ELEC-E7130 - Internet Traffic Measurements and Analysis

Assignment 1 Learning to use tools for measurement and analysis

Haibi Peng 875552

Task 1: Measuring latency

Solution:

1. Measurement setup:

	name server	research server	iperf server
1	a.cctld.us	pna-es.ark.caida.org	ok1.iperf.comnet-student.eu
2	b.cctld.us	per-au.ark.caida.org	blr1.iperf.comnet-student.eu
3	c.cctld.us	cjj-kr.ark.caida.org	N/A
Measurement type	DNS query/ICMP echo request	5 ICMP echo requests	5 ICMP echo requests/TCP connect latency test
Frequency	Once an hour	Every 10 minitues	Every 10 minitues
Configuration	Sending minute: 875552%60=32 Add time stamp to ping using -O and -D options	Sending minute: 875552%10=2	Sending minute:875552%10=2 Randomly choose port for each test in iperf; Request the 1K.bin file; *curl -w "%{time_total},%{speed_download}"*

Mycountry tool:

```
pengh1@vdiubuntu040 ~ % bash
pengh1@vdiubuntu040:~$ source /work/courses/unix/T/ELEC/E7130/general/use.sh
pengh1@vdiubuntu040:~$ mycountry
us OK (United States): b.cctld.us, e.cctld.us, a.cctld.us, c.cctld.us, k.cctld.us, f.cctld.us
Your UID is 3180588, thus your ccTLD is us (United States)
```

2. Scripts for each server and crontab settings

✧ nameserver

```
#!/bin/bash

d=$(date -Isec | tr -d : | sed s/+.*//)

dig @8.8.8.8 a.cctld.us >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/nameserver1/Latency/ns1-ltc-$d.txt &
ping -c 1 a.cctld.us -O -D >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/nameserver1/ICMP/ns1-icpm-$d.txt &
dig @8.8.8.8 b.cctld.us >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/nameserver2/Latency/ns2-ltc-$d.txt &
ping -c 1 b.cctld.us -O -D >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/nameserver2/ICMP/ns2-icpm-$d.txt &
dig @8.8.8.8 c.cctld.us >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/nameserver3/Latency/ns3-ltc-$d.txt &
ping -c 1 c.cctld.us -O -D >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/nameserver3/ICMP/ns3-icpm-$d.txt
```

✧ research server

```
#!/bin/bash

d=$(date -Isec | tr -d : | sed s/+.*//)

ping -c 5 pna-es.ark.caida.org -O -D >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/researchserver1/rs1-icpm-$d.txt &
ping -c 5 per-au.ark.caida.org -O -D >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/researchserver2/rs2-icpm-$d.txt &
ping -c 5 cjj-kr.ark.caida.org -O -D >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/researchserver3/rs3-icpm-$d.txt
```

✧ iperf server

```
#!/bin/bash

d=$(date -Isec | tr -d : | sed s/+.*//)

curl -o /dev/null http://ok1.iperf.comnet-student.eu/1K.bin -w "%{time_total},%{speed_download}" >> /u/88/pengh1/unix/Desktop/Assign
ping -c 5 ok1.iperf.comnet-student.eu >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/iperfserver1/ICMP/is1-icpm-$d.txt &
curl -o /dev/null http://blr1.iperf.comnet-student.eu/1K.bin -w "%{time_total},%{speed_download}" >> /u/88/pengh1/unix/Desktop/Assig
ping -c 5 blr1.iperf.comnet-student.eu >> /u/88/pengh1/unix/Desktop/Assignment2/Task1/latencytest/iperfserver2/ICMP/is2-icpm-$d.txt
```

✧ crontab settings

```
SHELL=/bin/bash

32 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver.sh >> nameserver.log 2>&1

2,12,22,32,42,52 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task1/researchserver.sh >> researchserver.log 2>&1

2,12,22,32,42,52 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task1/iperfserver.sh >> iperfserver.log 2>&1

2 * * * * /bin/bash /u/88/pengh1/unix/Desktop/Assignment2/Task2/iperf3-send.sh >> iperfup.log 2>&1

32 * * * * /bin/bash /u/88/pengh1/unix/Desktop/Assignment2/Task2/iperf3-receive.sh >> iperfdn.log 2>&1

1 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task2/curl.sh >> curl.log 2>&1
```

