

Networking Lab Assignment 10

Concurrent Time Server -UDP

Albin Antony

14 March 2019

1 Concurrent Time Server-UDP

1.1 Aim

Implement Concurrent Time Server application using UDP to execute the program at remote server. Client sends a time request to the server, server sends its system time back to the client. Client displays the result.

1.2 Theory

1.2.1 Time Server

Implement Concurrent Time Server application using UDP to execute the program at remote server. Client sends a time request to the server, server sends its system time back to the client. Client displays the result. We have a UDP based application which sends back current system time to the server indicating that some operation has been performed at the server.

1.2.2 Client, Server and Socket

- Server- A server is a software that waits for client requests and serves or processes them accordingly.
- Client- a client is requester of this service. A client program request for some resources to the server and server responds to that request.
- Socket- Socket is the endpoint of a bidirectional communications channel between server and client. Sockets may communicate within a process, between processes on the same machine, or between processes on different machines. For any communication with a remote program, we have to connect through a socket port.

1.3 Algorithm

1.3.1 Server

Algorithm 1 Algorithm for creating a concurrent server

```

1 START
2 Create TCP SOCKET
3 Bind SOCKET to a PORT
4 Start listing at the binded PORT for connection from CLIENT
5 WHILE TRUE:
6     ACCEPT connection from CLIENT
7     RECEIVE time request from CLIENT
8     SEND server time to CLIENT
9 STOP

```

1.3.2 Client

Algorithm 2 Algorithm for creating a udp client

```

1 START
2 CREATE TCP SOCKET
3 Connect to server using IP and PORT
4 SEND time request to server
5 RECEIVE time from server
6 STOP

```

1.4 Program

1.4.1 Server

```

import socket
import datetime

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

port=8080

s.bind(('',port))
while True:
    msg, addr=s.recvfrom(1024)
    print ( 'Got Connection from ', addr)
    print 'Message from Client: ', msg
    now = datetime.datetime.now()
    time = now.strftime("%H:%M:%S")
    bytesToSend = str.encode(time)
    s.sendto(bytesToSend, addr)

```

1.4.2 Client

```

import socket

msgFromClient = "Client Requesting Time..."

bytesToSend = str.encode(msgFromClient)

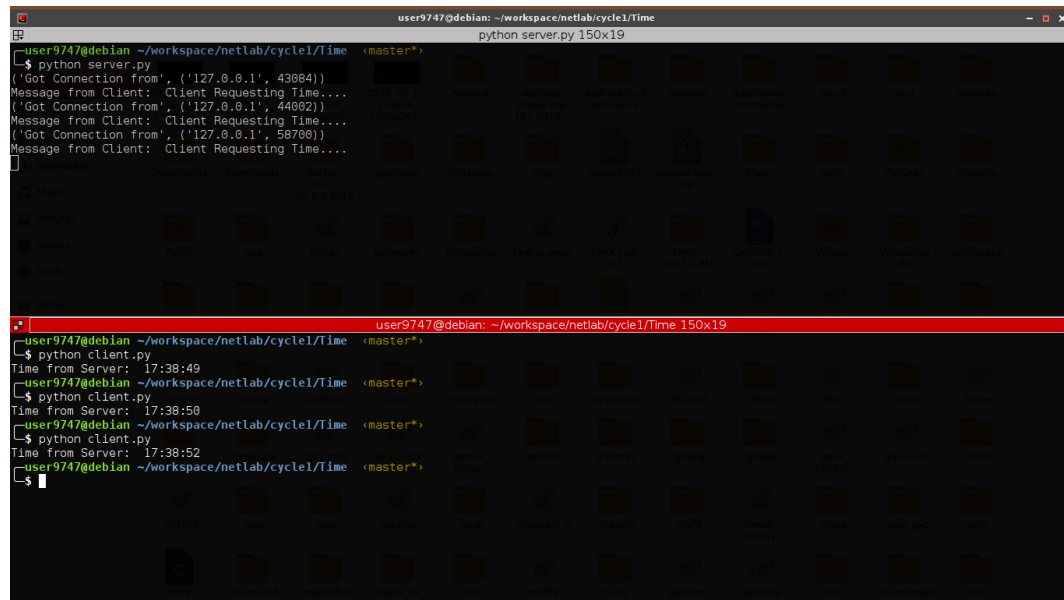
s= socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

port = 8080

```

```
s.sendto(bytesToSend,('127.0.0.1',port))
msgFromServer=s.recvfrom(1024)
# msg = "Message from Server: {}".format(msgFromServer[0])
print "Time from Server: ",msgFromServer[0]
s.close()
```

1.5 Output



```
user9747@debian: ~/workspace/netlab/cycle1/Time
python server.py 150x19
$ python server.py
('Got Connection from', ('127.0.0.1', 43884))
Message from Client: Client Requesting Time....
('Got Connection from', ('127.0.0.1', 44882))
Message from Client: Client Requesting Time....
('Got Connection from', ('127.0.0.1', 58788))
Message from Client: Client Requesting Time....

user9747@debian: ~/workspace/netlab/cycle1/Time 150x19
$ python client.py
Time from Server: 17:38:49
$ python client.py
Time from Server: 17:38:50
$ python client.py
Time from Server: 17:38:52
$
```

1.6 Result

Implemented UDP Concurrent time server on Python 2.7.13 and executed on Debian 4.9 Kernel 4.9 and outputs were verified.

Server code creates a udp socket using the socket library. Then binds the server to port 8080. In an infinite while loop the server receives time request and address from sending clients. Then it sends the current time to this address.

Client code creates a UDP socket same as above. Then sends a time request to the ip and port of the server. It then receives the server time from the server. Then displays the time from the server. Then closes the socket.