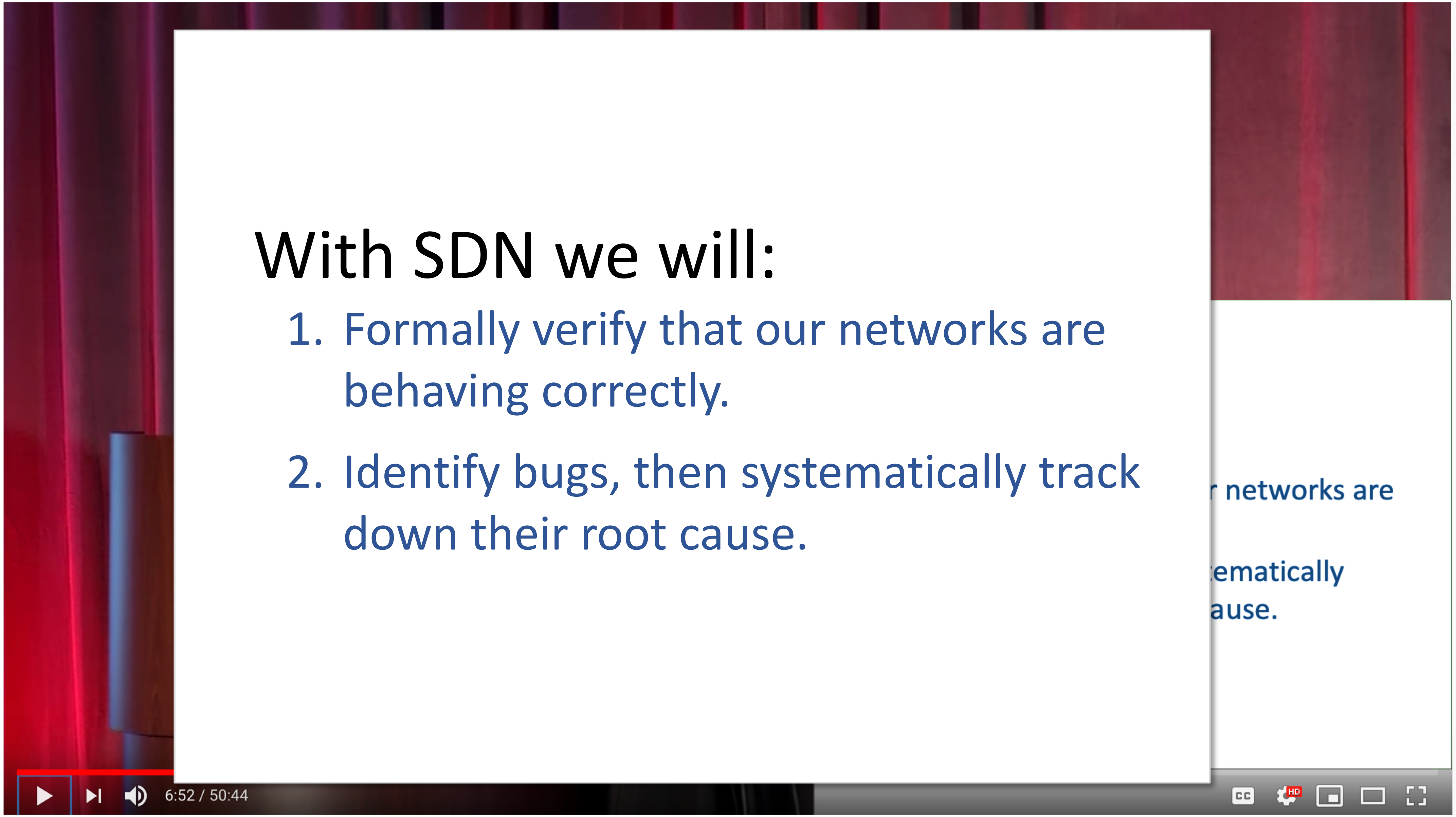
SDN

Phase 3: Getting the humans out of the way

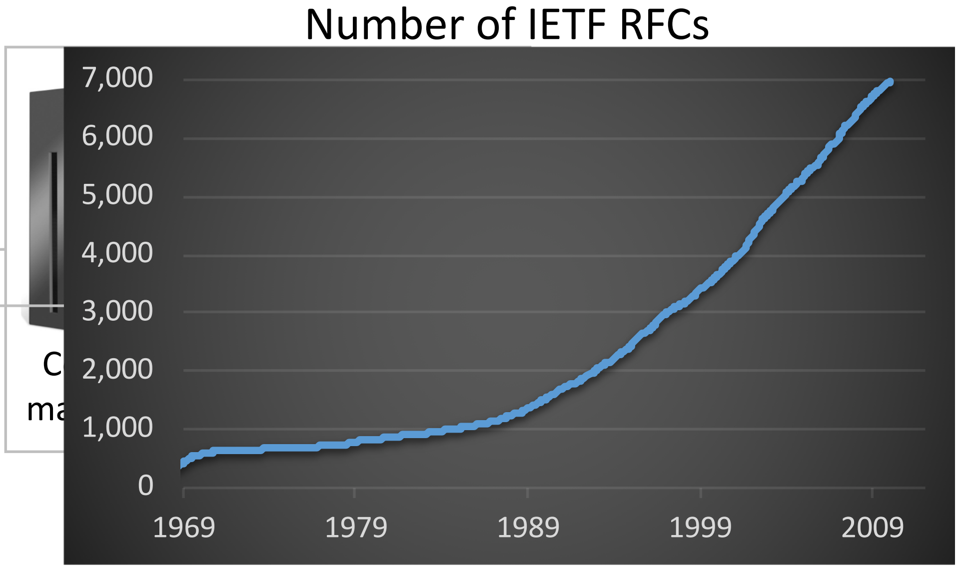
Nick McKeown

Stanford University

“Making SDNs Work” ONS 2012



With: Peyman Kazemian, George Varghese, James Zeng, David Erickson, Brandon Heller, Nikhil Handigol



