ONE SUMMIT

P4-Based Automated Reasoning

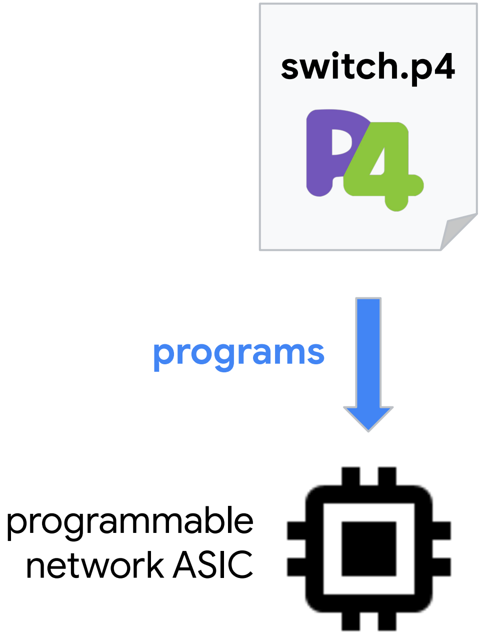
Steffen Smolka
smolkaj@google.com

Hosted By THE LINUX FOUNDATION | LF NETWORKING | LF EDGE

#onesummit

Google's surprising use of P4

P4 as intended

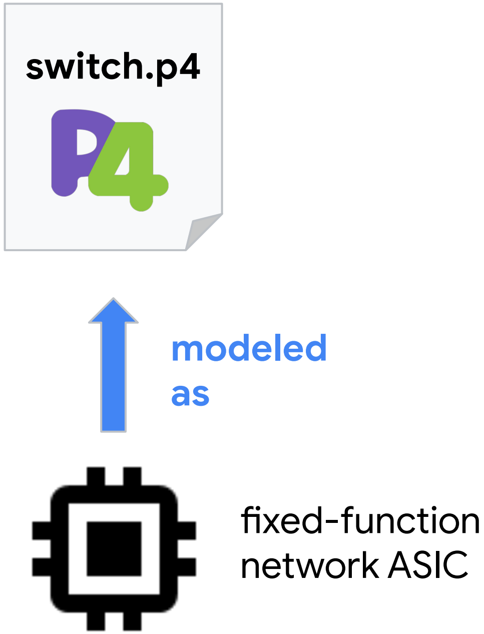


switch.p4

programs

programmable network ASIC

P4 at Google



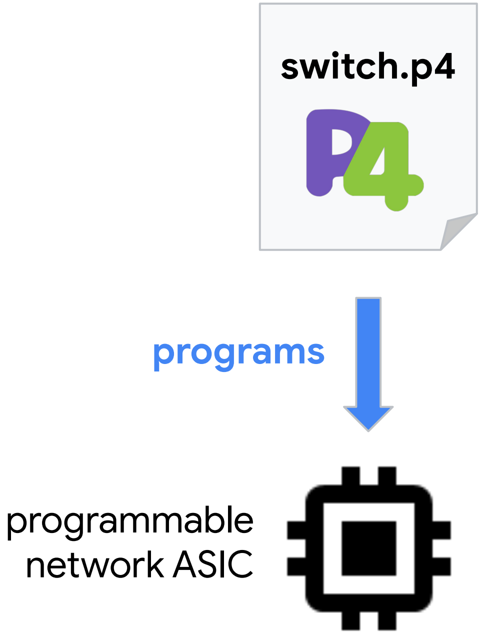
switch.p4

modeled as

fixed-function network ASIC

Google's surprising use of P4

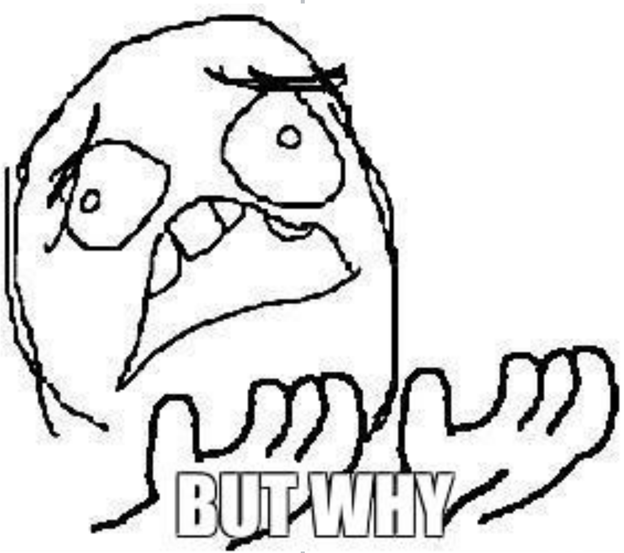
P4 as intended



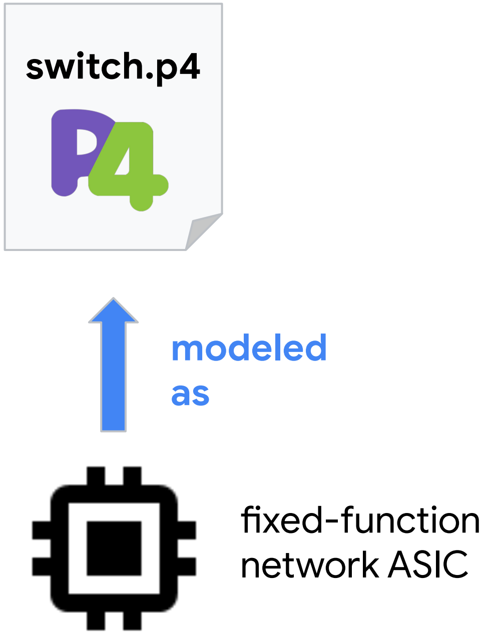
switch.p4

programs

programmable network ASIC



P4 at Google



switch.p4

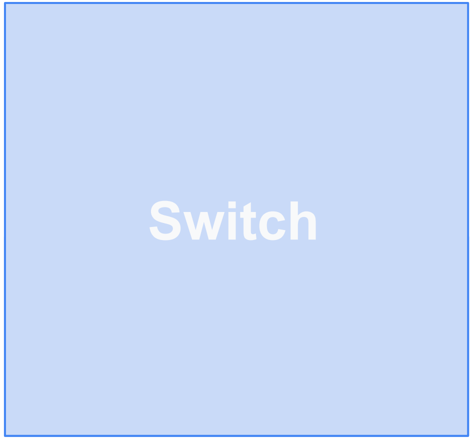
modeled as

fixed-function network ASIC

Building Google's latest data center switch

Backdrop: Building Google's latest SDN switch

Goals:

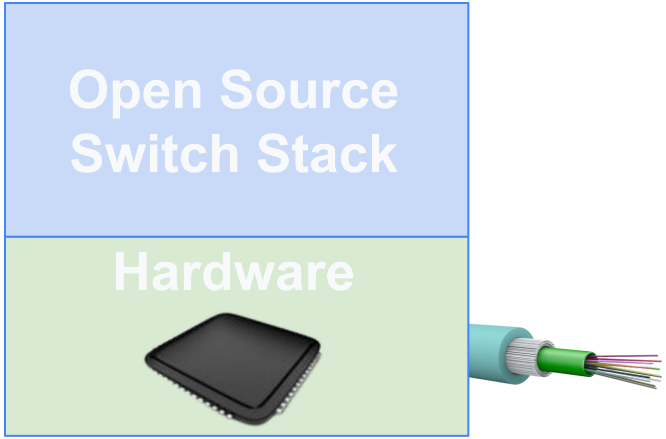


Switch

Backdrop: Building Google's latest SDN switch

Goals:

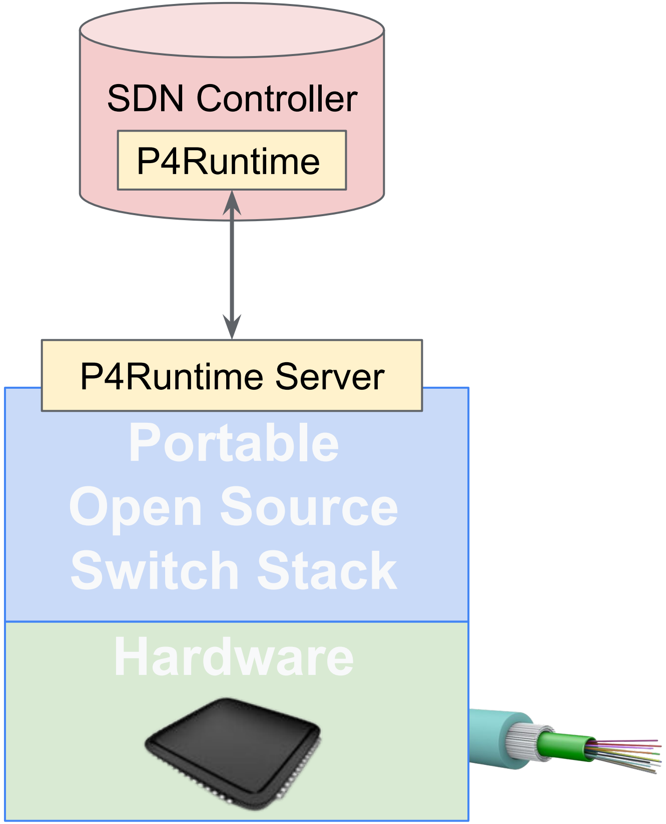
- ➢ **Fast** (new ASIC, new optics)
- ➢ **Open source & industry-aligned**



