# Telemetry for eXtended Reality services

Piotr Zuraniewski, Paweł Maćkowiak, Nassima Toumi,
Belma Turkovic, Simon Gunkel, Bart Gijsen (PM)

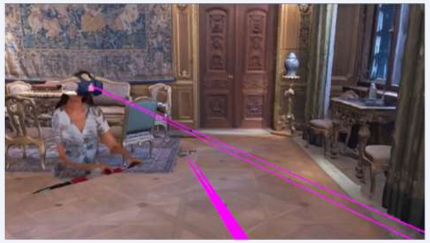
RON++ 2023-04-05

**ERP-SXR** Breaking the barriers of distance

# Social eXtended Reality (SXR)
# TNO Early Research Program (ERP) in 2019-23

- **Social interaction paradigm**, mediated by XR technologies, where individuals can experience social presence, and can engage in real-time interpersonal conversation and collaboration

- **Spatial presence**: feeling of being there
- **Social presence**: feeling of being with others
- **Self-embodiment**: feeling of body substitution
- **Agency**: feel that you can act in the environment
- **Multi-modal**: multiple modes of communication
- **Non-verbal cues**: expression, gaze, movement



Activity 1
**Photorealistic representation**

Activity 2
**Mediated Social Touch**

Activity 3
**Participants scale and device flexibility**

Activity 4
**Network-based media processing and transmission**

Activity 5
**Key factors for social presence in XR scenarios**

Source: [How Social XR (extended reality) reduces distances | TNO]

# Use-case class : expertise at a distance Three use-cases

Remote expert provides expertise to another person located at a given site,

- **Remote education**: The remote expert, in this case a teacher, brings knowledge to student(s) in AR.
- **Virtual training**: The remote expert, in this case a trainer, provides training to workers in a virtual environment reproducing the site (digital twin in VR).
- **Remote maintenance**: The remote expert, in this case a senior, bring knowledge and advice to a less senior person on site (AR).

