BML MUNJAL UNIVERSITY™
FROM HERE TO THE WORLD

CN PROJECT REPORT

Design and Develop Hardware Configuration of a Server and Host a Website on it

TEAM MEMBERS

Surya Pratap Singh

Puran Singh

Ranbir Singh

Gaurav Saboo

Divyank Behniwal

ABSTRACT

A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients.

The default Python distribution has a built-in support to the HTTP protocol that one can use to make a simple stand-alone Web server. The Python module that provides this support is called BaseHTTPServer and can be used by just including it in our source code.

Wireshark is an open source protocol analyser one use to capture and view the data traveling back and forth within the network. It provides the ability to drill down and read the contents of each packet and is filtered to meet our specific needs.

Once the Web server is up and the clients are connected to the server within the network, tracing the movement of packet from the client to server can be done using Wireshark.

TShark is a network protocol analyser. It helps capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file.

Using Tshark and pandas, CSV file can be created keeping the parameters accordingly.

Pandas is a Python library used to deal with data frames of .csv or .xls format.
Matplotlib.pyplot helps in retrieving conclusion from those data through graphs and charts.

So, using the above software, server creation, web hosting and tracing, tracking and extracting information from packet transfer is possible.

Table of Contents

INTRODUCTION

A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. All computers that host Web sites must have Web server programs.

A web server in Python can be easily made using the built-in support to the HTTP protocol. The Python module that provides this support is called BaseFTTPServer and can be used in our programs just including it in the source code.

Once the web server is made and connection between n server and the client is set up, we can see and analyse the packet movement using a software called Wireshark. It helps us to capture and view the data traveling in the network.

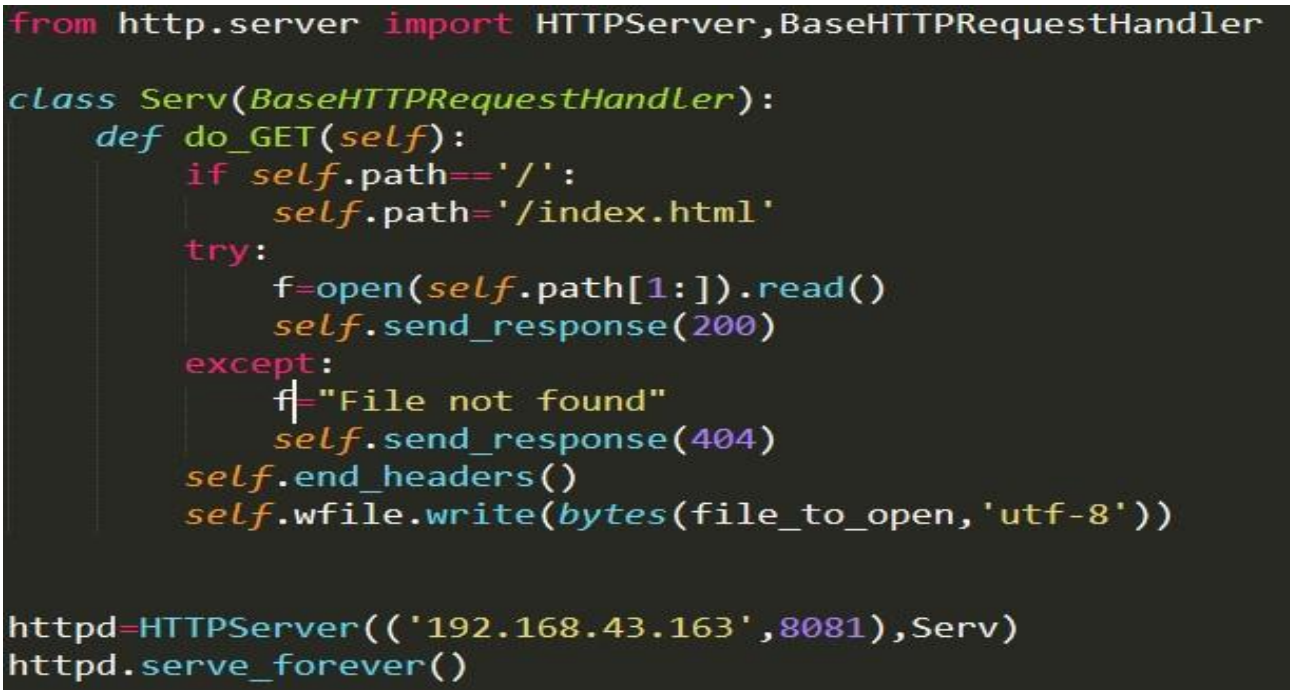
We can see the data travelling to and from the webserver made by using filters by specifying the destination (client) IP address.

