ELEC-E7130 Assignment 2

Task 1: Measuring latency

Requirement (task 1)

In this task you will collect latency data from the Internet and create a report from it. The report should include description of measurements, summary of the results and conclusions based on the results.

Choose 3 name servers, 3 research servers and 2 iperf servers based on instructions at the last section of this exercise. Start measurements and collect data for at least 24 hours. Leave measurements running for a minimum of two weeks as this data will be used in the Final Assignment.

- For the name servers set, measure latency once an hour both with DNS query and ICMP echo request (check -O and -D options!). Send one of each request per hour. Use student id modulo 60 to select the right minute to send.
- For the research server set, run the test every 10 minute by sending 5 ICMP echo requests. Select minute by studentid modulo 10.
- For the Iperf servers set, run a test every 10 minute by sending 5 ICMP echo requests and with TCP connect latency test. Request the 1K.bin file, for example, not to cause too much load. Select minute by studentid modulo 10.

TIPS: Create three shell scripts, one for research server measurements, one for iperf servers and one for name servers. Run all of them with cron. You may log all output and make filtering and data collection later or you can do filtering right away. Import CSV to

Libreoffice/Excel/Google spreadsheet and calculate the values. Or if you want to get hands on R or python, we recommend taking the dive already now.

When multiple measurements over a period of 24 hours are requested, measurements should be more or less evenly spaced: not like 10 measurements in 10 minutes and 24 hours later 2 more. Using cron makes this easy. It is recommended not to run all tests at the start of the hour but distribute it further. A recommended way to get evenly distributed (among students) value for minutes is to run expr $(id -u) % 601. Adapt accordingly for more frequent measurements.

For the report describe your measurement setup.

It's possible to calculate the results by hand from the logs as there are only a low number of measurements but in the future we will have a bigger dataset that is tedious or impossible to go through by hand.

Report a table with the following metrics for each of your target servers.

- Median delay with lost packets with delay of infinity, thus if more than 50 % of packets are lost, then consider as infinity.
- Mean delay with lost packets not counted.
- Loss ratio.
- Delays spread as a difference with 75th and 25th percentiles. If more than 25 % of packets are lost, then consider infinity.

Finally, make conclusions about stability of network delay. Were some of the hosts different from the others? Could you observe any day-time variations? Do the timezones where target servers (or you) have an impact?

Report (task 1)

1 Describe your measurement setup

1.1 Confirm server lists and sending time

My student id is "902166"

```
yanj3@force ~% echo $((902166 % 3))
0
yanj3@force ~% echo $((902166 % 6))
0
yanj3@force ~% echo $((902166 / 7 % 6))
0
```

My research server list is: {pna-es.ark.caida.org, hlz-nz.ark.caida.org}
My iperf server list is: {ok1.iperf.comnet-student.eu, blr1.iperf.comnet-student.eu}
My name server list is: {d.dns.br, e.dns.br, f.dns.br, a.dns.br, c.dns.br, b.dns.br}

```
yanj3@force ~% echo $((902166 % 60))
6
yanj3@force ~% echo $((902166 % 10))
6
```

The crontab setting should be the 6th minute per hour for the name server scripts.

The crontab setting should be the 6th minute per ten minutes for the research server and iperf server scripts.

1.2 Create shell scripts

Create a shell script for name server set related testing

```
yanj3@force ~ % cat task1/task1-1.sh
for name_server in d.dns.br e.dns.br f.dns.br a.dns.br c.dns.br b.dns.br
do
date
dig @8.8.8.8 $name_server
ping $name_server -D -O -c1
done
```

Create a shell script for research server set related testing

```
yanj3@force ~ % cat task1/task1-2.sh
for research_server in pna-es.ark.caida.org hlz-nz.ark.caida.org
do
date
ping $research_server -c5
done
```

Create a shell script for iperf server set related testing

```
yanj3@force ~ % cat task1/task1-3.sh
for iperf_server in ok1.iperf.comnet-student.eu blr1.iperf.comnet-student.eu
do
date
ping $iperf_server -c5
echo "curl iperf_server is $iperf_server"
curl -o /dev/null http://$iperf_server/1K.bin -w "%{time_connect}\n"
done
```

1.3 Crontab setting for shell scripts

```
yanj3@force ~/task1 % crontab -l
# task1
6 * * * * /u/94/yanj3/unix/task1/task1-1.sh >> /u/94/yanj3/unix/task1/task1-1 2>&1
6,16,26,36,46,56 * * * * /u/94/yanj3/unix/task1/task1-2.sh >>
/u/94/yanj3/unix/task1/task1-2 2>&1
6,16,26,36,46,56 * * * * /u/94/yanj3/unix/task1/task1-3.sh >>
/u/94/yanj3/unix/task1/task1-3 2>&1
```

2 Table of measurement results

2.1 Table of measurement results for name server set

2.1.1 Analyse the data for name server set

According to the shell script above, the order of date produced should be like “date info--dig info--ping info”, so checking whether there is data loss based on the following cmds.

There should be 180 records for date, dig and ping test in total for 6 name servers, (each has 30 records), among which, there are 48 dig records missing some dig info because of “*;;; connection timed out; no servers could be reached*” after some research.

```
yanjing@v901-381 task1 % cat task1-1 | grep 2020-09 | wc -l
     180
yanjing@v901-381 task1 % cat task1-1 | grep "ANSWER SECTION" | wc -l
     132
```

```
yanjing@v901-381 task1 % cat task1-1 | grep PING | wc -l
    180
```

Dealing with the abnormal data based on the following cmds.
The line ID for each output of “date” cmd should be n*22 then, if not, then the dig data is missing in the corresponding record. So the line 2244 - line 2724 is abnormal data info.

```
# Delete all blank lines in task1-1 file
yanjing@v901-381 task1 % grep -v '^$' task1-1 >> task1-1-tmp1
# Add ID (from 0) for each line of task-1 file
yanjing@v901-381 task1 % awk '$0=NR-1":"$0' task1-1-tmp1 >> task1-1-tmp2
```

Delete the abnormal data and get the info as follows:

```
yanjing@v901-394 task1 % cat task1-1| grep 2020-09| wc -l
    132
yanjing@v901-394 task1 % cat task1-1| grep PING| wc -l
    132
yanjing@v901-394 task1 % cat task1-1| grep "ANSWER SECTION"| wc -l
    132
# All pings succeed here
yanjing@v901-381 task1 % cat task1-1 | grep "0% packet loss" | wc -l
    132
yanjing@v901-394 task1 % cat task1-1 | head -n23
2020-09-26T11:06:01 EEST
; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 d.dns.br
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54485
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;d.dns.br.          IN  A
;; ANSWER SECTION:
d.dns.br.       21599   IN  A   200.219.154.10
;; Query time: 160 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Sep 26 11:06:01 EEST 2020
;; MSG SIZE  rcvd: 53
PING d.dns.br (200.219.154.10) 56(84) bytes of data.
[1601107561.487516] 64 bytes from d.dns.br (200.219.154.10): icmp_seq=1 ttl=52
time=166 ms
--- d.dns.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 166.197/166.197/166.197/0.000 ms
2020-09-26T11:06:01 EEST
```

