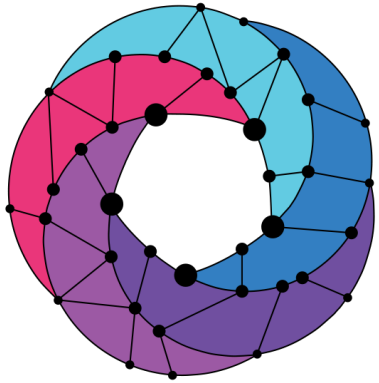


Open Source
MANO

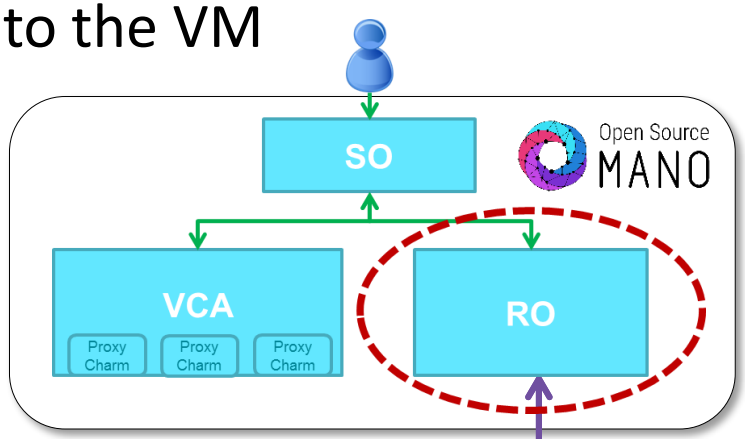
OSM Hackfest – Session 5
Modeling EPA capabilities in VNF

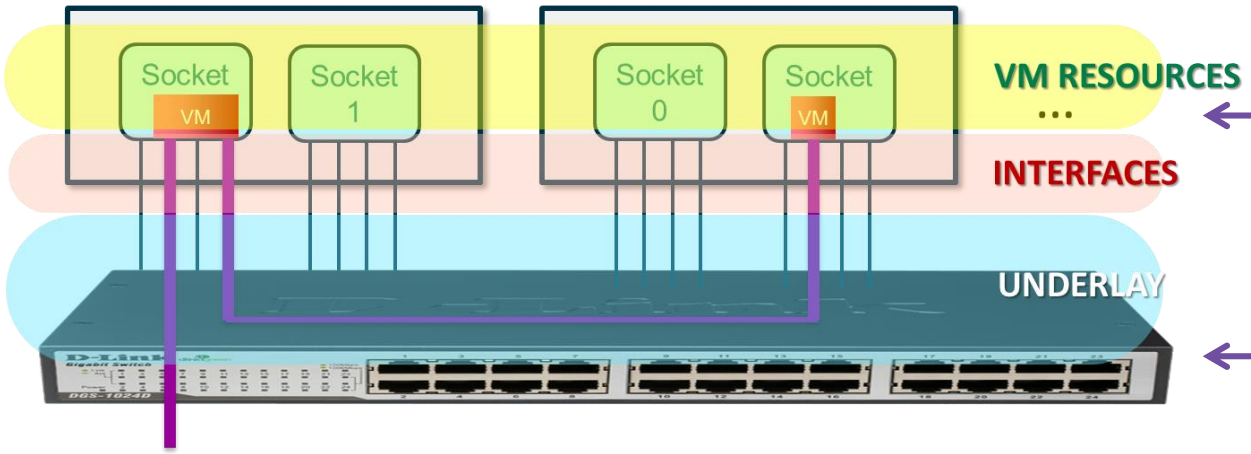
Gerardo García (Telefónica)

ETSI

EPA support combined with SDN Assist enables chaining of high performance VNFs

1. Accurate assignment of resources at VM level
2. Proper assignment of I/O interfaces to the VM
3. **SDN gives the ability to create underlay L2 connections**
   - Interconnecting VMs
   - Attaching external traffic sources





EPA

- **EPA features** like use of large hugepages memory, dedicated CPUs, strict NUMA node placement, and the use of passthrough and SR-IOV interfaces, **can be used in OSM's VNF descriptors since Rel Zero**.

