# Develop and test your P4 features with high-fidelity production environments using SONiC+CrystalNet

Hongqiang “Harry” Liu, Yibo Zhu, Jitu Padhye, Guohan Lu, Lihua Yuan

Microsoft

05/17/2017

# New network features keep emerging with P4

**Telemetry & Monitoring**
- FlowRadar (NSDI'16)
- Narayana et al (SIGCOMM'17)
- …

**Enhance Existing Protocols**
- NOPaxos (OSDI'16)
- Sharma et al (NSDI'17)
- …

**Middlebox**
- Dapper (SOSR'17)
- Miao et al (SIGCOMM'17)
- …

**In-network distributed systems**
- ZooKeeper
- In-network caching
- …

# How do these P4-features work in the wild?

**Network Researchers & Designers**

- To validate ideas and designs
- Lack of realistic test environment

**Network Engineers & Operators**

- To ensure the safety of production
- Lack of freedom to perform trials

# Challenging your confidence on your feature

How can you guarantee NO incident?

Albert Greenberg, CVP Azure Networking


You're
Fired!

# The potential approaches to build confidence

**Debugging**


**Verification**

**Testbed**

But, we are still not sure how it works in productions

Q1: Can you use a realistic control-plane?

Q2: Can you use realistic network topologies?

Q3: Can you be compatible with legacy devices?

# Outline

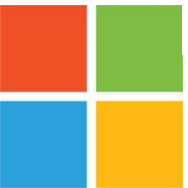
- Background & Motivation
- **SONiC + P4: a production quality control plane for P4**
- CrystalNet: a cloud-scale network emulator
- Experiences with SONiC + P4 + CrystalNet
- Conclusion

# Control-plane stack is the base of P4

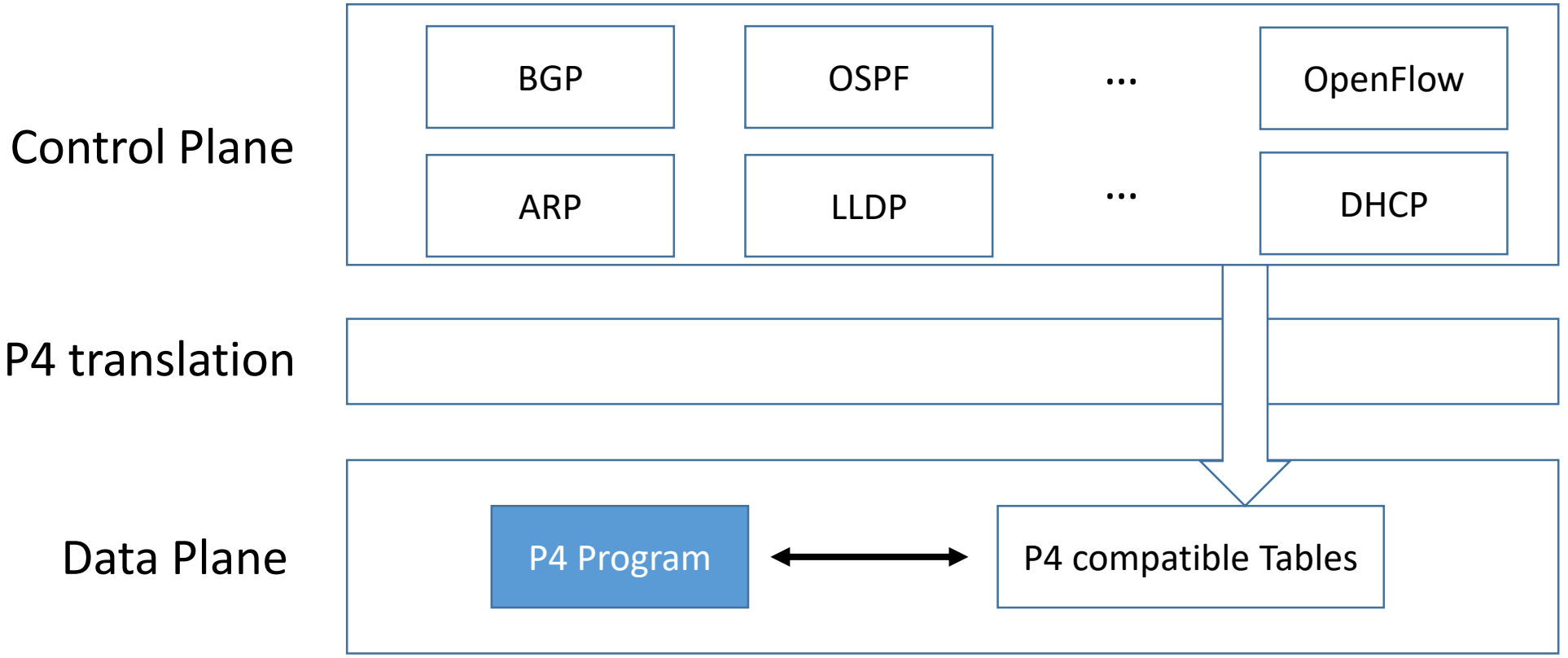
```
99      table ipv4_lpm {
100         key = {
101             hdr.ipv4.dstAddr: lpm;
102         }
103         actions = {
104             ipv4_forward;
105             drop;
106             NoAction;
107         }
108         size = 1024;
109         default_action = NoAction();
110     }
```