2.1.2 Deal with the data for name server set

```
# Collect date info
yanjing@v901-394 task1 % cat task1-1 | grep 2020-09 | head -n2
2020-09-26T11:06:01 EEST
2020-09-26T11:06:01 EEST
yanjing@v901-394 task1 % cat task1-1 | grep 2020-09 >> tmp1
# Collect dns query info
yanjing@v901-381 task1 % cat task1-1 | grep "ANSWER SECTION" -A2 | head -n5
;; ANSWER SECTION:
d.dns.br.       21599   IN  A   200.219.154.10
;; Query time: 160 msec
--
;; ANSWER SECTION:
yanjing@v901-381 task1 % cat task1-1 | grep "ANSWER SECTION" -A2 >> tmp2
yanjing@v901-381 task1 % sed -e '/ANSWER/d' tmp2 >> tmp3
```

```
yanjing@v901-381 task1 % sed -e '/--/d' tmp3 >> tmp4
yanjing@v901-381 task1 % vim tmp4 (with :%s/\;\; Query time\://g and then save it)
yanjing@v901-381 task1 % cat tmp4 | awk '{print $1}' >> tmp5
yanjing@v901-381 task1 % vim tmp5  (with :%s/br\.\n/br\t/g and then save it)
yanjing@v901-381 task1 % cat tmp5 | head -n2
d.dns.br    160
e.dns.br    76
# Collect ping info
yanjing@v901-381 task1 % cat task1-1 | grep PING -A4 | head -n5
PING d.dns.br (200.219.154.10) 56(84) bytes of data.
[1601107561.487516] 64 bytes from d.dns.br (200.219.154.10): icmp_seq=1 ttl=52
time=166 ms
--- d.dns.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 166.197/166.197/166.197/0.000 ms
yanjing@v901-381 task1 % cat task1-1 | grep PING -A4 >> tmp6
yanjing@v901-381 task1 % sed -e '/transmitted/d' tmp6 >> tmp7
yanjing@v901-381 task1 % sed -e '/--/d' tmp7 >> tmp8
yanjing@v901-381 task1 % sed -e '/PING/d' tmp8 >> tmp9
yanjing@v901-381 task1 % vim tmp9 (with :%s/64 bytes from//g and :%s/\nrtt/\trtt/g
and then save it)
yanjing@v901-381 task1 % cat tmp9 | awk '{print $1, $2, $11}' >> tmp10
yanjing@v901-381 task1 % head -n2 tmp10
[1601107561.487516] d.dns.br 166.197/166.197/166.197/0.000
[1601107561.610330] e.dns.br 33.761/33.761/33.761/0.000
```

2.1.3 Create a csv file for name server set

```
# Create a csv
yanjing@v901-381 task1 % paste -d ',' tmp1 tmp5 tmp10 > task1-1.csv
# After several times modifying (sed, awk, sort), the task1.csv file displays the
following info,
# And because of there is only 1 packet for ping cmd, so rtt_min, rtt_avg and
rtt_max are the same
yanjing@v901-394 task1 % head -n5 task1-1.csv
name_server,date,timestamp,dns_query_msec,rtt_min,rtt_avg,rtt_max
a.dns.br,2020-09-26T11:06:02 EEST,[1601107562.832638],468,305.721,305.721,305.721
a.dns.br,2020-09-26T12:06:02 EEST,[1601111162.521874],28,290.011,290.011,290.011
a.dns.br,2020-09-26T13:06:02 EEST,[1601114762.494924],24,304.461,304.461,304.461
a.dns.br,2020-09-26T14:06:01 EEST,[1601118362.159702],24,290.486,290.486,290.486
```

2.2 Table of measurement results for research server set

2.2.1 Deal with the data for name server set and create a csv file

(No abnormal data) (Similar with 2.1.2)

```
# Delete unuseless data
yanjing@v901-394 task1 % sed -e '/icmp_seq=/d' task1-2 > tmp1
yanjing@v901-394 task1 % grep -v '^$' tmp1 > tmp2
yanjing@v901-394 task1 % sed -e '/PING/d' tmp2 > tmp3
# After several times dealing with the date (including awk, sed, sort), got the
following file
yanjing@v901-394 task1 % cat task1-2.csv | head -n5
research_server,date,packets_sent,packet_,rtt_min,rtt_avg,rtt_max,rtt_mdev
hlz-nz.ark.caida.org,2020-09-26T10:56:06EEST,5,5,303.281,303.340,303.433,0.050
hlz-nz.ark.caida.org,2020-09-26T11:06:05EEST,5,5,303.277,303.302,303.338,0.021
hlz-nz.ark.caida.org,2020-09-26T11:16:07EEST,5,5,303.215,303.297,303.381,0.057
hlz-nz.ark.caida.org,2020-09-26T11:26:07EEST,5,5,303.213,303.305,303.398,0.070
```

2.3 Table of measurement results for research server set

2.3.1 Deal with the data for name server set and create a csv file

(No abnormal data) (similar with 2.1.2)

```
# Delete unuseless data
yanjing@v901-394 task1 % grep -v '^$' task1-3 > tmp1
yanjing@v901-394 task1 % sed -e '/PING/d' tmp1 > tmp2
yanjing@v901-394 task1 % sed -e '/icmp_seq/d' tmp2 > tmp3
yanjing@v901-394 task1 % sed -e '/curl/d' tmp3 > tmp4
yanjing@v901-394 task1 % sed -e '/Total/d' tmp4 > tmp5
yanjing@v901-394 task1 % sed -e '/--:--/d' tmp5 > tmp6
# After several times dealing with the date (including awk, sed, sort), got the
following file
yanjing@v901-394 task1 % head -n5 task1-3.csv
iperf_server,date,packets_sent,packets_received,tcp_connect_time,rtt_min,rtt_avg,rt
t_max,rtt_mdev
blr1.iperf.comnet-student.eu,2020-09-26T11:06:05EEST,5,5,0.323090,337.116,337.356,3
37.959,0.306
blr1.iperf.comnet-student.eu,2020-09-26T11:16:06EEST,5,5,0.335043,337.111,337.284,3
37.654,0.190
blr1.iperf.comnet-student.eu,2020-09-26T11:26:06EEST,5,5,0.324903,337.111,337.318,3
37.788,0.242
blr1.iperf.comnet-student.eu,2020-09-26T11:36:06EEST,5,5,0.335899,337.133,337.285,3
37.661,0.190
```

3 Conclusions on network stability

3.1 Create scripts to analyse the date of the name server set

3.1.1 Create a script to show latencies of dns query and ICMP echo request in a table

(No packets lost which can be see in 2.1.1)

```
import os
import csv
import pandas as pd
import numpy as np
filepath1 = open('/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-1.csv','r+')
csv1 = pd.read_csv(filepath1)
os.system("echo ''>
/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-1-result.csv")
result =
open('/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-1-result.csv','w')
writer = csv.writer(result)
dic1 = {}
dnslist_mean = ['dns_query_msec_mean']
dnslist_median = ['dns_query_msec_median']
rttlist_mean = ['rtt_msec_mean']
rttlist_median = ['rtt_msec_median']
for name_server in ["a", "b", "c", "d", "e", "f"]:
   dic1[name_server]={"dns_query_msec":[],"rtt":[]}
   for index, row in csv1.iterrows():
       name_server_row = row["name_server"]
       dns_query_msec_row = int(row["dns_query_msec"])
       rtt_row = float('%.3f'%row["rtt_min"])
       if name_server_row.startswith(name_server):
           dic1[name_server]["dns_query_msec"].append(dns_query_msec_row)
           dic1[name_server]["rtt"].append(rtt_row)
   for item in dic1[name_server]:
       if item.startswith("dns"):
           dnslist_mean.append(float('%.3f'%np.mean(dic1[name_server][item])))
```

```
            dnslist_median.append(float('%.3f'%np.median(dic1[name_server][item])))
        if item.startswith("rtt"):
            rttlist_mean.append(float('%.3f'%np.mean(dic1[name_server][item])))
            rttlist_median.append(float('%.3f'%np.median(dic1[name_server][item])))
writer.writerows([["2020-09-26T11:06:02 to 2020-09-27T16:06:02","a.dns.br",
"b.dns.br", "c.dns.br", "d.dns.br", "e.dns.br", "f.dns.br"], dnslist_mean,
dnslist_median, rttlist_mean, rttlist_median])
```

task1-1-result

2020-09-26T11:06:02 to 2020-09-27T16:06:02	a.dns.br	b.dns.br	c.dns.br	d.dns.br	e.dns.br	f.dns.br
dns_query_msec_mean	80.909	97.455	66.364	101.091	141.455	107.455
dns_query_msec_median	26.0	28.0	28.0	74.0	82.0	78.0
rtt_msec_mean	261.626	177.341	236.731	166.612	33.291	249.062
rtt_msec_median	269.711	177.049	236.082	166.254	32.646	248.308

3.1.2 Conclusion for name server set

1> No packet loss during “ping” any server, so the networks are stable enough.
2> The rtt time of “ping” e.dns.br is lowest, so this might be the closest one to my testing environment.
3> The latencies of the dns query obtained by dig cmd here for these six servers are very similar.

