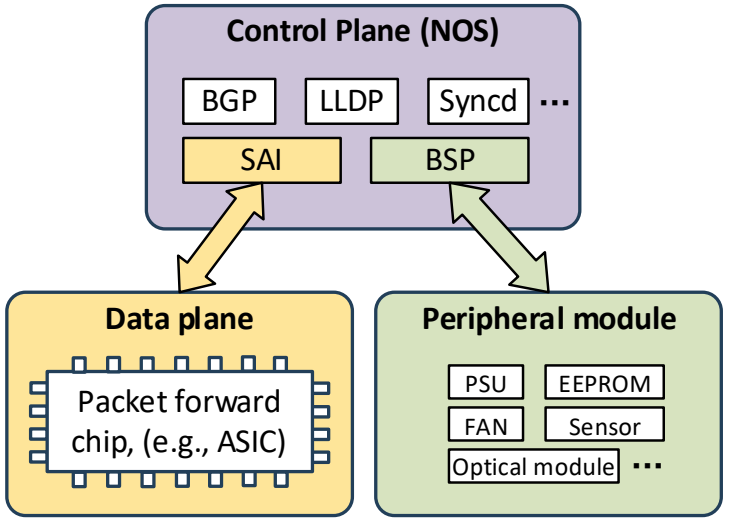
A Composable Emulation Framework for Whitebox Switches

Congcong Miao*, Xianneng Zou, Chuwen Zhang, Shiping Yang, Qihang Liu, Zhijie Yan, Yanke Zhang, Yong Jiang, Qiao Xiang, Xin Jin, Zili Meng, Ang Chen

Trend of Whitebox Switches

- Cloud providers deploys more whitebox switches:
  - To avoid vendor lock in
  - To reduce costs
  - To customize new features for application demands

Blackbox switch

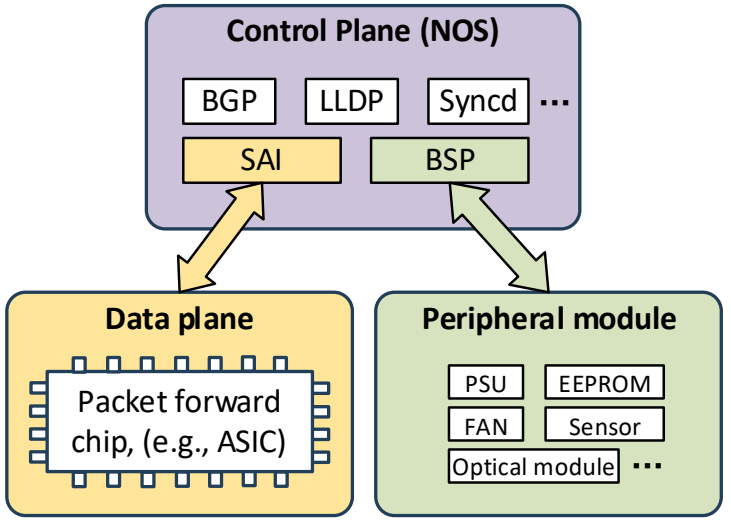


Whitebox Switch architecture

Trend of Whitebox Switches

- Cloud providers deploys more whitebox switches
- However, high-fidelity emulation becomes non-trivial
  - For blackbox switch, use vendor-provided firmware in containers or VMs (e.g., *CrystalNet*[1] and *Crescent*[2])
  - For whitebox switches, use?

