Network Virtualization

----Group 10----

“Advances in Network-Slicing”

Introduction to Software-defined Networking

Presented by
Hari Raghavendar Rao 11334055
Seshagiri Prabhu 21410690

OpenVirteX

Ali Al-Shabibi, Marc De Leenheer, Matteo Gerola, Ayaka Koshibe,
Guru Parulkar, William Snow

Network Virtualization

- Enable multi-tenancy
- Decouple the physical network from the virtual network
- Topology virtualization
- Allow security and user traffic isolation

Related works or Existing solutions

Proprietary

- Some use overlay based approaches
- Network core only for simple forwarding
- Use SDN to use NV but take SDN away from tenants

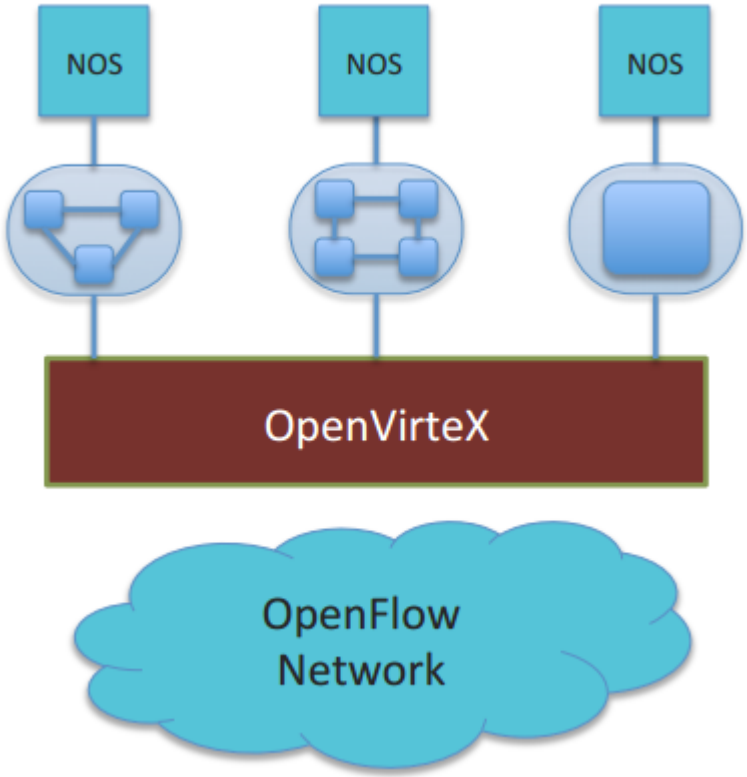
Open Source

- Flow Space Slicing
- Header space shared amongst tenants
- Configuration complexity increases exponentially with number of tenants

