

**ECE/CS 5565**  
**NETWORK ARCHITECTURE AND PROTOCOLS**

**Project 1: A Client-Server Application**

**NAME: PRANAVI RAMBHAKTA**

**STUDENT ID: 906109734**

**Tool(s) and version(s) used for development and operating system(s)  
and version(s) used for testing:**

**Operating System: Microsoft Windows Version 10.0.15063**

**Python 2.7.13**

**JAVA version 8 update 144**

**Date: 10/10/2017**

## IMPLEMENTATION OF CLIENT SERVER APPLICATION:

A network-based systems consist of a server, client, and a media for communication. A computer running a program that makes a request for a service is called a client machine and that which offers the requested service is called server machine. However, both the client and server can program can run on the same system as well. In this case, one system plays a dual role as both client machine as well as the server machine. From this, we can define a network application as an application which typically consists of two programs which includes a client program and a server program. The client and server programs might reside in the same system or different systems. When the two programs are executed, a client process and a server process are created and the communication takes place through these two processes.

Sockets provide an interface for programming networks at the transport layer. Sockets are bound to a specific port so that the data can be transferred to the destination carefully and the communication takes place by server listening to a client socket to accept a connection request, reading from server socket and writing to client socket. There could be multiple clients waiting to be connected to the same server and the server serves the requests sequentially in the order of requests. The most important task is to write programs for both the client and server. We build the client server application using python.

The programs should start with importing modules: import os, import sys, import socket:  
os and sys modules provide tools to deal with filenames, paths and directories while the socket modules allow as us to create sockets and communicate via sockets across client and server. The server should additionally import threading for multiple connections sequentially.

sys.argv is a list in Python, which contains the command-line arguments passed to the script. With the len(sys.argv) function you can count the number of arguments. The first argument will be the name of the script file itself. The server program should have two arguments (name of the server script file, server port number) and the client program should have three arguments (name of the client script file, server host name and server port number). The server program should accept the port number while the client program should accept the host name and port number.

These lines create a socket for the server and binds it with a port number and waits to listen for some client to request connection and a maximum of 50 queued connections are possible.

```
serverSocket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
serverSocket.bind((serverHost,serverPort))
serverSocket.listen(50)
```

When a connection is established, accept() method is invoked and a server socket is created.  
connectionSocket, addr=serverSocket.accept()

This line creates the client's socket and indicates that the underlying network is using IPv4 and the socket is of type SOCK\_STREAM i.e TCP socket.

```
clientSocket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
clientSocket.connect((serverHost,serverPort))
```

This line of code establishes a TCP connection between the client and server after the three-way handshake is performed.

These responses are initialized in the server

http200 = 'HTTP/1.0 200 OK\n'	when the file is found
http400 = 'HTTP/1.0 400 Bad Request\n'	when the syntax is wrong
http404 = 'HTTP/1.0 404 Not Found\n'	when the file could not be found

The user gives the filename in the client console and a HTTP request with filename is sent as a message to the server.

```
filename=raw_input("Enter the filename \n");
m='GET '+'/'+filename+' HTTP\1.0\n'
clientSocket.sendall(m)
```

The message is received by the server, and the server then splits the message. If the first element is GET, then the file is searched for in the server.

```
data = connectionSocket.recv(1024)
```

```
if os.path.isfile(filename):
```

If this is true, that means the file is found. Then a message is sent with http 200 followed by content length followed by the data in the file. If the file is not found, http404 is sent and if there is an error in the syntax.

Once the response is sent, the connection of the socket closes and it waits for the next connection request.

```
connectionSocket.sendall(message)
clientthread(connectionSocket)
```

Response is stored in a file and is split. If the code is 400 or 404, it's bad request and file not found respectively. If the code is 200, the file is found and the contents of the file are written into a new created file, with the same name as that requested, into the client. The contents are also printed.

```
file=clientSocket.makefile()
response=file.readline()
code=response.split(" ")
if code[1]=="200":
    contentlen=file.readline()
    lines=file.readlines()
    content=open(filename, 'w')

    for line in lines:
        content.write(line)
    content.close()
```

Exception handling: The server and client programs also handle exceptions.

```
except:
    print 'ERROR: Error processing the client. Closing connection...'
    connectionSocket.close()
```

