RED HAT SUMMIT

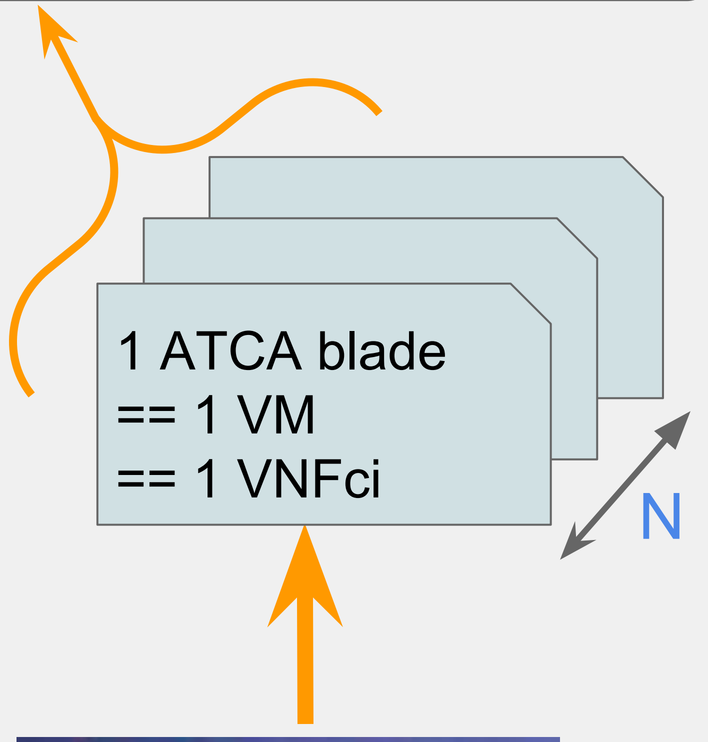
NFV go-live

Where are my containers?

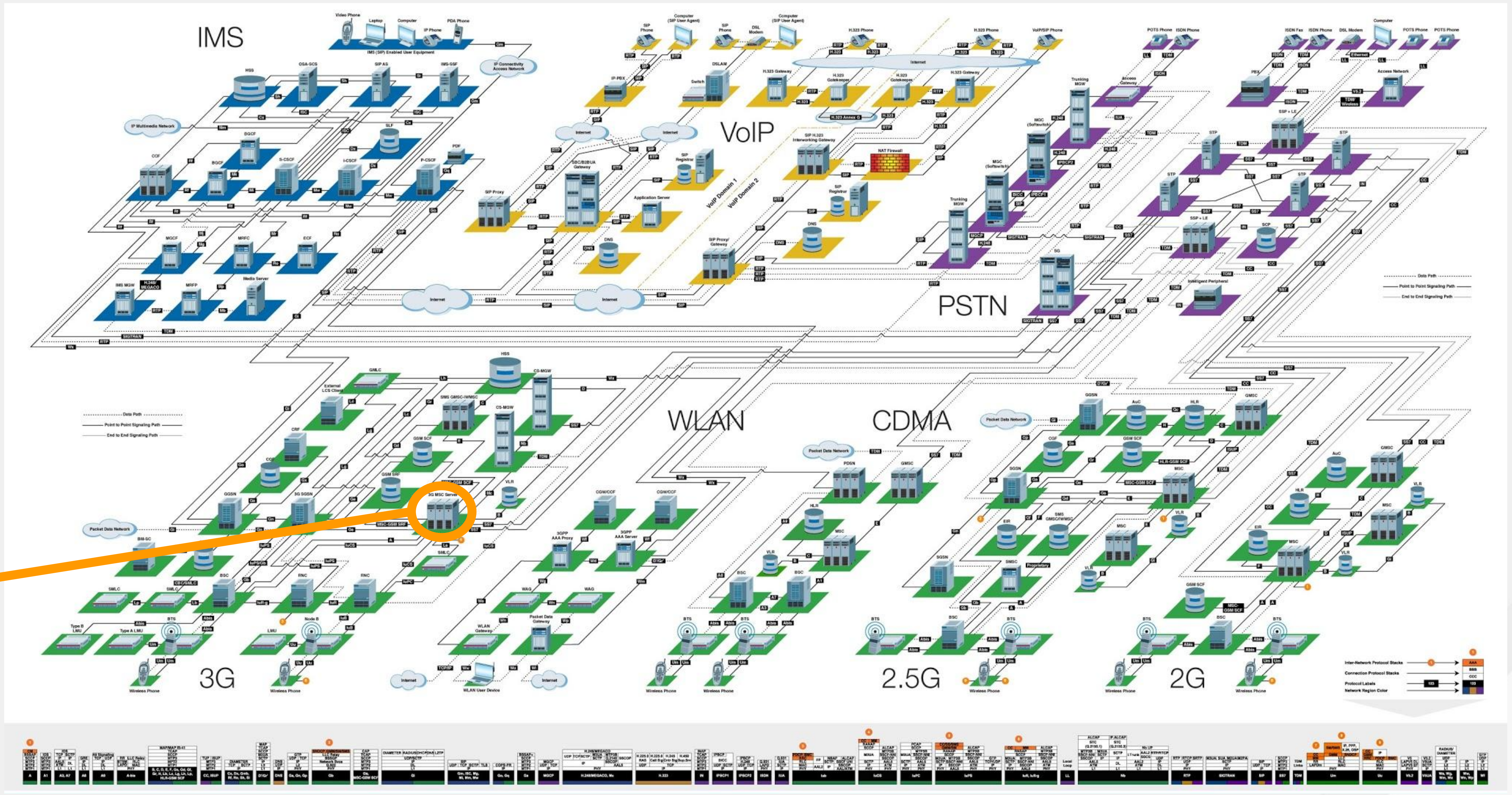
Franck Baudin
Sr Principal Product Manager - OpenStack NFV
May 9 , 2018

