EASY TOOLS

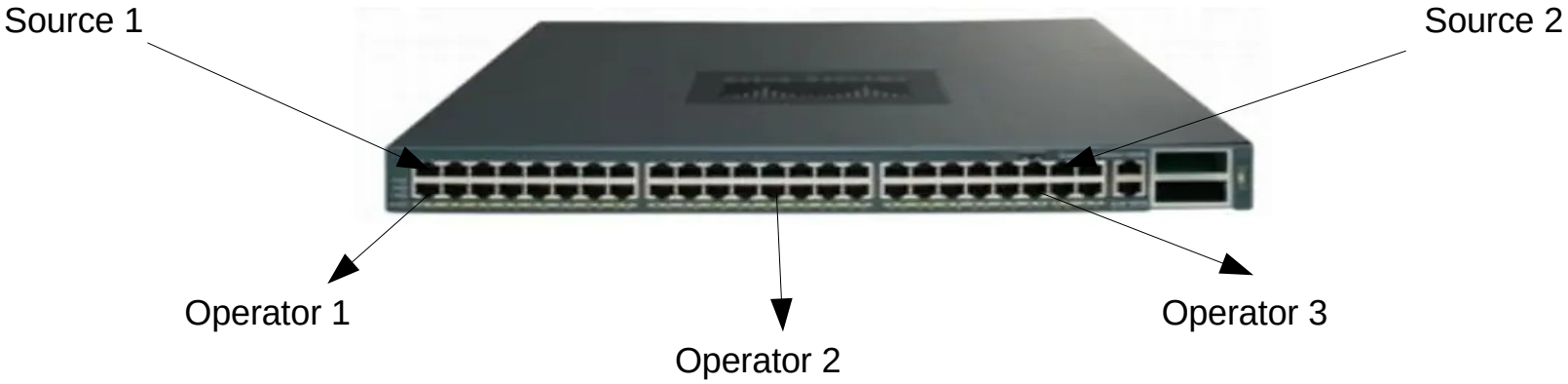
Christophe Massiot
cmassiot@upipe.org

Distributing multicast channels to 3rd parties

A case study with OSS and virtualization/SR-IOV

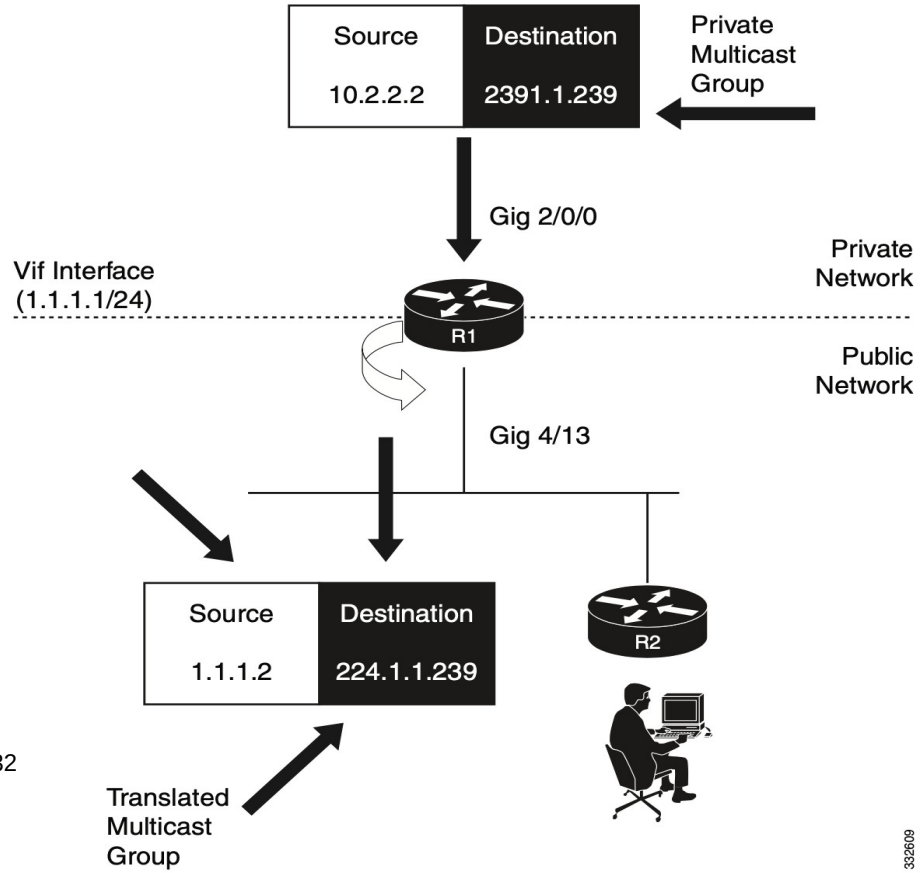
Context

- Linear channels = MPEG-TS over UDP/RTP multicast
- Well-known points of exchange (ex. Telehouse in Paris)
- Each source or destination in its own VLAN



Pure-network solution

- « Multicast Service Reflection »
- Not widely available
- Cannot add/remove RTP or more complex use cases



ip service reflect GigabitEthernet2/0/0 destination 239.1.1.100 to 225.1.1.100 mask-len 32 source 1.1.1.1

