

# High-Performance Virtualized SDN Switches for Experimental Network Testbeds

Richard Cziva

NORDUnet / Uni. of Glasgow

Jerry Sobieski

CRO, NORDUnet / AL, GEANT Testbeds Service

Yatish Kumar

CTO, Corsa Technologies

**INDIS Workshop**

SC16 Salt Lake City

November 13, 2016

- SDN has reached wide academic acceptance
  - OpenFlow has been cited **4876** times so far
  - Many SDN controllers have been proposed and used
- SDN/OpenFlow research continues!
  - In hardware (ironically),
  - In management & control plane services
  - In application layer
- How do we efficiently share SDN switching hardware in a scalable and secure fashion?

- OF offers a One switch-One Controller model
- Thus, sharing an OpenFlow switch has been the “elephant in the room” for years
- Many approaches have been tried
  - Proxy intercept
  - VLAN slicing (layer 2)
  - Port delegation
  - Controller based services
- We assert the problem is lack of virtualization support in OpenFlow switching platforms

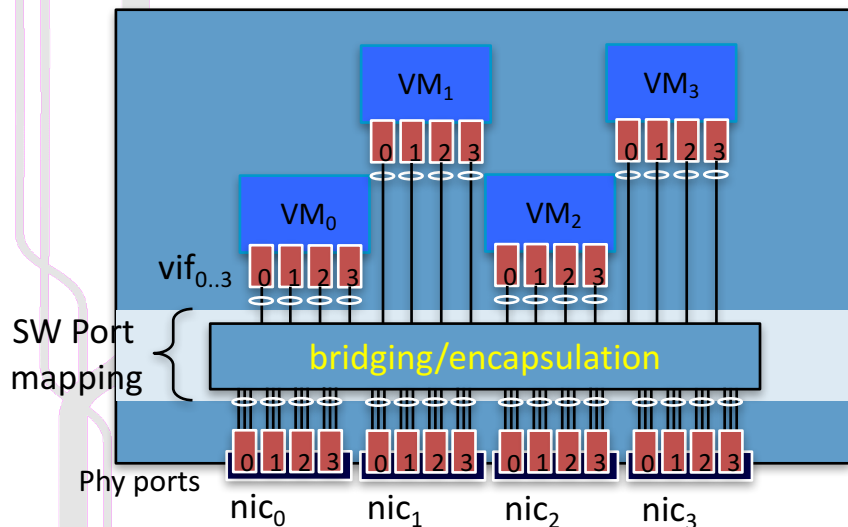


- This work has brought NORDUnet, GEANT and Corsica Technologies together to design and implement **Virtual Switch Instances** (VSIs)
- This paper presents
  - The functionality and benefits of VSIs
  - How we integrated VSIs into the GEANT Testbeds Service (GTS)

- SDN switches do not allow multiple controllers, simultaneously.
- Different SDN applications have different requirements:
  - Forwarding requirements,
  - Switching fanout requirements, and topology
  - Protocol requirements
- This is especially true of “on-ramp” R&D environments
  - E.g. AL2S, GENI, FIRE, AL2S, GEANT Testbeds Service, ...

