waipu.tv

Probing your network with GRE and MPLS packets

Oliver Herms @ DENOG 10, <oliver.herms@exaring.de>

Agenda

1. Who am I?
2. Problem statement
3. Whitebox monitoring
4. Blackbox monitoring
5. Software probing your network
6. Experiences
7. Operational Conclusions
8. Questions

Who am I?

Oliver Herms

Senior Network Engineer @ EXARING AG

Friend of robustness, reliability, velocity

Network Automation Enthusiast

Golang and gRPC fanboy

Problem Statement

- How do you know your core network works as expected?
- How do you know if you are meeting your SLO?
  - Packet Loss
  - Latency
  - Jitter

Whitebox Monitoring

- ➔ Checking exposed metrics of devices
  - ◆ Error counters
  - ◆ Drop counters
  - ◆ Packet counters
- ➔ Does not work for Equipment not under your control (e.g. L2VPN)
- ➔ Does not show hidden packet drops (yes, they happen)
- ➔ Does not give you an idea about Latency issues
- ➔ Does not give you an idea about convergence times

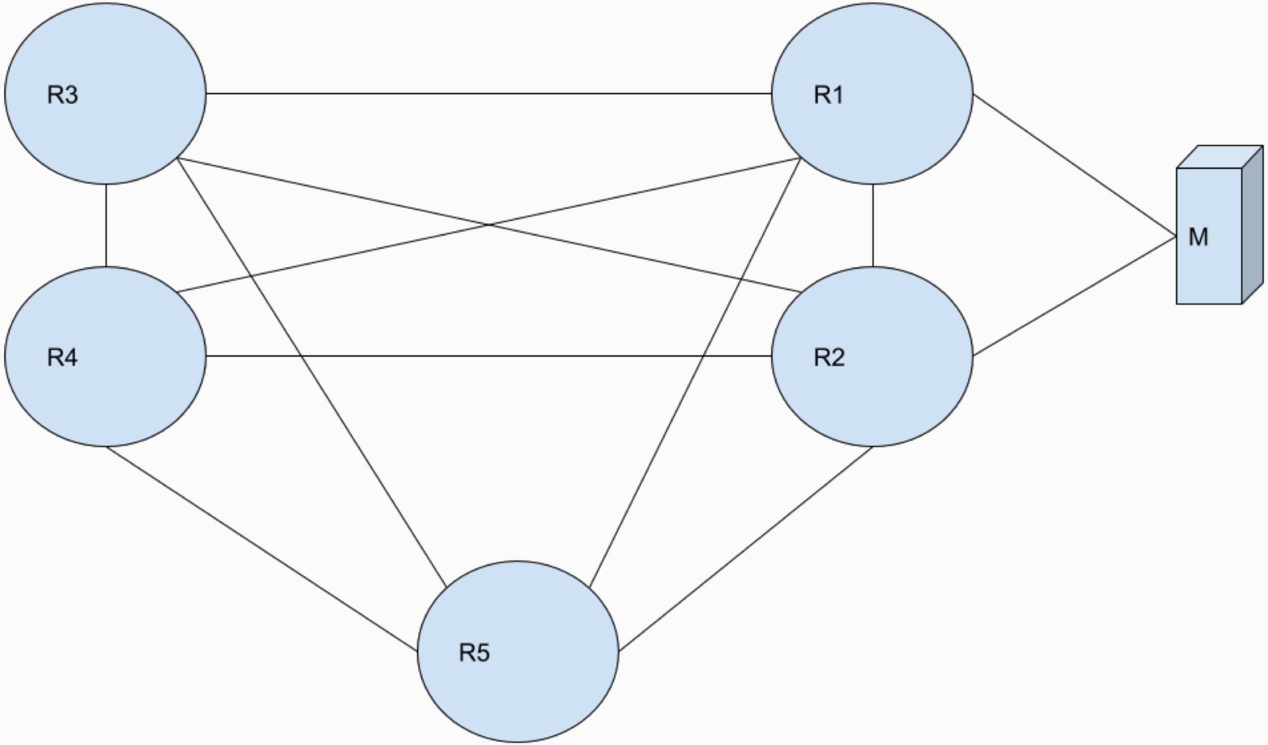
Blackbox Monitoring #1

- Actively sending test packets over the network (probes)
- Active probing allows you to get an idea about end to end
  - Packet loss
  - Latency
  - Jitter
- The more packets you send the better your information is
- Probing rate too low: Microbursts might not be catched
- Probing rate too high: Resource exhaustion on the network
  - Bandwidth mostly not a problem
  - CPUs