- If your VIM supports EPA, then you don't need to do anything extra to use it from OSM. VIM connectors in OSM take advantage of EPA capabilities if the VIM supports it. All you need to do is build your descriptors and deploy.

- Openstack configuration for EPA (reference guide):
  - https://osm.etsi.org/wikipub/index.php/Openstack_configuration_(Release_THREE)#Configure_Openstack_for_OSM_.28EPA.29

SDN assist

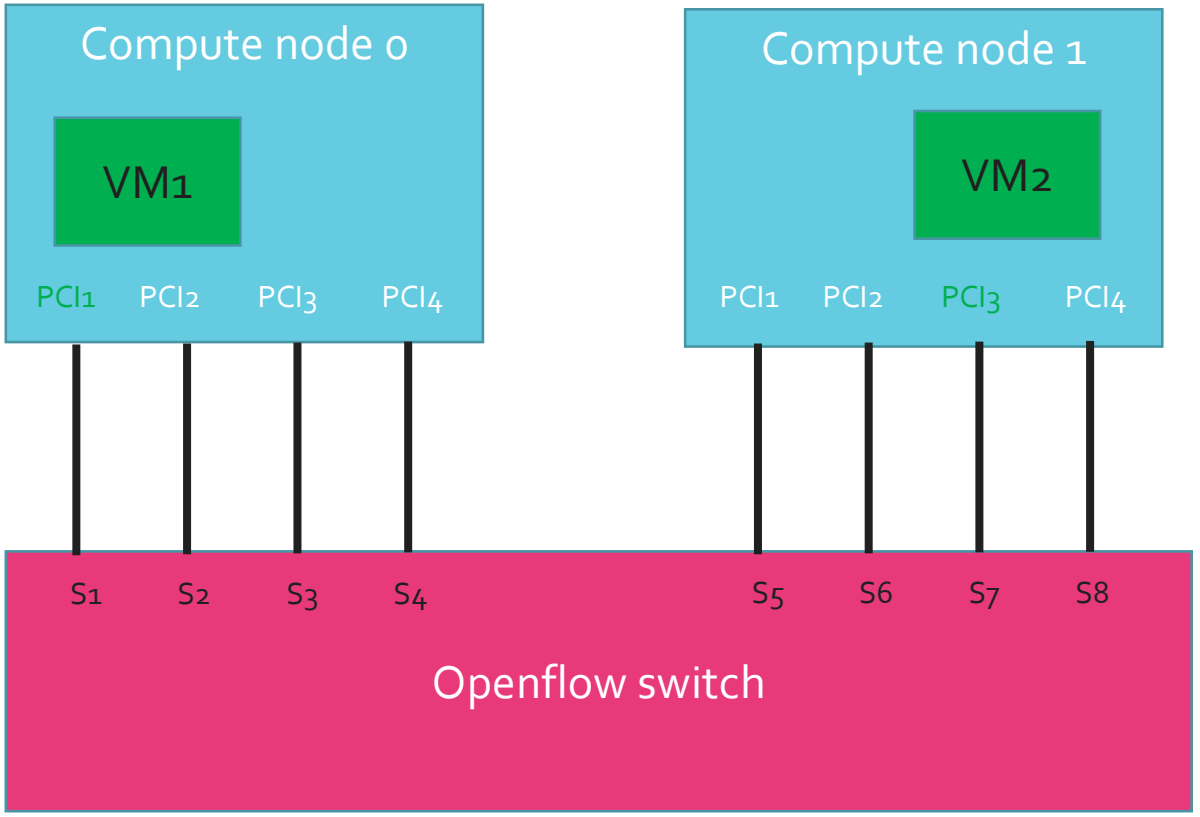
- This feature allows to use an external controller to create the underlying connectivity
- Wiki page:
  - https://osm.etsi.org/wikipub/index.php/EPA_and_SDN_assist
- Requirements:
  - A dataplane switch with Openflow capabilities that will connect the physical interfaces of the VIM compute nodes.
  - An external SDN controller controlling the previous dataplane switch.
  - The mapping between the switch ports (identified by name) and the compute node interfaces (identified by host-id and PCI address)
  - Some VIMs as Openstack requires admin credentials in order to be able to get the physical place of the SRIOV/passthrough VM interfaces

