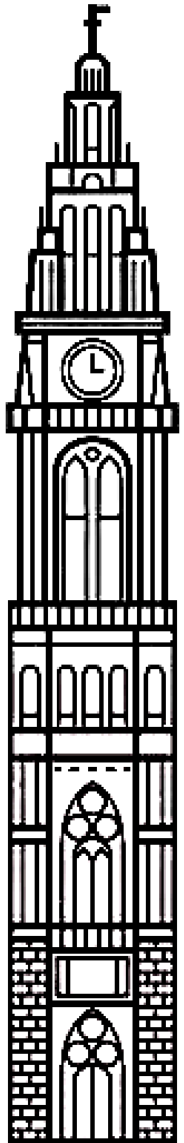
Remotely snooping on traffic patterns using network protocols

Kirils Solovjovs | kirils.org
@k@chaos.social
thanks to Pēteris Birkants

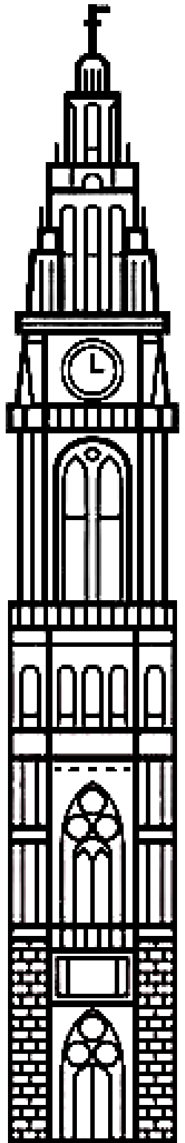
DEEPSEC

DeepSec | 2024-11-22



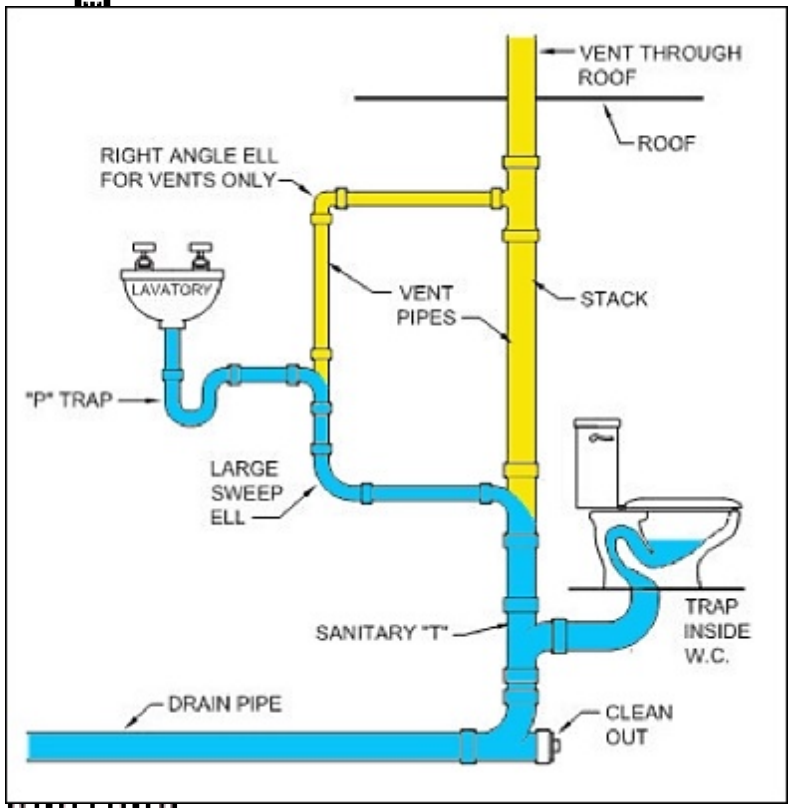
Kirils Solovjovs

- *Possible Security*, CEO
- Experienced in
  - network flow analysis
  - reverse engineering
  - social engineering
  - penetration testing
- @k@chaos.social



Contents

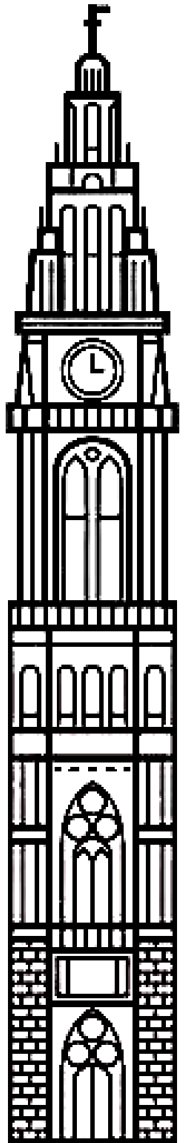
- Network throughput theory
- Setup and methodology
- Tooling
- Estimating bandwidth
- Traffic patterns



"This is a series of tubes"