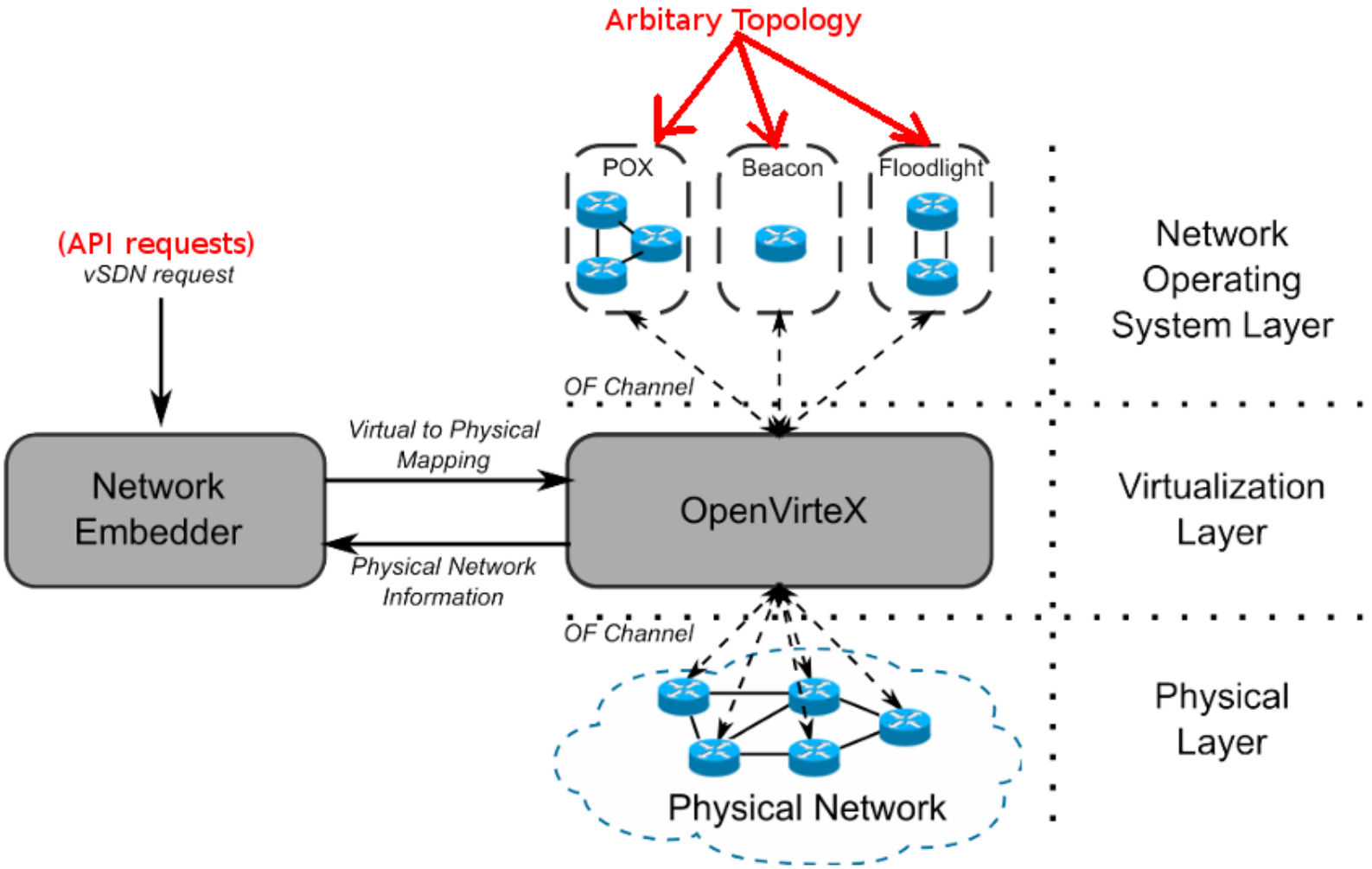
OpenVirteX

OpenVirteX enables the virtualization of OpenFlow networks

- Address Space Virtualization
- Topology Virtualization
- Programmability through OF
- Resiliency
- Portability



