Lab Session 2.2: TCP/UDP

Blocking Operations

Sending or receiving data

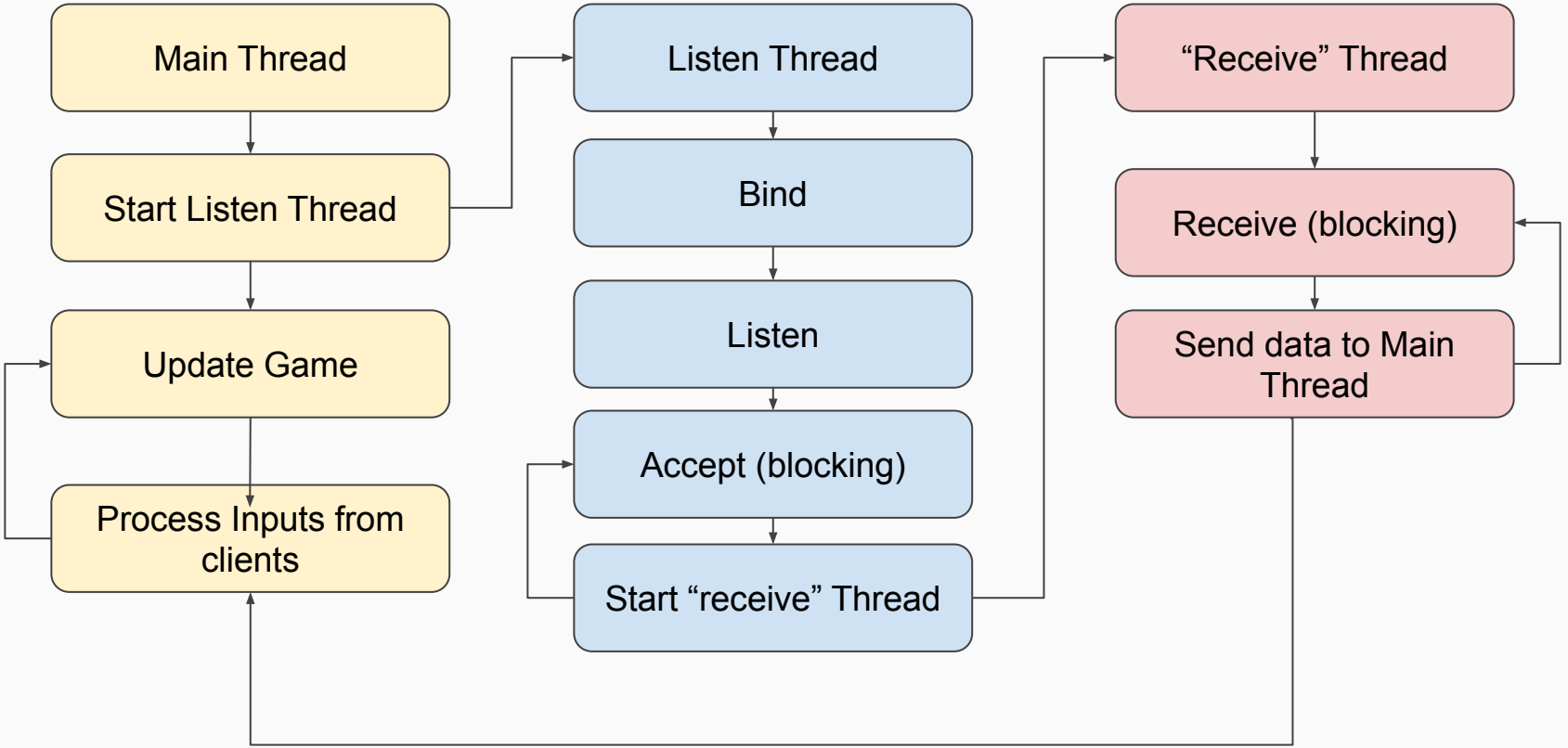
Listening for connections

Trying to connect

...

Avoid affecting the framerate!

Option 1: Multithreading



Option 1: Multithreading

- Scalability?

- How many Threads would you need for 5 clients?

- It’s a manageable solution for handling few sockets, but it can become tricky if not treated carefully

Option 2: Non-Blocking Methods

BeginAccept

BeginConnect

BeginReceive

...

Pro: Non-blocking. Throw an event and call a Callback function when “done” for asynchronous handling

Con: Still uses threads behind. Similar scalability problems when too many connections

⇒Use only with very few sockets (e.g. clients, but not servers)

Non-Blocking Sockets example:

```
void startReceive()
{
    StateObject state = new StateObject();
    client.BeginReceive( data, 0, BuffSize, 0,  new AsyncCallback(ReceiveCallback), state);
}

void ReceiveCallback( IAsyncResult ar )
{
    StateObject state = (StateObject) ar.AsyncState;
    Socket client = state.workSocket;
    // Read the data:
    int bytesRead = client.EndReceive(ar);
    Debug.Log(Encoding.ASCII.GetString(state.buffer,0,bytesRead));
    // Usually, keep reading because there might be more data to read
    client.BeginReceive(state.buffer,0,StateObject.BufferSize,0, new AsyncCallback(ReceiveCallback), state);
}
```

Option 3: Select() function

```
public static void Select (System.Collections.IList? checkRead, System.Collections.IList? checkWrite,
                           System.Collections.IList? checkError, int timeOut);
```

Consults a list of sockets to see what operation can be called on them without blocking

checkRead, checkWrite and checkError will return the same list with the “non-ready” elements removed

e.g. waiting for data on 5 sockets (clients) and 2 already sent a message

First, checkRead will have length of 5 (all the sockets you have active)

Select will change it to length 2 (only the sockets where you can read without blocking)

https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.socket.select?view=net-5.0

Exercise 2

Build a Room manager for your videogame:

On your game create 2 scenes:

- Server scene:
  - Probably named “create game” or similar
  - It starts a socket and waits for incoming messages from the client scene
  - It answers with the name of the server
- Client scene:
  - Probably named “join game”
  - It allows to insert an IP
  - It connects to a server, knowing it’s IP
  - It sends a message with the user’s name
- Both scenes:
  - Bonus: Lead to a waiting room
  - Bonus: Extend the waiting room with extra information and message exchanges in a chat
  - Bonus: Accept several connections

Implement both cases using UDP and using TCP

Extend it as you like

Submission rules

- .ZIP file
  - README: Txt/pdf with your improvements, instructions, any info you think I should know beforehand
  - .Exe
  - Small video showing the functionality
  - A package of all the relevant scenes
- Name it Surname_Name_Lab2

Evaluation

Acceptance criteria:

- Works as instructed
- No errors in console
- No crashes

Evaluation:

- UDP: 25%
- TCP: 25%
- Bonus: 40%
- Clean code: 10%

Q&A