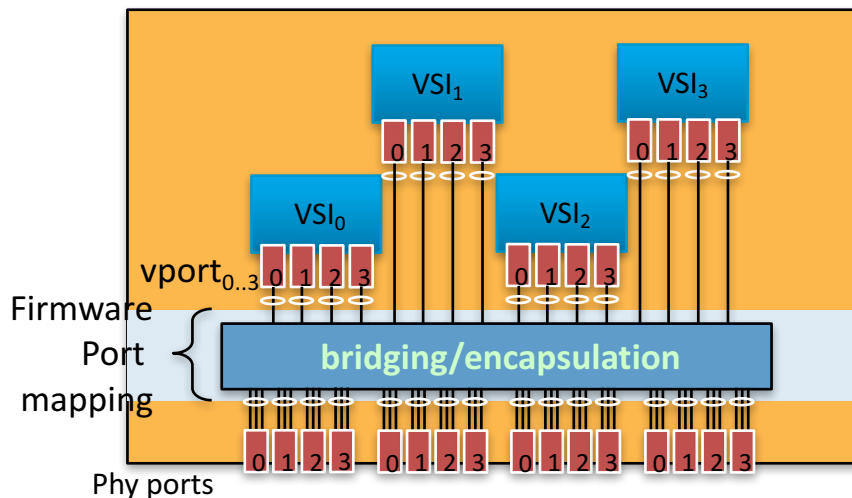
- Solution: Dis-associate and abstract switch attributes from the physical mapping
- -> Virtualized Switching Instances (VSIs)
- Each VSI has its own OpenFlow context
  - Separate controller, protocol version, IPAddr
  - Full network flow space, counters, etc.
  - Deterministic fabric forwarding performance
- Each VSI has its own set of **Virtual Ports**
  - Implications are complex

## Physical Server Platform



VM Port mapping: phyPort/VLAN > VM/vif,  
Pop tagging (inbound) or push tagging (outbound)

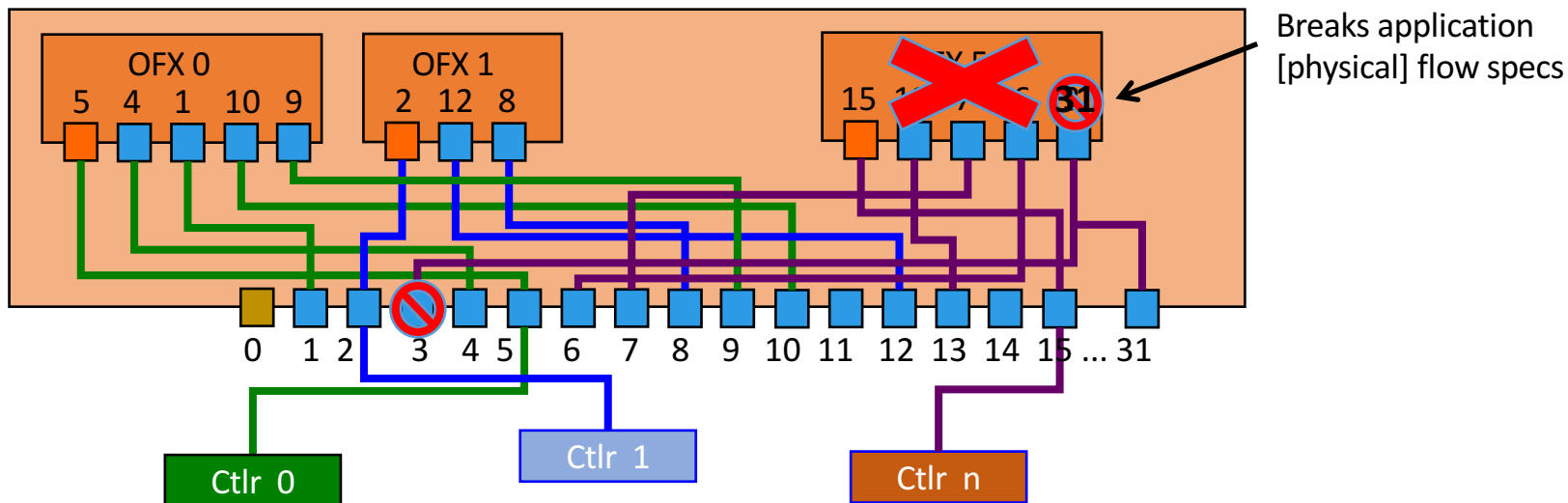
## Physical Switch Platform



VSI Port mapping: phyPort/VLAN > VSI/vport,  
Pop tagging(inbound) or push tagging (outbound)

# Switch Partitioning

## OFX Instances with port partitioning



Pros:

- Each instance has its own controller
- Except for port dimension, the user has full network flow space (no VLAN slicing is needed)

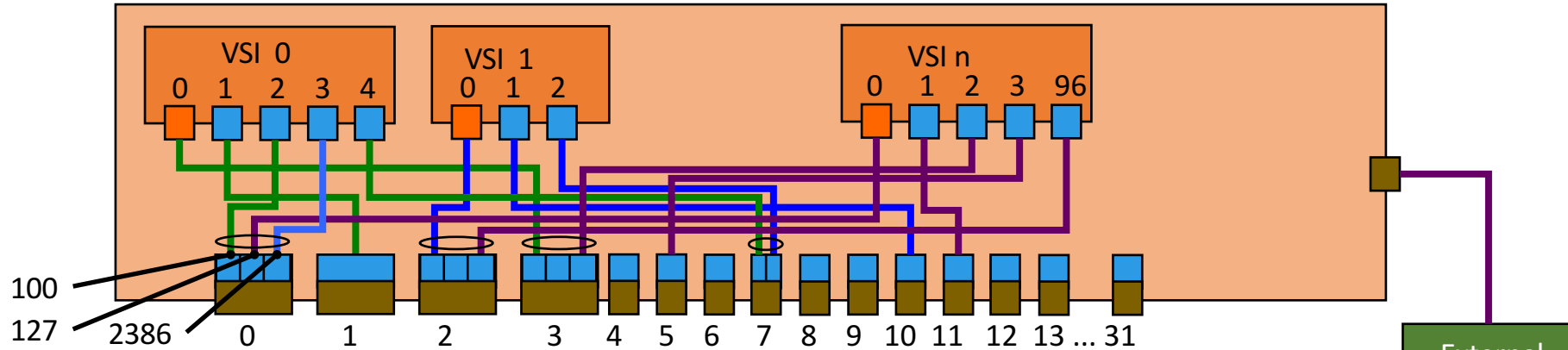
Cons:

- User flowspecs are **physical port** based flowspecs – the instance will break the flowspecs
- Ports cannot be split – the entire port is assigned to an instance