– Your plumber
– No, not Al Gore

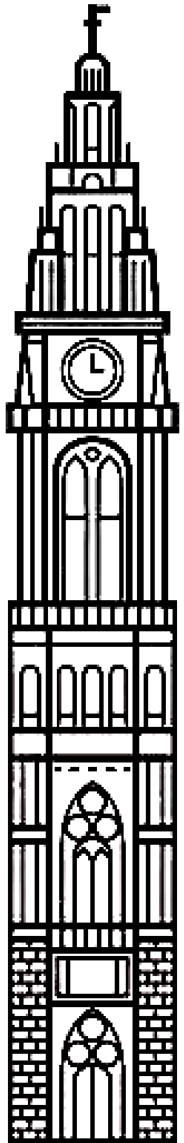
image source: Cornell University



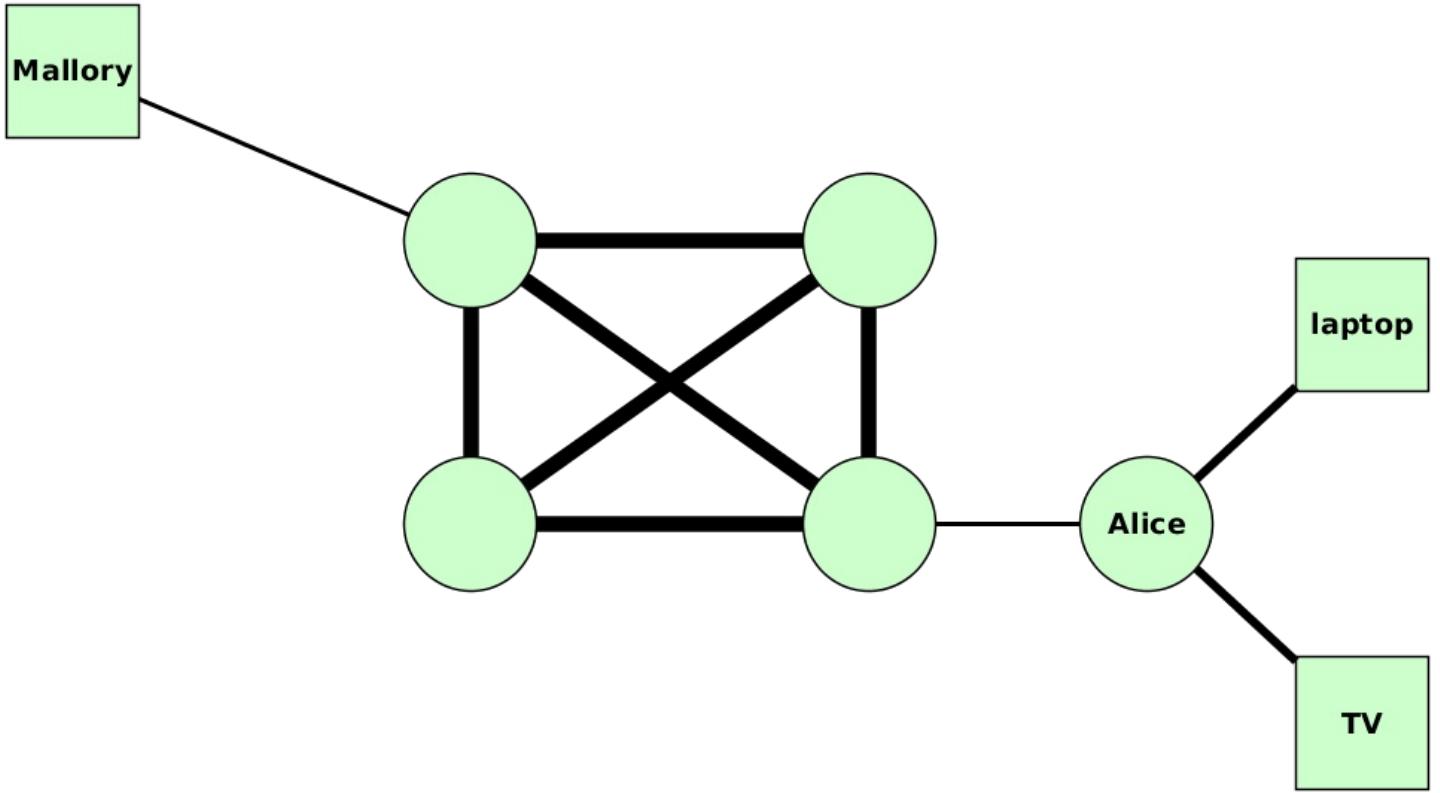
Bandwidth vs Throughput vs Latency

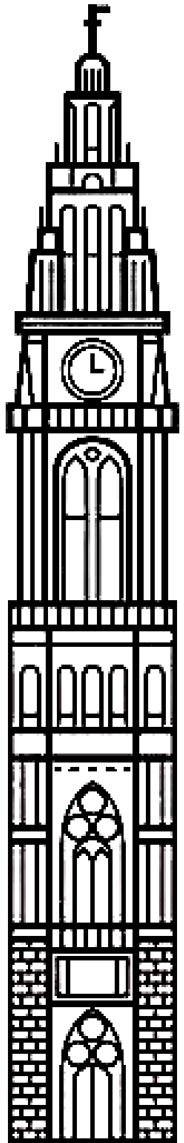
- Bandwidth – how wide is the pipe
- Throughput – how much stuff is traveling through the pipe
- Latency – how long until the one „drop” travels through

- latency = f(throughput/bandwidth)