« Broadcast » solution

OSS alternative: DVBlast

- Originally a DVB demux but supports multicast:

```
dvblast -D
@239.56.157.11:5004/ifaddr=192.168.56.214/ifname=eth3
```

- Forward a stream via a config file:

```
239.1.101.23:5004@172.23.1.114     0     *
```

- Options to turn on/off RTP, remap PIDs, SID, service name, spoof source address...

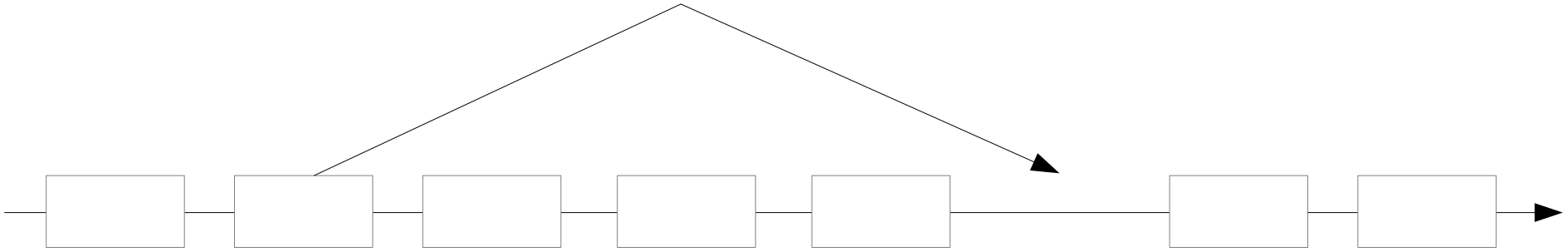
Virtualization

- Proxmox graphical frontend over KVM
- Client isolation
- Some clients have SSH access to their VM
- Every VLAN is bridged with a « virtio » network interface

PROXMOX

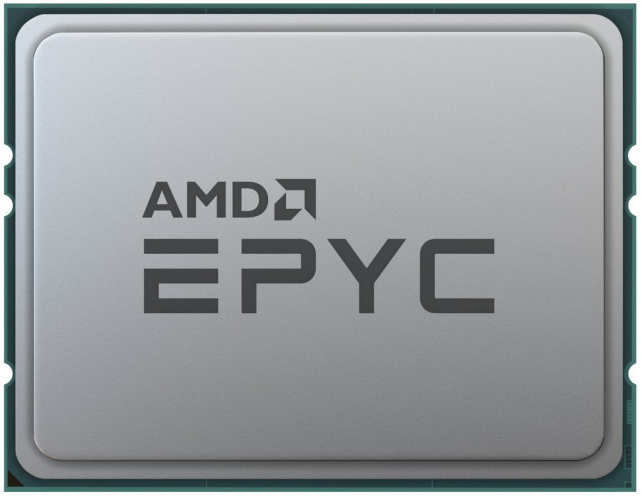
« There are discontinuities »

