Systems programming

2021/2022

Project Assignment – Part A

PongPong

In the first part of the project, students will implement two different versions of the pong game. In these games, users use a paddle to hit the ball that moves on the screen. When a ball touches a paddle or the edge of the screen, it bounces and changes direction.

The two versions of Pong to be implemented are:

- the Relay-Pong - In the first version of the Pong, each user takes turns controlling the paddle and hitting a ball. Only one user controls the paddle and after some time the user may release the ball, and the control passes to other user.
- the Super-Pong - In the second version, users control the paddles simultaneously, on the same board, and all are trying to hit the same ball.

In all the versions of Pong, the users run a client that:

- connects to a server using Internet domain datagram sockets
- allows the control of the user paddle
- shows the ball position

The servers allows the transfer of data between the clients.

1 Relay-Pong

In the Relay-Pong, users take turns playing the pong game individually on their client, until they release the ball and the control goes to other user. When a client is not controlling the ball, his paddle will not move, but the ball position (controlled by another client) is updated on the screen. When a client receives the ball, the user can start to move the paddle and hit the ball.

1.1 Clients / Server interaction

The essential interaction between the clients and the server is presented in Figure 1. The server is waiting for messages from the clients and, depending on the received message, changes its state or communicates with the clients.

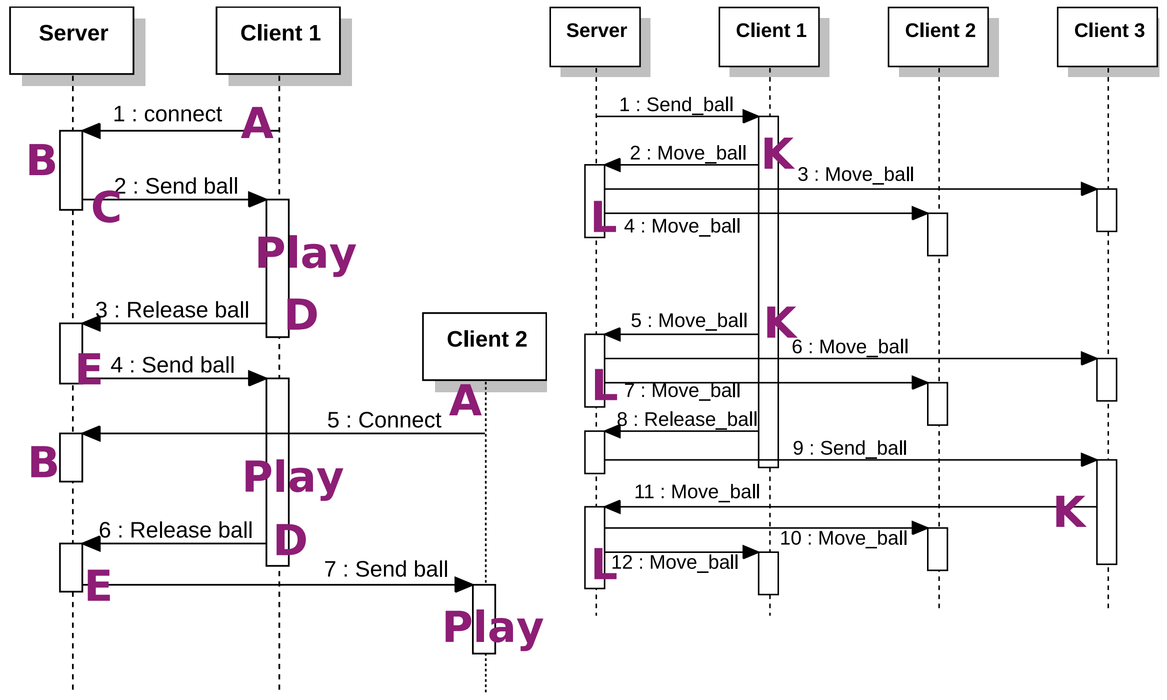


Figure 1: Client - Server Interactions

The messages exchanged between the server and client are:

- **Connect** (from client to the server)
- **Release_ball** (from the client to the server)
- **Send_ball** (from the server to the client)
- **Move_ball** (from client to server and from server to client)
- **Disconnect** (from client to server)

Every client needs to send a **Connect** message (**step A**) at startup. The server stores the relevant client information in a list (**step B**). If this client is the first one to connect, the server sends as reply a **Send_ball** message (**step C**). The user can now move the paddle and hit the ball (**Play state**). Any client that connects when other client is in **Play state** will wait for a **Send_ball** message.

