Network performance analysis in virtualized environments

Adel Belkhiri Michel Dagenais

May 5, 2019

Polytechnique Montréal

Laboratoire DORSAL

Agenda

Introduction

Investigations

- Main Concepts in Virtual Networking
- Preliminary Results
- Use cases

Demo

Conclusion and ongoing work

Context of the topic

- Most traditional network monitoring tools are not compatible with virtual environments (cloud computing) :
  - No support for some cloud computing properties (live migration, etc.)
  - Some performance metrics not adapted to virtual networks specificities.

Motivation

1) Profile different technologies used in networking virtualization : para-virtualized network cards, Linux Bridges, Open vSwitch, etc.

2) Analyze the performance of virtualized networks based on relevant performance metrics.

3) Propose efficient tools to help administrators (in IaaS environments) to

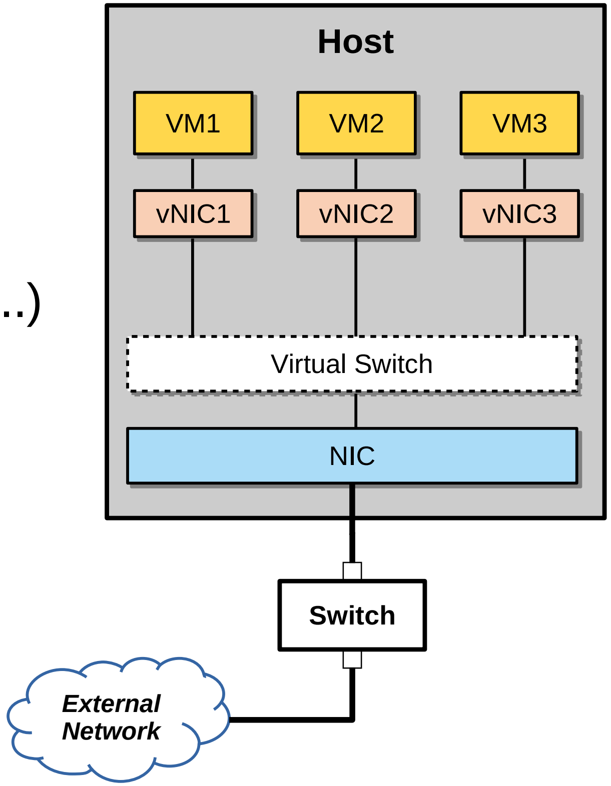
- Identify bottlenecks in virtual networks
- Understand the causes of latencies
- Troubleshoot networking problems

Main Concepts

- A VN (**Virtual Network**) is made of virtualized network components (such as network cards, switches, and routers).

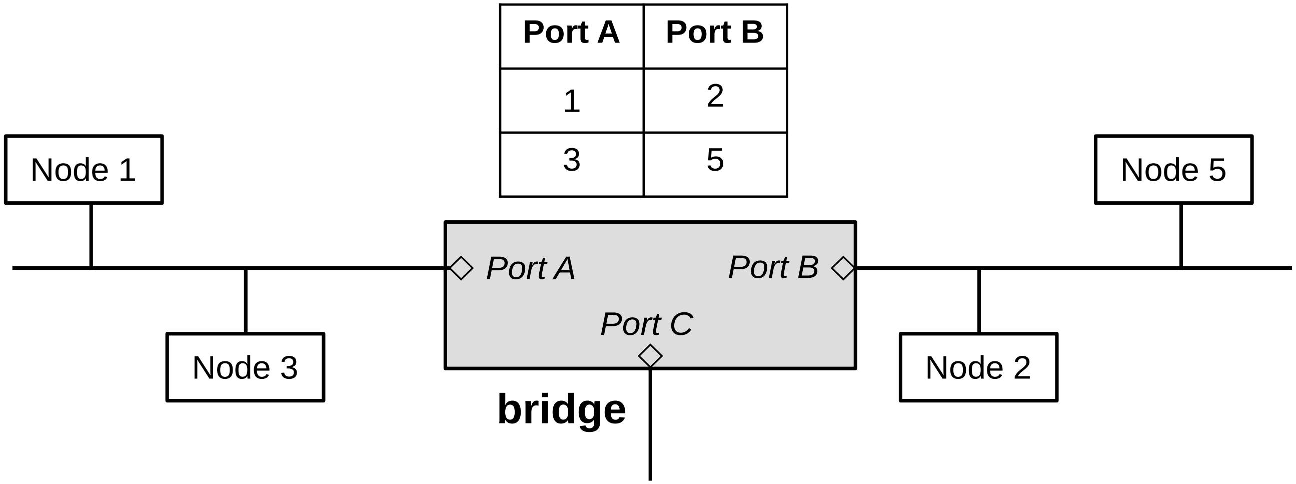
- **NIC virtualization :**
  - Full virtualization (e1000, ...)
  - Para-virtualization (virtio-net, vhost-net, ...)
  - Hardware-assisted virtualization (SR-IOV, VMDq, ...)

