Comput 410

Lab2

Sockets in Python

Ali Sajedi ????

All or some parts of the slides of this presentation are duplicated / changed from the References (last page)

Exercise

- Write a Python program to manage a server socket.
- The server should reply to each typed message in the client by adding **your name** to it, like;
  - Hello
  - Hello Ali
- It should be able to respond to different clients simultaneously.

- Test your server with at least three clients (using telnet).

* Follow the slides for guide

Additional Exercise (Optional)

1. Write the same code in a class
2. Use a client socket instead of telnet
3. Change the code of server socket so that it closes the socket and exits after pressing “Esc”

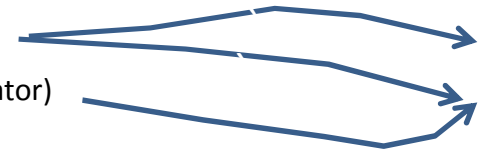
2- Tutorials

1. http://www.binarytides.com/python-socket-programming-tutorial/

2. http://docs.python.org/2/howto/sockets.html

Sockets: Terms and Definitions (1/3)

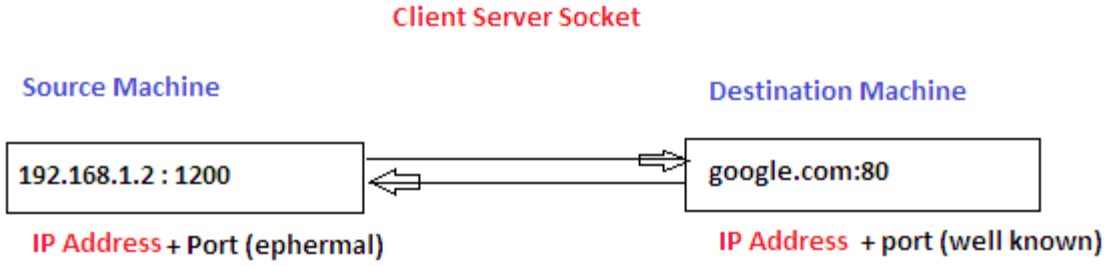
- Sockets are the fundamental "things" behind any kind of network communications done by your computer.
- Example:
  - when you type www.google.com in your web browser,
    - it opens a socket and connects to google.com to fetch the page and show it to you.
  - Chat clients like Skype
- Here:
  - INET sockets * (99%) → Used extensively by the client application (e.g. your browser) and also by the server
  - Stream sockets * (better behavior and performance)
- Socket:
  - blocking *
  - non-blocking
- Socket:
  - Client Socket (endpoint in a conversation)
  - Server Socket (more like a switchboard operator)



Sockets: Terms and Definitions (2/3)

- Socket: An endpoint of communication to which a name can be bound
- INET sockets: Internet (or IP protocol based) sockets which use IP addresses and ports
- Port: a channel for networks communications
  - Port numbers allow different applications on the same computer to utilize network resources without interfering with each other.
  - Port range in IP networking: 0 – 65535
  - Web sites usually use port 80 (Don't need to be mentioned, although you can)
- A TCP connection is defined by two endpoints (sockets):
  - An endpoint (socket) is defined by the combination of a **network address** and a ***port*** **identifier**.
  - Purpose of ports: differentiate multiple endpoints on a given network address ➔ port ≅ virtualized endpoint or logical gate in a device

Socket =
4 tuple ip + port number



Two endpoints in a unique TCP Connection:

client ip + client port number ↔ server ip + server port number

Sockets: Terms and Definitions (3/3)

- *Suggestion: To have a class; TcpConnection* with a *constructor* that takes two arguments:
  - *LocalEndpoint*
  - *RemoteEndpoint*
- Sockets are bidirectional (capable of send & receive)
- Low number ports are usually reserved for some well known services (HTTP, SNMP, etc.)
  - So try to use higher numbers (4 digits at least)

More on Sockets (1/2)

- After clicking on the link … going to a page, your browser did something like:

```
import socket
#create an INET, STREAMing socket:
s = socket.socket( socket.AF_INET, socket.SOCK_STREAM)
#now connect to the web server on port 80
# - the normal http port
s.connect(("www.coursera.org", 80))
```

- When the connect completes, the socket *"s"* can be used to send in a **request** for the text of the page. The same socket will read the **reply**, and then (maybe after some more exchanges) be destroyed.