Mobile networks deployment today/yesterday

1 VNF == N x VNFci



Picture credits: [wikipedia]



Cloud/Web-App world vs Networking/NFV world

Cultural shock... or at least persistent misunderstanding

For Cloud/Web-App peoples, a load balancer is part of the PaaS (detail of implementation)

- It just works
- This is called an L7 LB but in reality this is a REST/HTTP or TCP based (with TLS or not)

In Networking/NFV world, the load balancer **is** the application or part of the application

- A GTP or SIP session load-balancer
- An L7 LB is decoding the sessions content to load-balance (DPI)
  - Based on URL
  - Based on content: gzip/MIME to be sent in an IDS cluster

More vocabulary clash: performances, scalability, availability, network connectivity, Real Time, ...

NFV is live, and go-live rhythm accelerate!

Common questions/quotes around NFV and containers

- Government security agency for Telco: “Containers in NFV is an absolute no-go”
- VNF editor: “How to re-write my VM based VNF with Containers?”
- Telco Operator: “Are there still some VMs around?”

This presentation will showcase

- That Kubernetization of VNFs is already possible even if Kube NFV Networking is WIP upstream
- What is the most common design pattern to implement cloud native VNFs with Kubernetes

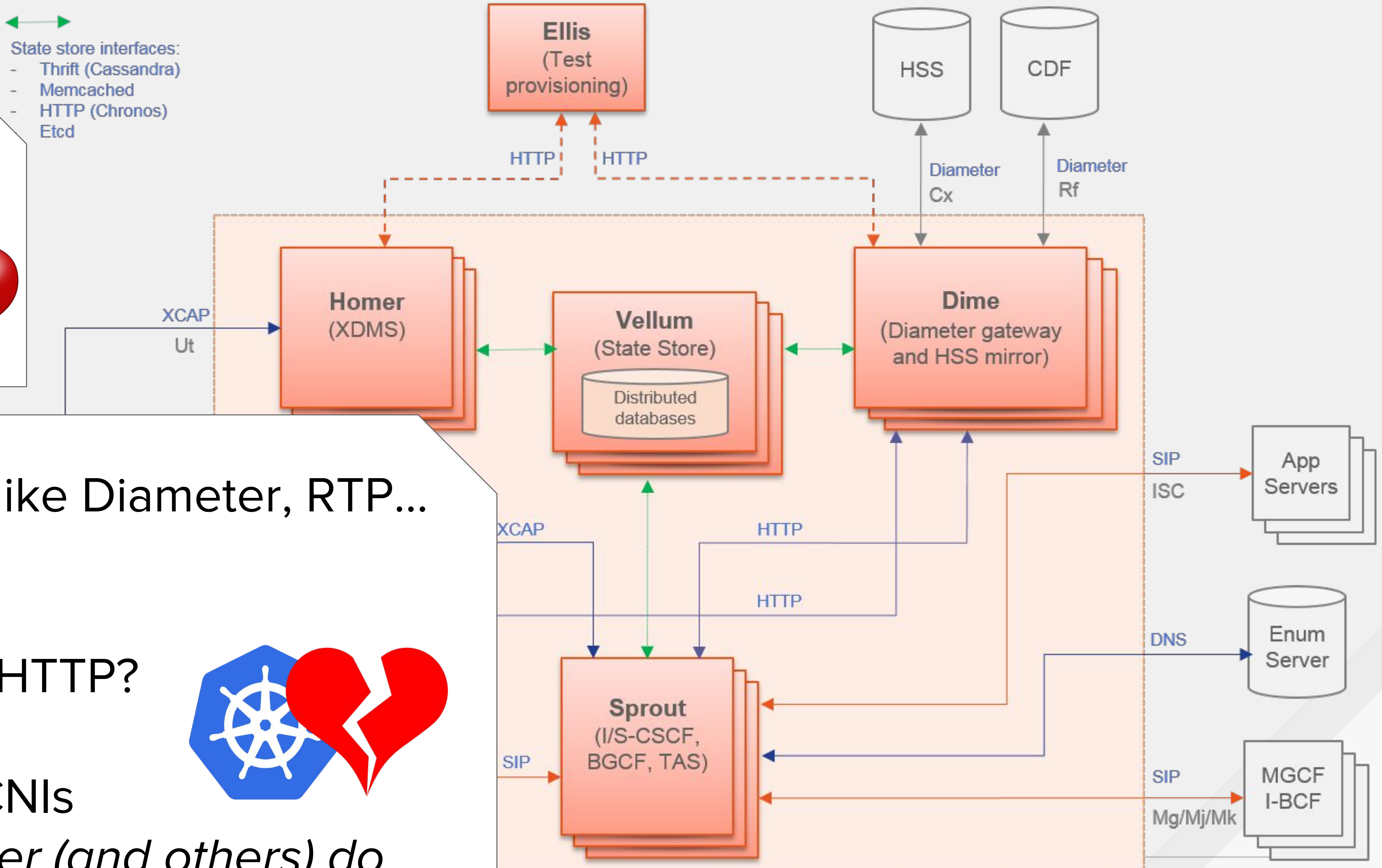
This presentation will not detail why NFV containers run in VMs for security reasons (yes, they do!)

