LFEnergy SEAPATH - svtrace Tools for Latency Analysis in Virtualized Networking Platforms

FOSDEM 2025 – February 2, 2025

Paul Le Guen de Kerneizon – Embedded software engineer

paul.leguendekerneizon@savoirfairelinux.com

Savoir-faire LINUX®

23+

years in industrial product engineering in many areas

Savoir-faire Linux is a team of experts in Free, Libre and Open Source technologies in Canada and Europe.

LF ENERGY

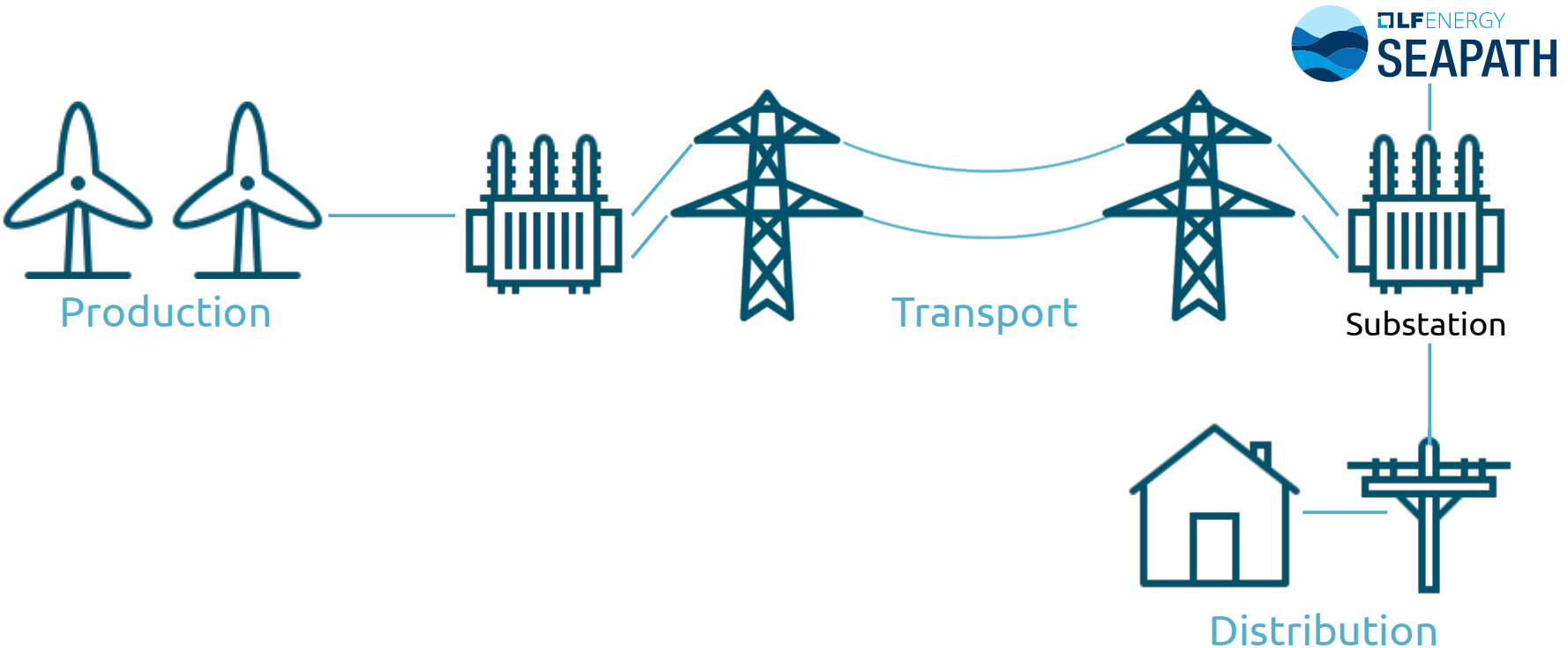
GENERAL MEMBER

THE LINUX FOUNDATION SILVER MEMBER

yocto PROJECT SILVER MEMBER

A quick presentation of the SEAPATH project

Context


LFENERGY
SEAPATH
Production
Transport
Substation
Distribution

Context

Industrial grade Open Source Software hypervisor designed for digital substation

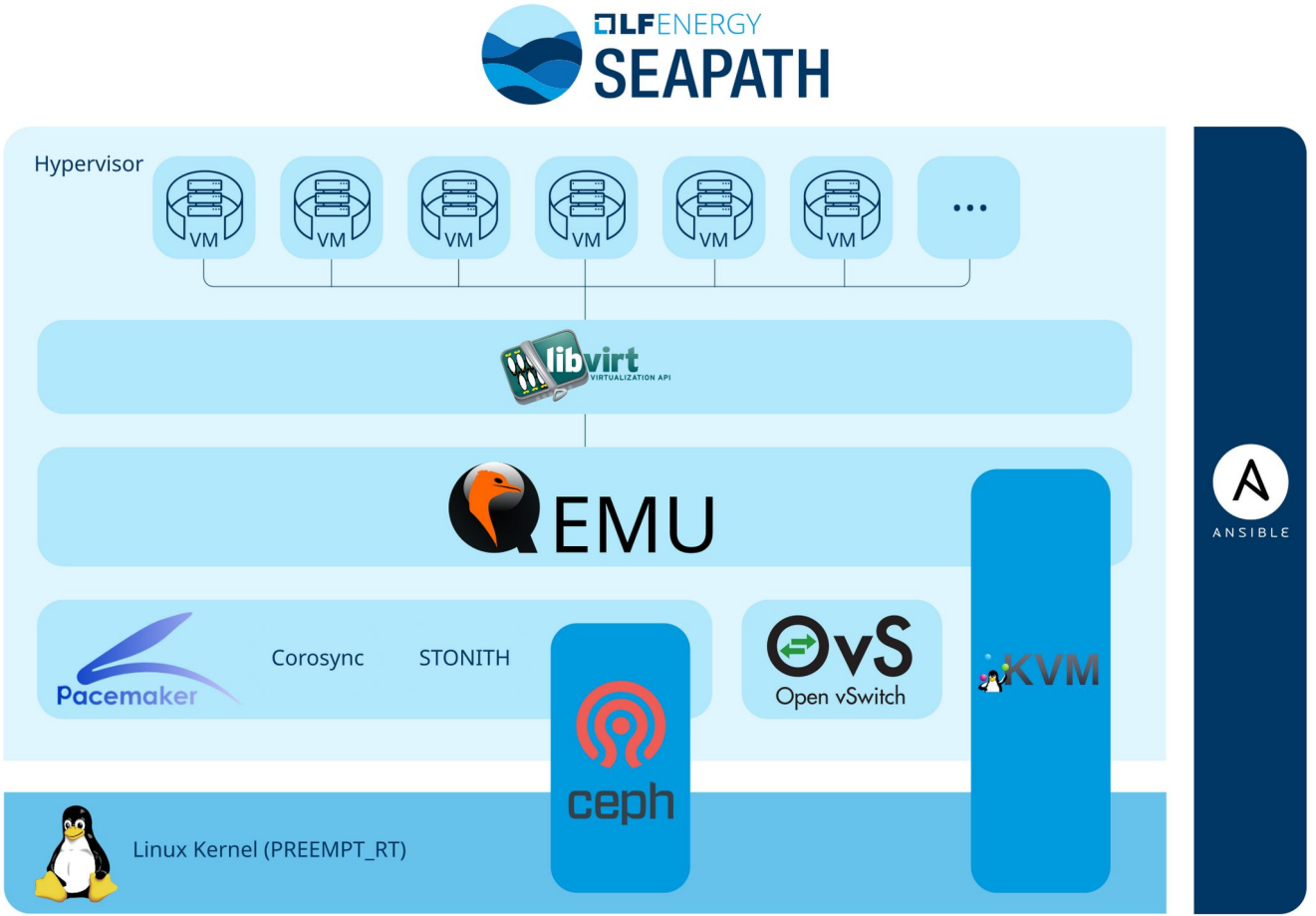
+

Hosting virtualized automation and protection applications with Real Time constraints

