

Measurement Based Modelling

We are trying to model the delay between two networks. First of all, we will send the packets from the source machine to the destination machine. Then we will capture the packet data from both machines using Tshark. Then we will analyze the packet file to calculate the arrival curve and service curve.

Note: Linux System is preferred for this modelling.

We need 2 Linux Machines for this modelling

1. Install Tshark on both Linux systems

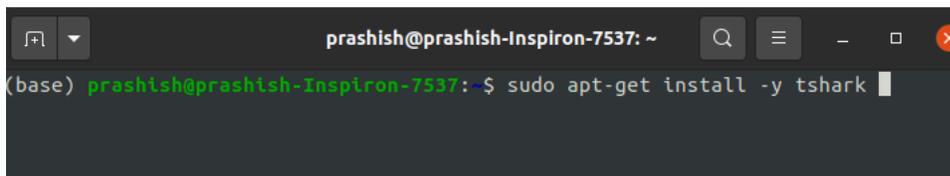
Tshark is a network analyzer tool that helps us to capture packet data from live network. It also enables us to read packets from a previously saved capture file. Tshark is a command line program. So, we will be able to capture packets from the terminal.

Run following command in terminal to install Tshark in both system:

```
sudo apt-get update -y
```

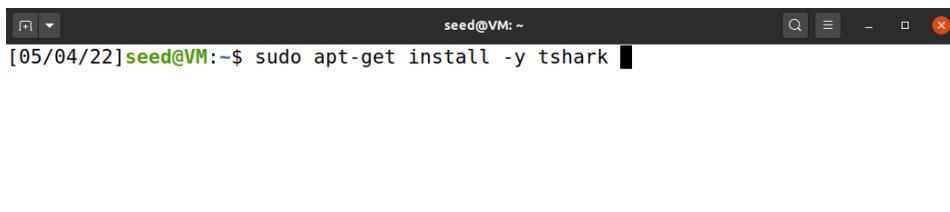
```
sudo apt-get install -y tshark
```

Machine 1:

A terminal window screenshot for Machine 1. The title bar shows 'prashish@prashish-Inspiron-7537: ~'. The terminal prompt is '(base) prashish@prashish-Inspiron-7537:~\$' and the command 'sudo apt-get install -y tshark' is entered and partially executed, with a cursor at the end of the line.

```
prashish@prashish-Inspiron-7537: ~  
(base) prashish@prashish-Inspiron-7537:~$ sudo apt-get install -y tshark
```

Machine 2:

A terminal window screenshot for Machine 2. The title bar shows 'seed@VM: ~'. The terminal prompt is '[05/04/22]seed@VM:~\$' and the command 'sudo apt-get install -y tshark' is entered and partially executed, with a cursor at the end of the line.

```
seed@VM: ~  
[05/04/22]seed@VM:~$ sudo apt-get install -y tshark
```

2. Download Files from the Portal into respective machines

Download **Receiver.py** from the portal into the destination machine, and download **Sender.py** to the source machine.

Link: [Portal](#)

3. Open Code in Python IDE

Open the downloaded code from the portal into the IDE of choice. Here we imported the code inside **Spyder**.

Sender.py on source machine

```
Sender.py X
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Apr 12 10:01:49 2022
4
5  @author: zby09
6  """
7  import numpy as np
8  import socket
9  import threading
10 import time
11 import sys
12 workstation_ip = '172.31.190.182'
13 port = 10052
14 interval = 0.01
15 packet_size = 1000
16 def Sender():
17     server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
18     start = time.time()
19     while(time.time()-start<=10):
20         data = b"1"
21         data = packet_size*data
22         server_socket.sendto(data, (workstation_ip, port))
23         time.sleep(interval)
24     print("done")
25 t1 = threading.Thread(target=Sender, args=())
26
27
28 t1.start()
29 # t2.start()
30 # Receiver()
31
```

Receiver.py on destination machine

```
Receiver.py X
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Apr 12 10:01:49 2022
4
5  @author: zby09
6  """
7  import numpy as np
8  import socket
9  import threading
10 import time
11 import sys
12
13 port = 10052
14
15
16 def Receiver():
17     s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
18     s.bind(("", port))
19     start = time.time()
20     total_amount = 0
21     while True:
22         data, addr = s.recvfrom(65535)
23         total_amount += sys.getsizeof(data)
24         cur = time.time()
25         if(cur-start >=1):
26             print("total_amount: ",total_amount)
27             print("transmission rate: ", str(total_amount*8/(cur-start)/1000000)+"Mbps")
28             total_amount = 0
29             start = cur
30     t2 = threading.Thread(target=Receiver, args=())
31
32 # t1.start()
33 t2.start()
34 # Receiver()
```

4. Install PTP

Since we are working on sending packets between two systems and computing the delay in the network. It is vital to keep the two system clocks synchronized and accurate. For this purpose, we will use PTP (Precision Time Protocol) which is capable of sub-microsecond accuracy.