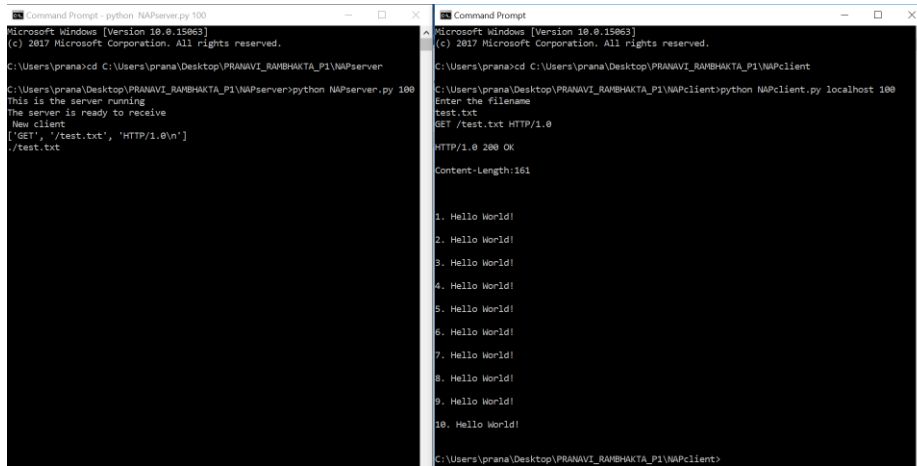
```
except:
    print 'Exiting due to error'
    #socket connection closed
    clientSocket.close()
```

These client, server programs are written in python but they can work with java client or server as well.

## TESTING RESULTS: (Server and client on the same host machine)

### Testing my client with my server:

Case 1: File found in the server and sent to the client



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
['GET', '/test.txt', 'HTTP/1.0\n']
./test.txt

1. Hello World!
2. Hello World!
3. Hello World!
4. Hello World!
5. Hello World!
6. Hello World!
7. Hello World!
8. Hello World!
9. Hello World!
10. Hello World!

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>
```

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test.txt
GET /test.txt HTTP/1.0

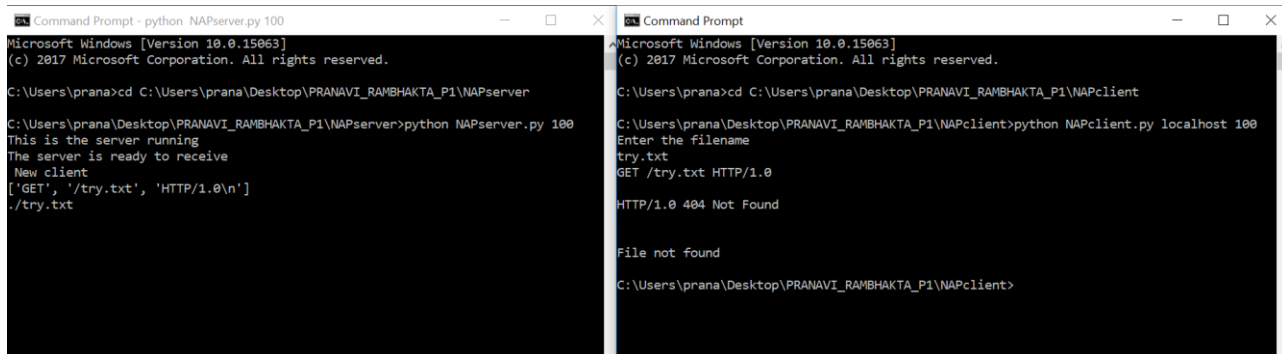
HTTP/1.0 200 OK

Content-Length:161

1. Hello World!
2. Hello World!
3. Hello World!
4. Hello World!
5. Hello World!
6. Hello World!
7. Hello World!
8. Hello World!
9. Hello World!
10. Hello World!

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>
```

Case 2: File not found in the server



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
['GET', '/try.txt', 'HTTP/1.0\n']
./try.txt
```

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

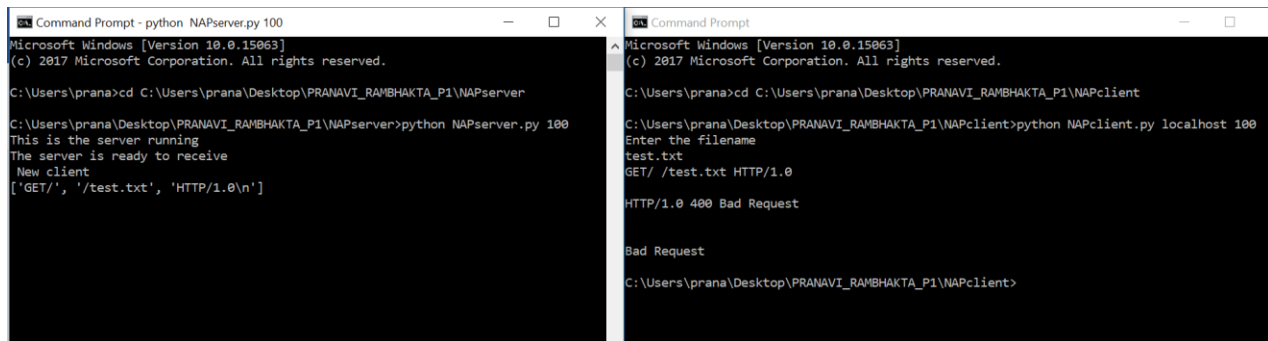
C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
try.txt
GET /try.txt HTTP/1.0

HTTP/1.0 404 Not Found

File not found

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>
```

Case 3: Bad request due to wrong syntax 'GET' in the following line to 'GET/'. For this we modify m='GET/ '+'/' +filename+' HTTP/1.0\n', server sends http 500 bad request is sent as a response to the client.