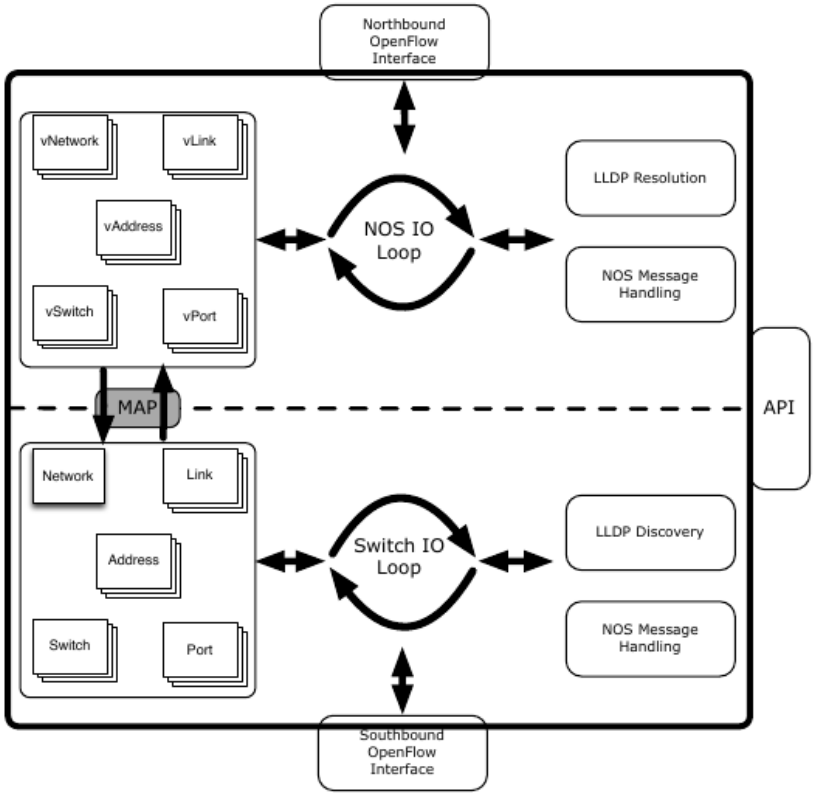
OpenVirteX - System Architecture



OpenVirteX - Internal Architecture

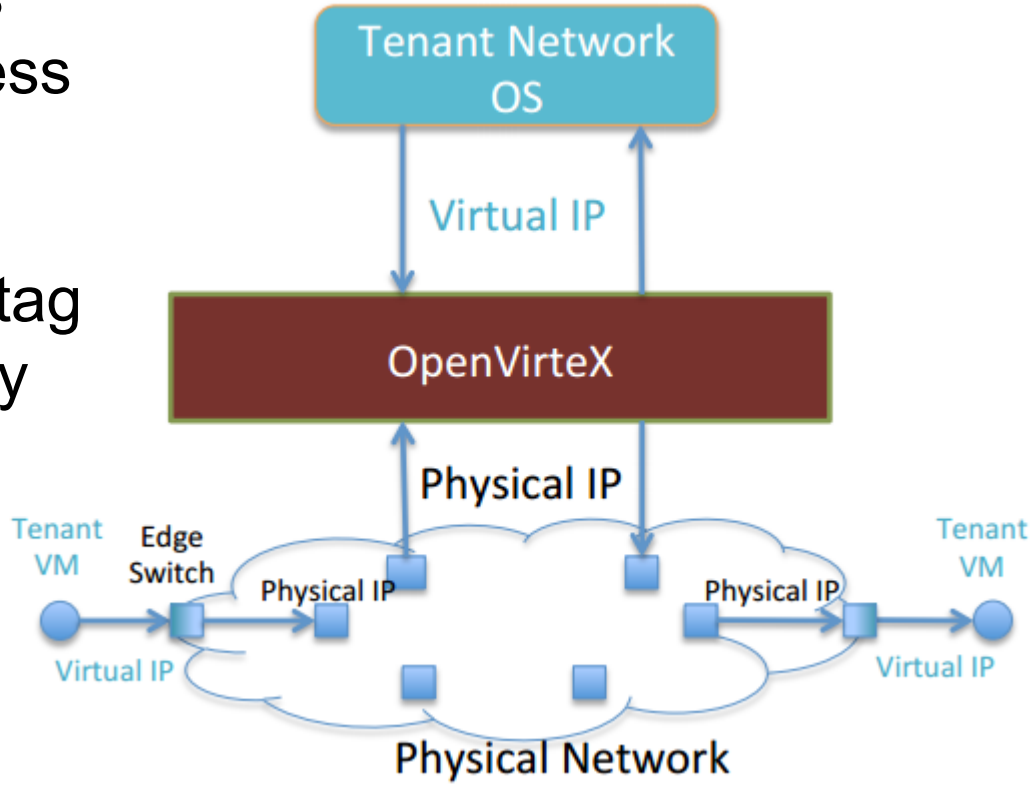
Tenant traffic isolation using MAC address fields

- Flow ID
  - Looks up the particular flow that entered the virtual link and restore its values when the traffic exits the virtual link
- Tenant ID
  - Isolates the different tenants on the virtual link
- Virtual Link ID
  - Keeps tracks of virtual links