First, we will install PTP in both source and destination machine by running following command.

```
sudo apt-get install -y linuxptp
```

Installing PTP on source machine.

```
prashish@prashish-Inspiron-7537: ~
(base) prashish@prashish-Inspiron-7537: $ sudo apt-get install -y linuxptp
```

Installing PTP on destination machine.



```
seed@VM: ~  
[05/04/22]seed@VM:~$ sudo apt-get install -y linuxptp
```

5. Enable PTP Synchronization

PTP uses hardware time stamping by default. If both machines are connected using cable, we can try this. Otherwise, we can also use software time stamping.

To enable PTP synchronization on both source and destination machine run the following command in Linux terminal.

For Hardware Time Stamping:

```
ptp4l -i eth3 -m
```

For Software Time Stamping:

```
ptp4l -i eth3 -m -S
```

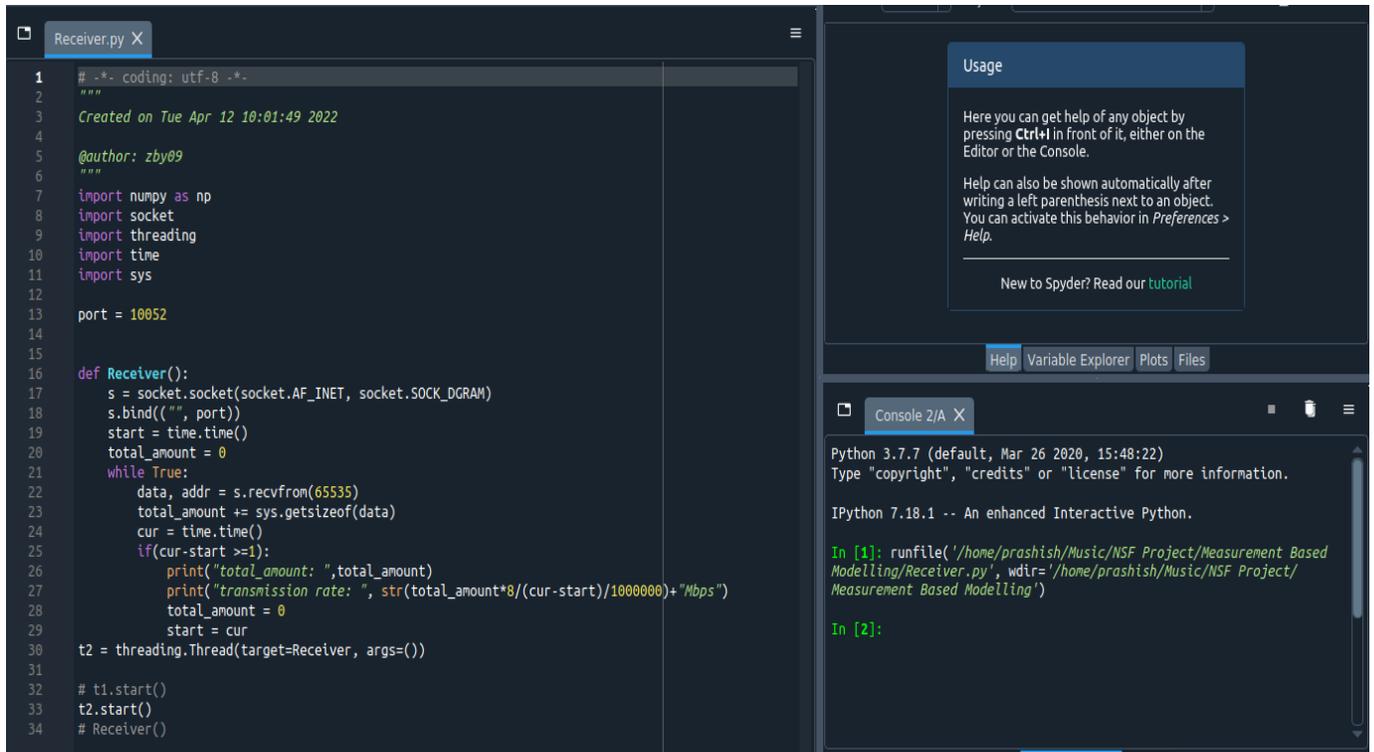
Note: In both commands, **eth3** is the interface you want to configure. You can use the interface of your own choice.

