Leveraging P4 to Automatically Validate Networking Switches

Stefan Heule
heule@google.com

Konstantin Weitz
konne@google.com

Waqar Mohsin
wmohsin@google.com

Lorenzo Vicisano
vicisano@google.com

Amin Vahdat
vahdat@google.com

Leveraging P4 to Automatically Validate Networking Switches

Stefan Heule
heule@google.com

Konstantin Weitz
konne@google.com

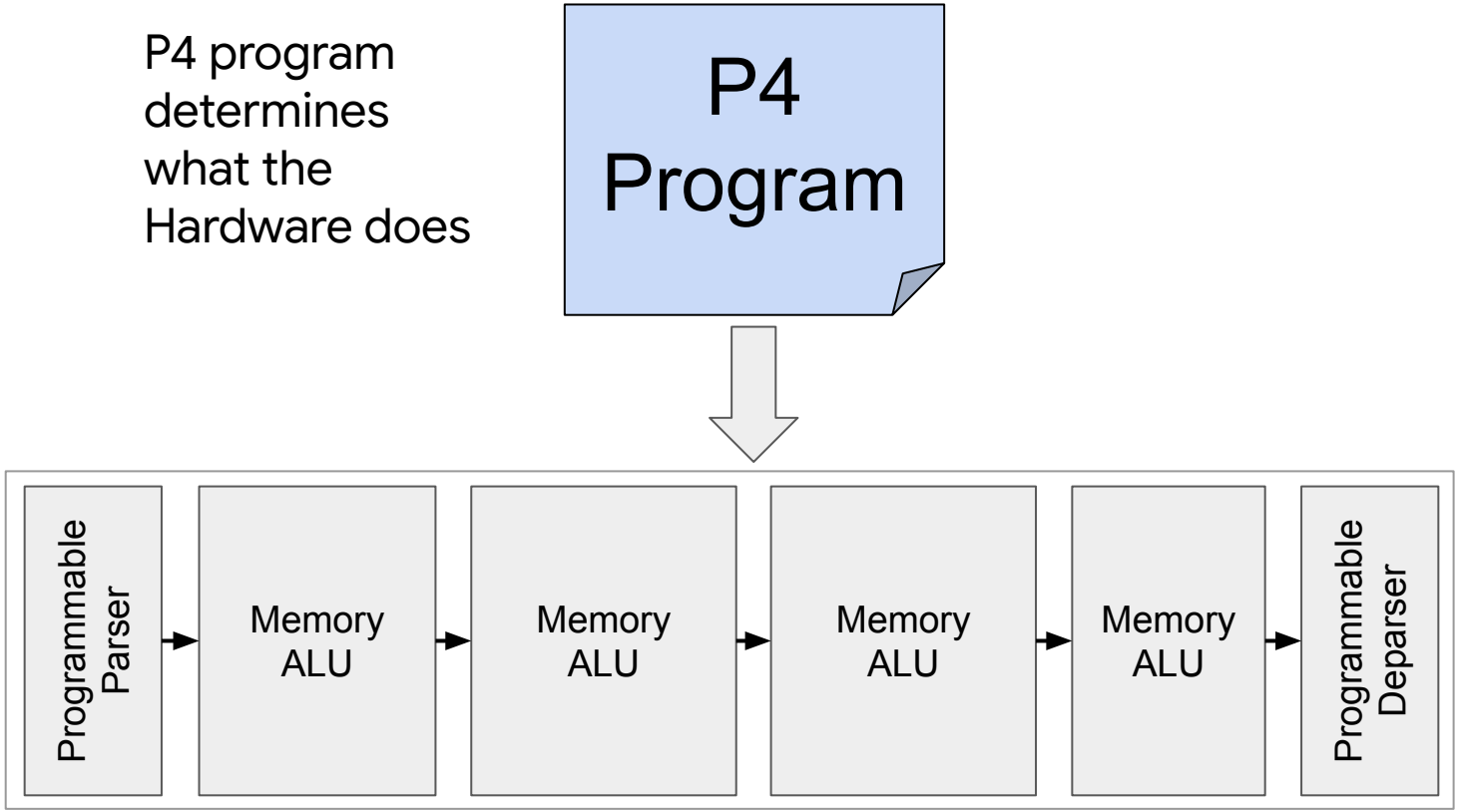
Waqar Mohsin
wmohsin@google.com

Lorenzo Vicisano
vicisano@google.com

Amin Vahdat
vahdat@google.com

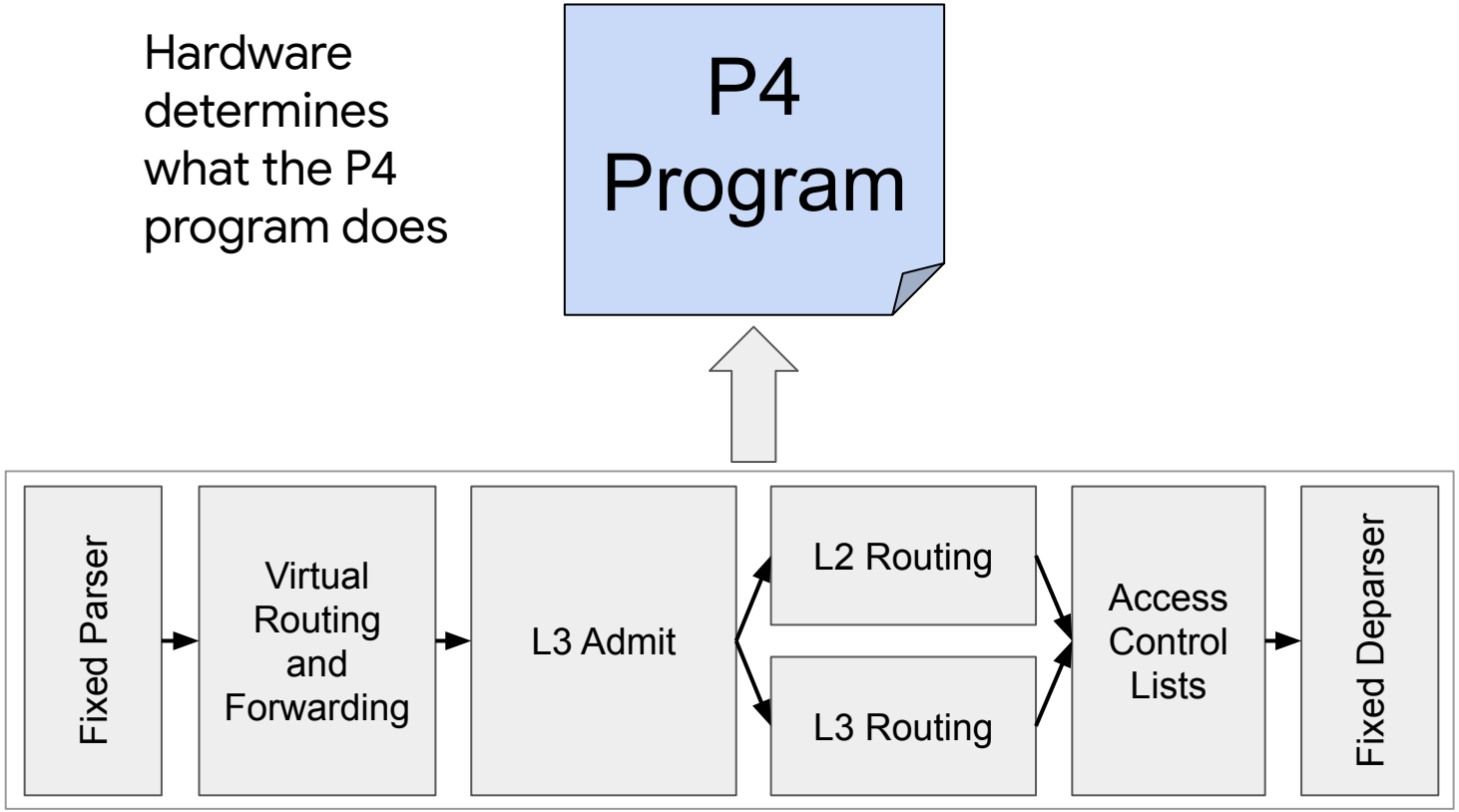
P4 on Programmable Switches

P4 program determines what the Hardware does



