex** compared to Go Back N

# Selective Repeat ARQ

- **Two windows are used**: a send window and a receive window as Go Back N
- The size of the sending window is smaller: $2^{m-1}$
- The received window is the same size as the send window (in Go Back N, the size is only **1**).

- For example, if m = 4, the sequence numbers go from 0 to 15, but the size of the window is just 8 (it is 15 in the Go-Back-N Protocol). The smaller window size means **less efficiency in transmission**, but the fact that there are fewer duplicate frames.

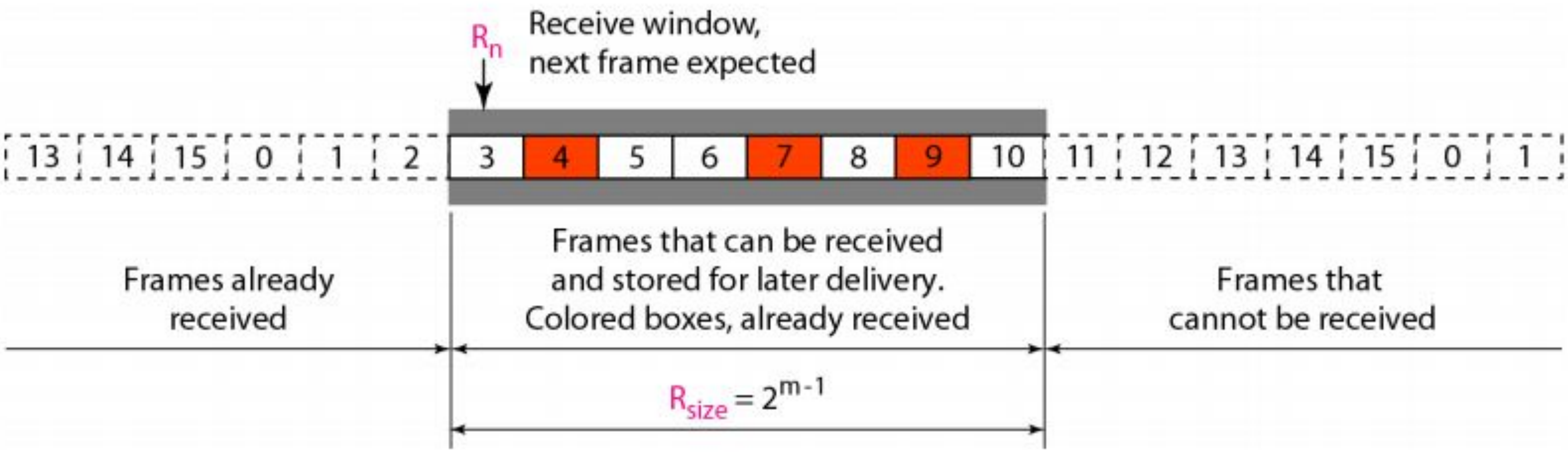
# Sending Windows (m=4)



# Received Windows (size $2^{m-1}$)

- Many frames can be arrived **out of order** and be kept until there is **a set of in-order frames** to be delivered to the network layer

- All the frames in the send frame can arrive out of order and be stored until they can be delivered.



# Design of Selective Repeat ARQ