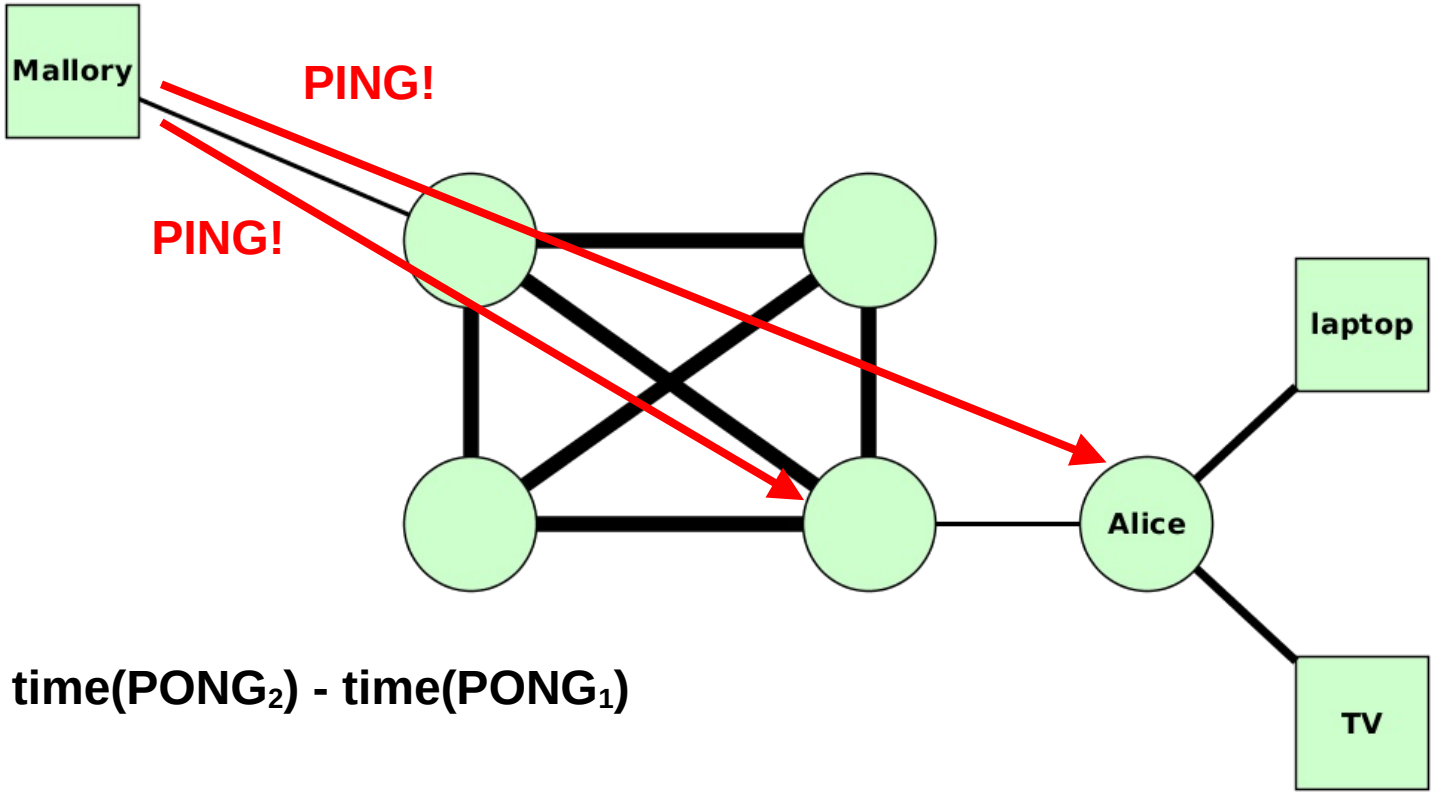


Targeted network layout

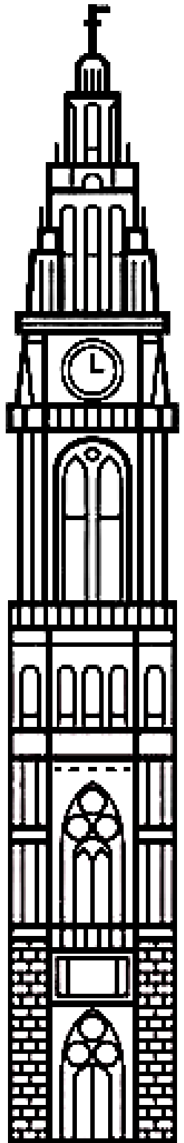




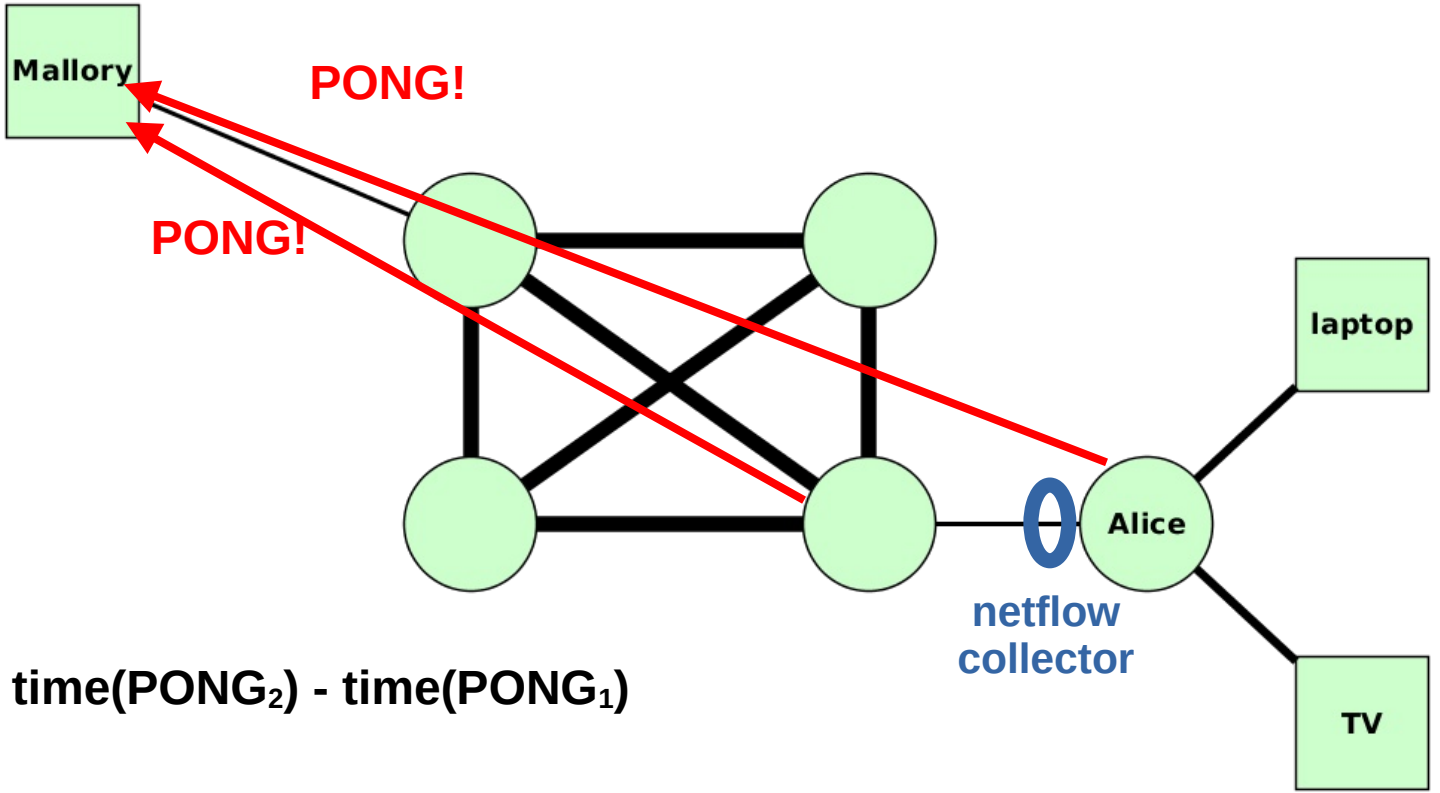
Measurement methodology



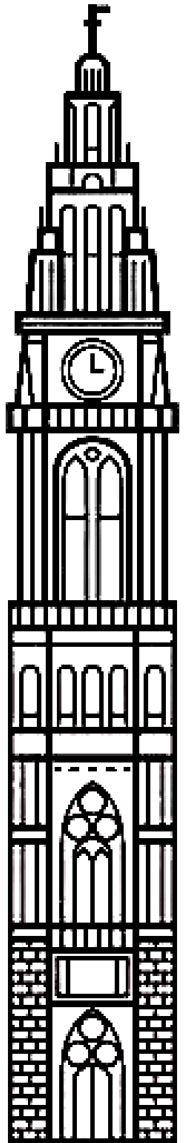
Δ = time($PONG_2$) - time($PONG_1$)



Measurement methodology



Δ = time(PONG$_2$) - time(PONG$_1$)

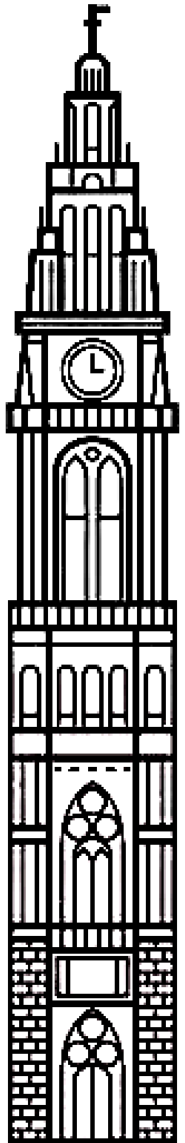


Quick GNU coreutils quiz

- What does `tr=""` mean in the command

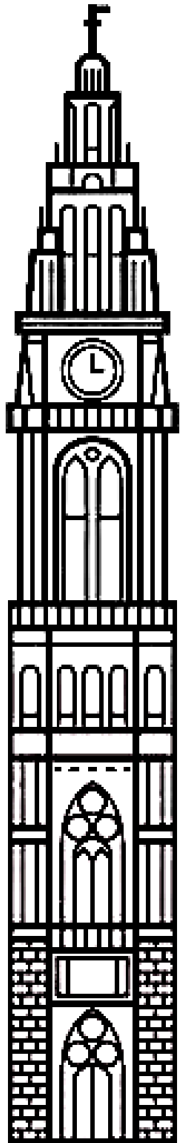
```
stdbuf -i0 -o0 -e0 tr = " "
```

?



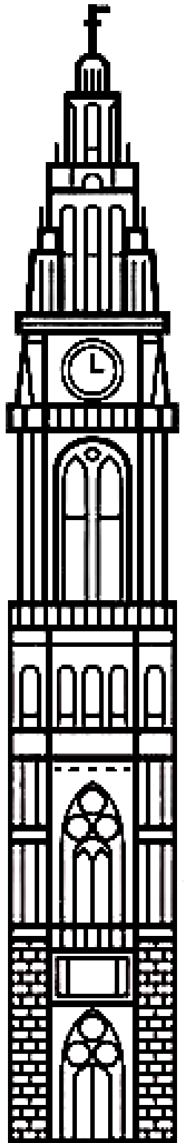
Tooling prototype (April 2024)