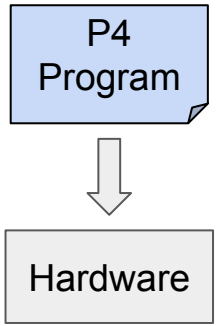
P4 on Fixed-Function Switches

Hardware determines what the P4 program does

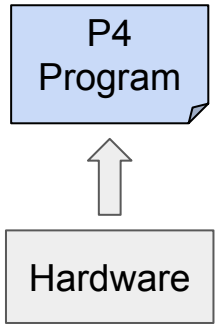


P4 at Google

P4 program determines what the hardware does



Hardware determines what the P4 program does



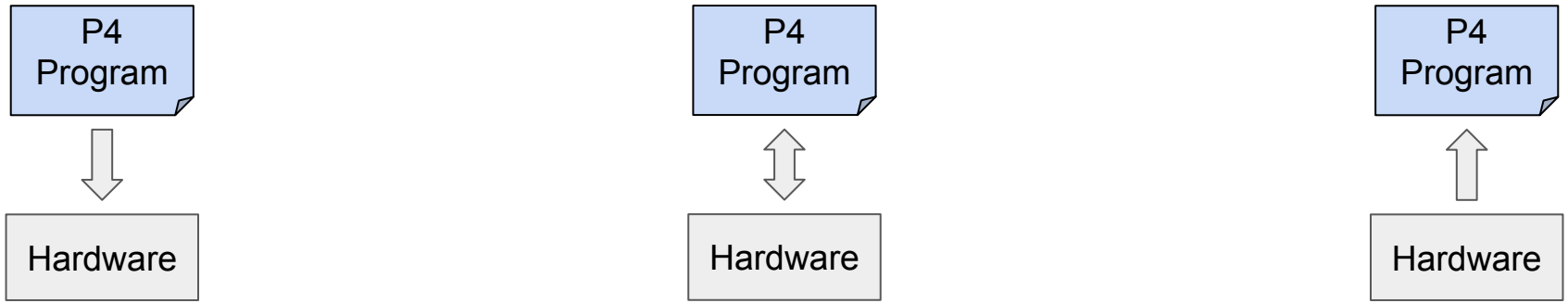
P4 at Google

Hardware limits what P4 program can do, but only model our **use case**:

- Only tables we use (e.g. no L2)
- Only match keys we use
- Logical tables that have semantic meaning (abstraction)

P4 program determines what the hardware does

Hardware determines what the P4 program does