**The concrete table from control-plane**

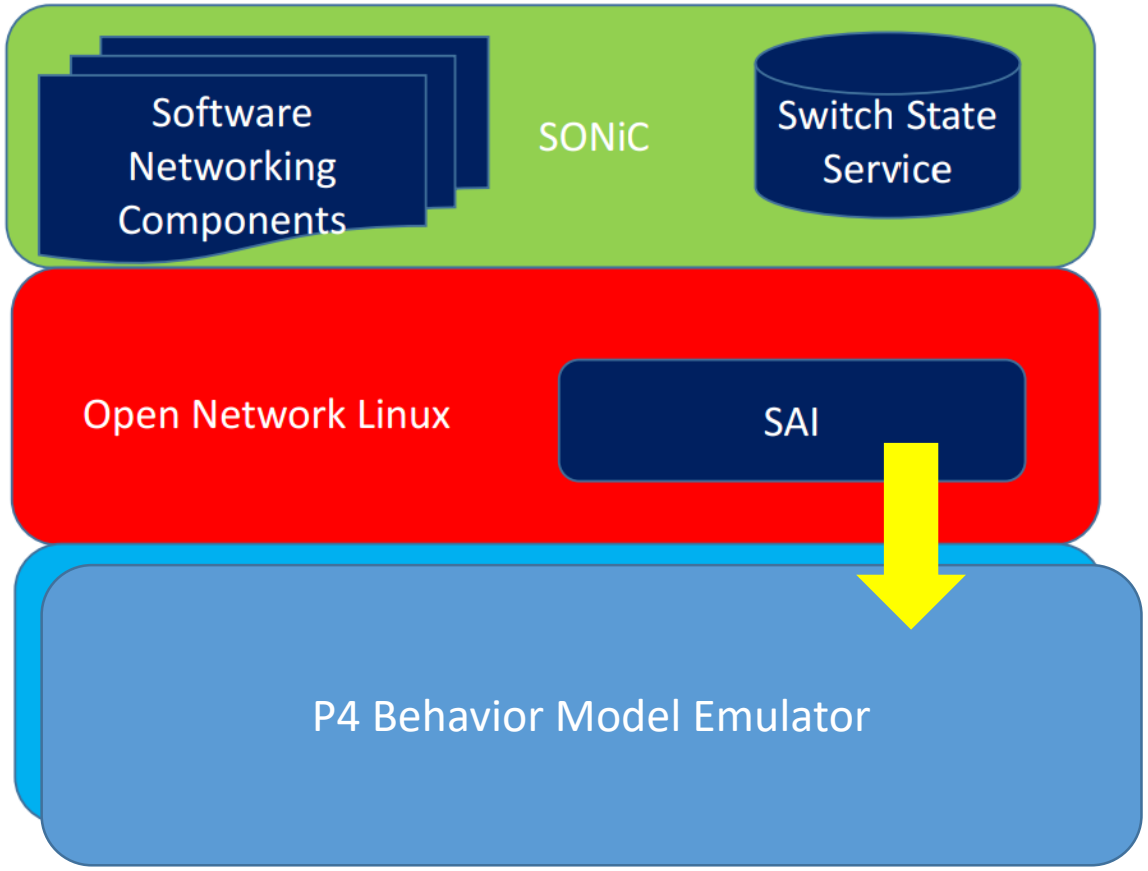
10.0.1.0/24 ipv4_lpm ipv4_forward
10.0.2.0/24 ipv4_lpm ipv4_forward
10.0.3.0/24 ipv4_lpm ipv4_forward
0.0.0.0/0 ipv4_lpm drop



# A realistic environment to run P4 features

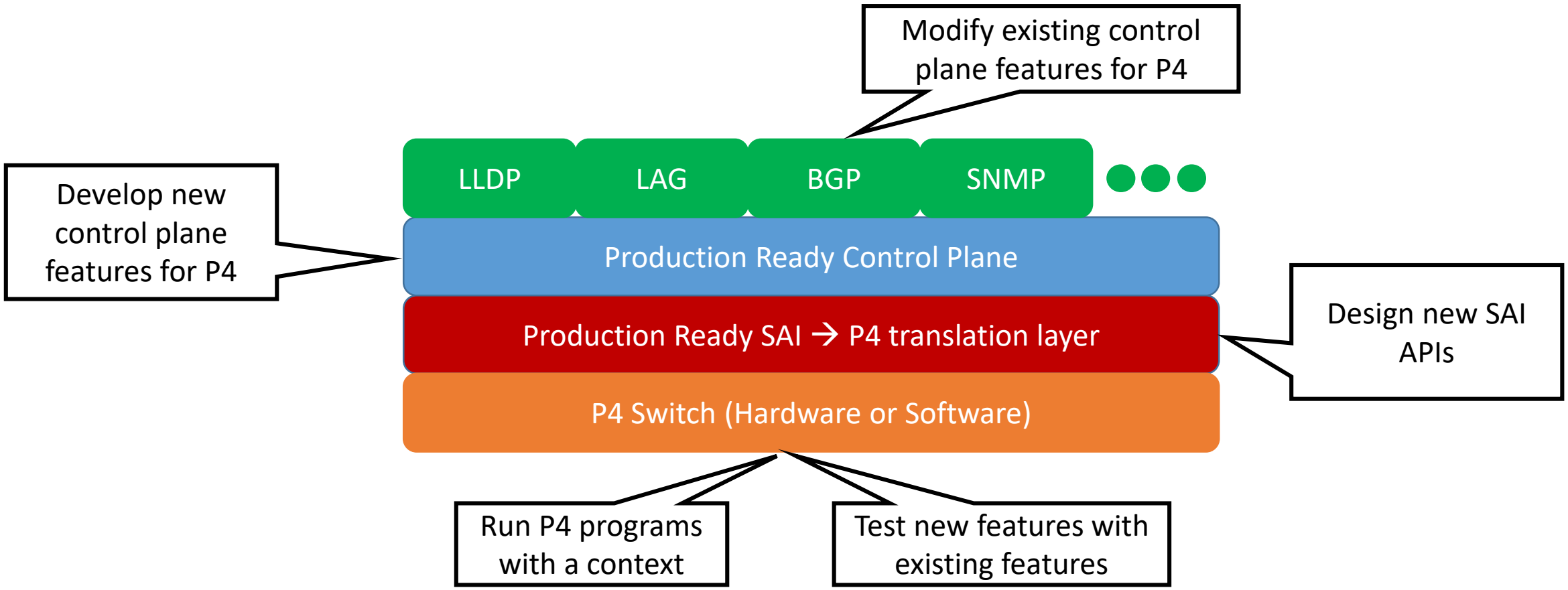


# SONiC: a production ready SwitchOS with P4




- **SONiC is being used in production**
  - Deployed Features: BGP, ECMP, QOS-ECN, PFC, WRED, COS, SNMP, SysLog, LLDP, NTP, LAG, COPP, QOS-RDMA, DHCP Relay Agent …
- **SONiC + P4 has been fully tested**
  - Control plane to SAI is implemented by Azure
  - SAI to P4 BM is implemented by Barefoot
- **SONiC + P4 has Docker container**
  - 20+ instances on one Azure VM (4 cores, 8GB)

