Lab2 - Socket Programming

COMP280 - Multiplayer Game Development
Hands-on Instructions

Purpose: Implement a Servers and Clients using Socket programming approach.

Due Date(s):

- Class Work portion(s): in the end of class(es)

ClassWork (50%):

- Follow the hands-on class work.

Contents

1. Implementations of a Simple TCPIP Server and TCPIP Client

- Note that in the following you'll need to substitute all the instances of **{YourInitials}** with your initials (even in the source code). For example, for me it will be **AT**.
- Make a folder named **SimpleTCPIPServer_and_Client_{YourInitials}**.

1.1. Python

- Install Python (if not already installed)
- Make a subfolder named **Python** of the folder **SimpleTCPIPServer_and_Client_{YourInitials}**.
- Write the file **simple_server_{YourInitials}.py**

```python
# simple_server_{YourInitials}.py
import socket
HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print(f"S: Received {data} from client")
            conn.sendall(data+b" from server")
```

- Write the file **simple_client_{YourInitials}.py**

```python
# simple_client_{YourInitials}.py
import socket
HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432        # The port used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b"Hello, world - from client")
    data = s.recv(1024)
print(f"Received {data}!")
```

- Open a command prompt; go to the folder with `cd {folderName}`
- Execute the server with `python simple_server_{YourInitials}.py`
- Open another command prompt; go again to the folder with `cd {folderName}`
- Execute the client with `python simple_client_{YourInitials}.py`

Notice that there will be just one connection of the client and both client and server will terminate. This is not a usual sever behaviour. The expectations for servers are that they should be **on** all the time and should be *shut down* by an explicit command from their admin. Also they should accept requests from diffwerent clients (at least more than one). The next example implements a simple server that runs *"forever"* and accepts connections from many clients. It just *echoes back* what data it gets from the respective client(s).

- Write the file **simple_forever_server_{YourInitials}.py**

```python
# simple_forever_server_{YourInitials}.py
import socket
HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    while True:
        conn, addr = s.accept()
        #print("Waiting for data:")
        #with conn:
        print(f"Connected by {addr}: conn={conn}")
        while True:
            #print(f"Receiving data from conn:{conn}")
            data = conn.recv(1024)
            if not data:
                break
            print(f"S: Received {data} from client")
            conn.sendall(data+b" from server")
```

- Test it (run this server first and then a few clients). Notice that the server stays up.

1.2. C

See the provided Zip file

1.3. C++

See the provided zip file

1.4. C#

1.4.1 simple_client_at.cs

```
using System.Net;
using System.Net.Sockets;
using System.Text;

using Socket client = new(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
//IPAddress iPAddress = IPAddress.Any;
IPAddress iPAddress = IPAddress.Parse("127.0.0.1");
int port = 54321;
IPEndPoint iPEndPoint = new IPEndPoint(iPAddress, port);

await client.ConnectAsync(iPEndPoint);
while (true)
{
    // Send message.
    var message = "Hi friends 👋!<|EOM|>";
    var messageBytes = Encoding.UTF8.GetBytes(message);
    _ = await client.SendAsync(messageBytes, SocketFlags.None);
    Console.WriteLine($"Socket client sent message: \"{message}\"");

    // Receive ack.
    var buffer = new byte[1_024];
    var received = await client.ReceiveAsync(buffer, SocketFlags.None);
    var response = Encoding.UTF8.GetString(buffer, 0, received);
    if (response == "<|ACK|>")
    {
        Console.WriteLine(
            $"Socket client received acknowledgment: \"{response}\"");
        break;
    }
    // Sample output:
    //     Socket client sent message: "Hi friends 👋!<|EOM|>"
    //     Socket client received acknowledgment: "<|ACK|>"
}

client.Shutdown(SocketShutdown.Both);
```

1.4.2 simple_server_at.cs

```
// See https://aka.ms/new-console-template for more information
using System.Net.Sockets;
using System.Net;
using System.Text;

Console.WriteLine("Hello, World from simple_server_at!");
//using System.Net.Socket;
using Socket listener = new(AddressFamily.InterNetwork,SocketType.Stream,ProtocolType.Tcp);
IPAddress iPAddress = IPAddress.Any;
//IPAddress iPAddress = IPAddress.Parse("127.0.0.1");
int port = 54321;
IPEndPoint iPEndPoint = new IPEndPoint(iPAddress, port);


listener.Bind(iPEndPoint);
listener.Listen(100);

var handler = await listener.AcceptAsync();
while (true)
{
    // Receive message.
    var buffer = new byte[1_024];
```

```
    var received = await handler.ReceiveAsync(buffer, SocketFlags.None);
    var response = Encoding.UTF8.GetString(buffer, 0, received);

    var eom = "<|EOM|>";
    if (response.IndexOf(eom) > -1 /* is end of message */)
    {
        Console.WriteLine($"Socket server received message: \"{response.Replace(eom, "")}\"");

        var ackMessage = "<|ACK|>";
        var echoBytes = Encoding.UTF8.GetBytes(ackMessage);
        await handler.SendAsync(echoBytes, 0);
        Console.WriteLine($"Socket server sent acknowledgment: \"{ackMessage}\"");

        break;
    }
    // Sample output:
    //    Socket server received message: "Hi friends 👋!"
    //    Socket server sent acknowledgment: "<|ACK|>"
}
```


Deliverables: - Submit a .zip of the work folder.