Internet Technologies and Application

Homework2

四電子三乙 B10702113 林修維

1. **Implementation**

   Because I don't have two computers, so I use Virtual box to simulate multiple devices. By the way, I choose ubuntu20.04. After this implementation, I consider that socket is such as an object, and it is also unit of client.

2. **TCP Client**

   Client code

```python
import socket

HOST = '140.118.208.214' # the IP address of destination
PORT = 7777 # the port of destination
clientMessage = 'hi, B10702113'

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# build the socket object to transmit text
# socket.AF_INET: IPv4 (Default)
# socket.SOCK_STREAM: TCP (Default)
client.connect((HOST, PORT)) # connection
client.sendall(clientMessage.encode())

serverMessage = str(client.recv(1024), encoding='utf-8')
# specific the message from the server(the message limit is 1k)
print('server response:', serverMessage)

client.close() # close the object
```

   Client output

```
PS C:\Users\jojowei\Desktop\python_first> python -u "c:\Users\jojowei\Desktop\python_first\main.py"
server response: Thank you, B10702113
PS C:\Users\jojowei\Desktop\python_first>
```

   And if I want to create multi-clients, I just declare multi-objects.

3. **TCP Server**

Server code

```python
import socket
HOST = '140.118.208.214' # the IP address of server
PORT = 7777 # the port of server (not default service port)

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# create a socket object to receive text
# socket.AF_INET: IPv4 (Default) transmission between different devices
# socket.SOCK_STREAM: TCP (Default)
server.bind((HOST, PORT)) # bind the address to the socket object
server.listen(10) # specific the limit number of clients

while True: # always running
    conn, addr = server.accept() # wait for client connecting
    print("conn: ",conn)
    # all information of client is in 'conn'
    # proto: kind of protocol, default is 0
    print("addr: ",addr)
    clientMessage = str(conn.recv(1024), encoding='utf-8')

    print('Client message is:', clientMessage)

    serverMessage = 'Thank you, B10702113'
    conn.sendall(serverMessage.encode()) # complete encoding, and send all of data (not specific length)
    conn.close()
```

Server output

```
mycode@mycode:~/wei-server$ python3 server.py
conn:  <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_S
TREAM, proto=0, laddr=('140.118.208.214', 7777), raddr=('140.118.208.95', 52794)
>
addr:  ('140.118.208.95', 52794)
Client message is: hi, B10702113
```

Server waits for clients connecting all the time.

4. **UDP Client**

Client code

```python
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # choose UDP protocol
s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
# specify the socket format
PORT = 1060
network = '140.118.208.232'

s.sendto('Hi, B10702113!!!'.encode('utf-8'), (network, PORT))
data, address = s.recvfrom(65535) # waiting until server connecting
print(data.decode('utf-8'))
s.close()
```

Client output

```
Thank you, B10702113~~
```

5. **UDP Server**

Server code

```python
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # choose UDP protocol
s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
# specify the socket format
PORT = 1060
s.bind(('140.118.208.232', PORT))
print('Listening for broadcast at ', s.getsockname())

while True: # wait all the time, and can't shut down
    data, address = s.recvfrom(65535) # waiting until client connecting
    print(data.decode('utf-8')) # need decoding
    s.sendto('Thank you, B10702113~~'.encode('utf-8'), address)
    print('Receive data from: ',address)
```

Server output

```
Listening for broadcast at  ('140.118.208.232', 1060)
Hi, B10702113!!!
Receive data from:  ('140.118.208.28', 49841)
Hi, B10702113!!!
Receive data from:  ('140.118.208.28', 61448)
Hi, B10702113!!!
Receive data from:  ('140.118.208.28', 61449)
```

We can find that UDP architecture is easier than TCP.

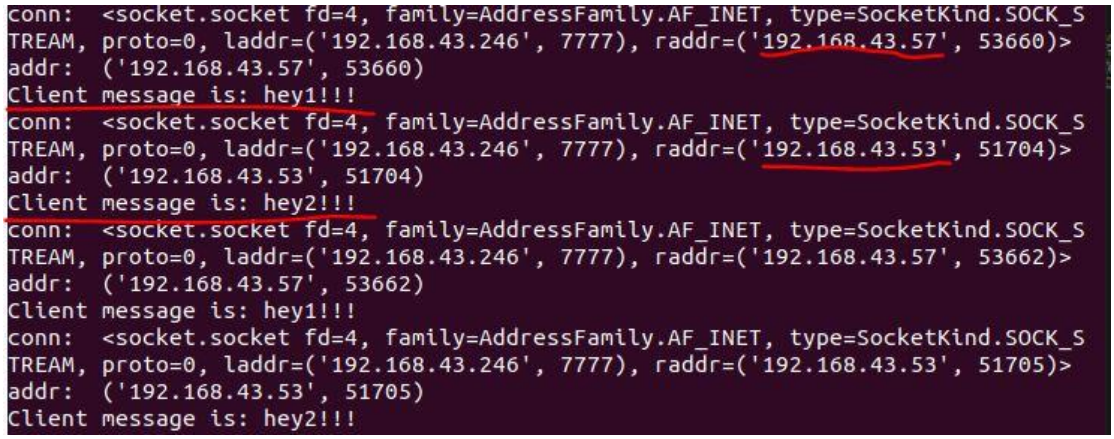
6. **Test**

Change “listen(10)” into “listen(1)”

Establish multiple socket objects from the same address

```
conn:  <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_S
TREAM, proto=0, laddr=('192.168.43.246', 7777), raddr=('192.168.43.53', 51667)>
addr:  ('192.168.43.53', 51667)
Client message is: hey1!!!
conn:  <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_S
TREAM, proto=0, laddr=('192.168.43.246', 7777), raddr=('192.168.43.53', 51668)>
addr:  ('192.168.43.53', 51668)
Client message is: hey2!!!
conn:  <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_S
TREAM, proto=0, laddr=('192.168.43.246', 7777), raddr=('192.168.43.53', 51669)>
addr:  ('192.168.43.53', 51669)
Client message is: hey1!!!
conn:  <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_S
TREAM, proto=0, laddr=('192.168.43.246', 7777), raddr=('192.168.43.53', 51670)>
addr:  ('192.168.43.53', 51670)
Client message is: hey2!!!
conn:  <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_S
TREAM, proto=0, laddr=('192.168.43.246', 7777), raddr=('192.168.43.53', 51671)>
addr:  ('192.168.43.53', 51671)
```

Establish multiple socket objects from the different address



So I find that the queue only happens at “client.connect()”, but there is very low probability of two socket objects connecting at the same time.

7. Compare TCP and UDP

- UDP is simpler than TCP.
- UDP loses packet more easily.
- TCP is more accurate.
- UDP server doesn’t need “listen()” and “accept()”, only need “sendto()” and “recvfrom()”
- UDP uses “sendto()” and “recvfrom()” to exchange address.
- TCP uses “connect()” and “accept()” to exchange address.
- There is no “shut down” in UDP protocol.

8. Summary

- I think I can see socket as an object.
- I can also use “socket.AF_INET” to do local transmission.
- There is low probability of two sockets conflicting.
- UDP format and TCP format are different.
- Different sockets will have its own port.
- UDP is easier than TCP.

9. Reference

Socket of Python: https://clay-atlas.com/blog/2019/10/15/python-chinese-tutorial-socket-tcp-ip/

Interpret of socket: https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/374721/

UDP Tutorial: http://cheng-min-i-taiwan.blogspot.com/2018/11/pythonudp.html