## Use case class: eXpeRtise at a Distance

Remote education

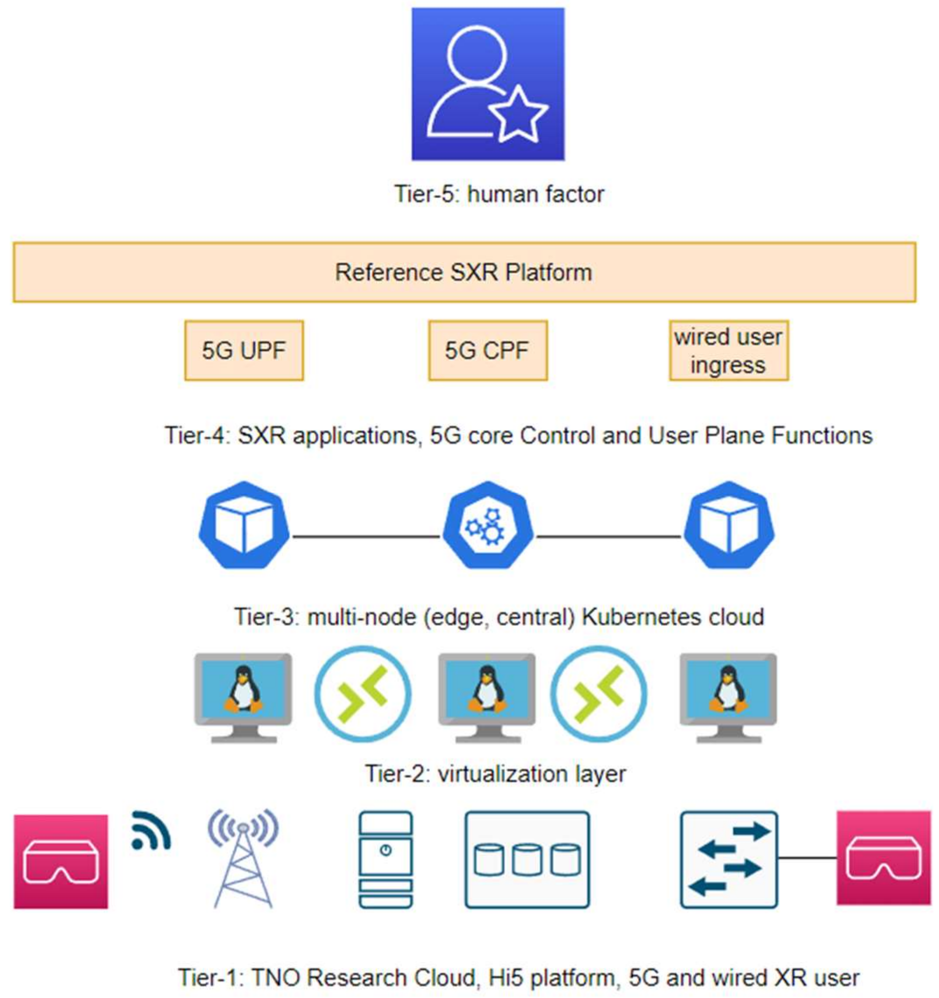
Virtual training

Remote maintenance

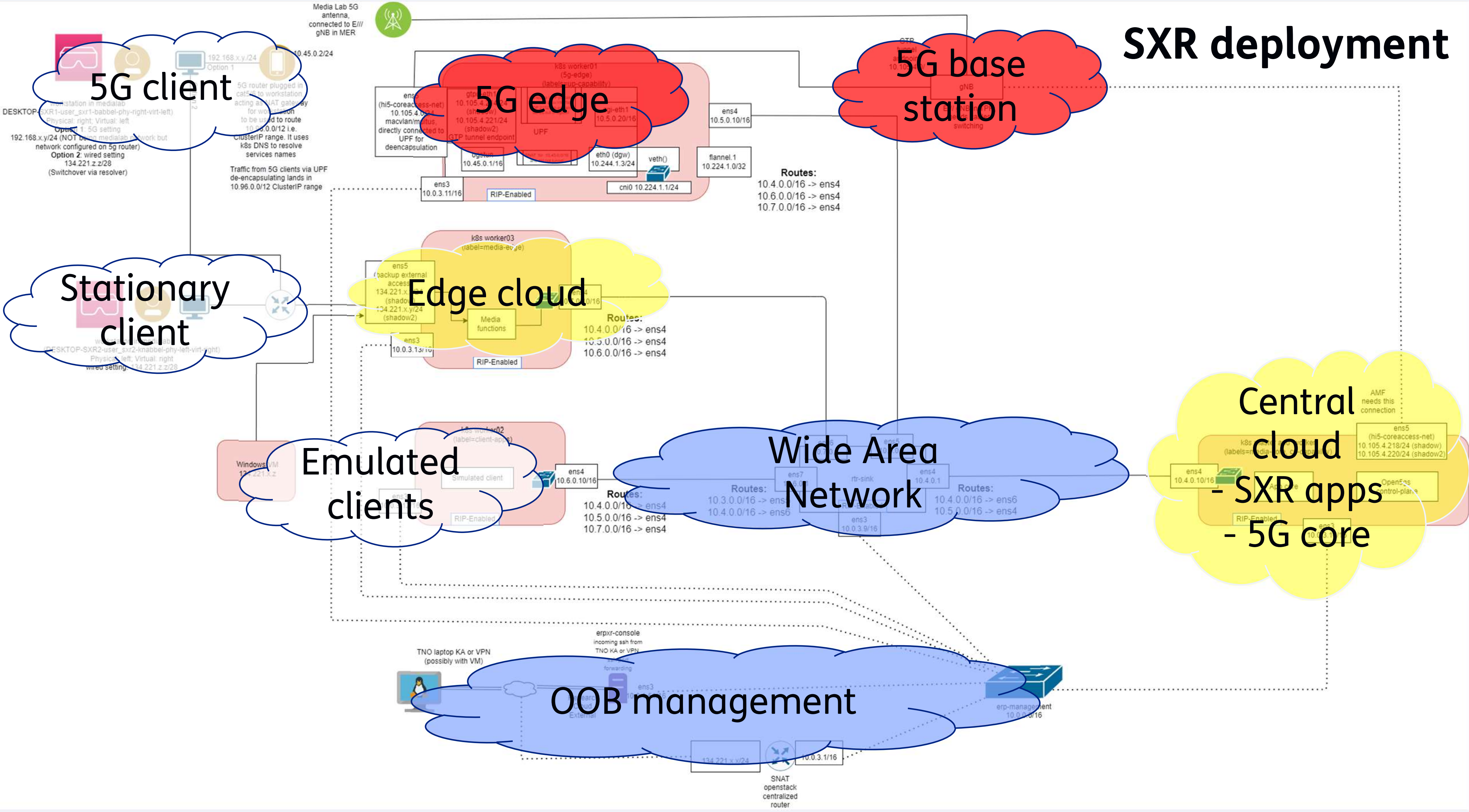
# TNO full stack XR developments

- TNO research in SXR covers complete stack
  - Human factor: QoE, ethics
  - Applications: XR-specific and cloud/5G related
  - Cloud orchestration, telemetry, AI-management
  - Virtualization
  - Hardware layer: XR-specific and cloud/5G related

- Social XR Platform utilizes shared TNO infrastructure
- TNO Research Cloud and Hi5 platform (Tier-1/2)
  - Developed outside of ERP SXR
  - Innovation enabler for multiple projects
  - Used in federated testbeds


Tier-5: human factor
Reference SXR Platform
5G UPF
5G CPF
wired user ingress
Tier-4: SXR applications, 5G core Control and User Plane Functions
Tier-3: multi-node (edge, central) Kubernetes cloud
Tier-2: virtualization layer
Tier-1: TNO Research Cloud, Hi5 platform, 5G and wired XR user

# SXR deployment


5G client
5G edge
5G base station
Stationary client
Edge cloud
Emulated clients
Wide Area Network
Central cloud
- SXR apps
- 5G core
OOB management
Media Lab 5G antenna, connected to E/// gNB in MER
10.45.0.2/24
192.168.x.y./24
Option 1
5G router plugged in cat5/6 to workstation, acting as NAT gateway for workstation to be used to route 10.96.0.0/12 i.e. ClusterIP range. It uses k8s DNS to resolve services names
workstation in medialab
DESKTOP-SXR1-user_sxr1-babbel-phy-right-virt-left)
Physical: right; Virtual: left
Option 1: 5G setting
192.168.x.y/24 (NOT being medialab network but network configured on 5g router)
Option 2: wired setting
134.221.z.z/28
(Switchover via resolver)
Traffic from 5G clients via UPF de-encapsulating lands in 10.96.0.0/12 ClusterIP range
ens5 (hi5-coreaccess-net) 10.105.4.0/24 macvlan/multus, directly connected to UPF for deencapsulation
k8s worker01 (5g-edge) (labels=up-capability)
gtpu-eth1 10.105.4.2../24 (shadow) 10.105.4.221/24 (shadow2) GTP tunnel endpoint
UPF
sgi-eth1 10.5.0.20/16
ogstun 10.45.0.1/16
eth0 (dgw) 10.244.1.3/24
veth()
cni0 10.224.1.1/24
ens3 10.0.3.11/16
RIP-Enabled
ens4 10.5.0.10/16
flannel.1 10.224.1.0/32
Routes:
10.4.0.0/16 -> ens4
10.6.0.0/16 -> ens4
10.7.0.0/16 -> ens4
GTP tunnel
gNB
switching
k8s worker03 (label=media-edge)
ens5 (backup external access 134.221.x.y/24 (shadow) 134.221.x.y/24 (shadow2)
Media functions
ens4 10.7.0.10/16
ens3 10.0.3.13/16
RIP-Enabled
Routes:
10.4.0.0/16 -> ens4
10.5.0.0/16 -> ens4
10.6.0.0/16 -> ens4
DESKTOP-SXR2-user_sxr2-knabbel-phy-left-virt-right)
Physical: left; Virtual: right
wired setting: 134.221.z.z/28
Windows VM 134.221.x.z
k8s worker02 (label=client-apps)
Simulated client
ens3
RIP-Enabled
ens4 10.6.0.10/16
Routes:
10.4.0.0/16 -> ens4
10.5.0.0/16 -> ens4
10.7.0.0/16 -> ens4
ens6
ens5
ens7
rtr-sink
ens4 10.4.0.1
RIP-Enabled
ens3 10.0.3.9/16
Routes:
10.3.0.0/16 -> ens6
10.4.0.0/16 -> ens6
Routes:
10.4.0.0/16 -> ens6
10.5.0.0/16 -> ens4
AMF needs this connection
ens5 (hi5-coreaccess-net) 10.105.4.218/24 (shadow) 10.105.4.220/24 (shadow2)
k8s master and worker (labels=media-core,cp-capability)
ens4 10.4.0.10/16
App core
Open5gs control-plane
RIP-Enabled
ens3 10.0.3.1
TNO laptop KA or VPN (possibly with VM)
erpxr-console incoming ssh from TNO KA or VPN
forwarding
ens3
research cloud External
erp-management 10.0.0.0/16
134.221.x.x/24
10.0.3.1/16
SNAT openstack centralized router