For more detailed info you can refer to this documentation.

Link: [Starting PTP](#)

6. Run Receiver.py

Run **Receiver.py** python file in destination machine.



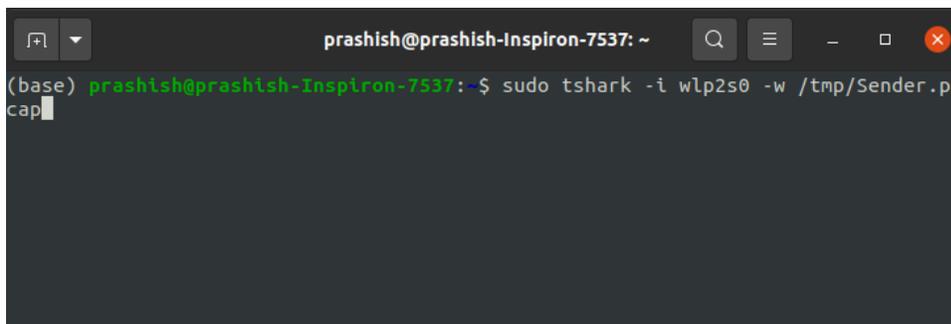
7. Run Tshark on machines 1 and 2

tshark [**-i** <capture interface>|-] [**-w** <outfile>|-]

where <capture interface> is the name of the interface and <outfile> is the name of a file.

Execute the following command to run Tshark on Source Machine

sudo tshark -i wlp2s0 -w /tmp/Sender.pcap



Execute the following command to run Tshark on Destination Machine

`sudo tshark -i wlp3s0 -2 /tmp/Receiver.pcap`

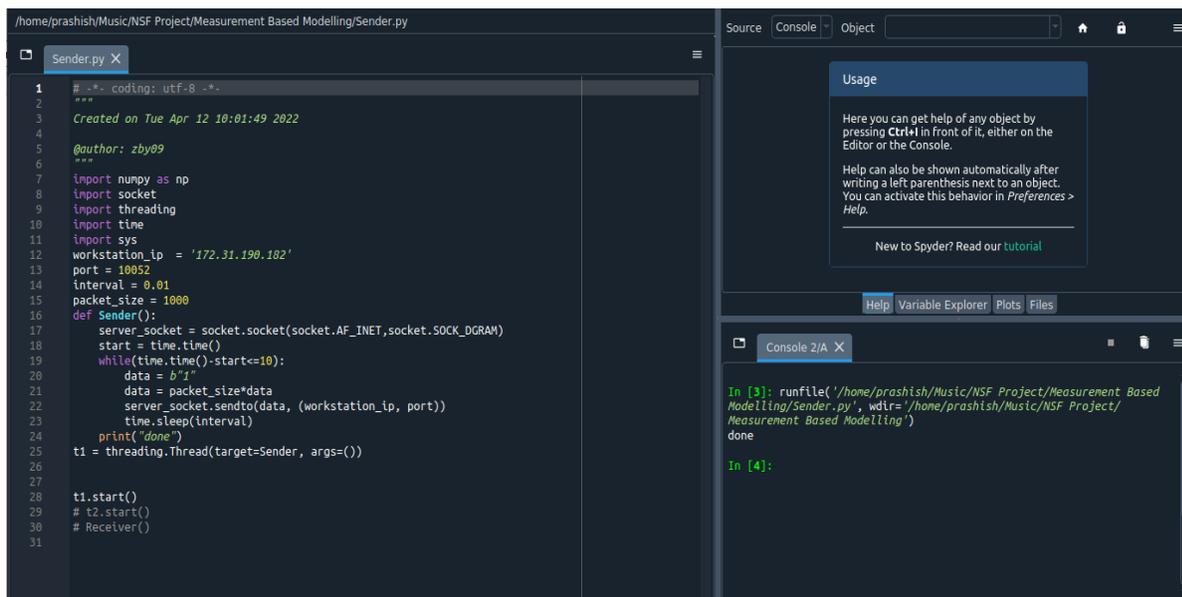


```
seed@VM: ~  
[05/04/22]seed@VM:~$ sudo tshark -i wlp3s0 -2 /tmp/Receiver.pcap
```

This will create **Sender.pcap** file on the source machine and **Receiver.pcap** file on destination machine.