"closed and proprietary"
"proliferation of standards"
"barrier to entry"
"stranglehold by vendors"
"ossification"
"clean slate"
"4D, Ethane, etc"

2010 **2020** **2030**

"closed and proprietary"
"proliferation of standards"
"barrier to entry"
"stranglehold by vendors"
"ossification"
"clean slate"
"4D, Ethane, etc"

Open-source

Disaggregation

SDN

OpenFlow

NFV

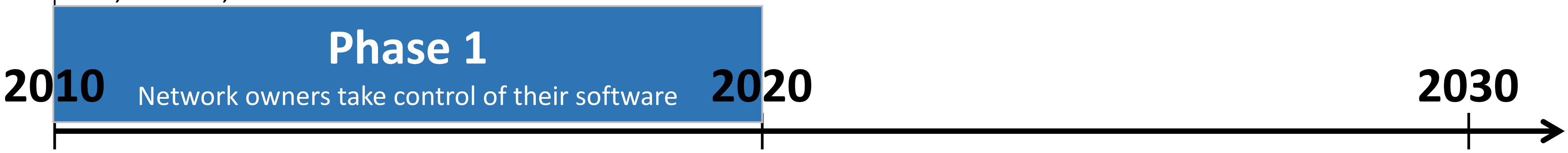
Network Virtualization

Programmable forwarding

Telemetry

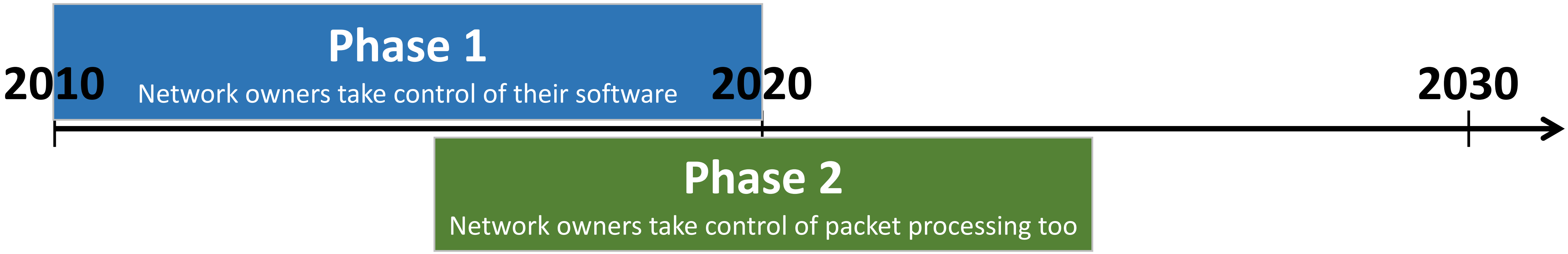
2010 **2020** **2030**

“closed and proprietary”
“proliferation of standards”
“barrier to entry”
“stranglehold by vendors”
“ossification”
“clean slate”
“4D, Ethane, etc”