Address Virtualization

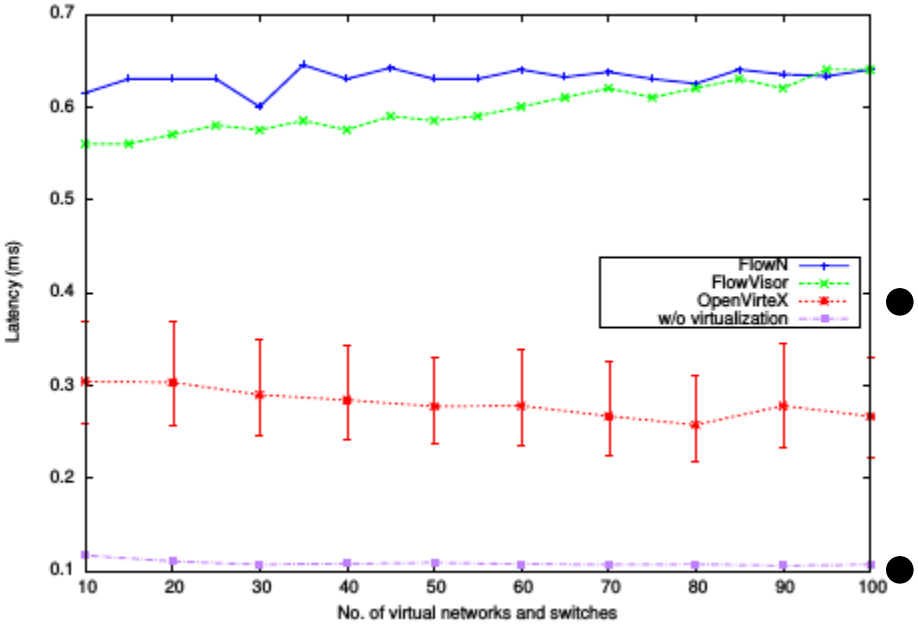
- Multiple virtual networks can use the same address space
- The rewriting inserts a tag to enable OVX to identify the packets owner
- Rewriting is completely transparent to NOS and end hosts.



Topology Virtualization

- OVX allows tenants to specify their own arbitrary topology
- OVX maps the vSDN topologies to the actual physical network. No need of isomorphism in the tenant design.
- When an LLDP message arrives at a virtual switch element with a certain outport specified, OVX “knows” where the other end of that link is.
- LIMITATION: Single physical switch cannot be partitioned

Results



Future works

- Snapshotting and migrations of vSDN:
  - Ability to preserve the state and data for fast recovery and ease of migration and duplication
- Evolving beyond OF*v1.0*:
  - Integrate OpenFlowJ-Loxi engine that generates version-agnostic OF libs for multi-languages
- vSDN based QoS:
  - In OF*v1.0*, bandwidth limit can be enforced by by statically assign a specific port queue to VN. Migration to OF*v1.3* will fix this