- **Switch virtualization :**
  - Linux bridges
  - Open vSwitch



Main Concepts

- **Switch/Bridge Operating :**
  - **Learning :** determine which machine is behind a specific port and store the information in its FDB (*Forwarding DataBase*) table.
  - **Flooding** : send the packet to all its ports (except the one from which the packet was received)
  - **STP** : protocol to disable ports when wiring loops are detected.

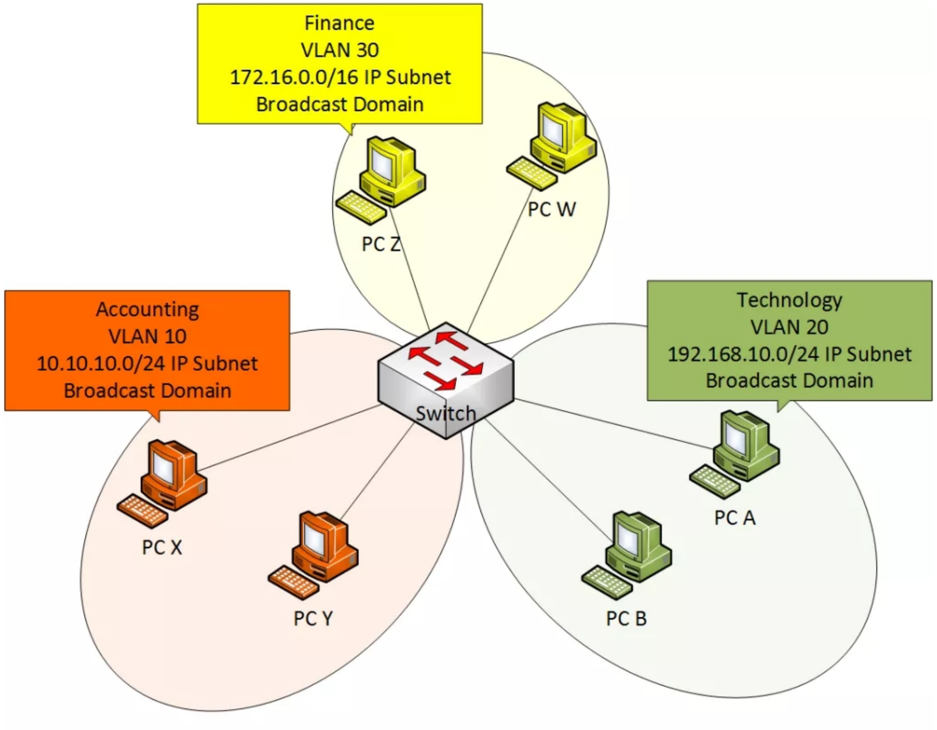


Port A	Port B
1	2
3	5

Main Concepts

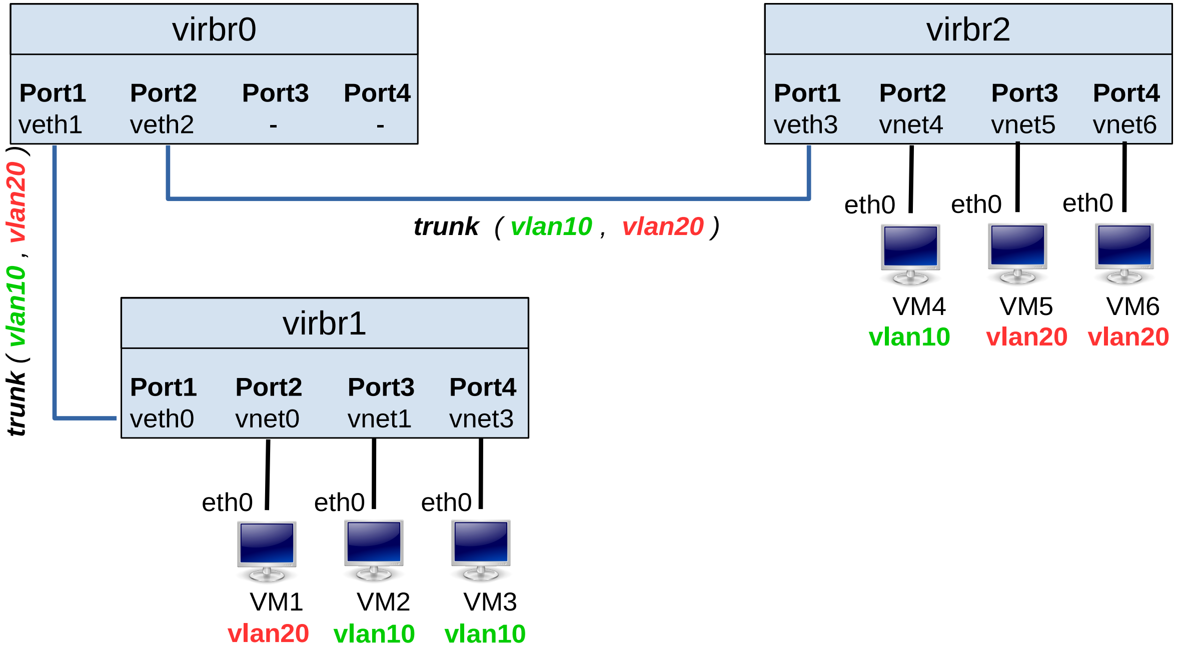
- **What is a VLAN ?**

“A virtual LAN (VLAN) is any broadcast domain that is partitioned and isolated in a computer network at the data link layer (OSI layer 2).” [1]



Main Concepts

- **How to create a VLAN ?**
  - The extent of a VLAN can span to more than one switch.



```
