Always-on latency monitoring with eBPF

Toke Høiland-Jørgensen
Red Hat

Acknowledgements

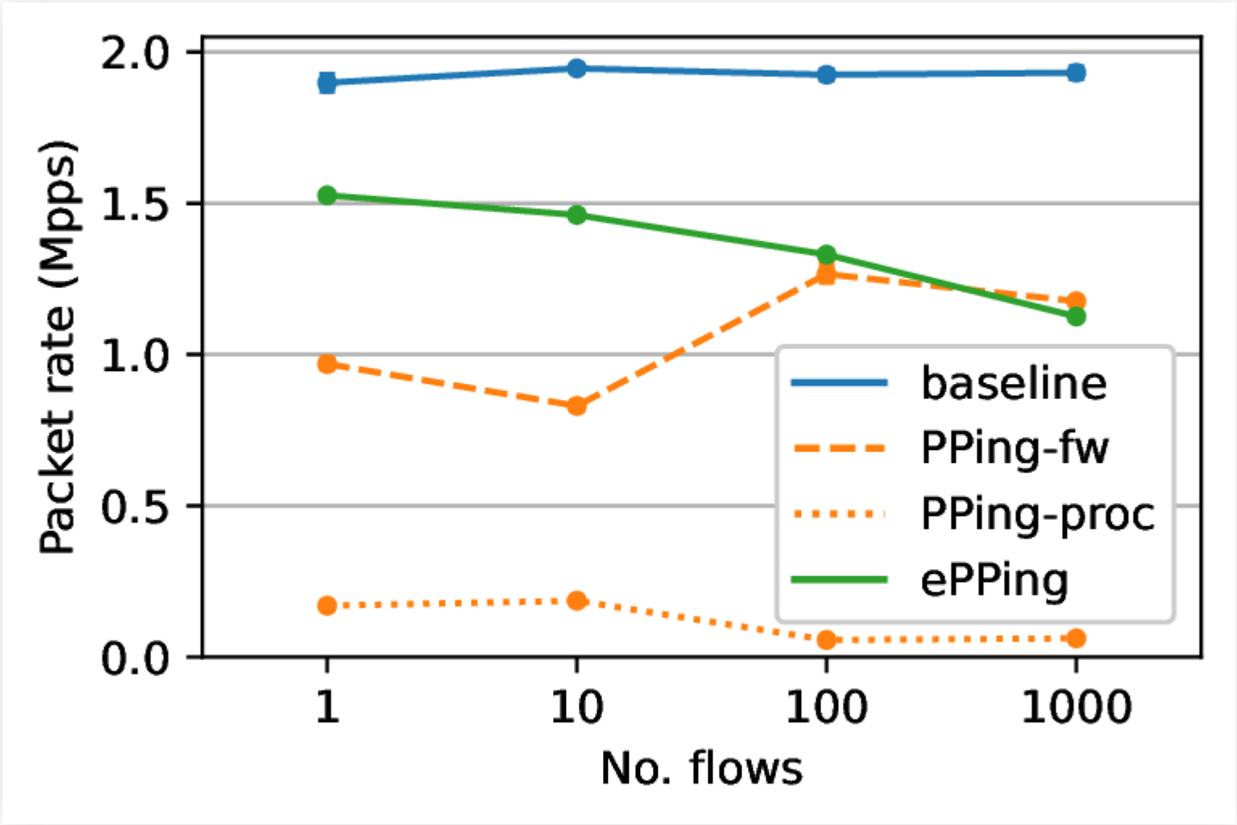
This is joint work with Simon Sundberg, Anna Brunstrom, Simone Ferlin-Reiter, Jesper Dangaard Brouer and Robert Chacón.

What is eBPF ?

- *Bytecode* - Architecture independent **Instruction Set**
  - *JIT* to native machine instructions (after loading into kernel)
- *Runtime environment* - Linux kernel
  - **Event based** BPF hooks all over the kernel
  - Per hook limited access to kernel functions via **helpers and kfuncs**
- ***Sandboxed*** by the eBPF *verifier*
  - Limits and verifies memory access and instructions limit

Passive latency monitoring

Kathie Nichols' pping + eBPF = ePPing



Sundberg, S. et al.: Efficient Continuous Latency Monitoring with eBPF. Passive and Active Measurement (PAM 2023).