Open Source Switch Stack

Hardware

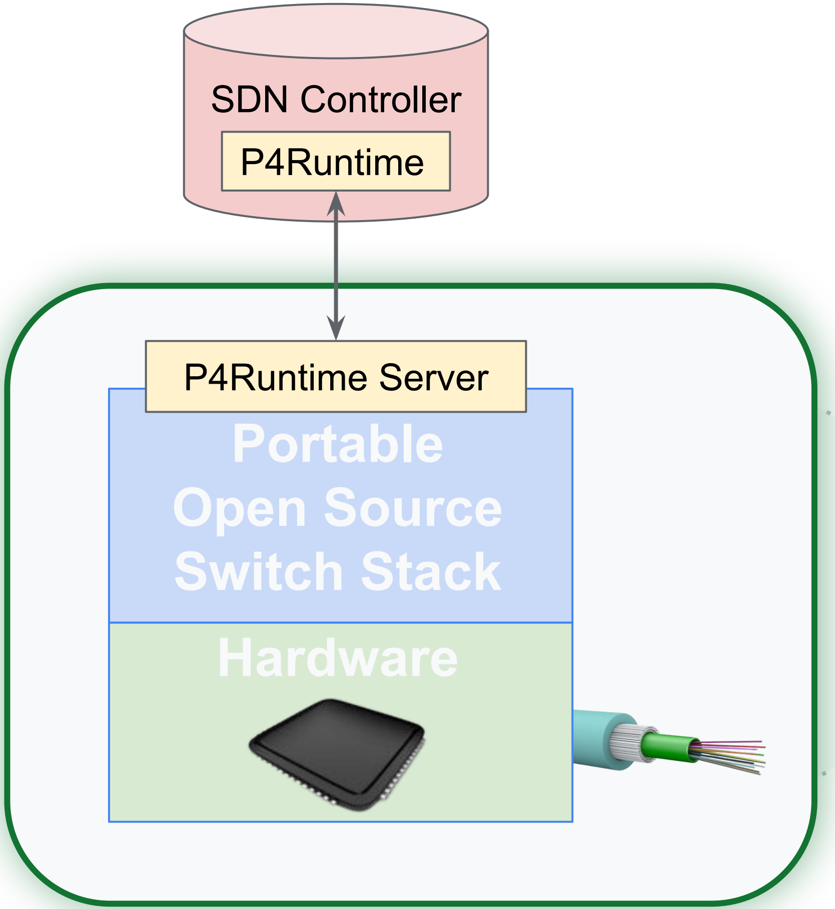
Backdrop: Building Google's latest SDN switch



Goals:

- **Fast** (new ASIC, new optics)
- **Open source & industry-aligned**
- **Silicon independent** (fixed/hybrid/programmable ASICs)
- **New SDN APIs** (OpenFlow -> P4Runtime)

Backdrop: Building Google's latest SDN switch



SDN Controller

P4Runtime

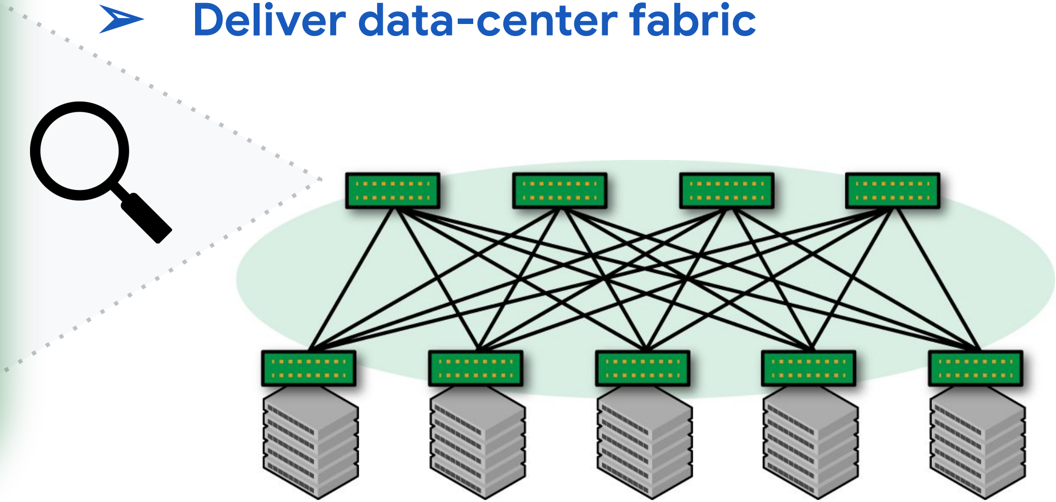
P4Runtime Server

Portable Open Source Switch Stack

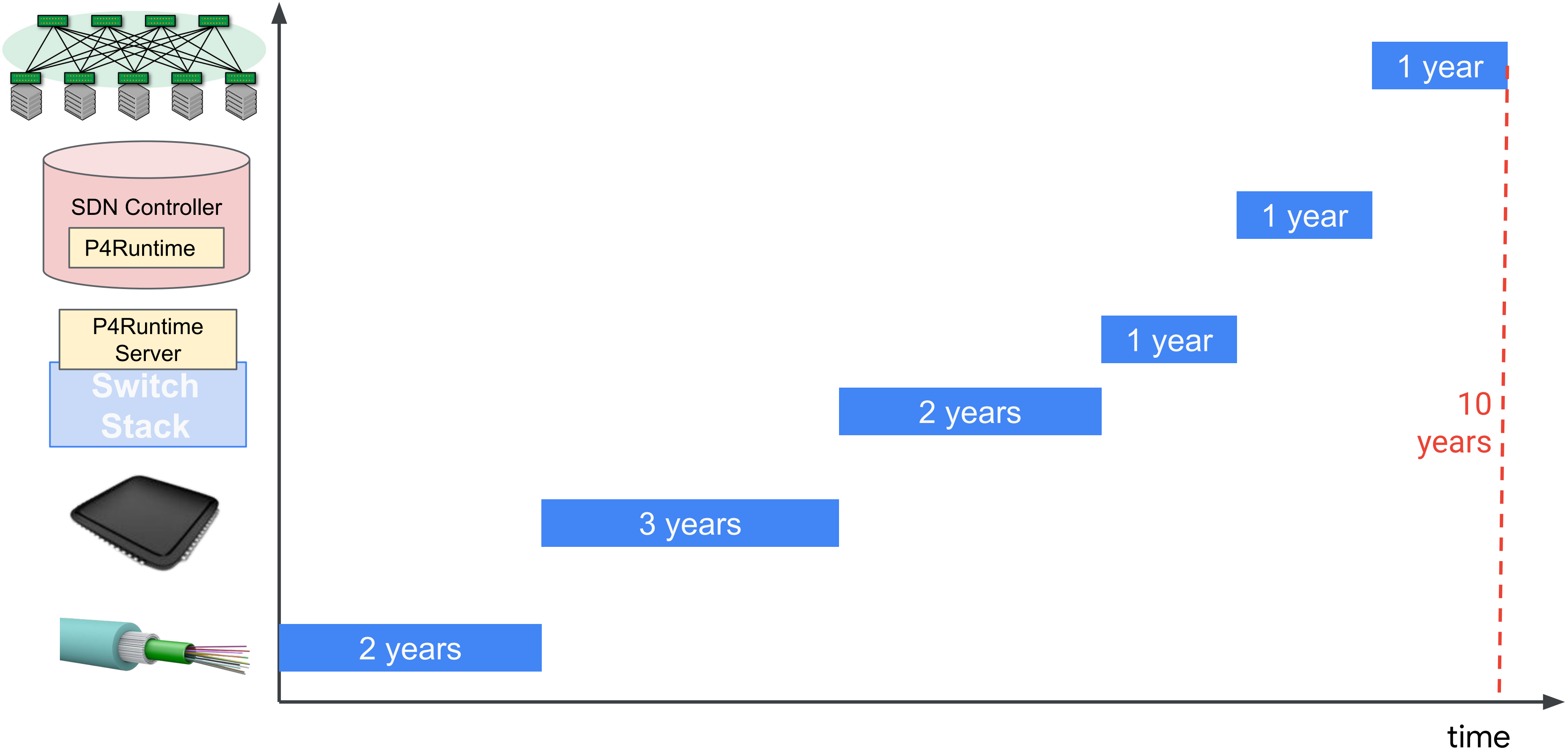
Hardware

Goals:

- ➢ **Fast** (new ASIC, new optics)
- ➢ **Open source & industry-aligned**
- ➢ **Silicon independent** (fixed/hybrid/programmable ASICs)
- ➢ **New SDN APIs** (OpenFlow -> P4Runtime)
- ➢ **Deliver data-center fabric**