3.2 Create scripts to analyse the date of the research server set

3.2.1 Create a script to show latencies and packets loss and latency ICMP echo request in a table

```
import os
import csv
import pandas as pd
import numpy as np
filepath1 = open('/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-2.csv','r+')
csv1 = pd.read_csv(filepath1)
os.system("echo ''>
/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-2-result.csv")
result =
open('/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-2-result.csv','w')
writer = csv.writer(result)
dic1 = {}
for research_server in ["hlz", "pna"]:

dic1[research_server]={"packets_sent":0,"packets_received":0,"rtt_min":[],"rtt_avg"
:[],"rtt_max":[],"rtt_mdev":[]}
   for index, row in csv1.iterrows():
       research_server_row = row["research_server"]
       packets_sent_row = int(row["packets_sent"])
       packets_received_row = int(row["packets_received"])
       rtt_min_row = float('%.3f'%row["rtt_min"])
       rtt_avg_row = float('%.3f'%row["rtt_avg"])
       rtt_max_row = float('%.3f'%row["rtt_max"])
       rtt_mdev_row = float('%.3f'%row["rtt_mdev"])
       if research_server_row.startswith(research_server):
           dic1[research_server]["packets_sent"] =
dic1[research_server]["packets_sent"] + packets_sent_row
           dic1[research_server]["packets_received"] =
dic1[research_server]["packets_received"] + packets_received_row
           dic1[research_server]["rtt_min"].append(rtt_min_row)
```

```
            dic1[research_server]["rtt_avg"].append(rtt_avg_row)
            dic1[research_server]["rtt_max"].append(rtt_max_row)
            dic1[research_server]["rtt_mdev"].append(rtt_mdev_row)
writer.writerow(["2020-09-26T10:56:06EEST-2020-09-27T16:36:07EEST",
"hlz-nz.ark.caida.org", "pna-es.ark.caida.org"])
writer.writerow(["package_loss",
float('%.3f'%(dic1['hlz']["packets_received"]/dic1['hlz']["packets_sent"])),
float('%.3f'%(dic1['pna']["packets_received"]/dic1['pna']["packets_sent"]))])
writer.writerow(["rtt_min_mean", float('%.3f'%np.mean(dic1['hlz']['rtt_min'])),
float('%.3f'%np.mean(dic1['pna']['rtt_min']))])
writer.writerow(["rtt_min_median", float('%.3f'%np.median(dic1['hlz']['rtt_min'])),
float('%.3f'%np.median(dic1['pna']['rtt_min']))])
writer.writerow(["rtt_avg_mean", float('%.3f'%np.mean(dic1['hlz']['rtt_avg'])),
float('%.3f'%np.mean(dic1['pna']['rtt_avg']))])
writer.writerow(["rtt_avg_median", float('%.3f'%np.median(dic1['hlz']['rtt_avg'])),
float('%.3f'%np.median(dic1['pna']['rtt_avg']))])
writer.writerow(["rtt_max_mean", float('%.3f'%np.mean(dic1['hlz']['rtt_max'])),
float('%.3f'%np.mean(dic1['pna']['rtt_max']))])
writer.writerow(["rtt_max_median", float('%.3f'%np.median(dic1['hlz']['rtt_max'])),
float('%.3f'%np.median(dic1['pna']['rtt_max']))])
writer.writerow(["rtt_mdev_mean", float('%.3f'%np.mean(dic1['hlz']['rtt_mdev'])),
float('%.3f'%np.mean(dic1['pna']['rtt_mdev']))])
writer.writerow(["rtt_mdev_median",
float('%.3f'%np.median(dic1['hlz']['rtt_mdev'])),
float('%.3f'%np.median(dic1['pna']['rtt_mdev']))])
```

task1-2-result

2020-09-26T10:56:06EEST-2020-09-27T16:36:07EEST	hlz-nz.ark.caida.org	pna-es.ark.caida.org
package_loss	0.002	0.136
rtt_min_mean	258.74	134.349
rtt_min_median	303.284	86.903
rtt_avg_mean	258.846	134.51
rtt_avg_median	303.345	86.965
rtt_max_mean	258.993	134.795
rtt_max_median	303.415	87.069
rtt_mdev_mean	0.093	0.177
rtt_mdev_median	0.047	0.055

3.2.2 Conclusion for research server set

1> The network from my testing environment to “hlz-nz.ark.caida.org” is more stable than the other one cause the lower package loss rate. Overall, these two networks are stable enough.
2> For each rtt time, the time basically costs less for “ping” “pna-es.ark.caida.org”, which means “pna-es.ark.caida.org” might be closer to my environment.

3.3 Conclusion for iperf server set

3.3.1 Create a script to show latencies of TCP connection and ICMP echo request in a table

```
import os
import csv
import pandas as pd
```

```
import numpy as np
filepath1 = open('/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-3.csv','r+')
csv1 = pd.read_csv(filepath1)
os.system("echo ''>
/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-3-result.csv")
result =
open('/Users/yanjing/Desktop/ITMA/Assignment-2/task1/task1-3-result.csv','w')
writer = csv.writer(result)
dic1 = {}
for iperf in ["blr1", "iperf"]:

dic1[iperf]={"packets_sent":0,"packets_received":0,"tcp_connect_time":[],"rtt_min":
[],"rtt_avg":[],"rtt_max":[],"rtt_mdev":[]}
   for index, row in csv1.iterrows():
       iperf_server = row["iperf_server"]
       packets_sent = int(row["packets_sent"])
       packets_received = int(row["packets_received"])
       tcp_connect_time = float('%.6f'%row["tcp_connect_time"])
       rtt_min = float('%.3f'%row["rtt_min"])
       rtt_avg = float('%.3f'%row["rtt_avg"])
       rtt_max = float('%.3f'%row["rtt_max"])
       rtt_mdev = float('%.3f'%row["rtt_mdev"])
       if iperf_server.startswith(iperf):
           dic1[iperf]["packets_sent"] = dic1[iperf]["packets_sent"] + packets_sent
           dic1[iperf]["packets_received"] = dic1[iperf]["packets_received"] +
packets_received
           dic1[iperf]["tcp_connect_time"].append(tcp_connect_time)
           dic1[iperf]["rtt_min"].append(rtt_min)
           dic1[iperf]["rtt_avg"].append(rtt_avg)
           dic1[iperf]["rtt_max"].append(rtt_max)
           dic1[iperf]["rtt_mdev"].append(rtt_mdev)
writer.writerow(["2020-09-26T11:06:05EEST-2020-09-27T16:46:05EEST",
"blr1.iperf.comnet-student.eu", "iperf.netlab.hut.fi"])
writer.writerow(["package_loss", float('%.3f'%(1 -
dic1['blr1']["packets_received"]/dic1["blr1"]["packets_sent"])), float('%.3f'%(1 -
dic1['iperf']["packets_received"]/dic1['iperf']["packets_sent"]))])
writer.writerow(["tcp_connect_time_mean",
float('%.6f'%np.mean(dic1['blr1']['tcp_connect_time'])),
float('%.6f'%np.mean(dic1['iperf']['tcp_connect_time']))])
writer.writerow(["tcp_connect_time_median",
float('%.6f'%np.median(dic1['blr1']['tcp_connect_time'])),
float('%.6f'%np.median(dic1['iperf']['tcp_connect_time']))])
writer.writerow(["rtt_min_mean", float('%.3f'%np.mean(dic1['blr1']['rtt_min'])),
float('%.3f'%np.mean(dic1['iperf']['rtt_min']))])
writer.writerow(["rtt_min_median",
float('%.3f'%np.median(dic1['blr1']['rtt_min'])),
float('%.3f'%np.median(dic1['iperf']['rtt_min']))])
writer.writerow(["rtt_avg_mean", float('%.3f'%np.mean(dic1['blr1']['rtt_avg'])),
float('%.3f'%np.mean(dic1['iperf']['rtt_avg']))])
writer.writerow(["rtt_avg_median",
float('%.3f'%np.median(dic1['blr1']['rtt_avg'])),
float('%.3f'%np.median(dic1['iperf']['rtt_avg']))])
writer.writerow(["rtt_max_mean", float('%.3f'%np.mean(dic1['blr1']['rtt_max'])),
float('%.3f'%np.mean(dic1['iperf']['rtt_max']))])
writer.writerow(["rtt_max_median",
float('%.3f'%np.median(dic1['blr1']['rtt_max'])),
float('%.3f'%np.median(dic1['iperf']['rtt_max']))])
writer.writerow(["rtt_mdev_mean", float('%.3f'%np.mean(dic1['blr1']['rtt_mdev'])),
float('%.3f'%np.mean(dic1['iperf']['rtt_mdev']))])
writer.writerow(["rtt_mdev_median",
float('%.3f'%np.median(dic1['blr1']['rtt_mdev'])),
float('%.3f'%np.median(dic1['iperf']['rtt_mdev']))])
```

