

Project #1 : Socket Programming with TCP

이름 : 최가온(GAON CHOI)

학번 : 2019009261

과제 설명 – 내용 및 목적 서술

The goal of Project0 is to learn how to build client/server application that communicate using sockets. Socket is a host-local, application-created, OS-controlled interface into which application process can both send and receive messages to/from another application process.

source Files : 본인이 작성한 소스 파일을 캡처 및 중요하고 필요하다고 생각하는 부분을 서술

This project consists of two .py files

1. TCPServer.py

Import necessary library. Specify information on the server IP, server port, and the buffer size.

```
import socket

HOST = "127.0.0.1" # server IP
PORT = 50001      # server Port
BUFFER = 4096     # buffer size
```

Create a socket object through the constructor to implement communication environment between the server and the client.

AF_INET is a constant which means the address (and protocol) families.

SOCK_DGRAM represents one of the socket types.

```
# server socket type (protocol type)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Bind server socket.

```
# bind server socket IP/Port
sock.bind((HOST, PORT))
```

After preparing for connection, print host's IP address and port number.

```
print("UDPServer listen at: %s:%s\n\r" % (HOST, PORT))
```

It consists of nested while loop. In the inner loop, the server waits for a message from the client. The condition for continuation is recv, which means the fact that the client sent message to the server.

socket.recvfrom returns a tuple (bytes, address). "bytes" means a byte stream object of the received data, and "address" means the address of sending socket.

After receiving the message, the server prints sender's address and the message with decoded format as "utf-8".

```
while True:
    # client_sock, client_addr = socket.accept()
    # print("%s:%s connect" % client_addr)

    while True:
        recv, client_addr = sock.recvfrom(BUFFER)

        if not recv:
            break
```

```

        # print the context from Client
        print("[Client %s:%s said]:%s" % (client_addr[0], client_addr[1],
recv.decode("utf-8")))

        # reply to Client
        msg2 = "UDPServer has received your message"
        sock.sendto(msg2.encode("utf-8"), client_addr)

sock.close()

```

2. TCPClient.py

Import necessary library. Specify information on the server IP, server port, and the buffer size.

```

import socket

HOST = "127.0.0.1" # server IP
PORT = 50001      # server Port
BUFFER = 4096     # buffer size

```

Create a socket object through the constructor to implement communication environment between the server and the client.

```

# client socket type (protocol type)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

```

Make a connection to a remote socket at address. If the connection is interrupted by a signal, the method waits until the connection completes, or raise a `socket.timeout` on timeout, if the signal handler doesn't raise an exception and the socket is blocking or has a timeout.

After connection, make a message for sending to TCPServer.

```

# server connection
sock.connect((HOST, PORT))

# send message to Server
my_msg = "Hello, UDPServer!"

```

Send the message object, `my_msg`, to the TCPServer. Before sending the message, it needs to be converted to "utf-8" format.

After sending the message, the TCPServer's response is saved to `recv` by `socket.recv()` method.

```

sock.send(my_msg.encode('utf-8'))

# save message from the server in buffer
recv = sock.recv(BUFFER)

```

As same as `my_msg`, the received message needs to be decoded. After decoding the message, print the message, and terminate by closing the socket.

```

# save context in buffer
print("[UDPServer said]: %s" % recv.decode("utf-8"))

sock.close()

```

Instructions: 본인의 소스 실행 방법 (본인이 실행시킨 방법)

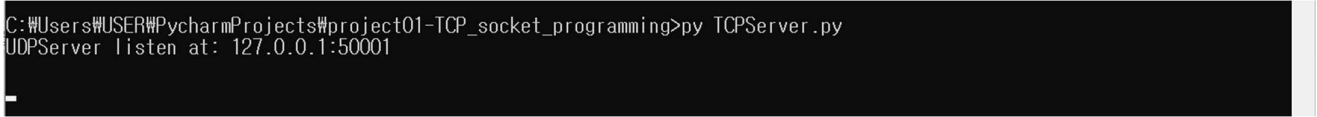
1. Unzip the folder in the executable space. After that, enter the following contents in the command window.

py TCPServer.py



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

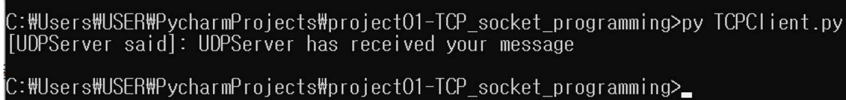
C:\Users\USER>cd C:\Users\USER\PycharmProjects\project01-TCP_socket_programming
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>
```



```
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>py TCPServer.py
UDPServer listen at: 127.0.0.1:50001
_
```