$ sudo bridge vlan add vid 20 pvid untagged dev vnet0
```

Work Environment

- **Virtualization Software :**
  - Qemu, KVM and libvirt
  - Vhost-net and Linux bridges
- **Data Collection :**
  - Tracing using LTTng (*Linux Trace Toolkit next generation*)
  - Kernel space tracepoints and tracing limited to the host machine
- **Performance Analyses :**
  - Trace Compass framework
  - Python babeltrace bindings

Preliminary Results

- Except for the **Stream List view**, Trace Compass does not offer any other performance analysis related to networks activities !!

☹ Moreover, this view is only dedicated to **PCAP traces**.

1) Generic Analyses : Analyses compatible with physical and virtual networks.

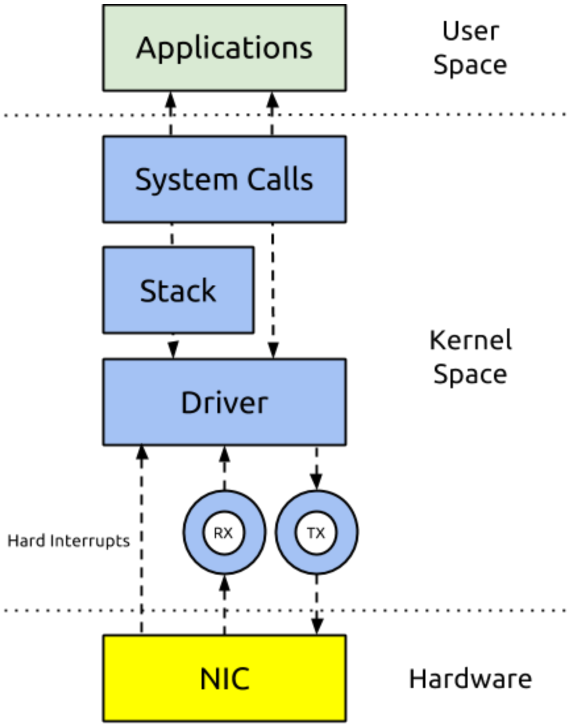
- Tx/Rx bandwidth per NIC
- Packet drop rate
- Packet offloading rate
- And a new Stream List view ;)