Why would you want to do this?

Clear *contract* of switch behavior enables:

- Operation of a heterogeneous fleet
- Automatically generating switch config
- Automated switch validation

Automated Switch Validation

Automated Switch Validation

Test inputs are automatically generated,
either from production data,
or by analyzing our P4 programs.

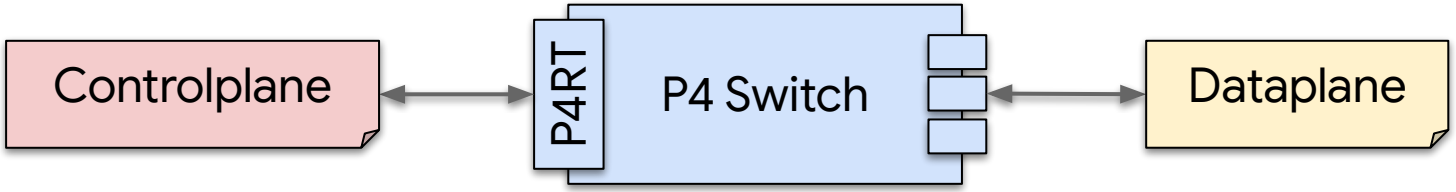
Automated Switch Validation

We validate a single switch chip, not the whole network.

Automated Switch Validation

Test outputs are compared to a P4 program simulation.

How do we test the switch?