SDN assist. The way it works

- OSM will deploy the VMs of a NS with Passthrough and/or SRIOV interfaces
- OSM will get from the VIM (in your case, Openstack) the compute node where the VM was deployed and the physical interface assigned to the VM (identified by its PCI address).
- OSM will map those interfaces to Openflow ports in the switch making use of the mapping that you should have introduced in the system
- OSM will create the dataplane networks by talking to the SDN controller and connecting the appropriate Openflow ports to the same network.

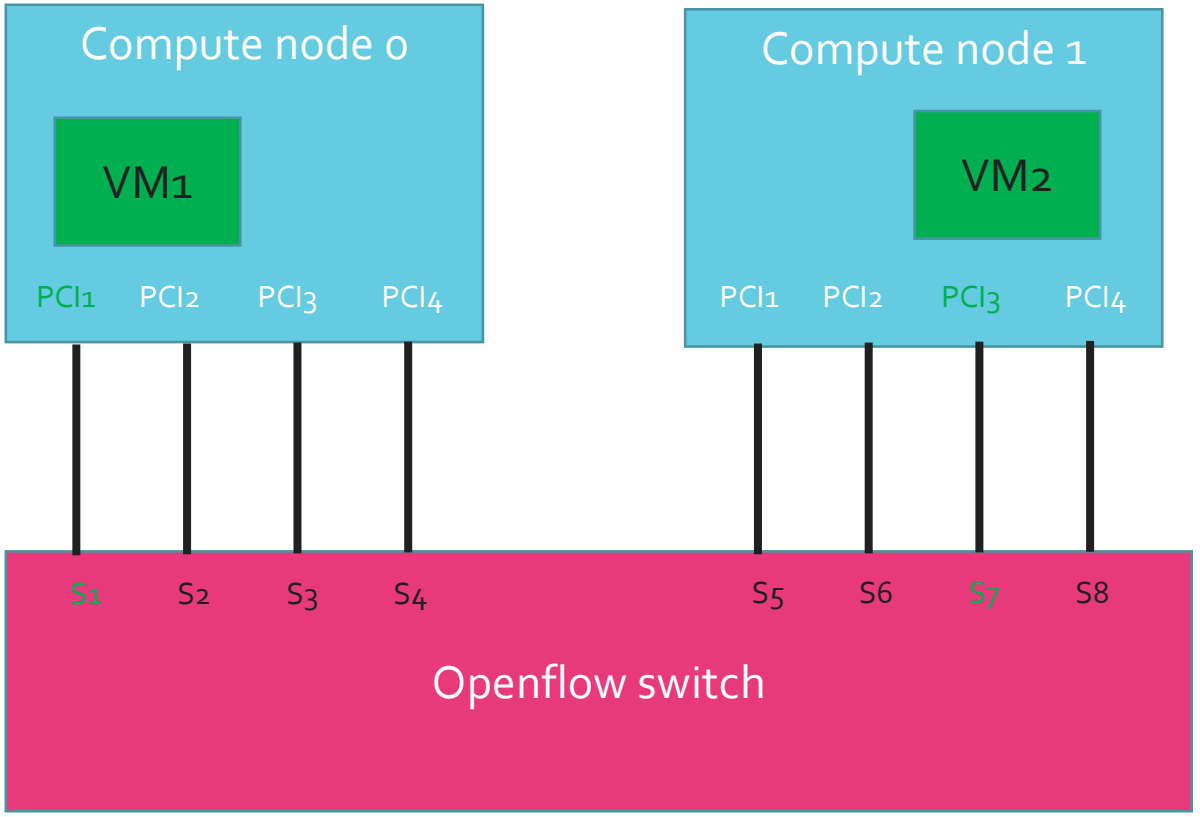
SDN assist. The way it works
1) VMs are deployed