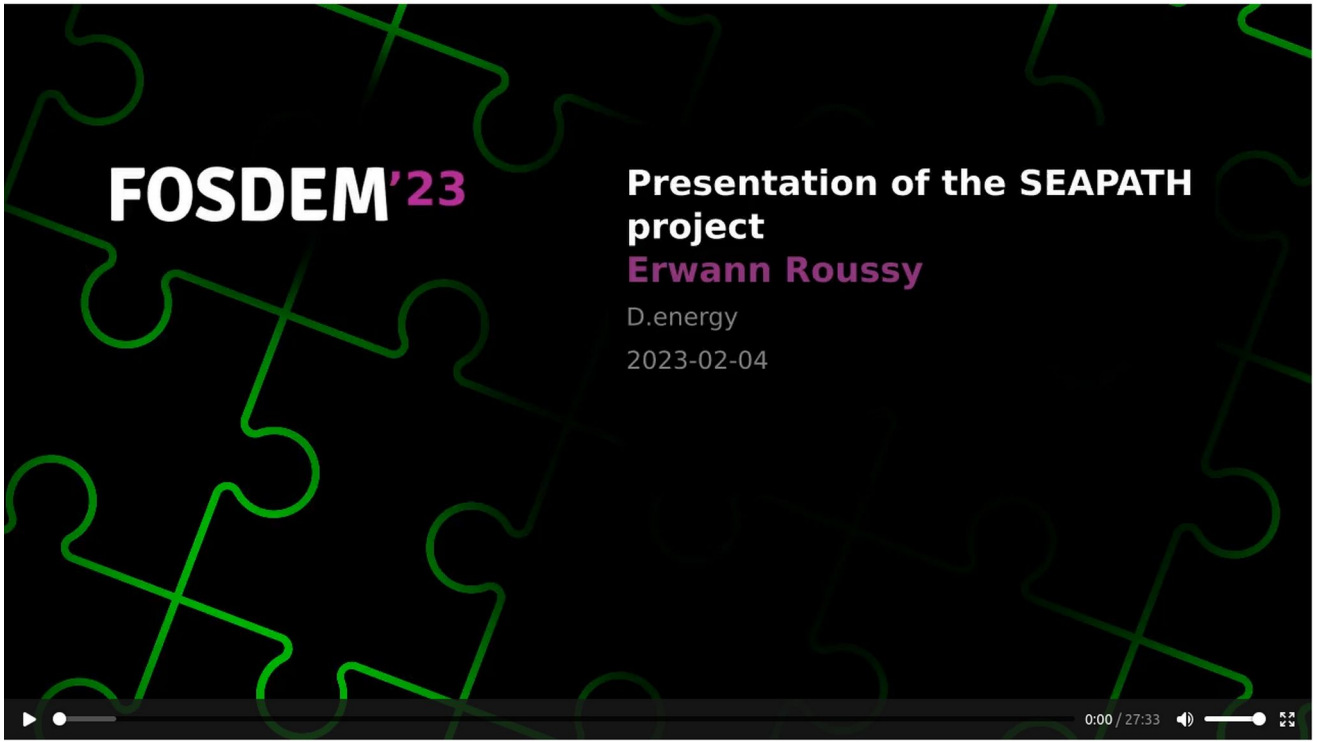
+

Combine performance, reliability and safety

Context


LFENERGY
SEAPATH
Hypervisor
VM
VM
VM
VM
VM
VM
...
libvirt
VIRTUALIZATION API
QEMU
Pacemaker
Corosync
STONITH
ceph
OvS
Open vSwitch
KVM
ANSIBLE
Linux Kernel (PREEMPT_RT)

For a deeper presentation of the Seapath project

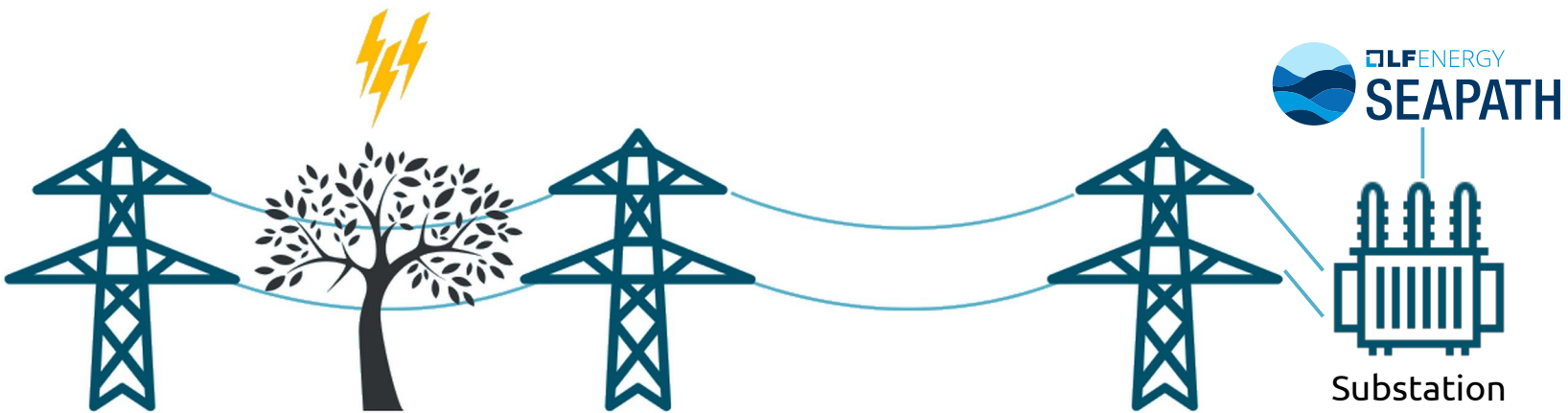


https://archive.fosdem.org/2023/schedule/event/energy_seapath/

Low latencies communication in a substation

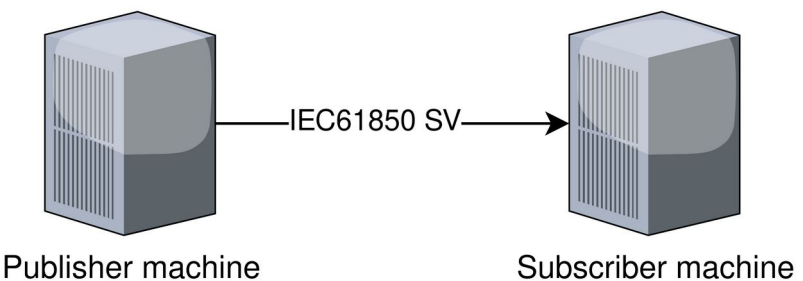
The need for low latencies

- Electrical substation safety and reliability is vital:
  - Protection of people
  - Substation provides energy to critical infrastructure: hospital, etc
- When an issue arise, virtualized protection application must react as soon as possible!