- Terminal1
  - while [$(date +%s) -lt $STARTTIME]; do sleep 0.001; done;
  - ping $TARGET -s 10000 | unbuffer tr = " " | unbuffer selcol 6 10 | unbuffer grep -v of | tee -a test-target
- Terminal2
  - while [$(date +%s) -lt $STARTTIME]; do sleep 0.001; done;
  - ping $UPLINK -s 10000 | unbuffer tr = " " | unbuffer selcol 6 10 | unbuffer grep -v of | tee -a test-gate



Tooling prototype (April 2024)

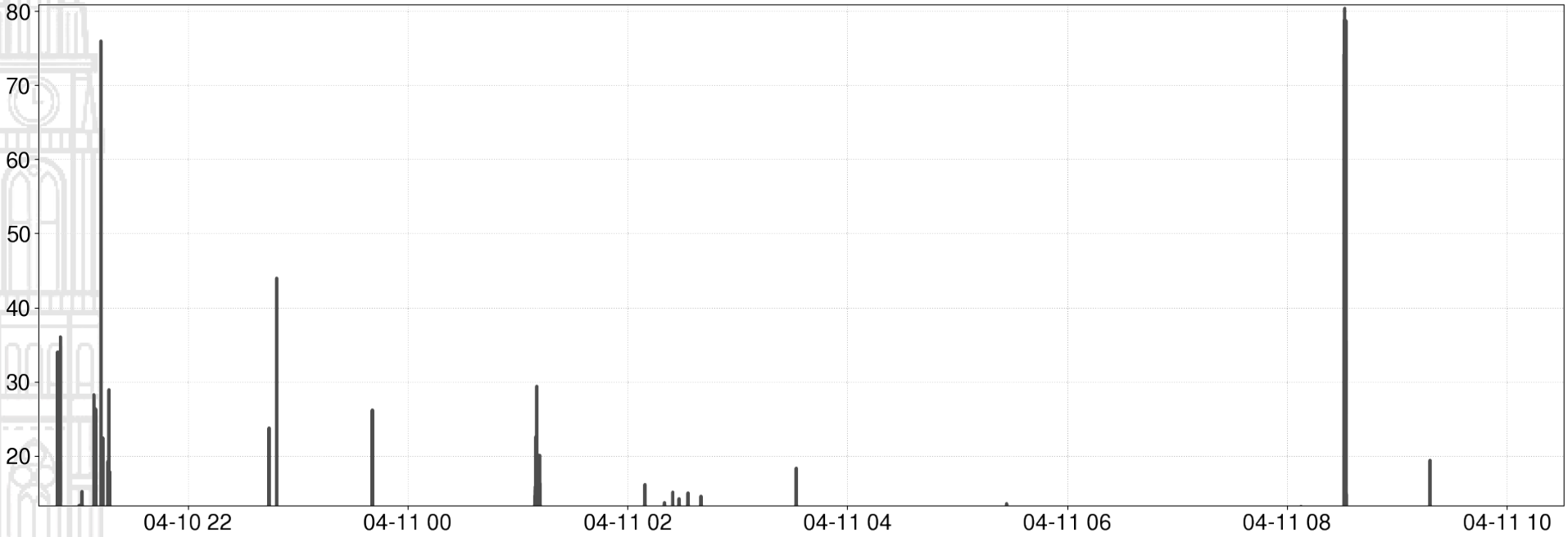
- Visualize
  - paste test-target test-gate | awk '{print $2-$4}' | chart plot output.png noheader noindex grid
- 5 lines, bash-terminal