Building the switch - Ideal



SDN Controller

P4Runtime

P4Runtime Server

Switch Stack

1 year

1 year

1 year

2 years

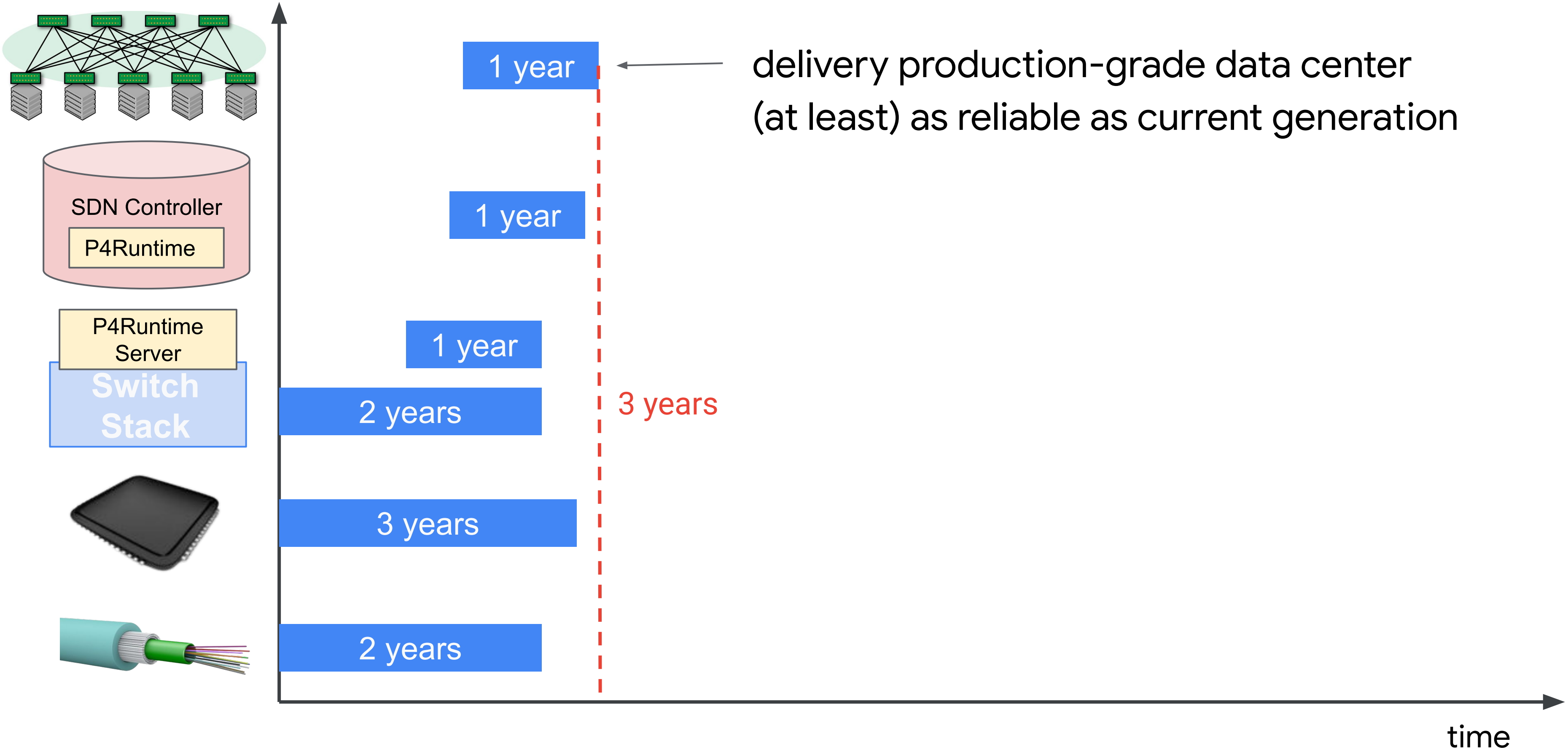
3 years

2 years

10 years

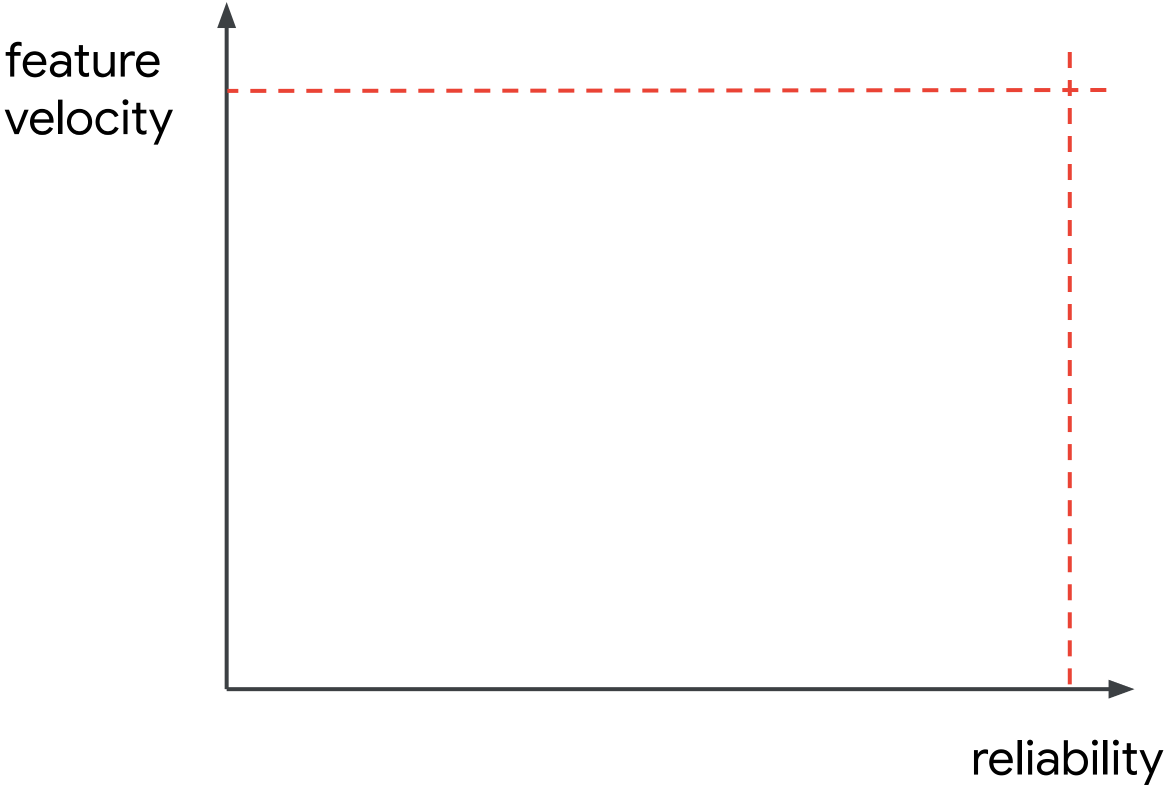
time

Building the switch - Reality

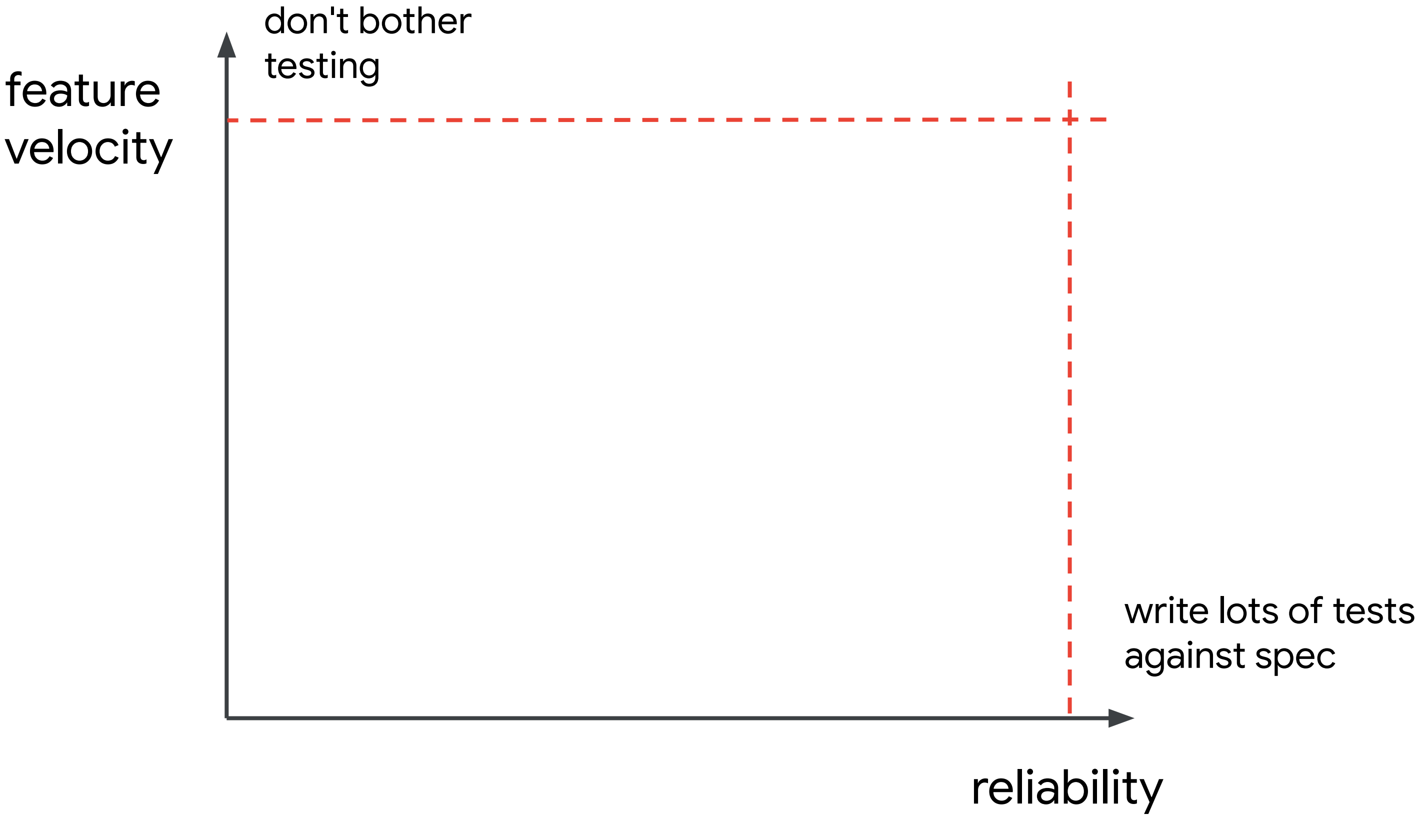


delivery production-grade data center (at least) as reliable as current generation

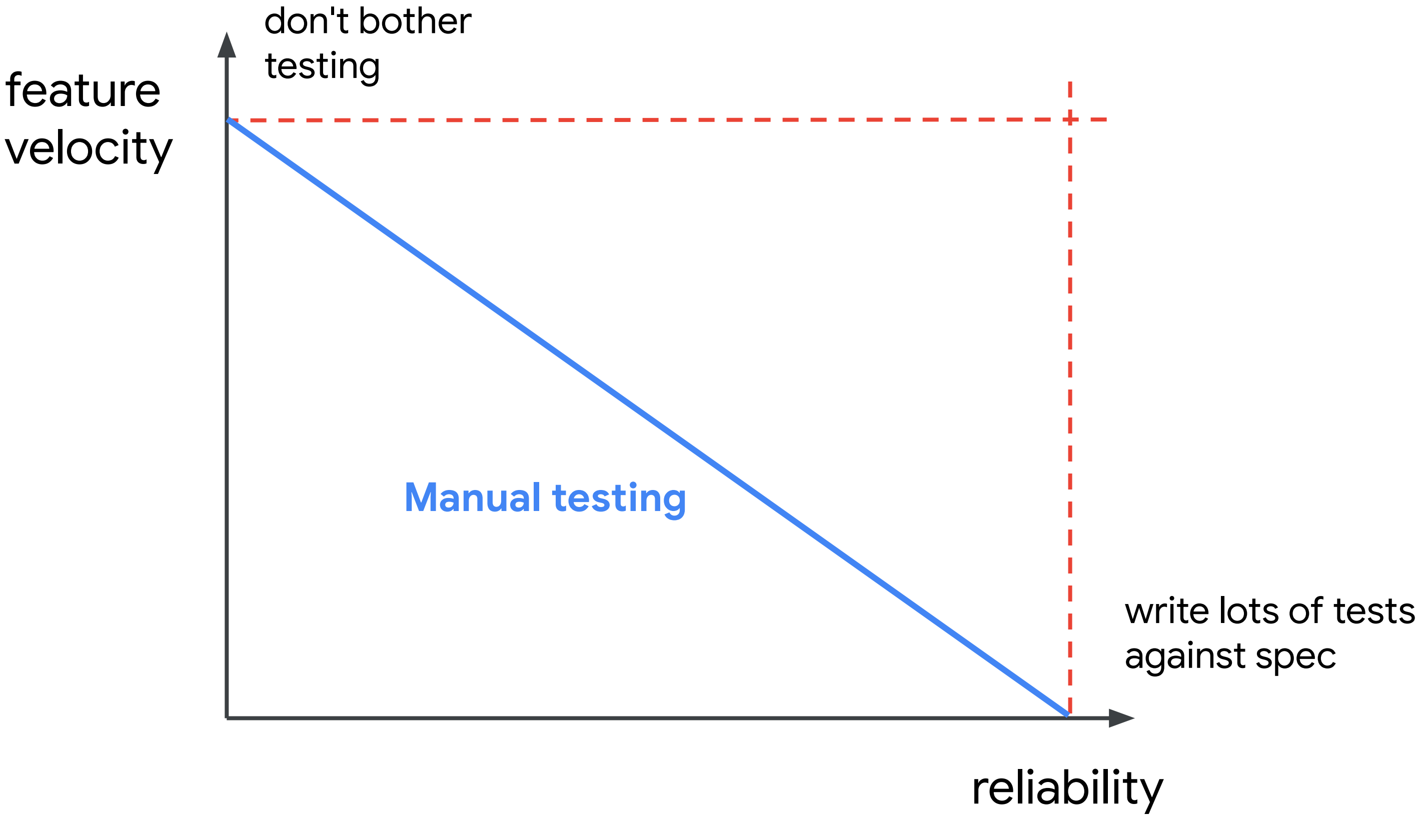
Problem: feature velocity vs reliability