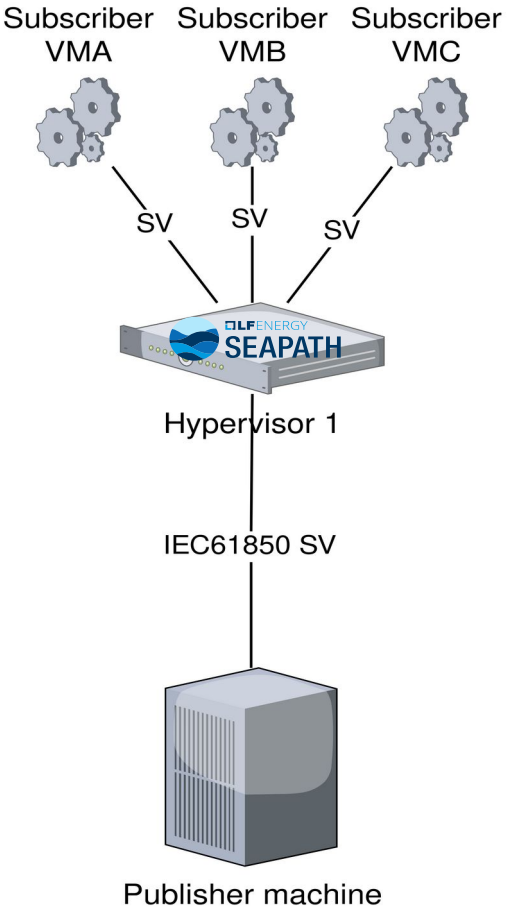


Communicating in a substation: introducing IEC61850 SV

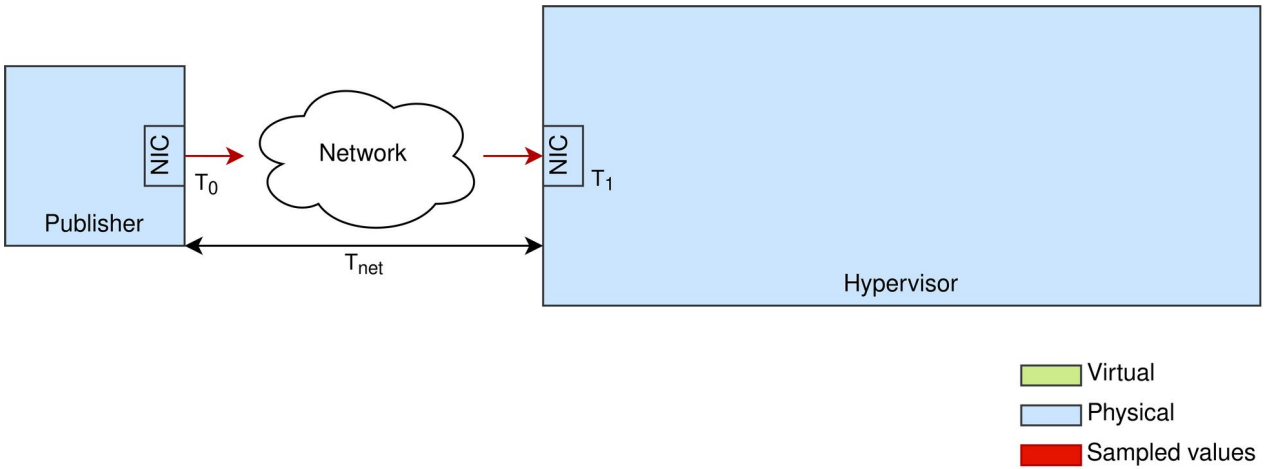
- The IEC61850 is an international standard defining communication protocols for intelligent electronic devices at electrical substations
- Multiple defined protocols: MMS, Goose, SV…
- **Today topic: IEC61850 SV "*Sampled Values*" protocol**
  - Ethernet L2 protocol
    - Protocol can be identified with 0x88BA Ethernet proto field
  - Publisher/Subscriber schema
    - Publisher periodically sends messages (every 250µs in a 50Hz electrical network) by stack of 1 to N (called streams)
    - Subscribers receive and decode them