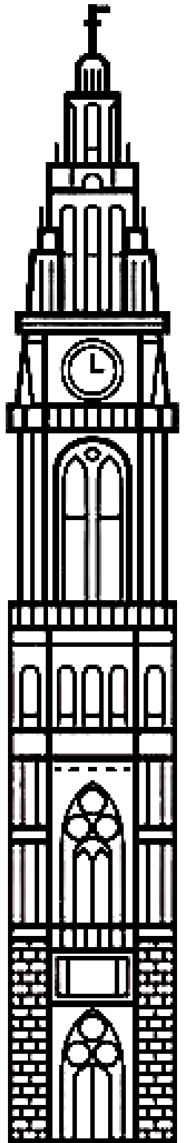
1st test

A computer on Alice's network (100BASE-TX) is downloading 1 GiB file (unshaped).

Accidental extended test



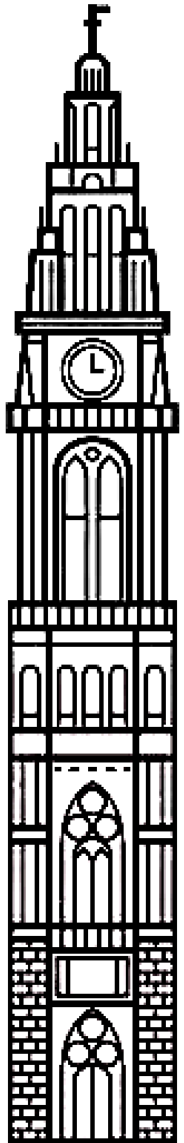
Alice's network (100BASE-TX) with multiple devices monitored overnight.



Tooling attempt (June 2024)

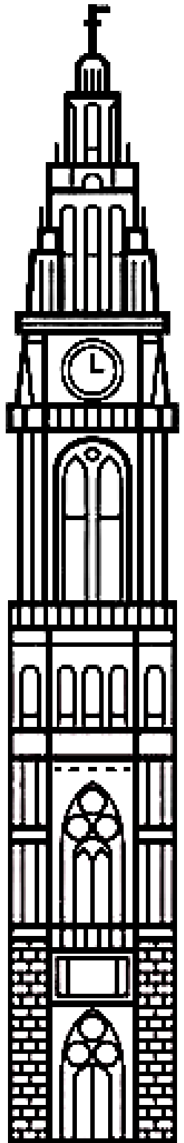
```
import asyncio
from concurrent.futures import ThreadPoolExecutor
from scapy.all import IP, ICMP, sr1
import time
from datetime import datetime
```

- Doesn’t work
- 58 lines, python3



Tooling (Aug 2024)

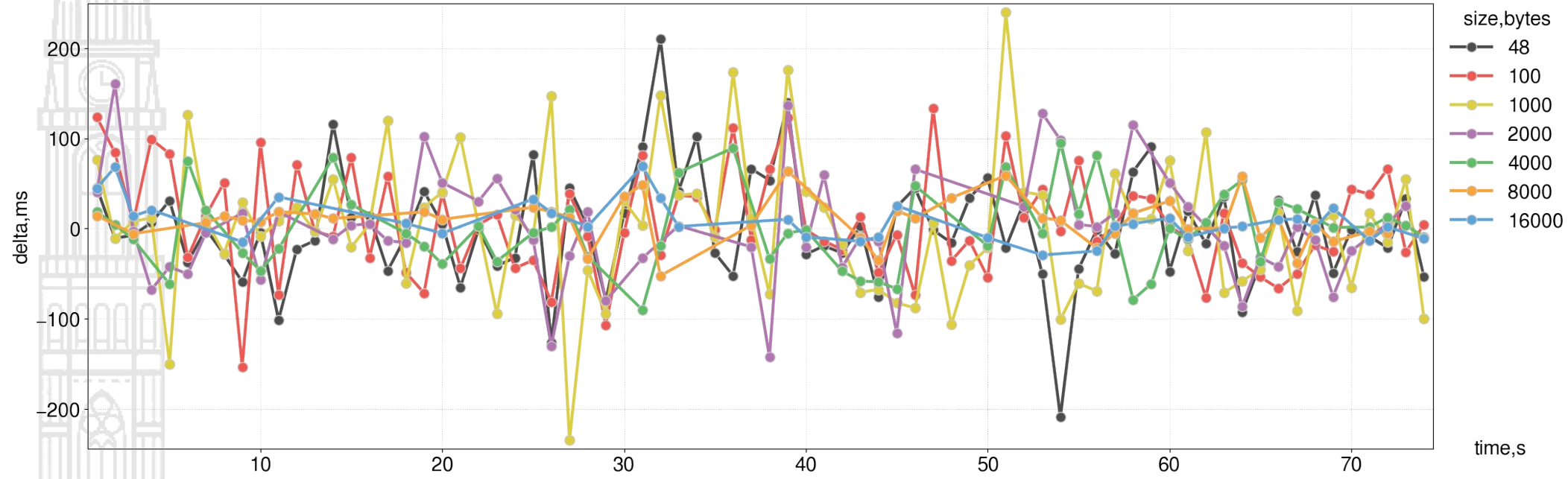
- Works
- Determines IP of ISP’s router
- Supports multiple payload sizes
- Can get desync’ed quickly
- 3 files, 71 lines, bash