3. Table of measurement results.

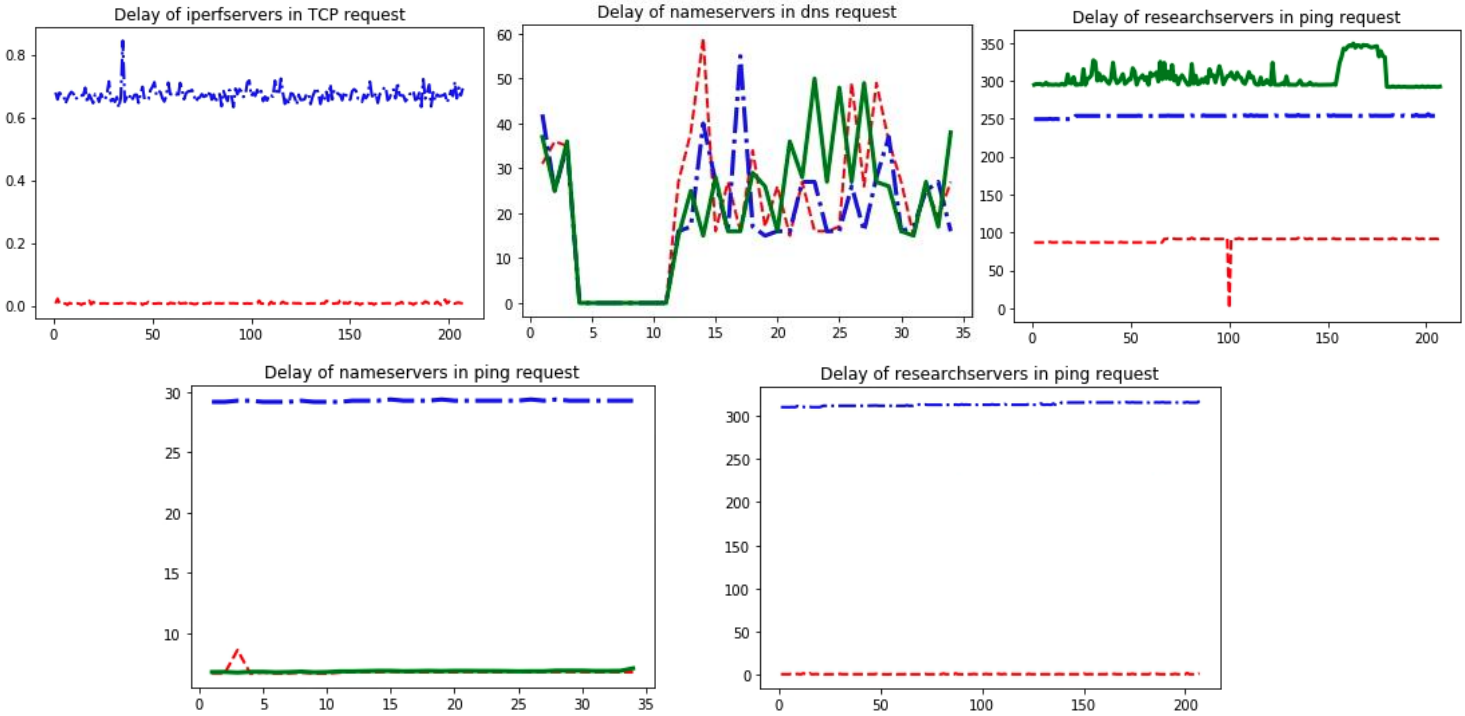
	A	B	C	D	E	F
1		Median delay	Mean delay	Loss ratio	Delay spead with 75th percentile	Delay spead with 25th percentile
2	nameserver1/ping	6.75	6.791	0	6.7675	6.7225
3	nameserver2/ping	29.3	29.288	0	29.3	29.3
4	nameserver3/ping	6.84	6.829	0	6.86	6.785
5	nameserver1/dns	26	26	0.23529	34.75	17
6	nameserver2/dns	16	24.269	0.23529	27	16
7	nameserver3/dns	25	27.5	0.23529	34.25	16.25
8	researchserver1/ping	91.379	90.096	0.00483	91.42925	87.027
9	researchserver2/ping	253.773	253.422	0	253.819	253.7135
10	researchserver3/ping	297.751	304.707	0	307.1995	294.637
11	iperfserver1/ping	0.733	0.825	0	0.766	0.704
12	iperfserver2/ping	312.871	313.267	0	315.3845	311.709
13	iperfserver1/TCP	0.007344	0.008	0	0.00801	0.00682
14	iperfserver1/tcp	0.670855	0.672	0	0.68002	0.65877

4. Conclusions on network stability

a. Was some of hosts different from the others?