# SONiC+P4 offers a realistic software stack

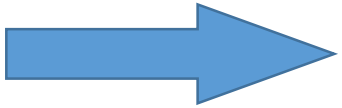


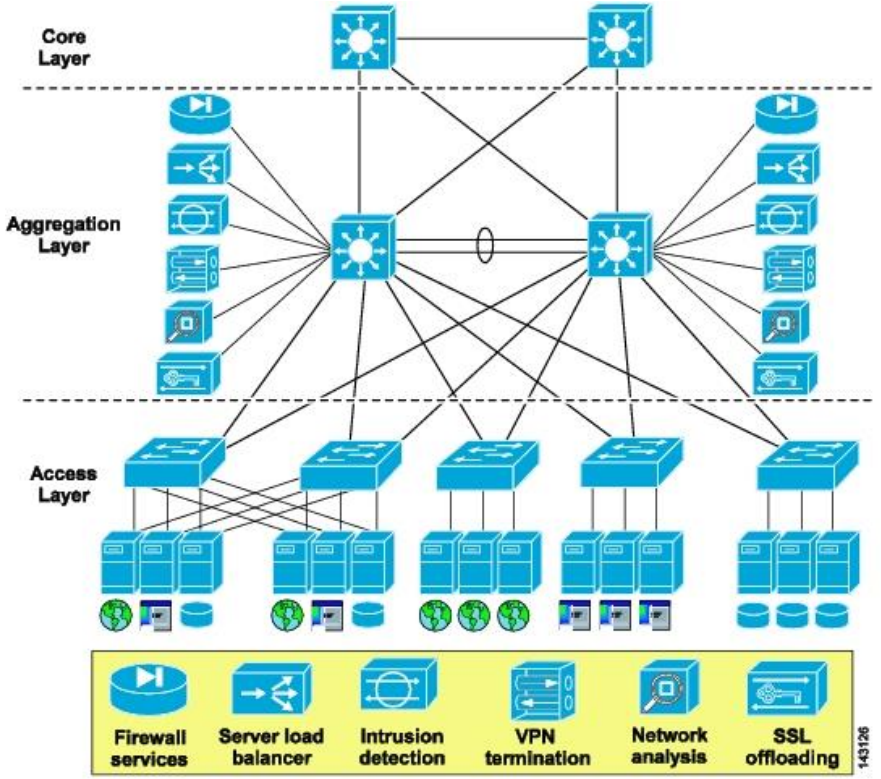
Q1: Can you use a realistic control-plane?

Q2:Can you use realistic network topologies?

Q3: Can you be compatible with legacy devices?