Conclusion

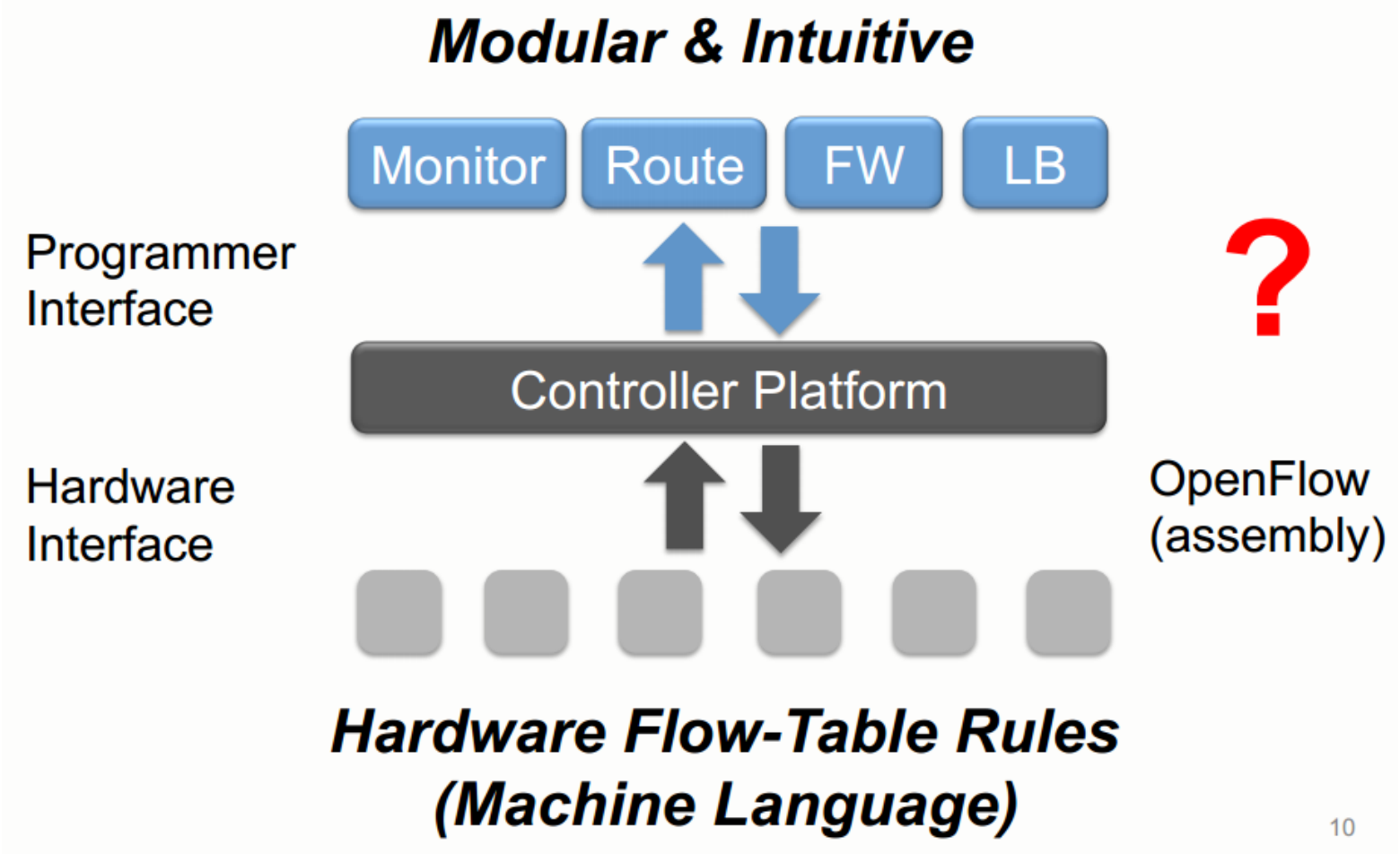
- OpenVirteX is capable of presenting tenants with configurable SDNs (topology customization)
- Full traffic address virtualization of tenant networks achieved (isolation & security)
- Modest performance when compared to the existing solutions.
- OpenVirteX Architecture enables the introduction of features and enhancements such as link resilience to tenant networks

CoVisor: A Compositional Hypervisor for Software-Defined Networks

Xin Jin, Jennifer Gossels, Jennifer Rexford, David Walker

Princeton University

Coordinating Control-plane Modules:



Covisor:

In past hypervisors used to focus on "*slicing*" the network into disjoint parts of "*separate control*" by "*separate entities*".

CoVisor allows multiple controllers to collaborate on processing the same traffic..

Compositional Network Hypervisor:

Assembly of multiple controllers:

- Single Network administrator assemble multiple controllers in flexible and configurable manner.
- Compose Data plane policies :

Parallel: Allow multiple controllers to act independently on same packets at same time.

Sequentially: Allow one controller to choose to process certain traffic before another.

Overriding: Allow one controller to choose to act or defer control to another controller.

Definition of abstract topologies:

- To protect the physical infrastructure:

"One physical switch to many logical switches"

"Allowing the administrator to provide a custom virtual topology to each controller"

Ex: "Firewall controller" the administrator may abstract the network as "big virtual switch" but the firewall doesn't need to know the underlying topology to determine if packet should be forwarded or dropped. In contrast a "routing controller" needs the exact topology to perform its task effectively.