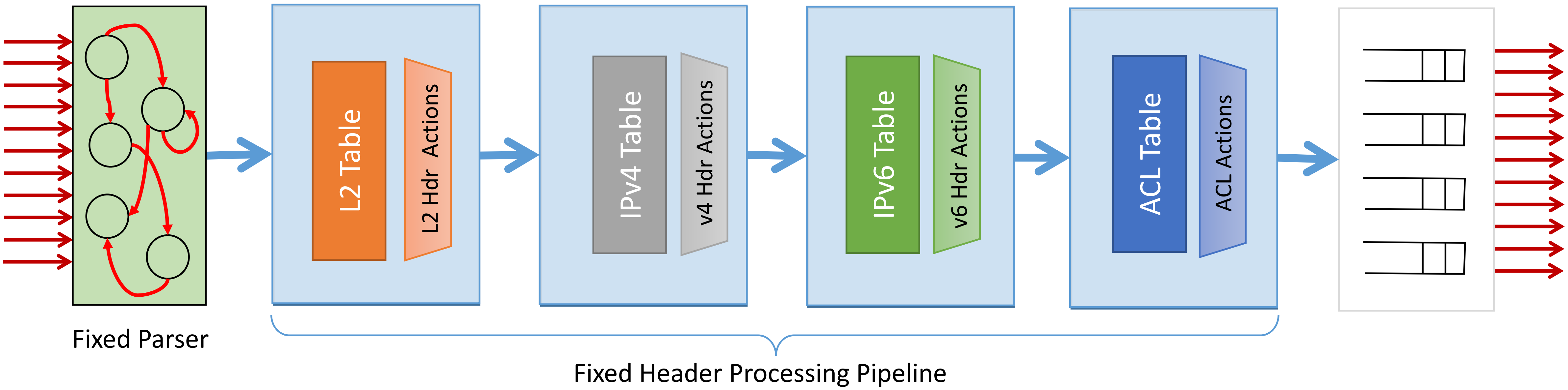


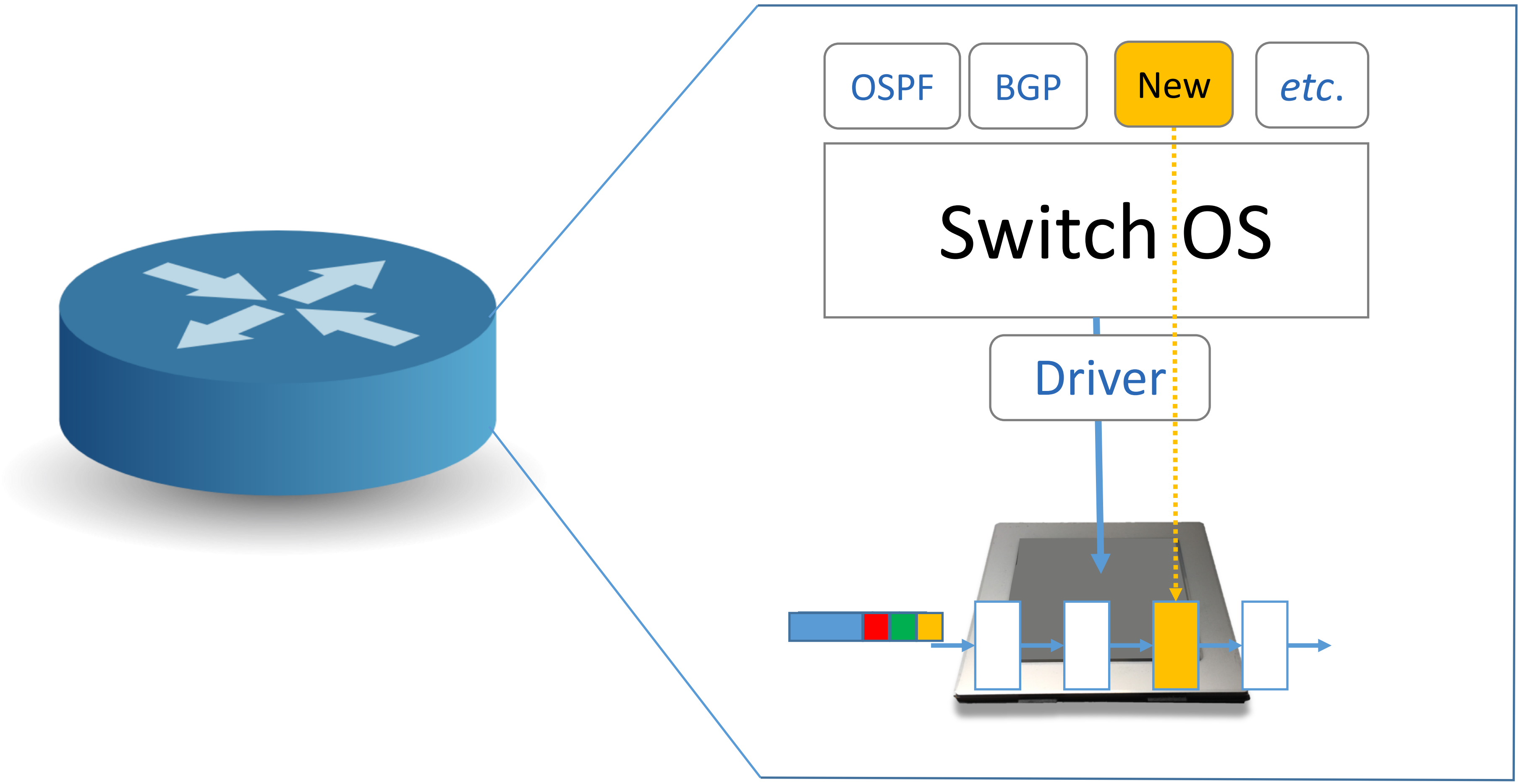
Now we take it for granted!

ONF has played a big role in this transformation:
ONOS, CORD, Trellis, SEBA, Stratum …


Phase 1
2010
Network owners take control of their software
2020
2030
Phase 2
Network owners take control of packet processing too