- “What happens in a VM stays in the VM”
  - one origin/vendor per VM
- no impact on containers performances (cycles/packet), containers lifecycle, containers start time, ...

A containerized vIMS

PROJECT Clearwater

Containerized vIMS != Kubernetized vIMS



A large portion of coms use REST/HTTP

Telco/Network protocols like Diameter, RTP... are **not** REST/HTTP

=> Tunnel protocols over HTTP?
- *this is a wrong idea*

=> host networking, DIY CNIs
- *this is what Clearwater (and others) do*

Schema credits: http://www.projectclearwater.org/

Kubernetes and NFV networking

Scratch status

Upstream Kubernetes networking is not adapted to NFV (yet)

- REST/HTTP/TCP versus Networking/Telco protocols IEEE/IETF/3GPP/...

But NFV do **not** require Networking/Telco protocols support in Kubernetes

- Multiple ethernet interfaces up to the container/pod will suffice
  - Can be virtio interfaces, based on any DPDK or kernel vswitch
  - Can be physical dedicated interfaces or dedicated virtual functions
- WIP upstream to support Telco protocol with multiple CNI plugins: multus, vhostuser, device plugin...

This explains how **some** VNFs are already using containers orchestrated by k8s or not, inside VMs (security/compartmentalization), with WIP/proprietary CNI, as PoCs

- **They all aim to converge on a full upstream/k8s implementation (no DIY/Fork for production)**

Cloud Native NFV design pattern

NFV/Networking is all about **state**

VNF components track user sessions, TCP sessions ... they track **state**

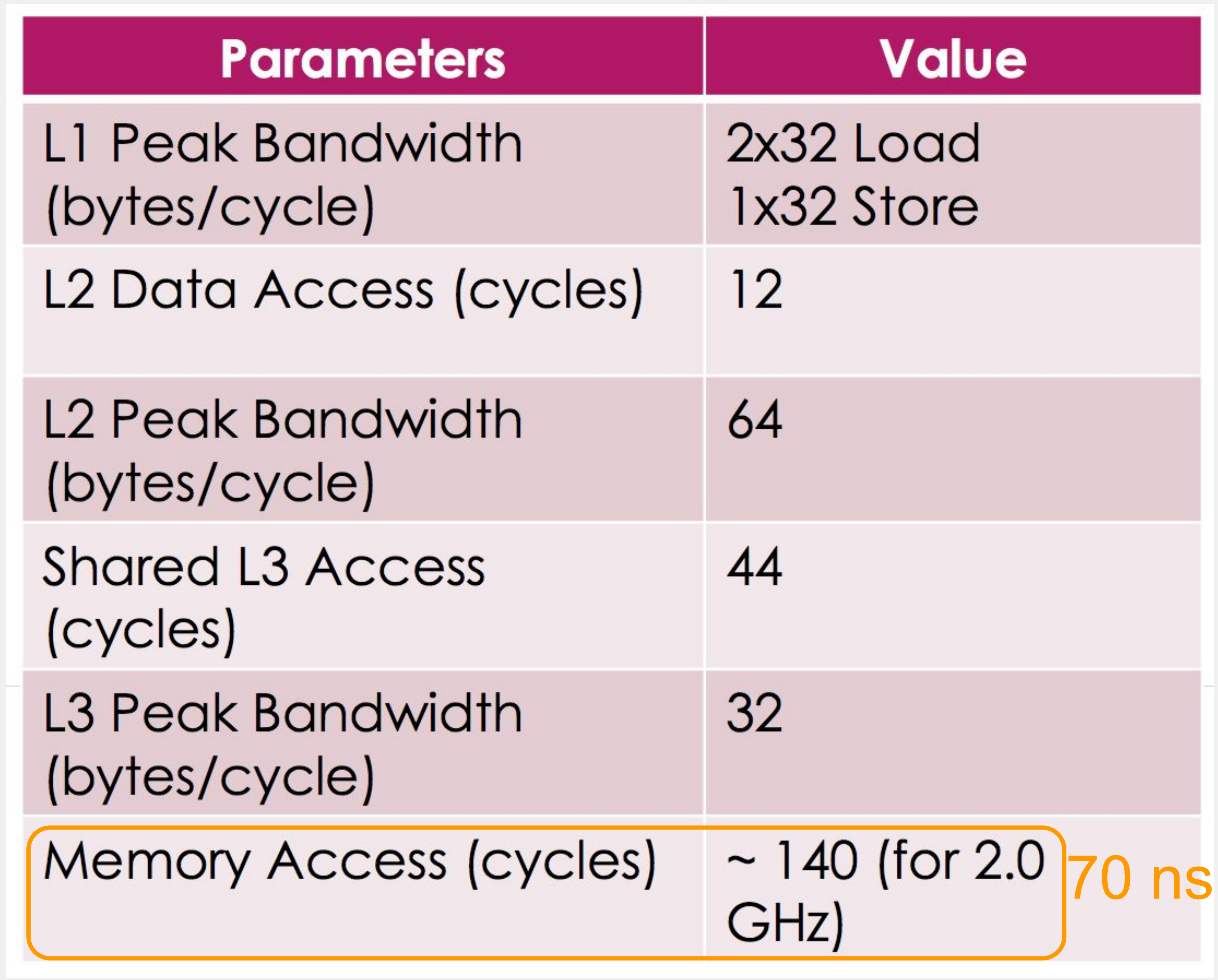
- Firewall: TCP ACK, checking TCP windows values
- Billing, QoS: statistics per session/user
- Legal Intercept, Intrusion Detection/Protection: content reconstruction (email, files, pictures), cross sessions/flows correlation (bot control channel detection)
- Video optimizer: video stream sequence
- ...