More on Sockets (2/2)

- But on the web server, a bit more complex things happen: first, a "server socket" is created:

```
import socket
#create an INET, STREAMing socket :
serversocket = socket.socket( socket.AF_INET, socket.SOCK_STREAM)
#bind the socket to a public host, and a well-known port
serversocket.bind((socket.gethostname(), 80))
```

→ The socket is visible to the outside world →

s.bind(('localhost', 80))
or
s.bind(('127.0.0.1', 80))
means that:
The socket is only visible within the same machine

```
#become a server socket
#queue up as many as 5 connect requests before refusing outside connections
serversocket.listen(5)
```

→ Queue up as many as 5 connect requests (the normal max) before refusing outside connections.

```
while 1:        #accept connections from outside
                (clientsocket, address) = serversocket.accept()
                #now do something with the clientsocket #in this case, we'll pretend this is a threaded server
                ct = client_thread(clientsocket)
                ct.run()
```

- There's actually 3 general ways in which this loop could work:
  - create a new process to handle clientsocket, or
  - dispatching a thread to handle clientsocket
  - restructure this app to use non-blocking sockets, and mulitplex between our "server" socket and any active clientsockets using select.

Running Sample 1

- Create a socket:

- Function socket creates a socket and returns a socket descriptor which can be used further.

- Next? Connect to a server (www.google.com) using this socket

Running Sample 2

- Connect to a server

- Next? Send some data to the remote server

Running Sample 3

- Tip: connection, here (SOCK_STREAM/TCP), means a reliable stream of data. These are like separate pipes connecting two endpoints without interfering.

- Send Data

- Next? Receive a reply from the server

Running Sample 4

- Receiving data

- Our web browser does the same thing when we open www.google.com
- This kind of activity represents a CLIENT (a system that connects to a remote system to fetch data)
- Other type of activity: SERVER (a system that uses sockets to receive incoming connections and provide them with data)
  - www.google.com == HTTP server your browser == HTTP client

- Next? Programming server sockets

Running Sample 5 (1/3)

- **Bind** a socket to a particular IP address and a certain port number.
  - By doing this we ensure that all incoming data which is directed towards this port number is received by this application (**not others**).
- Then we need to put the socket in **listening** mode.

Running Sample 5 (2/3)

- Bind a socket to a particular IP & port

  then, listening…

- Now, while this program is running, use “telnet client” to connect to this port:

Running Sample 5 (3/3)

- (Keep this program running and) Open a new terminal and type:
  - telnet localhost 8888 → a network protocol that provides bidirectional interactive text-oriented communication facility using a virtual termninal connection.
- Now the client is connected to the server

- Good, but not much productive
- Note that the connection is established for the purpose of communication

- Next? Reply to the client

Running Sample 6

- Function **sendall** can be used to send something to the socket of the incoming connection and the client should see it.

- And connect to this server in another terminal

  (using "telnet localhost 8888")

Running Sample 7

- So the client (telnet) received a reply from the server.
- But the connection is **closed immediately** after that simply because the server program ends after accepting and sending reply.
  - A server like www.google.com is **always up** to accept incoming connections.
- So we need to keep our server RUNNING **non-stop**.
- The simplest way:
  - To put the **accept** in a **loop** so that it can receive incoming connections all the time.

- A live server will be always alive …

Running Sample 8 – Handling connections using Threads

- Good, but not so effective communication between the server and the client:
  - The server program accepts connections in a loop and just send them a reply, after that it does nothing with them.
  - It is not able to handle more than 1 connection at a time.

- So … handle multiple connections together:
  - To handle every connection:
    - we need a separate handling code to run along with the main server accepting connections.
    - One way is … using **threads**
    - The main server program:
      - accepts a connection and creates a new thread to handle communication for the connection, and then
      - goes back to accept more connections.

References

All or some parts of this presentation were duplicated / changed from the References, thanks to all the authors:

1. http://docs.python.org/2/howto/sockets.html
2. http://www.cs.cf.ac.uk/Dave/C/node28.html
3. http://www.binarytides.com/python-socket-programming-tutorial/
4. http://whatismyipaddress.com/localhost
5. http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/url.html
6. http://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket
7. http://stackoverflow.com/questions/2305465/inet-socket-and-socket
8. http://resources.infosecinstitute.com/socket-programming/
9. http://en.wikipedia.org/wiki/Telnet