Protection against misbehaving controllers:

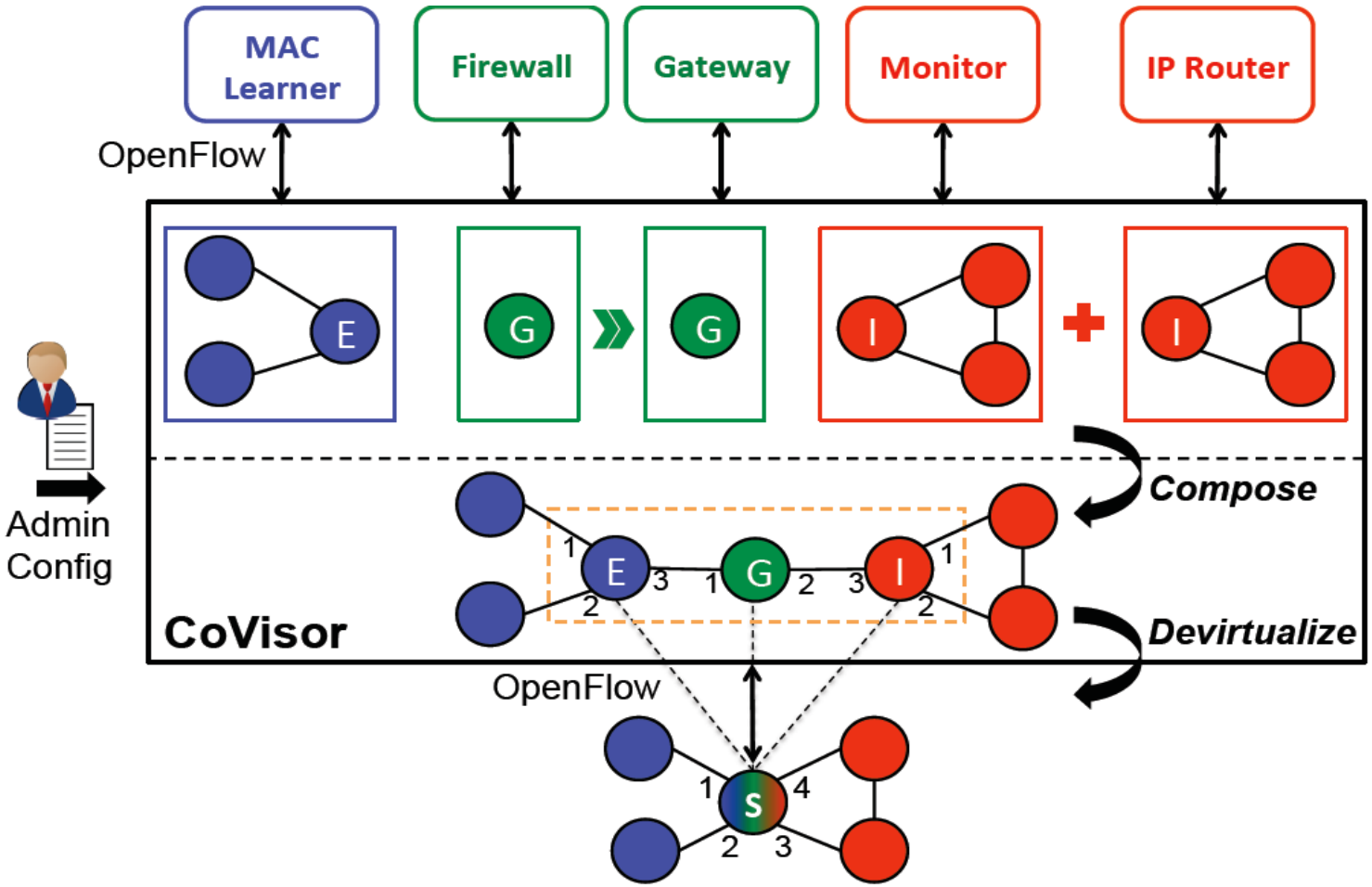
Restriction : what a controller can see of the physical topology.

Control: How a controller can process packets.

"CoVisor *Hypervisor can restrict by limiting functionality of the virtual switches exposed to other controller*"

Ex: firewall controllers should not be allowed to modify packets and MAC learner should not able to inspect IP or TCP headers.

CoVisor overview:



Functionality of CoVisor:

- Covisor uses a *novel algorithm* to *incrementally compose* applications.
- Rule priorities:calculate priorities for new rules using convenient Algebra .
- Removes the need to recompile from scratch for every rule update.
- It translates the composed policy into rules for physical topology. (one physical switch mapped to multiple virtual switches)
- After compiling the policy, covisor sends the necessary rule to update to switches.

Challenge:

- Optimise the compilation process
- Flow table compilation efficiency.