2) Specific Analyses : Analyses compatible with VNs only :

- Automatic topology discovery
- Network traffic flow analysis

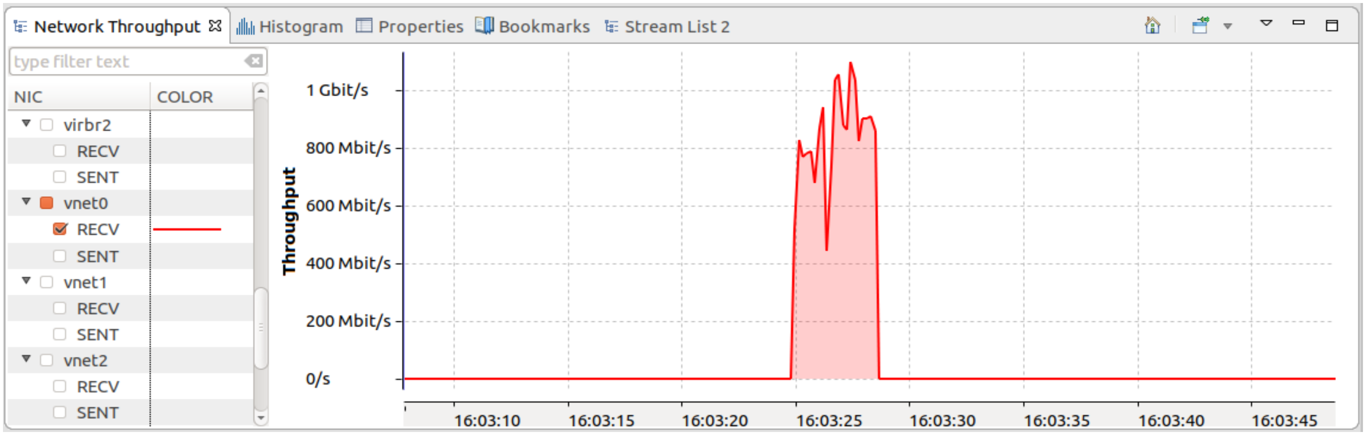
Generic Analyses

- **Many events describe the journey of packets through Linux networking stack :**
  - lttng_statedump_network_interface
  - napi_poll
  - napi_gro_receive
  - net_if_receive_skb
  - skb_copy_datagram_iovec
  - skb_consume
  - skb_free
  - net_dev_queue
  - net_dev_start_xmit
  - net_dev_xmit



