Name: Arka Haldi
UID: 2018130014
Date: 7/11/20
CEL 51, DCCN, Monsoon 2020
Lab 8: Socket Programming

Socket Programming

- Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents. We use the socket module in python to create and use sockets.

- Python provides two levels of access to network services. At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

- Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

  Sockets have their own vocabulary −

Sr.No.	Term & Description
1	**Domain** The family of protocols that is used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.
2	**type** The type of communications between the two endpoints, typically SOCK_STREAM for connection-oriented protocols and SOCK_DGRAM for connectionless protocols.
3	**protocol** Typically zero, this may be used to identify a variant of a protocol within a domain and type.
4	**hostname** The identifier of a network interface − - A string, which can be a host name, a dotted-quad address, or an IPV6 address in colon (and possibly dot) notation - A string "<broadcast>", which specifies an INADDR_BROADCAST address. - A zero-length string, which specifies INADDR_ANY, or - An Integer, interpreted as a binary address in host byte order.

5	**port**
	Each server listens for clients calling on one or more ports. A port may be a Fixnum port number, a string containing a port number, or the name of a service.

The *socket* Module

- To create a socket, you must use the *socket.socket()* function available in *socket* module, which has the general syntax −

  ```
  s = socket.socket (socket_family, socket_type, protocol=0)
  ```

- Here is the description of the parameters −

  - **socket_family** − This is either AF_UNIX or AF_INET, as explained earlier.

  - **socket_type** − This is either SOCK_STREAM or SOCK_DGRAM.

  - **protocol** − This is usually left out, defaulting to 0.

- Once you have *socket* object, then you can use required functions to create your client or server program.

Server Socket Methods

Sr.No.	Method & Description
1	**s.bind()** This method binds address (hostname, port number pair) to socket.
2	**s.listen()** This method sets up and start TCP listener.
3	**s.accept()** This passively accept TCP client connection, waiting until connection arrives (blocking).

Client Socket Methods

Sr.No.	Method & Description
1	**s.connect()** This method actively initiates TCP server connection.

General Socket Methods

Sr.No.	Method & Description
1	**s.recv()** This method receives TCP message
2	**s.send()** This method transmits TCP message
3	**s.recvfrom()** This method receives UDP message
4	**s.sendto()** This method transmits UDP message
5	**s.close()** This method closes socket
6	**socket.gethostname()** Returns the hostname.

CODE :

SERVER.py

```python
import socket
# create a socket, with AF_INET : Address from the internet
# AF_INET accepts two params : host address and port number
# SOCK_STREAM used to create TCP protocols
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# gethostname is used when client and server are on the same device
# bind the server to the client
s.bind((socket.gethostname(), 1025)) # >1023   ---->  Non - privilege address
# Enter the listening mode
s.listen(5)
# for every connection
while True :
    clt,addr = s.accept()
    print(f'Connection to {addr} established!')
    # pass a message to the connected client
    clt.send(bytes('Socket programming in python!','utf-8'))
    clt.close()
    # end the connection and keep listening
```

CLIENT.py

```python
import socket
# create a socket, with AF_INET : Address from the internet
# AF_INET accepts two params : host address and port number
# SOCK_STREAM used to create TCP protocols
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# using the localhost and the port number allowed on the server
# open a connection to the port on that address
s.connect((socket.gethostname(), 1025)) # connect to the server port

full_message = ''
# limit on the number of bytes read from the server at a time
n_bytes = 6

while True:
    msg = s.recv(n_bytes) # message received, with limit on socket
    print(msg.decode('utf-8'), len(msg)) # print the information received
    full_message += msg.decode('utf-8') # append to the full_message
    if len(msg) < n_bytes :
        break # close the connection, once the message is received

print(full_message)
```

Testing telnet on the server, if it works

```
Command Prompt                                      —    □    ✕

C:\Users\aurko>telnet 192.168.56.1 1025_
```

```
C:\Users\aurko\Documents\College\TY\network\expt8>python SERVER.py
Connection to ('192.168.56.1', 52912) established!
Connection to ('192.168.56.1', 52919) established!
```

```
Socket programming in python!

Connection to host lost.

C:\Users\aurko>_
```

After success, we test our CLIENT.py to connect to the server..

```
aurko@Arka-Haldi MINGW64 ~/Documents/College/TY/network/expt8
$ python SERVER.py
Connection to ('192.168.56.1', 54959) established!

```

```
aurko@Arka-Haldi MINGW64 ~/Documents/College/TY/network/expt8
$ python CLIENT.py
Socket 6
 progr 6
amming 6
 in py 6
thon! 5
Socket programming in python!
```

- As we see, our program works as expected and sends a message in packets of length 6 bytes, and reassembles it at the client side, demonstrating the use of socket programming using Telnet.

Conclusion

- I understood how to successfully establish a connection between client and server using socket programming.

References

1. https://www.geeksforgeeks.org/socket-programming-python/
2. https://www.youtube.com/watch?v=T0rYSFPAR0A
3. https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm
4. https://www.tutorialspoint.com/python_network_programming/python_sockets_programming.htm