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
['GET', '/test.txt', 'HTTP/1.0\n']
```

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test.txt
GET /test.txt HTTP/1.0

HTTP/1.0 400 Bad Request

Bad Request

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>
```

Case 4: Multiple requests from clients served in a queued sequential manner by the server.

Consider command in the left bottom to be client 1, right top client 2, right bottom as client 3. In the screenshots, it is seen that client 1 has requested a connection with the server followed by client 2 and then client 3. Client 1 requested test.txt and the server processes its request first. After that, client 3 has requested test2.txt before client 2 requested test1.txt but the server waits till it finishes serving client2 and it will start serving client 3 only after that.

```
Command Prompt - python NAPserver.py 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client

Command Prompt - python NAPclient.py localhost 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
```

Client 1 served:

```
Command Prompt - python NAPserver.py 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
['GET', '/test.txt', 'HTTP/1.0\n']
./test.txt
New client

test.txt
GET /test.txt HTTP/1.0

HTTP/1.0 200 OK

Content-Length:161

1. Hello World!
2. Hello World!
3. Hello World!
4. Hello World!
5. Hello World!
6. Hello World!
7. Hello World!

Command Prompt - python NAPclient.py localhost 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test2.txt
GET /test2.txt HTTP/1.0
```

All clients served, client 2 served followed by client 3

```
Command Prompt - python NAPserver.py 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
['GET', '/test.txt', 'HTTP/1.0\n']
./test.txt
New client
['GET', '/test1.txt', 'HTTP/1.0\n']
./test1.txt
New client
['GET', '/test2.txt', 'HTTP/1.0\n']
./test2.txt

GET /test.txt HTTP/1.0

HTTP/1.0 200 OK

Content-Length:161

1. Hello World!
2. Hello World!
3. Hello World!
4. Hello World!
5. Hello World!
6. Hello World!
7. Hello World!

Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test1.txt
GET /test1.txt HTTP/1.0

HTTP/1.0 200 OK

Content-Length:6

hello!

Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient
C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test2.txt
GET /test2.txt HTTP/1.0

HTTP/1.0 200 OK

Content-Length:12

hello world

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>
```

## Testing my client with sample server:

```
Command Prompt - java HttpServer 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\sample-server

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\sample-server>java HttpServer 100
0 connections served. Accepting new client...
1 connections served. Accepting new client...

Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test.txt
GET /test.txt HTTP/1.0

HTTP/1.0 200 OK

Content-Length: 171

1. Hello World!
2. Hello World!
3. Hello World!
4. Hello World!
5. Hello World!
6. Hello World!
7. Hello World!
8. Hello World!
9. Hello World!
10. Hello World!

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>
```

## Testing my server with sample client:

```
Command Prompt - python NAPserver.py 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
['SET', '/test.txt', 'HTTP/1.0\r\n\r\n']
New client
['GET', '/test.txt', 'HTTP/1.0\r\nHost:', 'localhost\r\n\r\n']
./test.txt

Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\sample-client

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\sample-client>java HttpClient localhost 100
Please input file name: test.txt
Send out a bad request.....
SET /test.txt HTTP/1.0

Receive response from server .....
HTTP/1.0 400 Bad Request

Send out the right request.....
GET /test.txt HTTP/1.0

Receive response from server .....
HTTP/1.0 200 OK
Content-Length:161

Closing I/O and quitting...

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\sample-client>
```

## Exception Handling:

### Server:

```
Select Command Prompt - python NAPserver.py 100
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\prana>cd C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPserver>python NAPserver.py 100
This is the server running
The server is ready to receive
New client
ERROR: Error processing the client. Closing connection...
New client
```

### Client:

```
Select Command Prompt

C:\Users\prana\Desktop\PRANAVI_RAMBHAKTA_P1\NAPclient>python NAPclient.py localhost 100
Enter the filename
test.txt
GET /test.txt HTTP/1.0

Exiting due to error
```

## Summary of test results:

The client and server programs work with any other server and client respectively, either on the same host machine or a different host machine. If both the programs reside on the same host machine, we use 'local host' and port number to connect and if it's in a different host machine, the IP address and the port number to connect. The server and client programs can be of different languages, i.e one can be in python and the other can be in java. Also, the server sequentially connects to multiple clients and processes the requests in the same queue as the connections made, irrespective of the order of requests. The server and the client can handle exceptions if any error occurs. The file which the client requested has to be locally available to the server. The server cannot fetch a file which is available in a different folder or drive in the host machine.