# Virtual Switch Instances – The model

## Virtual Switch with Virtual Circuit port mapping



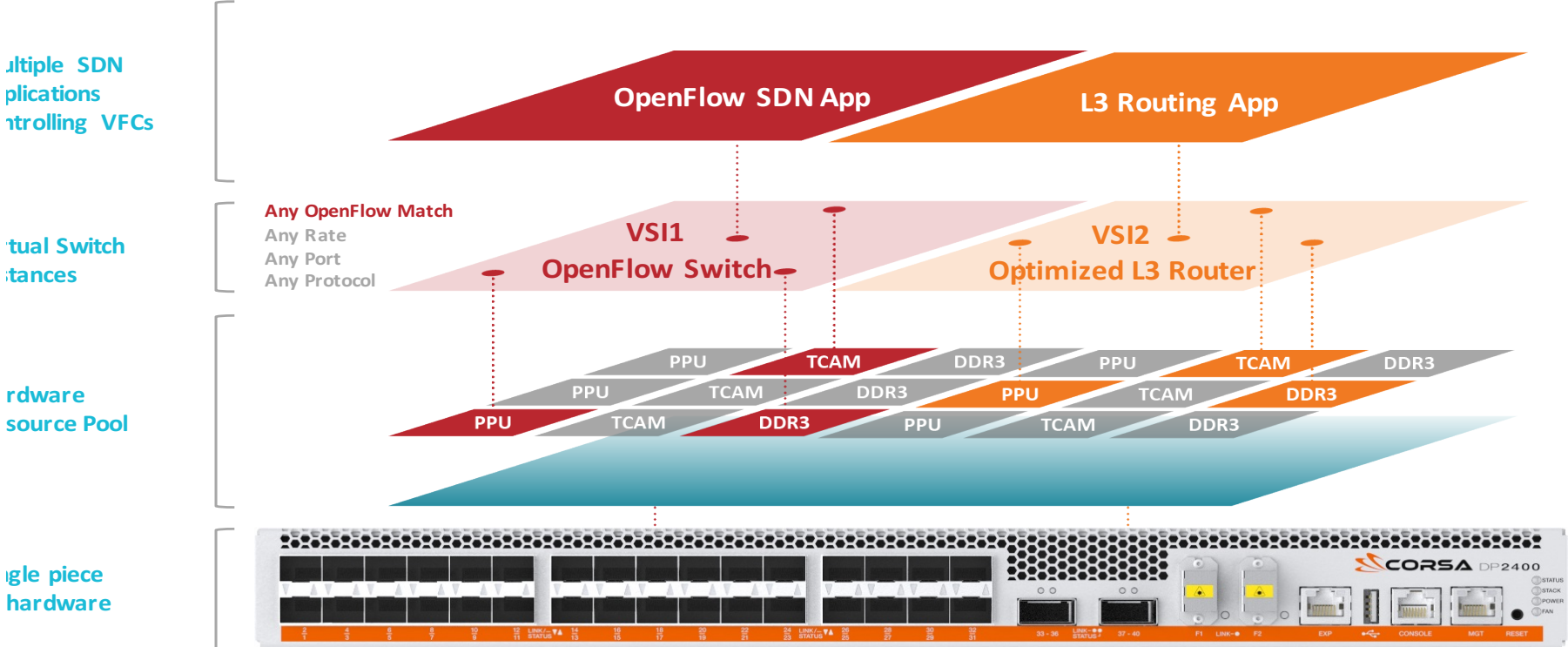
Port, label	-> VSI, vPort	header action
0, 100	0, 2	in: pop qtag; out: push qtag 100;
0, 127	n, 0	In: pop qtag; out: push qtag 127;
0, 2386	0, 3	in: pop qtag; out: push qtag 2386;
1, *	0, 1	in: no action; out: no action;
2, 100	n, 96	in: pop qtag; out: push qtag 96;
2, 3140	1, 0	in: pop qtag; out: push qtag 3140;
3, 25	0, 0	in: pop qtag; out: push qtag 25;
3, 1870	n, 2	in: pop qtag; out: push qtag 1870;

VSIs use virtual flowspecs  
 Allows instances to share a physical port  
 Allows transport tagging to be used for VCs, and to be popped before user sees it.  
 Enables full network flow space. Enables migration and grooming.

- For user virtual flow specs to work the inbound frame must be mapped to the appropriate VSI and appropriate port at line rate.
  - Must be done in the “fast path” – at 100G!
  - Must be a simple FAST operation
  - Must be done for both inbound and outbound traffic

