CS2S563 Operating Systems

Semester Two, Week 6 Tutorial

In this tutorial, we will demonstrate a concurrent datagram server using UDP, initially by running a server without concurrency, then improving the system to work concurrently.

Tasks 1 to 5 simulate clients connecting to a server which you can do on your own machine. If you have a small network set up though, you could ask one person (perhaps the tutor) to set up the server, and have others act as clients. If you do this, nominate one person to follow step 1 as the server, then the others should follow step 2, and execute the clients at the same time once the server is running. Once you have done this, you are free to work on task 6 either separately, or as a group.

Tasks

1. Create the following file **udpserver.py** and enter the following code.

```
1.  from socket import *
2.  import sys
3.  import time
4.  import os
5.  myHost = ""
6.  myPort = 2000
7.  # create a UDP socket
8.  s=socket(AF_INET, SOCK_DGRAM)
9.  # bind it to the server port number
10. s.bind((myHost, myPort))
11.
12. while True:
13. data, address = s.recvfrom(1024)
14. print ("UDP server:", data, "from", address)
15. if data:
16. start_time = time.time()
17. print ("processing request received")
18. time.sleep (5)
19. end_time = time.time()
20. print ("processing took: ", end_time-start_time, "seconds")
21. s.sendto(("echo -> " + data.decode('utf-8')).encode('utf-8'),
    address)
22. else:
23. break
```

Note the `time.sleep(5)` *on line 18 where we are simulating a large amount of server processing.*

2. Now create a new file **udpclient.py** and write the following.

```
1.  #!/usr/bin/python
2.  import sys, time
3.  from socket import *
```

```python
serverHost = "localhost"
serverPort = 2000
# create a UDP socket
s=socket(AF_INET, SOCK_DGRAM)

s.connect((serverHost, serverPort))
start_time = time.time()
s.send("Hello world".encode('utf-8'))
data = s.recv(1024)
end_time = time.time()
print (data.decode('utf-8'))
print ("time to send to server and get reply was", end_time - start_time, "seconds")
```

3. Now open up four command line terminal windows. In one, write and execute the following.

   ```
   python udpserver.py
   ```

4. and in the other three, write the following, but don't press enter yet!

   ```
   python udpclient.py
   ```

5. Execute the clients in quick succession. What happens? Why are you seeing this?

6. Editing the udpserver.py code, see if you can convert it into a concurrent server

   A successful implementation will allow both clients to connect simultaneously but will only incur a 5 second delay for a transaction.

   Some hints:

   a. read the lecture notes where I cover pseudocode for the UDP concurrent server
   b. in python, `exit(0)` is written `sys.exit(0)`
   c. To use `fork()` you need to `import os` and call `os.fork()` when needed