VM1 will be e.g. deployed in compute node 0 and will consume interface idenrified by PCI address PCI1 and MAC MACX



VM2 will be e.g. deployed in compute node 1 and will consume interface idenrified by PCI address PCI3 and MAC MACY

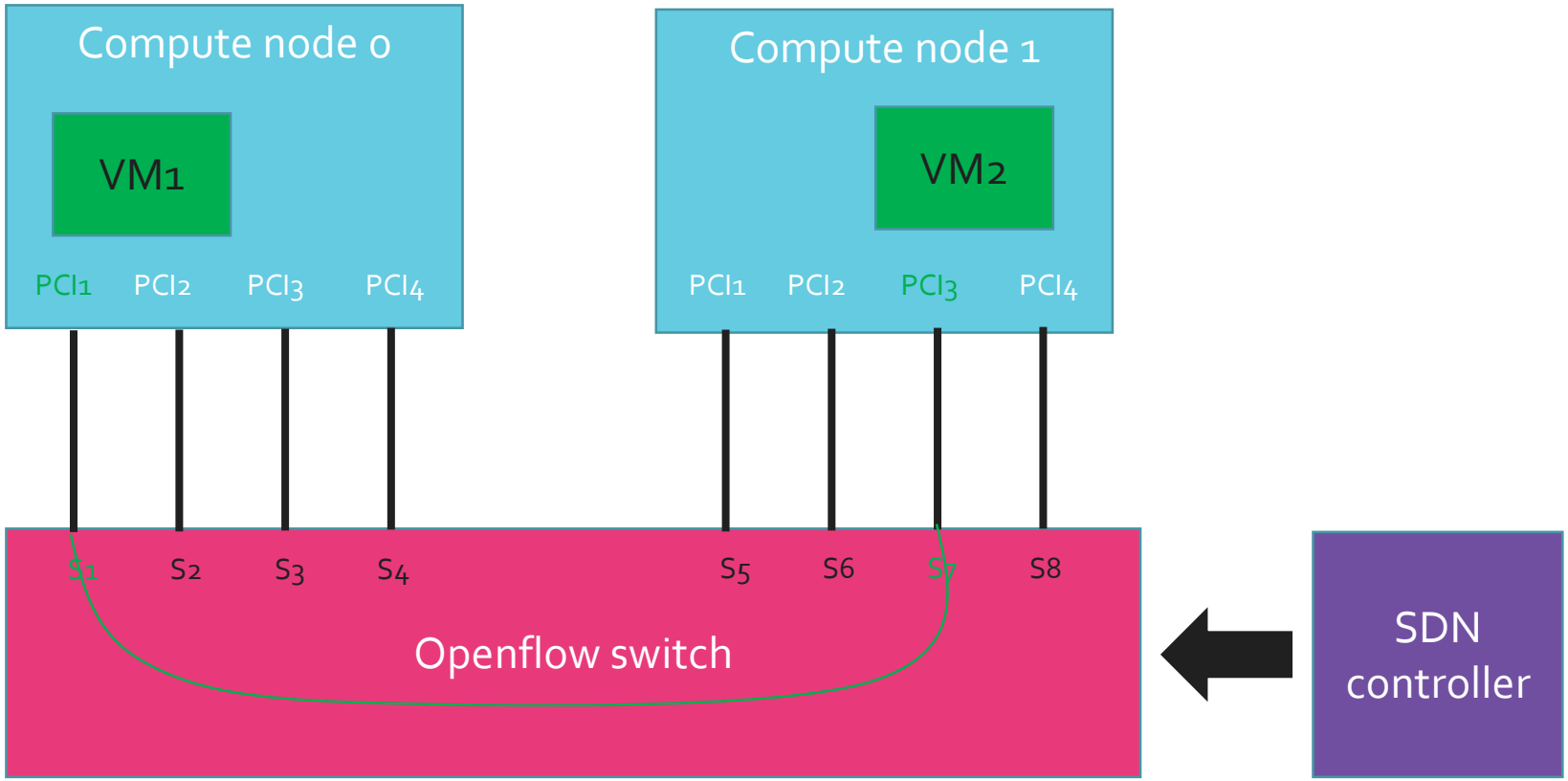
SDN assist. The way it works
2) OSM uses port mapping to identify the ports in the switch