Replay production table entries

Fuzzer to randomly create table entry insert/delete requests

ATPG: Automated Test Packet Generation

Counters, Meters, Hashing

Controlplane Fuzz Testing

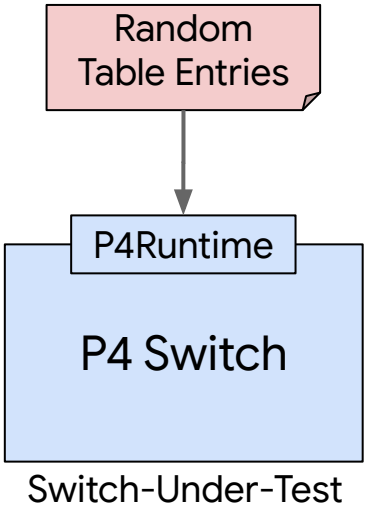
Controlplane Fuzzing

Randomly generate table entry requests according to P4 program grammar

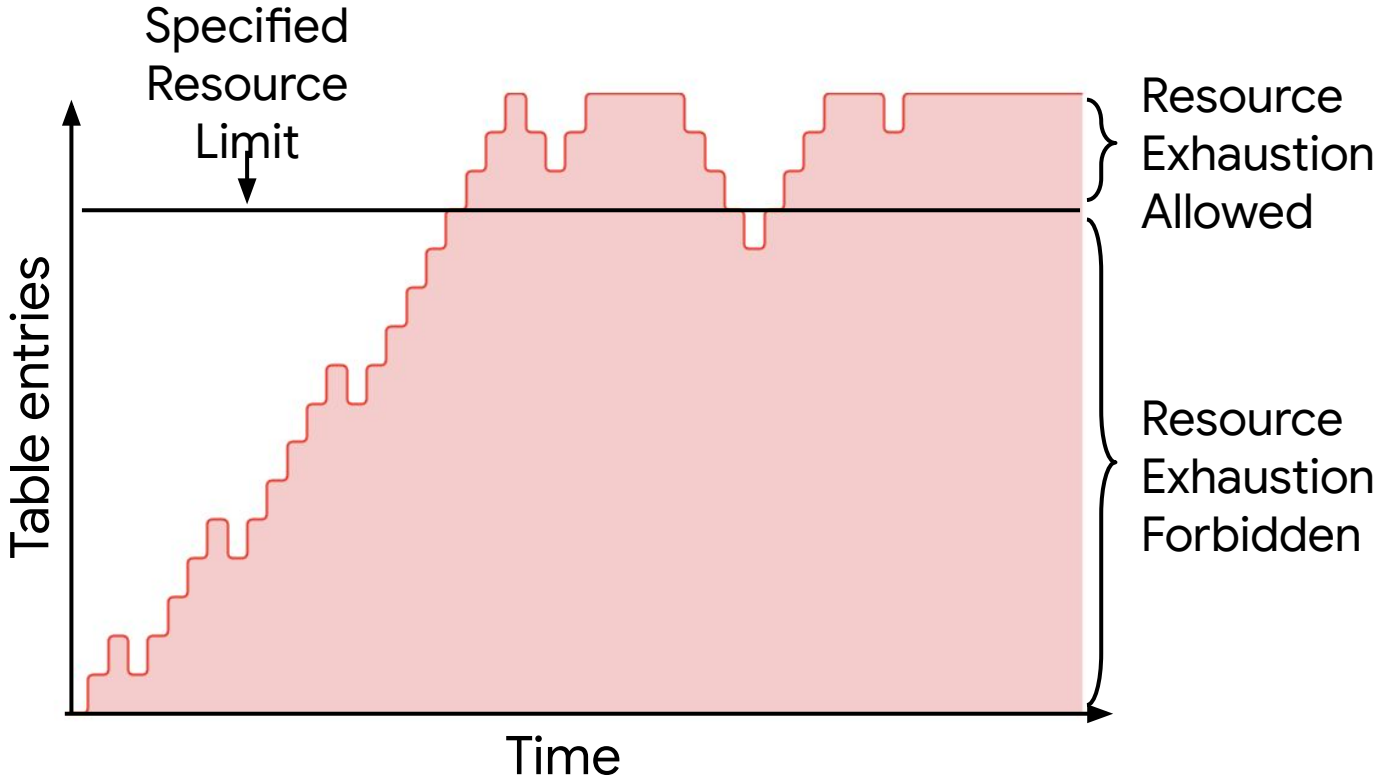
- Mostly generate well-formed requests
- Sometimes generate ill-formed ones
- Intuition: Need to be well-formed enough to not get rejected early