Blackbox switch

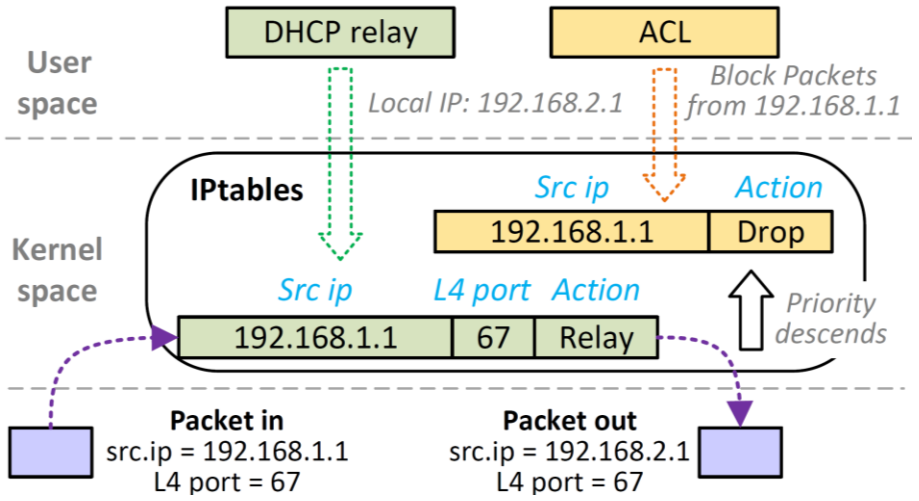


Whitebox Switch architecture

[1] H. H. Liu et al., "Crystalnet:Faithfully emulating large production networks", NSDI'17.
[2] G. Gao et al., "Crescent: Emulating heterogeneous production network at scale", NSDI'24

Why cannot we use existing emulators

1. High-speed and faithful data plane
   - Inconsistency, e.g., IP IPtables rule conflicts (e.g., SONIC-vs), ECMP hashing inconsistency (e.g., SONIC-vs, SONiC-bmv2, Off-the-shelf image)
   - Low forwarding capability (e.g., SONiC-simu)



DHCP-relay issues an IPtable entry with the same source IP address as an ACL entry but with a higher priority

	Control plane	Data plane	Peripheral modules
Whitebox switch	SONiC	ASIC	Yes
Off-the-shelf image	Private	Private	Unknown
SONiC-vs	SONiC	Linux Kernel	No
SONiC-simu	SONiC	ASIC simulator	No
SONiC-bmv2	SONiC	P4	No
MirSwitch(Ours)	**SONiC**	**VPP-extend**	**BSP**

Comparison of existing software switches

Why cannot we use existing emulators

1. High-speed and faithful data plane
2. Customized control plane as that in production
   - Private control plane (e.g., off-the-shelf image)

	Control plane	Data plane	Peripheral modules
Whitebox switch	SONiC	ASIC	Yes
Off-the-shelf image	Private	Private	Unknown
SONiC-vs	SONiC	Linux Kernel	No
SONiC-simu	SONiC	ASIC simulator	No
SONiC-bmv2	SONiC	P4	No
MirSwitch(Ours)	SONiC	VPP-extend	BSP

Comparison of existing software switches

Why cannot we use existing emulators

1. High-speed and faithful data plane
2. Customized control plane as that in production
3. Supporting heterogeneous peripheral modules (None except MirSwitch)
   - Port splitting
   - MAC address
   - Hash algorithm in ECMP

	Control plane	Data plane	Peripheral modules
Whitebox switch	SONiC	ASIC	Yes
Off-the-shelf image	Private	Private	Unknown
SONiC-vs	SONiC	Linux Kernel	No
SONiC-simu	SONiC	ASIC simulator	No
SONiC-bmv2	SONiC	P4	No
MirSwitch(Ours)	SONiC	VPP-extend	BSP