Solution:

- Incremental compilation
  - Incremental priority assignment
  - Advanced indexing for fast rule lookup

Incremental Compilation:

Monitoring M_R
$(1;\ srcip = 1.0.0.0/24;\ count)$
$(0;\ *;\ drop)$

Routing Q_R
$(1;\ dstip = 2.0.0.1;\ fwd(1))$
$(1;\ dstip = 2.0.0.2;\ fwd(2))$
(1; dstip=2.0.0.3; fwd(3))
$(0;\ *;\ drop)$

Load balancing L_R
$(3;\ srcip = 0.0.0.0/2, dstip = 3.0.0.0;\ dstip = 2.0.0.1)$
(2; srcip=0.0.0.0/1,dstip=3.0.0.0; dstip=2.0.0.3)
$(1;\ dstip = 3.0.0.0;\ dstip = 2.0.0.2)$
$(0;\ *;\ drop)$

Parallel composition: $comp_+(M_R, Q_R)$
$(2;\ srcip = 1.0.0.0/24, dstip = 2.0.0.1;\ fwd(1), count)$
$(2;\ srcip = 1.0.0.0/24, dstip = 2.0.0.2;\ fwd(2), count)$
(2; srcip=1.0.0.0/24,dstip=2.0.0.3; fwd(3),count)
$(1;\ srcip = 1.0.0.0/24;\ count)$
$(1;\ dstip = 2.0.0.1;\ fwd(1))$
$(1;\ dstip = 2.0.0.2;\ fwd(2))$
(1; dstip=2.0.0.3; fwd(3))
$(0;\ *;\ drop)$

Sequential composition: $comp_{\gg}(L_R, Q_R)$
$(25;\ srcip = 0.0.0.0/2, dstip = 3.0.0.0;\ dstip = 2.0.0.1, fwd(1))$
(17; srcip=0.0.0.0/1,dstip=3.0.0.0; dstip=2.0.0.3,fwd(3))
$(9;\ dstip = 3.0.0.0;\ dstip = 2.0.0.2, fwd(2))$
$(0;\ *;\ drop)$

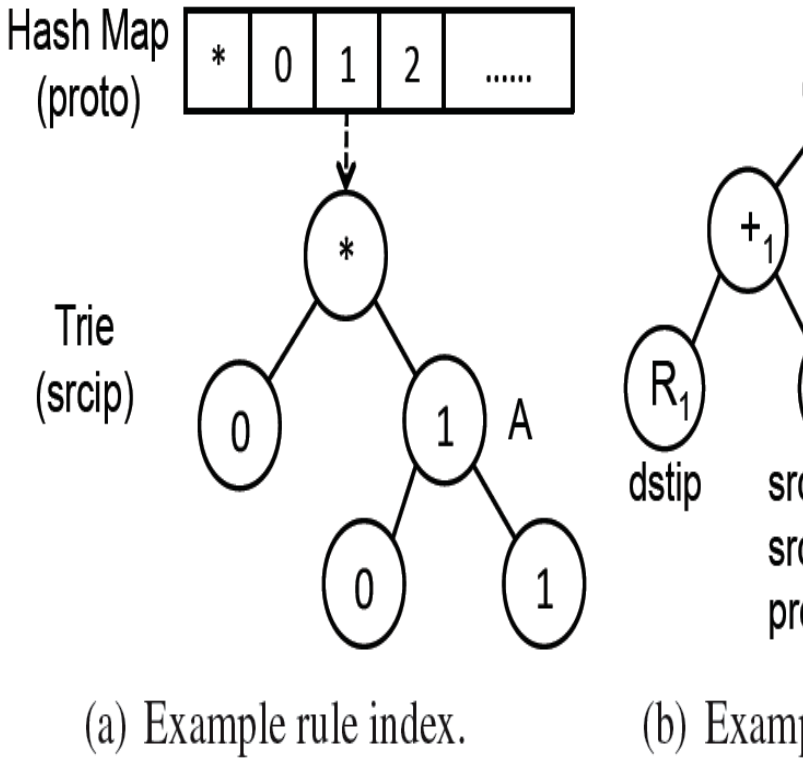
Incremental Priority Assignment:

- Heuristics
  - Parallel: $r_k.priority = r_{1i}.priority + r_{2j}.priority.$
  - Sequential: $r_k.priority = r_{1i}.priority \times MAX_{R_2} + r_{2j}.priority.$
  - Overriding: $r_k.priority = \begin{cases} r_k.mPriority \times MAX_{R_2} & \text{if } r_k \in R_1 \\ r_k.mPriority, & \text{if } r_k \in R_2 \end{cases}$

Advanced Indexing Structure:

- Given a target rule R_t, how to efficiently find the rules in a flow table that overlap with R_t?