8. Open Sender.py

Inside **Sender.py** set the ip address of **workstation_ip** to ip address of the receiver machine.



```
Sender.py X  
1 # -*- coding: utf-8 -*-  
2 ***  
3 Created on Tue Apr 12 10:01:49 2022  
4  
5 @author: zby09  
6 ***  
7 import numpy as np  
8 import socket  
9 import threading  
10 import time  
11 import sys  
12 workstation_ip = '172.31.190.102'  
13 port = 10952  
14 interval = 0.01  
15 packet_size = 1000  
16 def Sender():  
17     server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  
18     start = time.time()  
19     while(time.time()-start<=10):  
20         data = b"1"  
21         data = packet_size*data  
22         server_socket.sendto(data, (workstation_ip, port))  
23         time.sleep(interval)  
24         print("done")  
25     t1 = threading.Thread(target=Sender, args=())  
26  
27  
28     t1.start()  
29     # t2.start()  
30     # Receiver()  
31
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

Help Variable Explorer Plots Files

```
Console 2/A X  
In [3]: runfile('/home/prashish/Music/NSF Project/Measurement Based Modelling/Sender.py', wdir='/home/prashish/Music/NSF Project/Measurement Based Modelling')  
done  
In [4]:
```

9. Run Sender.py

Run Sender.py on source machine.

```
1  -*- coding: utf-8 -*-
2
3  Created on Tue Apr 12 10:01:49 2022
4
5  @author: zby09
6  """
7  import numpy as np
8  import socket
9  import threading
10 import time
11 import sys
12 workstation_ip = '172.31.190.102'
13 port = 10052
14 interval = 0.01
15 packet_size = 1000
16 def Sender():
17     server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
18     start = time.time()
19     while (time.time()-start)<=10:
20         data = b'J'
21         data = packet_size*data
22         server_socket.sendto(data, (workstation_ip, port))
23         time.sleep(interval)
24         print("done")
25     t1 = threading.Thread(target=Sender, args=())
26
27
28     t1.start()
29     # t2.start()
30     # Receiver()
31
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

Help Variable Explorer Plots Files

```
In [3]: runfile('/home/prashish/Music/NSF Project/Measurement Based Modelling/Sender.py', wdir='/home/prashish/Music/NSF Project/Measurement Based Modelling')
done
In [4]:
```

10. Collect PCAP File

Tshark has built the pcap file on both source machine and destination machine.

Collect those, **Sender.pcap** and **Receiver.pcap** file and copy them into 1 system. We will need these files to calculate the delay in the system.

