

Basic Protocols and Error Control Mechanisms

Nicola Dragoni
Embedded Systems Engineering
DTU Compute

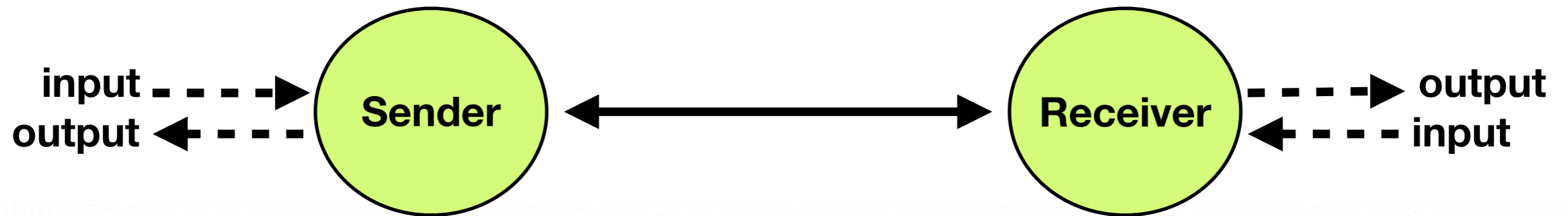
- ACK/NACK Protocol
- Polling Protocol
- PAR Protocol
- Exchange of State Information
 - ▶ Two-Way Handshake Protocol
 - ▶ Three-Way Handshake Protocol



Error
Control
Mechanisms



Simple ACK/NACK Protocol



- M = domain of correct messages
- M' = domain of corrupted messages
- $M \cap M' = \{\}$
- “checksum” model: $x \in M$ or $x \in M'$