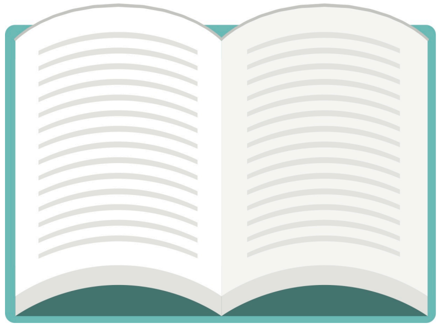
Switch with fixed function pipeline

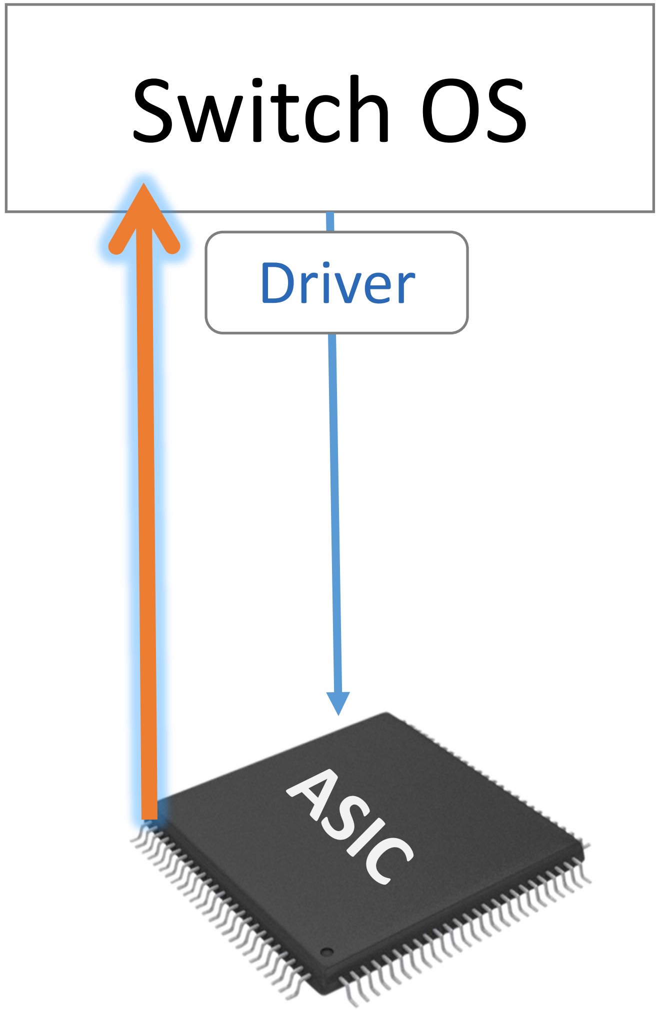



OSPF
BGP
New
etc.
Switch OS
Driver

Network systems were built "bottom-up"

"This is how I process packets …"





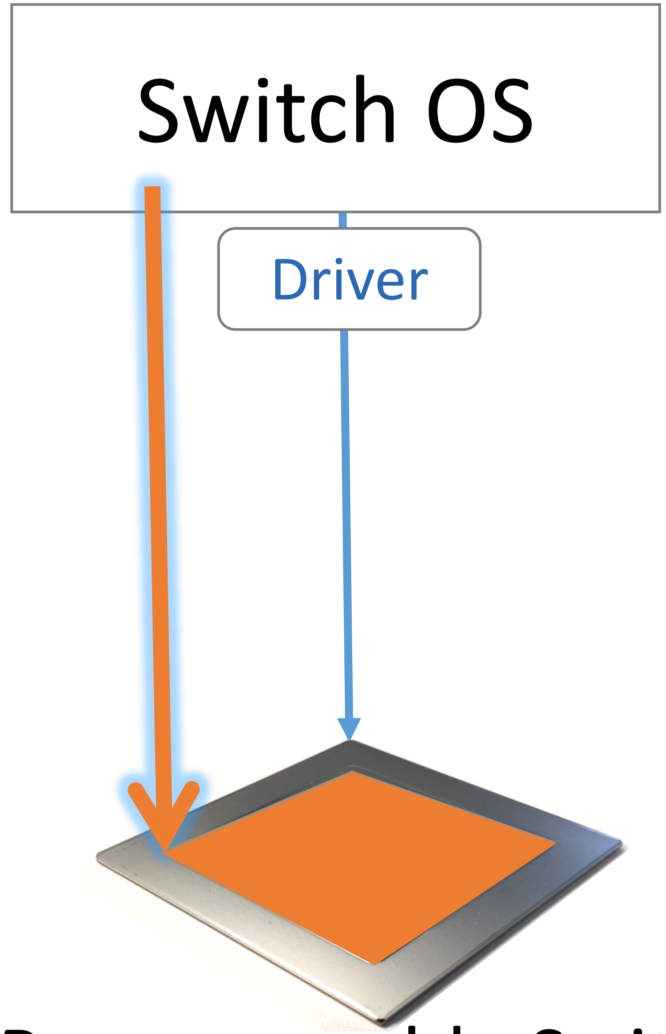
Fixed-function switch

Network systems starting to be built "top-down"

"This is precisely how you must process packets"