Depending on the VNF type, losing state means

- security breach
- Improper billing
- Low Quality of Experience
- ...

Remote state storage for NFV?

From https://dpdksummit.com/Archive/pdf/2016USA/Day02-Session03-PeilongLi-DPDKUSASummit2016.pdf



Parameters	Value
L1 Peak Bandwidth (bytes/cycle)	2x32 Load 1x32 Store
L2 Data Access (cycles)	12
L2 Peak Bandwidth (bytes/cycle)	64
Shared L3 Access (cycles)	44
L3 Peak Bandwidth (bytes/cycle)	32
Memory Access (cycles)	~ 140 (for 2.0 GHz)

70 ns

VNF performances are about **cycles/packet**

- Throughput
- Latency

Typical ranges

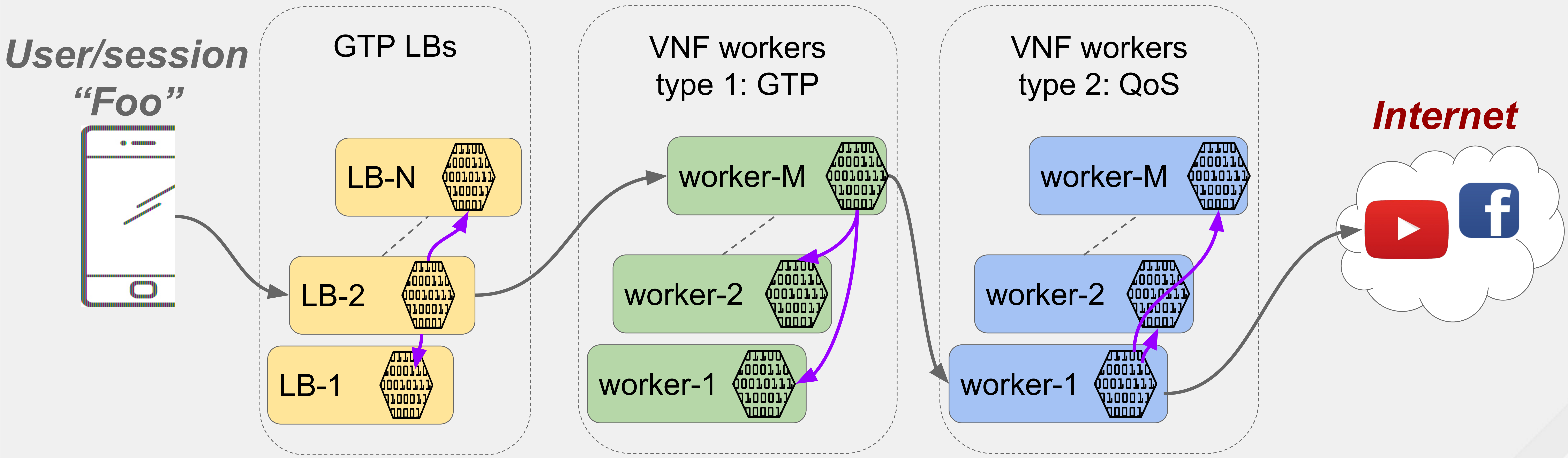
- Dataplane: 300 and 2000 CPU cycles
- Control-plane: between 1000 and 20k CPU cycles

A **single** remote lookup, **at best**

- N x 10 usec == N x 20k's cycles (for 2.0 GHz)

https://dpdksummit.com/Archive/pdf/2015Userspace/DPDK-Userspace2015-SevenDeadlySinsPacketProcessing.pdf