SONIC+P4






**Real control-plane:**

- Real device firmware
- Real topology
- Real configuration

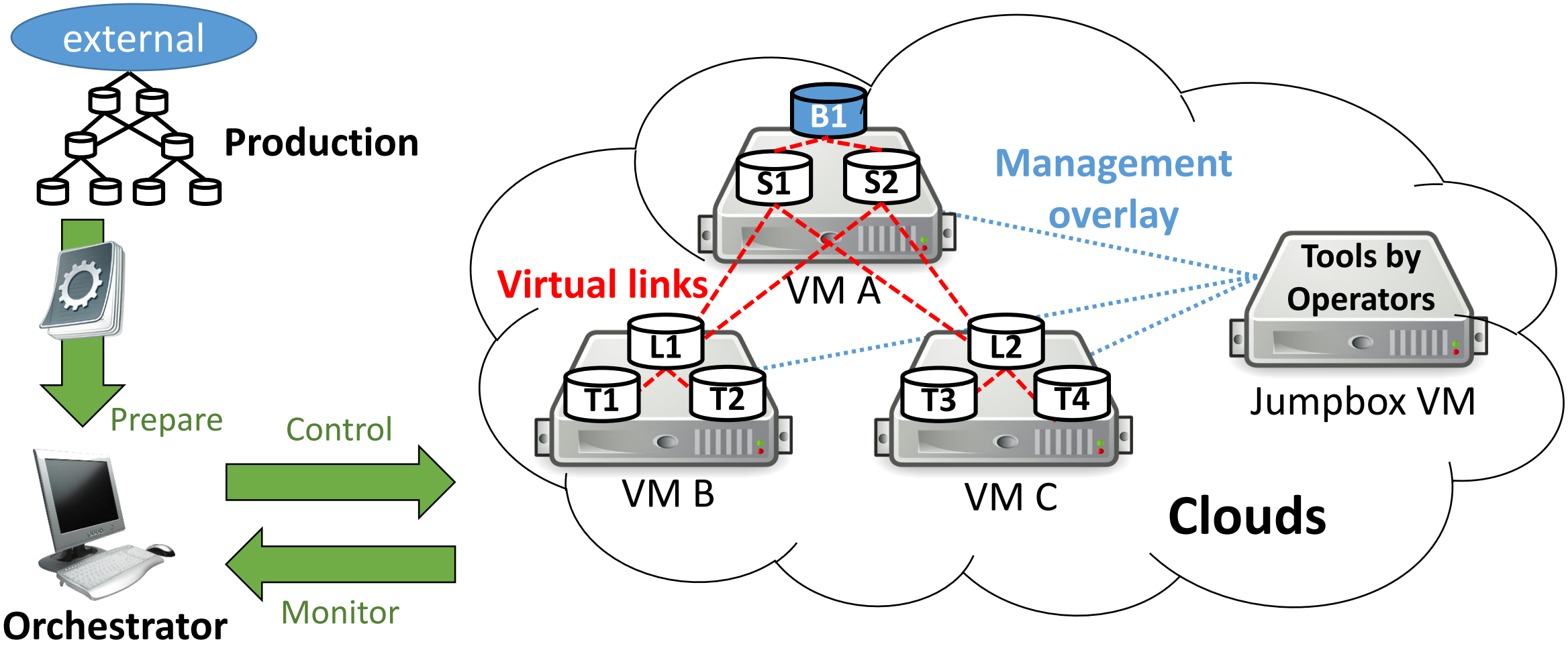
**Virtualized (mostly) data-plane**

- ASIC behavior models (e.g. BMv2)
- Virtual interfaces and virtual links

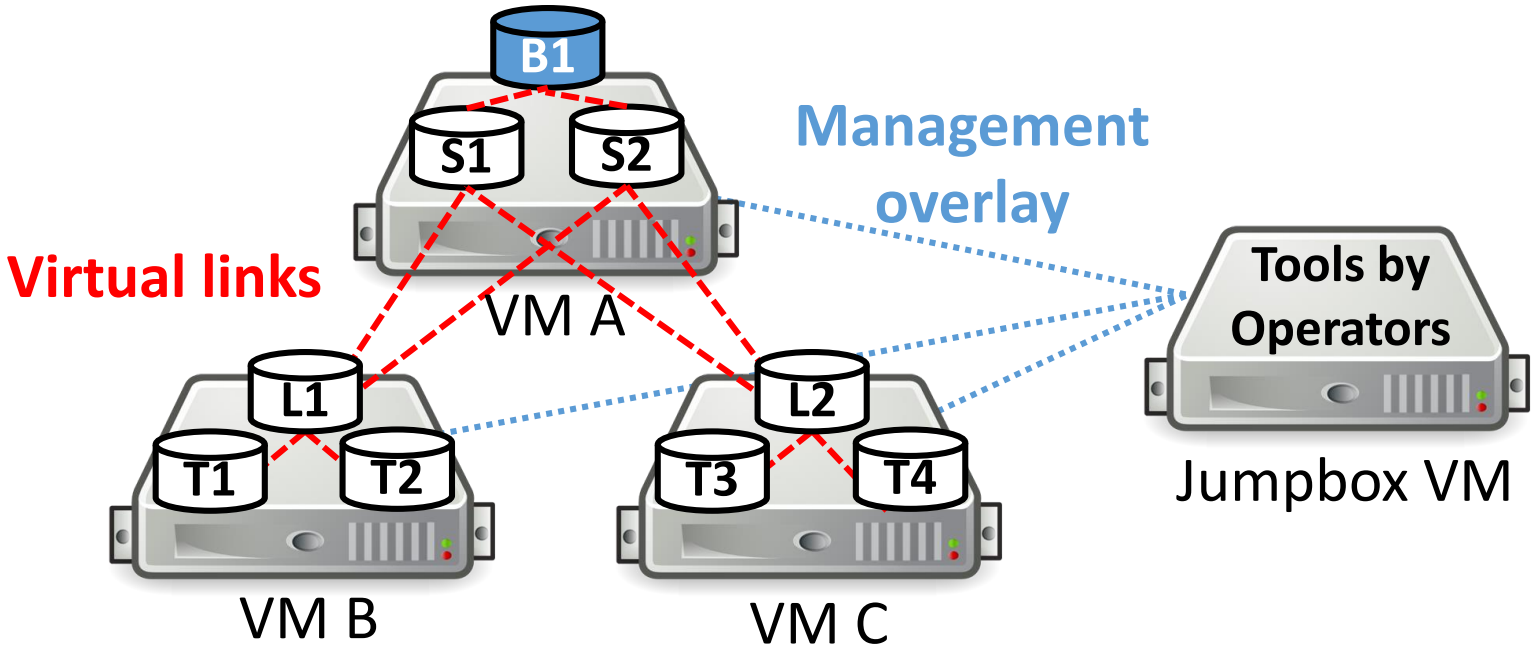
# Outline

- Background & Motivation
- SONiC + P4: a production quality control plane for P4
- **CrystalNet: a cloud-scale network emulator**
- Experiences with SONiC + P4 + CrystalNet
- Conclusion

