CSE 461: Introduction to Computer Communication Networks

Rajalakshmi Nandakumar
Goal

• Help finish Project 0

• Learn
  – Socket Programming
  – Threads
  – Asynchronous events
Socket Programming HOWTO – Python 2.7.14 documentation
https://docs.python.org/2/howto(sockets).html
The client application (your browser, for example) uses “client” sockets exclusively; the web server it’s talking to uses both “server” sockets and “client” sockets.

TCP/IP Client and Server - Python Module of the Week
https://pymotw.com/2/socket/tcp.html
Echo Server. This sample program, based on the one in the standard library documentation, receives incoming messages and echos them back to the sender. It starts by creating a TCP/IP socket. Then bind() is used to associate the socket with the server address.

(Very) basic Python client socket example - Stack Overflow
https://stackoverflow.com/questions/.../very-basic-python-client-socket-example
Oct 13, 2011 - It’s trying to connect to the computer it's running on on port 5000, but the connection is being refused. Are you sure you have a server running?

7.2.3 Example
https://docs.python.org/2.4/lib/socket-example.html
Oct 18, 2006 - Here are four minimal example programs using the TCP/IP protocol: a server that echoes all data that it receives back (servicing only one client), and a client using it. ... Also note that the server does not send()/recv() on the socket it is listening on but on the new socket returned by accept().
UDP Server

Receiving

Here's simple code to receive UDP messages in Python:

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

while True:
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    print "received message:" , data
```
UDP Client

Sending

Here's simple code to post a note by UDP in Python:

```python
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = "Hello, World!"

print "UDP target IP:", UDP_IP
print "UDP target port:", UDP_PORT
print "message:", MESSAGE

sock = socket.socket(socket.AF_INET, # Internet
                      socket.SOCK_DGRAM) # UDP
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```
Echo Server Client

CLIENT

MSG

ECHO

SERVER
Solution

1) Thread based

2) Event-loop based
1) Thread based

Thread – Sequence of instructions that are run independently. Main code is a thread

- Create multiple threads each waiting on one input

- You can interact with the server at least with one thread.
Example – multi threaded Server

Server Accept

Thread 1
Waiting for input from client

Thread 2
Waiting for input from stdin
Example – multi threaded Server

Use Threading package

Define a thread with the function

Class ServerThreadPool
def listenToClient(self, client, address):
Some important Thread functions

- Create a thread
- Kill a thread
- Join thread
- Number of active thread
- Current thread
2) Event Loop method

Waits for and notifies events

Occurs asynchronously

- Create an event handle

- Wait for it to notify the occurrence of the event
Example – Event handling

Use pyuv package

Define a event handler function