Programming udp sockets in python

By Silver Moon | October 16, 2012 23 Comments

UDP sockets

UDP or user datagram protocol is an alternative protocol to its more common counterpart TCP. UDP like TCP is a protocol for packet transfer from 1 host to another, but has some important differences. UDP is a connectionless and non-stream oriented protocol. It means a UDP server just catches incoming packets from any and many hosts without establishing a reliable pipe kind of connection.

In this article we are going to see how to use UDP sockets in python. It is recommended that you also learn about programming tcp sockets in python.

Create udp sockets

A udp socket is created like this

```
1 s = socket.socket(socket.AF_INET, socket.SOC
```

The SOCK_DGRAM specifies datagram (udp) sockets.

Sending and Receiving

Since udp sockets are non connected sockets, communication is done using the socket functions sendto and recvfrom. These 2 functions dont require the socket to be connected to some peer. They just send and receive directly to and from a given address

Udp server

The simplest form of a udp server can be written in a few lines

```
import socket
port = 5000
s = socket.socket(socket.AF_INET, socket.SOC
s.bind(("", port))
print "waiting on port:", port
while 1:
    data, addr = s.recvfrom(1024)
    print data
```

A udp server has to open a socket and receive incoming data. There is no listen or accept. Run the above server from a terminal and then connect to it using ncat. Ncat is a telnet alternative that is more powerful and more featureful.

```
$ ncat localhost 5000 -u -v
Ncat: Version 6.00 ( http://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:5000.
hello
ok
```

The u flag indicates the udp protocol. Whatever message we send, should display on the server terminal.

Lets write a complete echo server now.

```
'''
    Simple udp socket server
'''

import socket
import sys

HOST = ''   # Symbolic name meaning all ava
PORT = 8888 # Arbitrary non-privileged port

# Datagram (udp) socket
try :
    s = socket.socket(socket.AF_INET, socke
    print 'Socket created'
```

```
except socket.error, msg :
    print 'Failed to create socket. Error C
    sys.exit()


# Bind socket to local host and port
try:
    s.bind((HOST, PORT))
except socket.error , msg:
    print 'Bind failed. Error Code : ' + st
    sys.exit()

print 'Socket bind complete'

#now keep talking with the client
while 1:
    # receive data from client (data, addr)
    d = s.recvfrom(1024)
    data = d[0]
    addr = d[1]

    if not data:
        break

    reply = 'OK...' + data

    s.sendto(reply , addr)
    print 'Message[' + addr[0] + ':' + str(

s.close()
```

The above will program will start a udp server on port 8888. Run the program in a terminal. To test the program open another terminal and use the netcat utility to connect to this server. Here is an example

```
$ ncat -vv localhost 8888 -u
Ncat: Version 5.21 ( http://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:8888.
hello
OK...hello
how are you
OK...how are you
```

Use ncat again to send messages to the udp server and the udp server replies back with "OK..." prefixed to the message.

The server terminal also displays the details about the client

```
$ python server.py
Socket created
Socket bind complete
```

```
Message[127.0.0.1:46622] - hello
Message[127.0.0.1:46622] - how are you
```

It is important to note that unlike a tcp server, a udp server can handle multiple clients directly since there is no connection. It can receive from any client and send the reply. No threads, select polling etc is needed like in tcp servers.

Udp client

Now that our server is done, its time to code the udp client. It connects to the udp server just like netcat did above.

```
'''
    udp socket client
    Silver Moon
'''

import socket   #for sockets
import sys  #for exit

# create dgram udp socket
try:
    s = socket.socket(socket.AF_INET, socke
except socket.error:
    print 'Failed to create socket'
    sys.exit()

host = 'localhost';
port = 8888;

while(1) :
    msg = raw_input('Enter message to send

    try :
        #Set the whole string
        s.sendto(msg, (host, port))

        # receive data from client (data, a
        d = s.recvfrom(1024)
        reply = d[0]
        addr = d[1]

        print 'Server reply : ' + reply

    except socket.error, msg:
        print 'Error Code : ' + str(msg[0])
        sys.exit()
```

The client will connect to the server and exchange messages like this

```
$ python simple_client.py
Enter message to send : hello
Server reply : OK...hello
Enter message to send : how are you
Server reply : OK...how are you
Enter message to send :
```

Overall udp protocol is simple to program, keeping in mind that it has no notion of connections but does have ports for separating multiple udp applications. Data to be transferred at a time should be send in a single packet.

Last Updated On : 6th August 2016

Related Post

Code a simple socket server in Python

Code a network packet sniffer in python for Linux