RON22

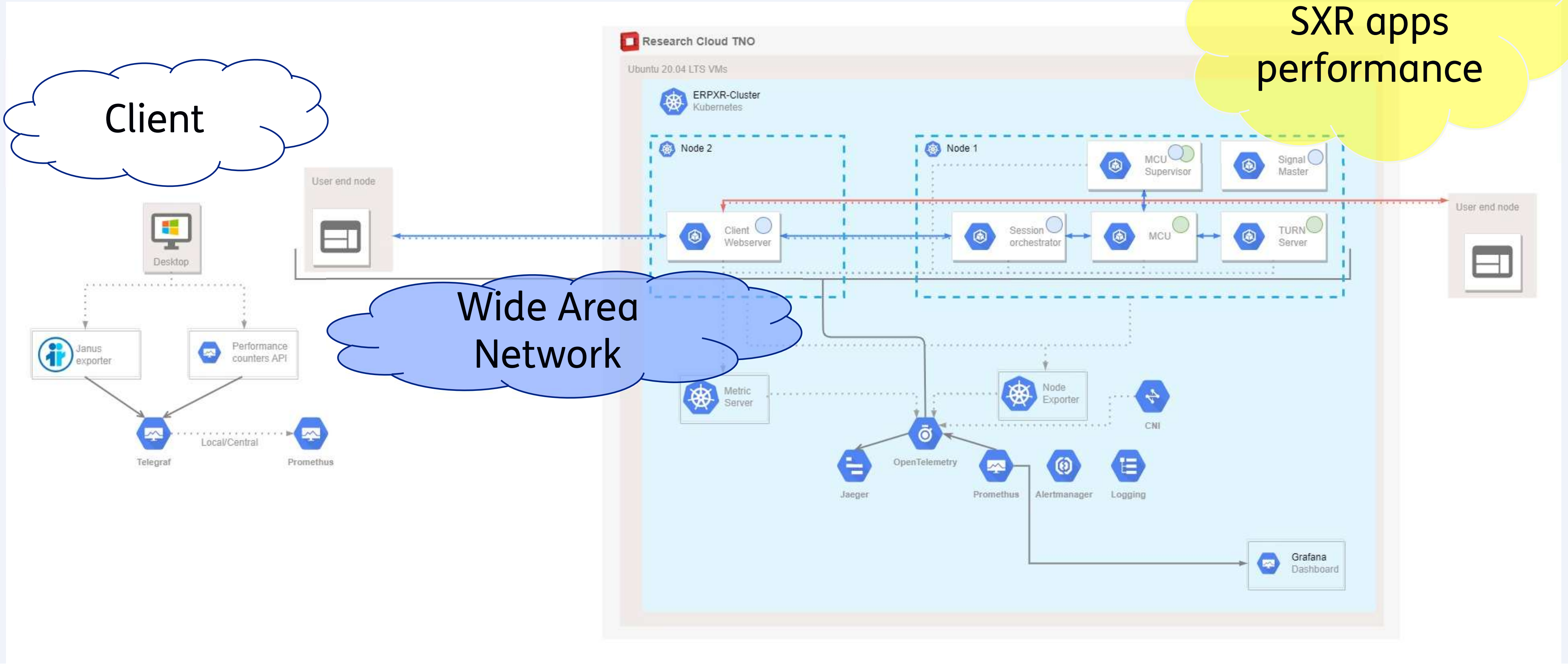
# RoN22 research objectives

- Investigate how programmable telemetry (RoN’20) can be composed into an end-to-end telemetry service that collects and correlates measurement data from network, cloud and application for performance demanding services such as SXR?
  - What is the state of the art?
  - How can telemetry functions be composed for providing end-to-end telemetry?
  - Can programmable network tech be used to (accurately) degrade network conditions in controlled experiments?

- Can end-to-end telemetry data be correlated to SXR QoE?
  - Which SXR relevant metrics can be measured by the end-to-end performance telemetry?
  - Is it feasible to collect data from all telemetry sources and calculate metrics in real-time?
  - Is telemetry information actionable (can it be used to re-program the underlying system to enhance QoE or prevent its degradation)?