Communicating in a substation: introducing IEC61850

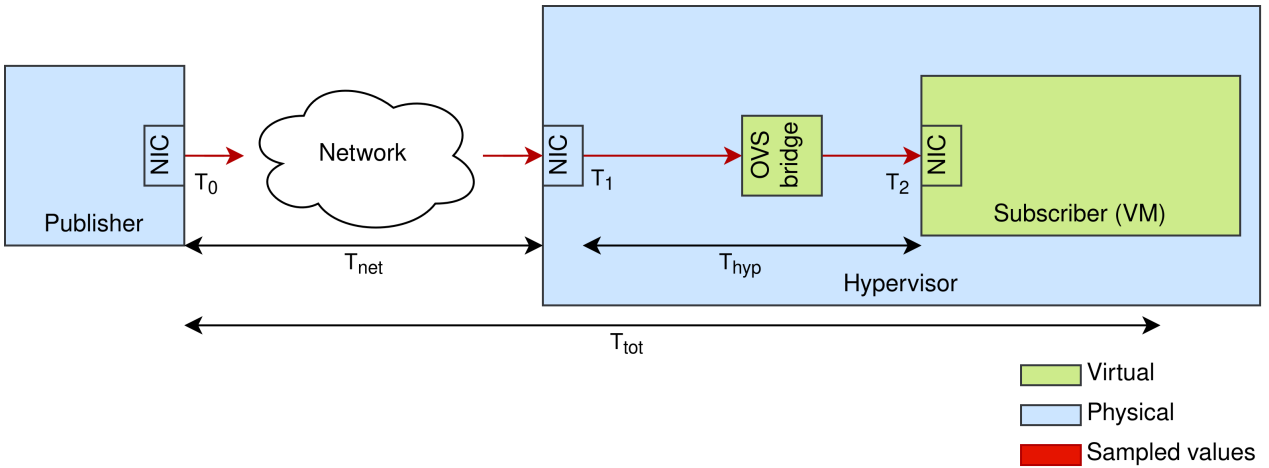


Lab setup – The path of an SV



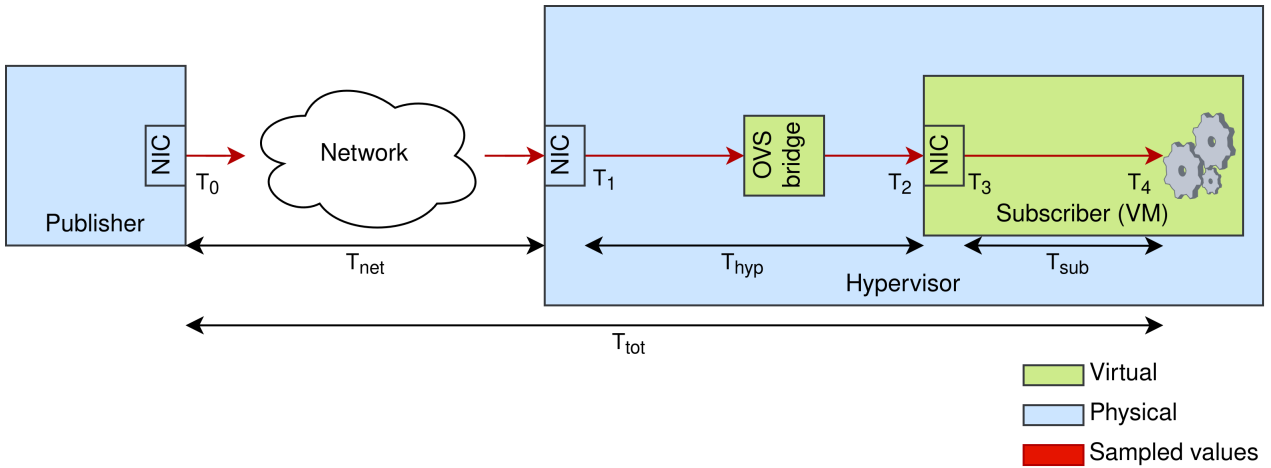
- SV are sent from publisher NIC (T_0) to the SEAPATH hypervisor NIC (T_1)
- T_{net} = elapsed time between publisher NIC to hypervisor NIC

Lab setup – The path of an SV



- SV are sent from hypervisor NIC (T_1) to the VM NIC (T_2) through OVS port
- T_{hyp} = elapsed time between hypervisor NIC to VM NIC

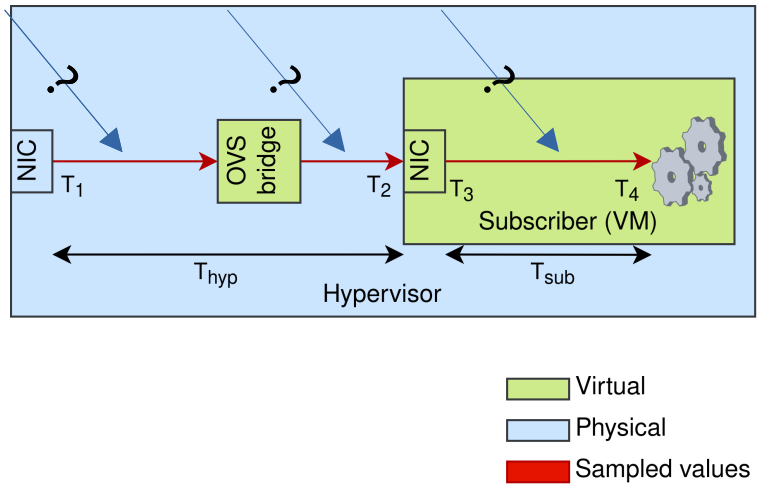
Lab setup – The path of an SV



- SV are sent from the VM NIC (T_3) to a user application (T_4)
- T_{sub} = elapsed time (subscriber time) between VM NIC to user application

Track latency sources

Identify latencies sources



- Where the latencies are coming from?

Identify latencies sources – Hypervisor side

- **First step: identify the kernel network stack when an SV is received**
- Solution: pwru tool (*Packets Where are U?*) with custom SV filter

```
root@adv:/home/virtu#  docker run --privileged --rm -t --pid=host -v /sys/kernel/debug/:/sys/kernel/debug/ cilium/pwru pwru 'ether proto 0x88BA'
2024/06/04 08:50:28 Attaching kprobes (via kprobe-multi)...
1404 / 1404
[----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------] 100.00% ? p/s
2024/06/04 08:50:28 Attached (ignored 0)
2024/06/04 08:50:28 Listening for events..
               SKB    CPU          PROCESS                    FUNC
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]              skb_push SV Arrival
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]          __skb_get_hash
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]              skb_clone
0xffffa04646232540      0 [irq/36-enp7s0-TxRx-0(1098)]         pskb_expand_head
