Internet Traffic Measurements - basic measurements

Sebastian Sonntag Markus Peuhkuri Tran Thien Thi

This work has three tasks that much complement ones for the first exercise. Read all instructions before starting because it is useful to identify common work. Task 3 for example makes reporting 1 and 2 easier.

- Task 1: Measuring latency
- Task 2: Measuring throughput
- Task 3: Producing data files

Also review supporting videos from the videos.

> **TIP:** Regarding crontab and SSH usage, check out `supporting_material.pdf` and `linux_intro.pdf`. When you edit crontab script in Aalto server, the edited crontab will be located there and will not show in any other computer (other server, workstation). Therefore, you need to edit your existing crontab script via SSH again.

Task 1: Measuring latency

In this task you will collect latency data from 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 minimum of two weeks as this data will be used in the Final Assignment.

- For the *name servers* set, measure latancy once a hour both with DNS query and ICMP echo request (check `-O` and `-D` options!). Send one of each requests per hour. Use *studentid modulo 60* to select the right minute to send.
- For the research servers set, run test every 10 minute by sending 5 ICMP echo requests. Select minute by *studentid modulo 10*.
- For the Iperf servers set, run 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 mush load. Select minute by *studentid modulo 10*.

TIPS: Create three shell scripts, one for reserch 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 a way. 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 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 start of hour but distribute it further. A recommended way to get evenly distributed (among students) value for minutes is to run `expr $(id -u) % 60`[1]. 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 quote low number of measurements but future we will have bigger dataset that are tedious or impossible to go through by hand.

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

1. Median delay with lost packets with delay of infinity, thus if more than 50 % of packets are lost, then consider as infinity.
2. Mean delay with lost packets not counted.
3. Loss ratio.
4. Delay spread as difference with 75th and 25th percentiles. If more than 25 % of packets is lost, then consider as infinity.

Finally, **make conclusions** about stability of network delay. Was some of 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

- Describe your measurement setup
- Table of measurement results.
- Conclusions on network stability.

[1] The `$id` is not actually student id, but will serve an approximate one

Task 2: Measuring throughput

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 about 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.

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

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

Start performing measurements once a hour with methods 1 and 2. Use same method 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 mesurements over minimum of 24 hours. Leave measurements running for mimimium of two weeks. Run few measurement by hand with method 3 within same time frame and write down the date, time and results from it.

Make a table where you results 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 separetly

Table 1: Example results from throughput measurements in megabits per second.

	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

- Describe your measurement setup. Include (example) code and samples of results.
- Table of results
- Conclusions, give answers for at least following topics. It may be beneficial to graph results to identify some trends.

1. Are the results between methods in line with each other?
2. Did some method has lot of deviation? What do think might cause this?
3. Was there some method that gives higher values than other? What do you think might cause this?
4. Is there variation due time? For example did you get higher throughput during day or night?
5. Was there are anomalies? For example, no connection or very different capacity.

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

Task 3: Producing data files

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 makes also 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 transfered, failed)
- additional data (tcp retransmissions, system load)

Implement a method to convert raw result out put to defined format. You can use for example python for this piurpose but other scripting languages are possible. CSV type file is one option but there may be other too.

Report, Task 3

- Describe your data model and file format.
- Describe program to generate data.

- Sample of results file.

Servers

Research servers

There are few distributed research testbeds, including Caida ARK where researchers can utilize for internet-wide measurements. Here we test latency for few of these sites.

Here we use following targets. Select from tables below hosts based on following formula (use integer division). One from the first table, two from the second.

- id_0 = studentid % 3
- id_1a = studentid % 6
- id_1b = studentid / 7 % 6

Table 2: select one: id0

id_0	server
0	pna-es.ark.caida.org
1	arn-se.ark.caida.org
2	lej-de.ark.caida.org

Table 3: select two: id1a and id1b

id_1	server
0	hlz-nz.ark.caida.org
1	cjj-kr.ark.caida.org
2	per-au.ark.caida.org
3	scl-cl.ark.caida.org
4	jfk-us.ark.caida.org
5	san-us.ark.caida.org

Traffic towards these destination can be more frequent (measurement intervals of 10 minutes towards these destinations are acceptable).

Iperf servers

Iperf3 servers accept only one connection at time. The hosts running Iperf3 servers used in this course are configured to run 11 different instances, each on different port from 5200 to 5210. Utilise for example `$RANDOM` variable to select port (`-p` option) in random.

The other one of iperf3 servers is `ok1.iperf.comnet-student.eu`.

The another server is selected according to *student id modulo 2* according to following table.

Table 4: iperf server.

id_0	Far away iperf server
0	blr1.iperf.comnet-student.eu
1	sgp1.iperf.comnet-student.eu

In addition to iperf3 server, the servers will serve following files over HTTP at TCP port 80.

Table 5: Sample file sizes

file	size
1K.bin	1 KiB
5K.bin	5 KiB
10M.bin	10 MiB
50M.bin	50 MiB
100M.bin	100 MiB
500M.bin	500 MiB
500G.bin	500 GiB

URL is for example http://blr1.iperf.comnet-student.eu/10M.bin. Based on iperf tets, select that file size that is most appopriate.

As it is not known in advance how much capacity there is available in the network, it would be prudient to define maximum time for transfer. Depending on used tool, there are alternatives:

- **curl** supports `-m secs` max-time option that will abort transfer if it takes more than `secs` seconds. Remember to include amount of *bytes transfered* to your output format.
- With any program it is possible to use **timeout** command that will kill (or send signal) it the program runs longger than set timeout. Used signal can be specified with `-s INT` to be `SIGINT` for example. To set **wget** command timeout to 60 seconds following command can be used: `timeout 60 wget http://...`

Nameservers

Course tools include mycountry tool. It assigns your some country based on your "userid" and performing few checks. That command must be run in one of

Aalto server computers.

```
bash
source /work/courses/unix/T/ELEC/E7130/general/use.sh
mycountry
```

The command will print something like this:

```
br OK (Brazil): d.dns.br, e.dns.br, f.dns.br, a.dns.br, c.dns.br,
b.dns.br Your UID is 1346517, thus your ccTLD is br (Brazil)
```

For this user the nameservers are:

- `d.dns.br`
- `e.dns.br`
- `f.dns.br`
- `a.dns.br`
- `c.dns.br`
- `b.dns.br`

If the nameserver does not respond to ICMP messages, try for other remote services or use **traceroute** to find the last hop before that server and use it as target (it it responds). Do not test traffic more frequently than twice an hour.

If you do not know which domain names there exists, try for example `news site:br` search on web. Actually, it should not make a difference if the domain exists or not.

Notice: You need to run the `mycountry` command yourself at Aalto IT `kosh.aalto.fi`, `lyta.aalto.fi`, `brute.aalto.fi` and `force.aalto.fi` server. You can access them easily via SSH connection remotely. In Aalto campus network direct DNS queries are disallowed from normal client networks. However, those are allowed from above servers. The same filtering applies typical residential networks too. You need to *run actual DNS latency tests from those servers* from where a direct DNS requests are allowed.