Based on the table of measurement results, we can see that among the same kind of server, there are some differences between them. For example, name server 2 is apparently different from the other two name servers, since its delays(both ping&dns) are not in line with the other two's(ping request 29.3 ms vs 6.8 ms, dns request 16 ms vs 25 ms), while the other two's data are very close to each other. For research servers, research server 1 seems to have smaller delays than the other two, and name server 2 and 3 seems to have closer delays. And for iperf servers, the differences are easy to observe, since the delays of each server are two orders of magnitude difference.

b. Could you observe any day-time variations?



Based on the graphs about, we can see among name servers, ping request remained stable during the requesting period, while dns request had seen an dramatic drop during the morning (probably because server failure or other reasons) and also had considerable fluctuations in rest of the time. And for research servers, two of them (1 and 2) remained stable during the requesting period, while the last one(3) had experienced an increase in the delay during the midnight and the early morning of the second day measurement. As for iperf servers, in TCP request, iperf server 2 fluctuated in delay slightly but some big changes might occur sometimes, while iperf server 1 remained stable. In ping request, both servers remained stable all the time.

c. Do the timezones where target servers (or you) have an impact?
I believe timezones of the target servers will have an impact on the delays, since the timezones of three research servers are different from each other(Spanish-UTC1, Australia-UTC8/9.5/10, Korea-UTC9). And I guess my timezone(client) also will influence the data.

Task 2: Measuring throughput

Solution:

1. Measurement setup:

	By file transfer	By special measurement tool	By using measurement service
Tools	HTTP download tool-***curl***	***iperf3***	Speed Test
Frequency	Once an hour	Once an hour	Few times by hand
Configuration	*curl -w fmt="%{time_total}, %{speed_download}, %{size_download}"* Choose 10M.bin file	Randomly choose port for each test in iperf; Option *-t 10* for 10 seconds Option *-R* set client as	N/A

	Use *curl -m secs* to set maximum time(10s)	reciever	

2. Scripts for each server and crontab settings

✧ HTTP request

```
#!/bin/bash

d=$(date -Isec | tr -d : | sed s/+.*//)
fmt="%{time_total}, %{speed_download}, %{size_download}"

curl -w "$fmt" -o /dev/null http://ok1.iperf.comnet-student.eu/10M.bin -m 10 >>
curl -w "$fmt" -o /dev/null http://blr1.iperf.comnet-student.eu/10M.bin -m 10 >>
```

✧ iperf request

```
#!/bin/bash

function rand(){
    min=$1
    max=$(($2-$min+1))
    num=$(date +%s%N)
    echo $(($num%$max+$min))
}

d=$(date -Isec | tr -d : | sed s/+.*//)
port=$(rand 5200 5210)
iperf3 -c ok1.iperf.comnet-student.eu -t 10 -p $port  >>
iperf3 -c blr1.iperf.comnet-student.eu -t 10 -p $port  >>
```

```
#!/bin/bash

function rand(){
    min=$1
    max=$(($2-$min+1))
    num=$(date +%s%N)
    echo $(($num%$max+$min))
}

d=$(date -Isec | tr -d : | sed s/+.*//)
port=$(rand 5200 5210)
iperf3 -c ok1.iperf.comnet-student.eu -t 10 -p $port -R >>
iperf3 -c blr1.iperf.comnet-student.eu -t 10 -p $port -R >>
```

✧ crontab settings

```
SHELL=/bin/bash

32 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver.sh >> nameserver.log 2>&1

2,12,22,32,42,52 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task1/researchserver.sh >> researchserver.log 2>&1

2,12,22,32,42,52 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task1/iperfserver.sh >> iperfserver.log 2>&1

2 * * * * /bin/bash /u/88/pengh1/unix/Desktop/Assignment2/Task2/iperf3-send.sh >> iperfup.log 2>&1

32 * * * * /bin/bash /u/88/pengh1/unix/Desktop/Assignment2/Task2/iperf3-receive.sh >> iperfdn.log 2>&1

1 * * * * /bin/sh /u/88/pengh1/unix/Desktop/Assignment2/Task2/curl.sh >> curl.log 2>&1
```

3. Table of measurement results.

	iperfserver1/HTTP	iperfserver2/HTTP	iperfserver1/iperf up	iperfserver1/iperf dn	iperfserver2/iperf up	iperfserver2/iperf dn	ST up	ST dn
Median	190.65	2.278	7940	9170	55.8	47	374.64	266.5
Mean	193.571	2.3	7806.667	9101.111	44.645	43.254	381.067	273.413
Max	262.144	2.688	8290	9280	61	54.9	439.61	322.88
Min	79.438	1.83	6540	8480	2.6	6.47	328.95	230.86
Avg deviation	31.0500911143	0.155426142857	324.444333333	126.666703704	15.023	7.96472727273	39.029	32.977

Note: column 7,8 is the calculated throughput results of 3 tests by using Speed Test.

