Telecommunications Network

Practice 4

TCP (review)

TCP (review)

- socket()

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- bind()

```
server_address = ('localhost', 10000)
server.bind(server_address)
```

- listen()

```
server.listen(1)
```

- accept()

```
connection, client_address = sock.accept()
```

TCP (review)

- send(), sendall()

```
connection.sendall(data.encode())
```

- recv()

```
data = connection.recv(16).decode()
```

- close()

```
connection.close()
```

- connect()

```
server_address = (‚localhost’, 10000)
client.connect(server_address)
```

Socket timeout

- setblocking() or settimeout()
- If an operation can't be completed on the socket, wait until it can be completed:

```
sock.setblocking(1)    # or sock.setblocking(True)
                       # or sock.settimeout(None)
```

- Throw an exception after a given amount of time (settimeout with nonzero value) or immediately, if the operation can't be completed:

```
sock.setblocking(0)     # or sock.setblocking(False)
                        # or sock.settimeout(0.0)
                        # or sock.settimeout(1.0)
```

Socket configuration

- socket.setsockopt(level, optname, value): sets a given socket option
- The **level** values we will use in this course:
  - socket.**IPPROTO_IP**: IP level option
  - socket.**SOL_SOCKET**: socket API level option
- **optname** is the name of the parameter we would like to set, e.g.:
  - socket.**SO_REUSEADDR**: allow the reusage of the ip address + port combination after the connection terminates
- List of all symbolic constants

Socket configuration

- The **value** parameter can be a bytestring or an integer:
  - In the former case, the caller has to make sure that the bytestring contains the right bits (for example with the use of a struct)
  - In the case of socket.**SO_REUSEADDR** the value 0 turns off the option and all other integers turn it on.

```
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

Select

```
inputs = [ server ]
outputs = [ ]
timeout=1
readable, writable, exceptional = select.select(inputs, outputs, inputs,timeout)
...
for s in readable:
    if s is server: #new client connects
        ...
    else:
        ...             #handle client
```

Exercise I.

- Write a TCP client-server application, where multiple clients can send messages to the server.
- The server should reply with the `b"OK"` bytestring to all messages.
- Use the select function in your solution!

Exercise II.

- Modify the calculator server so that it's able to communicate with multiple clients 'simultaneously'. To achieve this use the select function.
- Modify the calculator client as well:
  - The client should send 5 requests instead of just one. After sending a request it should wait for the server's response before sending another one.
  - Before every request the client should wait an additional 2 seconds (`time.sleep(2)`).
  - The client should only terminate the connection after the arrival of the 5th response.

Chat application

- Write a TCP chat application where multiple clients can ‘talk’ to each other through the chat server.
- After establishing the connection, the clients should send their name to the server.
- When the server receives a message from a client it should extend it with the original sender’s name and forward it to all other clients. Form:[<name>] ; e.g. [Joseph] She turned me into a newt!
- The clients should print the received messages to the standard output.

Assignment III.

Guessing game

- Description

Assignment III.

- Write a guessing game application. The server should be able to handle multiple clients ‘simultaneously’. The server chooses an integer between 1 and 100 and the clients try to guess this number.
- The client’s message is a comparison operator: <, >, = and an integer. The meaning of the message is: Is the chosen number smaller/greater than or in the third case equal to the integer sent by the client.
- The server replies with Yes/No/You won/You are out/End of game messages. The You won and You are out replies are only possible when the client used the ‘=’ operator in it’s message.
- If a client guesses correctly (with the ‘=’ operator) then the server replies with End of game messages to all other clients.
- The server chooses a new integer after the game has ended and all clients have terminated the connection.
- After receiving a You won,You are out or End of game message the clients first terminate the connection then themselves. If they receive a Yes or No message they carry on guessing.

Assignment III.

- The client uses logarithmic search (binary search) to guess the chosen number. The client knows what range was used to choose the number.
- Hence, the client should **NOT** get it’s guesses from the standard input!
- Before every new guess the client should wait a random amount (between 1 and 5) of seconds. This gives us the ability to test with multiple clients.
- Message format:
    - From the client: a single character in binary form, a 32 bit integer. Possible values for the character: <: smaller than, >: greater than, =: equal to
    - From the server: the same format as in the case of the client, but the integer has no meaning (it can be anything). Possible values for the character: I: Yes, N: No, K: You are out, Y: You won, V: End of game .

Logarithmic search (binary search) review

Submission

Arguments:

```
python3 server.py <bind_address> <bind_port> # The pair to be used to bind the socket
python3 client.py <server_address> <server_port> # The ip and port of the server
```

Submission: The program should be submitted through the TMS system in .zip format, which contains a `client.py` and a `server.py` file.

Deadline: See TMS