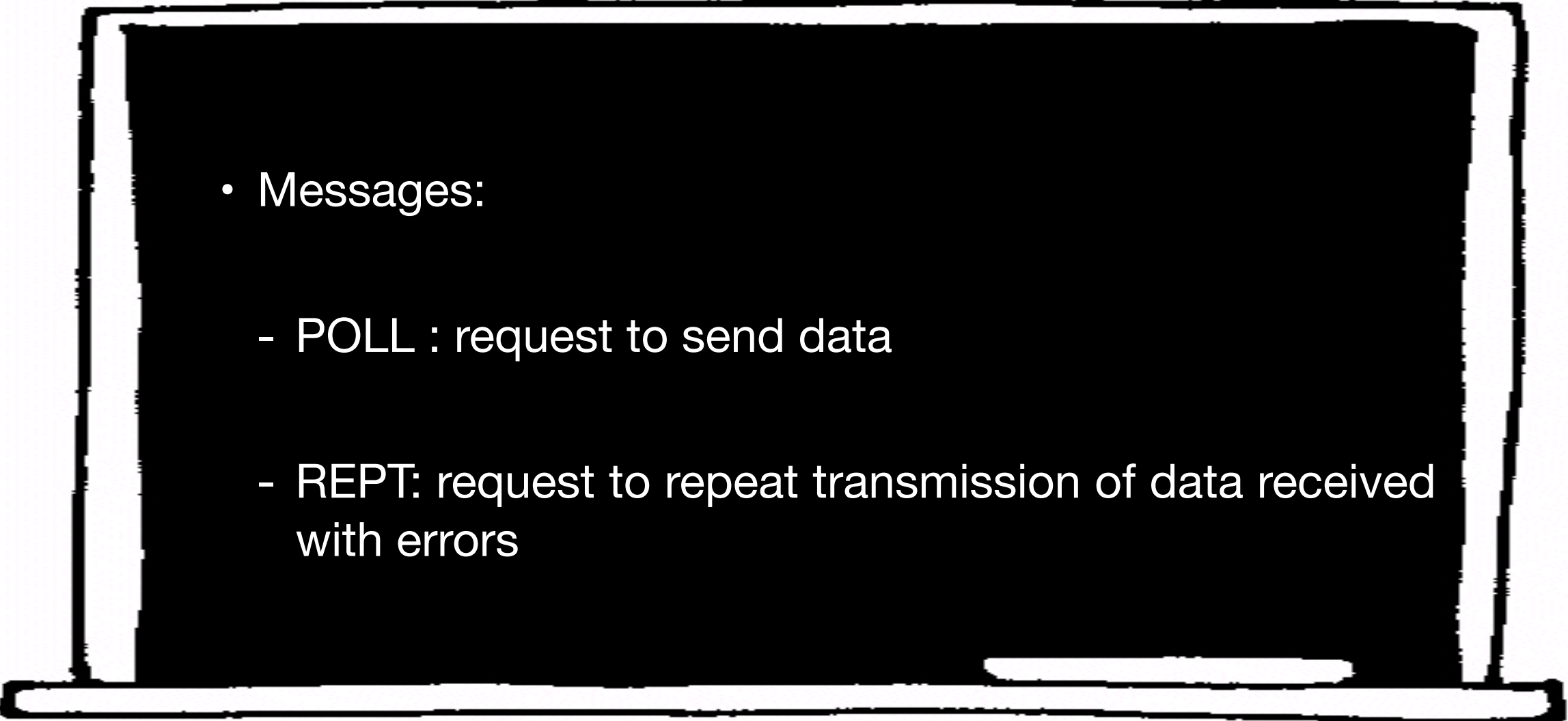
Polling

- In the previous simple ACK/NACK protocol:
 - ▶ it is **the sender** who **takes the initiative** for sending a message
 - ▶ the **receiver** merely **responds to this**.
- Effectively, this obliges the receiver