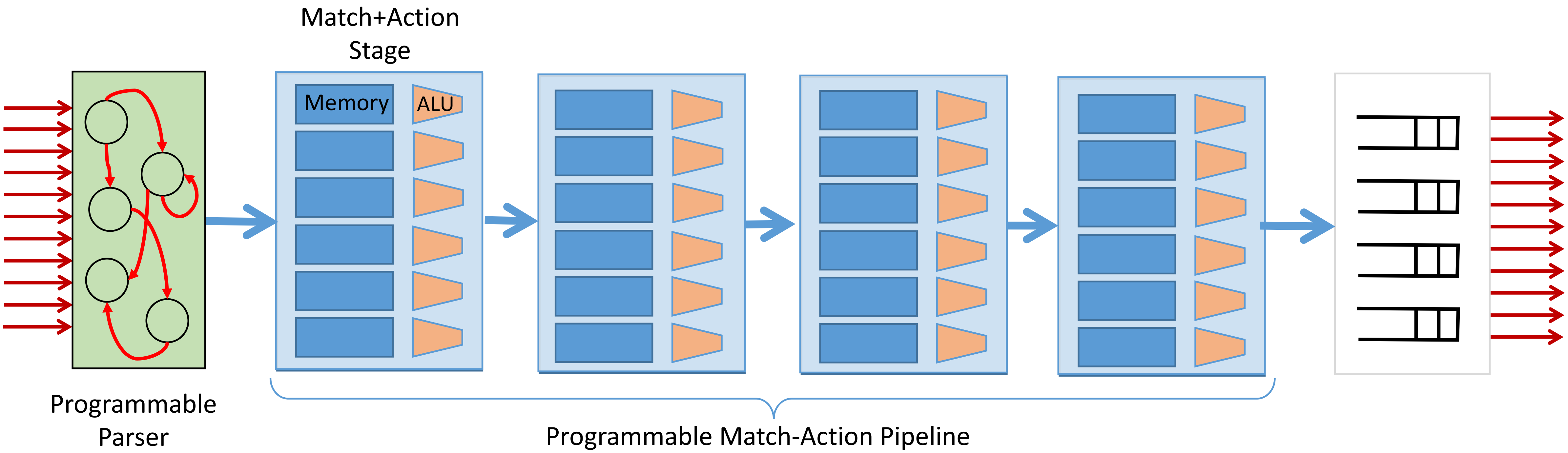
```
table int_table {
  reads {
    ip.protocol;
  }
  actions {
    export_queue_latency;
  }
}
```

```
action export_queue_latency (sw_id) {
  add_header(int_header);
  modify_field(int_header.kind, TCP_OPTION_INT);
  modify_field(int_header.len, TCP_OPTION_INT_LEN);
  modify_field(int_header.sw_id, sw_id);
  modify_field(int_header.q_latency,
               intrinsic_metadata.deq_timedelta);
  add_to_field(tcp.dataOffset, 2);
  add_to_field(ipv4.totalLen, 8);
  subtract_from_field(ingress_metadata.tcpLength,
                      12);
}
```



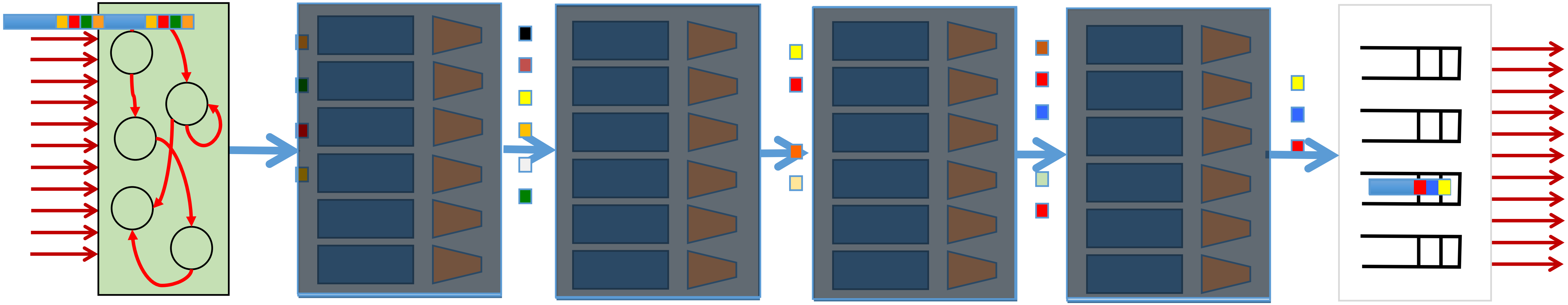
Programmable Switch

PISA: Protocol Independent Switch Architecture



Generalization of RMT [Sigcomm'13]

PISA: Protocol Independent Switch Architecture



Example P4 Program

Parser Program

```
parser parse_ethernet {
   extract(ethernet);
   return switch(ethernet.ethertype) {
      0x8100 : parse_vlan_tag;
      0x0800 : parse_ipv4;
      0x8847 : parse_mpls;
      default: ingress;
}
```

Header and Data Declarations

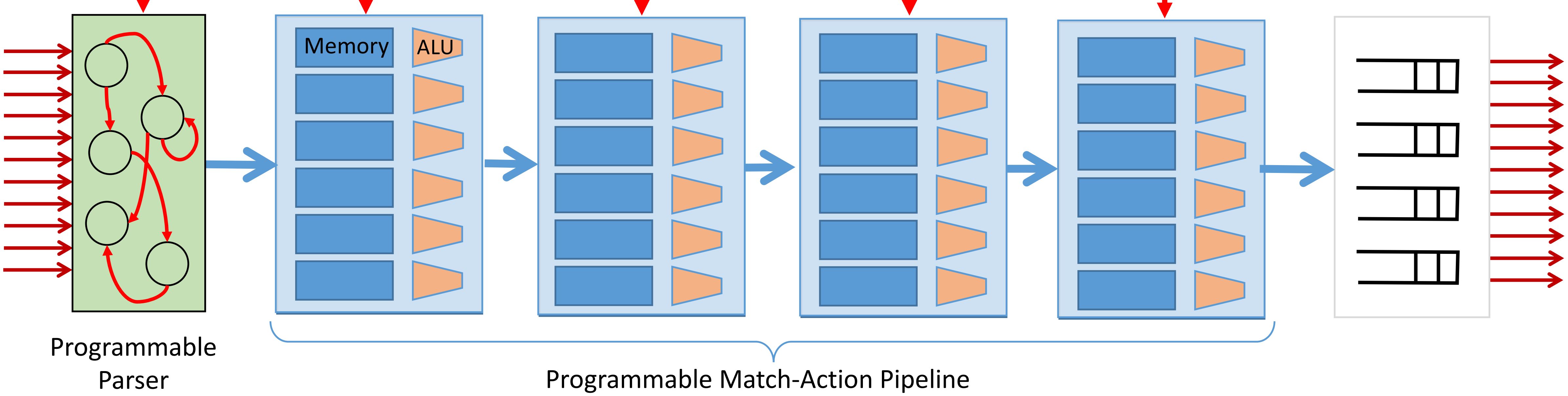
```
header_type  ethernet_t     { … }
header_type  l2_metadata_t { … }

header     ethernet_t     ethernet;
header     vlan_tag_t
vlan_tag[2];
metadata   l2_metadata_t l2_meta;
```