task1-3-result

2020-09-26T11:06:05EEST-2020-09-27T16:46:05EEST	blr1.iperf.comnet-student.eu	iperf.netlab.hut.fi
package_loss	0.001	0.0
tcp_connect_time_mean	0.334033	0.002034
tcp_connect_time_median	0.332378	0.00182
rtt_min_mean	340.466	0.671
rtt_min_median	340.276	0.678
rtt_avg_mean	340.916	0.803
rtt_avg_median	340.387	0.715
rtt_max_mean	342.124	0.955
rtt_max_median	340.755	0.751
rtt_mdev_mean	0.682	0.102
rtt_mdev_median	0.205	0.027

3.3.2 Conclusion for iperf server set

1> The network from my testing environment to “iperf.netlab.hut.fi” is more stable than the other one cause the lower package loss rate. Overall, these two networks are stable enough.
2> The time of tcp connection costs less when downloading the file through curl from “iperf.netlab.hut.fi”, which means faster downloading speed. (Might be bandwidth issue or distance issue, not sure the direct factors.)
3> For each rtt time, the time obviously costs less for “ping” “iperf.netlab.hut.fi”, which means “iperf.netlab.hut.fi” might be closer to my environment.

Task 2: Measuring throughput

Requirement (Task 2)

In this task you will measure throughput in three different ways: by file transfer, by special measurement tool, and by using measurement service. The measurement service method must be done manually, the rest could and should be automatized with, for example, crontab. Finally, you will compare their results.

Most network users are interested only in a single factor: how many bits per second one can download or send with this network connection. Therefore, if possible, run these throughput tests at your home - remember to check power saving settings of your computer. If you have metered (mobile) broadband then we do recommend to use Aalto workstations or servers for the task.

- Download files from the target server using some HTTP download tool (curl recommended)
- Network performance measurement tool iperf3 for 10 seconds in both directions.
- Use measurement service such as Speed Test

Target servers for 1 and 2 can be found from the table in annex Servers.

Start performing measurements once an hour with methods 1 and 2. Use the same student id method to distribute measurements over time. In optimum case, one of delay measurements should be run at the same time as the throughput measurements. Collect statistics when you have made measurements over a minimum of 24 hours. Leave measurements running for a minimum of two weeks. Run a few measurements by hand with method 3 within the same time frame and write down the date, time and results from it.

Make a table where you result from these 3 methods and calculate basic statistics, such as mean, median, max, min and average deviation. Note that for method 2 and 3 you will receive upload and download readings. Report them separately

	HTTP	iperf up	iperf dn	ST up	St dn
17:05 14.9.2017	125	16	137	16	135
19:05 14.9.2017	117	15	132		
Mean	121	15	135	16	135
Median	117	15	132	16	135
Min	117	15	132	16	135
Max	125	16	137	16	135
Avg deviation	4	1	2	0	0

See definition of Avg deviation.

Report (task 2)

1 Describe your measurement setup.

Note: Include (example) code and samples of results.

1.1 Create a shell script

```
yanj3@force ~/task2 % cat task2.sh
date
echo "HTTP download tool"
curl -o /dev/null http://ok1.iperf.comnet-student.eu/500M.bin
echo "iperf3 tool - up"
iperf3 -c ok1.iperf.comnet-student.eu
echo "iperf3 tool - down"
iperf3 -c ok1.iperf.comnet-student.eu -R
yanj3@force ~/task2 % python3 /u/94/yanj3/unix/task2/speedtest.py;date
Retrieving speedtest.net configuration...
Testing from Aalto University (130.233.224.200)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by Elisa Oyj (Helsinki) [12.96 km]: 1.526 ms
```

```
Testing download
speed.....................................................................
..
Download: 3097.88 Mbit/s
Testing upload
speed.....................................................................
.......................
Upload: 2599.04 Mbit/s
2020-09-26T19:06:01 EEST
```

1.2 Crontab setting

```
yanj3@force ~/task2 % crontab -l
# task2
6 * * * * /u/94/yanj3/unix/task2/task2.sh >> /u/94/yanj3/unix/task2/task2 2>&1
```

2 Table of results

2.1 Deal with the file generated by the script above

2.1.1 After simple checking the number of record, find the following abnormal data

Use “0” to fill in these two records

```
 9 iperf3 tool - up
10 iperf3: error - the server is busy running a test. try again later
11 iperf3 tool - down
12 iperf3: error - the server is busy running a test. try again later
```