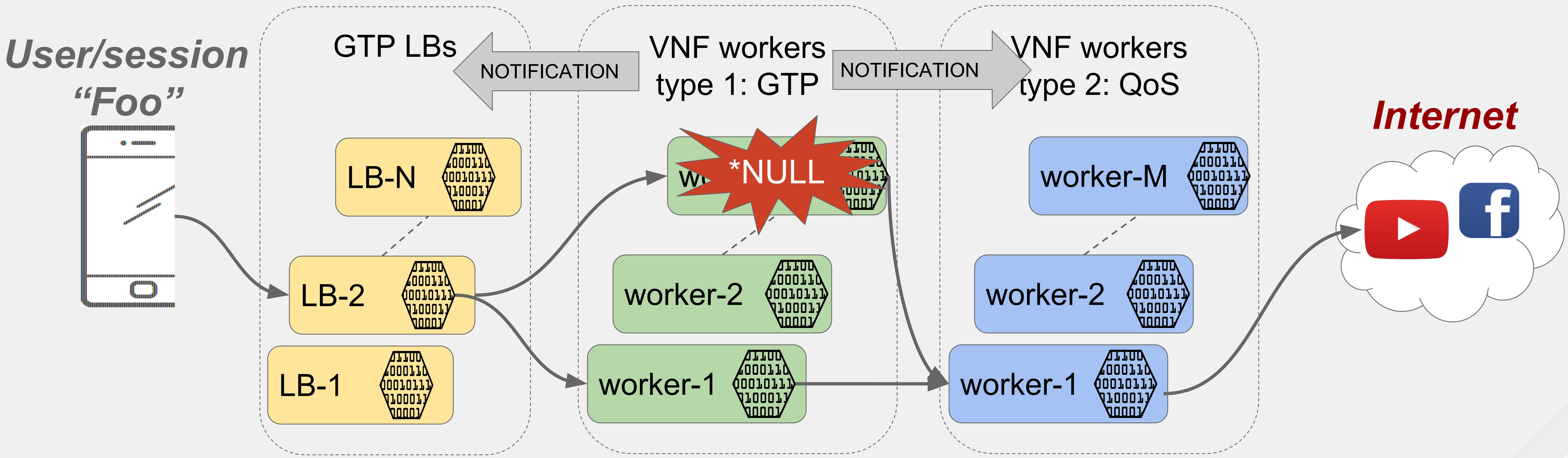
VNF typical structure: pipeline of scalable and resilient stages



GTP sessions LB info replicated (HA: 1+1 or n+p)

VNF states replicated (HA: 1+1 or n+p)

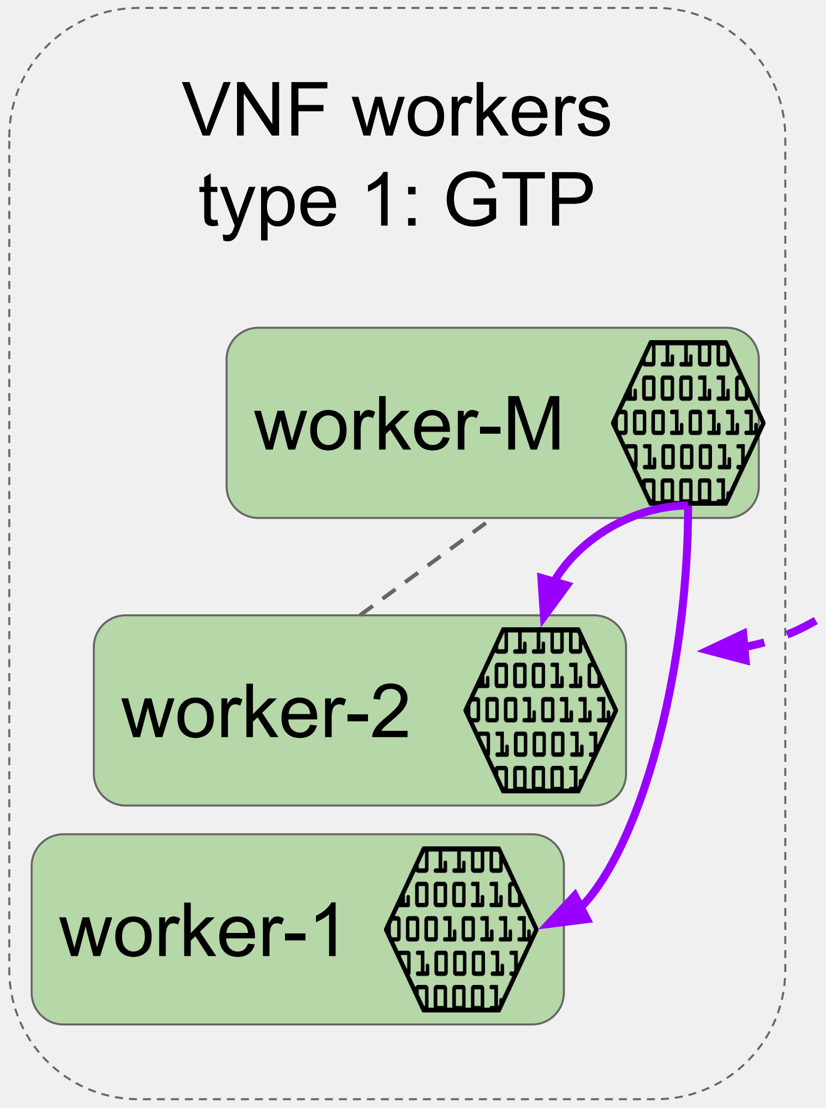
Failure of one element



On an established session, synced across replicas: few user datas lost or retransmitted, minimum impact