The interface at PCI address PCI1 of compute node 0 is connected to port S1 in the switch

The interface at PCI address PCI3 of compute node 1 is connected to port S7 in the switch

SDN assist. The way it works
3) OSM talks to the SDN controller and connect the ports to the same network.



Example of rules for E-LINE and PCI passthrough interfaces:
IN: Port S1 → OUT: Port S7
IN: Port S7 → OUT: Port S1

Device role tagging

- Whenever a new NS is deployed, and if the VIM allows it, VM interfaces will be tagged with the interface name specified in the descriptor
- This allows proper identification of interfaces in the VM
- A service and a script have to be pre-installed in the VM:
  - RedHat-based VMs: https://github.com/oglok/udev_data
  - Ubuntu-based VMs: https://github.com/gcalvino/udev_data

Adding new VIM account: openstack2-epa

- VIM:
  - openstack2-epa: 172.21.2.22
- Test VIM:
  - ping <IP>
  - curl http://<IP>:5000/v2.0
  - Load Openstack credentials:
    - export OS_AUTH_URL=http://172.21.2.22:5000/v2.0
    - export OS_USERNAME=xxx
    - export OS_TENANT_NAME=xxx
    - export OS_PASSWORD=xxx
  - Run some commands:
    - openstack image list
    - openstack network list
    - openstack flavor list
    - openstack server list

Adding new VIM account: openstack2-epa

- Add a new VIM ‘openstack2-epa’ with the OSM client:
  - osm vim-create --name openstack2-epa --account_type openstack \
    --auth_url http://172.21.2.22:5000/v2.0 \
    --user xxx --password xxx --tenant xxx \
    --description "ETSI openstack site 2, with EPA, with tenant xxx“ \
    --config '{dataplane_physical_net: physnet_sriov, microversion: 2.32}'
  - osm vim-list
  - osm vim-show openstack2-epa
- Config options:
  - dataplane_physical_net:
    - Used to instantiate VMs with SR-IOV and Passthrough interfaces
    - Value: The physical network label used in Openstack both to identify SRIOV and passthrough interfaces (nova configuration) and also to specify the VLAN ranges used by SR-IOV interfaces (neutron configuration).
  - microversion:
    - Used for device role tagging
    - Value: 2.32