ePPing in action

```
$ sudo ./pping -i eth0
Starting ePPing in standard mode tracking TCP on eth0
15:02:47.835948282 TCP 198.49.23.145:443+45.145.92.2:58188 opening due to first observed packet from dest
15:02:47.859153254 TCP 45.145.92.2:58188+198.49.23.145:443 opening due to first observed packet from dest
15:02:47.885873388 21.2851 ms 21.2851 ms TCP 198.49.23.145:443+45.145.92.2:58188
15:02:50.248235439 21.1454 ms 21.1454 ms TCP 198.49.23.145:443+45.145.92.2:58188
^C
```

Cool! But doesn’t really scale so well…

Enabling *always on* monitoring

```
$ sudo ./pping -i eth0 --aggregate 10
Starting ePPing in standard mode tracking TCP on eth0
Aggregating RTTs in histograms with 250 4 ms wide bins every 10 seconds
15:10:45.084560144: 198.49.23.0/24 -> rxpkts=53, rxbytes=102917, txpkts=51, txbytes=5233, rtt-count=4,
                                      min=21.1356 ms, mean=25 ms, median=24 ms, p95=29.4 ms, max=29.6763 ms
15:10:45.084560144: 45.145.92.0/24 -> rxpkts=113, rxbytes=71232, txpkts=114, txbytes=116857
15:10:45.084560144: 0.0.0.0/0 -> rxpkts=26, rxbytes=4456, txpkts=24, txbytes=4485
15:10:45.084560144: ::/0 -> rxpkts=19, rxbytes=3215, txpkts=19, txbytes=3215
15:10:45.118005597: TCP=(pkts=202, bytes=181238), UDP=(pkts=31, bytes=8720), ICMP=(pkts=22, bytes=2156),
                    ECN=(Not-ECT=196, ECT1=49, ECT0=10)
^C
```

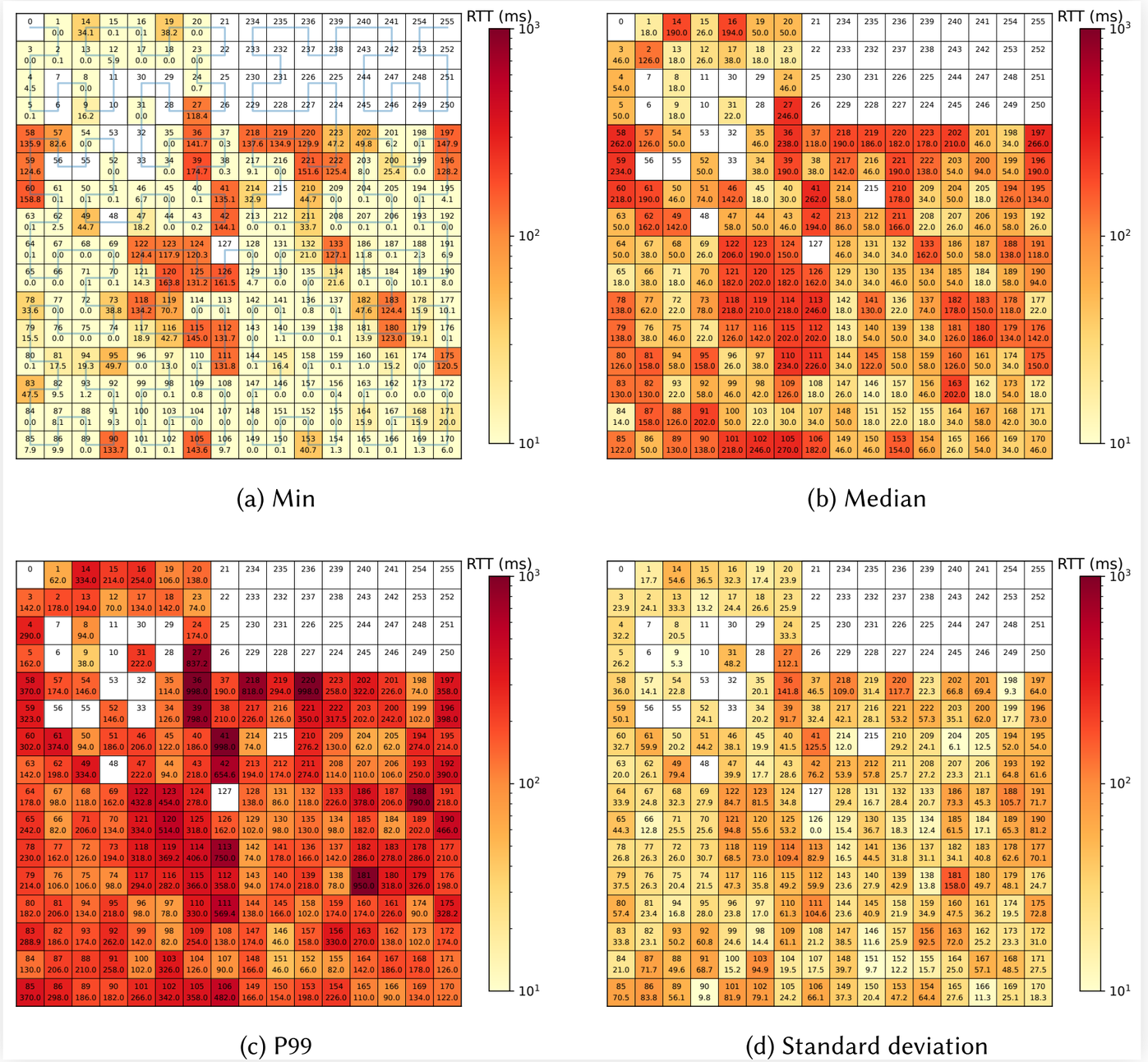
Histogram aggregation happens inside the kernel!

