LẬP TRÌNH SOCKET

PING PONG

Giảng viên: Mai Xuân Phú

Mục tiêu:

- *Hiểu và giải thích được các hàm cơ bản với lập trình socket trên nền tảng sử dụng ngôn ngữ python.*
- *Lập trình 1 ứng dụng đơn giản minh họa mô hình kết nối client-server.*

Nội dung chính:

- *Các bước thiết lập kết nối với socket theo ngữ cảnh kết nối TCP client-server bằng ngôn ngữ python.*
- *Cung cấp ví dụ minh họa về kết nối với socket*
- *Yêu cầu xây dựng ứng dụng đơn giản theo mô hình client-server*

1. *Socket programming*

a. *Socket*

Low-level networking interface (per the BSD API).

Class methods for the Socket module:

Class method	Description
Socket	Low-level networking interface (per the BSD API)
socket.socket(family, type, proto)	Create and return a new socket object
socket.getfqdn(name)	Convert a string quad dotted IP address to a fully qualified domain name
socket.gethostbyname(hostname)	Resolve a hostname to a string quad dotted IP address
socket.fromfd(fd, family, type)	Create a socket object from an existing file descriptor

- Family: The family of protocols that is used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.
- Type: socket.SOCK_STREAM for connection-oriented protocols and socket.SOCK_DGRAM for connectionless protocols.
- Proto (Protocol): Typically zero, this may be used to identify a variant of a protocol within a domain and type.
- Name: IP address
- Hostname: The identifier of a network interface: A string, which can be a host name, a dotted-quad address, or an IPV6 address in colon (and possibly dot) notation. A string "<broadcast>", which specifies an INADDR_BROADCAST address. A zero-length string, which specifies INADDR_ANY, or An Integer, interpreted as a binary address in host byte order.

Instance methods for the Socket module:

Instance method	Description
sock.bind((adrs, port))	Bind the socket to the address and port
sock.accept()	Return value is a pair (conn, address) where conn is a new socket object usable to send and receive data on the connection, and address is the address bound to the socket on the other end of the connection
sock.listen(backlog)	Place the socket into the listening state, able to pend backlog outstanding connection requests *Backlog*: maximum number of queued connections (0-5)
sock.connect((adrs, port))	Connect the socket to the defined host and port

sock.recv(buflen[, flags])	Receive data from the socket, up to *buflen* bytes The *return value* is a string representing the data received
sock.recvfrom(buflen[, flags])	Receive data from the socket, up to *buflen* bytes, returning also the remote host and port from which the data came
sock.send(data[, flags])	Send the *data* through the socket Returns the number of bytes sent
sock.sendall(data[, flags])	Send *data* to the socket (this method continues to send data from string until either all data has been sent or an error occurs) *None* is returned on success
sock.sendto(data[, flags], addr)	Send the *data* through the socket Return the number of bytes sent
sock.close()	Close the socket
sock.getsockopt(level, optname)	Get the value for the specified socket option
sock.setsockopt(level, optname, value)	Set the value for the specified socket option

b. SocketServer

The SocketServer module is an interesting module that simplifies the development of socket servers.

2. Example

This part presents an example which demonstrates socket progamming. (source: http://www.tutorialspoint.com/python/python_networking.htm)

A Simple Server

To write Internet servers, we use the ***socket*** function available in socket module to create a socket object. A socket object is then used to call other functions to setup a socket server.

Now call ***bind(hostname, port)*** function to specify a port for your service on the given host.

Next, call the ***accept*** method of the returned object. This method waits until a client connects to the port you specified, and then returns a connection object that represents the connection to that client.

```
#!/usr/bin/python                          # This is server.py file
import socket                              # Import socket module
s = socket.socket()                        # Create a socket object
host = socket.gethostname()                # Get local machine name
```

```
port = 12345                          # Reserve a port for your service.
s.bind((host, port))                  # Bind to the port
s.listen(5)                           # Now wait for client connection.
while True:
  c, addr = s.accept()                # Establish connection with client.
  print 'Got connection from', addr
  c.send('Thank you for connecting')
  c.close()                           # Close the connection
```

A Simple Client

Let us write a very simple client program which opens a connection to a given port 12345 and given host. This is very simple to create a socket client using Python's socket module function.

The socket.connect(hosname, port) opens a TCP connection to hostname on the port. Once you have a socket open, you can read from it like any IO object. When done, remember to close it, as you would close a file.

The following code is a very simple client that connects to a given host and port, reads any available data from the socket, and then exits −

```
#!/usr/bin/python                # This is client.py file
import socket                    # Import socket module
s = socket.socket()              # Create a socket object
host = socket.gethostname()      # Get local machine name
port = 12345                     # Reserve a port for your service.
s.connect((host, port))
print s.recv(1024)
s.close                          # Close the socket when done
```

3. *PingPong exercise*

From above example, students rewrite a client-server system, which:

- Server listens on a specified port (determined by user), accepts all client connection requests, shows client information, receives a string from client, then capitalizes string and resends this new string to client.
- Client connects to server on specified port (determined by user), sends a string (by user) to server, receives new string from server and display it on screen.

4. *Assignment*

Chatting program

Reference:

1. Charles Severance. *Python for Informatics - Exploring Information*. 2013
2. https://www.ibm.com/developerworks/linux/tutorials/l-pysocks/
3. http://www.tutorialspoint.com/python/python_networking.htm