

# Computer Labs: Communication Protocols

## 2º MIEIC

Pedro F. Souto (`pfs@fe.up.pt`)

May 3, 2024

# Overview

## Transforming a Single-player Game into a Multi-player Game

### Receiving Input from the Remote Player

- Messages

- Communication Errors

- Synchronization Issues

### Sending (or not) the Output to the Remote Player

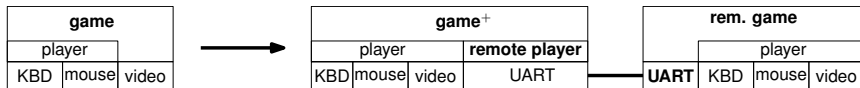
# Problem

**Assumption** So, you are happy with the single-player game you have developed and want to implement a multiplayer version, using the serial port

**Problem** How to go about this?

**Solution** There are several approaches, but the simplest one is probably to use a centralized solution where:

- ▶ All the game's logic is implemented on the first computer
- ▶ The second computer just:
  - ▶ Gets the input from the 2nd player and forwards it, via the serial port, to the first computer
  - ▶ Receives the "frames", via the serial port, and displays them on the screen



## First computer

- ▶ Implements the game's logic;
  - ▶ Like in the single player version
- ▶ Interfaces with the 1st player;
  - ▶ Like in the single player version
- ▶ Communicates with the second computer to:
  - ▶ Receive the input from the 2nd player
  - ▶ Send the "frames" to display to the 2nd player

For uniformization purposes, you can hide this behind the abstraction of a **remote player**

## Second computer

- ▶ Only interfaces with the 2nd player:
  - ▶ To get the input from the keyboard and the mouse
    - ▶ And send it to the first computer
  - ▶ To display the frames
    - ▶ That it receives from the first computer

# Overview

Transforming a Single-player Game into a Multi-player Game

Receiving Input from the Remote Player

- Messages

- Communication Errors

- Synchronization Issues

Sending (or not) the Output to the Remote Player

# Getting the Input from the Remote Player

**Assumption** The game uses both the keyboard and the mouse as input

**Problem** How to send that input to the main computer?

**Solution** Use the serial port to send

Scan codes

Mouse packets

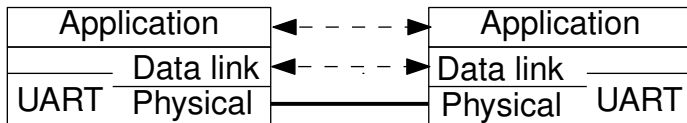
**Issue** The serial port allows to send only characters

**Solution** Define multi-character messages

**Scan code messages** with the scan codes

**Mouse messages** with the mouse packets

- I.e. build an application protocol on top of the communications protocol provided by the UART



# Messages

**Issue 1** How to distinguish between the two types of messages?

**Solution** Use a **message header** with one field specifying the type of message. E.g., the first character of each message, indicates the type of message:

'K'	<i>KBD scan code</i>
'M'	<i>Mouse packet</i>

Although a bit would be enough, it is not a good idea:

- ▶ Inflexible: what if you want to add another message?
- ▶ Hard to debug.

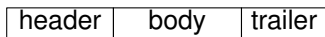
**Issue 2** Scan codes do not have fixed length

**Solution** After receiving a byte, we always know whether this is the last one. Other solutions are:

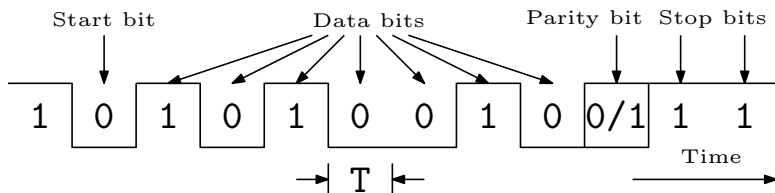
1. Include the length of the message in the message header
2. Add some special bit-sequence/char in the **message trailer**
  - ▶ It **must not** occur in either the message head or the **message body**

# Message Structure

- Independently of the layer, the general message structure is:



- This is true even at the data link layer:





# Communication Errors

**Problem** What if some characters of a message are lost?

**Solution**

**Just drop message** this may make sense, if message loss is not that frequent

**Use acknowledgments** and retransmit

- ▶ Similar to the protocol used between the KBC and the keyboard (**ACK(0xFA)**, **NACK/RESEND(0xFE)**, **ERROR(0xFC)**)
- ▶ When to send the acknowledgment?
- ▶ In any case, need to detect the loss of a character (the UART usually does it for us)
  - ▶ But, what if the problem is with the remote process?
- ▶ May need to know also when no more chars of the damaged message will be received
  - ▶ This is a synchronization issue

# Synchronization Issues

**Problem** Communication errors or the initial asynchrony between the communicating processes may lead to synchronization issues

**Solution** Similar to that used by Synaptics in the touchpad: any error reported leads to retransmission of the entire message.

- ▶ Use of special bit-patterns that do not occur in the **body** of a message helps resynchronization.
- ▶ Unfortunately, this is not the case with the bytes of a mouse packet
  - ▶ In the case of scan codes, some byte values may not be used

**Question** How can we ensure synchronization, without the limitations of the mouse interface?

# More Synchronization Issues

**Assumption** You may have some game logic on both processes, and both of them need to keep the program state synchronized

**Problem** How do both processes synchronize initially

**Solution** Two approaches

**Asymmetric** One of the processes takes a passive stance and the other a proactive stance

- ▶ The former waits until the latter gets in touch with it
- ▶ What if the passive process is not up when the proactive tries to contact it?

**Symmetric** Either process can take a proactive stance

- ▶ When a process starts up it behaves proactively, but if the other process is not up it switches to passive behavior.
- ▶ This has also a few problems:
  - ▶ What if the other process is up, but the message is lost?
  - ▶ What if the two processes wake up at the same time?

# Overview

Transforming a Single-player Game into a Multi-player Game

Receiving Input from the Remote Player

Messages

Communication Errors

Synchronization Issues

Sending (or not) the Output to the Remote Player

# Sending the Output to the Remote Player

**Problem** Cannot send the entire frame buffer over the serial port

- ▶ The serial port has a very limited bandwidth

**Solution 1 "Compress"** the frame buffer data

- ▶ Video encoding algorithms actually apply compression algorithms, but
- ▶ We mean some "ad-hoc" compression. E.g.
  - ▶ Send the new state of sprites whose state changes from one frame to the next

**Solution 2 Use state-machine replication.** The idea is:

- ▶ Each processes keeps the entire game state
- ▶ Each process generates the frames to be presented locally
- ▶ Each process processes the inputs from both
  - ▶ the local player
  - ▶ the remote player

**in the same order:**

**This may not be trivial**