# CrystalNet: a cloud-scale network emulator



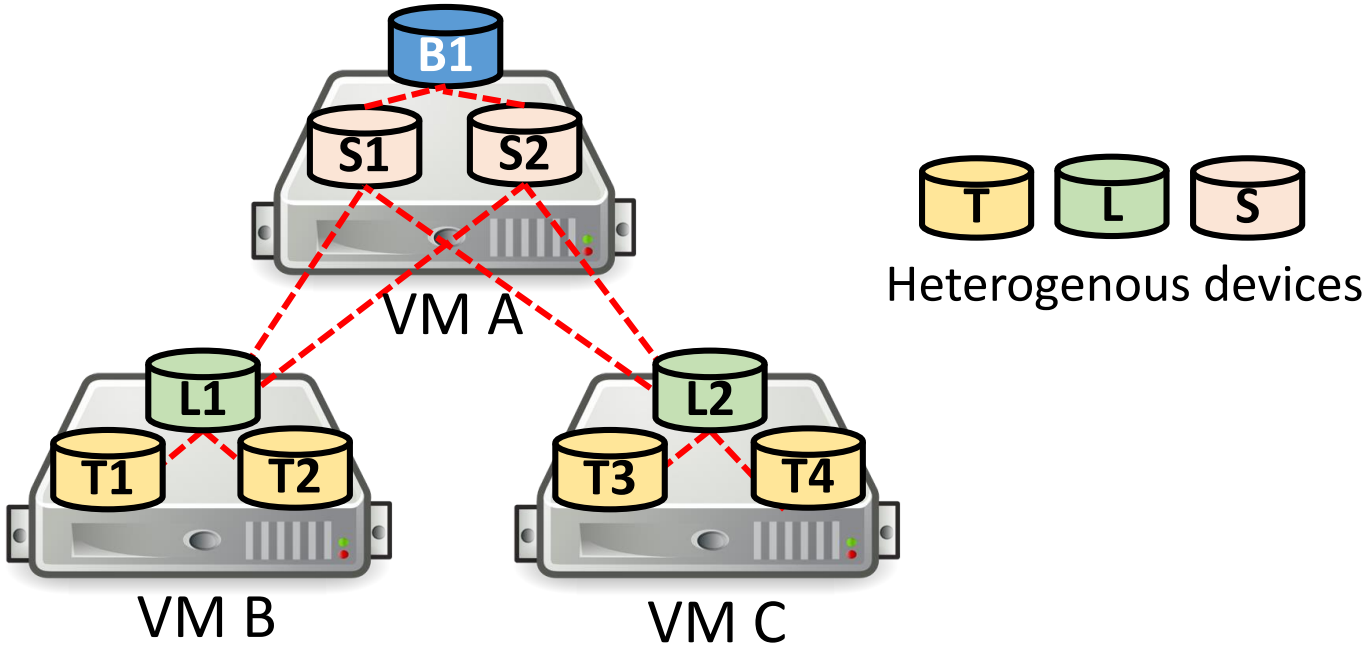
# Feature of CrystalNet: On-demand scalability




**Key idea: decouple design and deployment**

- Decoupling overlay networks with under-layer VM clusters
- Decoupling the states in different VMs

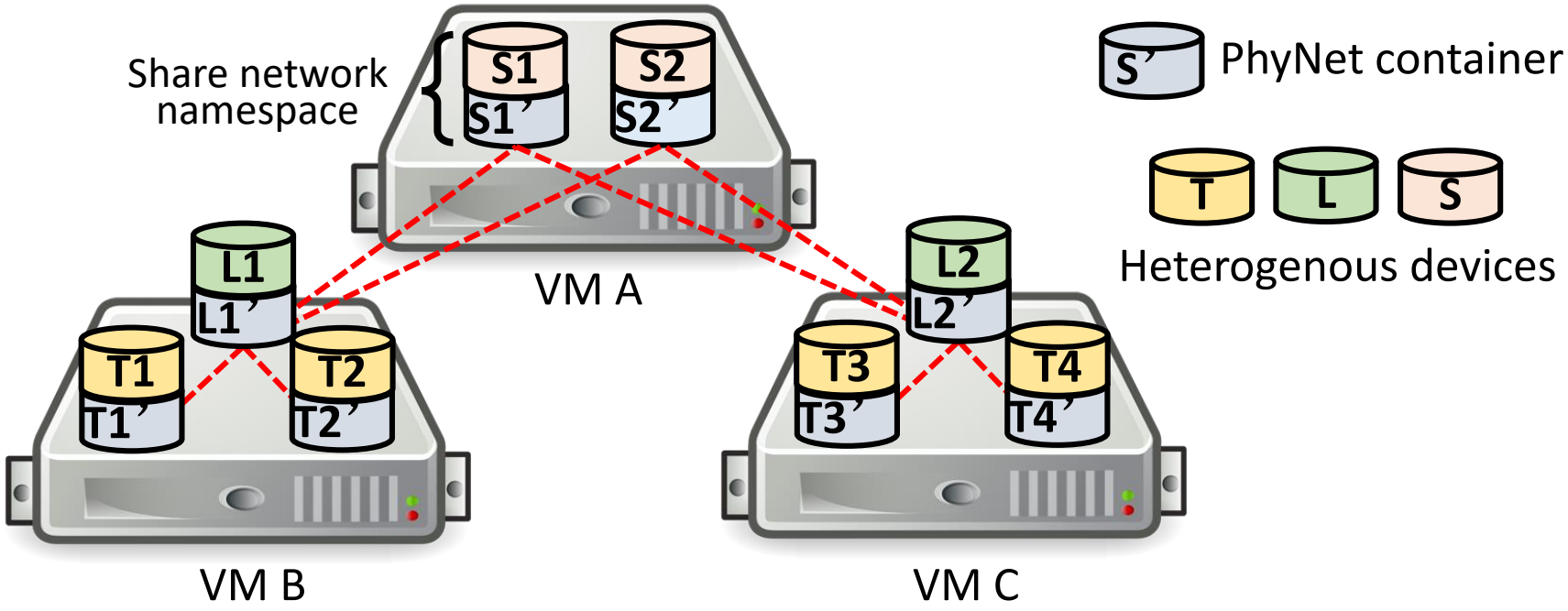
# Feature of CrystalNet: uniform network layer




**Problems**

- Chicken&Egg: veth created after container starts v.s. switch OS checks interfaces when container starts
- hard to insert veth or vlinks into closed device sandboxes
- cannot easily manage network within closed sandboxes