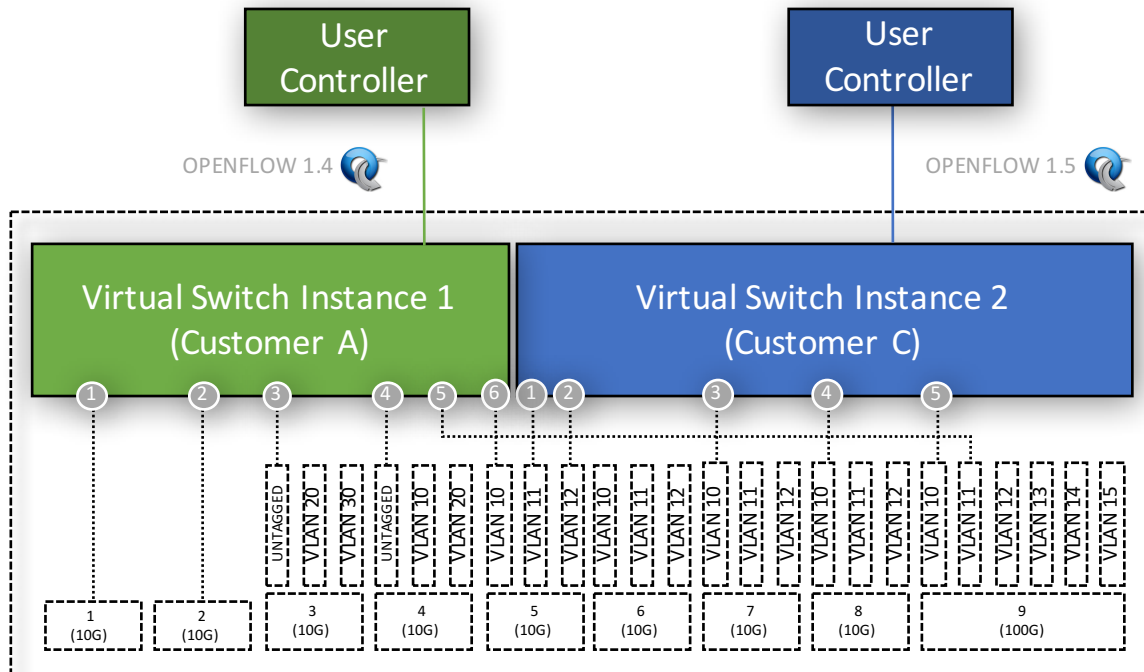
- Key operation: 2-tuple swap – in the fast path
- On ingress:
  - **phyPort / transTag -> VSI / vport;** pop\* transTag
- On egress:
  - **VSI / vport -> phyPort / transTag;** push\* transTag;
- Look up is ~cost as an MPLS label swap ... Very fast
- Pop & Push actions are configurable
- TransTag can be outer VLAN or MPLS label

# Multiple VSIs on one switch



# Hardware design challenges

- Corsa has done some impressive advanced hardware design to support VSIs:
- Increased number of OpenFlow tables
  - Reduction in memory usage
- New algorithmic lookup for flow entries
  - This allows increase in flow table size to 1 Million entries
- Virtualization of QoS, metering and statistics
  - Specialised ASIC performs these
- We will let Corsa describe their work themselves (in another talk😊)