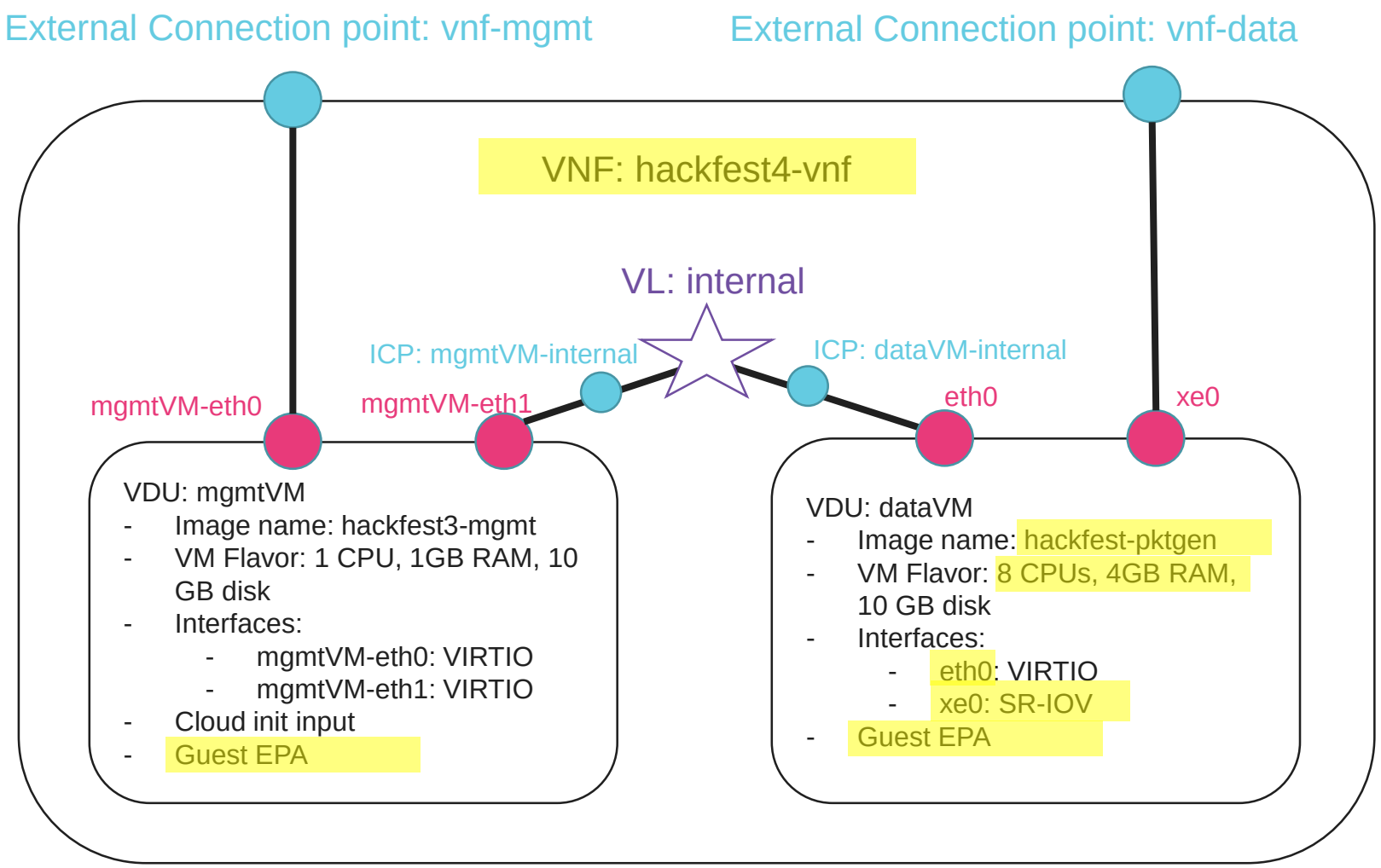
Adding an SDN controller

- Two step process:
  - Add an SDN controller
  - Update a VIM to use that SDN controller with a specific port mapping

- Port mapping file:
  - https://osm-download.etsi.org/ftp/osm-4.0-four/3rd-hackfest/other/port-mapping-openstack2.yaml

- Instructions:
  - osm sdnc-create --name mysdnctler --type floodlight --ip_address xxxx --port xxxx --switch_dpid xxxxxx
  - osm vim-update openstack2-epa --sdn_controller mysdnctler --sdn_port_mapping <PORT_MAPPING_FILE>

VNF diagram
Changes highlighted in yellow



Creating the VNF (1/2)

- Use the IM tree representation of VNFD as a reference:
  - http://osm-download.etsi.org/ftp/osm-doc/vnfd.html
- Copy VNF hackfest3 folder to hackfest4
- Rename descriptor file
- Edit the descriptor
  - Name: hackfest4-vnf
  - Modify VDU mgmtVM:
    - Guest EPA
      - Mempage size: LARGE
      - CPU pinning policy: DEDICATED
      - CPU thread pinning policy: **PREFER (HW threads)** or ISOLATE (cores)
      - NUMA Policy: numa-aware
        - Node CNT: 1
        - Mem policy: STRICT