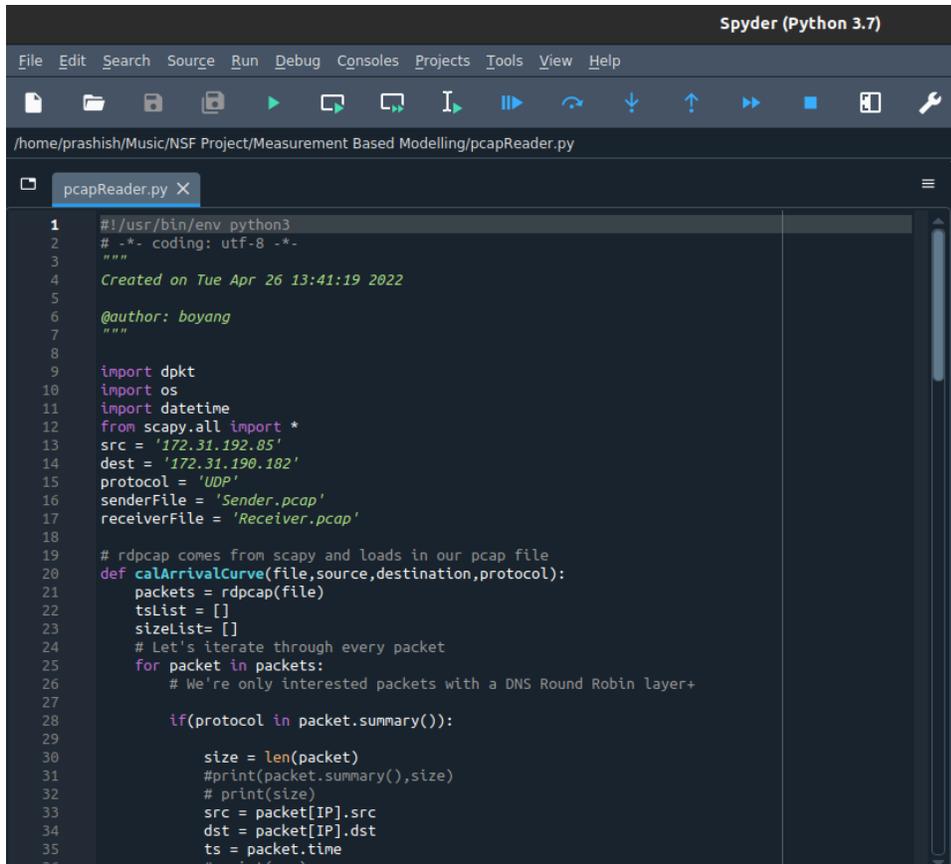
11. Download pcapReader.py

Download **pcapReader.py** from the portal. We will use this python script to compute delay between 2 systems.

Link: [Portal](#)

12. Open pcapReader.py

Open the downloaded code from the portal into the IDE of choice. Here we imported the code inside **Spyder**.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Tue Apr 26 13:41:19 2022
5
6  @author: boyang
7  """
8
9  import dpkt
10 import os
11 import datetime
12 from scapy.all import *
13 src = '172.31.192.85'
14 dest = '172.31.190.102'
15 protocol = 'UDP'
16 senderFile = 'Sender.pcap'
17 receiverFile = 'Receiver.pcap'
18
19 # rdpcap comes from scapy and loads in our pcap file
20 def calArrivalCurve(file,source,destination,protocol):
21     packets = rdpcap(file)
22     tsList = []
23     sizeList = []
24     # Let's iterate through every packet
25     for packet in packets:
26         # We're only interested packets with a DNS Round Robin layer+
27
28         if(protocol in packet.summary()):
29
30             size = len(packet)
31             #print(packet.summary(),size)
32             # print(size)
33             src = packet[IP].src
34             dst = packet[IP].dst
35             ts = packet.time
36             # print(src)
```

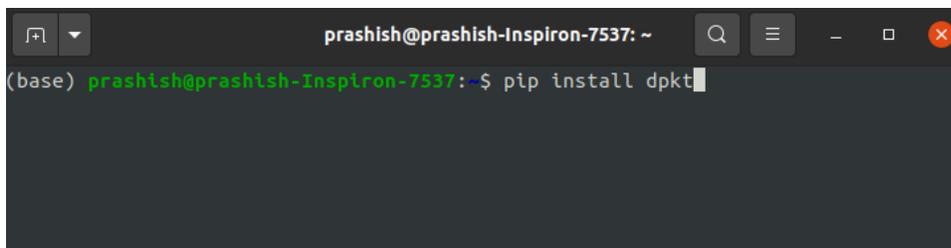
13. Install required Python modules

Install required python modules using following command.

E.g., To install **dpkt** and **Scapy**. Go to terminal and run following commands.

Pip install dpkt

Pip install scapy



```
prashish@prashish-Inspiron-7537: ~
(base) prashish@prashish-Inspiron-7537:~$ pip install dpkt
```