4. Conclusions on network stability

a. Are the results between methods in line with each other?

It seems the results between methods are not in line with each other, at least in my case. There are quite big differences between methods and even iperf server 1 and 2.

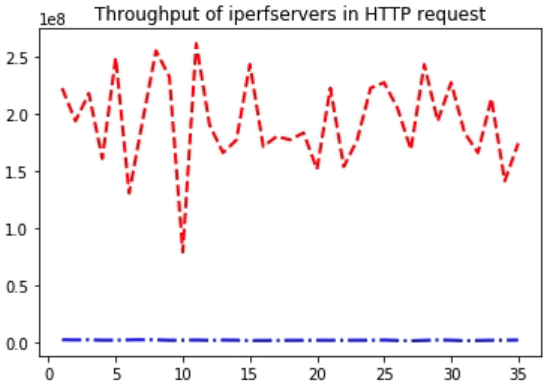
b. Did some method have lot of deviation? What do think might cause this?

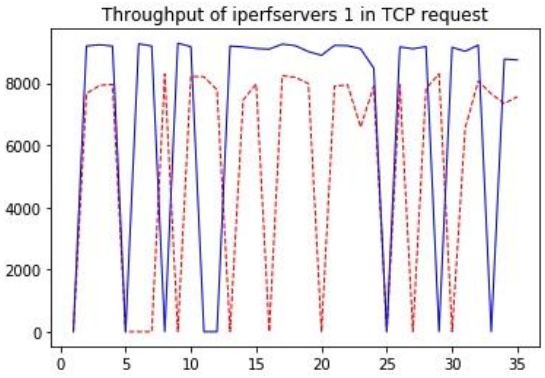
Among the cases of my measurement, the HTTP request of iperf server 1(column 2) and the uplink/downlink iperf request of iperf server 2(column 6, 7) had major deviation, compared to their mean values. To my best knowledge it may be because of the properties of the iperf server 2, which also performed not good in the HTTP request.

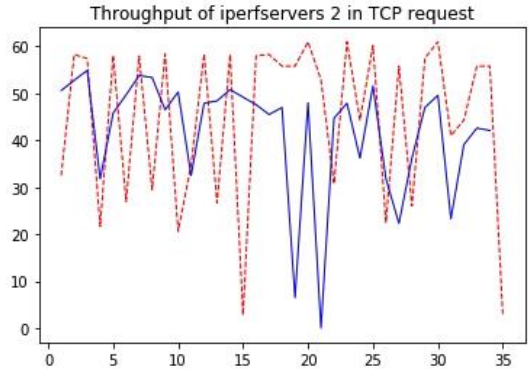
c. Was there some method that gives higher values than other? What do you think might cause this?

It can be easily observed that in the second method by special measurement tool ***iperf3*** used against iperf server 1, the results were apparently higher than others', since the bitrate reached Gbits/s. I think it might result from the properties of the iperf server 1, which also performed better in the HTTP method than iperf server 2. Besides, HTTP is on a layer above TCP. The question is really about how much overhead the stuff above TCP adds. HTTP is relatively chunky because each transmission requires a bunch of header cruft in both the request and the response. So I guess lower layers works faster than upper layers is because there is less layers need to access when doing data transfers between two computers.

d. Is there variation due time? For example did you get higher throughput during day or night?







Based on the graphs above, we can see frequent fluctuations in the throughput in both servers during the request period, except for the HTTP request for iperf server 2. However, we can basically observe that in the second graph the downlink throughput performed well and relatively stable from the afternoon(15:00) of the first day to the midnight(4:00) of the second day, and in the third graph the uplink throughput performed similarly while lasting time was shorter. So basically we may say that the throughput is higher during the night.

e. Was there are anomalies? For example, no connection or very different capacity.