to be able to receive data at any time after it has send an acknowledgment

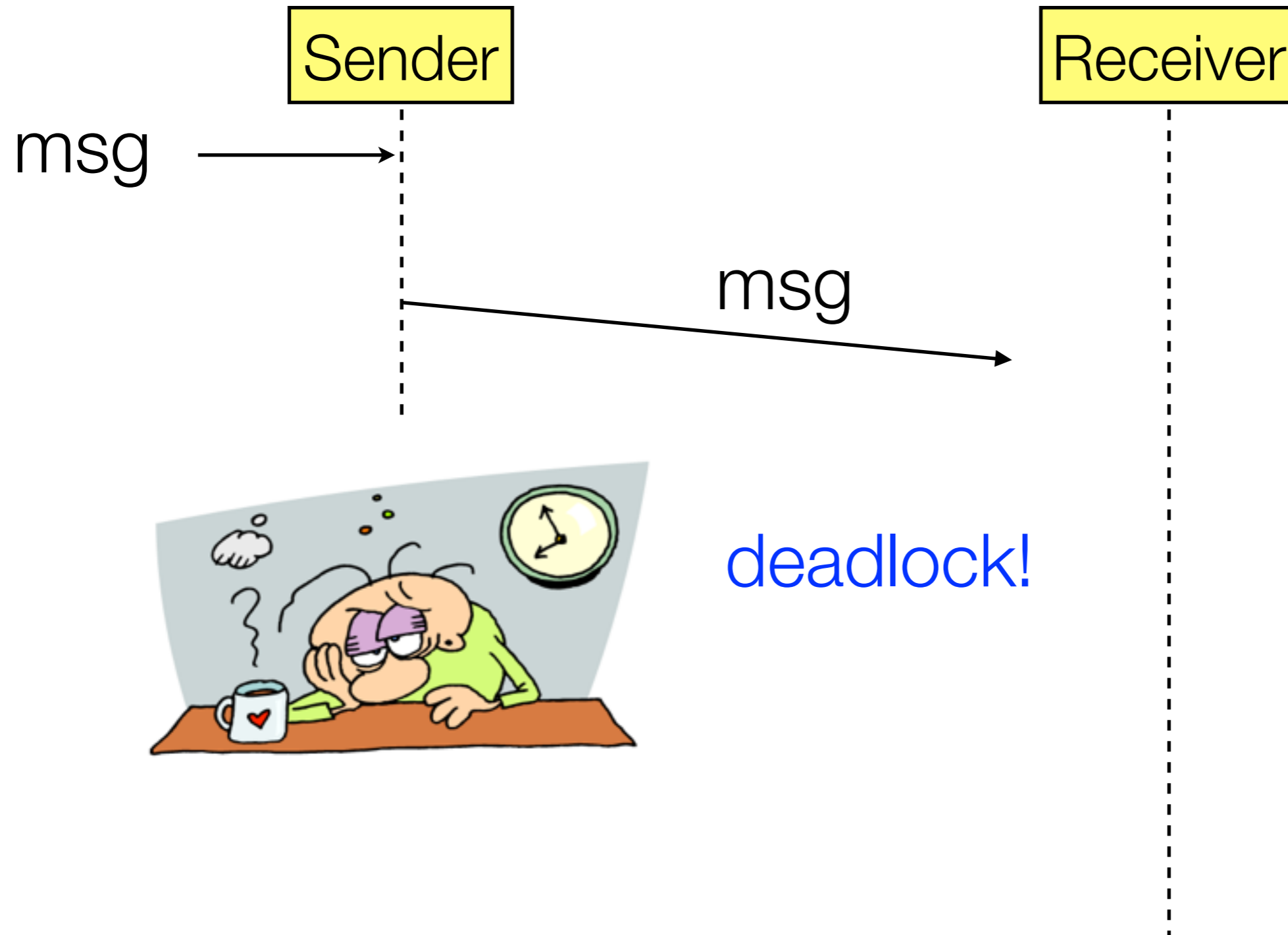
- Alternative strategy (**POLLING**):
 - ▶ the **receiver** explicitly **takes the initiative**, requesting data when it is able to receive them