On a session creation: session to restart (user lag) or accept inaccurate session tracking (risk)

Sacrificial state, window of vulnerability

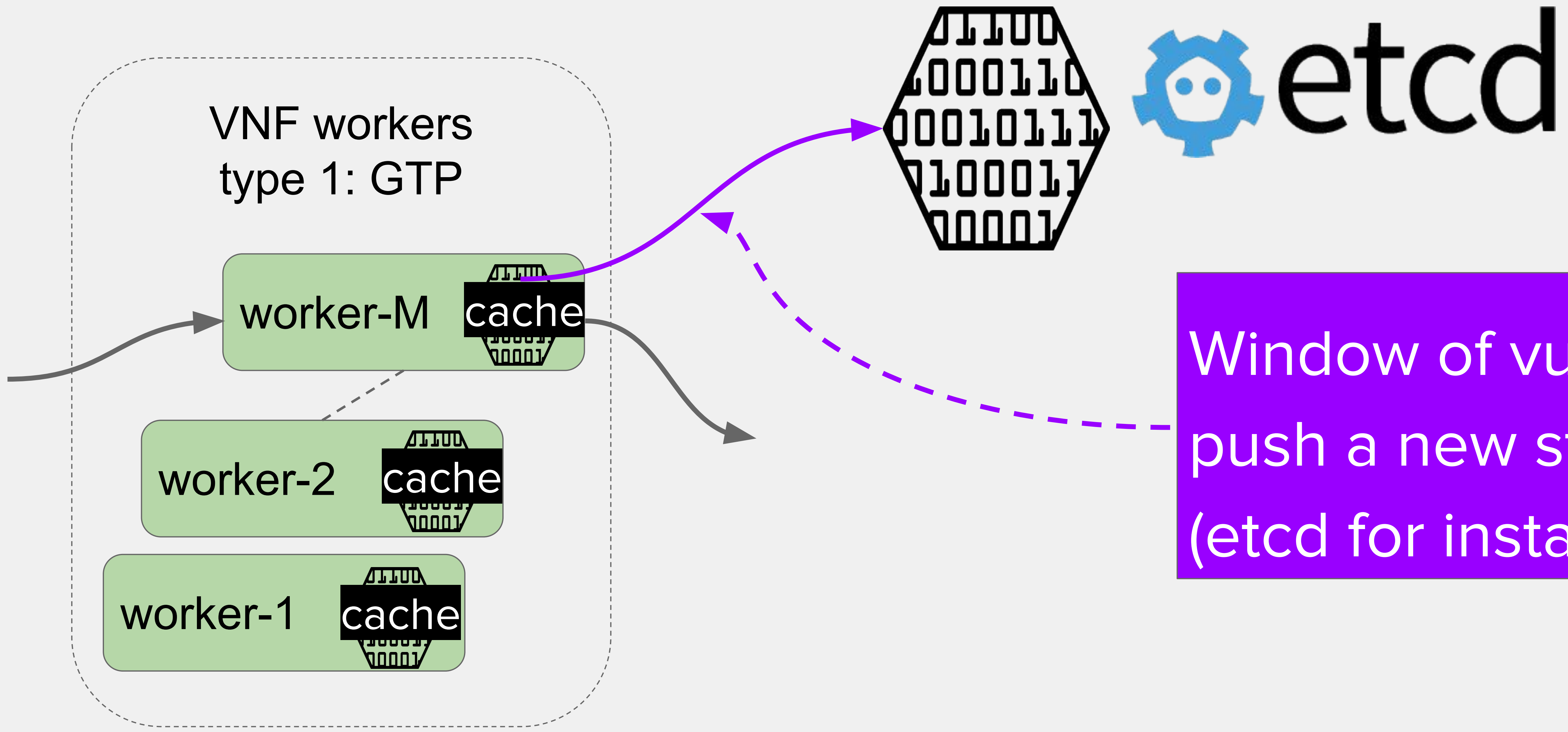


This is the window of time that elapse until a state is safely stored in multiple places (at least two)