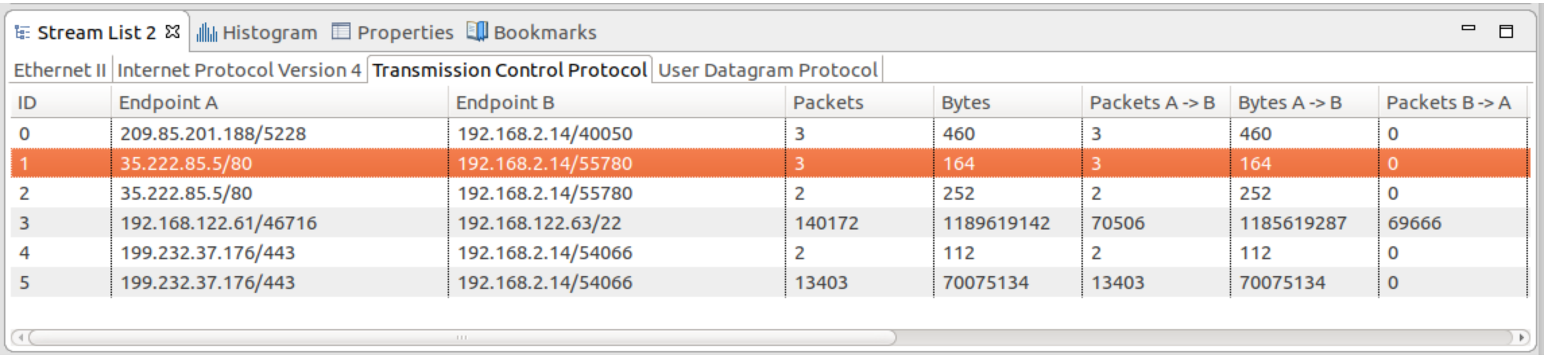
Generic Analyses

- Network Bandwidth view



NIC	COLOR
▼ □ virbr2	
□ RECV	
□ SENT	
▼ ■ vnet0	
☑ RECV	🟥
□ SENT	
▼ □ vnet1	
□ RECV	
□ SENT	
▼ □ vnet2	
□ RECV	
□ SENT	

- New “Stream List” view in Trace Compass



ID	Endpoint A	Endpoint B	Packets	Bytes	Packets A -> B	Bytes A -> B	Packets B -> A
0	209.85.201.188/5228	192.168.2.14/40050	3	460	3	460	0
1	35.222.85.5/80	192.168.2.14/55780	3	164	3	164	0
2	35.222.85.5/80	192.168.2.14/55780	2	252	2	252	0
3	192.168.122.61/46716	192.168.122.63/22	140172	1189619142	70506	1185619287	69666
4	199.232.37.176/443	192.168.2.14/54066	2	112	2	112	0
5	199.232.37.176/443	192.168.2.14/54066	13403	70075134	13403	70075134	0

Automatic Topology Discovery (1)

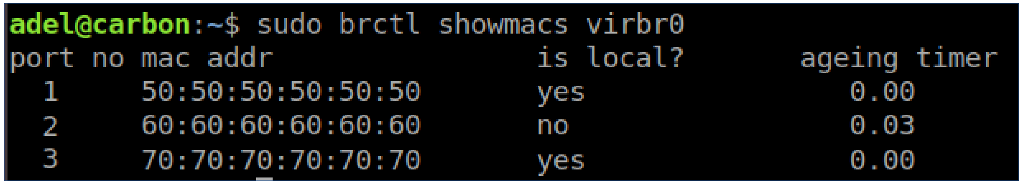
- Most of network diagnostic tools rely on the topology to understand network element dependencies.

- Manual mapping is not practical due to the size and dynamic behavior of virtual networks.

- Many algorithms and protocols were devised to automatically discover the topology of traditional networks :
  - ***SNMP*** (*Simple Network Management Protocol*) and ***MIB*** (*Management Information Base*) data.
  - ***LLDP*** (*Link Layer Discovery Protocol*) allows neighboring devices to become aware of each other and populate their MIBs
  - ***CDP*** (*Cisco Discovery Protocol*) : proprietary protocols

Automatic Topology Discovery (1)

- **To discover the topology of VNs, our analysis uses the events describing the transactions applied on bridges FDBs (*Forwarding DataBase*) entries :**
  - br_fdb_update
  - br_fdb_add
  - br_fdb_external_learn_add
  - fdb_delete
  - lttng_statedump_network_interface
  - *lttng_statedump_network_bridge**



```
adel@carbon:~$ sudo brctl showmacs virbr0
port no mac addr                is local?       ageing timer
  1     50:50:50:50:50:50       yes                0.00
  2     60:60:60:60:60:60       no                 0.03
  3     70:70:70:70:70:70       yes                0.00
```

```
[16:18:02.065703012] (+0.000005589) carbon br_fdb_update: { cpu_id = 3 }, { br_dev = "virbr0", dev = "veth0", addr = [ [0] = 0x30, [1] = 0xD6, [2] = 0x25, [3] = 0xF8, [4] = 0x4F, [5] = 0x44 ], vid = 1, added_by_user = 0 }
```