Comparison of existing software switches

Build a composable software switch

SONiC With customized features

Control plane

NFV, e.g., FD.io's VPP

Data plane

environment to mock up

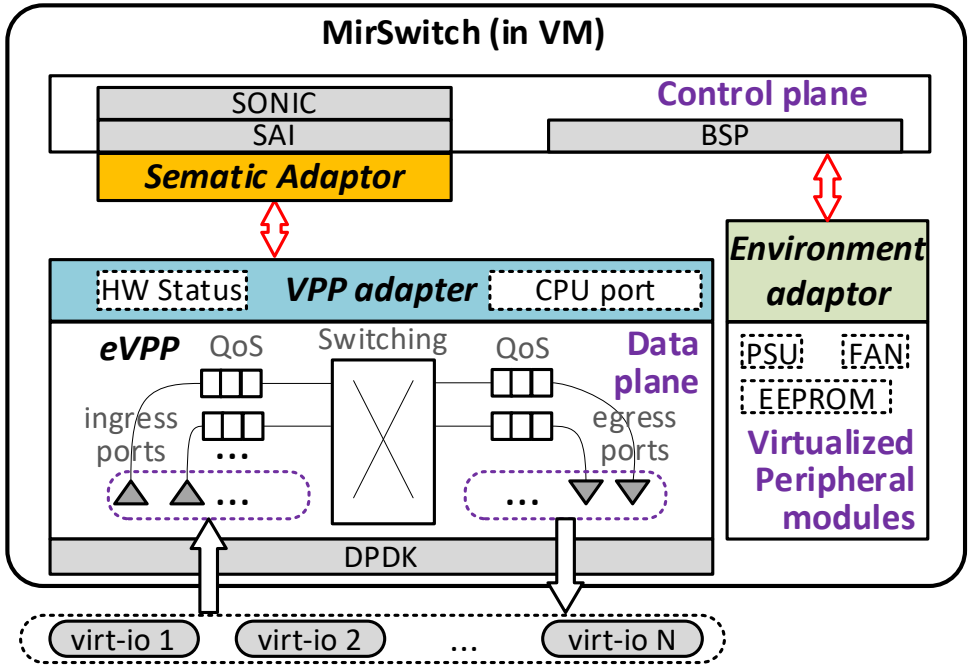
peripheral modules

- **Challenges:**
  - Closed and unfaithful VPP
  - Semantic gaps between SAI and VPP interface.
  - Lack of support for BSP

System Design

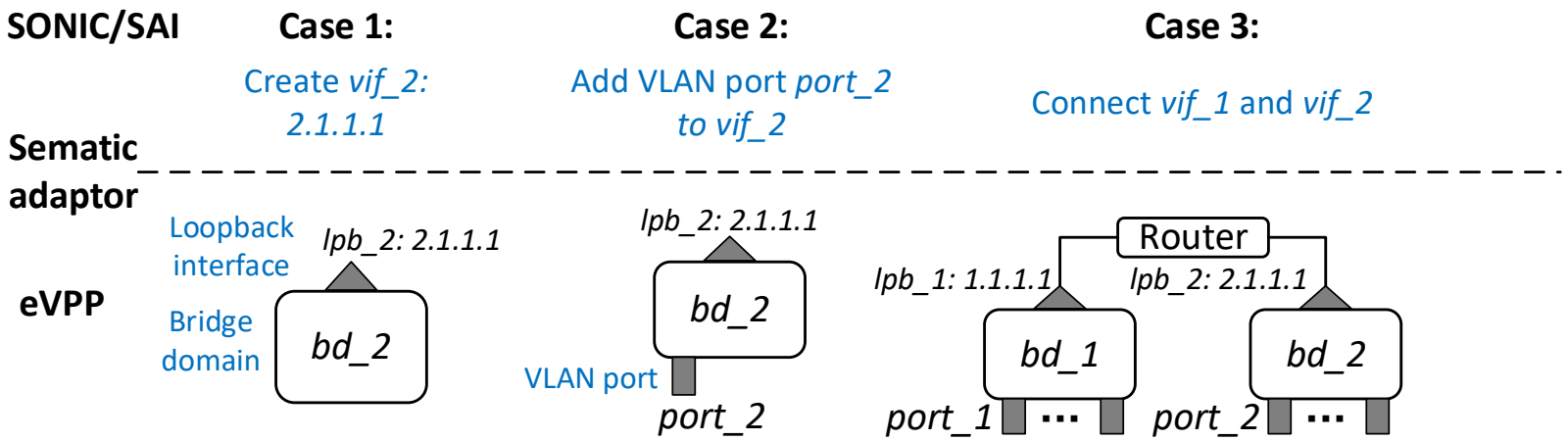
High-level Design

- MirSwitch is a novel composable emulation framework
  - Control plane: production SONiC
  - Data plane: *eVPP*
  - Peripheral modules: virtualized env
  - Adaptor-centric approach:
    - *VPP adaptor*
    - *Semantic adaptor*
    - *Environment adaptor*