0xffffa04646232540      0 [irq/36-enp7s0-TxRx-0(1098)]         skb_release_data
0xffffa04646232540      0 [irq/36-enp7s0-TxRx-0(1098)] skb_headers_offset_update
0xffffa04646232540      0 [irq/36-enp7s0-TxRx-0(1098)]               skb_push
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]            consume_skb
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]   skb_release_head_state
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]         skb_release_data
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]            skb_free_head
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]             kfree_skbmem
0xffffa04646232040      2 [ovs-vswitchd(854)]                skb_clone
0xffffa04646232540      2 [ovs-vswitchd(854)] kfree_skb_reason(SKB_DROP_REASON_NOT_SPECIFIED)
0xffffa04646232540      2 [ovs-vswitchd(854)]   skb_release_head_state
0xffffa04646232540      2 [ovs-vswitchd(854)]         skb_release_data
0xffffa04646232540      2 [ovs-vswitchd(854)]             kfree_skbmem
0xffffa04646232040      2 [ovs-vswitchd(854)]         __dev_queue_xmit
0xffffa04646232040      2 [ovs-vswitchd(854)]      netdev_core_pick_tx
0xffffa04646232040      2 [ovs-vswitchd(854)]        validate_xmit_skb
0xffffa04646232040      2 [ovs-vswitchd(854)]       netif_skb_features
0xffffa04646232040      2 [ovs-vswitchd(854)]  passthru_features_check
0xffffa04646232040      2 [ovs-vswitchd(854)]     skb_network_protocol
0xffffa04646232040      2 [ovs-vswitchd(854)]       validate_xmit_xfrm
0xffffa04646232040      2 [ovs-vswitchd(854)]      dev_hard_start_xmit
0xffffa04646232040      2 [ovs-vswitchd(854)]   Tovs
0xffffa04646232040      7 [qemu-system-x86(5639)]            consume_skb SV departure
```