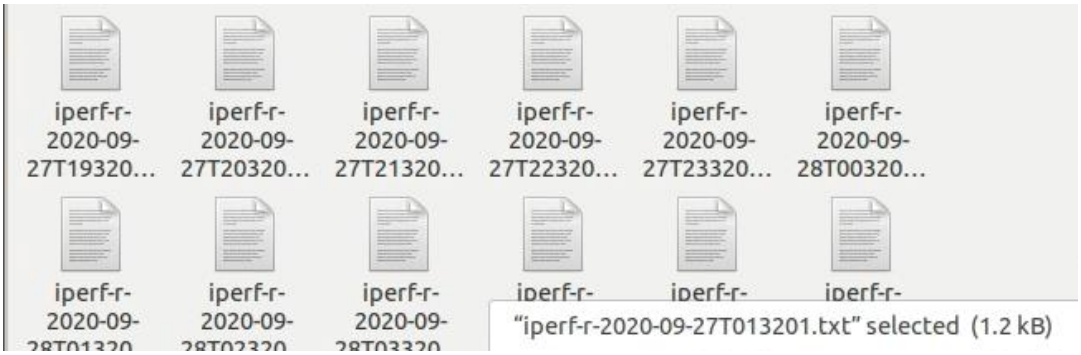
Apparently, there were some anomalies like no connection (output files were blank) and failed connection (time out), which resulted in throughput of 0.

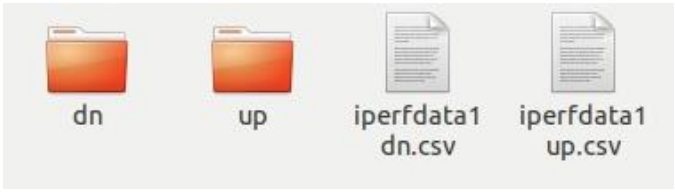
Task 3: Producing data files

Solution:

1. Describe your data model and file format.

Original files were .txt and I use Python scripts to grab needed data and write them into .csv files.





2. Describe program to generate data.

First, I read the .txt files and put their address into a list:

```
#-------------------------------------Get data--
def readfiles(DIRCT):
    dirct = DIRCT
    dirList=[]
    files=os.listdir(dirct)
    for f in files:
        if os.path.isfile(dirct + '/'+f):
            dirList.append(dirct+'/'+f)
    #print(dirList)
    return dirList
```

Then I use different ***def function():*** to grab the information that are needed, regarding different methods by using ***regular expression***(***re*** library in Python), for example:

```
#get HTTP data of Task2
def getHTTPdata(dirList):
    HTTPinfo=[]
    temp=[]
    for f in dirList[0:]:
        if len(open(f).readlines()) == 0 :
            failed = 'True'
            temp=[None, None, None, 0, 0, 0, failed]
        else:
            failed = 'False'
            line=open(f).readlines()[0].split(', ')
            timeelapsed=line[0]
            bitrate=line[1]
            bytestransfered=line[2]
            timestamp=re.findall('-(.*?).txt', f)[0]
            typeofmeasurement='HTTP'
            if re.findall(re.compile(r'curl(.{1})'), f)[1]=='1':
                target='iperf.netlab.hut.fi (195.148.124.36)'
            if re.findall(re.compile(r'curl(.{1})'), f)[1]=='2':
                target='blr1.iperf.comnet-student.eu (142.93.213.224)'
            temp=[timestamp, typeofmeasurement, target, bitrate, timeelapsed,
                  bytestransfered, failed]
        HTTPinfo.append(temp)
    #print(ICMPinfo)
    return HTTPinfo
```

```
#get iperf data of Task2
def getiperfdata(dirList):
    iperfinfo=[]
    temp=[]
    for f in dirList[0:]:
        if len(open(f).readlines()) < 5 :
            failed = 'True'
            temp=[None, None, None, 0, 0, 0, failed]
        else:
            failed = 'False'
            timeelapsed=10
            if re.findall('iperf-(.*?)-', f)[0]=='r':
                bitrate=re.findall('Bytes  (.*?)  ', open(f).read())[-1]
                bytestransfered=re.findall('  sec  (.*?)  ', open(f).read())[-1]
            else:
                bitrate=re.findall('Bytes  (.*?)  ', open(f).read())[-2]
                bytestransfered=re.findall('  sec  (.*?)  ', open(f).read())[-2]
            timestamp=re.findall('-(.*?).txt', f)[0][2:]
            typeofmeasurement='iperf'
            if re.findall(re.compile(r'iperf(.{1})'), f)[1]=='1':
                target='iperf.netlab.hut.fi (195.148.124.36)'
            if re.findall(re.compile(r'iperf(.{1})'), f)[1]=='2':
                target='blr1.iperf.comnet-student.eu (142.93.213.224)'
            temp=[timestamp, typeofmeasurement, target, bitrate, timeelapsed,
                  bytestransfered, failed]
        iperfinfo.append(temp)
    #print(ICMPinfo)
    return iperfinfo
```