Tables and Control Flow

```
table port_table { … }

control ingress {
    apply(port_table);
    if (l2_meta.vlan_tags == 0) {
        process_assign_vlan();
    }
}
```



Why I devoted 5 years to programmable forwarding…

Programmable switch chips can have the same power, performance and cost as fixed function switches.

Beautiful new ideas are now owned by the programmer, not the chip designer.

Which means more innovation.

How do we know if a programmable switch chip has the same power, performance and cost as a fixed function switch chip?

Comparison

	P4 Programmable "Tofino"	Fixed Function
L2/L3 Throughput	6.4Tb/s	6.4Tb/s
Number of 100G Ports	64	64
Availability	Yes	Yes
Max Forwarding Rate	5.1B packets per sec	4.2B packets per sec
Max 25G/10G Ports	256/258	128/130
Programmability	Yes (P4)	No
Typical System Power draw	4.2W per port	5.3W per port
Large Scale NAT	Yes (100k)	No
Large scale stateful ACL	Yes (100k)	No
Large Scale Tunnels	Yes (192k)	No
Packet Buffer	Unified	Segmented
Segment Rtg/Bare Metal	Yes/Yes	No/No
LAG/ECMP Hash Algorithm	Full entropy, programmable	Hash seed, reduced entropy
ECMP	256 way	128 way
Telemetry and Analytics	Line-rate per flow stats	Sflow (Sampled)
Latency	Under 400 ns	450 ns

Otherwise, both systems are identical:

- # of Ports
- CPU
- Power Supplies

SDN, Part 2: Programmable Forwarding

How it gets used

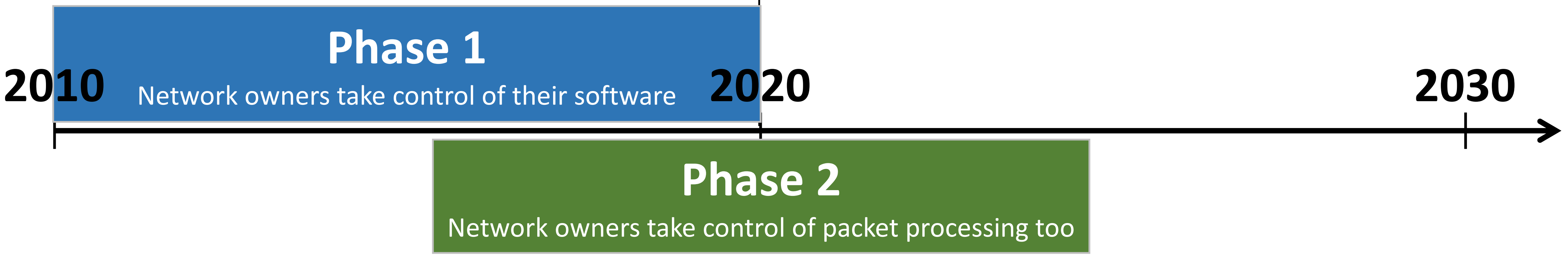
1. Reducing complexity
2. Adding new features to the network
3. Telemetry

P4.org

- Now part of ONF
- Lots of activities and workshops: get involved!
- P4-16 stable. Device independent: Switches, NICs, FPGAs, vSwitches
- P4Runtime part of Stratum, launched this week

A cast of many, led by: Nate Foster (Cornell), Amin Vahdat (Google), Jennifer Rexford (Princeton), Chang Kim (Barefoot)

The network (switch, router, NIC, firewall, 5G…) is now a programmable platform.
Top down, including the control plane and the forwarding plane.




2020
2030

Extrapolating to 2030

1. NICs, Switches, vSwitches, stacks will have been programmable for 10 years.
2. We will think of a network as a programmable platform.
   Behavior described at the top.
   Then partitioned, compiled and run across elements.
3. Every large network will work slightly differently, programmed and tailored locally.

Extrapolating to 2030

4. We will no longer think in terms of protocols. Instead, we will think in terms of software. All functions and “protocols” will have migrated up and out of hardware into software.
5. Networking students will learn how to program a network top-down, as a distributed computing platform. Protocols will be described in quaint historical terms.
6. “Routing” and “Congestion control” will be programs, partitioned across the end-to-end system by a compiler.

If we want to get the humans out of the way, what else do we need?