SV NIC IRQ

OVS daemon

QEMU vhost driver

Identify latencies sources – Hypervisor side

- **First step: identify the kernel network stack when an SV is received**
- Solution: pwru tool (*Packets Where are U?*) with custom SV filter

```
root@adv:/home/virtu#  docker run --privileged --rm -t --pid=host -v /sys/kernel/debug/:/sys/kernel/debug/ cilium/pwru pwru 'ether proto 0x88BA'
2024/06/04 08:50:28 Attaching kprobes (via kprobe-multi)...
1404 / 1404
[----------------------------------------------------------------------------------------------------------------------------------------
100.00% ? p/s
2024/06/04 08:50:28 Attached (ignored 0)
2024/06/04 08:50:28 Listening for events..
               SKB    CPU          PROCESS                     FUNC
0xffffa04646233300      0 [irq/36-enp7s0-TxRx-0(1098)]                 skb_push SV Arrival

0xffffa04646232040      7 [qemu-system-x86(5639)]              consume_skb SV departure
```

SV NIC IRQ

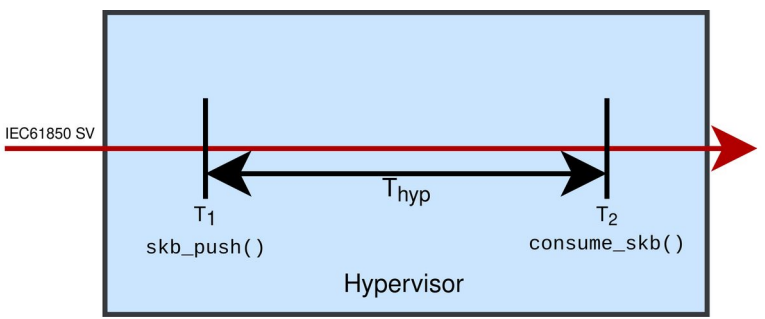
OVS daemon

QEMU vhost driver

Identify latencies sources – Hypervisor side

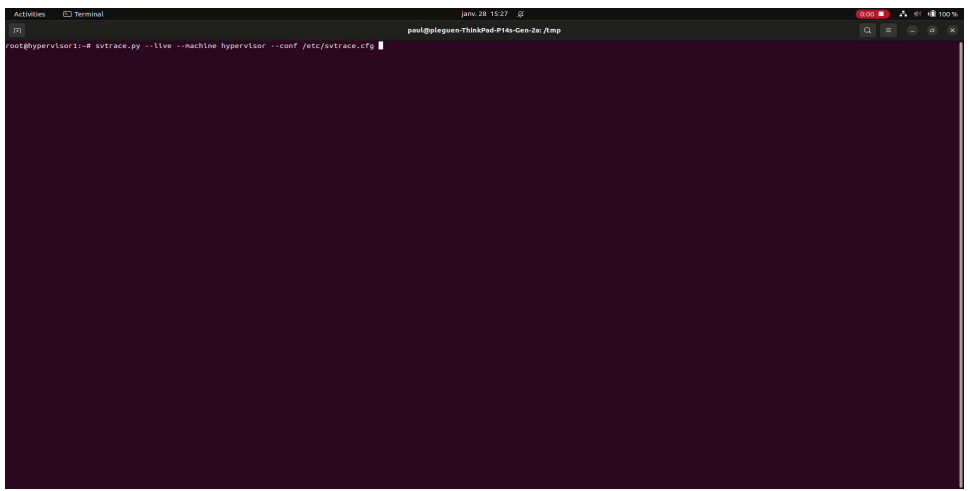
- **Second step: Automatize latencies computation**
- Solution: bpftrace
  - High level kernel tracer, based on eBPF
  - Hook on almost every kernel tracer
  - Dynamic tracing, custom script