eVPP

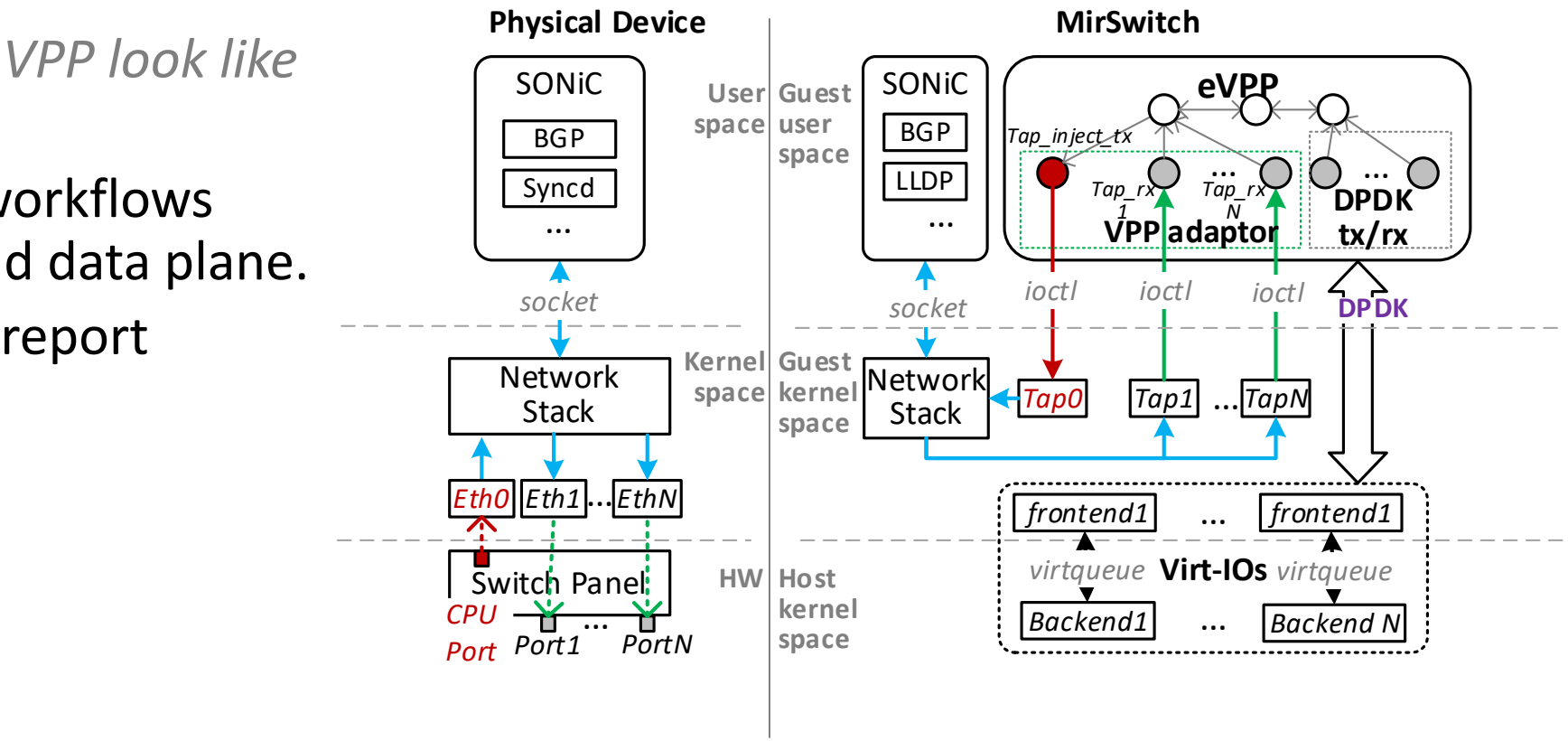
- eVPP, *faithfully realizing data plane functions, including:*
  - ECMP
  - VLAN Interface
  - …



Three typical cases of configuring VLANIF

VPP Adaptor

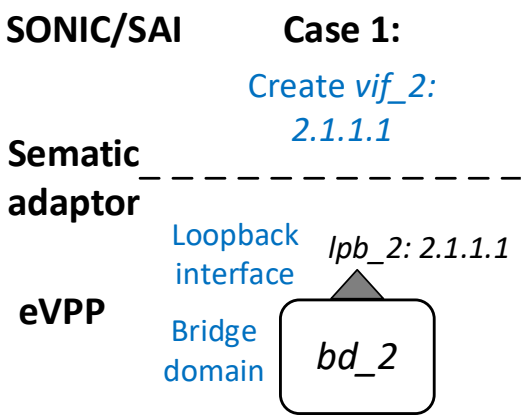
- VPP adaptor, *making VPP look like a real data plane*
  - TAP-based packet workflows between control and data plane.
  - Event-based status report



The packet forwarding workflows in a typical whitebox switch (left) and MirSwitch (right).

