Network Programming 3

Use of class Socket for working via UDP protocol in synchronous mode

If an UDP protocol is necessary, i.e. a protocol without establishing a connection, it can be used **SendTo** methods for sending messages, and in order to receive **datagrams** it can be used **ReceiveFrom.**

The mechanism for UDP listening to a socket is the following:

1. Create an object type **Socket**, having assigned his network type (in the below example AddressFamily.InterNetwork(IPv4), transport protocol type SocketType.Dgram (UDP) and ProtocolType.UDP) ;

```
socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolTy
pe.UDP);
```

2. **Bind** a received socket with an IP address and port on a server by means of requesting of socket Bind method. Bind accepts an object of **IPEndPoint** class in a capacity of the parameter and encapsulates and IP address of a server and port clients are going to be connected to.

```
socket.Bind(new IPEndPoint(IPAddress.Parse("10.2.21.129"), 100));
```

3. Request from a socket a **ReceiveFrom** method, and upon that an execution of the current flow will be suspended till the data emerges within an incoming buffer. **ReceiveFrom** returns a number of read bytes. If a size of a buffer for reading will be insufficient, then there can arise an exception SocketException;

```
int l = rs.ReceiveFrom(buffer, ref ep);
String strClientIP = ((IPEndPoint)ep).Address.ToString();
String str = String.Format(“\ nReceived from {0}\r\n{1}\r\n”, strClientIP,
System.Text.Encoding.Unicode.GetString(buffer, 0, l));
```

4. For data transmission, use a SendTo method

Lab-work project: Client-Server Chat

Status	Planned 🌱
Team	Start with you and add others with @mentions
Date	13.02.2018

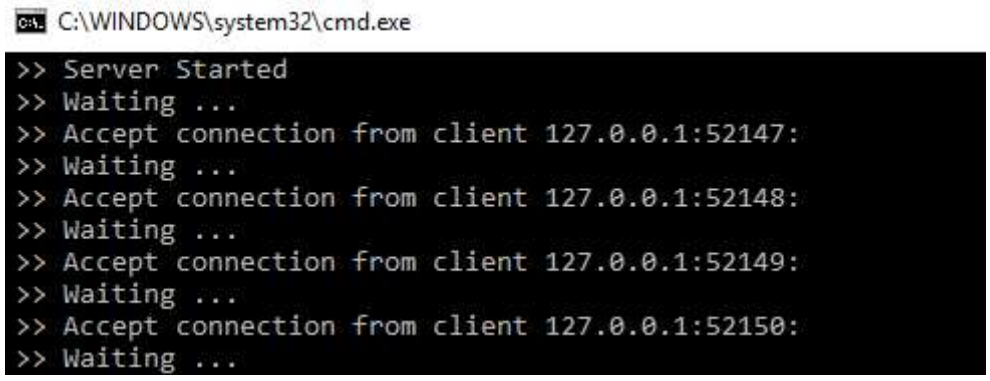
Action items

1. **UDP**

Server

Initially Console Application with argument <**port**>

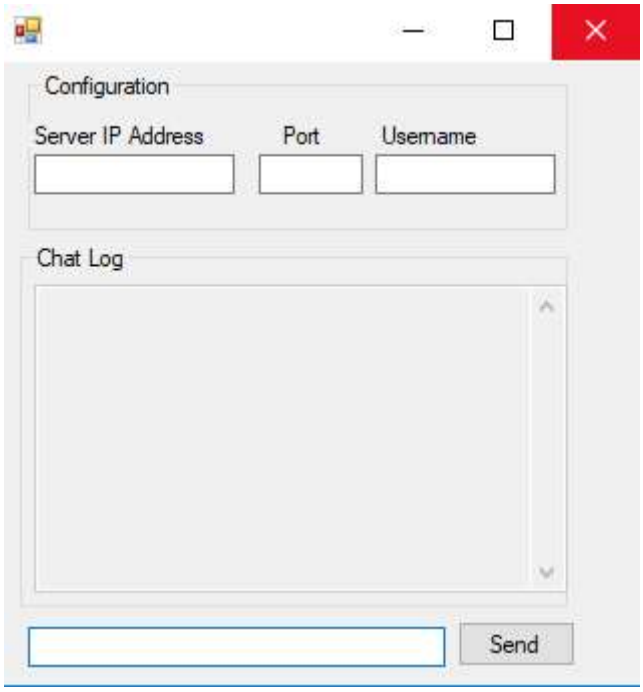
- [ ] Create an UDP server socket that listens on 127.0.0.1
  - create an IP address (localhost)
  - choose the same port for client and server
  - create a local endpoint that uses this address and port given as argument
  - display a message ">>Server started"
- [ ] Create main loop logic that interprets messages like: join, quit and default(switch after message type)
  - Create a remote endpoint that has any IP address and port 0
  - The socket receives from the client the message, at first that could be of type: join, quit or other(use **ReceiveFrom**)
  - Use Encoding.ASCII.GetString(data) for getting the message
  - On the join case, the server display " >> Accepted connection from .."
  - On the quit case, the server displays ">> Closed connection from .. "
  - On the default case, the server sends back the message received(use **SendTo**) and then breaks
  - In the main loop, the server shows ">> Waiting"
- [ ] Add possibility to accept connections from multiple clients: use a list of clients(use ArrayList)
- [ ] Add windows form for the server and show the list of clients that are connected and port



Client

Windows forms application

- [ ] Client: Create windows Forms that looks like this:



- [ ] On button Join event click: an UDP socket is created, a remote endpoint that uses the IP Address entered, message join is sent to the server. On a newly created thread, the client expects receiving messages
- [ ] On button Quit event click: message quit is sent to the server
- [ ] On button Send, the message in the text area, it is sent to the server. Create a method “Send" that represents the callback for a newly created thread
- [ ] Window title: “Connected to <server name> on <port no>"

2. **TCP**

- [ ] Adapt the application to work with TCP Sockets

Background

Goal

Client-Server Chat

- +Network Programming 1
- +Network Programming 2