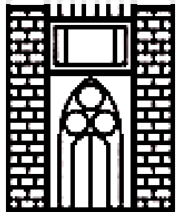


Tooling (Sep 2024)

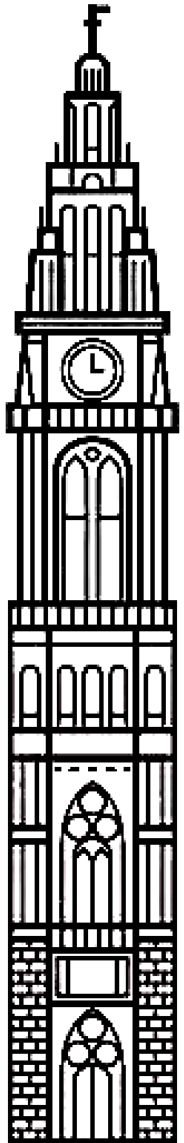
- Resynchronizes every tick
- Not limited to 1 packet per second
- 75 lines, bash

What's the best packet size?



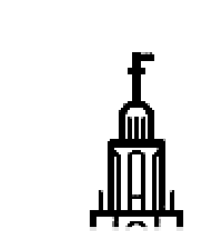


Idle network, sending all packet sizes simultaneously.

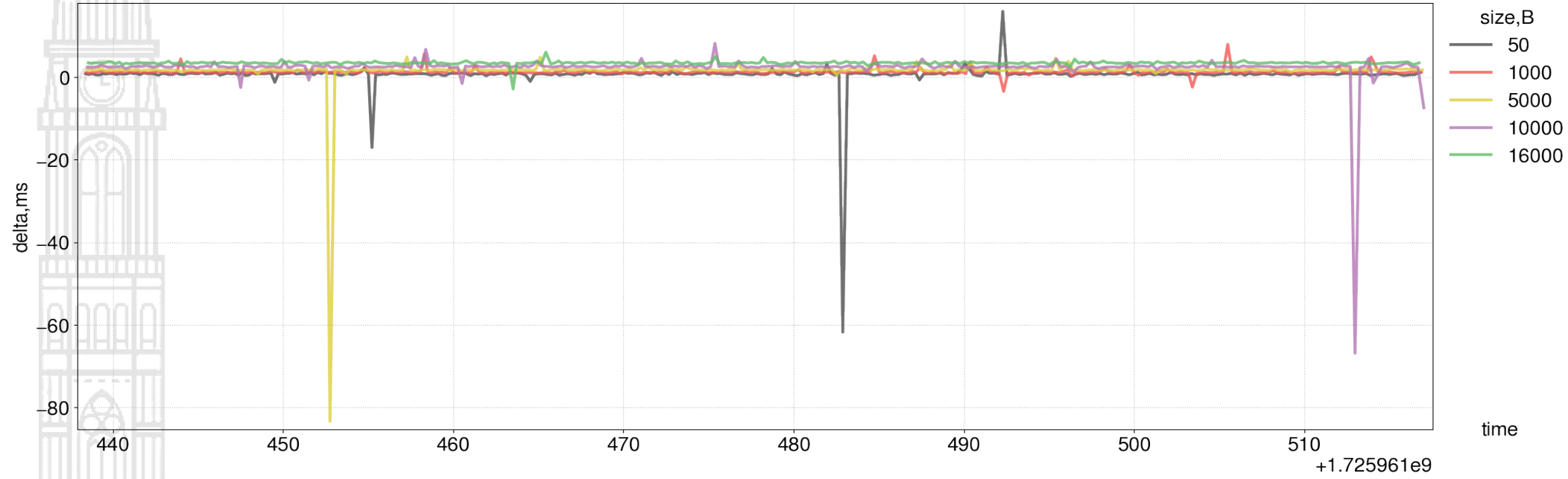


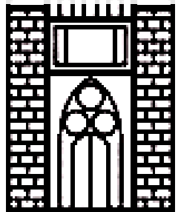
Tooling (Oct 2024)