Semantic Adaptor

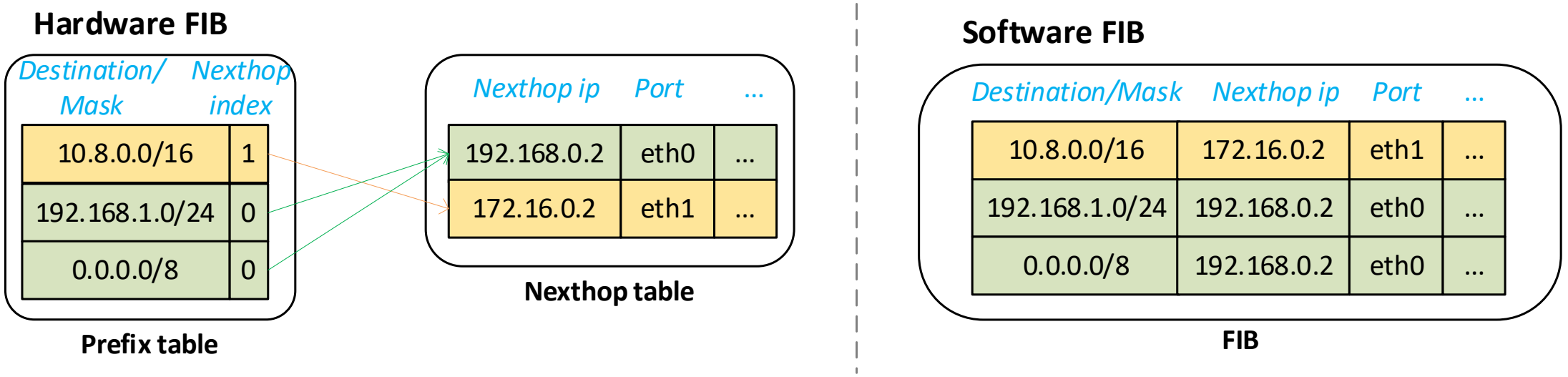
- Semantic Adaptor, *enables eVPP to be controlled just like a real data plane.*
- Two basic types to translate requests between SAI and eVPP:
  - one-to many mapping (e.g., VLANIF)



The one-to-many mapping in configuring VLANIF.

Semantic Adaptor

- Semantic Adaptor, *enables eVPP to be controlled just like a real data plane.*
- Two basic types to translate requests between SAI and eVPP:
  - one-to many mapping (e.g., VLANIF)
  - many-to-one mapping (e.g., FIB)



The different structure of FIB in Switching ASIC and eVPP.

Environment Adaptor

- Environment Adaptor, *supports peripheral modules on the software switch*
  - Shielding software data plane, by a fake EEPROM file with the platform information to cheat ONIE
  - Adapting to heterogeneous hardware platforms, by loading the specific BSP drivers from the complete set.