Blackbox Monitoring #2

- Who’s sending what packets where?
- Option A: Control Plane
  - CPU of Control Plane is too valuable for more than few PPS
- Option B: Some host to some host
  - You’re now testing machines too
  - Will probably cause false positive alerts
  - Machines often under someone else administration
- Option C: Deploy dedicated hosts everywhere
  - No problem if you have a cash cow
- Option D: Dedicated host to your network
  - How does that work?

Example Network

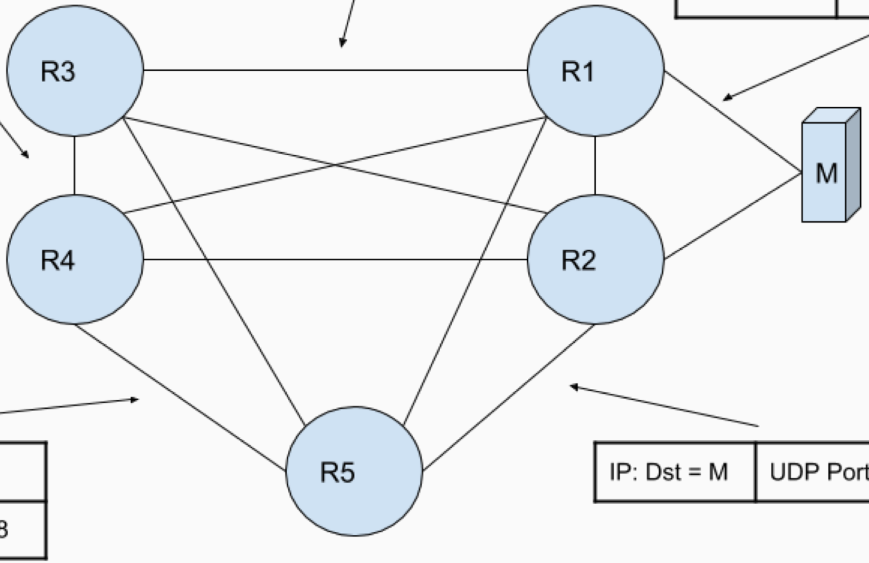


GRE Stacking to the rescue

IP: Dst = R4	GRE
IP: Dst = R5	GRE
IP: Dst = M	UDP Port 32768

IP: Dst = R3	GRE
IP: Dst = R4	GRE
IP: Dst = R5	GRE
IP: Dst = M	UDP Port 32768

IP: Dst = R1	GRE
IP: Dst = R3	GRE
IP: Dst = R4	GRE
IP: Dst = R5	GRE
IP: Dst = M	UDP Port 32768

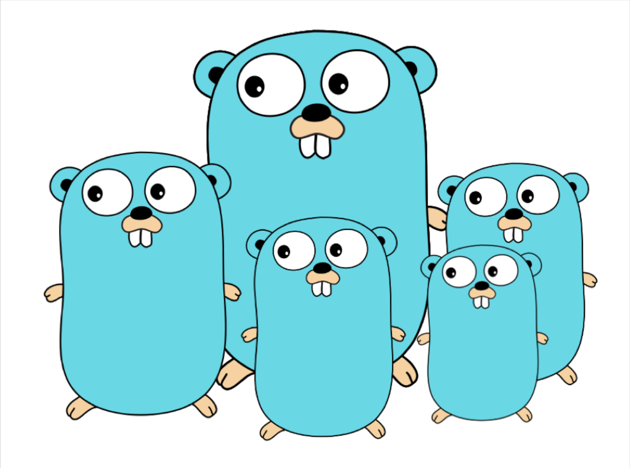


IP: Dst = R5	GRE
IP: Dst = M	UDP Port 32768

IP: Dst = M	UDP Port 32768

Matroschka Prober

- ➔ Just released for you! github.com/exaring/matroschka-prober
- ➔ Prober sending IP/GRE packets to routers
- ➔ Routers decapsulate the packets
- ➔ Inner IP header points back to the probing machine
- ➔ UDP packet containing a sequence number + timestamp returns