- Sends packets one by one
- Data is actually useful again
- 72 lines, bash



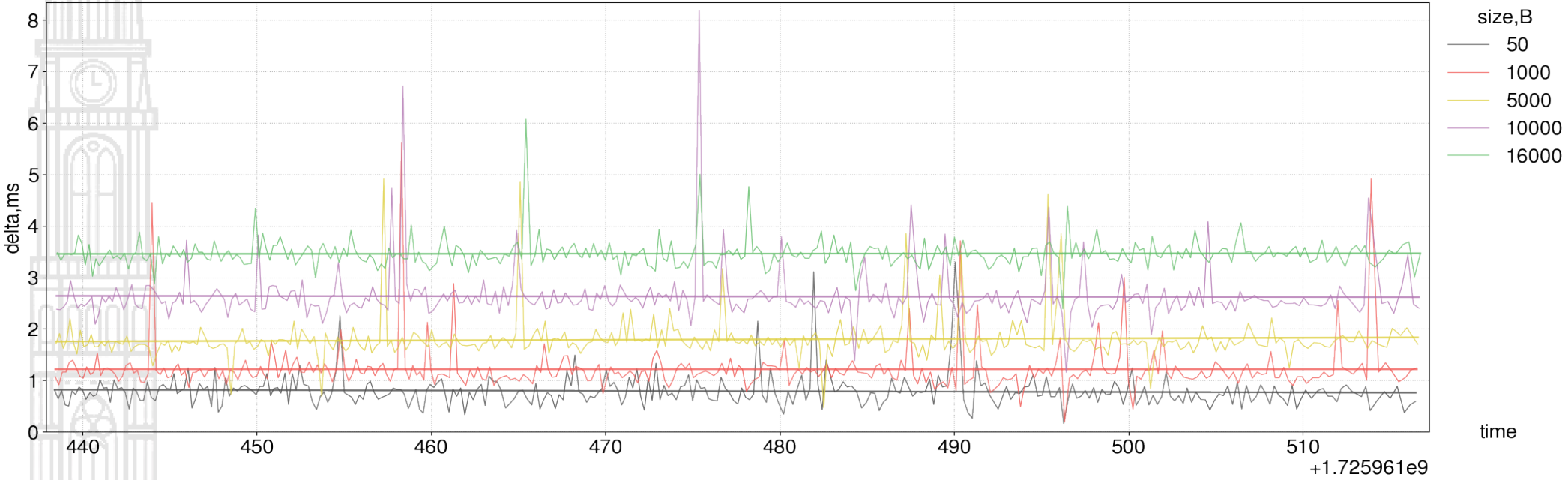
What's the best packet size?

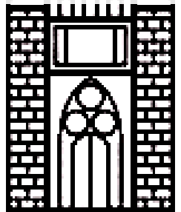




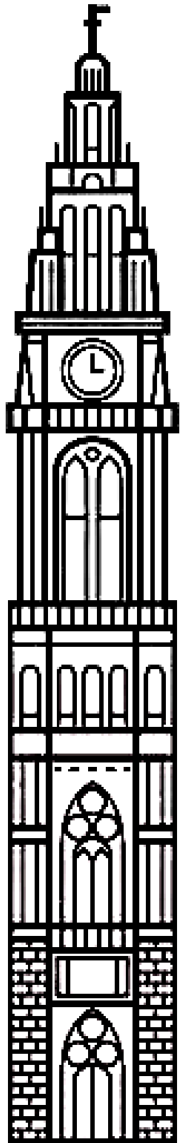
Idle network, sending different packet sizes one-by-one.

What's the best packet size?





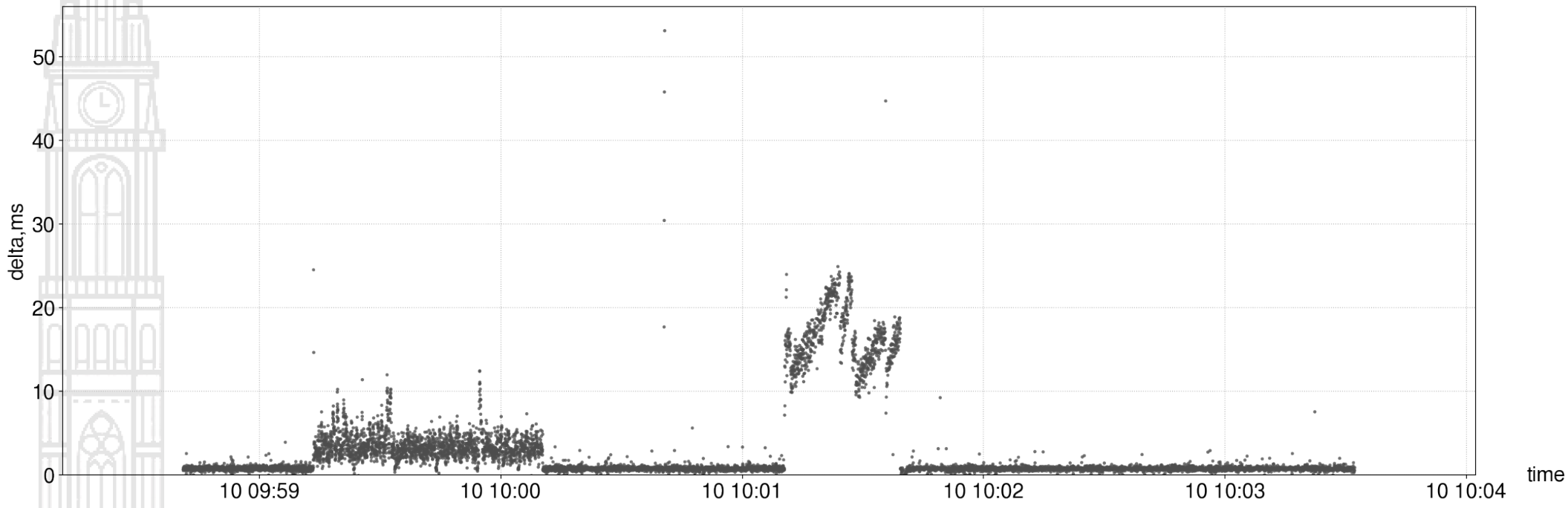
Idle network, sending different packet sizes one-by-one.
Filtered out negative values and network jitter.