RON22

# Telemetry architecture and instrumentation of the SXR demonstrator



Client

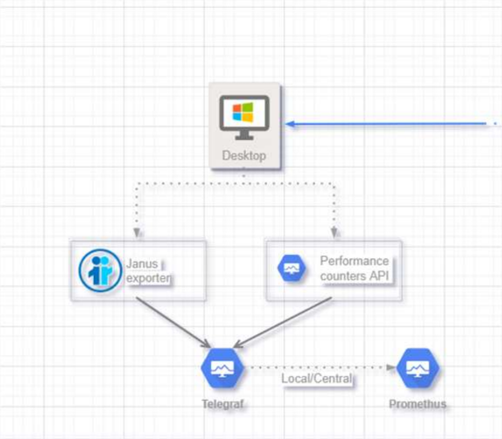
Wide Area Network

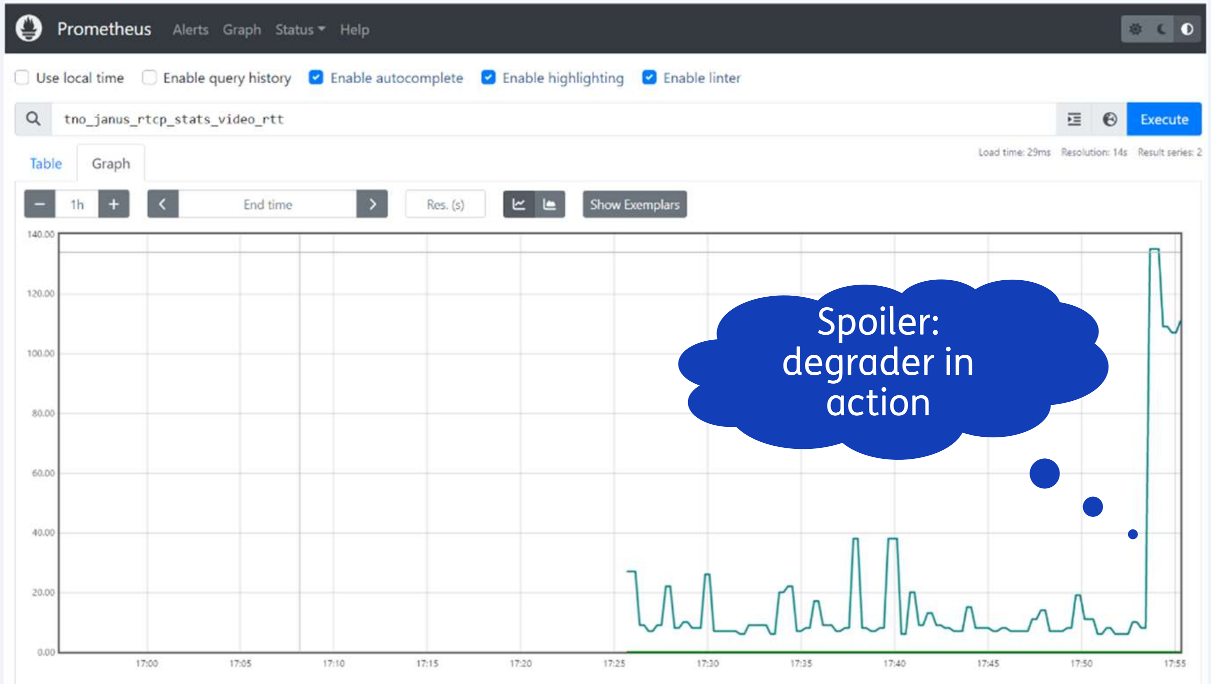
Cloud & SXR apps performance

RON22

# Client side telemetry

- Downlink/uplink parameters
  - Bandwidth
  - Packet loss
  - Delay
- Source: WebRTC Janus server
  - Custom Janus -> Prometheus exporter
- End-to-end measurements
  - From client to WebRTC server
  - Audio, video
- Problems: for unknow reason, only downlink stats reported
- Not done: Windows performance counters
  - Needs Telegraph-like app to export





□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

```
tno_janus_rtcp_stats_video_rtt
```

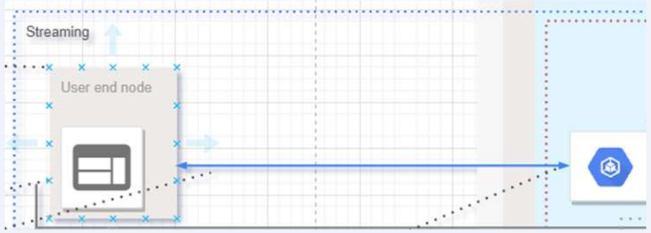
Load time: 29ms Resolution: 14s Result series: 2

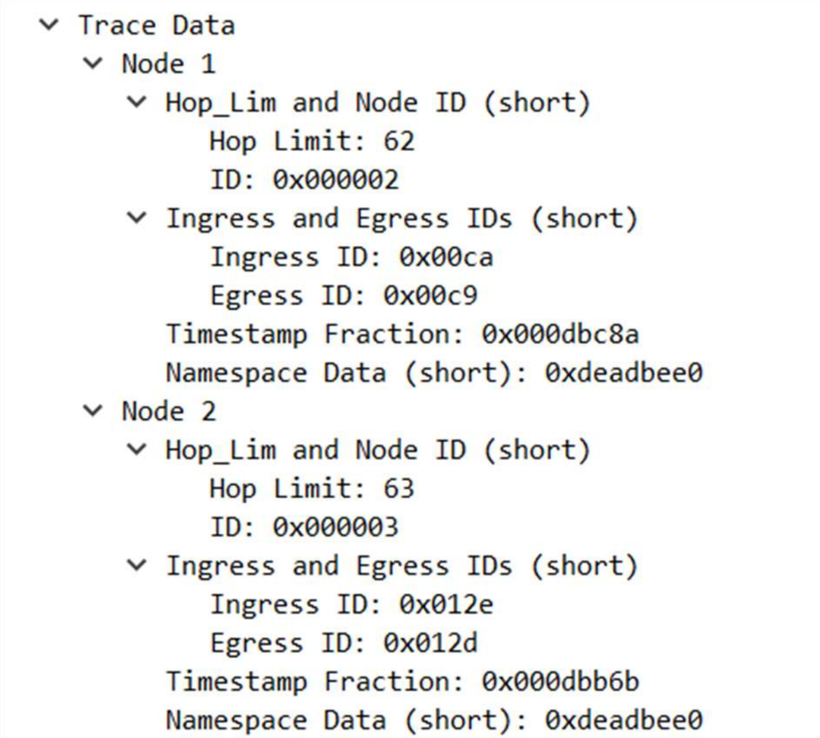
Spoiler: degrader in action

RON22

# Wide Area Network telemetry

- iOAM: In-Situ Operations, Administration, and Maintenance
- iOAM for IPv6, uses Hop by Hop header;
  - Node ID
  - Engress/egress interface ID
  - Timestamp,...
- Problems:
  - Initially hoped to use RON’20 VPP fd.io iOAM contribution
  - Patch not merged, developers not responsive, VPP project less active ?