- Virtio reorders packets
- UDP packet order is not guaranteed but the media industry relies on it

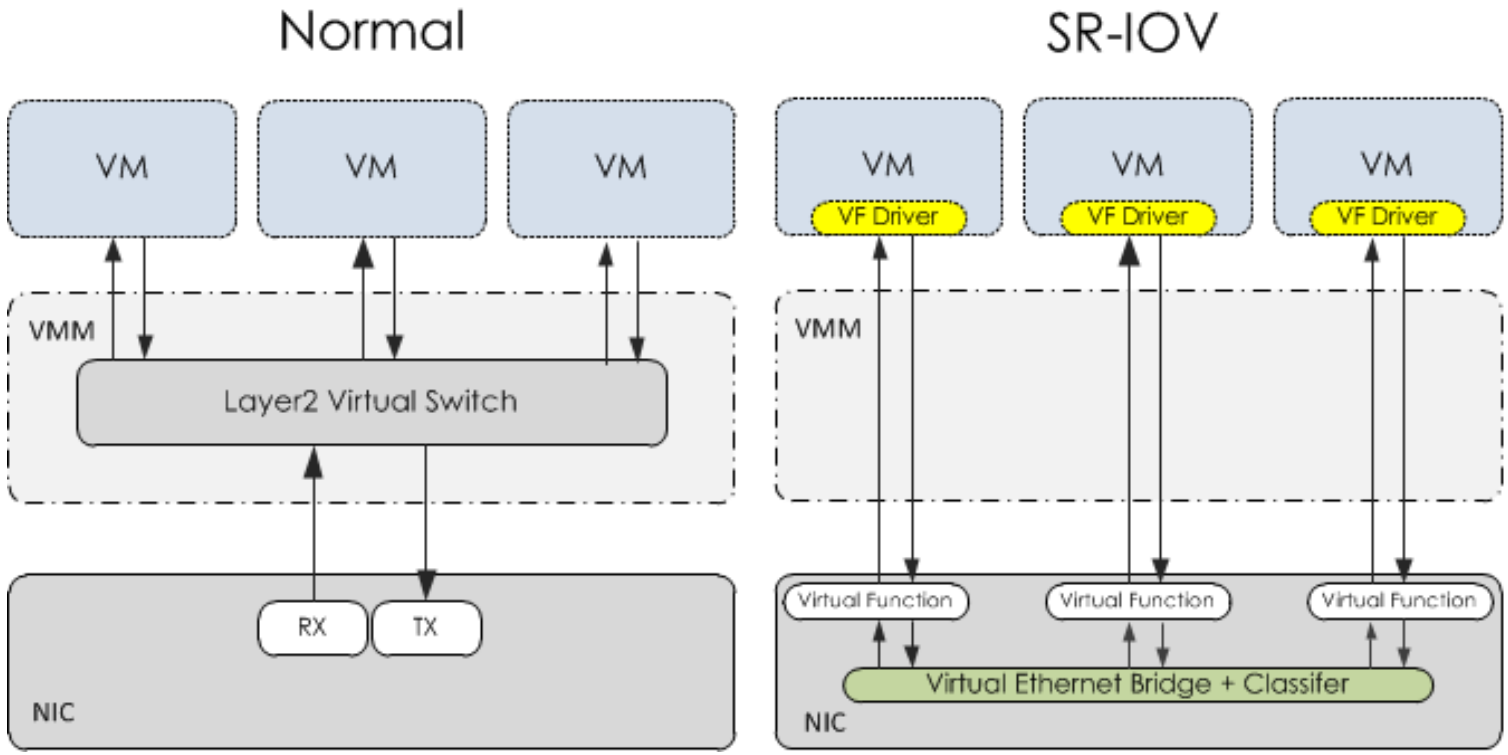


First workaround

- Using driver vmxnet3 instead of virtio solves the problem
- 30 % more CPU
- Need to optimize