Using tshark (or Wireshark) one can get overall data; including ports, destination and get the overall conclusion from that data.

CONTENT

CREATING WEB SERVER

A web server is a program. In our project we made a basic python program

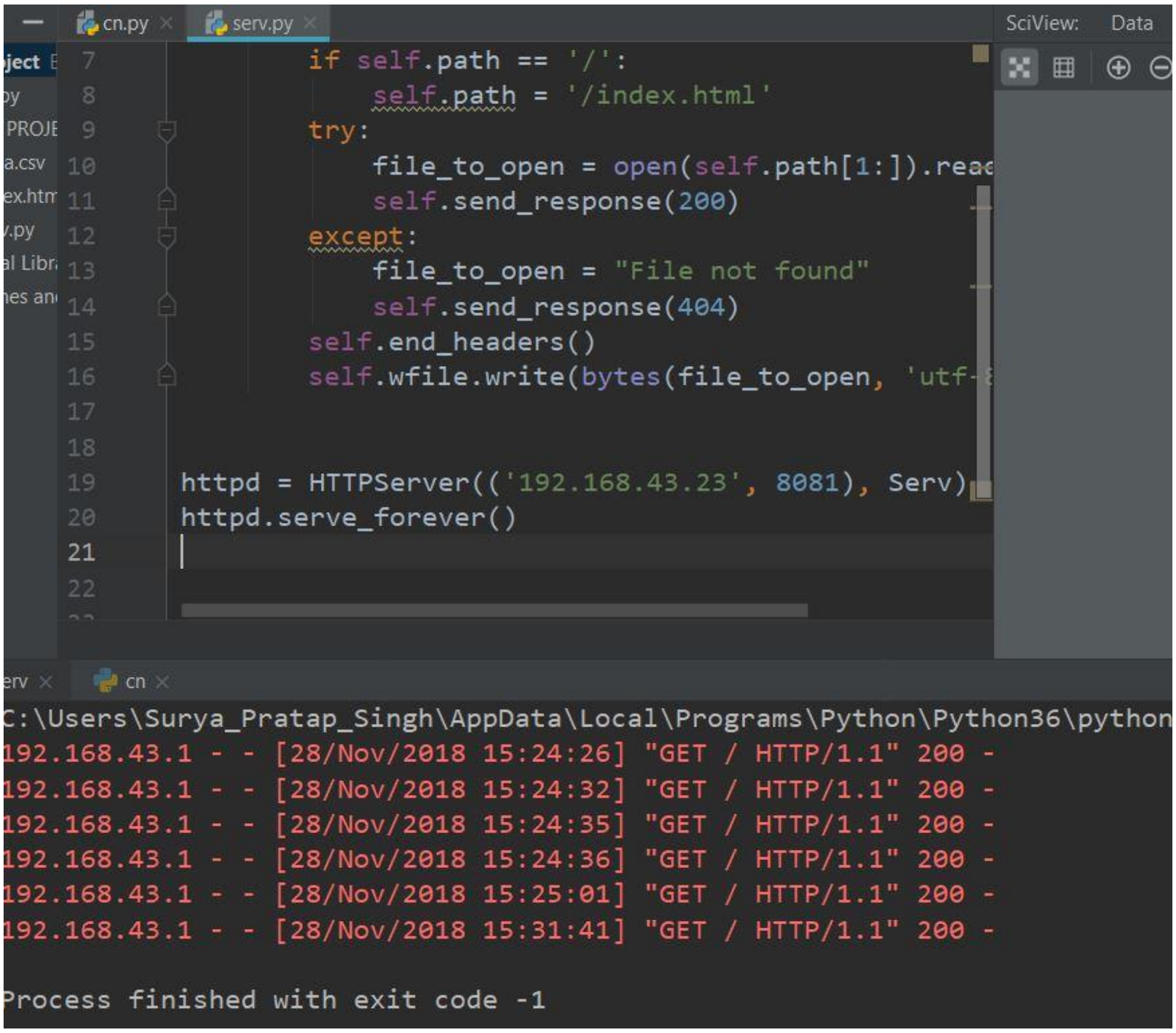


```python
from http.server import HTTPServer,BaseHTTPRequestHandler

class Serv(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path=='/':
            self.path='/index.html'
        try:
            f=open(self.path[1:]).read()
            self.send_response(200)
        except:
            f="File not found"
            self.send_response(404)
        self.end_headers()
        self.wfile.write(bytes(file_to_open,'utf-8'))


httpd=HTTPServer(('192.168.43.163',8081),Serv)
httpd.serve_forever()
```