```
prashish@prashish-Inspiron-7537: ~  
(base) prashish@prashish-Inspiron-7537:~$ pip install scapy
```

14. Run pcapReader.py

Inside pcapReader.py

Set **src** to ip address of source machine.

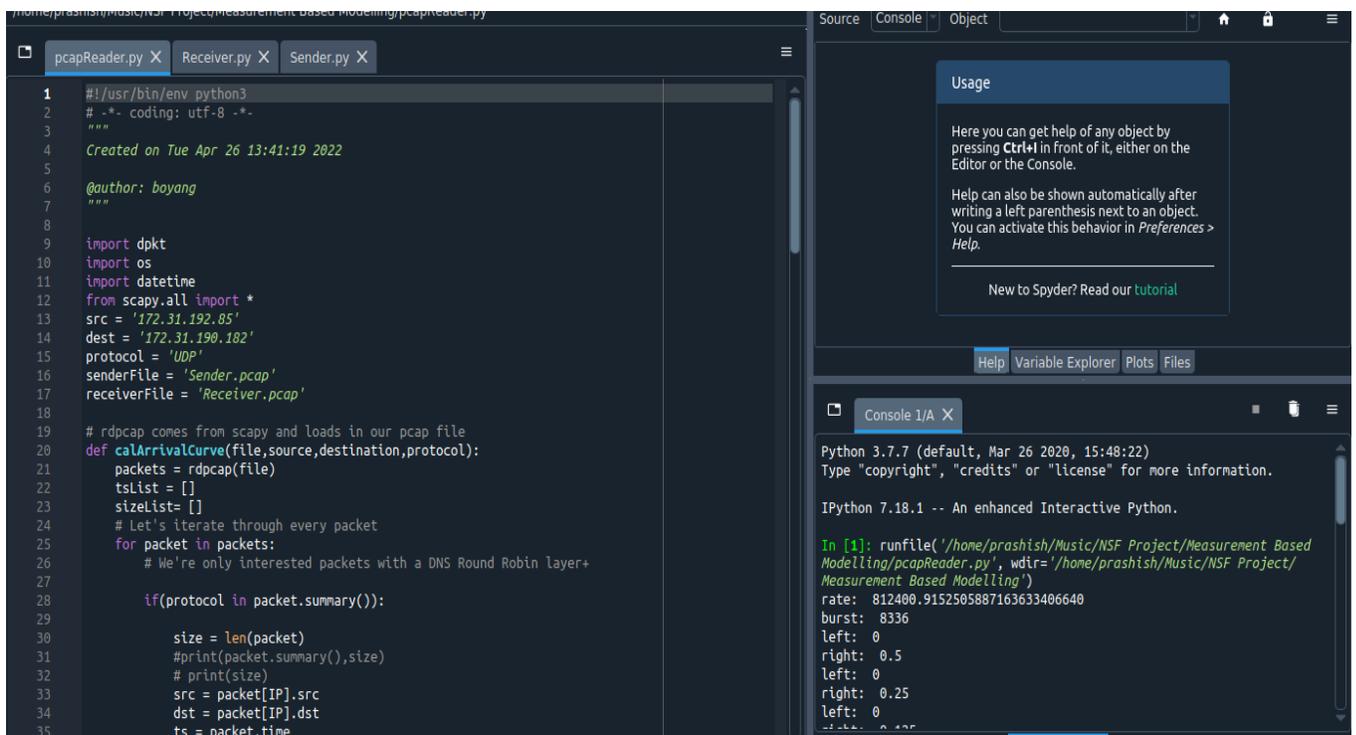
Set **dest** to ip address of destination machine.

Set **senderFile** to name of pcap file downloaded from the source machine.

Set **receiverFile** to name of pcap file downloaded from the destination machine.

Then,

RUN pcapReader.py.



```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
Created on Tue Apr 26 13:41:19 2022  
@author: boyang  
import dpkt  
import os  
import datetime  
from scapy.all import *  
src = '172.31.192.85'  
dest = '172.31.190.182'  
protocol = 'UDP'  
senderFile = 'Sender.pcap'  
receiverFile = 'Receiver.pcap'  
  
# rdpcap comes from scapy and loads in our pcap file  
def calArrivalCurve(file,source,destination,protocol):  
    packets = rdpcap(file)  
    tslist = []  
    sizeList= []  
    # Let's iterate through every packet  
    for packet in packets:  
        # We're only interested packets with a DNS Round Robin layer+  
        if(protocol in packet.summary()):  
            size = len(packet)  
            #print(packet.summary(),size)  
            # print(size)  
            src = packet[IP].src  
            dst = packet[IP].dst  
            ts = packet.time
```

```
Python 3.7.7 (default, Mar 26 2020, 15:48:22)  
Type "copyright", "credits" or "license" for more information.  
  
IPython 7.18.1 -- An enhanced Interactive Python.  
  
In [1]: runfile('/home/prashish/Music/NSF Project/Measurement Based  
Modelling/pcapReader.py', wdir='/home/prashish/Music/NSF Project/  
Measurement Based Modelling')  
rate: 812400.9152505887163633406640  
burst: 8336  
left: 0  
right: 0.5  
left: 0  
right: 0.25  
left: 0  
right: 0.125
```

The result will display rate and burst of the arrival curve.

```
IPython 7.18.1 -- An enhanced Interactive Python.
```

```
In [1]: runfile('/home/prashish/Music/NSF Project/Measurement Based  
Modelling/pcapReader.py', wdir='/home/prashish/Music/NSF Project/  
Measurement Based Modelling')  
rate: 812400.9152505887163633406640  
burst: 8336  
left: 0  
right: 0.5  
left: 0  
right: 0.25  
left: 0
```

Similarly, the last line of results displays the delay i.e latency.

```
right: 0.0107421875  
left: 0.0106201171875  
right: 0.0107421875  
left: 0.0106201171875  
right: 0.01068115234375  
latency: 0.01068115234375
```