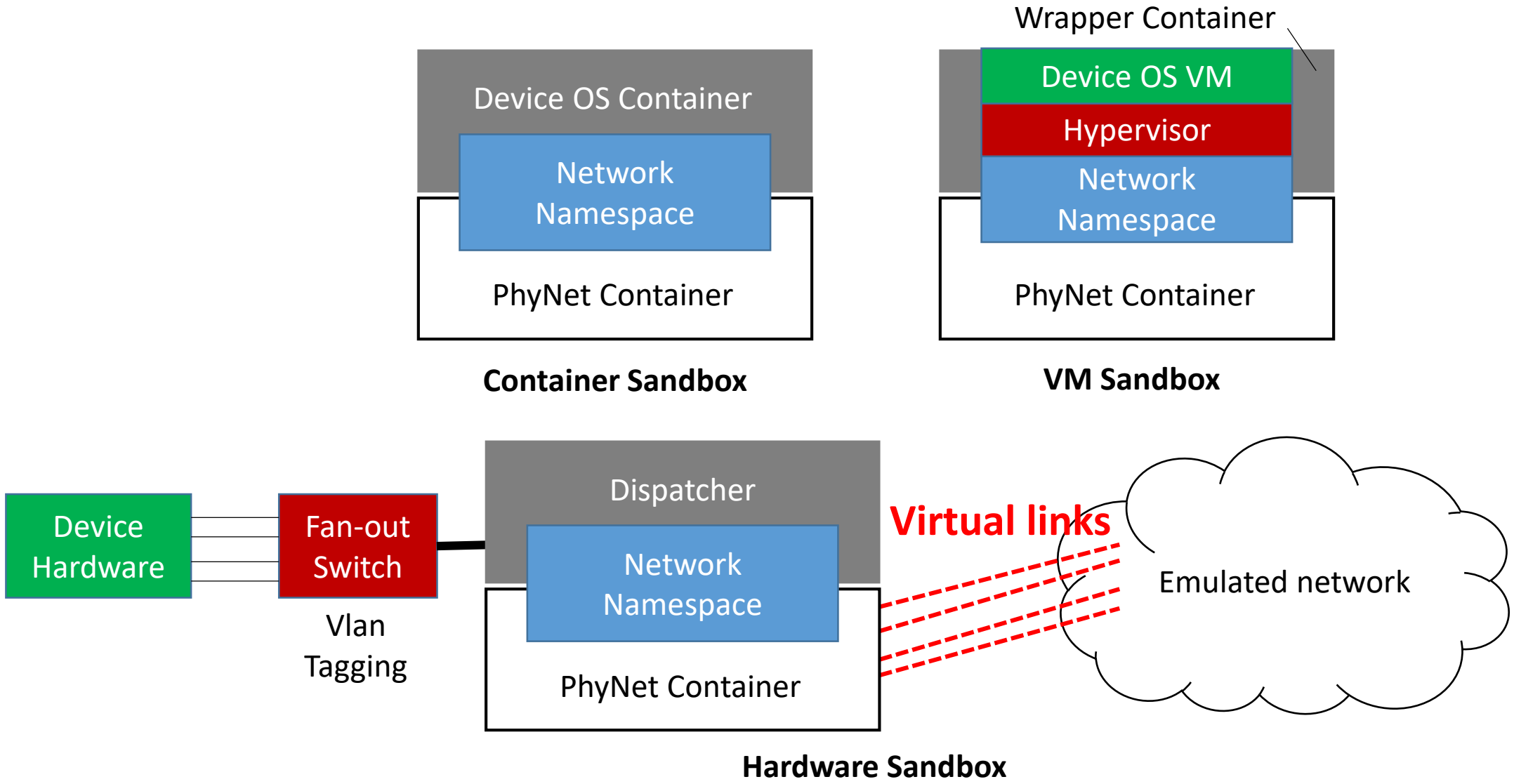
# Feature of CrystalNet: uniform network layer



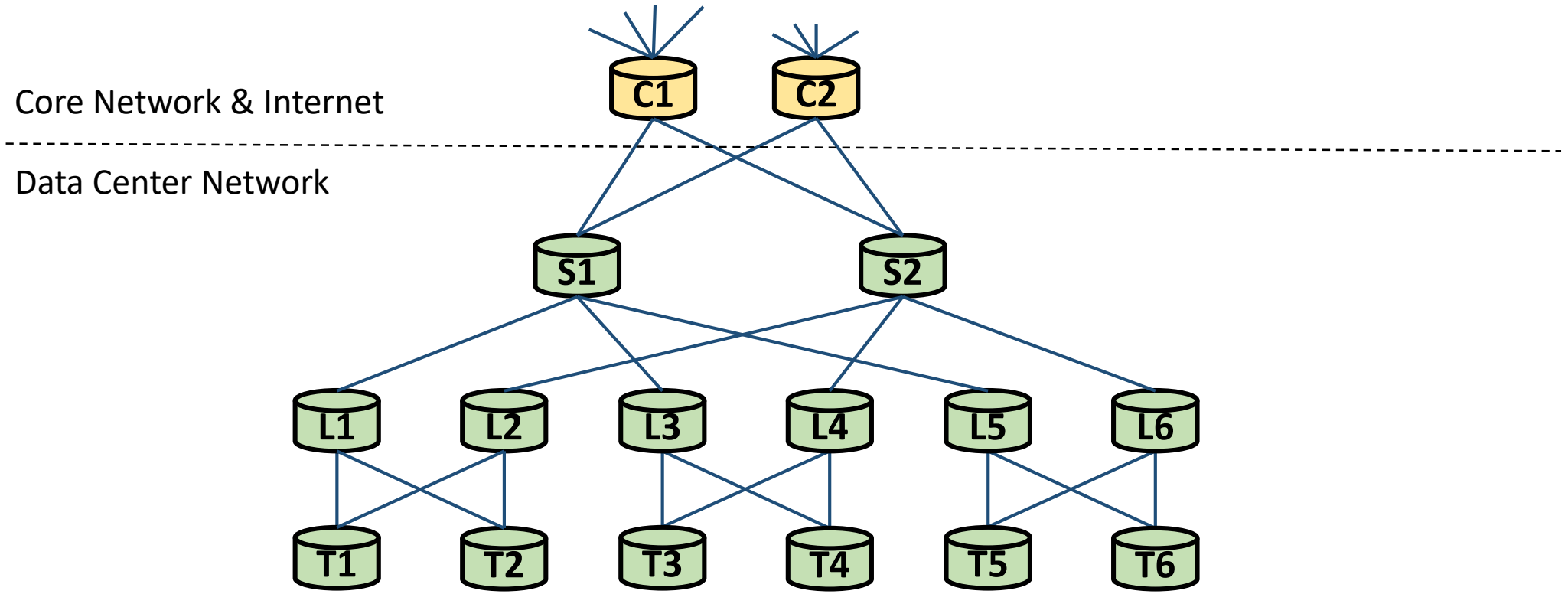

**Key idea: maintaining network with a homogenous layer of containers**

- start a PhyNet container for each device
- build overlay networks among PhyNet containers
- Managing overlay networks with in PhyNet containers