2.1.2 Generate a csv file according to the data

```
# Dealing with date info
yanjing@v901-381 task2 % cat task2 | grep EEST > tmp1-date
yanjing@v901-381 task2 % head -n3 tmp1-date
2020-09-26T18:06:01 EEST
2020-09-26T19:06:01 EEST
2020-09-26T20:06:01 EEST
```

```
# Dealing with HTTP data
yanjing@v901-381 task2 % cat task2 | grep "HTTP download tool" -A3 > tmp1-http
yanjing@v901-381 task2 % head -n5 tmp1-http
HTTP download tool
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  500M  100  500M    0     0   874M      0 --:--:-- --:--:-- --:--:--  875M
--
yanjing@v901-381 task2 % sed -e '/HTTP download tool/d' tmp1-http > tmp2
yanjing@v901-381 task2 % sed -e '/Total/d' tmp2 > tmp3
# After manually deleting the middle process of curl output, got the file as
follows
yanjing@v901-381 task2 % head -n4 tmp3
100  500M  100  500M    0     0   874M      0   875M
100  500M  100  500M    0     0   728M      0   727M
100  500M  100  500M    0     0   910M      0   909M
yanjing@v901-381 task2 % cat tmp3 | awk '{print $16}' > tmp1-http
yanjing@v901-381 task2 %
yanjing@v901-381 task2 % head -n3 tmp1-http
874M
728M
910M
```

```
# Dealing with iperf up data
```

```
yanjing@v901-381 task2 % cat task2 | grep "iperf3 tool - up" -A19 > tmp1-iperfup
yanjing@v901-381 task2 % head -n20 tmp1-iperfup
iperf3 tool - up
Connecting to host ok1.iperf.comnet-student.eu, port 5201
[  5] local 130.233.224.200 port 48020 connected to 195.148.124.36 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  1.03 GBytes  8.81 Gbits/sec    0   1.66 MBytes
[  5]   1.00-2.00   sec  1.05 GBytes  9.05 Gbits/sec    0   1.66 MBytes
[  5]   2.00-3.00   sec  1.06 GBytes  9.07 Gbits/sec    0   1.66 MBytes
[  5]   3.00-4.00   sec  1.05 GBytes  9.03 Gbits/sec    0   1.66 MBytes
[  5]   4.00-5.00   sec  1.05 GBytes  9.00 Gbits/sec    0   1.66 MBytes
[  5]   5.00-6.00   sec  1.05 GBytes  9.05 Gbits/sec    0   1.66 MBytes
[  5]   6.00-7.00   sec  1.05 GBytes  8.98 Gbits/sec    0   1.66 MBytes
[  5]   7.00-8.00   sec  1.05 GBytes  9.01 Gbits/sec    0   1.66 MBytes
[  5]   8.00-9.00   sec  1.06 GBytes  9.11 Gbits/sec    0   1.66 MBytes
[  5]   9.00-10.00  sec  1.04 GBytes  8.97 Gbits/sec    0   1.66 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  10.5 GBytes  9.01 Gbits/sec    0             sender
[  5]   0.00-10.00  sec  10.5 GBytes  9.01 Gbits/sec                  receiver

iperf Done.
yanjing@v901-381 task2 % cat tmp1-iperfup | grep sender > tmp1
yanjing@v901-394 task2 % cat tmp1 | awk '{print $7 $8}' > tmp1-iperfup
yanjing@v901-394 task2 % head -n3 tmp1-iperfup
9.01Gbits/sec
9.01Gbits/sec
8.98Gbits/sec
```

```
# Dealing with iperf down data
yanjing@v901-381 task2 % cat task2 | grep "iperf3 tool - down" -A20 >
tmp1-iperfdown
yanjing@v901-381 task2 % head -n21 tmp1-iperfdown
iperf3 tool - down
Connecting to host ok1.iperf.comnet-student.eu, port 5201
Reverse mode, remote host ok1.iperf.comnet-student.eu is sending
[  5] local 130.233.224.200 port 48028 connected to 195.148.124.36 port 5201
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  1.01 GBytes  8.70 Gbits/sec
[  5]   1.00-2.00   sec  1.09 GBytes  9.33 Gbits/sec
[  5]   2.00-3.00   sec  1.09 GBytes  9.32 Gbits/sec
[  5]   3.00-4.00   sec  1.09 GBytes  9.34 Gbits/sec
[  5]   4.00-5.00   sec  1.09 GBytes  9.33 Gbits/sec
[  5]   5.00-6.00   sec  1.09 GBytes  9.33 Gbits/sec
[  5]   6.00-7.00   sec  1.09 GBytes  9.33 Gbits/sec
[  5]   7.00-8.00   sec  1.09 GBytes  9.34 Gbits/sec
[  5]   8.00-9.00   sec  1.09 GBytes  9.34 Gbits/sec
[  5]   9.00-10.00  sec  1.08 GBytes  9.25 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  10.8 GBytes  9.26 Gbits/sec   62             sender
[  5]   0.00-10.00  sec  10.8 GBytes  9.26 Gbits/sec                  receiver

iperf Done.
yanjing@v901-394 task2 % cat tmp1-iperfdown | grep receiver > tmp1
yanjing@v901-394 task2 % cat tmp1 | awk '{print $7 $8}' > tmp1-iperfdown
yanjing@v901-394 task2 % head -n3 tmp1-iperfdown
9.26Gbits/sec
9.27Gbits/sec
9.30Gbits/sec
```

```
# Dealing with speedtest data
yanjing@v901-394 task2 % cat tmp1-speedup
2599.04 Mbit/s
yanjing@v901-394 task2 % cat tmp1-speeddown
```

```
3097.88 Mbit/s
```

```
# Generate a csv file
yanjing@v901-394 task2 % paste -d ',' tmp1-date tmp1-http tmp1-iperfup
tmp1-iperfdown tmp1-speedup tmp1-speeddown > task2.csv
yanjing@v901-394 task2 % head -n3 task2.csv
date,HTTP(Mbytes/s),iperf_up(Gbits/s),iperf_down(Gbits/s),ST_up(bit/s),ST_down(bit/
s)
2020-09-26T18:06:01 EEST,874,9.01,9.26,2599.04 ,3097.88
2020-09-26T19:06:01 EEST,728,9.01,9.27,
```

date	HTTP(Mbytes/s)	iperf_up(Gbits/s)	iperf_down(Gbits/s)	ST_up(Mbit/s)	ST_down(Mbit/s)
2020-09-26T18:06:01 EEST	874	9.01	9.26	2599.04	3097.88
2020-09-26T19:06:01 EEST	728	9.01	9.27	nan	nan
2020-09-26T20:06:01 EEST	910	8.98	9.3	nan	nan
2020-09-26T21:06:01 EEST	843	9.03	9.21	nan	nan
2020-09-26T22:06:01 EEST	574	0	0	nan	nan
2020-09-26T23:06:01 EEST	917	7.52	9.23	nan	nan
2020-09-27T00:06:01 EEST	850	7.38	9.26	nan	nan
2020-09-27T01:06:02 EEST	496	8.86	9.25	nan	nan

2.2 Create a script to produce a table like above

```
import csv
import pandas as pd
import numpy as np
filepath1 = open('/Users/yanjing/Desktop/ITMA/Assignment-2/task2/task2.csv','r')
csv1 = pd.read_csv(filepath1)
filepath2 = open('/Users/yanjing/Desktop/ITMA/Assignment-2/task2/task2.csv','a')
writer2 = csv.writer(filepath2)
writer2.writerow(["Mean", float('%.3f'%np.mean(csv1["HTTP(Mbytes/s)"])),
float('%.3f'%np.mean(csv1["iperf_up(Gbits/s)"])),
float('%.3f'%np.mean(csv1["iperf_down(Gbits/s)"])),
float('%.3f'%np.mean(csv1["ST_up(bit/s)"])),
float('%.3f'%np.mean(csv1["ST_down(bit/s)"]))])
writer2.writerow(["Median", np.median(csv1["HTTP(Mbytes/s)"]),
np.median(csv1["iperf_up(Gbits/s)"]), np.median(csv1["iperf_down(Gbits/s)"]),
np.median(csv1["ST_up(bit/s)"]), np.median(csv1["ST_down(bit/s)"])])
writer2.writerow(["Min", np.min(csv1["HTTP(Mbytes/s)"]),
np.min(csv1["iperf_up(Gbits/s)"]), np.min(csv1["iperf_down(Gbits/s)"]),
np.min(csv1["ST_up(bit/s)"]), np.min(csv1["ST_down(bit/s)"])])
writer2.writerow(["Max", np.max(csv1["HTTP(Mbytes/s)"]),
np.max(csv1["iperf_up(Gbits/s)"]), np.max(csv1["iperf_down(Gbits/s)"]),
np.max(csv1["ST_up(bit/s)"]), np.max(csv1["ST_down(bit/s)"])])
writer2.writerow(["Avg_deviation",
float('%.3f'%pd.Series(csv1["HTTP(Mbytes/s)"]).mad()),
float('%.3f'%pd.Series(csv1["iperf_up(Gbits/s)"]).mad()),
float('%.3f'%pd.Series(csv1["iperf_down(Gbits/s)"]).mad()),
float('%.3f'%pd.Series(csv1["ST_up(bit/s)"]).mad()),
float('%.3f'%pd.Series(csv1["ST_down(bit/s)"]).mad())])
```

2020-09-28T13:06:02 EEST	940	4.05	8.93	nan	nan
2020-09-28T14:06:01 EEST	936	7.05	0	nan	nan
2020-09-28T15:06:01 EEST	970	5.48	9.03	nan	nan
2020-09-28T16:06:01 EEST	801	3.34	8.68	nan	nan
Mean	827.915	7.957	8.57	2599.04	3097.88
Median	850.0	8.76	9.25	nan	nan
Min	496	0.0	0.0	2599.04	3097.88
Max	970	9.13	9.33	2599.04	3097.88
Avg_deviation	81.176	1.184	1.166	0.0	0.0

3 Conclusions

Note: give answers for at least following topics. It may be beneficial to graph results to identify some trends.
3.1 Are the results between methods in line with each other?
⇒ HTTP's throughout is similar to Iperf.