The class Serv handles any incoming request from the browser. Do_Get is Http handler for the GET requests. Then the try catch block opens the static file requested and sends it to the client. Then web server is created with the IP address and port number the program is running in and the handler is defined to manage the incoming request from the client.

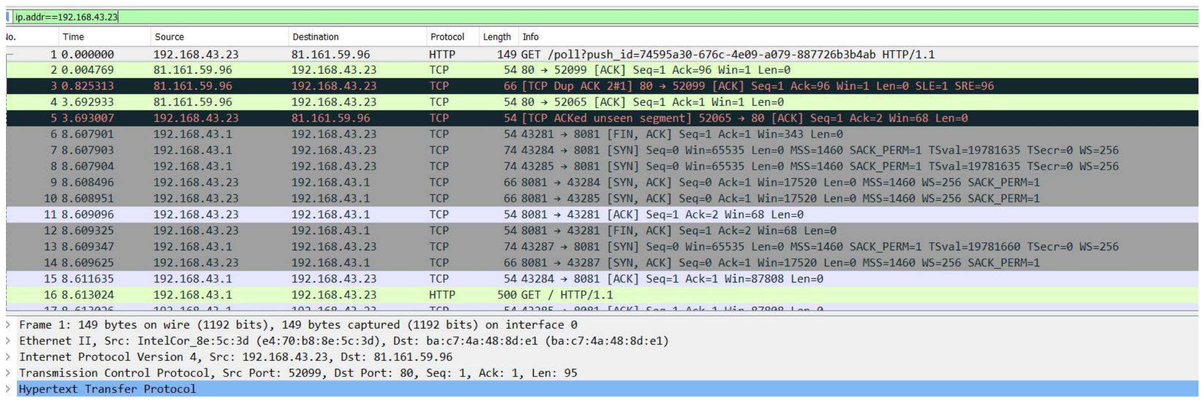
TRACKING PACKETS ON WIRESHARK

Once the server is connected to the client and it starts running packets transfer takes place which can be traced through wireshark.



```
        if self.path == '/':
            self.path = '/index.html'
        try:
            file_to_open = open(self.path[1:]).reac
            self.send_response(200)
        except:
            file_to_open = "File not found"
            self.send_response(404)
        self.end_headers()
        self.wfile.write(bytes(file_to_open, 'utf-


httpd = HTTPServer(('192.168.43.23', 8081), Serv)
httpd.serve_forever()
```