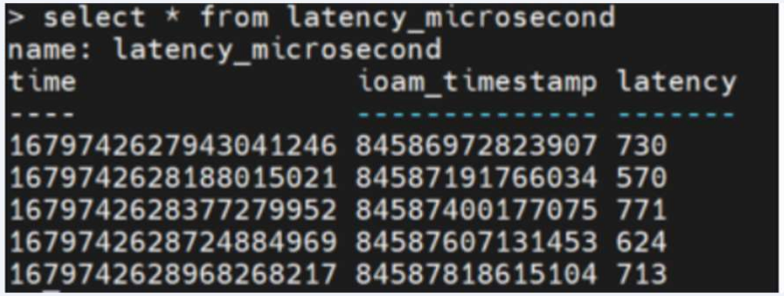
```
v Trace Data
  v Node 1
    v Hop_Lim and Node ID (short)
        Hop Limit: 62
        ID: 0x000002
    v Ingress and Egress IDs (short)
        Ingress ID: 0x00ca
        Egress ID: 0x00c9
      Timestamp Fraction: 0x000dbc8a
      Namespace Data (short): 0xdeadbee0
  v Node 2
    v Hop_Lim and Node ID (short)
        Hop Limit: 63
        ID: 0x000003
    v Ingress and Egress IDs (short)
        Ingress ID: 0x012e
        Egress ID: 0x012d
      Timestamp Fraction: 0x000dbb6b
      Namespace Data (short): 0xdeadbee0
```

RON22

# Wide Area Network telemetry

- Used Linux kernel iOAM implementation
  - Needs modern kernel (5.18+, 6.0.0 used)
  - Needs some tweaks (MTU, *iproute2,...*)
  - Certain parameters like path length need to be known in advance

- iOAM data collection and parsing with custom XDP/eBPF programs
  - Can pick any value from any header as condition
  - Allows for very specific measurements
  - High precision (us) latency stats stored in InfluxDB.
  - Bug in *bpf_ktime_getns()* : retrieving value from last hop problematic
    - Provides time from boot, not epoch



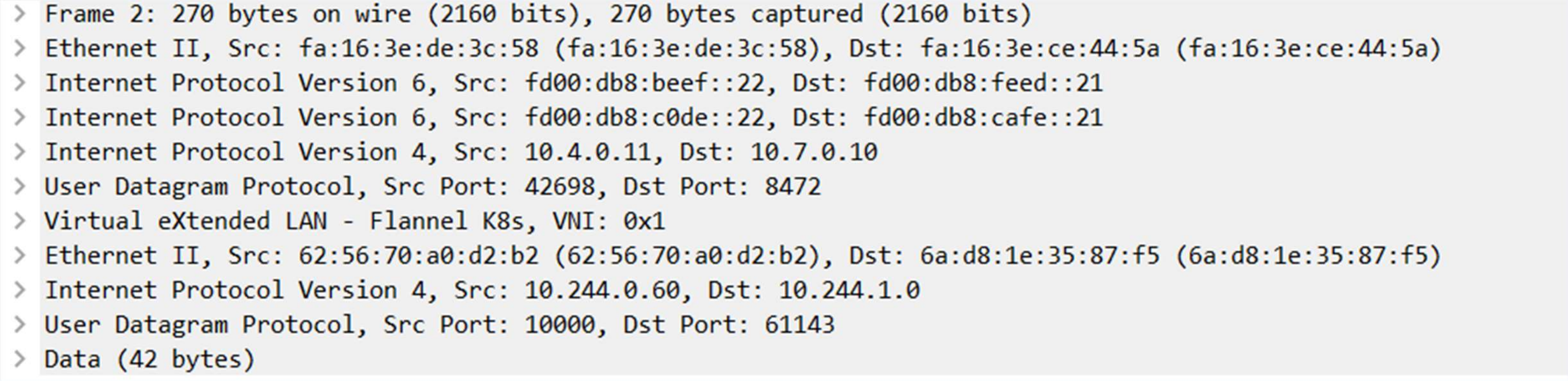
```
> select * from latency_microsecond
name: latency_microsecond
time                ioam_timestamp latency
----                -------------- -------
1679742627943041246 84586972823907 730
1679742628188015021 84587191766034 570
1679742628377279952 84587400177075 771
1679742628724884969 84587607131453 624
1679742628968268217 84587818615104 713
```

TNO innovation for life

RON22

# ...but watch your MTU :-O

- In our case: IPv4 WAN connects central and edge Kubernetes cloud
- ...which introduces its own VXLAN overlay
- ...which is encapsulated in IPv6
- ...which needs another IPv6 because it seems hop-by-hop cannot be inserted in existing IPv6 header
- Custom parser picks flows based on (innermost) UDP port