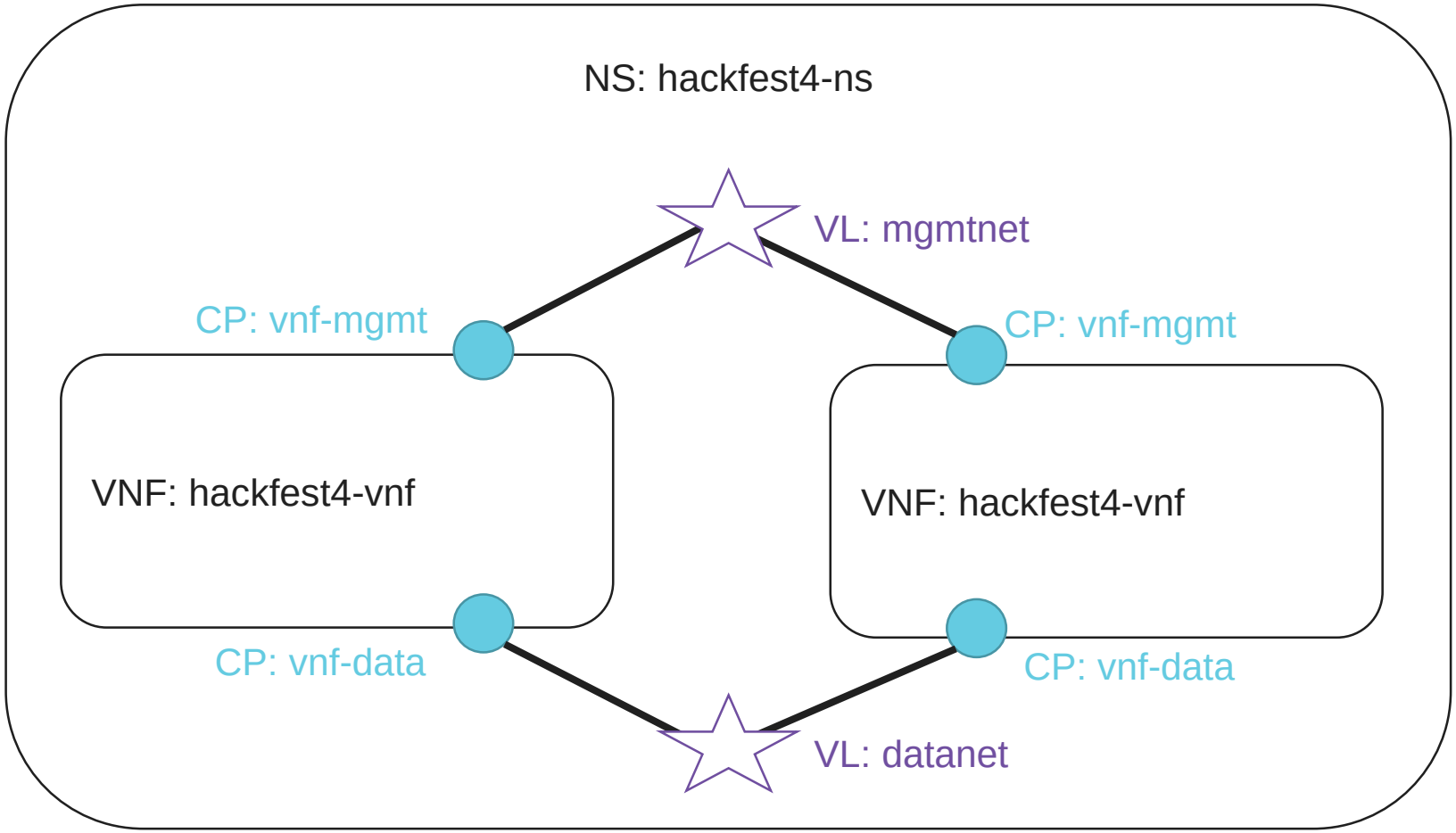
Creating the VNF (2/2)