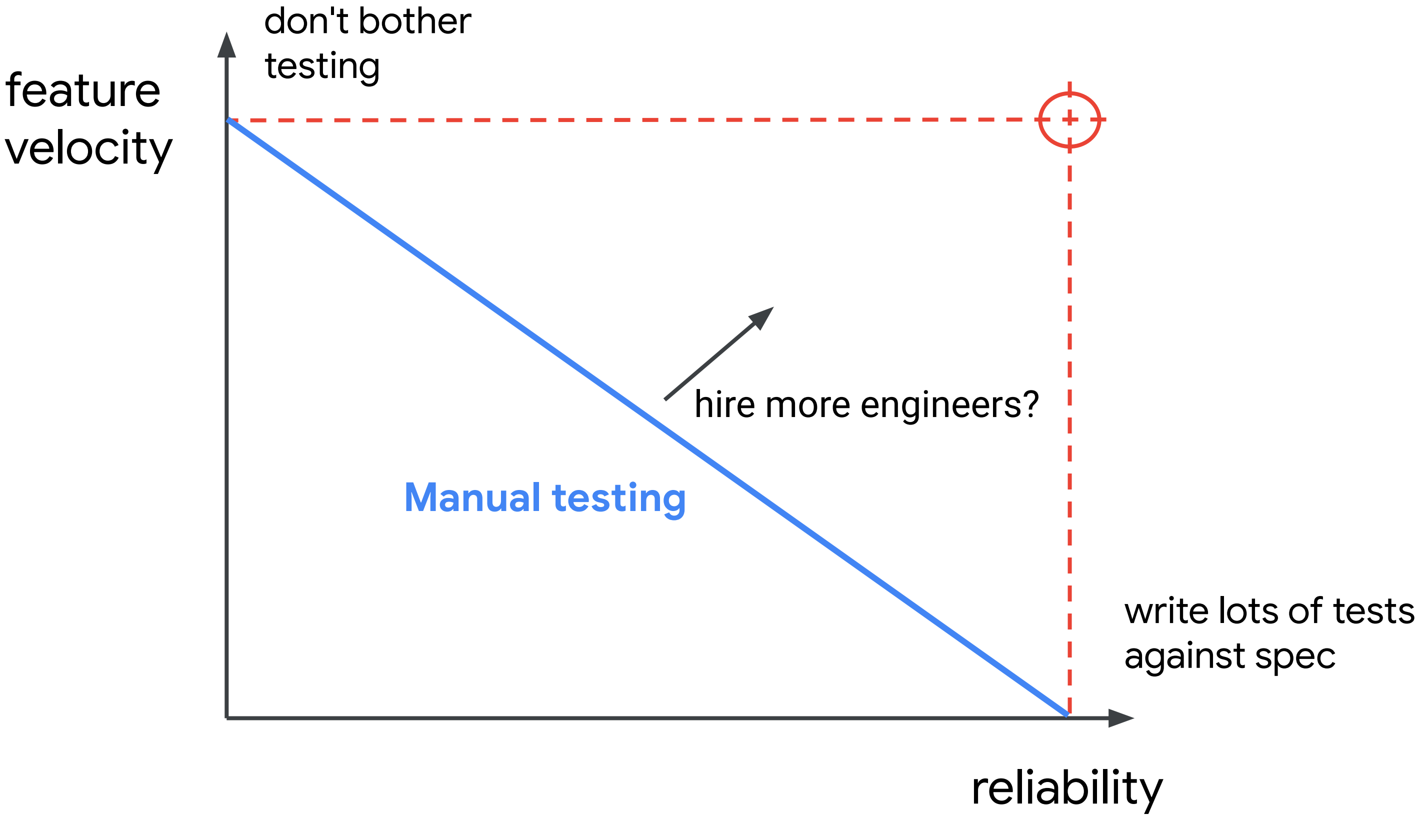
Problem: feature velocity vs reliability



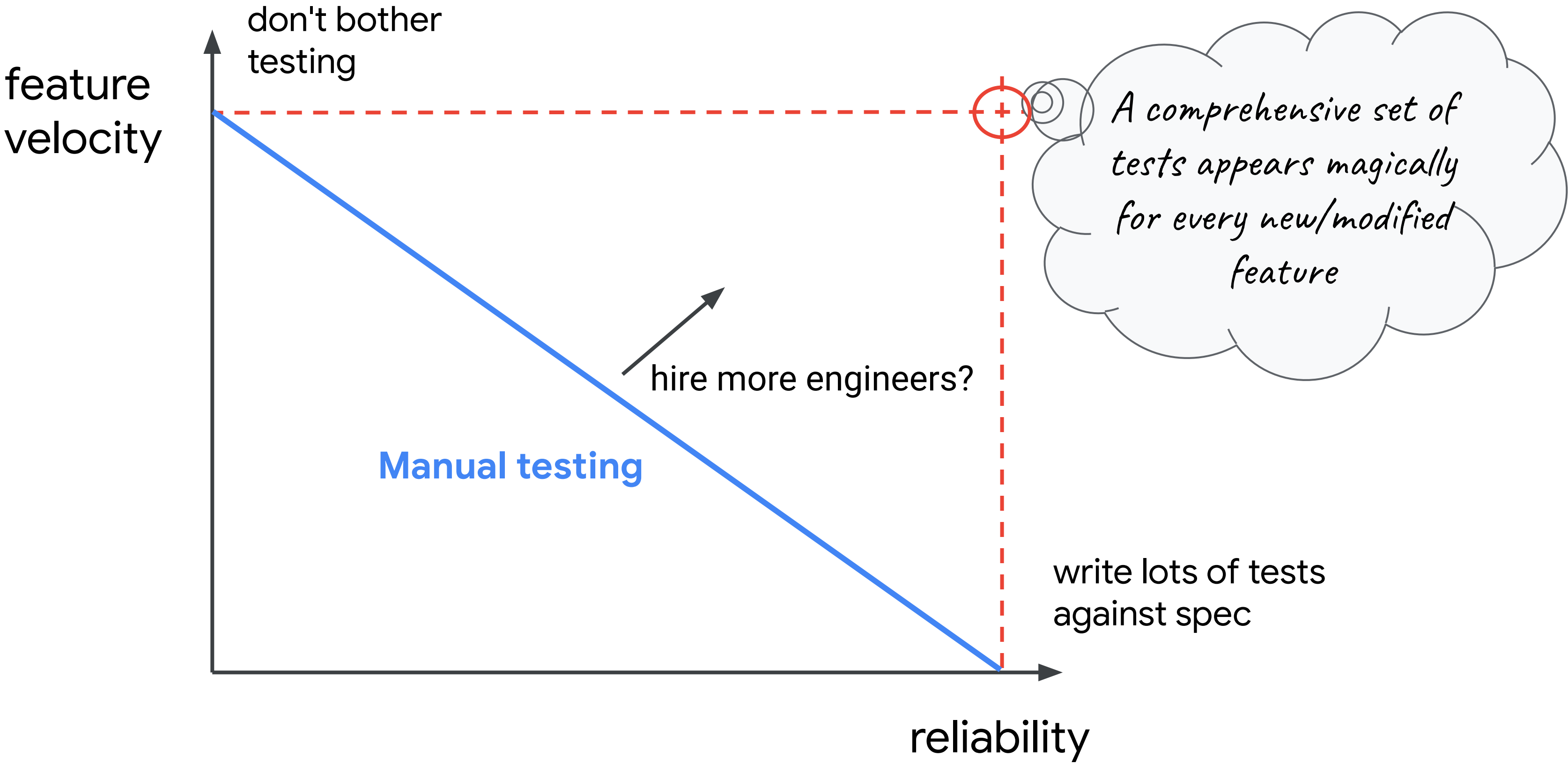
Problem: feature velocity vs reliability