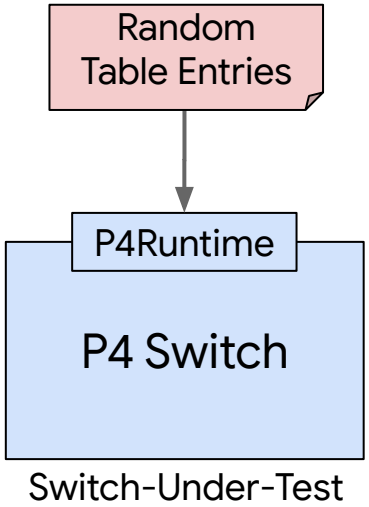
Send table entry to switch, check that they are handled correctly

- E.g. well-formed insert must succeed (unless resource exhausted or already present)
- P4 allows us to accurately predict the expected error (or success)



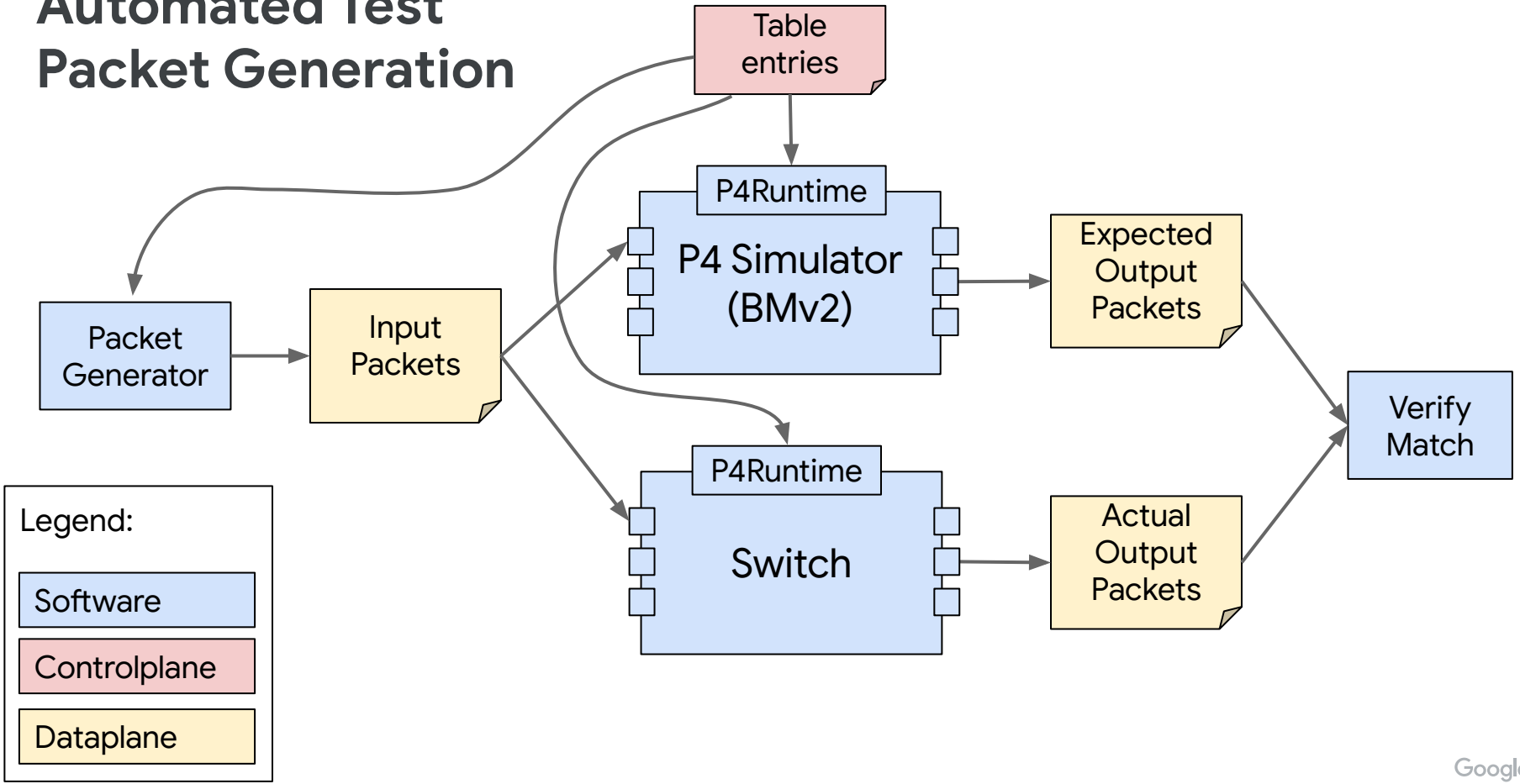
Controlplane Fuzzing: Resource exhaustion