After some in the **Play state**, the user can release the ball (**step D**) and stop controlling the paddle. The client sends the **Release_ball** message to the server. After the reception of the **Release_ball** message, the server chooses one of the available clients and sends the **Send_ball** message (**step E**). If the client that released the ball is the only one connected, it will receive the ball back; otherwise the server should select a different client to receive the ball.

After the client receives a **Send_ball** message, it goes into **Play state**, and the user can start controlling the paddle with the keyboard. The ball movements are only calculated on the client that is in the **Play state**. Every time the paddle moves, the client performs the following operations (**step K**):

- move the paddle;
- calculates and updates the new ball position,
- sends the new ball position to the server with the message **Move_ball.**

When the server receives the **Move_ball** message, it must forward the same ball movement to all other connected clients (**step L**), so that every client can update the corresponding screen to show the ball in the new position.

When a client terminates orderly, it should send a **Disconnect** message.

1.2 Relay-pong Server

The **Relay-pong server** is a C program with an Internet domain datagram socket to receive messages from the various clients that want to play Relay-pong.

Whenever the server receives one message, it should update its state or send suitable messages following the protocol and description presented in Section 1.1.

The sever should store a list of the clients' information to send them messages when necessary. A client is inserted into this list when the server receives a **Connect** message and removed when the server receives a **Disconnect** message from such client.

1.3 Relay-pong Client

The **Relay-pong client** is a C program that interacts with a Relay-pong server for the user to play the game. The address of the server should be supplied to the program as a command line argument.

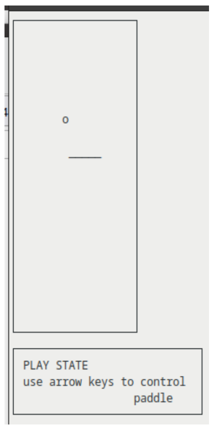
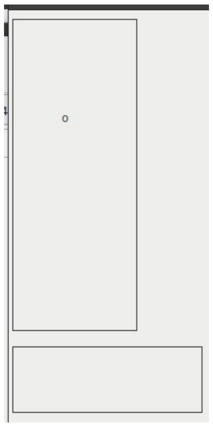
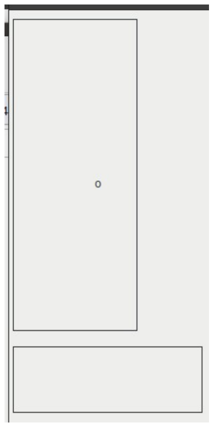
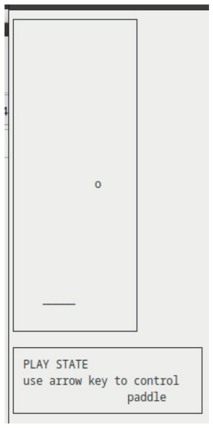
After sending the **Connect** message, the client will wait for a **Send_ball** message. Until this message is received the user can not control the paddle. When receiving this message the client goes into **Play state** where the paddle can move and hit the ball:

- Whenever the user presses one of the arrow keys of the keyboard, the paddle position on the screen is updated.
- After updating the paddle the ball is moved and its position updated.
- If the paddle hit the ball (or vice-versa) the ball movement should be updated.
- After updating the ball position, the client should send a **Move_ball** message to the server.

Ten seconds after the reception of the **Send_ball** message, a user can release the ball by pressing the **r** key. The client will send a **Release_ball** message to the server, and the user will stop controlling the paddle.

During **Play state**, if the user presses the **q** key, the client should terminate and send a **Disconnect** message to the server.

The client should print on the screen a message informing the user whether the client is in **Play state** and that paddle can be controlled.