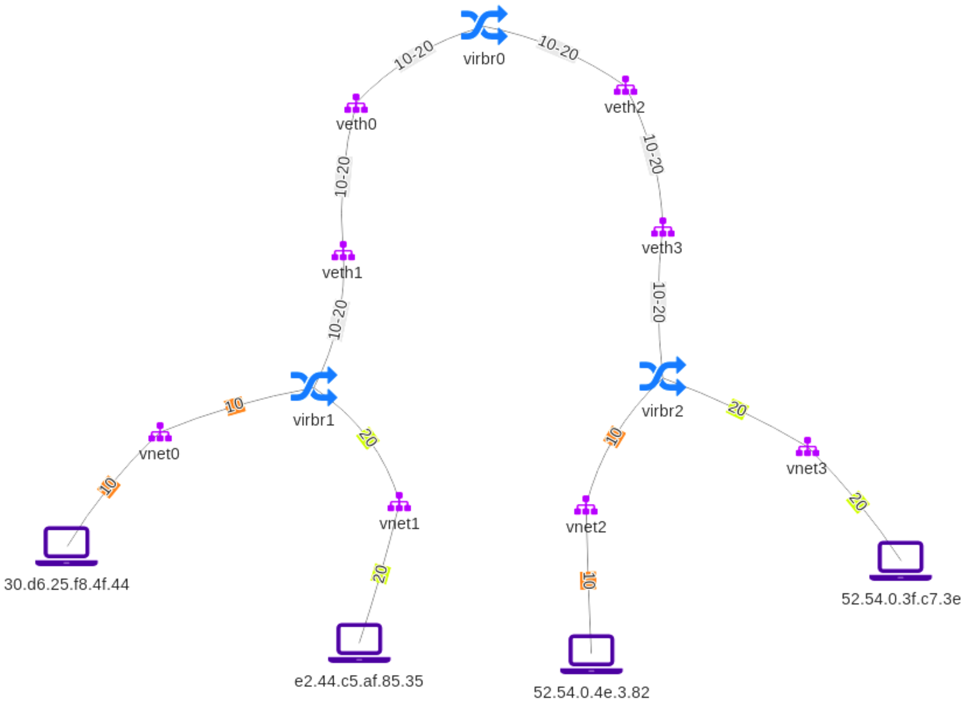
```
[16:17:59.332156065] (+0.000001185) carbon lttng_statedump_network_bridge: { cpu_id = 3 }, { name = "virbr1", hardware_addr = [ [0] = 0x60, [1] = 0x60, [2] = 0x60, [3] = 0x60, [4] = 0x60, [5] = 0x60 ], bridge_type = ( "master" : container = 1 ), enslaved = { ifce_name = "vnet2", ifce_hardware_addr = [ [0] = 254, [1] = 68, [2] = 197, [3] = 175, [4] = 133, [5] = 53 ] } }
```

Automatic Topology Discovery (2)

- We implemented and adapted the algorithm published in [2] to discover the topology of VNs based on FDBs tracing events.

  ✔ Dynamically detect changes in the topology (after a live migration of a VM for example).

  ✔ Discover associated VLANs.



Traffic Flows in VNs (1)

- How can we track and measure network traffic between VMs hosted in the same host ?

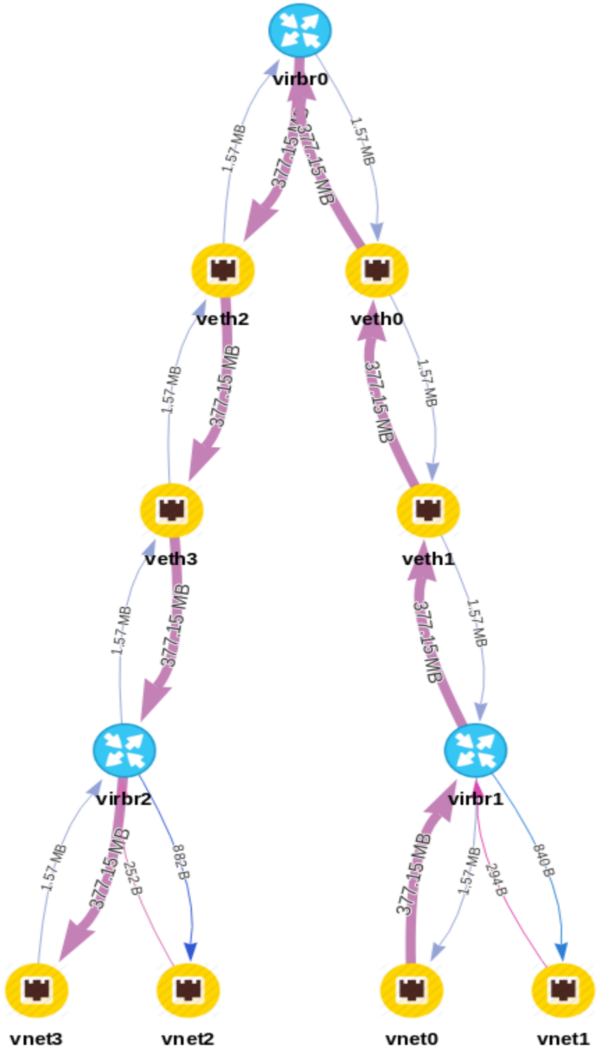
- **LTTng tracepoints harnessed for this analysis :**
  - net_if_receive_skb
  - net_dev_xmit
  - skb_kfree
  - skb_consume
  - *br_forward_skb_entry**

```
[16:17:59.689844924] (+0.000005409) carbon br_forward_skb_entry: { cpu_id = 2 },
{ skbaddr = 0xFFFF9CBE890B8100, len = 28, name = "virbr2", local_orig = 0 }
```