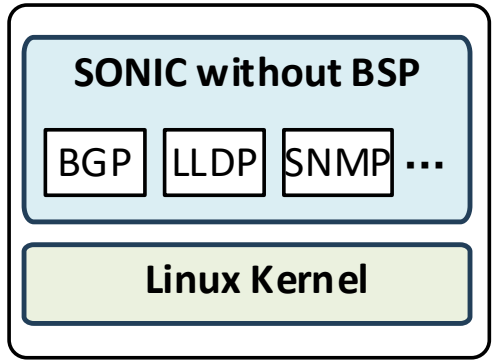


Image without BSP

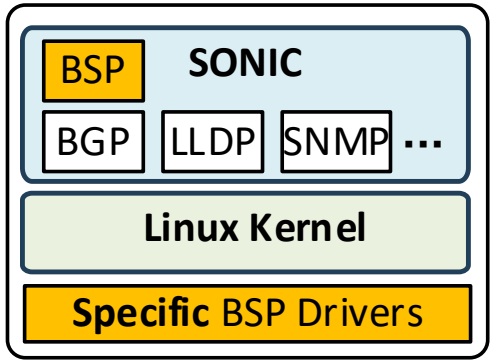


Image with BSP

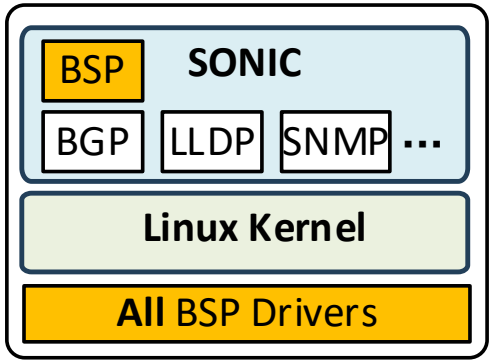
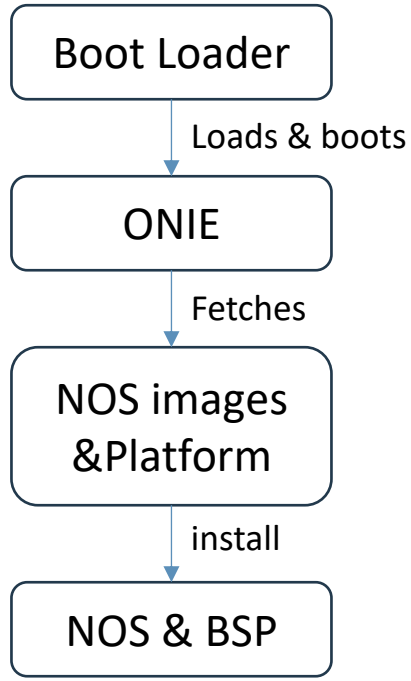


Image with all BSP



Booting workflow

Evaluation

Testbed evaluation: Setup

- 16 server nodes with
  - AMD EPYC 7K62 CPU (48 cores, 2.6/3.3GHz)
  - 512GB DDR4 memory
- Linking VM-based MirSwitch by
  - Linux bridge inside a server node
  - VXLAN between server nodes
- Using GNS3 to construct the following network topology
  - DCN-S, a small pod with 8 leaf switches, 96 TOR switches, and total 96*24 servers
  - DCN-M, comprising two pods by 8 spine switches
  - DCN-L, comprising four pods by 16 spine switches

Testbed: switch-level fidelity

Category	Functions	MirSwitch	SONiC -vs	SONiC - bmv2
Ctrl plane	BGP	✓	✓	✓
	Routing stack	✓	✓	✓
	Warm reboot	✓	X	X
	IPv6 linklocal	✓	✓	X
Mgmt plane	SNMP	✓	✓	✓
	Platform-specific commands	✓	X	X
	CMIS	✓	X	X
	SFP utilities	✓	X	X
Data plane	Drop counter	✓	X	X
	Faithful ECMP	✓	X	X
	IPv4/IPv6	✓	✓	✓

Comparison between traditional solutions and MirSwitch on **some critical functionalities** in the SONiC roadmap planning

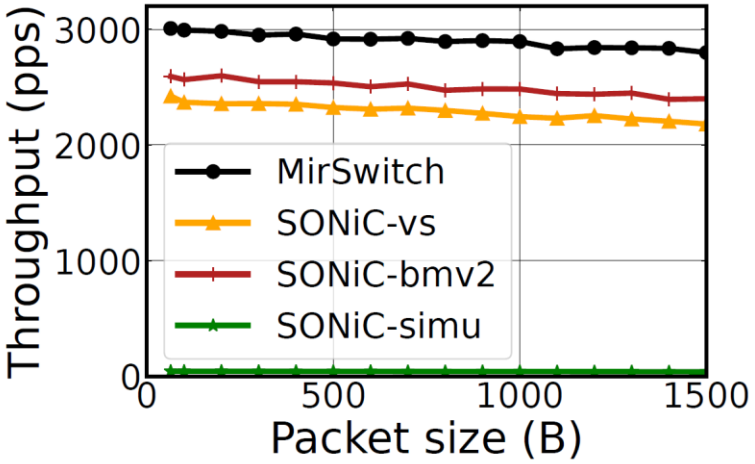
- MirSwitch supports all 26 functions
- SONiC-vs and SONiC-bmv2 support fewer, e.g., *warm reboot*

- MirSwitch supports all 15 functions
- SONiC-vs and SONiC-bmv2 support 11/15 without BSP, e.g., *platform-specific cmds*