3.2 Did some method have a lot of deviation? What do you think might cause this?
⇒ HTTP method has a lot of deviation, I think maybe it is related to downloading a 500M file that needs lots of TCP packets in an order (sometimes there are lots of TCP reconnection shown as follows), so the network speed waves a lot.

```
HTTP download tool
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
^M  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0^M
49  500M   49  247M    0     0   784M      0 --:--:-- --:--:-- --:--:--  782M^M100
500M  100  500M    0     0   728M      0 --:--:-- --:--:-- --:--:--  727M
```

3.3 Was there some method that gives higher values than others? What do you think might cause this?
⇒ The network speed order is: Iperf is a little higher than HTTP, HTTP is much higher than Speedtest tool. Network speed and throughput of Iperf are a little more stable than of HTTP, cause downloading a 500M file needs lots of TCP packets in an order (sometimes there are lots of TCP reconnection), but basically the throughout of them are close . As for SpeedTest tool, I do not think it is necessary to compare them, cause I did not point out the exact server which is used by SpeedTest, it is chosen by SpeedTest itself, so the server in SpeedTest is different from the servers used for HTTP and Iperf test.

3.4 Is there variation due time? For example, did you get higher throughput during day or night?
⇒ No obvious difference for HTTP throughput during day or night. But for iperf throughput, the data is higher and more stable during night from overall view.

3.5 Was there any anomalies? For example, no connection or very different capacity.
⇒ Yes, there are some anomalies for iperf testing as follows:

```
 9 iperf3 tool - up
10 iperf3: error - the server is busy running a test. try again later
```

```
11 iperf3 tool - down
12 iperf3: error - the server is busy running a test. try again later
```

TIPS: Look how the structure of the log file looks like, and you could modify the existing parse.py file to suit the objective.

Task 3: Producing data files

Requirement (task 3)

While the small number of results is easy to collect, to provide data for later exercises, we need to automate things further. If you do this task while collecting data for the Tasks 1 and 2, it also makes them easier.

Make a plan for the data model for both latency and throughput measurements. Provided you will have at least following items for each sent latency measurement:

- timestamp
- type of measurement (ping, dns, tcp,…)
- target
- result (delay, failed)

For the throughput items may include

- timestamp
- type of measurement (iperf, http, …)
- result items (bitrate, time elapsed, bytes transferred, failed)
- additional data (tcp retransmissions, system load)

Implement a method to convert raw result output to defined format. You can use python for this purpose but other scripting languages are possible. CSV type files are one option but there may be others too.

Report (task 3)

3.1 Generate ping statistics in abcde_ping.csv for name server set

```
yanjing@v901-394 task3 % cat task1-1 | grep "2020-09" > abcde-ping-date

yanjing@v901-394 task3 % cat task1-1 | grep "PING" -A5 > abcde-ping-data
yanjing@v901-394 task3 % grep -v '^$' abcde-ping-data > abcde-ping-data_1
yanjing@v901-394 task3 % vim abcde-ping-data_1 (%s/--- //g; %s/ ---//g)
yanjing@v901-394 task3 % grep -v '^$' abcde-ping-data_1 > abcde-ping-data_2
yanjing@v901-394 task3 % sed -e '/PING/d' abcde-ping-data_2 >> abcde-ping-data_3
yanjing@v901-394 task3 % vim abcde-ping-data_3 (%s/ ping statistics//g)

yanjing@v901-394 task3 % cat abcde-ping-data_3 | grep rtt > abcde_ping_rtt
yanjing@v901-394 task3 % vim abcde_ping_rtt (%srtt min\/avg\/max\/mdev = //g,
%s/\//\,/g, %s/ms//g)
yanjing@v901-394 task3 % cat abcde-ping-data_3 | grep transmitted >
abcde_ping_packet
yanjing@v901-394 task3 % vim abcde_ping_packet (%s/packets transmitted//g,
%s/received//g, %s/ //g)
```

```
yanjing@v901-394 task3 % cat abcde_ping_packet | awk -F',0%' '{print $1}' >
abcde_ping_packet_tr_re

yanjing@v901-394 task3 % sed -e '/rtt/d' abcde-ping-data_3 > tmp1
yanjing@v901-394 task3 % sed -e '/transmitted/d' tmp1 > tmp2
yanjing@v901-394 task3 % cat tmp2 | awk '{print $1}' > abcde_ping_timestamp_server
yanjing@v901-394 task3 % cat abcde_ping_timestamp_server | grep "dns" >
abcde_ping_server
yanjing@v901-394 task3 % sed -e '/dns/d' abcde_ping_timestamp_server >
abcde_ping_timestamp

yanjing@v901-394 task3 % paste -d ',' abcde_ping_server abcde_ping_timestamp
abcde-ping-date abcde_ping_packet_tr_re abcde_ping_rtt > tmp.csv
yanjing@v901-394 task3 % sort tmp.csv > abcde_ping.csv (insert
"server,time_stamp,date,packet_tr,packet_re,rtt_min,rtt_avg,rtt_max,rtt_mdev" to
the first line)
```

abcde_ping

server	time_stamp	date	packet_tr	packet_re	rtt_min	rtt_avg	rtt_max	rtt_mdev
a.dns.br	[1601107562.832638]	2020-09-26T11:06:02 EEST	1	1	305.721	305.721	305.721	0.000
a.dns.br	[1601111162.521874]	2020-09-26T12:06:02 EEST	1	1	290.011	290.011	290.011	0.000
a.dns.br	[1601114762.494924]	2020-09-26T13:06:02 EEST	1	1	304.461	304.461	304.461	0.000
a.dns.br	[1601118362.159702]	2020-09-26T14:06:01 EEST	1	1	290.486	290.486	290.486	0.000
a.dns.br	[1601121962.131707]	2020-09-26T15:06:01 EEST	1	1	296.516	296.516	296.516	0.000
a.dns.br	[1601125563.301995]	2020-09-26T16:06:02 EEST	1	1	307.932	307.932	307.932	0.000
a.dns.br	[1601129162.897983]	2020-09-26T17:06:02 EEST	1	1	293.519	293.519	293.519	0.000
a.dns.br	[1601132762.421651]	2020-09-26T18:06:02 EEST	1	1	267.950	267.950	267.950	0.000

3.2 Generate ping statistics in pna_hlz_ping.csv for research server set

```
yanjing@v901-394 task3 % cat task1-2| grep EEST > tmp1-date
yanjing@v901-394 task3 % sed -e '/EEST/d' task1-2 > tmp1
yanjing@v901-394 task3 % grep -v '^$' tmp > tmp2
yanjing@v901-394 task3 % sed -e '/PING/d' tmp2 > tmp3
yanjing@v901-394 task3 % sed -e '/icmp_seq/d' tmp3 > tmp4
yanjing@v901-394 task3 % cat tmp4 | grep "ping statistics" > tmp1-server
yanjing@v901-394 task3 % vim tmp1-server (%s/ ping statistics ---//g, %s/--- //g)
yanjing@v901-394 task3 % cat tmp4 | grep "transmitted" > tmp1-packet
yanjing@v901-394 task3 % vim tmp1-packet (%s/packets transmitted//g,
%s/received//g, %s/ //g)
yanjing@v901-394 task3 % cat tmp1-packet | awk -F',' '{print $1}' > tmp1
yanjing@v901-394 task3 % cat tmp1-packet | awk -F',' '{print $2}' > tmp2
yanjing@v901-394 task3 % paste -d ',' tmp1 tmp2 > tmp1-packet-tr-re
yanjing@v901-394 task3 % cat tmp4 | grep "rtt" > tmp1-rtt
yanjing@v901-394 task3 % vim tmp1-rtt (%s/rtt min\/avg\/max\/mdev = //g,
%s/\//\,/g, %s/ms//g)
yanjing@v901-394 task3 % paste -d ',' tmp1-server tmp1-date tmp1-packet-tr-re
tmp1-rtt > tmp.csv
yanjing@v901-394 task3 % sort tmp.csv > hlz_pna_ping.csv (insert
"server,date,packet_tr,packet_re,rtt_min,rtt_avg,rtt_max,rtt_mdev" to the first
line)
```