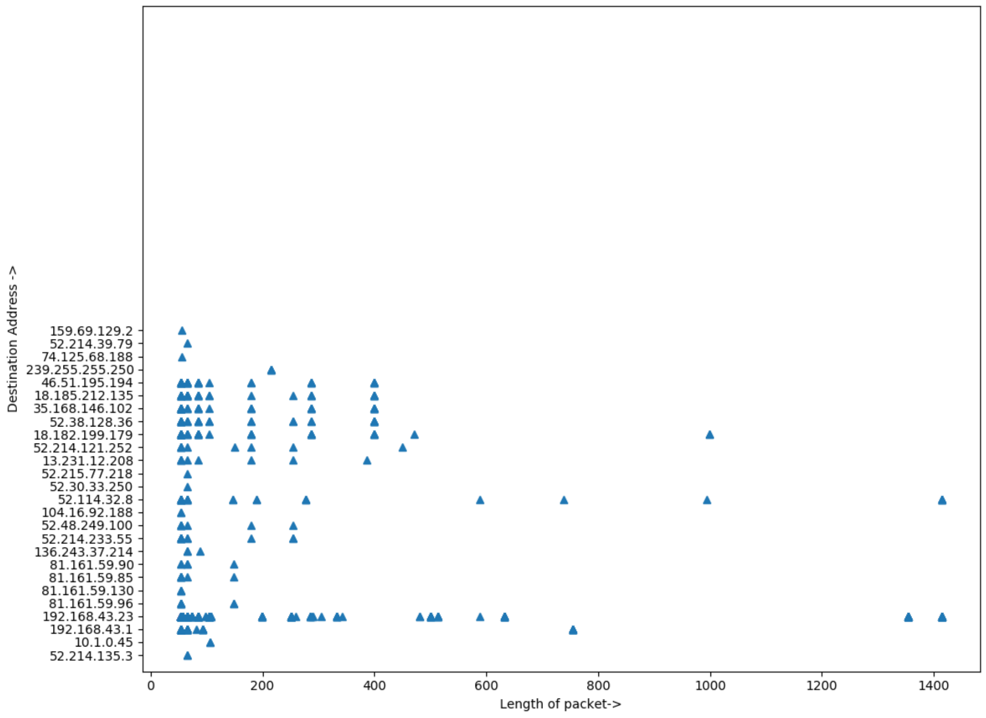
```
C:\Users\Surya_Pratap_Singh\AppData\Local\Programs\Python\Python36\python
192.168.43.1 - - [28/Nov/2018 15:24:26] "GET / HTTP/1.1" 200 -
192.168.43.1 - - [28/Nov/2018 15:24:32] "GET / HTTP/1.1" 200 -
192.168.43.1 - - [28/Nov/2018 15:24:35] "GET / HTTP/1.1" 200 -
192.168.43.1 - - [28/Nov/2018 15:24:36] "GET / HTTP/1.1" 200 -
192.168.43.1 - - [28/Nov/2018 15:25:01] "GET / HTTP/1.1" 200 -
192.168.43.1 - - [28/Nov/2018 15:31:41] "GET / HTTP/1.1" 200 -

Process finished with exit code -1
```



```
ip.addr==192.168.43.23
No.  Time      Source          Destination     Protocol  Length Info
  1 0.000000  192.168.43.23   81.161.59.96    HTTP      149 GET /poll?push_id=74595a30-676c-4e09-a079-887726b3b4ab HTTP/1.1
  2 0.004769  81.161.59.96    192.168.43.23   TCP        54 80 → 52099 [ACK] Seq=1 Ack=96 Win=1 Len=0
  3 0.825313  81.161.59.96    192.168.43.23   TCP        66 [TCP Dup ACK 2#1] 80 → 52099 [ACK] Seq=1 Ack=96 Win=1 Len=0 SLE=1 SRE=96
  4 3.692933  81.161.59.96    192.168.43.23   TCP        54 80 → 52065 [ACK] Seq=1 Ack=1 Win=1 Len=0
  5 3.693007  192.168.43.23   81.161.59.96    TCP        54 [TCP ACKed unseen segment] 52065 → 80 [ACK] Seq=1 Ack=2 Win=68 Len=0
  6 8.607901  192.168.43.1    192.168.43.23   TCP        54 43281 → 8081 [FIN, ACK] Seq=1 Ack=1 Win=343 Len=0
  7 8.607903  192.168.43.1    192.168.43.23   TCP        74 43284 → 8081 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=19781635 TSecr=0 WS=256
  8 8.607904  192.168.43.1    192.168.43.23   TCP        74 43285 → 8081 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=19781635 TSecr=0 WS=256
  9 8.608496  192.168.43.23   192.168.43.1    TCP        66 8081 → 43284 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
 10 8.608951  192.168.43.23   192.168.43.1    TCP        66 8081 → 43285 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
 11 8.609096  192.168.43.23   192.168.43.1    TCP        54 8081 → 43281 [ACK] Seq=1 Ack=2 Win=68 Len=0
 12 8.609325  192.168.43.23   192.168.43.1    TCP        54 8081 → 43281 [FIN, ACK] Seq=1 Ack=2 Win=68 Len=0
 13 8.609347  192.168.43.1    192.168.43.23   TCP        74 43287 → 8081 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=19781660 TSecr=0 WS=256
 14 8.609625  192.168.43.23   192.168.43.1    TCP        66 8081 → 43287 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
 15 8.611635  192.168.43.1    192.168.43.23   TCP        54 43284 → 8081 [ACK] Seq=1 Ack=1 Win=87808 Len=0
 16 8.613024  192.168.43.1    192.168.43.23   HTTP      500 GET / HTTP/1.1

> Frame 1: 149 bytes on wire (1192 bits), 149 bytes captured (1192 bits) on interface 0
> Ethernet II, Src: IntelCor_8e:5c:3d (e4:70:b8:8e:5c:3d), Dst: ba:c7:4a:48:8d:e1 (ba:c7:4a:48:8d:e1)
> Internet Protocol Version 4, Src: 192.168.43.23, Dst: 81.161.59.96
> Transmission Control Protocol, Src Port: 52099, Dst Port: 80, Seq: 1, Ack: 1, Len: 95
> Hypertext Transfer Protocol
```