Simple Polling Protocol

- Receiver has initiative!

- 
- Messages:
 - POLL : request to send data
 - REPT: request to repeat transmission of data received with errors

ACK/NACK Problem





ACK/NACK + TIMEOUT

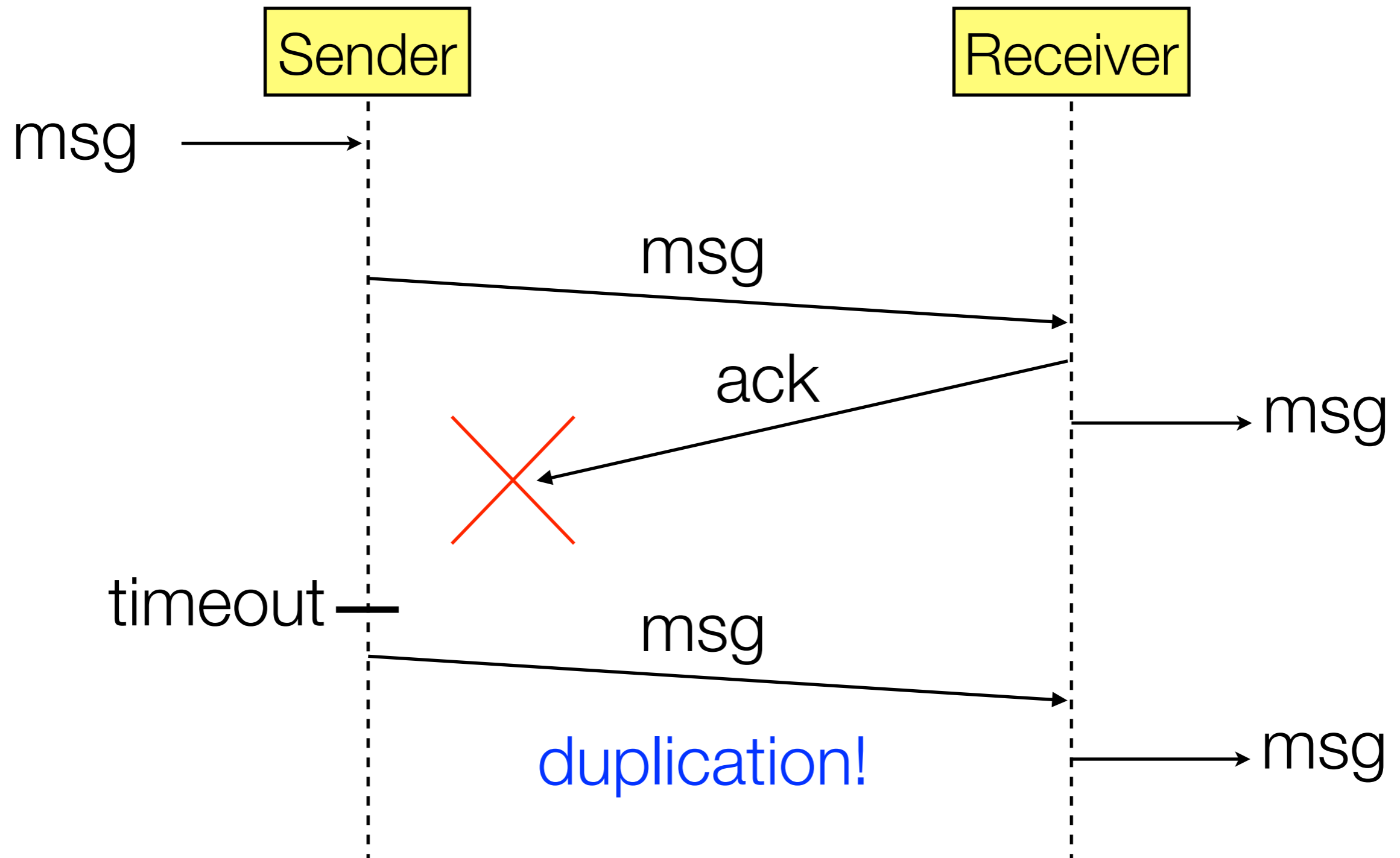
- Deadlock caused by **loss of the acknowledgment message**
- Corrected by retransmission after a *certain time* with no acknowledgment