State loss is only one part of the service unavailability picture

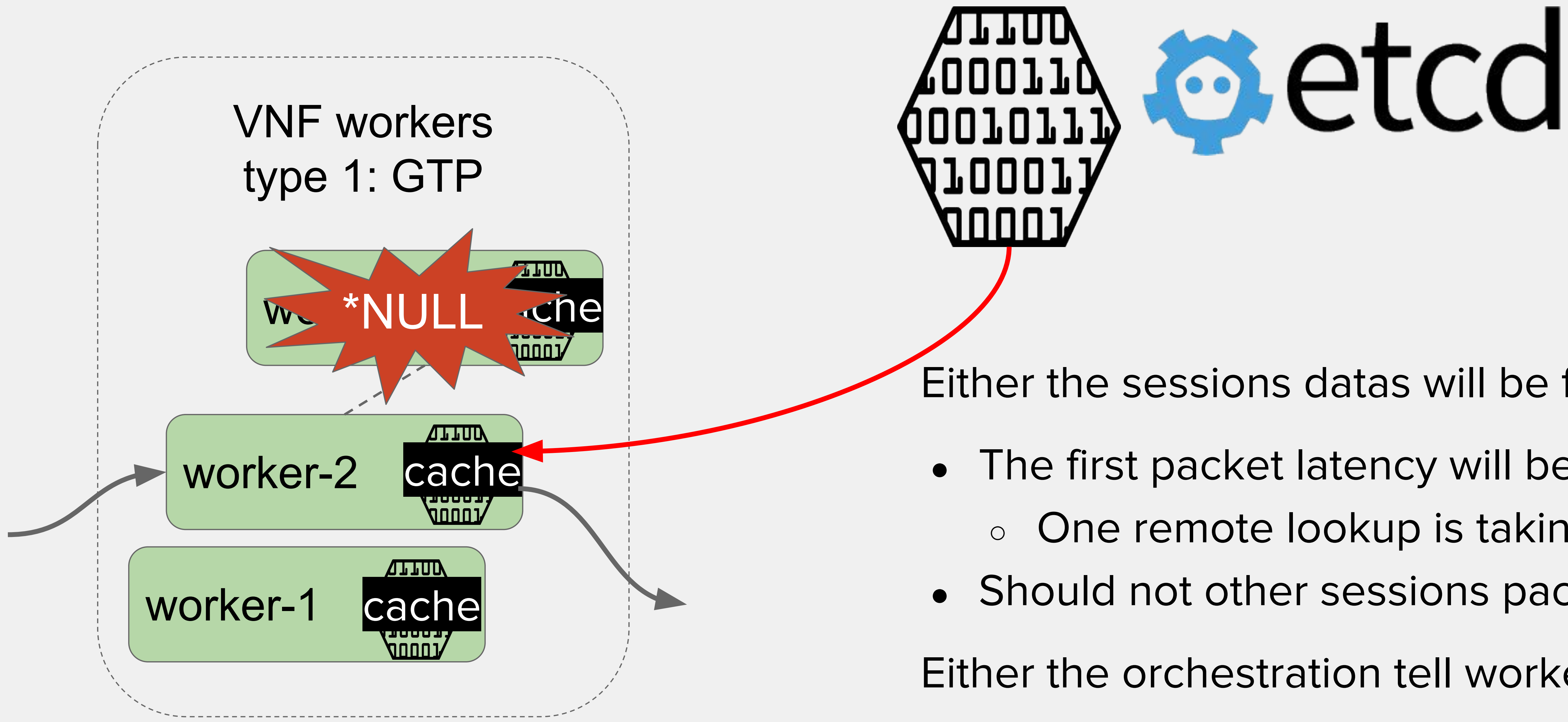
- Fault detection time (failure of one VNF element, aka VNFCI)
- Remediation time (reconfiguration with remaining VNFCI"s)
- Availability level recovery: replacement of the failed VNFCI
  - What if a second VNFCI fails?

The k8s and cloud-native way



Window of vulnerability == time to push a new state into the central (etcd for instance) state storage

Failure of one container



Either the sessions datas will be fetched on demand

- The first packet latency will be way higher than other packets
  - One remote lookup is taking 20k CPU cycles at best
- Should not other sessions packets (no blocking)

Either the orchestration tell worker-2 to prefetch sessions datas

- Fall-back on next implementation when packets arrive before the prefetch completion

Conclusion: Kubernetization of VNFs is WIP!

Containerisation is not what you want: you want *Kubernetization*… now!

- All VNFs components are not at the same containerization/Kubernetization stage
  - Re-architect as microservices, one VNFc (one pipeline stage) at a time
    - Re-implement the most critical VNFc first
    - Keep VMs for others VNFc or re-package them as big-fat containers
- Interconnecting VMs and Containers has no overhead when using OpenStack Kuryr (new!)
  - OpenShift pods directly connected to any Neutron network, without any overhead
- Networking backend proprietary/non standard CNI
  - Plenty already validated with OpenShift, streamlined and fast process
  - Red Hat leading upstream communities to provide an NFV grade CNI
- Containers deployed in VMs for security reasons: Security Government agencies requirement
    - Virtualization overhead is negligible