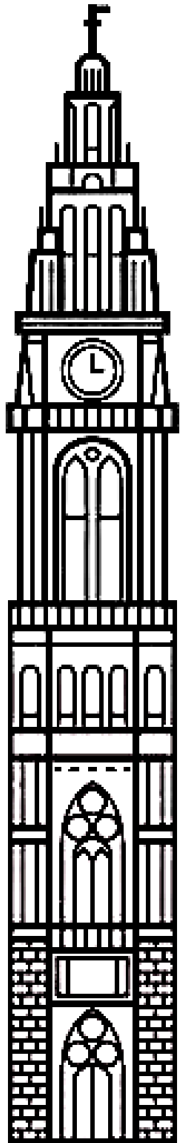
Estimating bandwidth remotely

$$\frac{(L+h)-(S+h)}{\Delta_L-\Delta_S}\times 2=\frac{L-S}{\Delta_L-\Delta_S}\times 2=\frac{16000-50}{0.00346-0.000789}\times 2=11943092\,Bps=95.54\,Mbps$$

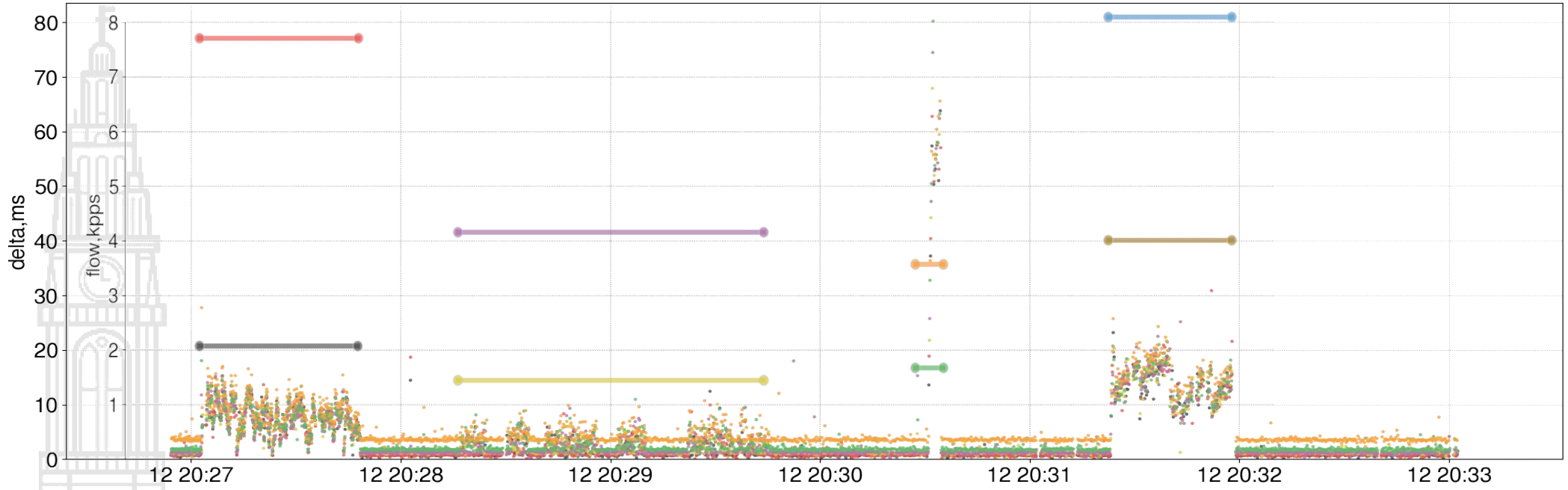
Speed test



Speed test being run from the target network.
Packet size: 50 bytes

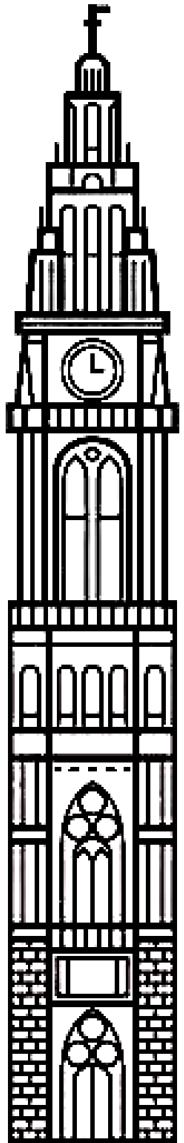


Traffic patterns

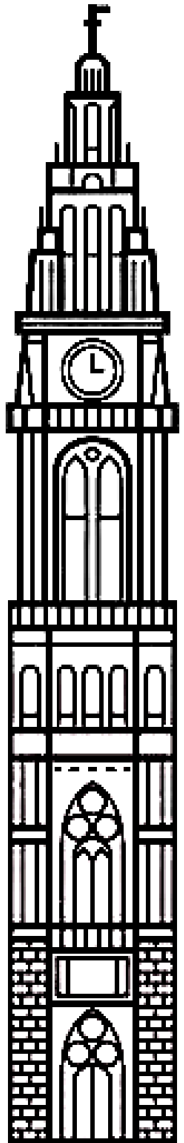


size,B
- 50
- 100
- 1000
- 2000
- 5000
- 16000

connection
- 1.2 Mbps target:49296->90.130.70.73:80/tcp
- 92.5 Mbps 90.130.70.73:80->target:49296/tcp
- 0.7 Mbps target:56346->80.233.168.140:443/udp
- 42.4 Mbps 80.233.168.140:443->target:56346/udp
- 0.6 Mbps target:50062->185.199.109.153:443/tcp
- 40.9 Mbps 185.199.109.153:443->target:50062/tcp
- 97.1 Mbps target:45664->90.130.70.73:80/tcp
- 1.6 Mbps 90.130.70.73:80->target:45664/tcp



Live demo?



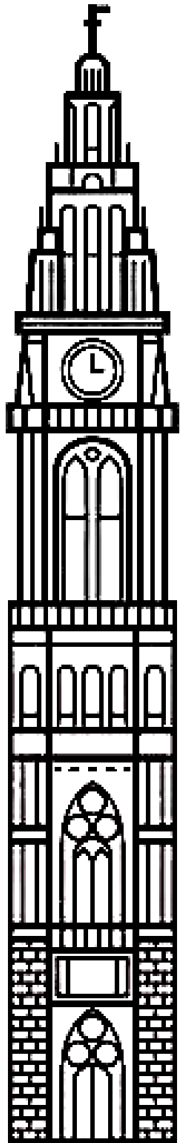
Any questions?

scan this pidgeon code
to access the source[1]

DEEPSEC

Kirils Solovjovs | DeepSec | 2024-11-22 | @k@chaos.social | kirils.org

[1] RFC 1149 compatible network device required



Support tools

https://github.com/0ki/presentation-toolkit

- (chart)

https://github.com/0ki/shellscripts

- (command line magic)