Problem: feature velocity vs reliability



Problem: feature velocity vs reliability

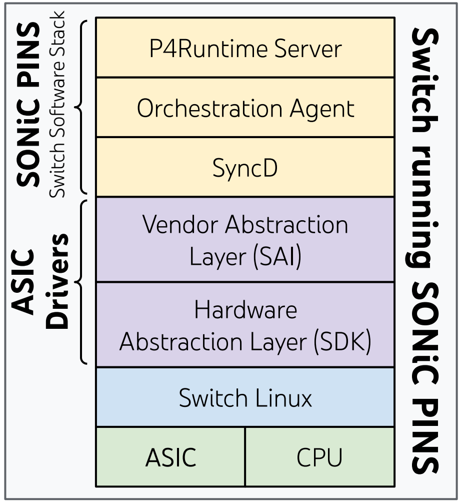


Idea: Automated testing

Idea: Automated testing

🟪 control plane API

🟩 data plane

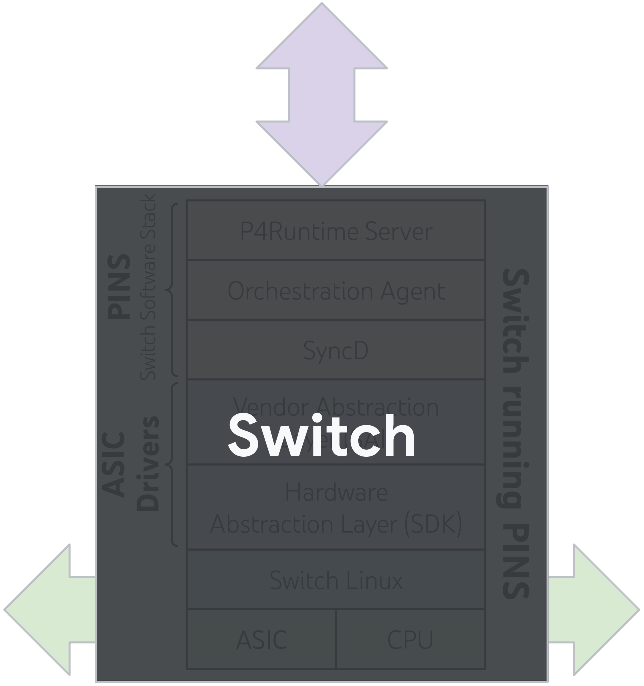


> 10^7 LOC

Idea: Automated testing

🟪 control plane API

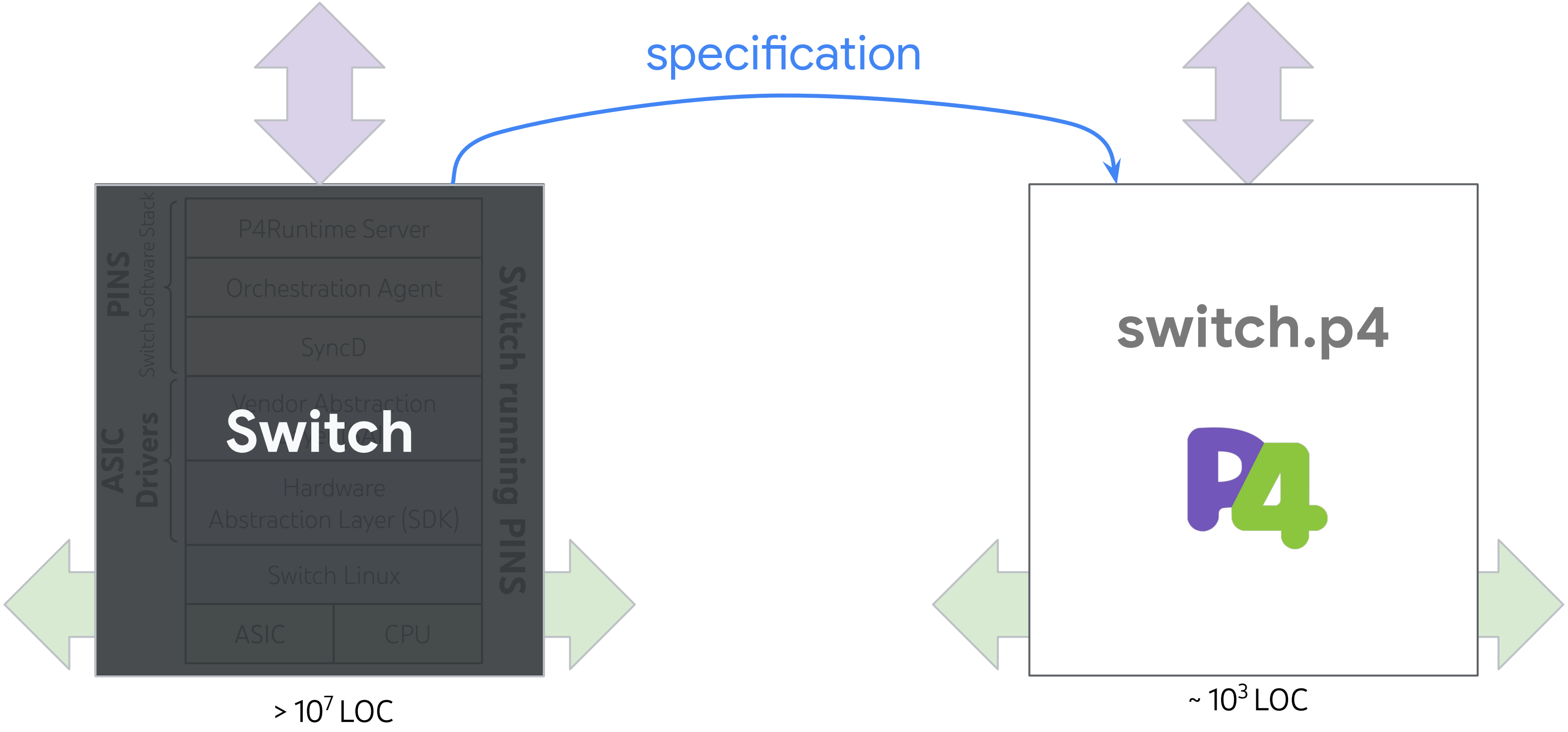
🟩 data plane



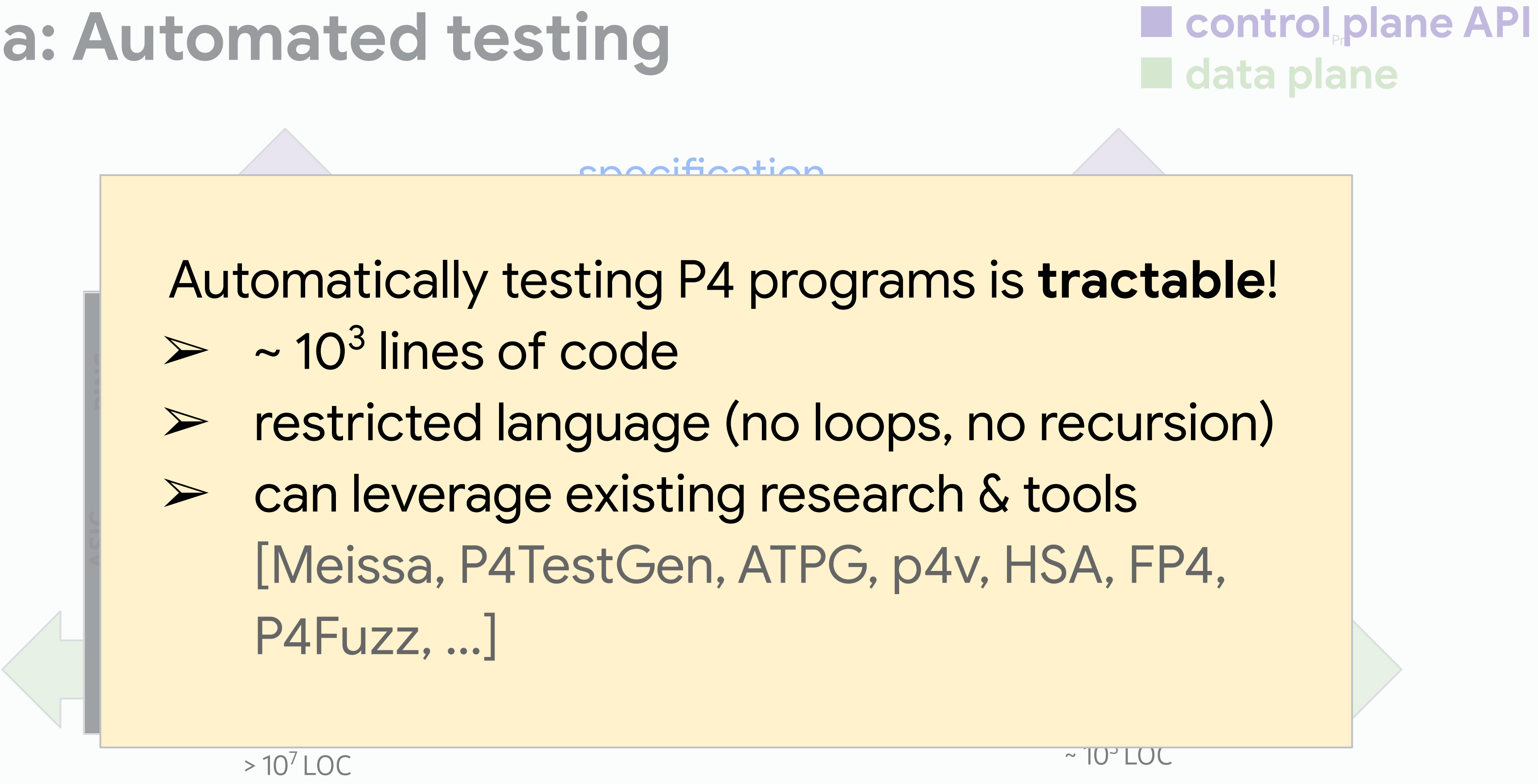
> 10^7 LOC

Idea: Automated testing

🟪 control plane API
🟩 data plane



Idea: Automated testing



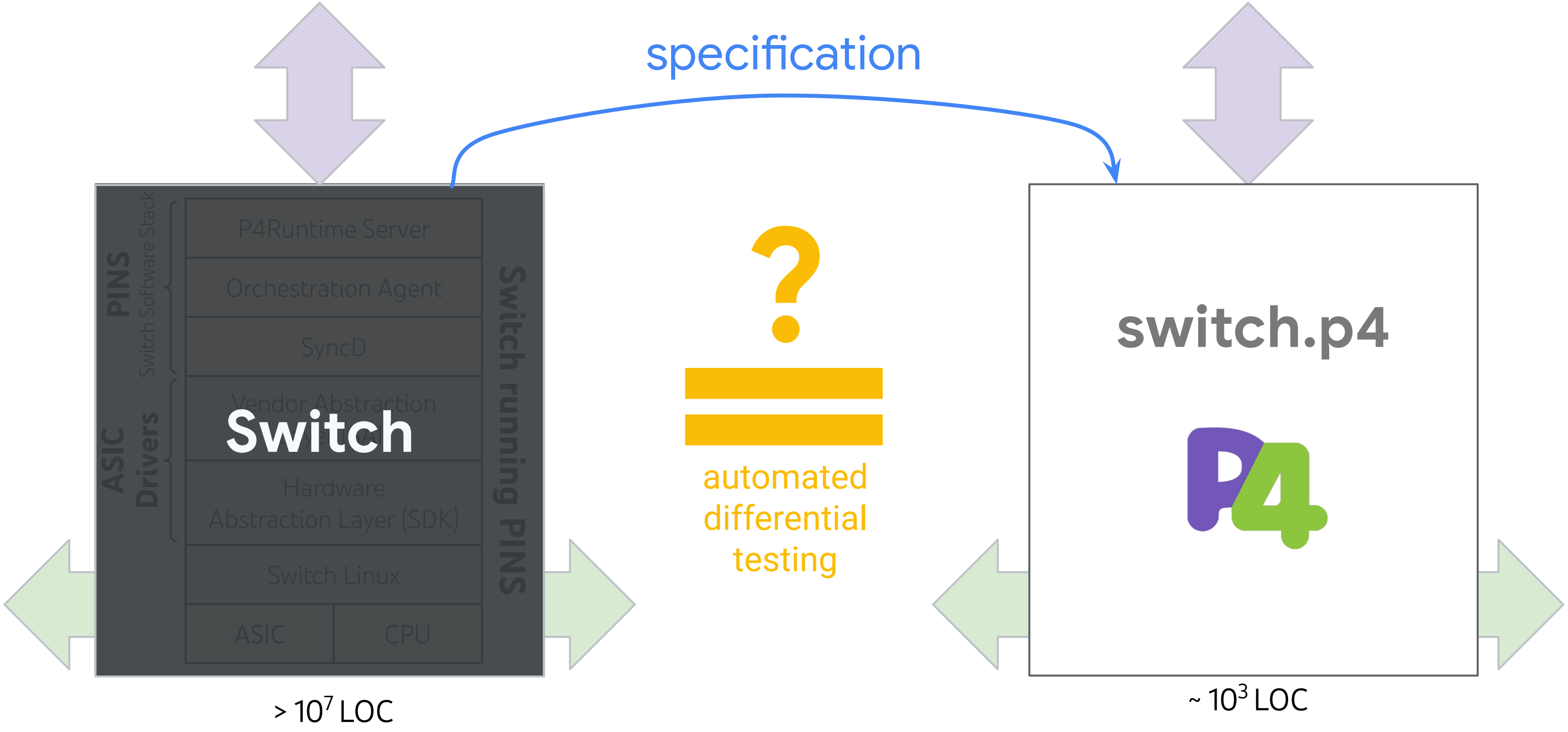
Automatically testing P4 programs is **tractable**!