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

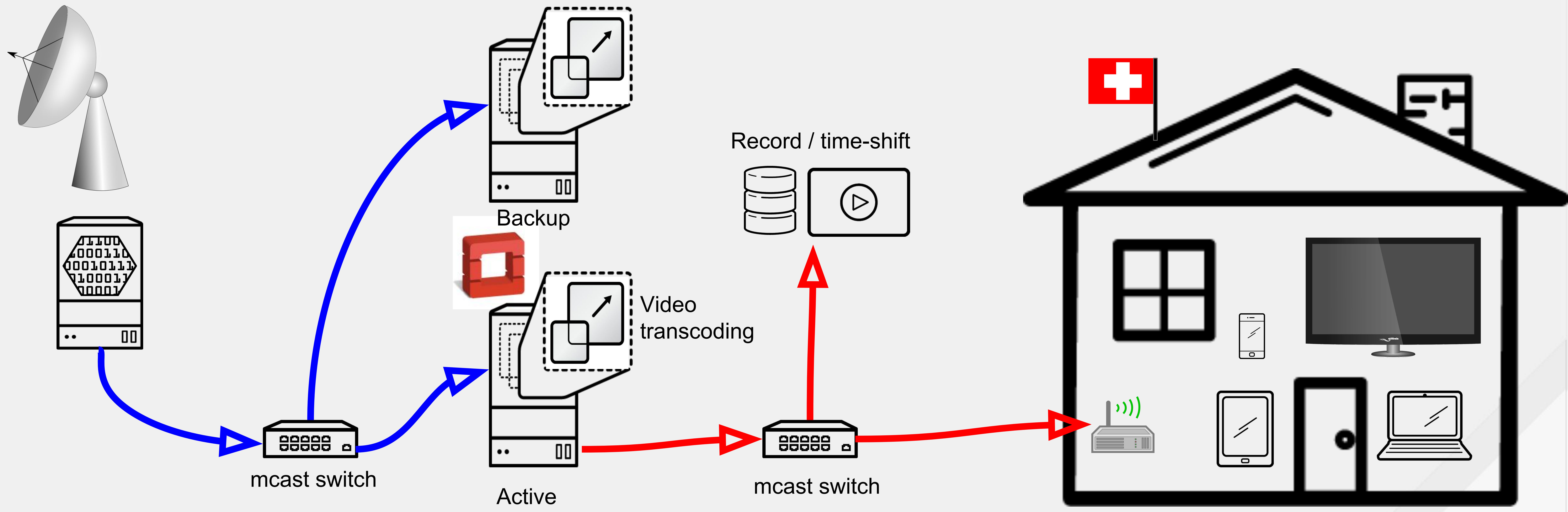
youtube.com/user/RedHatVideos

Backup-Slides

IP-TV live on OpenStack: Another High Availability model

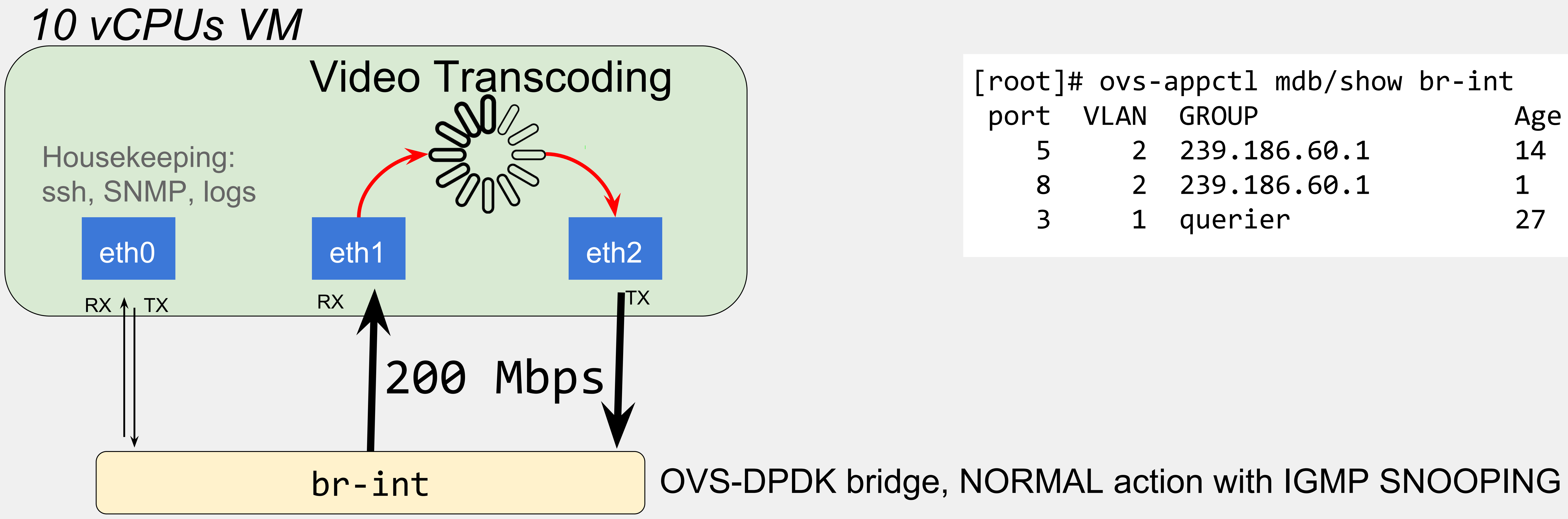
IP TV architecture at swisscom

Simplified, per TV channel



Zoom on one (active) VM

One VM == one TV channel == one input mcast stream + one output mcast stream



```
[root]# ovs-appctl mdb/show br-int
 port  VLAN  GROUP                Age
    5     2  239.186.60.1         14
    8     2  239.186.60.1         1
    3     1  querier              27
```

```
ovs-vsctl set Bridge br-int mcast_snooping_enable=true
ovs-vsctl set Bridge br-int other_config:mcast-snooping-disable-flood-unregistered=true
```