Exercise to deliver 1

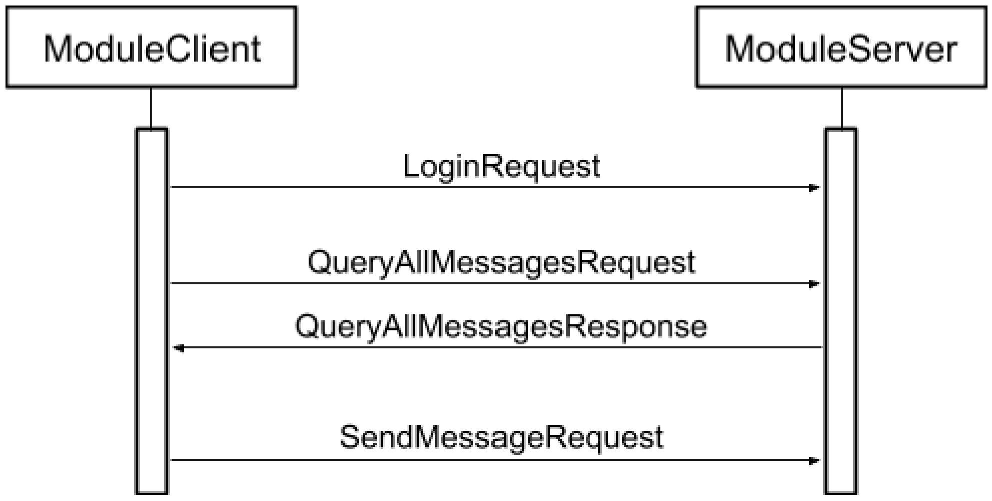
So far we have seen how to use the functions provided by the sockets library in Windows at a very low level. We have learned how to configure sockets to work in both UDP and TCP protocols, and we have learned the most relevant functions to connect hosts, send, and receive data over the network. In this exercise, we will see all these concepts integrated in a more practical scenario.

In this exercise you have available the basic code base of a client/server messaging application. The tasks to complete are the following ones:

1) Fill the **TODOs** in the classes *ModuleClient* and *ModuleServer*. You will have to complete the code that performs the serialization and deserialization of messages between client and server instances of the application.
2) **Add some functionalities** (to be decided by students) that extends the basic functionality of the application. For example, you could add the possibility of deleting messages, searching for messages, include the date and time of messages, etc.
3) Make it work using a connection to a data base that stores all the messages sent by clients. Fill the **TODOs** in the class *MySQLDatabaseGateway* that send SQL statements to a database created by you. Add the necessary methods for the specific extra functionalities added by you at point 2).

TODOs in ModuleClient and ModuleServer

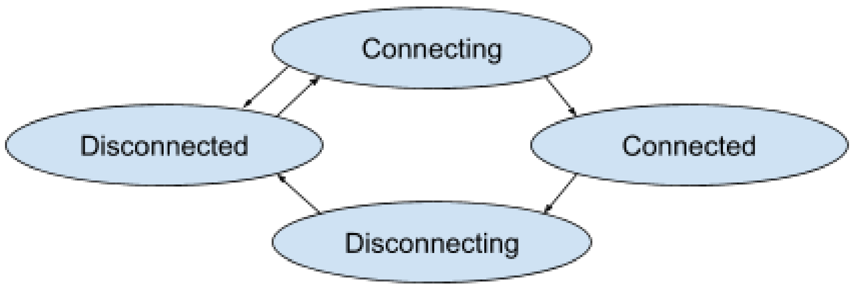
You will have to implement the serialization and deserialization of the following four types of packets:



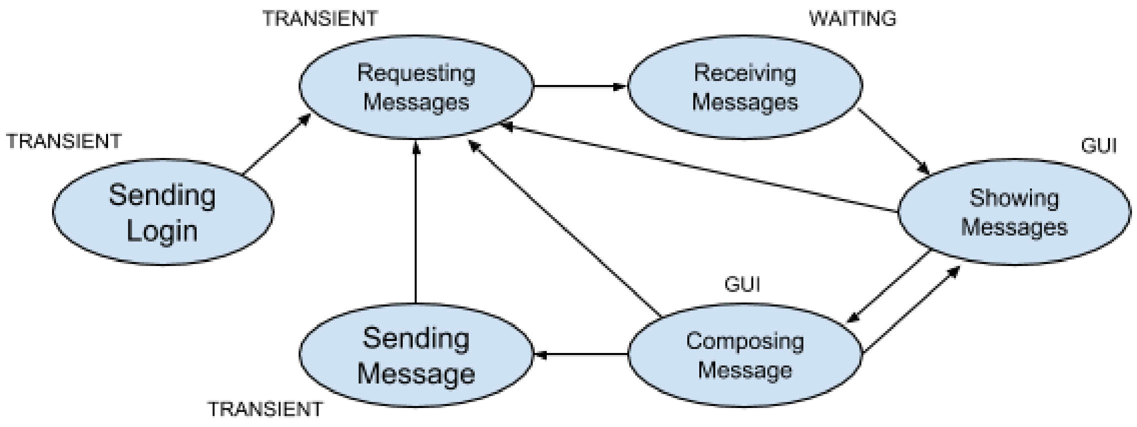
Each serialized packet sent from an instance of the application will be received in another host, and then, it will have to be deserialized to obtain the information correctly and carry out the required operation.

State machine of the client application

The function *ModuleClient::update()* executes client different code depending on the state of the following state machine:



Whenever the client is connected to the server (Connected), the function *ModuleClient::updateMessenger()* executes different code depending on this state machine:



Files with important data types for the exercise

You will have to use different data types defined in these two files:

- *serialization/PacketTypes.h* - Contains the identifiers of differents types of packets
- *serialization/MemoryStream.h* - Contains the classes *OutputMemoryStream* and *InputMemoryStream* to perform the serialization and deserialization of packets respectively.
- *database/DatabaseTypes.h* - Contains the data structure for messages

To-do list

Type the serialization and deserialization code of the Login packet:

1. *ModuleClient::sendPacketLogin(...)*
2. *ModuleServer::onReceivePacket(...)*
3. *ModuleServer::onReceivePacketLogin(...)*

Type the serialization and deserialization code to obtain all messages in the inbox:

4. *ModuleClient::sendPacketQueryMessages(...)*
5. *ModuleServer::onPacketReceivedQueryAllMessages(...)* // Nothing to do actually
6. *ModuleServer::sendPacketQueryAllMessagesResponse(...)*
7. *ModuleClient::onPacketReceivedQueryAllMessagesResponse(...)*

Type the serialization and deserialization code to send a new message:

8. *ModuleClient::sendPacketSendMessage(...)*
9. *ModuleServer::onPacketReceivedSendMessage(...)*

Add other functionalities

You also have to extend the basic functionality of this application with some extra features. For example:

- Add a password to the login (the server should store a list of registered users and a hash of the password to validate login requests).
- Add the possibility of deleting messages.
- Add data and time to the sent messages.
- Mailing lists.
- Online chat...
- Others…

Imagination will be considered when evaluating the exercise. You have the necessary tools to send and receive data among clients and server, and you know how to use ImGUI to create user interfaces. No restrictions to this part of the exercise, be creative!

Connection to MySQL

To enable the connection with a database, you will have to follow these steps:

1) **Create a database in a server.** You can use the database (with the same name of your user in the virtual campus) available in citmalumnes.upc.es. You can create the table to store the messages right there. Alternatively, if you want, you can use a local server using a XAMP enviroment or similar. The table of messages should have at least the following fields: *(sender, receiver, subject, body)*.
2) **Fill the TODOs** in the file ***MySqlDatabaseGateway.cpp***, containing the **SQL statements** to insert and consult messages from the database.

NOTE: Make sure that any modification you introduce either in *IDatabaseGateway* and *MySqlDatabaseGateway*, you also make it in *SimulatedDatabaseGateway*, so that the application can be executed in simulated mode, without needing an actual connection to a database.

Delivery instructions

You will have to upload a .zip file with the contents of the solved exercise. The .zil file should contain the following:

1) The Visual Studio with the actual source code
2) A **README** file in **pdf** format with the following information:
    - Name of the students in the group.
    - A short explanation + screenshots of the extra functionalities/features added.
    - List of the new (or modified) packets introduced in the application.
    - New state machine diagram of the client application, in case the original state machine graph has changed in the new version.
    - Description of the tables (and their fields) created in the database.

NOTA: Do not write your user and password in any place for the delivery (even if it is required for the connection to the database). Do not worry about that. I will evaluate the delivered code and the presented documentation in the README.pdf. To evaluate the exercise, the server instance will be executed in simulated mode.