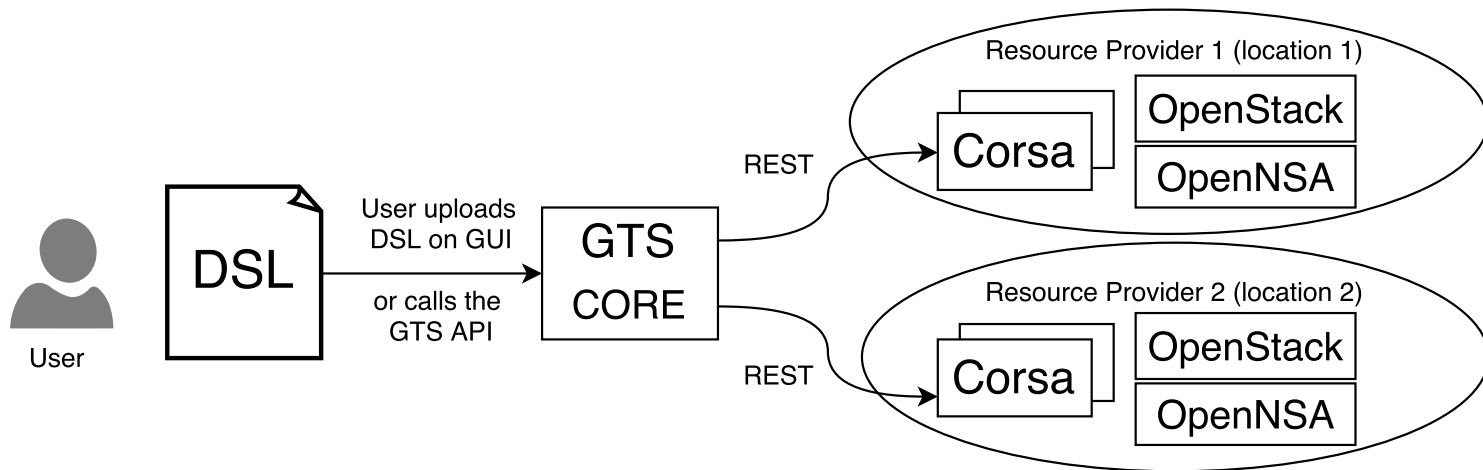


- **VSI**s are “well bounded” service objects
  - They can be allocated securely to arbitrary users
  - Users only see their own traffic
  - Multiple VSIs are hosted on a single device
  - Support full transport encapsulation
- **VSI**s can be migrated
  - Enables operational maintenance of HW
  - Enables grooming of VSI for HW efficiency
- **VSI** 2-tuple mapping enable port / link sharing
- **VSI**s can be applied to native transport tags

- VSIs are seen as dedicated OpenFlow switches
- VSIs run at line rate – even up to 100Gbps(!)
- VSI virtual ports reduced complexity for controllers/applications
- VSIs solve a major festering SDN scaling problem:
  - Inter-domain control authorization
  - Inter-domain topology visibility
- VSI are specified by users to fit their requirements



- VSIs have been integrated to GTS



GTS High-level overview

### Current GTS Pod locations:

- In-service: **Amsterdam, Bratislava, Ljubljana, Prague, London, Milan, Hamburg, Paris, Madrid**

### Current NORDUnet GVS locations

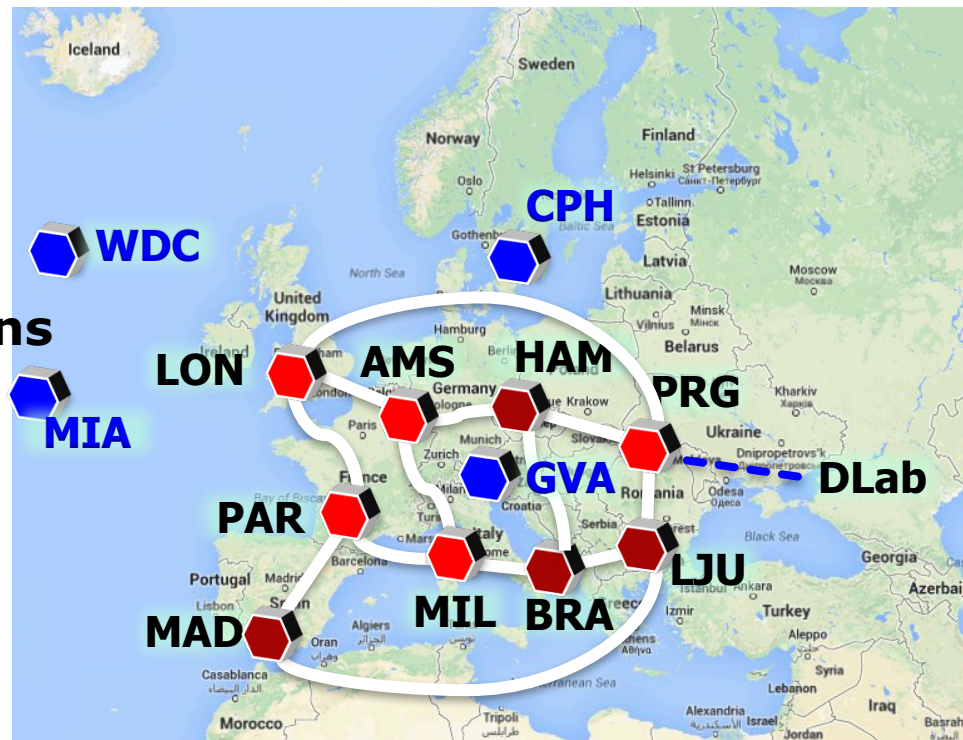
- In Service: **Copenhagen, Geneva, WashingtonDC, Miami**

### Others in the pilots:

- HEAnet:** Dublin
- CESnet:** Prague, Bruno
- DFN:** Nuremburg (Erlangen),

### Other interest:

- StarLight (Chicago), CENIC (Sunnyvale), Ciena(US & CA), others in discussion...