 PLAY STATE use arrow keys to control paddle **Player 1** In **Play State** Controls paddle Controls ball Sends **Move_ball**	 **Player 2** Not in Play State Waiting for a **Send_ball** message Receives **Move_ball** and updates ball position
Player 1 presses the **r** key to release ball	
 **Player 1** Not in **Play State** Waiting for a **Send_ball** message Receives Move_ball and updates ball position	 PLAY STATE use arrow key to control paddle **Player 2** In **Play State** Controls paddle Controls ball Sends **Move_ball**

2 Super-Pong

In Super-pong, all players (connected to the server in multiple clients) can move their paddles and try to hit the ball simultaneous. This is accomplished by reading the paddle movement on each client, and sending such movements to the server, which updates the ball and sends back to all the clients the new board state.

The clients send every paddle movement to the server and receive as reply the positions of the ball and all the other paddles.

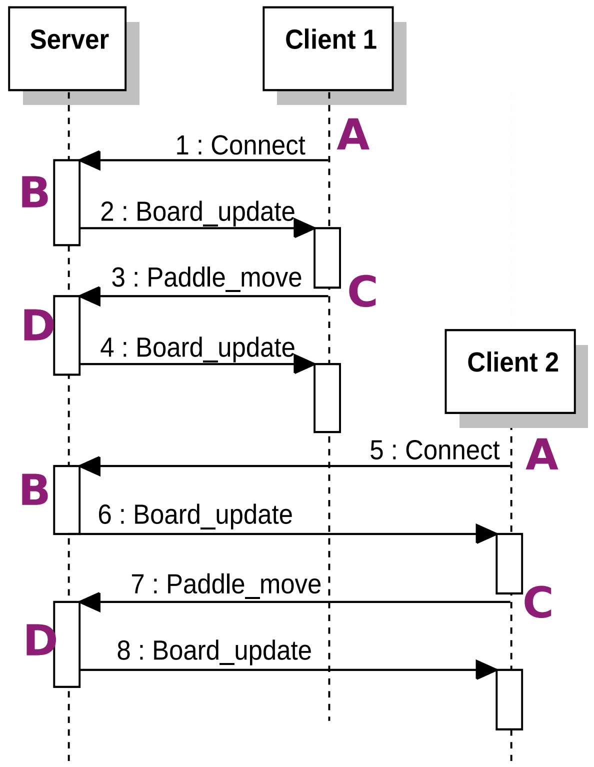


Figure 2: Super-Pong Client-server interaction

2.1 Clients / Server interaction

The essential interaction between the clients and the server is presented in Figure 2. The server is waiting for connect and paddle movement messages from the clients, after which sends the new state of the board to the client.

The messages exchanged in Super-Pong are:

- **Connect** – from client to server
- **Paddle_move** – from client to server
- **Board_update** – from server to client
- **Disconnect** – From client to server

Every client starts by sending a C**onnect** message to the Server (**step A**). The server inserts the relevant data into the clients' list, defines a random position for the paddle of this client, and replies with a **Board_update** message (**step B**).

The **Board_update** message should include:

- the position of the ball
- the position of this client paddle
- the position of other clients paddles
- players scores

Whenever the user presses one of the arrow keys to move the paddle, the client sends a **Paddle_move** message (**step C**) to the server, and as a reply the server sends back the **Board_update** message (**step D**).
The **Board_update** message is only send as a reply to the **Connect** and **Paddle_move** message. A client that does not move its paddle will not see the updates on the ball and other clients' paddles.

When a client terminates orderly, it should send a **Disconnect** message.

2.2 Super-pong Server

The **Super-pong server** is a C program with an Internet domain datagram socket to receive messages from the various clients that want to play Super-pong.

The maximum number of simultaneous players is ten (10). Students should decide what happens to a client that send a **Connect** message when 10 clients are already connected.

Whenever the server receives one message, it should update the state of the game and send suitable messages following the protocol and description presented in Section 2.1.

The server should store a list of all the clients with all the relevant information (e.g. paddle position). A client is inserted into this list when the server receives a **Connect** message and removed when the server receives a **Disconnect** message from such client.

When the server receives a **Paddle_move** message, it should calculate and update the ball and paddle positions.

2.2.1 Ball movement

The ball movement should be calculated on the server after receiving the **Paddle_move** messages from the clients.

The ball should move (following the code in the provided skeleton) after every **n** **Paddle_move** messages received from the clients (where **n** is the number of connected clients).

The ball should bounce on the wall and when touching the paddles. Students can use and modify the code provided. In order to calculate the bounces on the walls, the server needs information about the board size.