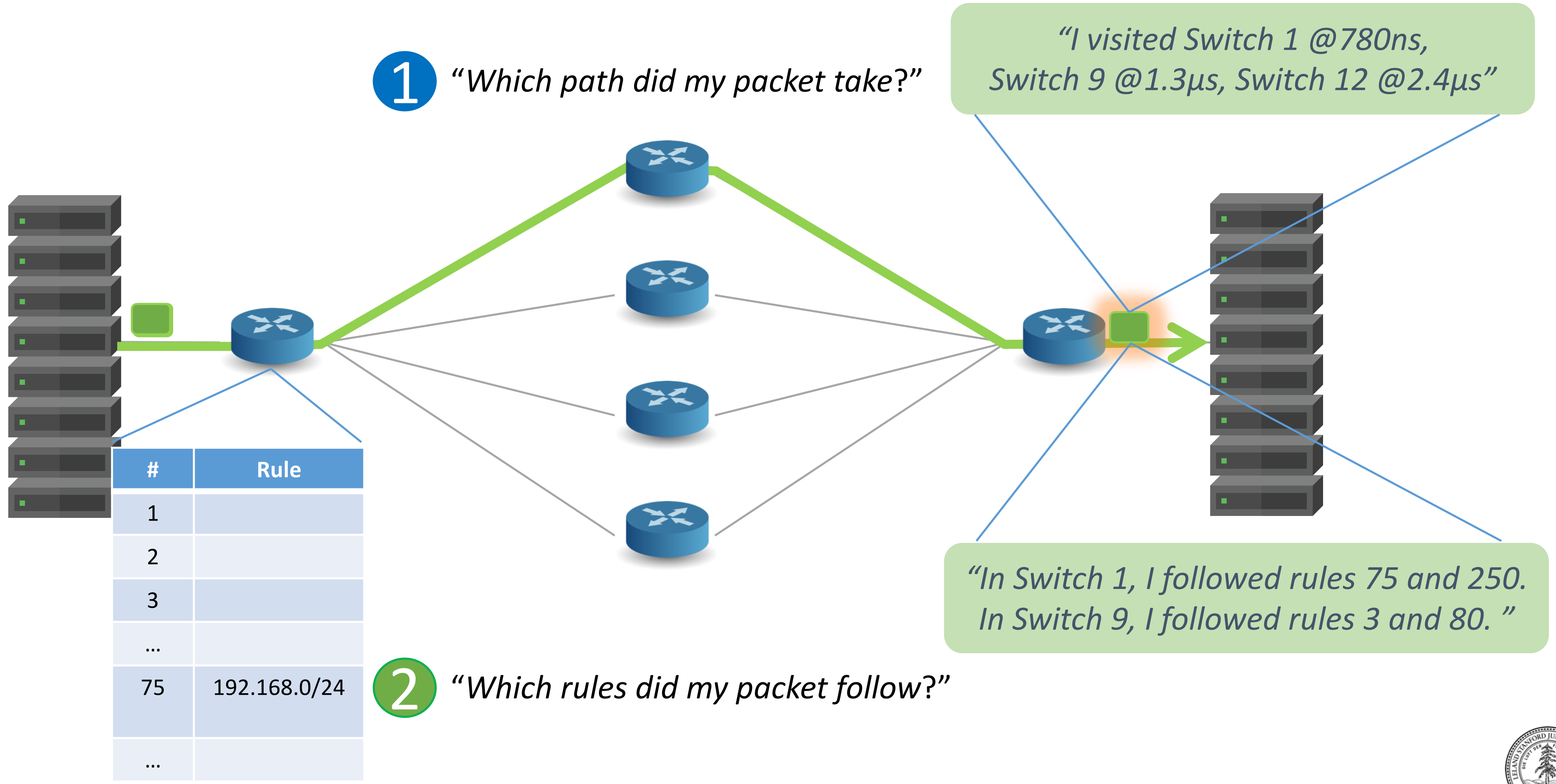
Three pieces

1. The ability to observe packets, network state and code, in real-time.
2. The ability to generate new control and forwarding behaviors, on the fly, to correct errors.
3. The ability to verify newly generated code and deploy it quickly.