Traffic Flows in VNs (2)

Algorithm :

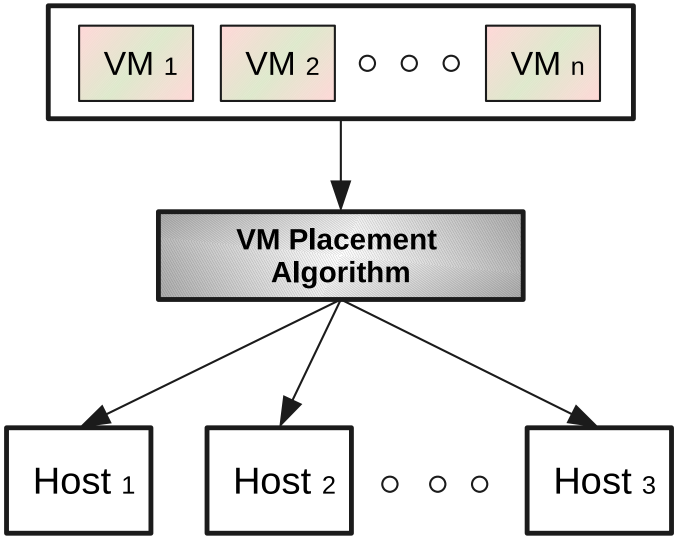
```
node_list ← {}
IF (event.name == net_if_receive_skb) OR (event.name ==
net_dev_xmit) OR (event.name == br_forward_skb_entry)
THEN
    skbaddr ← event['skbaddr']
    dev ← event['name']
    size ← event['len']
    add_node_to_list (node_list, skbaddr, dev, size)
ELSE  IF (event.name == skb_kfree) OR (event.name ==
skb_consume)  THEN
        skbaddr ← event['skbaddr']
        IF (skbaddr in node_list) THEN
            dump_node_list_to_graph (G, node_list[skbaddr])
            delete (node_list[skbaddr])
        ENDIF
    ENDIF
ENDIF
```



Use Cases (1)