Finally, using ***csv*** library to write data into .csv files:

```
#Write into .csv files
def writeCSVfile(filename, firstline, info):
    with open(filename,"w") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(firstline)
        writer.writerows(info)
```

And what is next is calculation part, for exmaple:

```
#-----------------------------------Calculation-----------------------------
#Median delay with lost packets with delay of infinity, thus if more than 50%
#of packets are lost, then consider as infinity.
def mediandelay(dataList):
    ltc=[]
    for i in dataList:
        ltc.append(float(i[3]))
    LTC=sorted(ltc, key=float)
    med=LTC[(len(LTC))/2]
    return med

#Mean delay with lost packets not counted.
def meandelay(dataList):
    ltc=[]
    for i in dataList:
        if i[3]!=0:
            ltc.append(float(i[3]))
        meanltc=round(sum(ltc)/len(ltc), 3)
    return meanltc
```

```
#Loss ratio
def lossratio(dataList):
    loss=0
    for i in dataList:
        if i[3]==0:
            loss+=1
    lossratio=round(float(loss)/float(len(dataList)), 5)
    return lossratio

#Delay spread as difference with 75th and 25th percentiles
def percentile(dataList, percentile):
    ltc=[]
    for i in dataList:
        if i[3]!=0:
            ltc.append(float(i[3]))
    xth=round(np.percentile(ltc, percentile), 5)
    return xth
```

```
def calthroughput(dataList):
    data=[]
    for i in dataList:
        if i[3]==0:
            #data.append(float(i[3]))
            continue
        elif len(i[3].split(' '))==2:
            data.append(float(i[3].split(' ')[0]))
        else:
            data.append(float(i[3])/1000000)
    thpt=sorted(data, key=float)
    med=round(thpt[len(thpt)/2], 3)
    mean=round(sum(data)/len(data), 3)
    Max=round(max(data), 3)
    Min=round(min(data), 3)
    data[:]=[x - mean for x in data]
    Avgdev= sum(data)/float(len(data))
    results=[med, mean, Max, Min, Avgdev]
    return results
```

So basically the process went like this:

```
#read files
ICMPfileList1 = readfiles('/u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver1/ICMP')
ICMPfileList2 = readfiles('/u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver2/ICMP')
ICMPfileList3 = readfiles('/u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver3/ICMP')

#get data
ICMPinfo1 = getICMPdata(ICMPfileList1)
ICMPinfo2 = getICMPdata(ICMPfileList2)
ICMPinfo3 = getICMPdata(ICMPfileList3)

#write .csv file
writeCSVfile('/u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver1/ICMPdata1.csv', latencyitems, ICMPinfo1)
writeCSVfile('/u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver2/ICMPdata2.csv', latencyitems, ICMPinfo2)
writeCSVfile('/u/88/pengh1/unix/Desktop/Assignment2/Task1/nameserver3/ICMPdata3.csv', latencyitems, ICMPinfo3)
```

```
        #-------------------------------------Result calculation-------------------------------------
        #-------------------------------------Taske 1------------------------------------------------
        #Median delay with lost packets with delay of infinity, thus if more than 50%
        #of packets are lost, then consider as infinity.

        mddl = [mediandelay(ICMPinfo1), mediandelay(ICMPinfo2), mediandelay(ICMPinfo3),
                mediandelay(DNSinfo1), mediandelay(DNSinfo2), mediandelay(DNSinfo3),
                mediandelay(RICMPinfo1), mediandelay(RICMPinfo2), mediandelay(RICMPinfo3),
                mediandelay(IICMPinfo1), mediandelay(IICMPinfo2),
                mediandelay(TCPinfo1), mediandelay(TCPinfo2)]
        #print(mddl)

        #Mean delay with lost packets not counted.
        mndl = [meandelay(ICMPinfo1), meandelay(ICMPinfo2), meandelay(ICMPinfo3),
                mediandelay(DNSinfo1), meandelay(DNSinfo2), meandelay(DNSinfo3),
                meandelay(RICMPinfo1), meandelay(RICMPinfo2), meandelay(RICMPinfo3),
                meandelay(IICMPinfo1), meandelay(IICMPinfo2),
                meandelay(TCPinfo1), meandelay(TCPinfo2)]
        #print(mndl)

        #Loss ratio
        lssrt = [lossratio(ICMPinfo1), lossratio(ICMPinfo2), lossratio(ICMPinfo3),
                lossratio(DNSinfo1), lossratio(DNSinfo2), lossratio(DNSinfo3),
                lossratio(RICMPinfo1), lossratio(RICMPinfo2), lossratio(RICMPinfo3),
                lossratio(IICMPinfo1), lossratio(IICMPinfo2),
                lossratio(TCPinfo1), lossratio(TCPinfo2)]
        #print(lssrt)

        #Delay spread as difference with 75th and 25th percentiles
        percentile75 = [percentile(ICMPinfo1, 75), percentile(ICMPinfo2, 75), percentile(ICMPinfo3, 75),
                percentile(DNSinfo1, 75), percentile(DNSinfo2, 75), percentile(DNSinfo3, 75),
                percentile(RICMPinfo1, 75), percentile(RICMPinfo2, 75), percentile(RICMPinfo3, 75),
                percentile(IICMPinfo1, 75), percentile(IICMPinfo2, 75),
                percentile(TCPinfo1, 75), percentile(TCPinfo2, 75)]
        #print(percentile75)

        percentile25 = [percentile(ICMPinfo1, 25), percentile(ICMPinfo2, 25), percentile(ICMPinfo3, 25),
                percentile(DNSinfo1, 25), percentile(DNSinfo2, 25), percentile(DNSinfo3, 25),
                percentile(RICMPinfo1, 25), percentile(RICMPinfo2, 25), percentile(RICMPinfo3, 25),
                percentile(IICMPinfo1, 25), percentile(IICMPinfo2, 25),
                percentile(TCPinfo1, 25), percentile(TCPinfo2, 25)]
        #print(percentile25)
```