- A DSL can define every parameter of the user's VSI

```
VSI {  
  location="COPENHAGEN"  
  switchIP="10.10.10.2"  
  switchSubnetMask="255.255.255.0"  
  switchDPID="0000000000000001"  
  controllerIP="10.10.10.100"  
  controllerPort="6633"  
  port { ofport=1 id="P1" }  
  port { ofport=2 id="P2" }  
  port { id="CTRL" mode="CONTROL" }  
}
```

Switch DPID

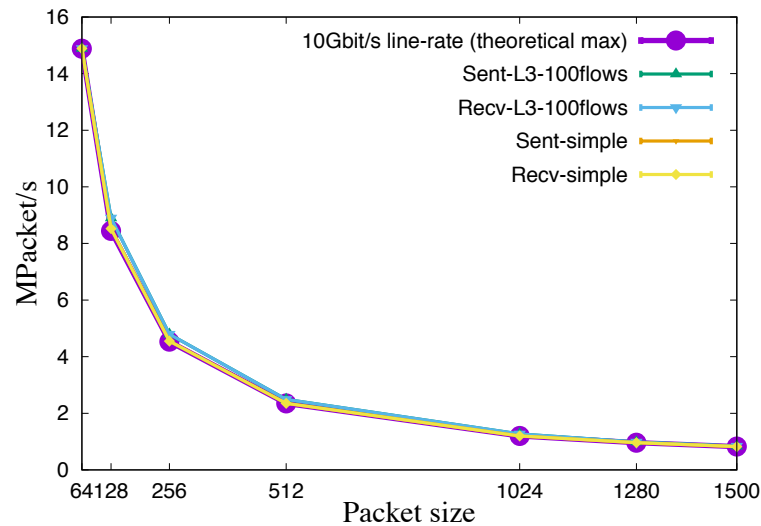
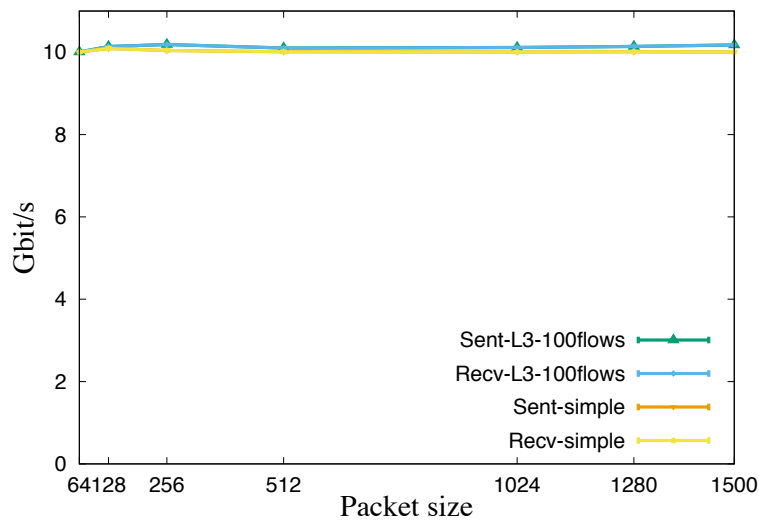
Controller IP, port