So what can we show with this?

One-month measurement study from a WISP middlebox (running LibreQoS)

Under submission

Subnet RTTs



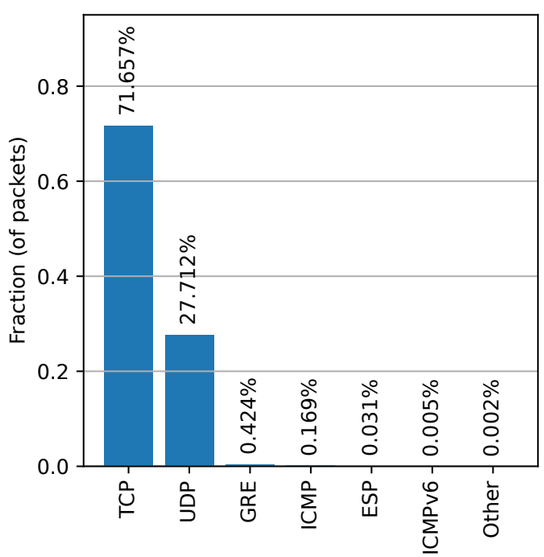
(a) Min

(b) Median

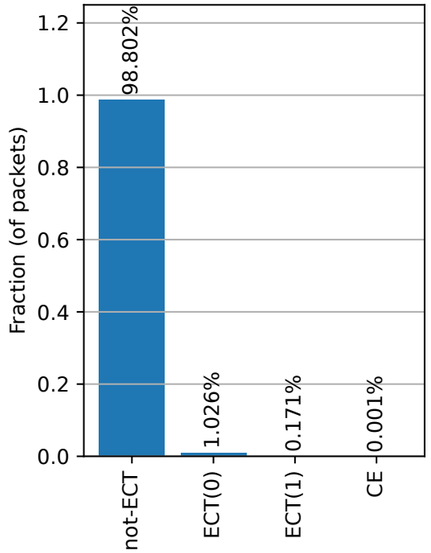
(c) P99

(d) Standard deviation

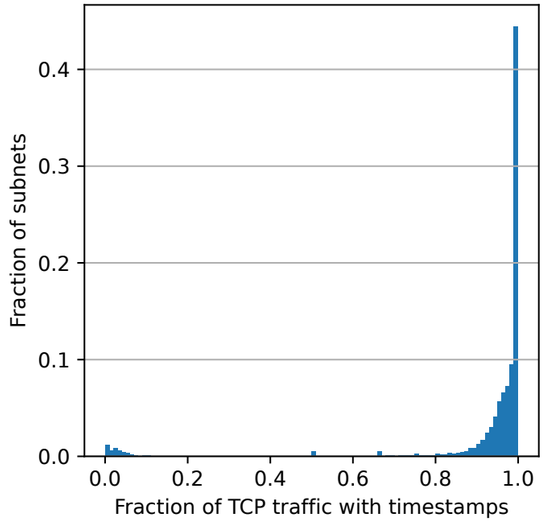
Traffic features



(a) Protocol occurrence

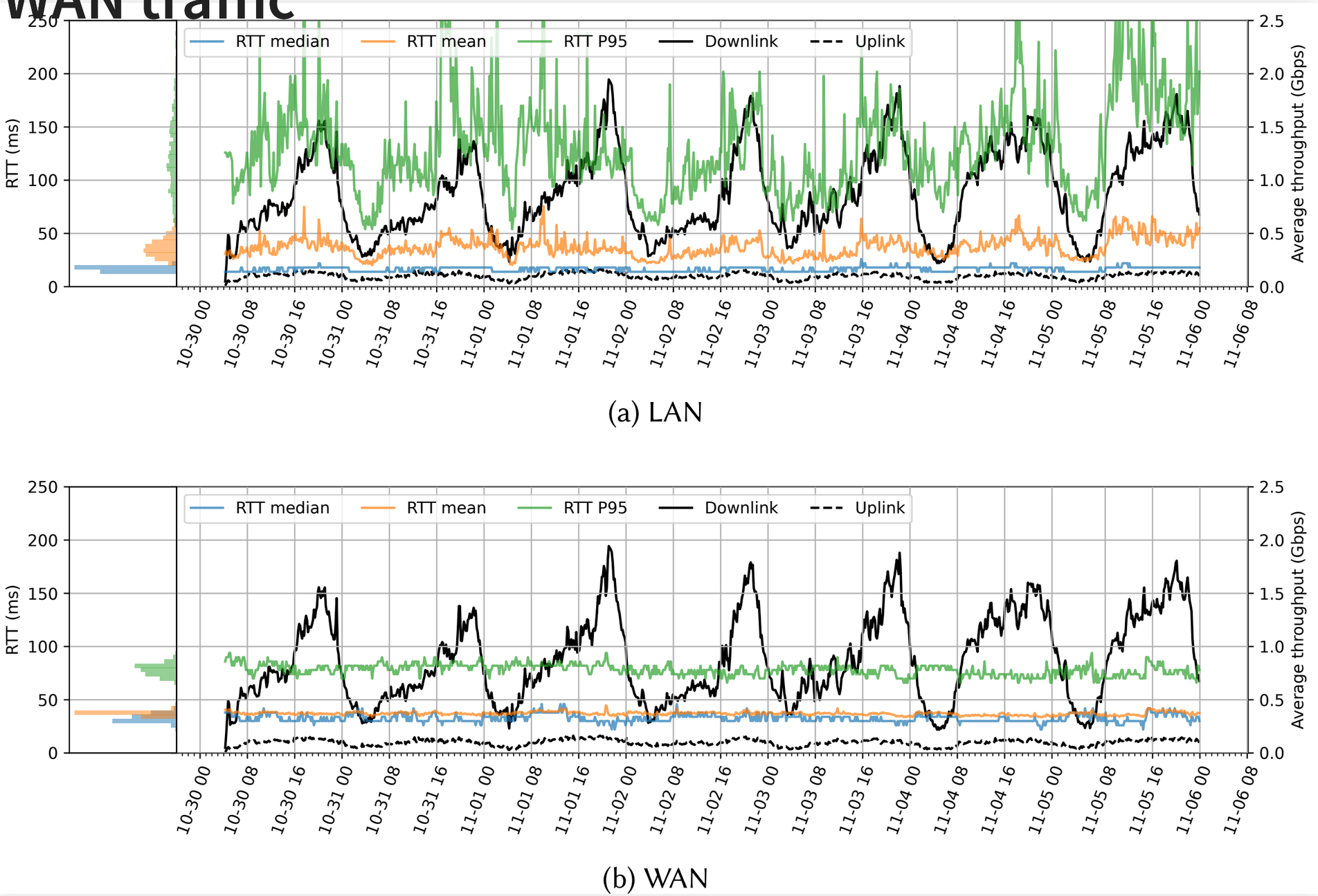


(b) ECN occurrence



(c) TCP timestamps

LAN and WAN traffic



(a) LAN

(b) WAN

Takeaways

With ePPing we can:

- Passively measure RTTs at scale on any Linux machine (server or middlebox)
- Aggregate metrics per subnet over the whole internet
- Investigate traffic characteristics on both the WAN and LAN side

What can this tell you about **your** network?

https://github.com/xdp-project/bpf-examples/tree/master/pping