hlz_pna_ping

server	date	packet_tr	packet_re	rtt_min	rtt_avg	rtt_max	rtt_mdev
hlz-nz.ark.caida.org	2020-09-26T10:56:06 EEST	5	5	303.281	303.340	303.433	0.050
hlz-nz.ark.caida.org	2020-09-26T11:06:05 EEST	5	5	303.277	303.302	303.338	0.021
hlz-nz.ark.caida.org	2020-09-26T11:16:07 EEST	5	5	303.215	303.297	303.381	0.057
hlz-nz.ark.caida.org	2020-09-26T11:26:07 EEST	5	5	303.213	303.305	303.398	0.070
hlz-nz.ark.caida.org	2020-09-26T11:36:06 EEST	5	5	303.330	303.356	303.400	0.024
hlz-nz.ark.caida.org	2020-09-26T11:46:06 EEST	5	5	303.192	303.298	303.351	0.060
hlz-nz.ark.caida.org	2020-09-26T11:56:06 EEST	5	5	303.239	303.291	303.342	0.036
hlz-nz.ark.caida.org	2020-09-26T12:06:06 EEST	5	5	303.335	303.380	303.441	0.034

3.3 Generate ping statistics in iperf_ping.csv for iperf server set

```
yanjing@v901-394 task3 % cat task1-3| grep EEST > tmp1-date
yanjing@v901-394 task3 % cat task1-3| grep PING -A9 > tmp1
yanjing@v901-394 task3 % cat tmp1 | grep statistics > tmp1-server
yanjing@v901-394 task3 % vim tmp1-server (:%s/--- //g,  %s/ping statistics ---//g)
yanjing@v901-394 task3 % cat tmp1 | grep transmitted > tmp1-packet
yanjing@v901-394 task3 % cat tmp1-packet | awk -F',' '{print $1}' > tmp1-tr
yanjing@v901-394 task3 % cat tmp1-packet | awk -F',' '{print $2}' > tmp1-re
yanjing@v901-394 task3 % vim tmp1-tr (%s/ packets transmitted//g)
yanjing@v901-394 task3 % vim tmp1-re (%s/ received//g)
yanjing@v901-394 task3 % paste -d ',' tmp1-tr tmp1-re > tmp1-packet-tr-re
yanjing@v901-394 task3 % cat tmp1 | grep rtt > tmp1-rtt
yanjing@v901-394 task3 % vim tmp1-rtt (%s/rtt min\/avg\/max\/mdev = //g,
%s/\//\,/g, %s/ms//g)
yanjing@v901-394 task3 % paste -d ',' tmp1-server tmp1-date tmp1-packet-tr-re
tmp1-rtt > tmp1
yanjing@v901-394 task3 % sort tmp1 > iperf_ping.csv (insert
"server,date,packet_tr,packet_re,rtt_min,rtt_avg,rtt_max,rtt_mdev" to the first
line)
```

iperf_ping

server	date	packet_tr	packet_re	rtt_min	rtt_avg	rtt_max	rtt_mdev
blr1.iperf.comnet-student.eu	2020-09-26T11:06:05 EEST	5	5	337.116	337.356	337.959	0.306
blr1.iperf.comnet-student.eu	2020-09-26T11:16:06 EEST	5	5	337.111	337.284	337.654	0.190
blr1.iperf.comnet-student.eu	2020-09-26T11:26:06 EEST	5	5	337.111	337.318	337.788	0.242
blr1.iperf.comnet-student.eu	2020-09-26T11:36:06 EEST	5	5	337.133	337.285	337.661	0.190
blr1.iperf.comnet-student.eu	2020-09-26T11:46:05 EEST	5	5	337.132	337.166	337.228	0.032
blr1.iperf.comnet-student.eu	2020-09-26T11:56:05 EEST	5	5	339.712	339.824	340.147	0.162
blr1.iperf.comnet-student.eu	2020-09-26T12:06:05 EEST	5	5	339.682	339.703	339.734	0.019
blr1.iperf.comnet-student.eu	2020-09-26T12:16:05 EEST	5	5	339.650	339.771	340.132	0.182
blr1.iperf.comnet-student.eu	2020-09-26T12:26:05 EEST	5	5	337.128	337.265	337.665	0.201

3.4 Generate dns query in abcdef_dns_query.csv for name server set

```
yanjing@v901-394 task3 % cat task1-1| grep 2020-09 > tmp1-date
yanjing@v901-394 task3 % sed -e '/EEST/d' task1-1 > tmp1
yanjing@v901-394 task3 % sed -e '/PING/d' tmp1 > tmp2
yanjing@v901-394 task3 % sed -e '/icmp_seq/d' tmp2 > tmp3
yanjing@v901-394 task3 % sed -e '/statistics/d' tmp3 > tmp4
yanjing@v901-394 task3 % sed -e '/transmitted/d' tmp4 > tmp5
yanjing@v901-394 task3 % sed -e '/rtt/d' tmp5 > tmp6
```

```
yanjing@v901-394 task3 % grep -v '^$' tmp6 > tmp7
# For the failed records with ";; connection timed out; no servers could be
reached", fill in ";; Query time: null msec"
yanjing@v901-394 task3 % cat tmp7 | grep "ANSWER SECTION" -A2 > tmp1-dns
yanjing@v901-394 task3 % vim tmp1-dns (%s/;; ANSWER SECTION//g:, %s/;; Query
time://g)
yanjing@v901-394 task3 % grep -v '^$' tmp1-dns > tmp2-dns
yanjing@v901-394 task3 % cat tmp2-dns| awk '{print $1}' > tmp3-dns
yanjing@v901-394 task3 % vim tmp3-dns (%s/br\.\n/br\,/g)
yanjing@v901-394 task3 % cat tmp3-dns | awk -F',' '{print $1}' > tmp3-dns-server
yanjing@v901-394 task3 % cat tmp3-dns | awk -F',' '{print $2}' > tmp3-dns-time
yanjing@v901-394 task3 % paste -d ',' tmp3-dns-server tmp1-date tmp3-dns-time >
tmp1
yanjing@v901-394 task3 % sort tmp1 > abcdef_dns_query.csv
```

abcdef_dns_query

server	date	dns_query_time
a.dns.br	2020-09-26T11:06:02 EEST	468
a.dns.br	2020-09-26T12:06:02 EEST	28
a.dns.br	2020-09-26T13:06:02 EEST	24
a.dns.br	2020-09-26T14:06:01 EEST	24
a.dns.br	2020-09-26T15:06:01 EEST	28
a.dns.br	2020-09-26T16:06:02 EEST	92
a.dns.br	2020-09-26T17:06:02 EEST	24
a.dns.br	2020-09-26T18:06:02 EEST	24
a.dns.br	2020-09-26T19:06:02 EEST	476
a.dns.br	2020-09-26T20:06:02 EEST	24
a.dns.br	2020-09-26T21:06:02 EEST	24
a.dns.br	2020-09-26T22:06:02 EEST	28
a.dns.br	2020-09-26T23:06:02 EEST	92
a.dns.br	2020-09-27T00:06:02 EEST	28
a.dns.br	2020-09-27T01:06:03 EEST	28
a.dns.br	2020-09-27T02:06:02 EEST	96
a.dns.br	2020-09-27T03:06:02 EEST	180
a.dns.br	2020-09-27T04:06:47 EEST	null
a.dns.br	2020-09-27T05:06:47 EEST	null