Virtual Port ID

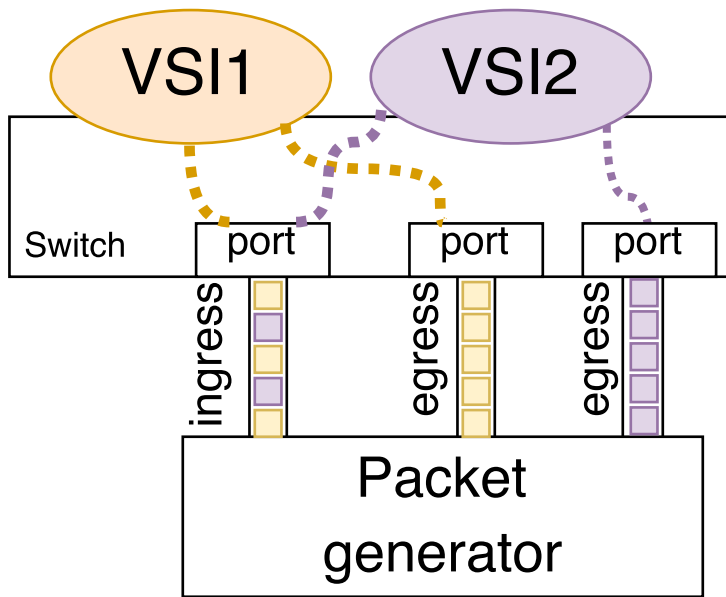
- Performance of VSIs is crucial(!)
- We evaluated throughput of VSIs with various packet sizes
- Used:
  - “Software-Defined Exchange” pipeline on the switches
  - DPDK-pktgen to generate and measure received packets

Two experiments:

- 100flows: 100 L3 flow entries matched
- Simple: input port-output port flow entry matched

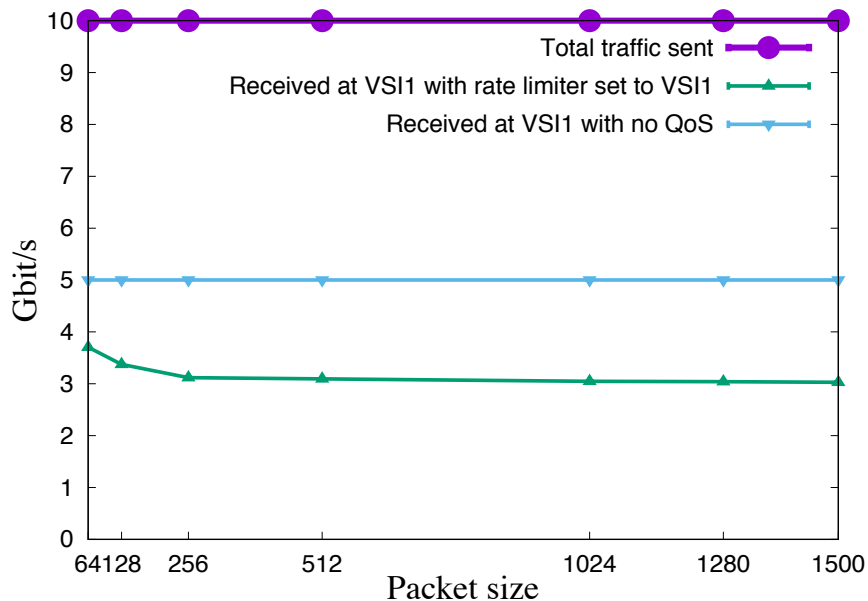


VSIs can share the same physical ports.  
We used this setup to evaluate resource sharing:



Two scenarios measured:

1. No rate limiting set (equal sharing of link)
2. 3Gbit/s rate limiter set to VSI1



- The VSI works and solves a number of SDN challenges
  - Many thanks to **Corsa Technologies** for their collaboration on this!
- **The “VSI” is an open concept.**
  - It is not proprietary, we hope other vendors will adopt it
- VSIs are being deploy[ing] now:
  - Now: NORDUnet Global Virtualization Service, GEANT Testbeds Service (GTS)
  - Future: DFN, CESnet, HEAnet, US in discussions...
- Y'all come play! Help us refine VSIs!