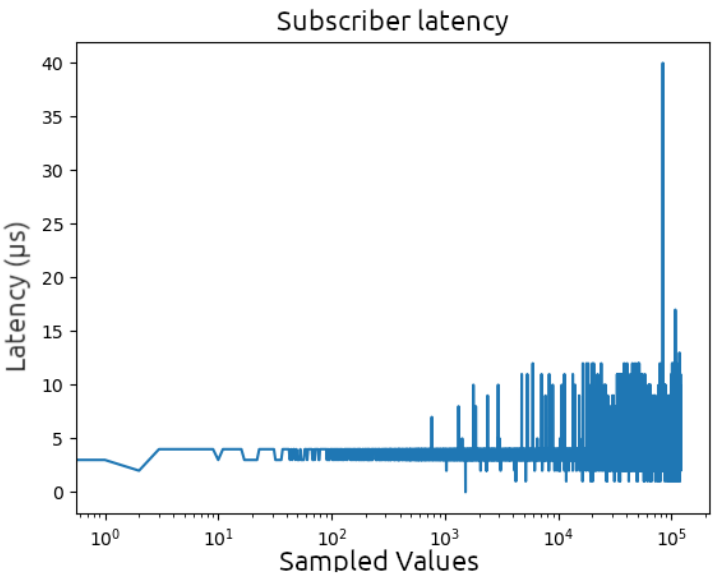
- Each skb_push(), call if packet is a SV, record T_1 timestamp
- Each consume_skb(), call if packet is a SV, record T_2 timestamp
- $T_2 - T_1 = T_{hyp}$



Identify latencies sources – Hypervisor side

- **Third step: Wrap automation in a Python wrapper, based on bpftrace scripts**
- Solution: svtrace
  - Two core features:
    - **live**: Show direct latencies received on machine, in terminal
    - **record**: Record latencies in a log file, which can be used later in post-process

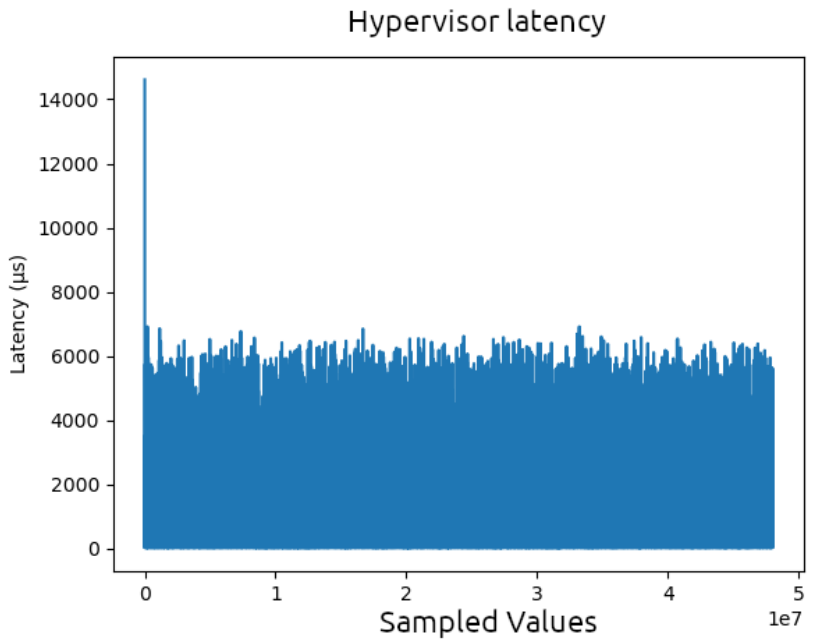




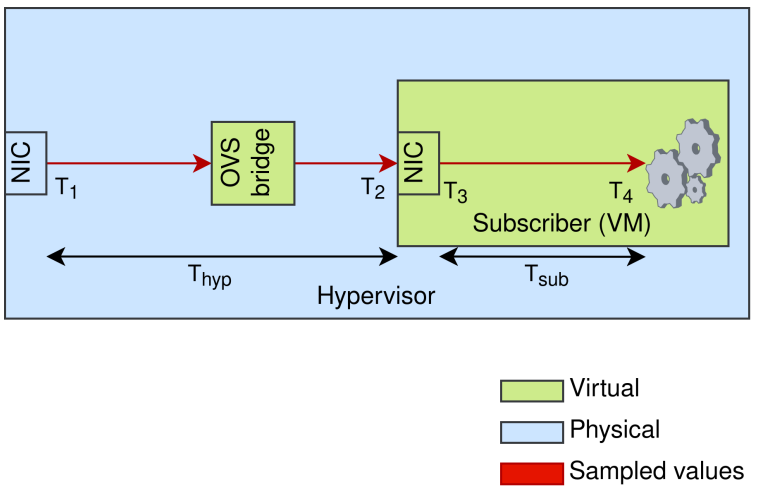
Practical case

Nominal case

- **Test condition:**
  - 48 000 000 SV sent (4 hours)
  - No optimization: core isolation, scheduler priority…