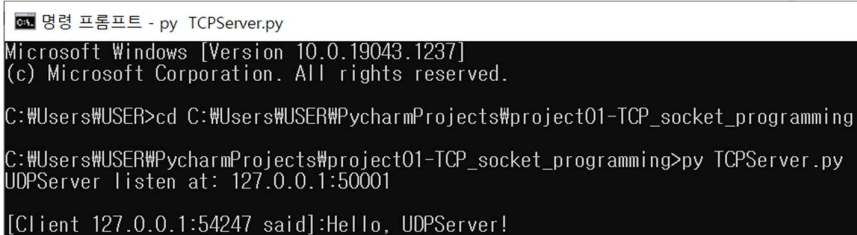
2. Open another command window and enter the following contents in that command window.

py TCPClient.py



```
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>py TCPClient.py
[UDPServer said]: UDPServer has received your message
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>
```

3. You can observe that the TCPServer received a message from the TCPClient. After sending a message, TCPClient terminates automatically.

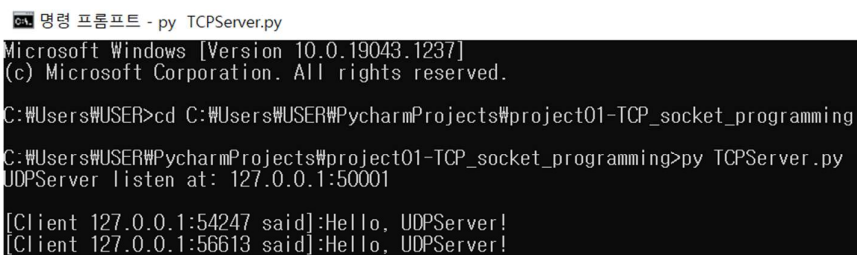


```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>cd C:\Users\USER\PycharmProjects\project01-TCP_socket_programming
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>py TCPServer.py
UDPServer listen at: 127.0.0.1:50001

[Client 127.0.0.1:54247 said]:Hello, UDPServer!
_
```

4. TCPServer awaits for another TCPClient forever. Thus, if another TCPClient tries to connect to TCPServer, the TCPServer can receive a message.



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>cd C:\Users\USER\PycharmProjects\project01-TCP_socket_programming
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>py TCPServer.py
UDPServer listen at: 127.0.0.1:50001

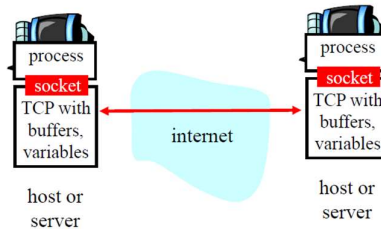
[Client 127.0.0.1:54247 said]:Hello, UDPServer!
[Client 127.0.0.1:56613 said]:Hello, UDPServer!
```

5. After running TCP program, terminate TCPServer.

How the program works: 프로그램 구동 방식

메커니즘 및 프로그램의 작동 방식 서술

Socket is a door between application process and end-to-end transport protocol(UDP or TCP). In this project, TCP is used as a primary transport protocol. TCP service is a reliable transfer of bytes from one process to another. The following image represents two hosts and Internet structure.



For connection, the server and the client must be connected to each other. To this end, the server must run first. And the server should be prepared for this client's contact by creating a socket.

In client's side, it contacts the server by creating client-local TCP socket. IP address, port number of server process must be specified. When client creates socket, client TCP establishes connection to server TCP. When contacted by client, server TCP creates new socket for server process to communicate with client. It allows server to talk with multiple clients.

Results: 결과 화면 캡처 및 설명.

The result of execution is listed below. (same as the image of **Instructions**)

1) TCPServer

```
cmd 명령 프롬프트 - py TCPServer.py
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>cd C:\Users\USER\PycharmProjects\project01-TCP_socket_programming

C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>py TCPServer.py
UDPServer listen at: 127.0.0.1:50001

[Client 127.0.0.1:54247 said]:Hello, UDPServer!
```

2) TCPClient

```
C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>py TCPClient.py
[UDPServer said]: UDPServer has received your message

C:\Users\USER\PycharmProjects\project01-TCP_socket_programming>
```

something Else: 과제에 대해서 건의점, 조교에게 전달되는 점, 질문, 앞으로의 과제에서의 희망사항 등

I could see the detailed process of transmission of message between the server and client using TCP protocol. The server can handle multiple clients. In this project, Project0, there were only one client. I think we can improve this program with parallel computing or thread, to implement multiple users.

References

- [1] MIR Lab, "SocketProgramming.pdf", page 3~6
- [2] Python Software Foundation, <https://docs.python.org/3/library/socket.html>