2.2.2 Paddles and scores

Paddles should not share the same space. When a user tries to move a paddle to a position already occupied by another paddle, the moving paddle should not move.

If a ball touches a paddle, the corresponding user will get a point added to the score.

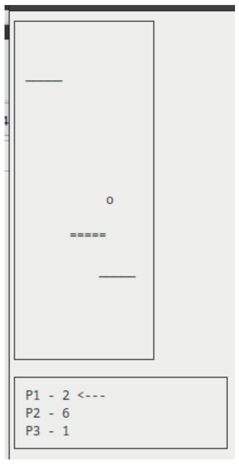
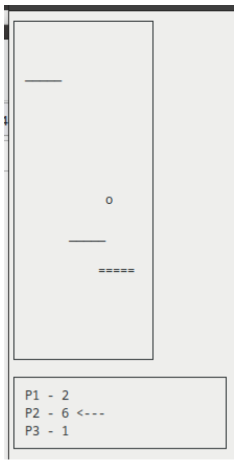
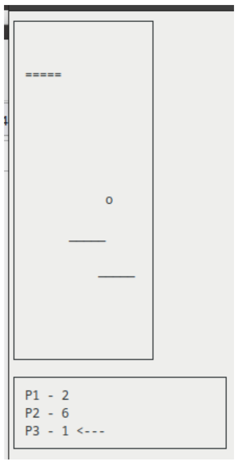
2.3 Super-pong Client

The client should contact a Super-pong server for the user to play. The address of the server should be supplied to the client as a command line argument.

After sending the **Connect** message, the client will start reading the keyboard. If the user presses one of the arrow keys, a **Paddle_move** should be sent to the server.

After receiving the **Board_update**, the client should draw the ball and all the paddles. The client paddle can be drawn using the **=** character, to distinguish it from the other paddles (that can be drawn with _).

The various players’ points should be written on the screen and updated every time a **Board_update** message is received.

Player 1	Player 2	Player 3
 ===== o P1 - 2 <--- P2 - 6 P3 - 1	 o ===== P1 - 2 P2 - 6 <--- P3 - 1	 ===== o P1 - 2 P2 - 6 P3 - 1 <---
Each player controls his paddle (marked with =). All screens are synchronized: • When a paddle move it is updated on all other clients. • The ball position is updated on all clients.		

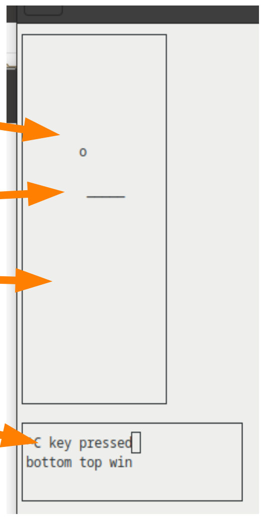
3 Client user interface

The client will use **ncurses** to read the keyboard and draw the paddles and ball on the screen.

Students were provided with the sample code of a client skeleton. This provided code implements functions and data structures to:

- ball information storage

- ball initialization, draw, and movement (taking into consideration the bounces on the walls)
- paddle information storage
- paddle initialization, draw, and movement
- window to draw the board
- reading of the keyboard arrow keys
- window to print warnings and messages



```
o
```

```
C key pressed
bottom top win
```

Since the keyboard reading is blocking (waits for the user to press a key and the program cannot do any other thing) and threads were not taught, all extra processing and communication should be done on the same loop as the keyboard reading (reception of **Board_update** message after the movement of the paddle and ball in Super-Pong) or in a loop without keyboard reading (reception of **Move_ball** and **Send_ball** messages in Relay-Pong).

The size of the board and the paddles can be defined as constants in a **.h** file to be included by the clients and server (as in the provided skeleton).

4 Project submission

Students should implement the two different Pong games, each composed of a client and server.

The deadline for the submission for the part A of the project will be 14th January at 19h00 on FENIX.

Before submission, students should create the project groups and register them at FENIX.

Students should submit a zip file containing the code for both games. The files for each game should be placed on two different directories, and if possible students should also provide a Makefile for the compilation of the various programs.

5 Project evaluation

The grade for this project will be given taking into consideration the following:

- Number of functionalities implemented
- Communication
- Code structure and organization
- Error validation and treatment
- Comments