- Modify VDU dataVM:
  - Image name: hackfest-pktgen
  - Flavor:
    - 8 CPUs
    - 4096 MB RAM
    - 10 GB disk
  - Guest EPA
    - Mempage size: LARGE
    - CPU pinning policy: DEDICATED
    - CPU thread pinning policy: **PREFER (HW threads)** o ISOLATE (cores)
    - NUMA Policy: numa-aware
      - Node CNT: 1
      - Mem policy: STRICT
  - Modify interfaces of the VDU
    - Interface 1:
      - Name: eth0
    - Interface 2:
      - Name: xe0
      - Virtual-interface:
        - Type: SR-IOV

Validate the VNF descriptor and generate VNF package

- Validate VNF descriptor
  - /usr/share/osm-devops/descriptor-packages/tools/validate_descriptor.py <DESCRIPTOR_FILE>
- Generate VNF package (from parent folder)
  - /usr/share/osm-devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd -N <VNFD_FOLDER>

NS diagram



Creating the NS

- Use the IM tree representation of NSD as a reference:
  - http://osm-download.etsi.org/ftp/osm-doc/nsd.html
- Copy NS hackfest3 folder to hackfest4
- Rename descriptor file
- Edit the descriptor
  - id: hackfest4-ns
  - name: hackfest4-ns

Validate the NS descriptor and generate NS package

- Validate NS descriptor
  - /usr/share/osm-devops/descriptor-packages/tools/validate_descriptor.py <DESCRIPTOR_FILE>
- Generate NS package (from parent folder)
  - /usr/share/osm-devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t nsd -N <NSD_FOLDER>

Before the deployment
Adding VNF and NS packages

- VNF package:
  - osm vnfd-create hackfest4_vnfd.tar.gz
- NS package:
  - osm nsd-create hackfest4_nsd.tar.gz

Deploying NS

- Instantiate NS:
  - osm ns-create --ns_name hf4 --nsd_name hackfest4-ns \
    --vim_account openstack2-epa
- Check VNF instances to see the instance and get the mgmt IP address of the VNF
  - osm vnf-list
  - osm vnf-show …

Testing NS (1/2)

- SSH to the mgmtVM of one of the VNFs
- Check that cloud-init worked in mgmtVM
- Check interface names in mgmtVM
  - They must be different than the ones in the descriptor (no udev metadata service)
- Jump to dataVM:
  - ssh -i test4.pem <IP_dataVM>
- Check interface names in mgmtVM
  - They must be the same names as in the descriptor
- Run ‘pktgen’

Testing NS (2/2)

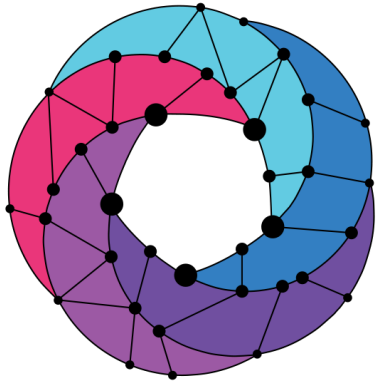
- SSH to the mgmtVM of the second VNF
- Check that cloud-init worked in mgmtVM
- Check interface names in mgmtVM
  - They must be different than the ones in the descriptor (no udev metadata service)
- Jump to dataVM:
  - ssh -i test4.pem <IP_dataVM>
- Check interface names in mgmtVM
  - They must be the same names as in the descriptor
- Run ‘pktgen’

Instructions to run ‘pktgen’

- Configure interfaces to use DPDK:
  - cd /opt/dpdk-17.11
  - sudo -E ./usertools/dpdk-devbind.py --status
  - sudo -E ./usertools/dpdk-devbind.py --bind=igb_uio xe0

- Run pktgen:
  - cd /opt/pktgen-3.4.5/
  - sudo -E ./app/x86_64-native-linuxapp-gcc/pktgen -n 3 -l 2-7 -- -P -m "[4-7].0“

- From pktgen console, set the destination MAC address to the one used by xe0 interface in the other VNF, and send traffic:
  - set all dst mac xx:xx:xx:xx:xx:xx <- destination MAC where to send traffic
  - start all <- to start traffic
  - stop all <- to stop traffic



Open Source
MANO

Find us at:

osm.etsi.org
osm.etsi.org/wikipub

ETSI