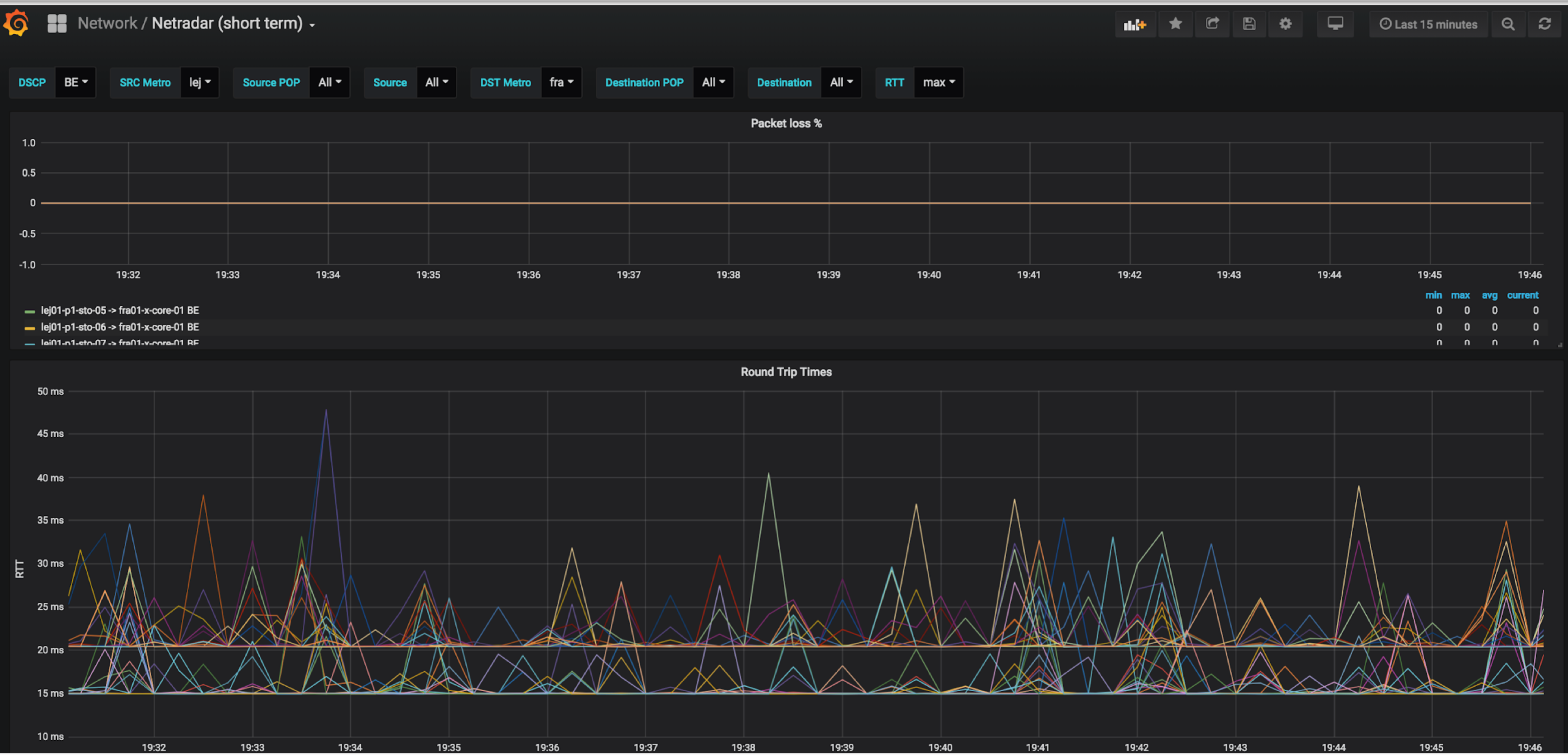
Matroschka Prober

- SRC IP-Addresses can be spoofed within IP subnets (default 169.254.0.0/16)
- DST IP-Addresses can be defined as IP subnets (/32 or shorter)
- Configurable Edge Loops core01.fra01-> core02.fra01 -> core01.dus01
- Configurable TOS/DSCP values (0x00)
- Configurable PPS rates (25)
- Configurable packet payload sizes (0)
- Configurable measurement durations (1000ms)
- Provides /metrics for Prometheus