CONCLUSION

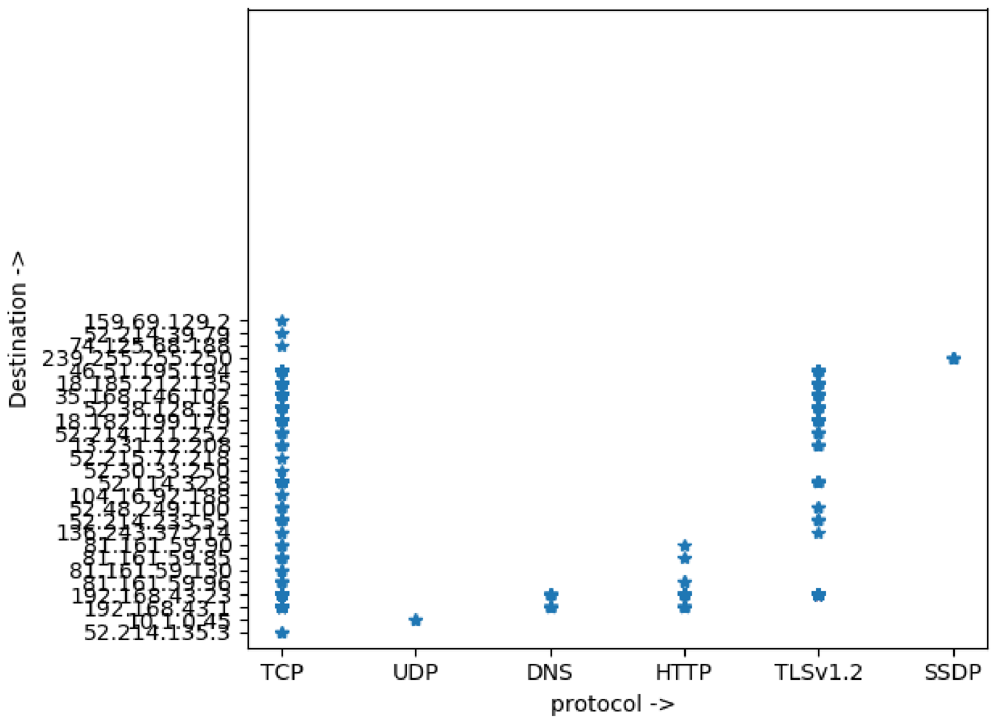
Once we have got the packets traveling in the network we can retrieve many conclusions from that data. In this demonstration, at a time, we connected 4 clients and hit the server.



1. IP address vs length of each packet

Conclusion: A client at different times send different size of packet.

2. IP Address vs Protocol



Conclusion: This graph shows all the protocols clients are using for data communication within the network. Out of all the protocols clients are using for data transfer, TCP is used the most.

REFERENCES

- https://www.acmesystems.it/python_http
- https://docs.python.org/3/library/http.server.html
- https://www.youtube.com/watch?v=TkCSr30UojM