# Feature of CrystalNet: uniform network layer

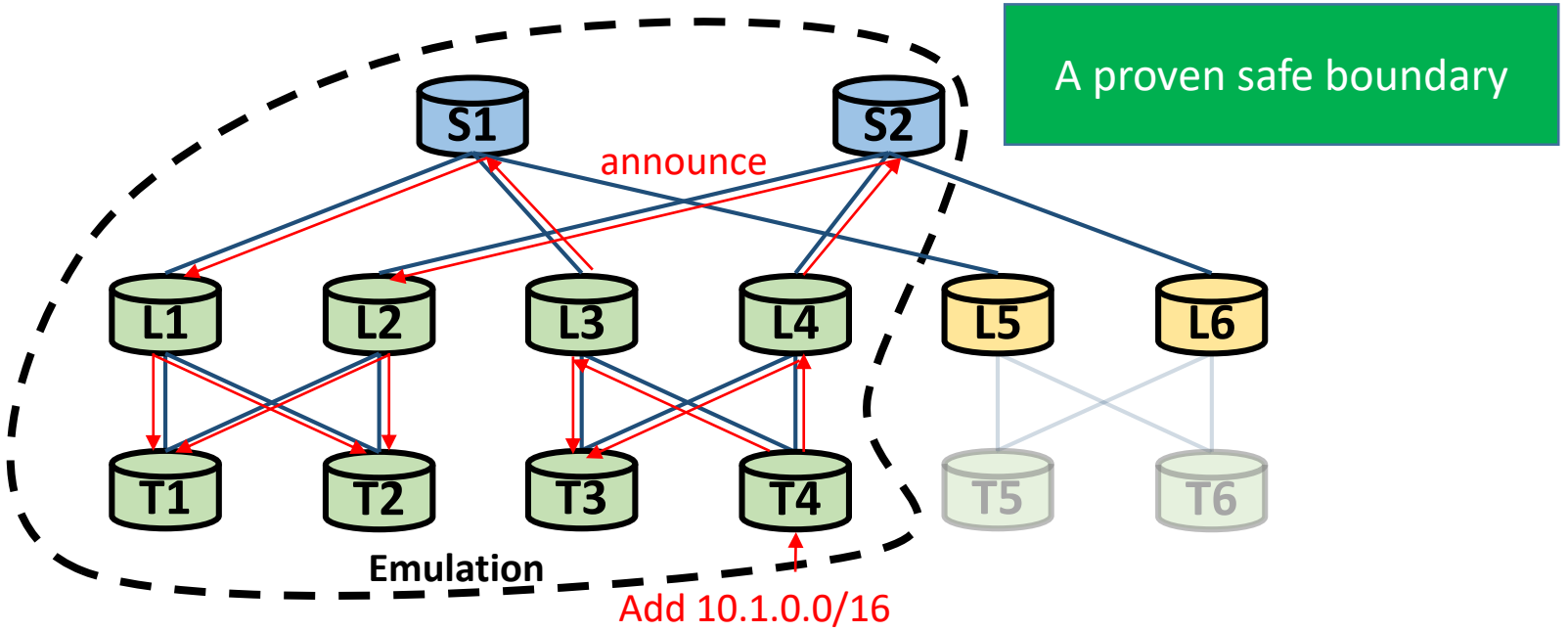


# Feature of CrystalNet: low cost




**Entire DC (Largest): $100/hour**

# Feature of CrystalNet: small & safe boundary

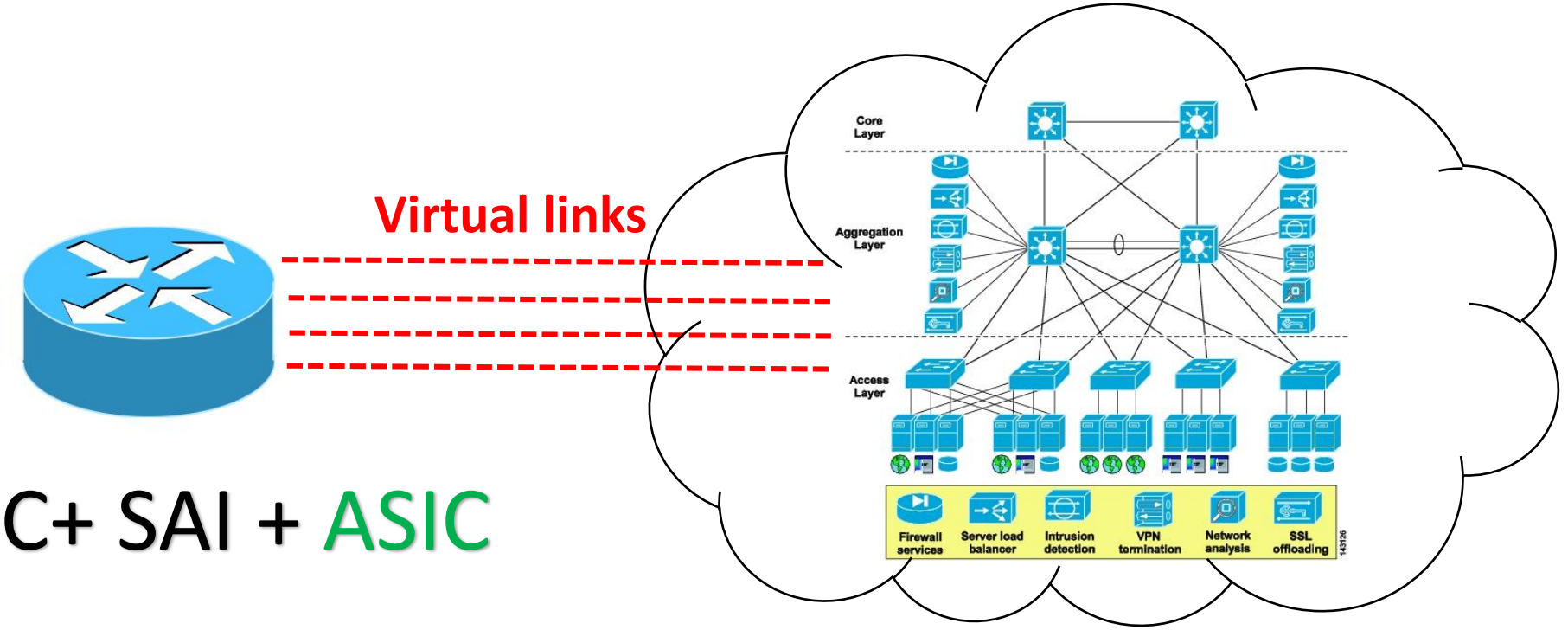



**Partial DC with Safe Boundary: $2/hour**

# Outline

- Background & Motivation
- SONiC + P4: a production quality control plane for P4
- CrystalNet: a cloud-scale network emulator
- **Experiences with SONiC + P4 + CrystalNet**
- Conclusion

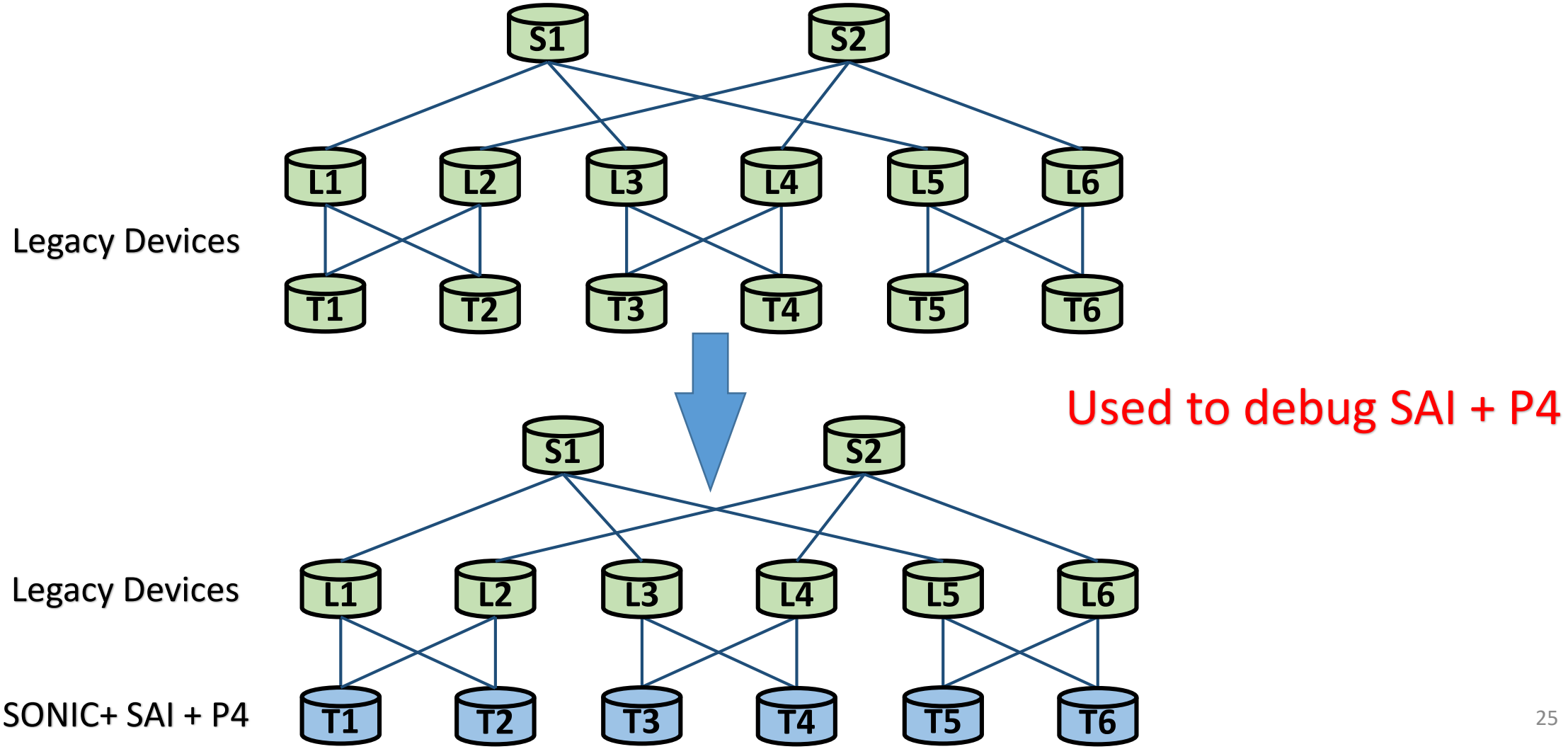
# Single SONiC hardware box testing




SONIC+ SAI + ASIC

Emulated Network in Cloud

# Multiple SONiC container box testing



# Outline

- Background & Motivation
- SONiC + P4: a production quality control plane for P4
- CrystalNet: a cloud-scale network emulator
- Experiences with SONiC + P4 + CrystalNet
- **Conclusion**

# How do these P4-features work in the wild?

Production-quality Switch Control Plane (SONiC)

Close-to production network emulation (CrystalNet)

# An ideal network feature deployment pipeline

Feature development & testing (vendors)

Testing against customers production environments

In production adoption (customers)

Questions?

**SONiC**: https://github.com/Azure/SONiC/
**CrystalNet**: crystalnet-dev@microsoft.com