- ~ 10^3 lines of code
- restricted language (no loops, no recursion)
- can leverage existing research & tools [Meissa, P4TestGen, ATPG, p4v, HSA, FP4, P4Fuzz, ...]

Idea: Automated testing

🟪 control plane API

🟩 data plane



P4-Based Automated Reasoning (P4-BAR)

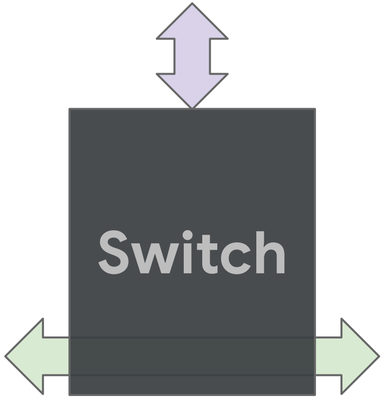
🟪 control plane API
🟩 data plane

A suite of tools that **automatically validate** that a given **switch conforms to** a given **P4 specification**.

p4-fuzzer: generates valid & invalid API requests

p4-symbolic: generates test packets

BMv2: simulates switch dataplane



P4 program = formal specification

P4-Based Automated Reasoning (P4-BAR)

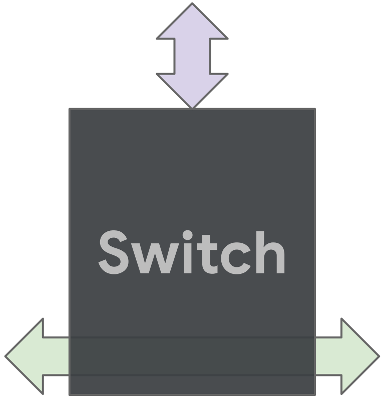
🟪 control plane API
🟩 data plane

A suite of tools that **automatically validate** that a given **switch conforms to** a given **P4 specification**.

p4-fuzzer: generates valid & invalid API requests

p4-symbolic: generates test packets

BMv2: simulates switch dataplane



P4 program = formal specification

Deep Dive:
Dataplane Validation

P4-Based Dataplane Validation

Prc

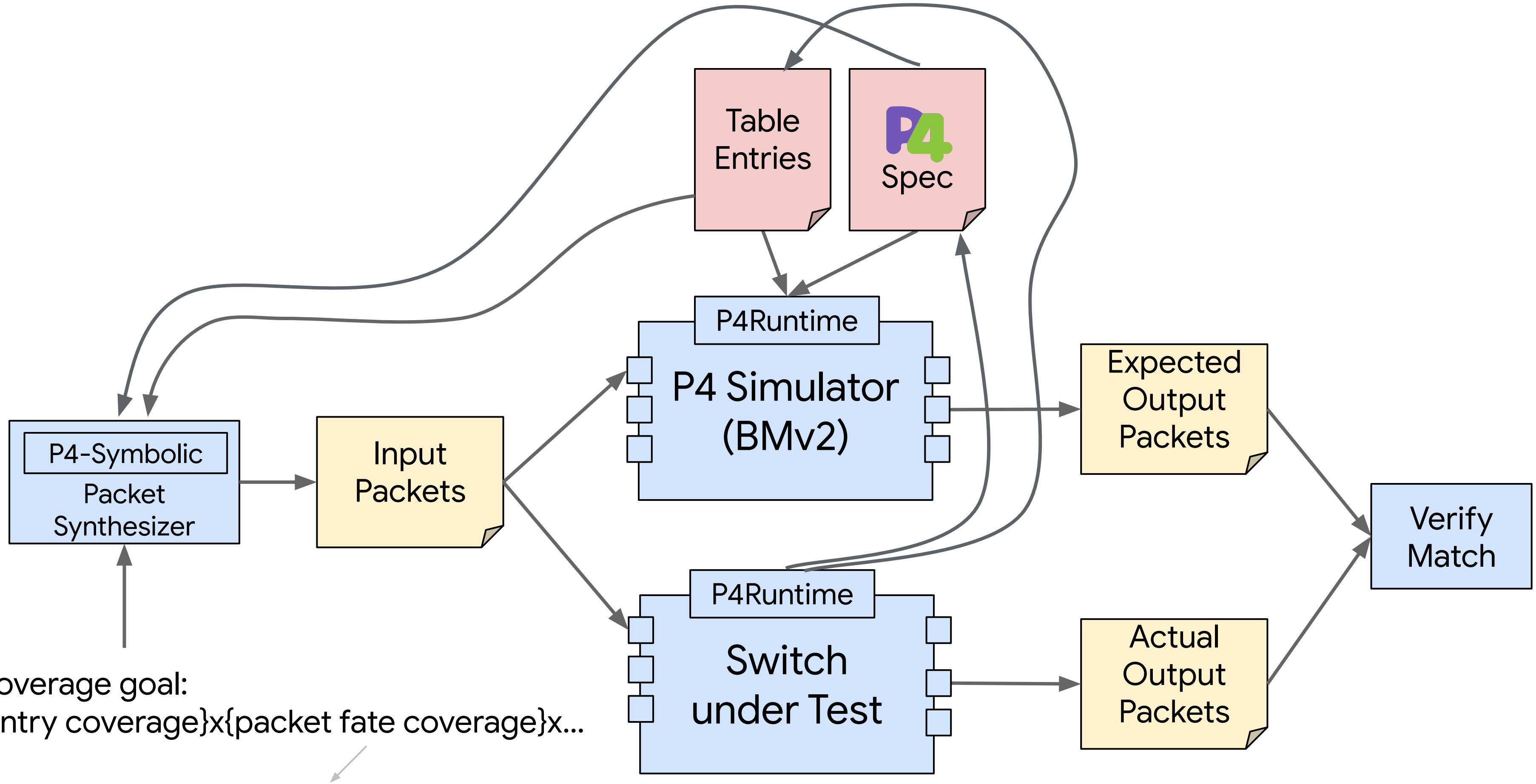


Table Entries

P4 Spec

P4-Symbolic

Packet Synthesizer

Input Packets

P4Runtime

P4 Simulator (BMv2)