3.5 Generate tcp connect in iperf_tcp_connect.csv for name server set

```
yanjing@v901-394 task3 % cat task1-3 | grep EEST > tmp1-date
yanjing@v901-394 task3 % cat task1-3 | grep curl -A4 > tmp1-curl
yanjing@v901-394 task3 % vim tmp1-curl (:%s/curl iperf_server is //g)
yanjing@v901-394 task3 % sed -e '/--/d' tmp1-curl > tmp2
yanjing@v901-394 task3 % sed -e '/Total/d' tmp2 > tmp3
yanjing@v901-394 task3 % vim tmp3 (:%s/eu\n/eu\,/g)
yanjing@v901-394 task3 % cat tmp3 | awk -F',' '{print $1}' > tmp1-server
yanjing@v901-394 task3 % cat tmp3 | awk -F',' '{print $2}' > tmp1-time
yanjing@v901-394 task3 % paste -d ',' tmp1-server tmp1-date tmp1-time > tmp.csv
yanjing@v901-394 task3 % sort tmp.csv > iperf_tcp_connect.csv
```

iperf_tcp_connect

server	date	tcp_connect_time
blr1.iperf.comnet-student.eu	2020-09-26T11:06:05 EEST	0.323090
blr1.iperf.comnet-student.eu	2020-09-26T11:16:06 EEST	0.335043
blr1.iperf.comnet-student.eu	2020-09-26T11:26:06 EEST	0.324903
blr1.iperf.comnet-student.eu	2020-09-26T11:36:06 EEST	0.335899
blr1.iperf.comnet-student.eu	2020-09-26T11:46:05 EEST	0.333144
blr1.iperf.comnet-student.eu	2020-09-26T11:56:05 EEST	0.342991
blr1.iperf.comnet-student.eu	2020-09-26T12:06:05 EEST	0.334076
blr1.iperf.comnet-student.eu	2020-09-26T12:16:05 EEST	0.328008
blr1.iperf.comnet-student.eu	2020-09-26T12:26:05 EEST	0.331894
blr1.iperf.comnet-student.eu	2020-09-26T12:36:05 EEST	0.327210
blr1.iperf.comnet-student.eu	2020-09-26T12:46:05 EEST	0.326267
blr1.iperf.comnet-student.eu	2020-09-26T12:56:05 EEST	0.338448
blr1.iperf.comnet-student.eu	2020-09-26T13:06:05 EEST	0.337612
blr1.iperf.comnet-student.eu	2020-09-26T13:16:05 EEST	0.326327
blr1.iperf.comnet-student.eu	2020-09-26T13:26:05 EEST	0.317608
blr1.iperf.comnet-student.eu	2020-09-26T13:36:05 EEST	0.344378
blr1.iperf.comnet-student.eu	2020-09-26T13:46:05 EEST	0.333928
blr1.iperf.comnet-student.eu	2020-09-26T13:56:05 EEST	0.328321
blr1.iperf.comnet-student.eu	2020-09-26T14:06:05 EEST	0.322656

3.6 Generate HTTP and Iperf throughput intask3.csv for ok1.iperf.comnet-student.eu

```
yanjing@v901-394 task3 % cat task2 | grep EEST > tmp1-date
yanjing@v901-394 task3 % cat task2 | grep HTTP -A3 > tmp1-http
yanjing@v901-394 task3 % sed -e '/HTTP/d' tmp1-http > tmp2
yanjing@v901-394 task3 % sed -e '/Total/d' tmp2 > tmp3
yanjing@v901-394 task3 % vim tmp3 (%s/--:--:-- --:--:-- --:--:--//g,
:%s/0:00:01//g, :%s/--:--:--//g, :%s/\n--//g)
yanjing@v901-394 task3 % cat tmp3 > tmp4
yanjing@v901-394 task3 % cat tmp4  | grep "100  500M  100  500M" > tmp5
yanjing@v901-394 task3 % grep -v '^$' tmp5 > tmp6
yanjing@v901-394 task3 % cat tmp6 | awk '{print $2}' > tmp1_http_filesize
yanjing@v901-394 task3 % cat tmp6 | awk '{print $4}' > tmp1_http_receive
yanjing@v901-394 task3 % cat tmp6 | awk '{print $7}' > tmp1_http_speed
# For the records with "iperf3: error - the server is busy running a test. try
again later", fill in "[  5]   0.00-10.00  sec  null GBytes  null Gbits/sec    null
sender/receiver":w
yanjing@v901-394 task3 % cat task2 | grep "iperf3 tool - up" -A17 > tmp1-iperup
yanjing@v901-394 task3 % cat task2 | grep "iperf3 tool - down" -A18  >
tmp1-iperfdown
yanjing@v901-394 task3 % cat tmp1-iperup| grep sender > tmp1_iperf_up
yanjing@v901-394 task3 % cat tmp1-iperfdown | grep receiver > tmp1_iperf_down
yanjing@v901-394 task3 % cat tmp1_iperf_up | awk '{print $5}' >
tmp1_iperf_up_transfer
yanjing@v901-394 task3 % cat tmp1_iperf_up | awk '{print $7}' >
tmp1_iperf_up_bitrate
yanjing@v901-394 task3 % cat tmp1_iperf_up | awk '{print $9}' > tmp1_iperf_up_retr
```

```
yanjing@v901-394 task3 % cat tmp1_iperf_down | awk '{print $5}' >
tmp1_iperf_down_transfer
yanjing@v901-394 task3 % cat tmp1_iperf_down | awk '{print $7}' >
tmp1_iperf_down_bitrate
yanjing@v901-381 task3 % paste -d ',' tmp1-date tmp1_iperf_up_transfer
tmp1_iperf_up_bitrate tmp1_iperf_up_retr tmp1_iperf_down_transfer
tmp1_iperf_down_bitrate tmp1_http_filesize tmp1_http_receive tmp1_http_speed >
task3.csv
```

task3

date	iperf_up_transfer	iperf_up_bitrate	iperf_up_retr	iperf_down_transfer	iperf_down_bitrate	http_filesize	http_receive	http_speed
2020-09-26T18:06:01 EEST	10.5	9.01	0	10.8	9.26	500M	500M	874M
2020-09-26T19:06:01 EEST	10.5	9.01	0	10.8	9.27	500M	500M	728M
2020-09-26T20:06:01 EEST	10.4	8.98	0	10.8	9.30	500M	500M	910M
2020-09-26T21:06:01 EEST	10.5	9.03	0	10.7	9.21	500M	500M	843M
2020-09-26T22:06:01 EEST	null	null	null	null	null	500M	500M	574M
2020-09-26T23:06:01 EEST	8.76	7.52	20	10.7	9.23	500M	500M	917M
2020-09-27T00:06:01 EEST	8.59	7.38	98	10.8	9.26	500M	500M	850M
2020-09-27T01:06:02 EEST	10.3	8.86	0	10.8	9.25	500M	500M	496M
2020-09-27T02:06:01 EEST	10.4	8.92	0	10.8	9.30	500M	500M	939M
2020-09-27T03:06:01 EEST	10.5	9.02	0	10.8	9.30	500M	500M	834M
2020-09-27T04:06:01 EEST	10.4	8.91	0	9.79	8.41	500M	500M	931M
2020-09-27T05:06:01 EEST	10.4	8.90	0	10.8	9.26	500M	500M	805M
2020-09-27T06:06:01 EEST	10.6	9.13	0	10.9	9.33	500M	500M	915M
2020-09-27T07:06:01 EEST	10.4	8.97	0	10.7	9.22	500M	500M	736M
2020-09-27T08:06:01 EEST	10.2	8.73	71	10.7	9.20	500M	500M	646M
2020-09-27T09:06:01 EEST	10.4	8.94	0	10.8	9.28	500M	500M	886M
2020-09-27T10:06:01 EEST	10.5	8.99	0	10.8	9.31	500M	500M	840M
2020-09-27T11:06:01 EEST	10.4	8.95	0	10.8	9.29	500M	500M	829M