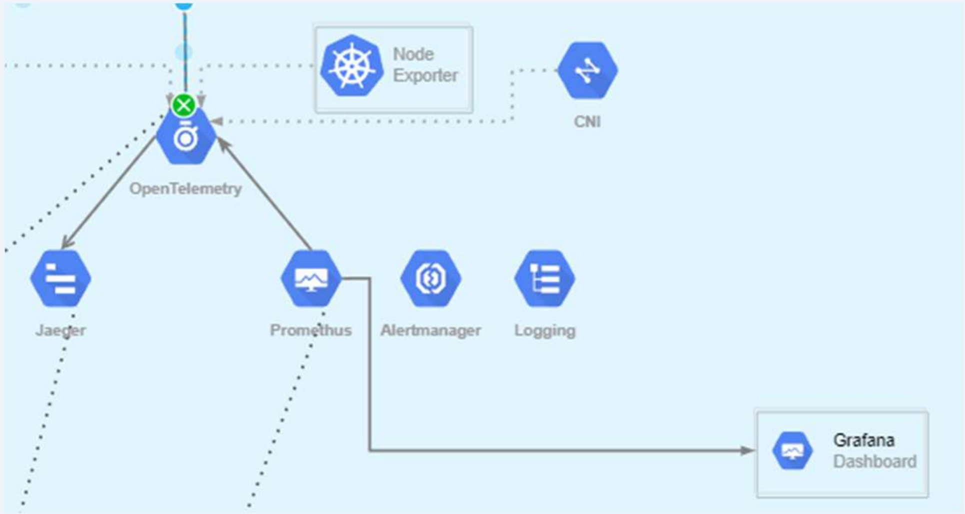


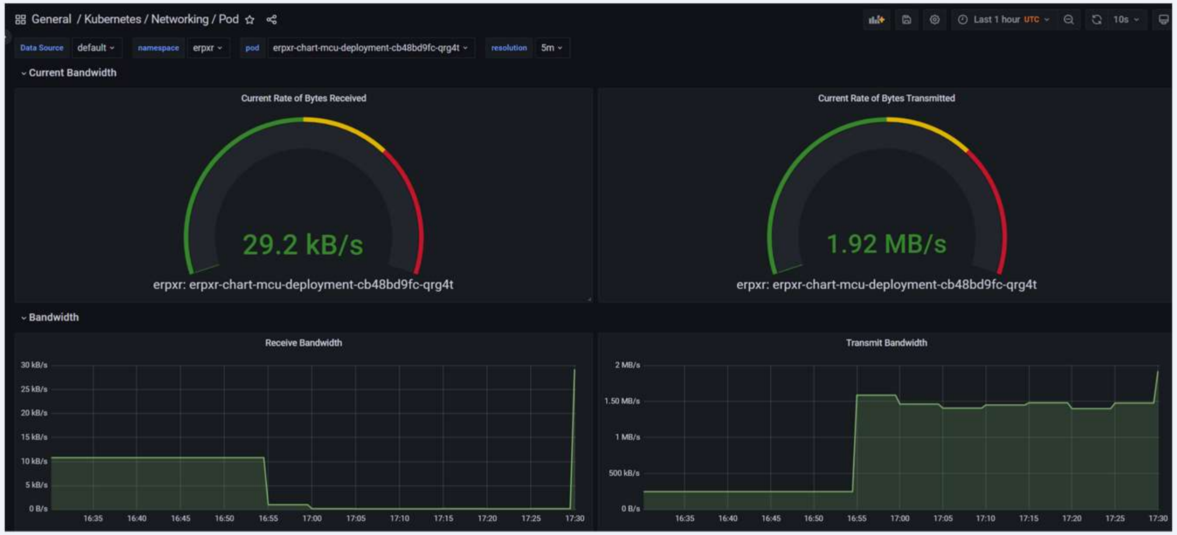
```
> Frame 2: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits)
> Ethernet II, Src: fa:16:3e:de:3c:58 (fa:16:3e:de:3c:58), Dst: fa:16:3e:ce:44:5a (fa:16:3e:ce:44:5a)
> Internet Protocol Version 6, Src: fd00:db8:beef::22, Dst: fd00:db8:feed::21
> Internet Protocol Version 6, Src: fd00:db8:c0de::22, Dst: fd00:db8:cafe::21
> Internet Protocol Version 4, Src: 10.4.0.11, Dst: 10.7.0.10
> User Datagram Protocol, Src Port: 42698, Dst Port: 8472
> Virtual eXtended LAN - Flannel K8s, VNI: 0x1
> Ethernet II, Src: 62:56:70:a0:d2:b2 (62:56:70:a0:d2:b2), Dst: 6a:d8:1e:35:87:f5 (6a:d8:1e:35:87:f5)
> Internet Protocol Version 4, Src: 10.244.0.60, Dst: 10.244.1.0
> User Datagram Protocol, Src Port: 10000, Dst Port: 61143
> Data (42 bytes)
```

RON22

# Kubernetes cloud telemetry

- Cluster level (VM)
  - Node CPU/RAM/Network Interfaces ,...
- Networking within cluster
  - To/from/between services and pods
  - Bandwidth/packets/dropped packets/...
  - Certain k8s network fabrics offer good flow-level insight
  - Example: Hubble Flows
  - Drawback: vendor specific, cannot change





General / Kubernetes / Networking / Pod

Last 1 hour UTC · 10s

Data Source: default · namespace: erpxr · pod: erpxr-chart-mcu-deployment-cb48bd9fc-qrg4t · resolution: 5m

Current Bandwidth

Current Rate of Bytes Received: 29.2 kB/s — erpxr: erpxr-chart-mcu-deployment-cb48bd9fc-qrg4t

Current Rate of Bytes Transmitted: 1.92 MB/s — erpxr: erpxr-chart-mcu-deployment-cb48bd9fc-qrg4t

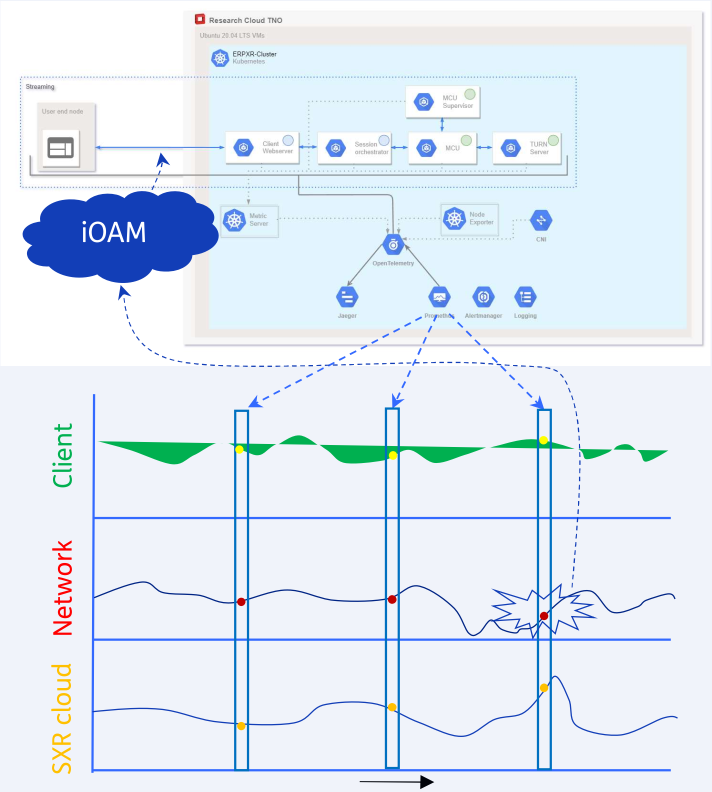
Bandwidth

RON22

# ‘Zoom-in’ telemetry

- Data from various sources (k8s, apps) collected in Prometheus
- Typical Prometheus scraping interval period: 5-15 seconds
- Alert: 2-3x scraping interval
- QoE telemetry should be more fine-grained: < 200 ms