- 
- ACK/NACK protocol with TIMEOUT

ACK/NACK + TIMEOUT - Duplication Problem

- Consider the following situation:
 - ▶ the **receiver** receives a correct message via its channel left and then **sends a positive acknowledgment**
 - ▶ this acknowledgment message **gets lost**
 - ▶ the **sender** will eventually time out, and retransmit the **same message** to the receiver
 - ▶ **so the receiver receives the message twice and passes it on to the user (output) twice**

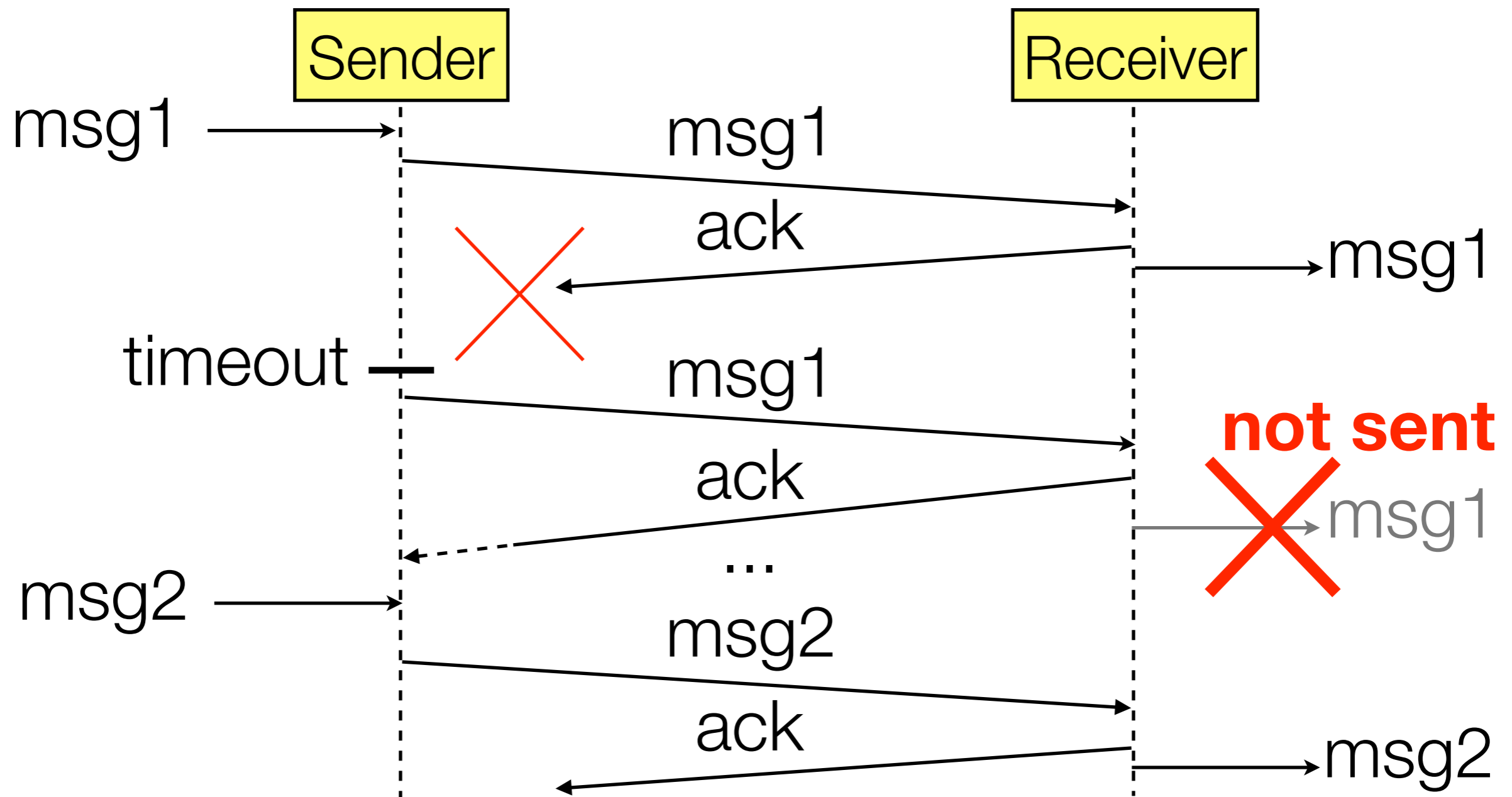
ACK/NACK + TIMEOUT - Duplication Problem