P4Runtime

Switch under Test

Expected Output Packets

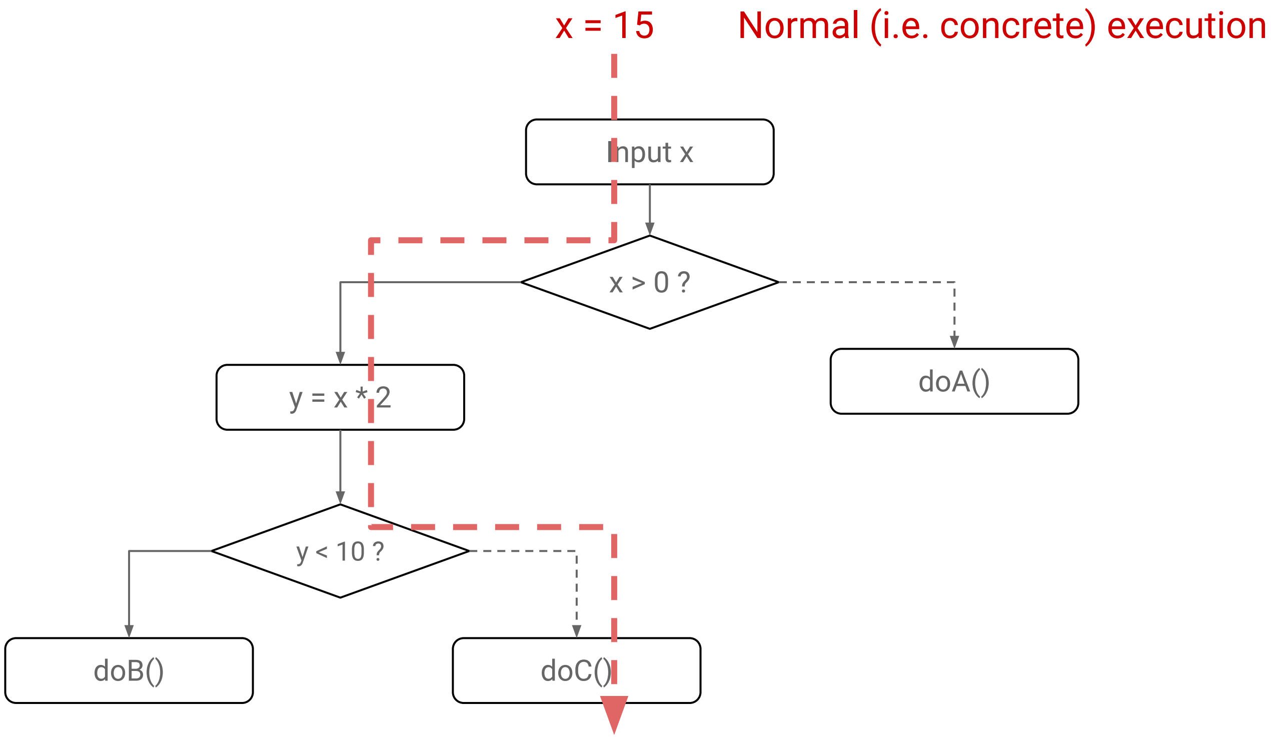
Actual Output Packets

Verify Match

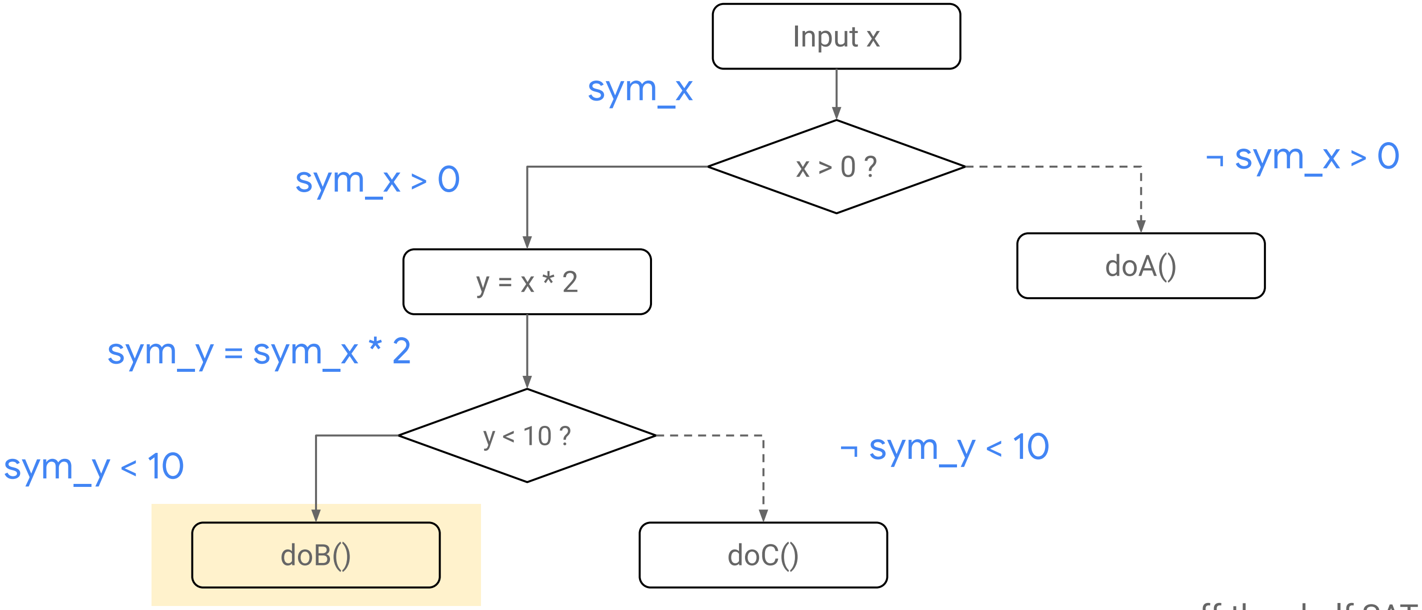
Coverage goal:
{entry coverage}x{packet fate coverage}x...

E.g. drop, forward

Symbolic Execution



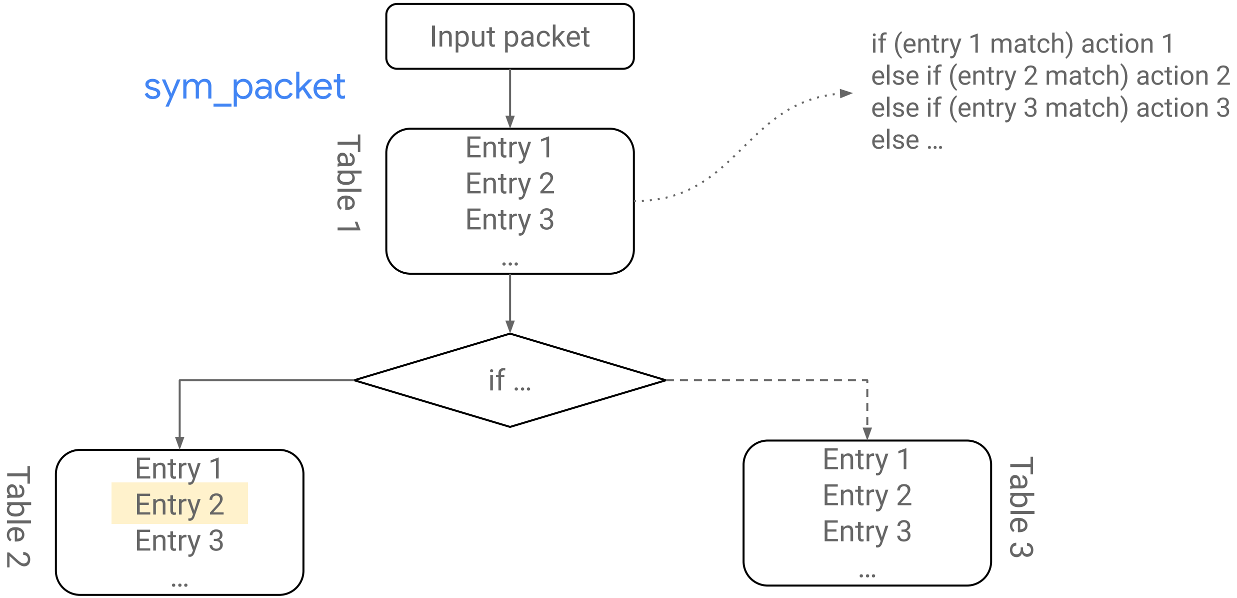
Symbolic Execution



Solve(sym_x > 0 ∧ sym_y = sym_x * 2 ∧ sym_y < 10) → off-the-shelf SAT/SMT solver (e.g. Z3)

E.g. solution: {sym_x = 1, sym_y = 2}

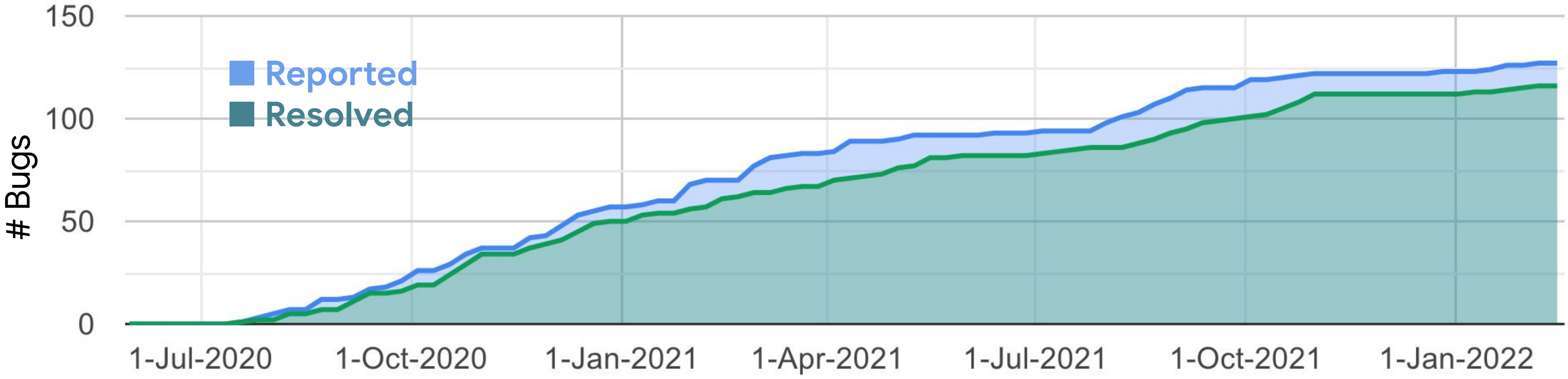
Symbolic Execution in P4



if (entry 1 match) action 1
else if (entry 2 match) action 2
else if (entry 3 match) action 3
else …

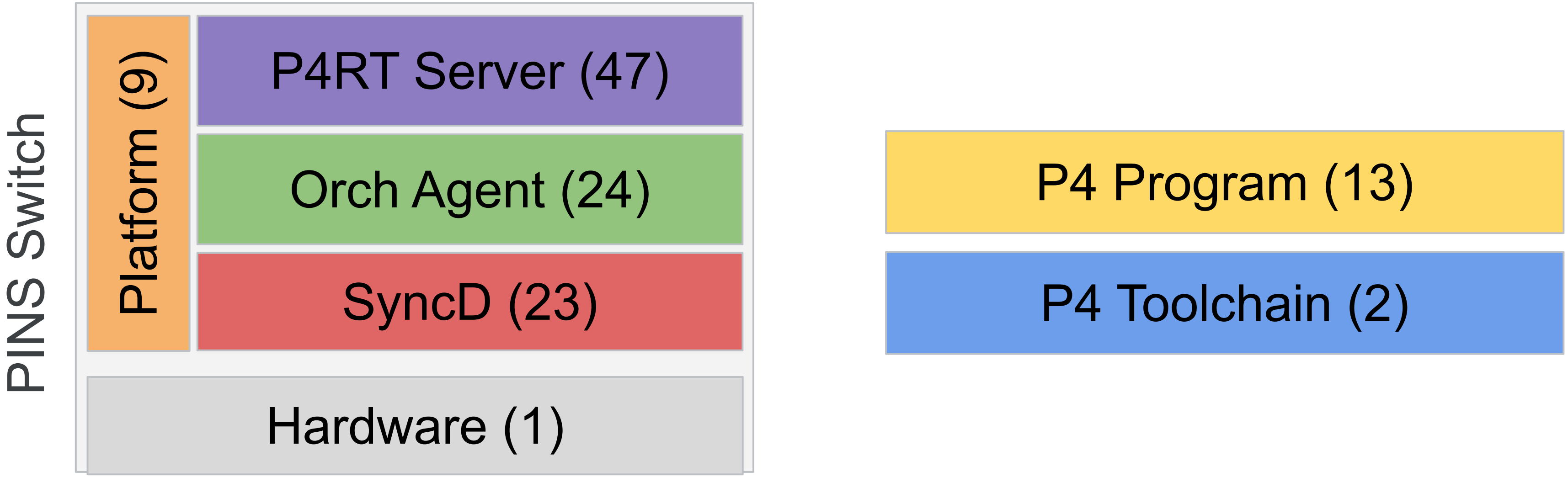
Experience

P4-BAR finds relevant bugs...