- Our case: iOAM activated on demand
  - REST API awaits calls
    - UDP client port, VM interface
  - Compiles and inserts kernel module
  - As result, only very specific flow is selected
  - Exports *us*-precision data to influxDB
  - Easily extended to be activated by e.g., Prometheus alert

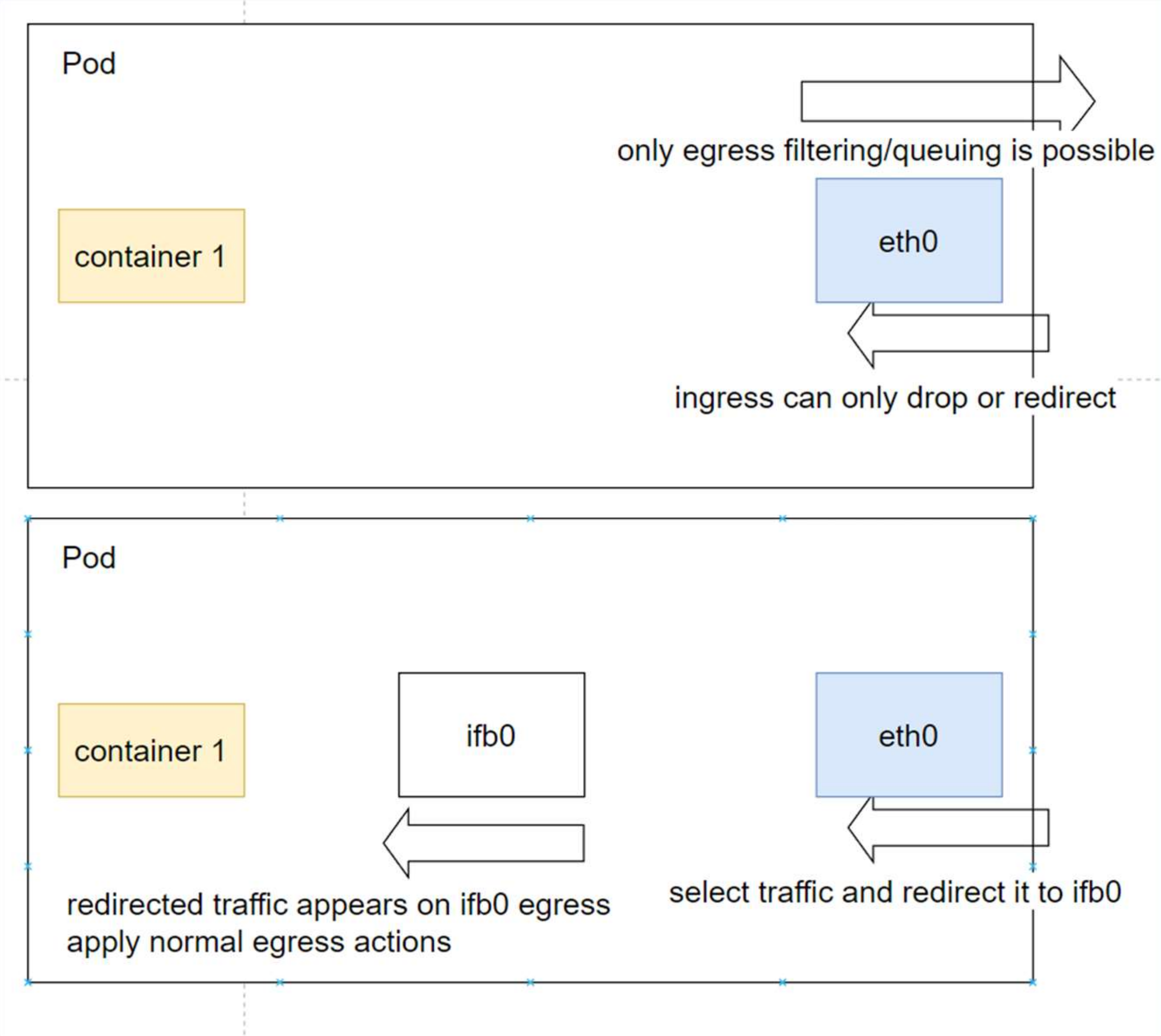


innovation for life

RON22

# Degrader for controlled experiments

- For Kubernetes
  - Depending on k8s network fabric, standard linux tools like *tc* may work (calico) or may not work (cilium)
  - *chaosmesh* deploys *tc* etc. as k8s resources
  - However, it only works for egress traffic (from pod point of view)
  - Using *chaosmesh nsexec* module we are able to inject commands into network namespace utilized by Kubernetes pod to perform ingress degradation
    - We use *IFB* interfaces, *mirred* , *u32, netem* etc.


Pod
only egress filtering/queuing is possible
container 1
eth0
ingress can only drop or redirect
Pod
container 1
ifb0
eth0
redirected traffic appears on ifb0 egress
apply normal egress actions
select traffic and redirect it to ifb0