Some tcpdump

```
oherms@lej01-p1-sto-01:~$ sudo tcpdump -n -i eth3 proto 47
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth3, link-type EN10MB (Ethernet), capture size 262144 bytes
18:50:47.466698 IP 10.255.0.204 > 10.201.0.128: GREv0, length 58: IP 10.255.0.204.51324 > 10.200.144.2.51396: UDP, length 26
18:50:47.466755 IP 10.255.0.180 > 10.25.0.255: GREv0, length 82: IP 10.255.0.180 > 10.1.0.255: GREv0, length 58: IP 10.255.0.180.51324 > 10.200.144.2.51504: UDP, length 26
18:50:47.468354 IP 10.255.0.135 > 10.25.0.255: GREv0, length 82: IP 10.255.0.135 > 10.11.2.7: GREv0, length 58: IP 10.11.2.7.51324 > 10.200.144.2.51489: UDP, length 26
18:50:47.468380 IP 10.255.0.135 > 10.1.1.255: GREv0, length 82: IP 10.255.0.135 > 10.11.2.7: GREv0, length 58: IP 10.11.2.7.51324 > 10.200.144.2.51475: UDP, length 26
```

Some graphs



Network / Netradar (short term)

Last 15 minutes

DSCP BE | SRC Metro lej | Source POP All | Source All | DST Metro fra | Destination POP All | Destination All | RTT max

Packet loss %

	min	max	avg	current
lej01-p1-sto-05 -> fra01-x-core-01 BE	0	0	0	0
lej01-p1-sto-06 -> fra01-x-core-01 BE	0	0	0	0
lej01-p1-sto-07 -> fra01-x-core-01 BE	0	0	0	0

Round Trip Times

Excursus: How to make the router decap packets (JunOS)

```
oherms@core01.fra01> show configuration firewall family inet filter NETRADAR
term DECAP {
    from {
        destination-address {
            10.1.0.255/32;
        }
        protocol gre;
    }
    then {
        decapsulate gre;
    }
}
term ACCEPT {
    then accept;
}
```

Excursus: How to make the router decap packets (Linux)

```
# ip tunnel add gre_decap mode gre local 192.0.2.0 ttl 255
```

```
# echo 1 > /proc/sys/net/ipv4/conf/ip_forwarding
```

Necessary if you can’t spoof addresses:

```
# echo 1 > /proc/sys/net/ipv4/conf/<input_dev>/accept_local
```

```
accept_local - BOOLEAN
        Accept packets with local source addresses. In combination
with
        suitable routing, this can be used to direct packets between two
        local interfaces over the wire and have them accepted properly.
        default FALSE
```

Experiences

- ➔ Disclaimer: The code on github is not what is running for a year in Exaring
- ➔ But it’s very close to it and the internal version will be deprecated soon
- ➔ What we’re running:
    - ◆ Measurement duration: 1s
    - ◆ PPS: 1600 from 8 Machines (200 each)
    - ◆ Hops: Mostly 1 explicit hop but also some 2 hops paths
    - ◆ Targets:
        - All Core Routers
        - All TOR Routers
        - AWS (it’s not as good as one might think)
    - ◆ Prometheus:
        - Scraping intervall: 1s
        - Retention time: 12h

Operational Conclusion

After one year running this in production we can say:

- ➔ In general a good idea
- ➔ Don’t tune too aggressively for alerting
- ➔ Great help for impact analysis in post mortems
- ➔ Routers don’t care at all

But what about MPLS?

- ➔ MPLS is another way to achieve this
- ➔ Our internal version supports stacking MPLS on top of GRE
- ➔ Soon to be in the open source version
- ➔ We also have plans for a prober that utilises IS-IS Segment Routing and automatically covers all circuits in the SR domain

Thank you for your attention!

Questions please!

github.com/exaring/matroschka-prober
Contributions welcome!

EXARING AG