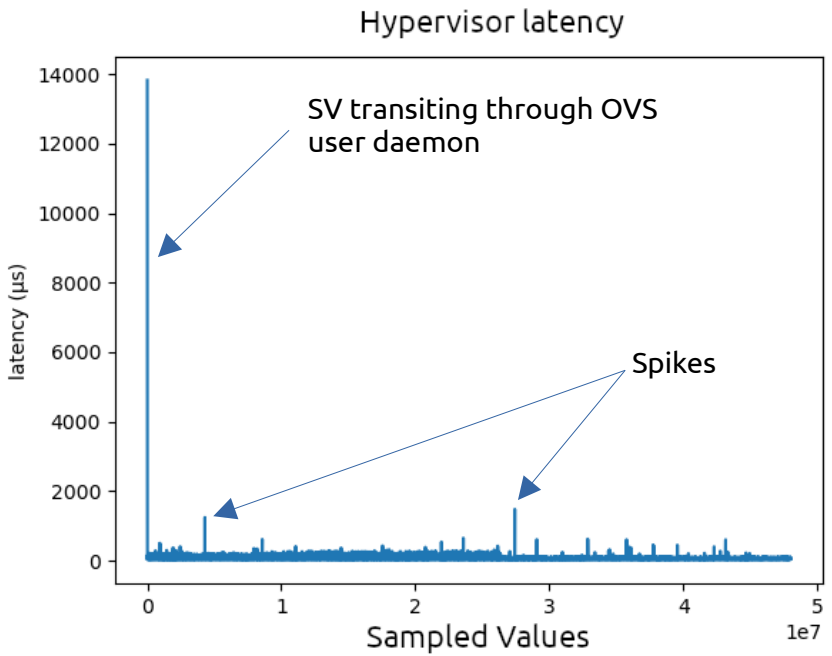


Mean = 58µs | Max = 14 612µs | Min = 26µs
Latency > 200µs : 352 533

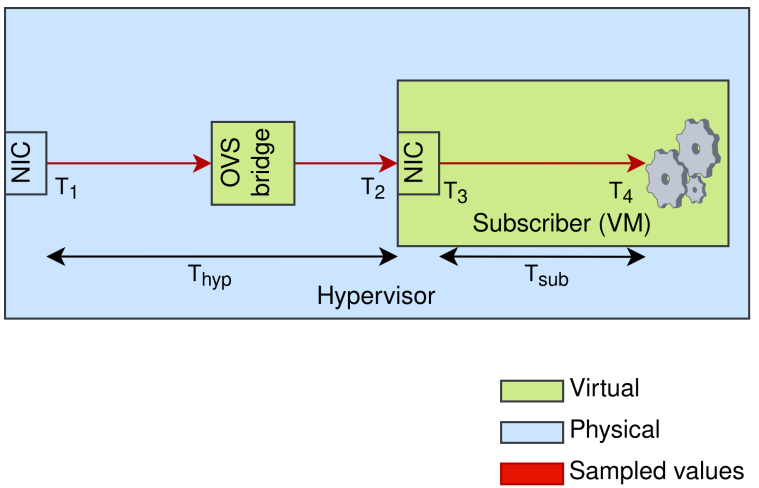


Optimized case

- **Test condition:**
  - 48 000 000 SV sent (4 hours)
  - Subscriber KVM cores isolated, QEMU process (emulator pin) isolated



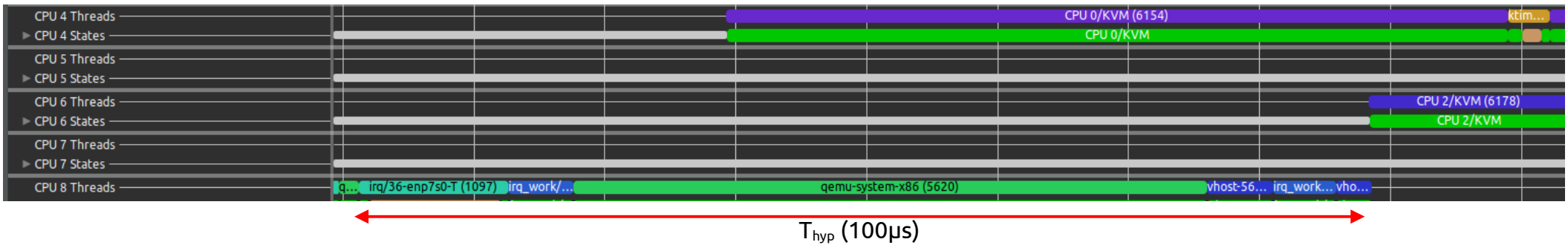
Mean = 45µs | Max = 13 826µs | Min = 29µs
Latency > 200µs : 512 (- 99,8 %)



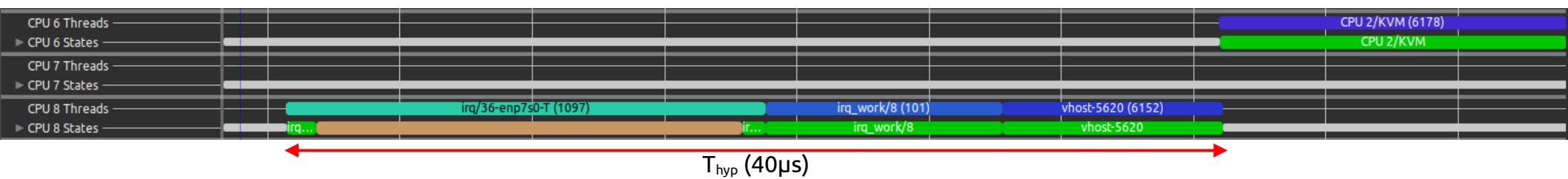
Fixing spikes

- Latencies are recorded between T_1 and T_2
- For deeper investigation → lttNG is required

- **Incorrect SV transit:**



- **Correct SV transit:**



Thank you for your attention!

Paul Le Guen de Kerneizon

paul.leguendekerneizon@savoirfairelinux.com

https://savoirfairelinux.com/en/services/seapath-engineering-services

seapath@savoirfairelinux.com

https://lfenergy.org/projects/seapath/

https://github.com/seapath

https://lf-energy.atlassian.net/wiki/spaces/SEAP/overview?homepageId=31817739

https://lfenergy.slack.com#seapath