REDES DE COMUNICAÇÕES 1

LABORATORY GUIDE

Objectives

- Sockets (in Python)
  - UDP Sockets
  - TCP Sockets with textual data, fixed sized binary data packets, and variable sized binary data packets.

Duration

- 1 week

UDP Sockets (Connection-Less data transmission)

1.1. Start the provided UDP server (*serverUDP.py*), which will open an UDP Socket, listen in all available IPv4 interfaces/addresses (using the IPv4 address 0.0.0.0) in port 5005 , and print all messages from clients.

1.2. Start a Wireshark capture on the server machine and analyze the received UDP packets. Start the provided UDP client (*clientUDP.py*) on the same machine or on a remote machine changing the server IPv4 address (variable *ip_addr*). From one (or more clients) send messages to the server. Analyze the code from both server and client. Explain the usage/choice of the source UDP ports by the client(s).

TCP Sockets (Connection Oriented data transmission)

Textual data messages and clients handled with Threads

2.1. Start the provided TCP server (*serverTCP.py*), which will open an TCP Socket, listen in all available IPv4 interfaces/addresses (using the IPv4 address 0.0.0.0) in port 5005, print all messages from clients and send an ECHO message back to the client. This server expects “messages with textual data”.

2.2. Start a new Wireshark capture on the server machine and analyze the received TCP packets. Start the provided TCP client (*clientTCP.py*) on the same machine or on a remote machine changing the server IPv4 address (variable *ip_addr*). From one (or more clients) send messages to the server. Analyze the code from both server and client. Explain the usage/choice of the source TCP ports by the client(s), how the sessions are created, and how different clients are handled by different threads.

Textual data messages and clients handled with Selector

3.1. Start the provided TCP server (*serverTCPsel.py*), which will open an TCP Socket, listen in all available IPv4 interfaces/addresses (using the IPv4 address 0.0.0.0) in port 5005, print all messages from clients and send an ECHO message back to the client. This server expects “messages with textual data”.

3.2. Start a new Wireshark capture on the server machine. Start the provided TCP client (*clientTCP.py*) on the same machine or on a remote machine changing the server IPv4 address (variable *ip_addr*). Analyze the code from both server and client. Explain know how different clients are handled by the server using Selectors and Selector keys.

Binary fixed size data messages and clients handled with threads

4.1. Start the provided TCP server (*serverTCPv2.py*), which will open an TCP Socket, listen in all available IPv4 interfaces/addresses (using the IPv4 address 0.0.0.0) in port 5005, and print all messages from clients. This server expects “messages with binary fixed size data”, where the header/data structure is: 1 byte for the protocol version, two unsigned longs (2x32 bytes) to packet order and original message size, and 20 chars/bytes to carry the message.

Note: the data structure is defined using the package *struct*. See more information: https://docs.python.org/3/library/struct.html

4.2. Start a new Wireshark capture on the server machine. Start the provided TCP client (*clientTCPv2.py*) on the same machine or on a remote machine changing the server IPv4 address (variable *ip_addr*). Analyze the code from both server and client. Explain how data is being sent and decoded.

4.3. Change the server/client code to include a server ECHO response.

Binary variable size data messages and clients handled with threads

5.1. Start the provided TCP server (*serverTCPv3.py*), which will open an TCP Socket, listen in all available IPv4 interfaces/addresses (using the IPv4 address 0.0.0.0) in port 5005, and print all messages from clients. This server expects “messages with binary variable size data”, where the header/data structure is: 1 byte for the protocol version, two unsigned longs (2x32 bytes) to packet order and original message size, and a number of chars/bytes (define by the size field) to carry the message.

Note: the data structure is defined using the package *struct*. See more information: https://docs.python.org/3/library/struct.html

5.2. Start a new Wireshark capture on the server machine. Start the provided TCP client (*clientTCPv3.py*) on the same machine or on a remote machine changing the server IPv4 address (variable *ip_addr*). Analyze the code from both server and client. Explain how data is being sent and decoded.

5.3. Change the server/client code to include a server ECHO response.