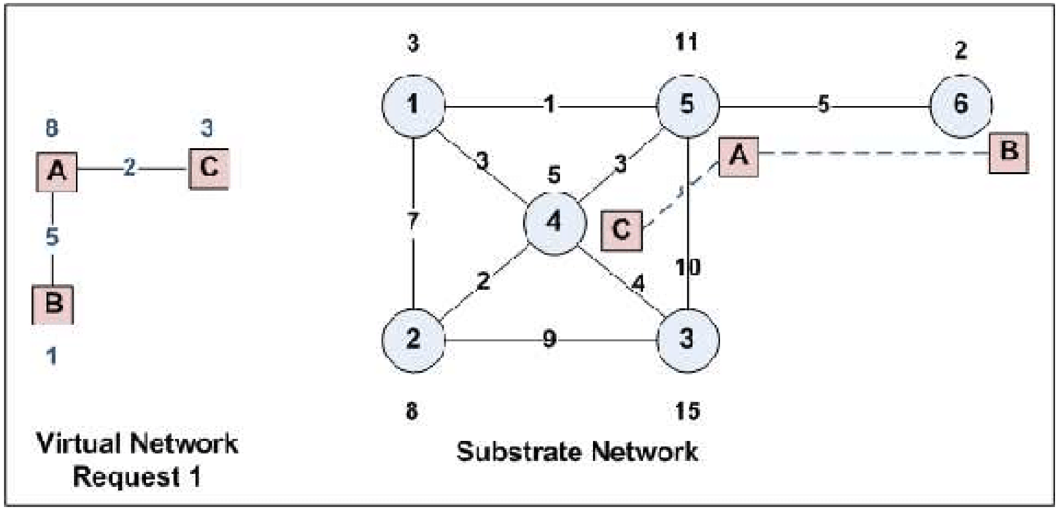
- **VM Placement**

  - It is a part of the VM migration process
  - **Goal :** find the best strategy to maximize resources utilization by mapping VMs to host machines
  - What about preveting congestion in datacenter network ?



Use Cases (2)

- **Virtual Network Resource Mapping**
  - Multiple heterogeneous VNs cohabit on the same shared substrate network.
  - **Goal :** allocate the substrate resources for the VNs with respect to their resource requirements and their topologies.



**Taken from : https://www.researchgate.net/figure/An-example-of-the-resource-mapping-problem_fig1_313455464*

Enough Talk, It's Time for a Demo

Conclusion

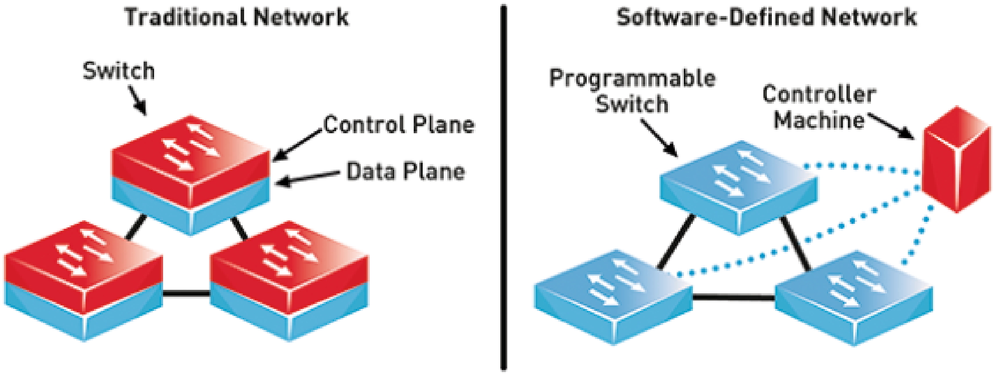
- Industry needs efficient tools to diagnose problems in virtual networks and identify the root causes of their latencies.

- Tracing techniques are great to collect low-level data needed to develop performance analyses specific to virtual networks.

- We are looking for new use cases and problems to solve in order to improve our analyses and tools

Ongoing Work

- **OpenvSwitch**
  - SDN (*Software-Defined Networking*) : Separation, at the hardware level, of the network control plane from the forwarding plane.



**** Taken from :*** *https://www.commsbusiness.co.uk/features /software-defined-networking-sdn-explained/*

Questions?

E-Mail : **adel.belkhiri@polymtl.ca**

References

[1] Introduction To Ethernet VLAN (Free Preview), https://ippacket.com.au/lesson/chapter-1-introduction-to-ethernet-vlan/

[2] "An Efficient Algorithm for Ethernet Topology Discovery in Large Multi-subnet Networks," U. Uzair, H. F. Ahmad, A. Ali and H. Suguri, 2007 IEEE International Conference on System of Systems Engineering, San Antonio, TX, 2007, pp. 1-7.

[3] "An example of the resource mapping problem", https://www.researchgate.net/figure/An-example-of-the-resource-mapping-problem_fig1_313455464

[4] An example of the resource mapping problem, https://www.researchgate.net/figure/An-example-of-the-resource-mapping-problem_fig1_313455464

[5] Software Defined Networking (SDN) Explained, https://www.commsbusiness.co.uk/features/software-defined-networking-sdn-explained/