RON22

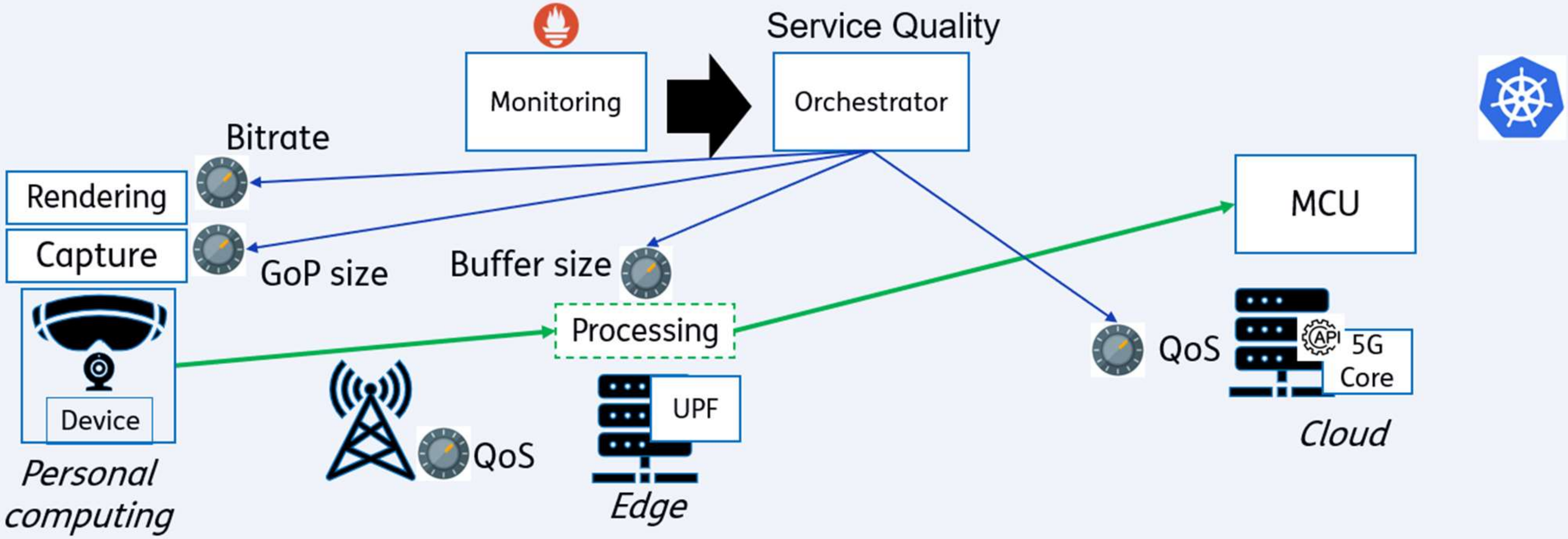
# Application: SXR Quality of experience

- More fine-grained telemetry enable more fine grained QoE experiments:

| | Functionality … | … enables experiment with research question |
|---|---|---|
| 1. | Implementation of degrader | Does telemetry & degrader instrumented SXR system behave as expected? |
| 2a. | Noticeability of degradation | Is degradation of QoS parameters (delay, packet loss, bandwidth) noticeable by user? |
| 2b. | Effect of degradation | Does gradual degradation of QoS parameters also lead to gradual decrease in QoE? |
| 3. | Degradation thresholds | What are the threshold of the different QoS parameters up to which the system performs as expected or up to which the system is still usable? |
| 4. | Parameter importance | Degradation of what QoS parameters have the largest effect on the QoE? |

# Application: network slice management

- Slice adaptation based on cross-layer aspects:
  - Uplink Media pipeline supports dynamic encoding parameter and buffering changes
  - Network support dynamic QoS changes with standard APIs
  - Apply adaptation actions in both domains based on the output of the orchestrator algorithm
- Work-in-progress: active slice management based on telemetry data

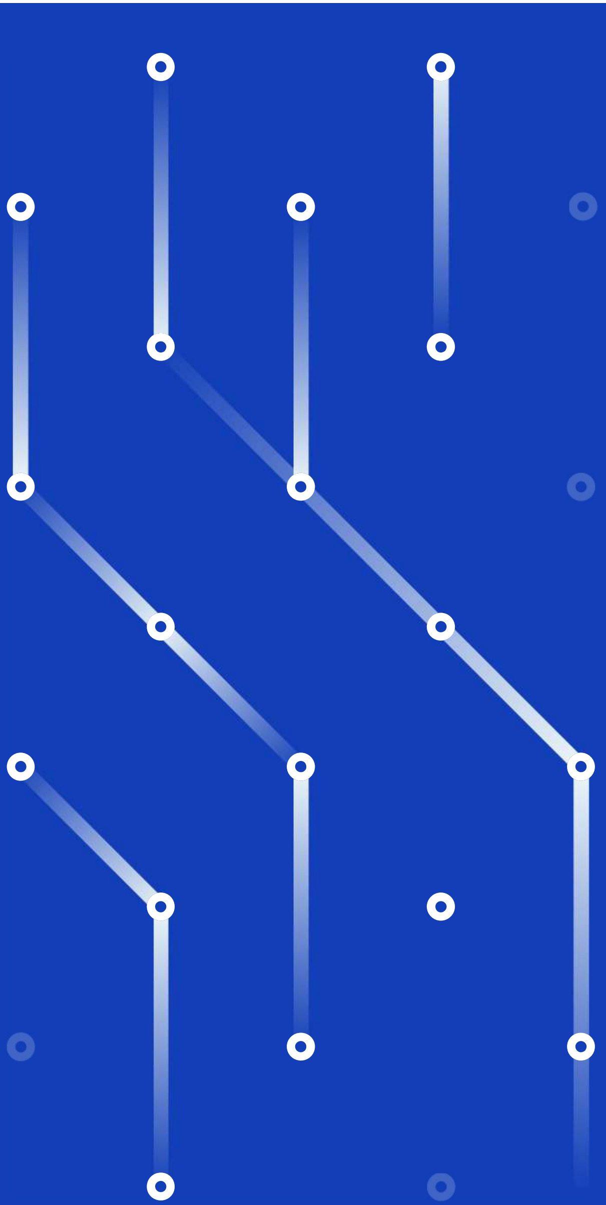


# Summary

- TNO SXR platform with telemetry continues to be developed
- Collaboration opportunities:
  - Horizon EU / ITEA
  - National Growth Funds (Nationaal Groeifonds), Future Network Services
  - TKI https://dutchdigitaldelta.nl/
  - Federated Lab
  - Master students

- Plans
  - Further development of telemetry functions and its utilization (e.g., integrating application data)
  - QoE experiments

- Piotr Zuraniewski | piotr.zuraniewski@tno.nl , LinkedIn ---->
- Bart Gijsen | bart.gijsen@tno.nl



TNO innovation for life

This project is co-funded by Holland High Tech
With  a PPP Grant for Research and Innovation
in Top Sector HTSM.