Computer Networks

Introduction to Socket Programming in Python

Socket programming is a way of connecting two applications on two (usually different) nodes of a network to communicate with each other. Therefore, to identify a socket we need to:

1. Identify the first node (using an IP address)
2. Identify the second node (using an IP address)
3. Identify an application in the first node (using a port number)
4. Identify an application in the second node (using a port number)

Therefore, a socket is identified by a 4-tuple (IP_1, $Port_1$, IP_2, $Port_2$)

Connection Types

Two different types of connections can be established between the nodes of a network:

- **Connection-oriented**

  For connection-oriented communication the following steps should be followed

  1. $Node_1$ sends a connection request to node
  2. $Node_2$ accepts the request and sends back a confirmation to $node_1$
  3. $Node_1$ and $node_2$ can exchange messages now. All exchanged messages are acknowledged. The missing or corrupt messages are retransmitted.
  4. Either $node_1$ or $node_2$ sends a close connection request
  5. The other node accepts the request and the connection is closed

- **Connectionless**

  $Node_1$ sends a message to $node_2$ without establishing a connection

  No confirmation is sent back from $node_2$. $Node_1$ has no way of finding out if the message was delivered or not.

  Discuss: What are the advantageous and the disadvantageous of each method?

Creating a Connection Oriented Socket

A connection-oriented socket is based on a client-server model. The server, identified by an IP and a port number, waits for the connection requests from the clients (passive waiting). The client should send a request to the server to open the connection. In the following example codes written in Python, the server and client exchange “hello” messages

Socket programming is started by importing the socket library and making a simple socket.

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Hint: for detailed description of the socket function parameters refer to

Parameters of socket function:

Here we made a socket instance and passed it two parameters. The first parameter is **AF_INET** which refers to the address family ipv4, and the second one is **SOCK_STREAM** which means a connection oriented socket using TCP protocol should be established. (TCP is a Transport layer protocol.)

The following code is a sample server written in Python.

```
# first of all import the socket library
import socket

# next create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print( "Socket successfully created" )

# reserve a port on your computer in this
# case it is 12345 but it can be any number between 1024 and 49151
port = 12345

# Next bind the port and IP to the socket
# we have not typed any IP which will cause the socket to use the IP
# allocated to the host
s.bind(('', port))
print("socket bind successful ")

# Create a backlog of size 5. If the server is busy when a request arrives,
# the request will be stored here temporarily. If there are already 5
requests waiting, the new request will be ignored.
s.listen(5)


# a forever loop until we interrupt it or
# an error occurs
while True:

   # Establish connection with client.
   c, addr = s.accept()
   print 'Got connection from', addr

   # a new socket named c is created to communicate with this client.
   # The server will listen to socket s for new requests.
   # send a thank you message to the client.
   c.send('Hello, Thank you for connecting')
   msg = c.recv( )
   print( 'The message from client: ',msg)

   # Close the connection with the client
   c.close()
```

Client Code

```
# Import socket module
import socket

# Create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Define the port on which you want to connect
port = 12345

# connect to the server on local computer
s.connect(('127.0.0.1', port))

# receive data from the server
print( 'server says: ', s.recv(1024))
s.send('Hello there')
# close the connection
s.close()
```

In order to get a better grasp of the socket programming concepts do the following exercises:

- Port scanner:
  - A port is said to be open if there is an application listening to that port. **A port scanner is an application which lists the open ports**.
  - Write a Python code to get **an IP as input** and **list the open ports on that IP**. (*Hint*: if a port is open, the connect request from the client will be accepted by the server.)
  - Try to find out which applications use those ports (you may use web)
  - Use the port-scanner of your operating system to verify if you have found the same ports. Use **netstat** to list the open ports.

Deliverable:

1. The **source code** of your port scanner.
2. A **report** about your algorithm, how it decides which port is open and is used by some specific application, and which open ports are not detected by your implemented method.

Parameters of socket function:

family – The valid values for *address Family* are AF_INET, AF_INET6, AF_UNIX, AF_CAN, AF_PACKET and AF_RDS.

The meaning of these address families and the scenarios under which they are used are given below:

AF_INET	For protocols based on IP addresses using IPv4. An IPv4 address consists of four numbers each ranging from 0 to 255 and each of them separated by a dot. IPv4 is a 32-bit number and supports up to 2 to the power 32 IP addresses.
AF_INET6	For protocols using IPv6 addresses. IPv6 addresses solve the limitation of number of IP addresses in IPv4 as they are represented as a 128-bit number.
AF_UNIX	Used for Unix domain protocols.
AF_CAN	Very shortly, various controllers and applications in automotive vehicles can talk using CANSockets and using the protocol family PF_CAN.AF_CAN is used while creating CANSockets.
AF_PACKET	Allows to create a packet socket. Packet sockets are used while implementing new higher level protocols.
AF_RDS	Used for reliable datagram sockets.

type:

The valid values for socketType are

socket.SOCK_STREAM,

socket.SOCK_DGRAM,

socket.SOCK_RAW,

`socket.SOCK_RDM` and

`socket.SOCK_SEQPACKET`.

The meaning of these socket types and when to use which type are given below:

`socket.SOCK_STREAM`	Specifies a stream socket. Needs to be provided when a streaming connection based client or server is needed which will be used for reliable communication
`socket.SOCK_DGRAM`	Specifies an UDP socket. This option needs to be provided while creating a datagram socket, which will be used in datagram based unreliable communication.
`socket.SOCK_RAW`	Raw sockets allow customizing the TCP and IP layer headers of the packet. The kernel will not modify the packets from a raw socket.
`socket.SOCK_RDM`	Used in reliable datagram based communication though same ordering of packets are not guaranteed.
`socket.SOCK_SEQPACKET`	Used for reliable, sequence ordered datagram based communication.

`proto:`

The default value is zero. It is typically omitted and used in cases like dealing with raw sockets as per the address family specified. If AF_CAN is specified as address family then either CAN_RAW or CAN_BCM can be passed.

`Fileno`:

File descriptor of a socket. When this parameter is specified, the other parameters are ignored and a socket object as given by the File descriptor is returned. The socket object returned is not a duplicate of the original socket.