Hashtable : exact match
Trie : prefix-match
list of arbitrary: wildcard match



(a) Example rule index. (b) Example syntax tree.

Implementation and Evaluation:

- Implement with 4k lines of java code on OpenVirteX.
- Simulation experiments against a strawman compiler.

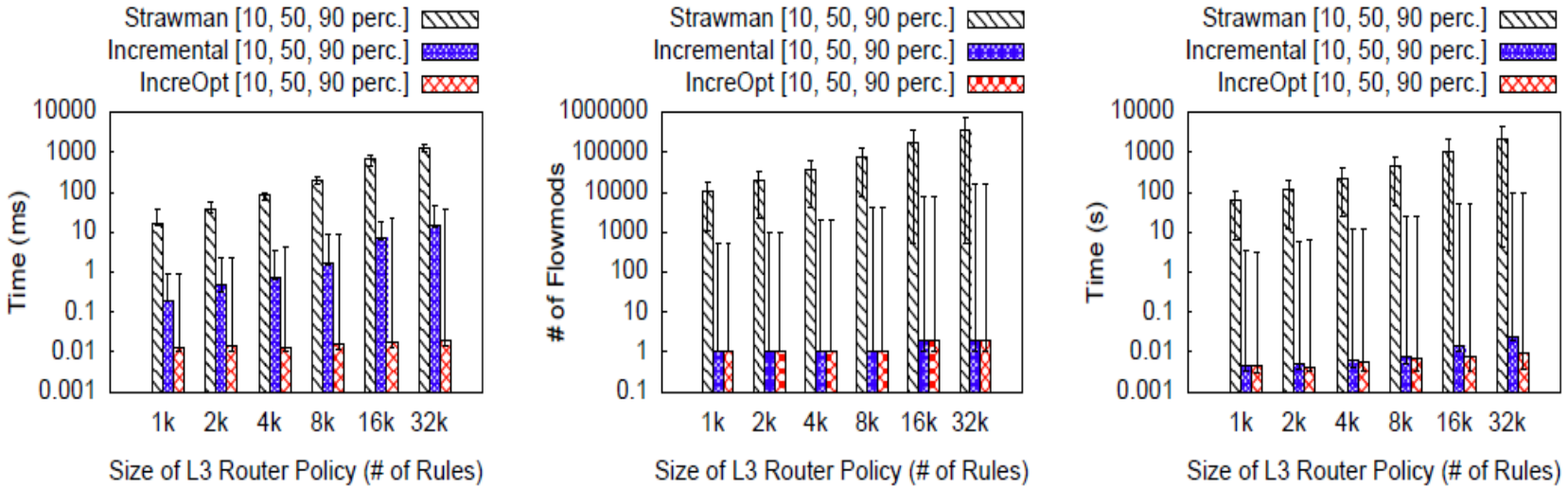


Figure 10: Per-rule update overhead of L3-L4 Firewall ≫ L3 Router (log-log scale).

Conclusion:

- Administrator can combine multiple controllers to collaboratively process a network's traffic.
- Covisor uses a combination of *novel algorithms* and data structures to efficiently compile policies in an incremental manner.
- Evaluations on covisor prototype shows it faster than a naive implementation.

References:

Ali Al-Shabibi, M.De Leenheer, M. Gerola,A.Koshibe,G.Parulkar, E. Salvadori, and B. Snow , “OpenVirteX: Make Your Virtual SDNs Programmable”, *ACM SIGCOMM HotSDN* workshop,August 2014.

Xin Jin ,Jennifer Gossels,Jennifer Rexford,David Walker *Princeton University* “CoVisor: A Compositional Hypervisor for Software-Defined Networks”, *USENIX NSDI May 4-6 2015.*

Thank You

Any Questions?