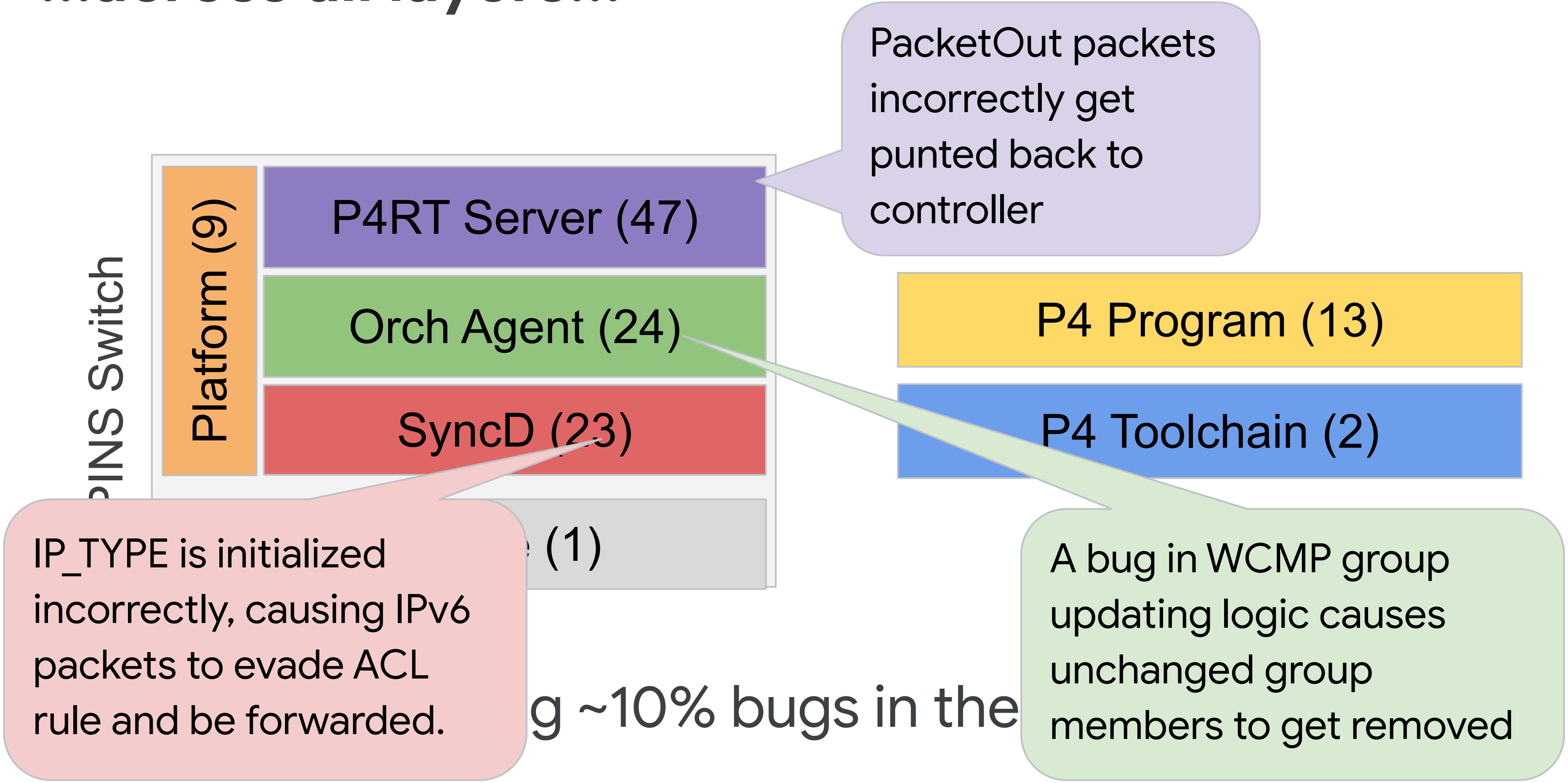
130+ bugs found in ~18 months
(and X bugs prevented)

...across all layers...



Including ~10% bugs in the P4 spec/toolchain

...across all layers...



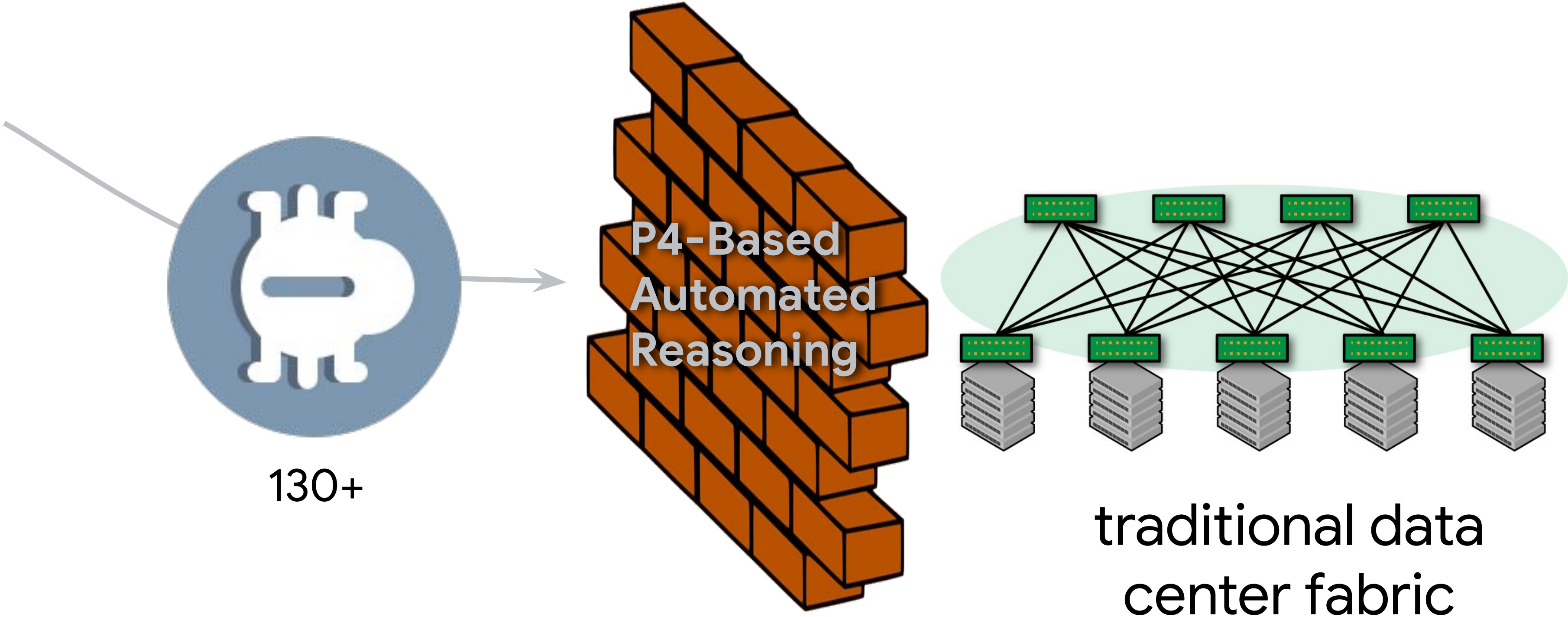
PacketOut packets incorrectly get punted back to controller

A bug in WCMP group updating logic causes unchanged group members to get removed

IP_TYPE is initialized incorrectly, causing IPv6 packets to evade ACL rule and be forwarded.

g ~10% bugs in the

...with only 4 bugs* escaping to fabric-level testing



130+

P4-Based Automated Reasoning

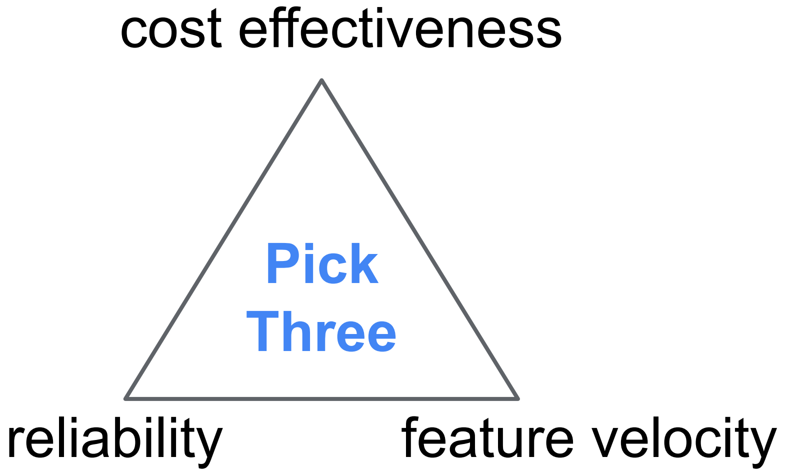
traditional data center fabric integration testing

*in the switch forwarding plane

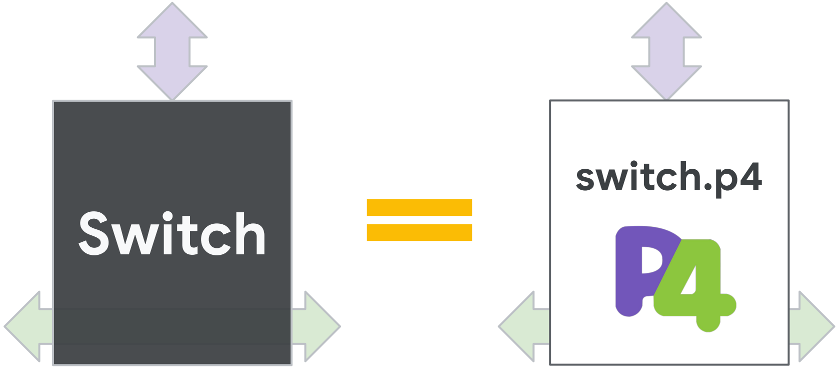
Wrapping Up

Conclusion

- An emerging paradigm: **P4 programs as formal specifications**
- Resolves tension between **velocity** and **reliability**
- Allows deriving tests automatically
  - comprehensive coverage
  - effortless evolution



Beyond finding bugs



- **Precise requirements** for vendors
- **Guaranteed switch behavior** for SDN controllers
  - no incidental differences between models/vendors
- **Switch & network simulation** for testing/development
- **Network verification** based on P4 model