And to produce final .csv files with calculation results:

```
#produce final .csv
serversrequests=['nameserver1/ping', 'nameserver2/ping', 'nameserver3/ping',
                 'nameserver1/dns', 'nameserver2/dns', 'nameserver3/dns',
                 'researchserver1/ping', 'researchserver2/ping', 'researchserver3/ping',
                 'iperfserver1/ping', 'iperfserver2/ping', 'iperfserver1/TCP', 'iperfserver1/tcp']

calname1=['Median delay', 'Mean delay', 'Loss ratio',
                'Delay spead with 75th percentile', 'Delay spead with 25th percentile']

calresultsltc=[mddl, mndl, lssrt, percentile75, percentile25]

data = dict(zip(calname1, calresultsltc))
dataframe = pd.DataFrame(data,index = serversrequests)
dataframe.to_csv(r'/u/88/pengh1/unix/Desktop/Assignment2/Task1/Task1.csv',columns = calname1)
```

3. Sample of results file.

Delay measurement: ICMP echo request for name server 1

	A	B	C	D	E
1	timestamp	typeofmeasurement	target	delay	failed
2	[1601159521.531663]	ICMP echo request	a.cctld.us(156.154.124.70)	6.67	False
3	[1601163121.453273]	ICMP echo request	a.cctld.us(156.154.124.70)	6.67	False
4	[1601166721.166117]	ICMP echo request	a.cctld.us(156.154.124.70)	8.59	False
5	[1601170321.659530]	ICMP echo request	a.cctld.us(156.154.124.70)	6.64	False
6	[1601173922.035444]	ICMP echo request	a.cctld.us(156.154.124.70)	6.73	False
7	[1601177521.657790]	ICMP echo request	a.cctld.us(156.154.124.70)	6.65	False
8	[1601181121.376502]	ICMP echo request	a.cctld.us(156.154.124.70)	6.66	False
9	[1601184722.068659]	ICMP echo request	a.cctld.us(156.154.124.70)	6.72	False
10	[1601188321.458477]	ICMP echo request	a.cctld.us(156.154.124.70)	6.65	False
11	[1601191922.083196]	ICMP echo request	a.cctld.us(156.154.124.70)	6.64	False
12	[1601195522.069051]	ICMP echo request	a.cctld.us(156.154.124.70)	6.72	False
13	[1601199121.824563]	ICMP echo request	a.cctld.us(156.154.124.70)	6.77	False
14	[1601202721.825880]	ICMP echo request	a.cctld.us(156.154.124.70)	6.77	False
15	[1601206321.178251]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False
16	[1601209921.360053]	ICMP echo request	a.cctld.us(156.154.124.70)	6.79	False
17	[1601213521.136389]	ICMP echo request	a.cctld.us(156.154.124.70)	6.76	False
18	[1601217121.964139]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False
19	[1601220721.618900]	ICMP echo request	a.cctld.us(156.154.124.70)	6.74	False
20	[1601224321.144720]	ICMP echo request	a.cctld.us(156.154.124.70)	6.76	False
21	[1601227921.723900]	ICMP echo request	a.cctld.us(156.154.124.70)	6.74	False
22	[1601231521.214419]	ICMP echo request	a.cctld.us(156.154.124.70)	6.77	False
23	[1601235121.562400]	ICMP echo request	a.cctld.us(156.154.124.70)	6.76	False
24	[1601238722.171992]	ICMP echo request	a.cctld.us(156.154.124.70)	6.76	False
25	[1601242321.645330]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False
26	[1601245921.888587]	ICMP echo request	a.cctld.us(156.154.124.70)	6.83	False
27	[1601249521.852693]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False
28	[1601253121.149514]	ICMP echo request	a.cctld.us(156.154.124.70)	6.78	False
29	[1601256721.714617]	ICMP echo request	a.cctld.us(156.154.124.70)	6.76	False
30	[1601260321.374895]	ICMP echo request	a.cctld.us(156.154.124.70)	6.77	False
31	[1601263921.599674]	ICMP echo request	a.cctld.us(156.154.124.70)	6.76	False
32	[1601267521.874926]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False
33	[1601271121.780960]	ICMP echo request	a.cctld.us(156.154.124.70)	6.77	False
34	[1601274721.251000]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False
35	[1601278321.735607]	ICMP echo request	a.cctld.us(156.154.124.70)	6.75	False