SR-IOV and VT-d



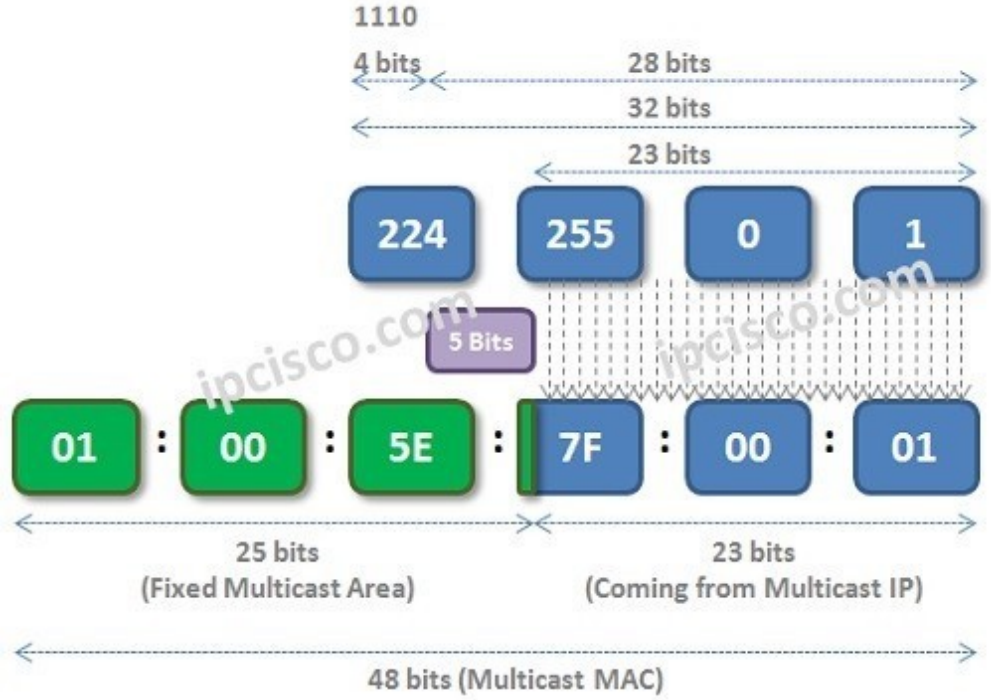
Source: https://honser.github.io/openstack/2019/10/31/Provisioning-a-VM-with-SR-IOV-Interfaces/

Using SR-IOV

- Requires support from: motherboard, CPU, BIOS and network card
- Enable IOMMU
- Upgrade card firmware and drivers
- Create virtual functions:
  `echo 64 > /sys/class/net/eth1/device/sriov_numvfs`
- Pass-through the virtual devices to VMs
- VMs have potential access to all VLANs

Receiving multicast with SR-IOV

- A MAC filter is set-up on the multicast MAC address
- Limits on the number of MAC filters
- VM is drawing dead if it reaches the limit



Source: https://ipcisco.com/lesson/multicast-mac-addresses/

First workaround: promiscuous mode

- VM receives all packets
- Increases CPU usage
- Increases the load on the network adapter
- `echo add mcast > /sys/class/net/p1p2/device/sriov/1/promisc`

Second workaround: vlan_mirror

- Sends all traffic from a VLAN to a VM
- One VLAN can only be sent to one VM
- Encourages VLAN isolation of the sources
- `echo add 2,4,6,18-22 > /sys/class/net/p1p1/device/sriov/3/vlan_mirror`

Third workaround: virtio

- Revert back to virtio for reception
- The bridge includes IGMP snooping to reduce load
- No packet inversion on RX
- No other solution if vlan_mirror cannot be used

Receiving multicast from another VM

- Packets sent with SR-IOV are not looped back to vlan_mirror or virtio
- Workaround: mirroring all egress traffic of a VF to the input of another VF
- `echo add 7 > /sys/class/net/p1p2/device/sriov/1/egress_mirror`

Conclusions

- Multicast on virtualized environments is no picnic
- This is the summary of years of work, with side effects on production