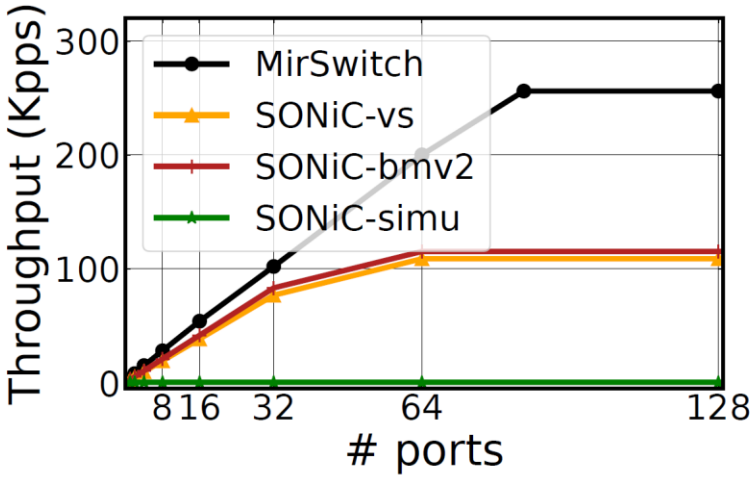
- MirSwitch supports the highest 12/16 functions, including *ECMP*.
- SONiC-vs (6/16) and SONiC-bmv2 (5/16)

Testbed: Network throughput

- Single-port throughput
  - MirSwitch performs the best, reaching up to 3000pps
- Switch-level throughput
  - MirSwitch supports a network throughput of 256Kpps with 128 ports



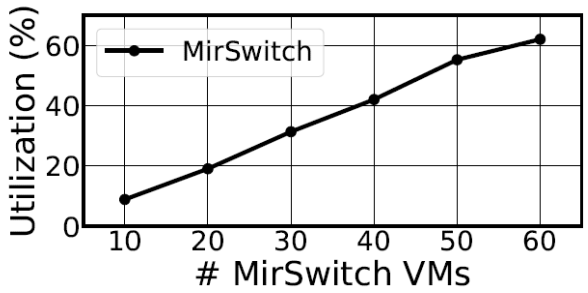
Single-port throughput



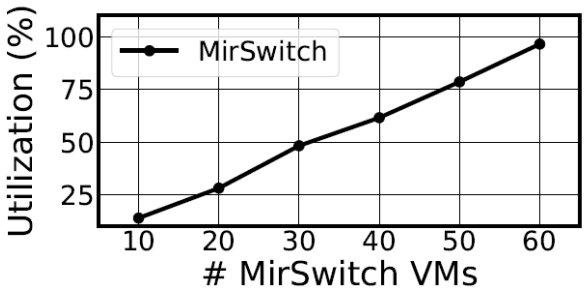
Switch-level throughput

Testbed: Scalability

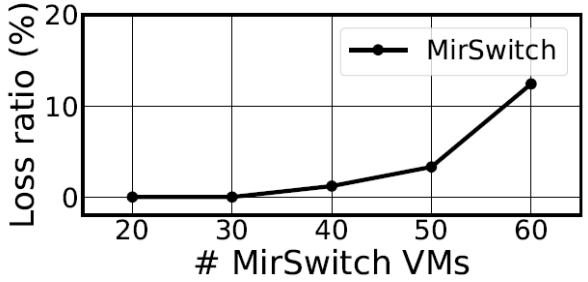
- Resource consumption
  - CPU and memory utilization increase linearly
  - No packet loss occurs for up to 30 VMs per server



(a) CPU utilization



(b) Memory utilization

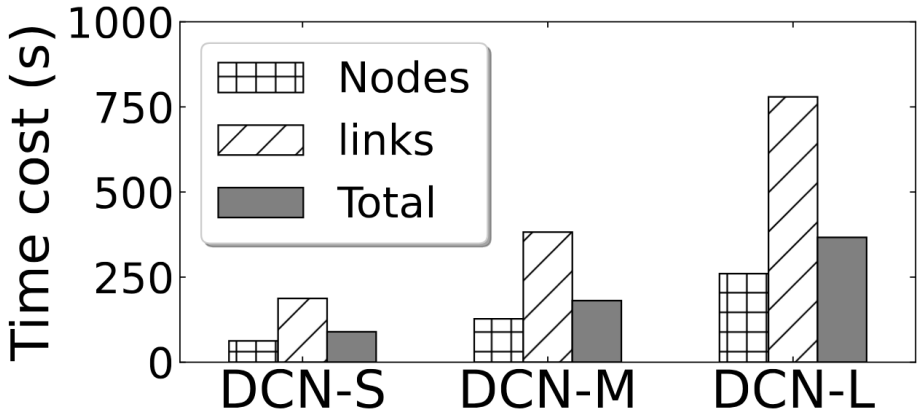


(c) Packet loss rate

CPU utilization, memory footprint and packet loss ratio as MirSwitch VMs increases