Throughput measurement: iperf request for iperf server 1:

	A	B	C	D	E	F	G
1	timestamp	typeofmeasurement	target	bitrate	timeelapsed	bytestransfered	failed
2				0	0	0	True
3	2020-09-27T013201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.19 Gbits/sec	10	10.7 GBytes	False
4	2020-09-27T023201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.23 Gbits/sec	10	10.7 GBytes	False
5	2020-09-27T033201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.19 Gbits/sec	10	10.7 GBytes	False
6				0	0	0	True
7	2020-09-27T053202	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.26 Gbits/sec	10	10.8 GBytes	False
8	2020-09-27T063201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.19 Gbits/sec	10	10.7 GBytes	False
9				0	0	0	True
10	2020-09-27T083202	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.28 Gbits/sec	10	10.8 GBytes	False
11	2020-09-27T093201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.17 Gbits/sec	10	10.7 GBytes	False
12				0	0	0	True
13				0	0	0	True
14	2020-09-27T123201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.19 Gbits/sec	10	10.7 GBytes	False
15	2020-09-27T133201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.16 Gbits/sec	10	10.7 GBytes	False
16	2020-09-27T143201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.11 Gbits/sec	10	10.6 GBytes	False
17	2020-09-27T153201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.09 Gbits/sec	10	10.6 GBytes	False
18	2020-09-27T163201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.25 Gbits/sec	10	10.8 GBytes	False
19	2020-09-27T173201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.20 Gbits/sec	10	10.7 GBytes	False
20	2020-09-27T183201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.01 Gbits/sec	10	10.5 GBytes	False
21	2020-09-27T193201	iperf	iperf.netlab.hut.fi (195.148.124.36)	8.89 Gbits/sec	10	10.4 GBytes	False
22	2020-09-27T203201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.21 Gbits/sec	10	10.7 GBytes	False
23	2020-09-27T213201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.20 Gbits/sec	10	10.7 GBytes	False
24	2020-09-27T223201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.10 Gbits/sec	10	10.6 GBytes	False
25	2020-09-27T233202	iperf	iperf.netlab.hut.fi (195.148.124.36)	8.48 Gbits/sec	10	9.87 GBytes	False
26				0	0	0	True
27	2020-09-28T013201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.16 Gbits/sec	10	10.7 GBytes	False
28	2020-09-28T023201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.10 Gbits/sec	10	10.6 GBytes	False
29	2020-09-28T033201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.17 Gbits/sec	10	10.7 GBytes	False
30				0	0	0	True
31	2020-09-28T053201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.15 Gbits/sec	10	10.7 GBytes	False
32	2020-09-28T063201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.02 Gbits/sec	10	10.5 GBytes	False
33	2020-09-28T073201	iperf	iperf.netlab.hut.fi (195.148.124.36)	9.22 Gbits/sec	10	10.7 GBytes	False
34				0	0	0	True
35	2020-09-28T093201	iperf	iperf.netlab.hut.fi (195.148.124.36)	8.77 Gbits/sec	10	10.2 GBytes	False
36	2020-09-28T103201	iperf	iperf.netlab.hut.fi (195.148.124.36)	8.74 Gbits/sec	10	10.2 GBytes	False