Automated Test Packet Generation

Automated Test Packet Generation



Strategy: Hitting every table entry on the switch

VRF Classifier

EthType	SrcMac	Port	Set VRF
0x800	aa:bb:cc: dd:ee:ff	*	1337
0x800	*	4	42

IPv4 LPM

VRF	DstIP
42	10.152.8/24
42	10.152/16
...	...

Want to hit this entry

```
VRF == 42 & DstIP[32:16] == "10.152"                    // hit target IPv4 LPM entry
& !(VRF == 42 & DstIP[32:8] == "10.152.8") & !(...)     // avoid all other IPv4 LPM entry
// encode VRF assignment
& ((!(EthType == 0x800 & SrcMac == "aa:bb:cc:dd:ee:ff")
   & (EthType == 0x800 & Port == 4)) ⟶ VRF == 42)
```

SAT solver finds packets to satisfy the formula

Dataplane Testing: Why Does It Work?

SAT is an excellent match for switches/P4:

- Everything is finite
  (no lists, loops, recursion, etc)

- Switch semantics are rigorously defined in the P4 program

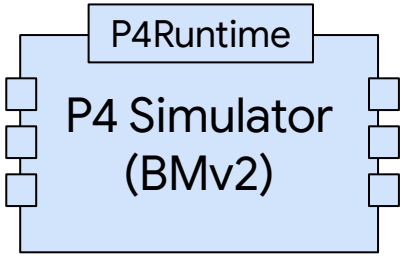
Powerful tool to ask complex questions about behavior of the switch

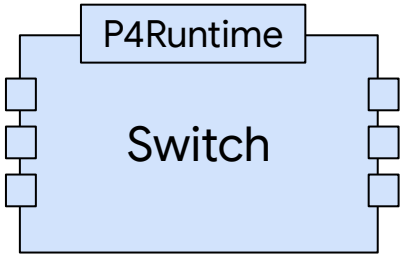
Testing Other Aspects: Counters, Meters

Comparing the switch against simulator is very general

- Allows us to easily test other aspects like counters

Challenge: hashing





Dataplane Testing: why it works

P4

Test oracle: Clear semantics allow simulator to precisely predict switch behavior

Test generation: Semantics are simple enough that tools can reason about them automatically

OpenFlow

Lack of formal and computer-readable specification makes both difficult to do automatically

Does Automated Switch Validation Work?

Small number of devs create extensive set of automated tests

So far, we found over 100 bugs, in several components:

- Bugs in the Switch Software Stack
- Bugs in our SDN Controller
- Bugs in our P4 program
- Bugs in the P4 Runtime protocol
- Bugs in BMv2

Conclusion

Key Takeaways

P4 provides a clear contract of switch behavior:

- Enables operation of a heterogeneous fleet
- Enables automated switch validation
  (it's fast and finds a broad spectrum of bugs)

Sounds interesting? We're hiring! Talk to us :)

Email: heule@google.com