Testbed: Scalability

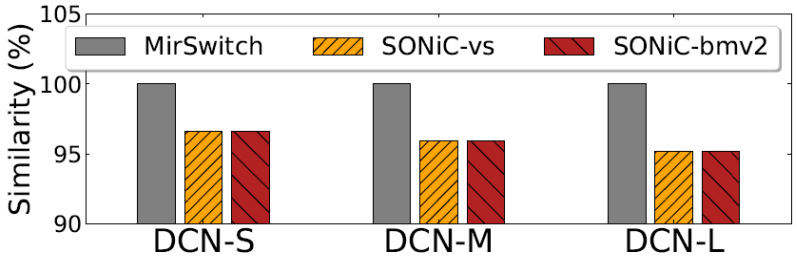
- Construction time
  - 1.5 minutes for a production network with ~100 switches



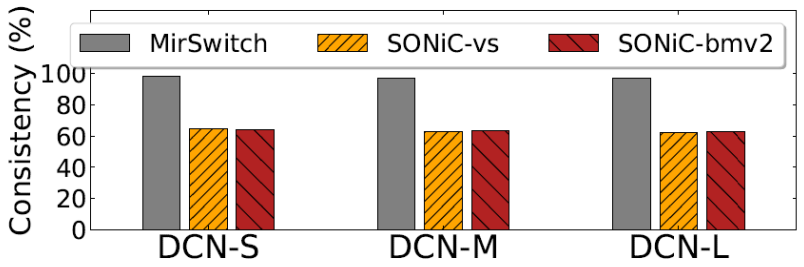
The construction time in different networks.

Testbed: network-level fidelity

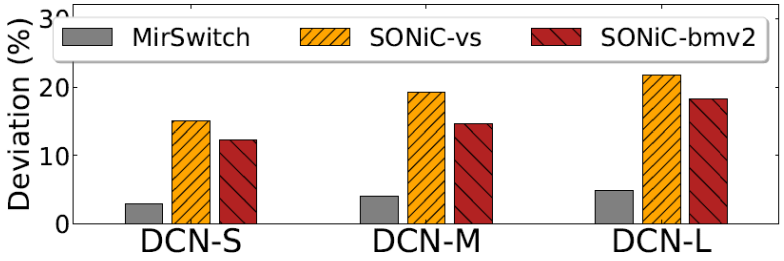
- Network topology similarity
  - achieving 100% identical topology due to supporting port splitting
- Packet forwarding path
  - achieving 100% identical flow path due to the same topology and ECMP
- Network performance
  - estimating bandwidth utilization with an average error of 5%



GED-based topology similarity



The consistency of packet forwarding paths



Standard deviations of the observed link's utilization

Production results: Detected network failures

- **Control plane:** helps detect nearly all SONiC bugs, all network config bugs and all BSP driver bugs.
- **Data plane:** helps optimize the production network, e.g., avoid hash collisions
- **Peripheral module：** detects 100% of the stand-alone config

Field	Issues	Examples	MirSwitch	Trad	Total
Control plane	SONiC bug	bugs in SONiC components	56	NA	57
	Network config bug	wrong configs for BGP and ACL	74	62	74
	BSP driver bug	bugs when light power fluctuates	2	NA	2
Data plane	Network optimize	load imbalance, and hash collision	23	10	27
Peripheral module	Stand-alone config bug	wrong host name and port down	43	36	43

Comparison of the traditional software switch and MirSwitch to detect network issues in the last year.
NA means Not Applicable, and Trad means the traditional approach

Lessons learned

- Tradeoff between fidelity and performance
  - High fidelity rather than full fidelity
  - SONiC-simu simulates the packet forwarding details in the switching ASIC, but encounters extremely low speed
- Leveraging open-sourced frameworks with abundant functions
  - We first relied on SONiC-bmv2, but it takes huge labor cost.
- Functions vs usability
  - “Better to sharpen your tools before using them”
  - Modify settings to lessen resource usage, if needed.

Conclusion

- Emulating customized whitebox switch with high fidelity and performance is key for reliable operation.
- We propose a *composable emulation framework*, i.e., MirSwitch, to supports high customizability.
- MirSwitch uses an adaptor-centric approach to reconcile interface differences across emulator modules.
- MirSwitch has been deployed in production network.