Possible Solution: Numbering Scheme

- Introducing a **numbering scheme for the messages**: duplicated messages can be filtered off by the receiver before messages are passed to the user.



HOMEWORK



Exercise

- Write a specification of an ACK/NACK protocol able to handle the following failures:

1. deadlock caused by the loss of the acknowledgment message



2. duplication of messages sent to the user



PAR Protocols

- We can also **remove the NACK type of acknowledgment**. Why?
 - ▶ When a timeout mechanism is used, **negative acknowledgments only have an effect on the response time of the protocol**, since they can be used to provoke retransmission before the timeout period runs out.
 - ▶ *Negative acknowledgments do not affect the logical properties of the protocol in any way.*

- Protocols with:

- ▶ **only positive acknowledgments + ACK**
- ▶ **using a timeout mechanism to control retransmission**



are often called **Positive Acknowledge and Retransmission (PAR)** protocols

PAR Protocol (ACK + TIMEOUT + NUMBERING SCHEME)

- PAR Protocol with Timeout and Sequence Numbers

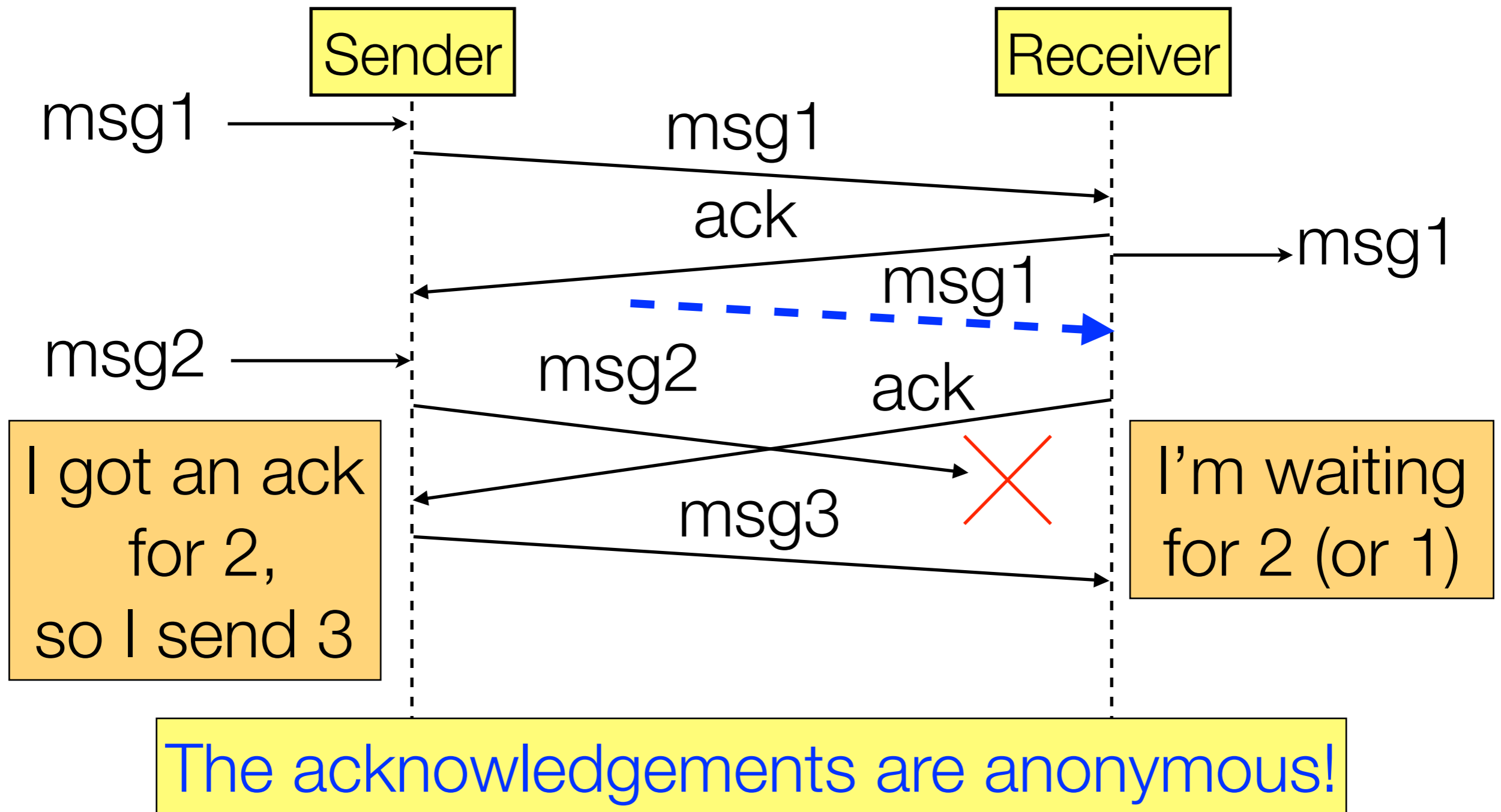
HOMEWORK



Exercise: Polling Protocol

1. Extend the polling protocol with sequence numbers and timeout.
2. Analyse your proposal to see which problems (if any) the protocol might still have.

PAR Protocol - Problem



Problem: Anonymous ACK

- **Anonymous Acknowledgement Problem:** all protocols we have seen so far rely on *anonymous messages*
 - ▶ **ACK** messages:
 - just tell the sender that the other party has received the data which came in the right order
 - the sender has no means of knowing exactly which data is referred to

This reflects a **general problem in distributed systems**: the cooperating parties do not in general know what their collective **global state** is.

- ▶ Parties have to make decisions on the basis of
 - whatever **information they locally have available** or
 - the **information their cooperators have sent them**

Solution: Sequence Numbers in ACKs

- We include an **identification on the acknowledgments**, indicating the **sequence number of the latest correctly received data**
- **Sender:**
 - ▶ repeats message with number n until it receives an acknowledgment explicitly denoting n
- **Receiver:**
 - ▶ replies to each correct incoming data with an **acknowledgment** that **includes the sequence number of the *last correctly received message*** (which of course may be the message just received or a previous one)

Example: PAR Protocol + NUMBERED ACK

- The ack message now consists of the **NUMBER OF THE LATEST CORRECTLY RECEIVED** data message

- 
- PAR Protocol with Numbered ACK



Sequence Numbers?

- **Simple idea:** Sequence numbers are successive natural numbers 0, 1, 2, 3, ...
- **Problem:** Only a finite number can be represented in a real message.
- **New idea:** If acknowledgment is received within relatively short time, it is only necessary to count **modulo** some small value S_{mod} , so

$$succ(n) \stackrel{\text{def}}{=} (n + 1) \bmod S_{mod}$$

- **Example** [PAR protocol with numbered ACK]: Sender always waits for positive ACK for latest transmitted message before using next sequence number. OK to count modulo 2 (“**Alternating Bit Protocol**”)
- If more messages can be outstanding (sent but not acknowledged), S_{mod} must be larger

ESSENTIAL RULE: messages with number n must be guaranteed to be “dead” before n is re-used

PAR Protocol + NUMBERED ACK

- Protocol now gives both parties sufficient knowledge of what is happening, so it protects against
 - ▶ loss
 - ▶ duplication
 - ▶ corruption
- of both data messages and ack messages

**But it can still fail.
How?**