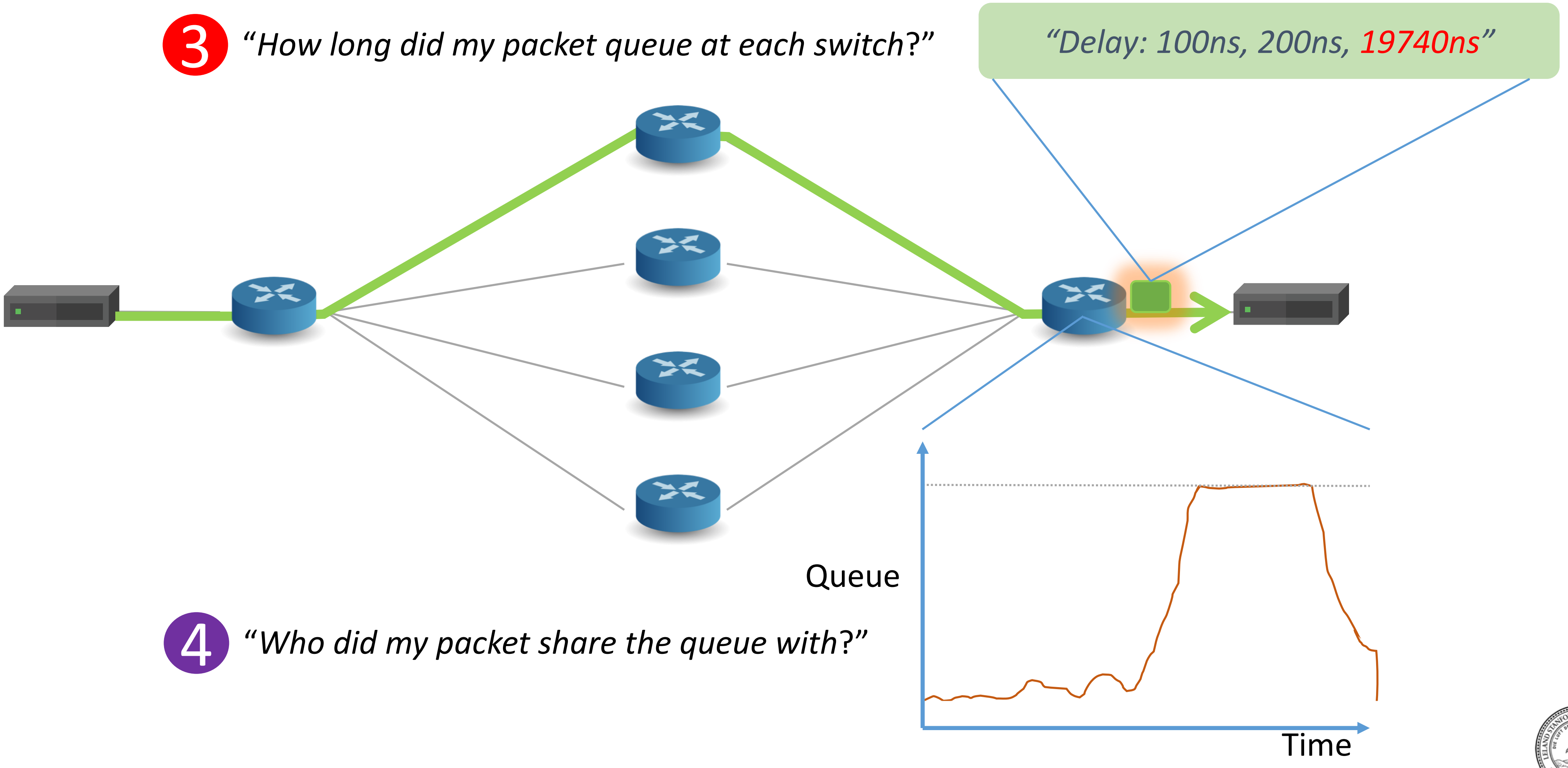
Observing packets

Per-packet telemetry is already starting to happen

Today, basic information is hard to find

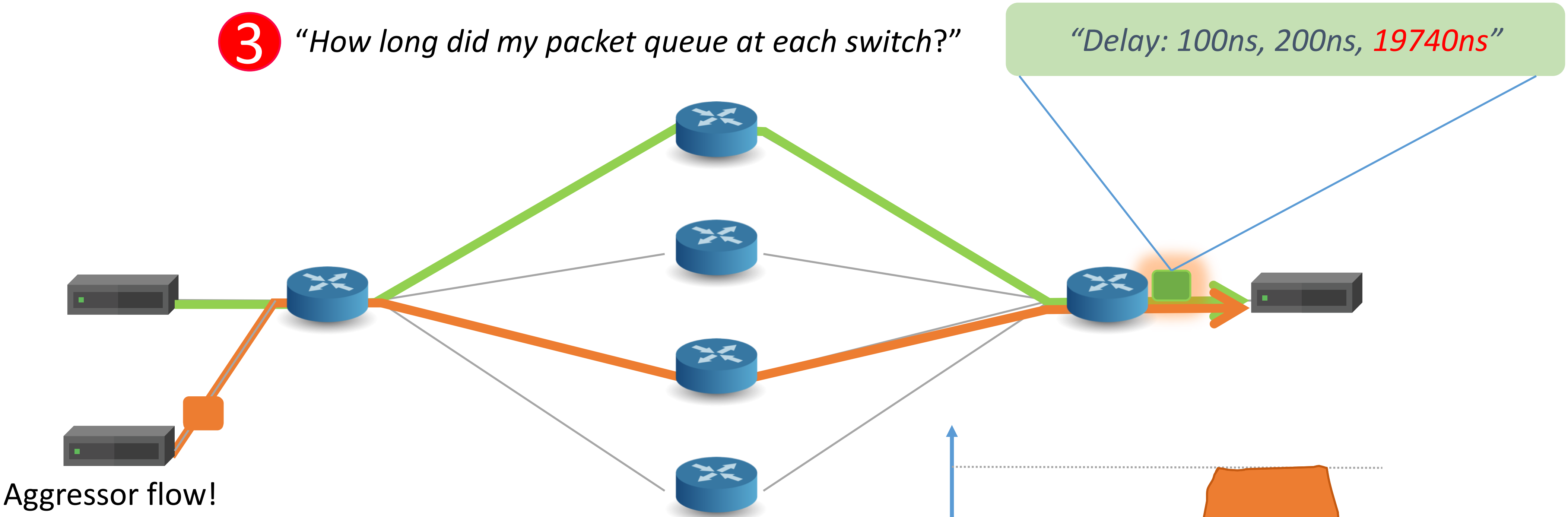


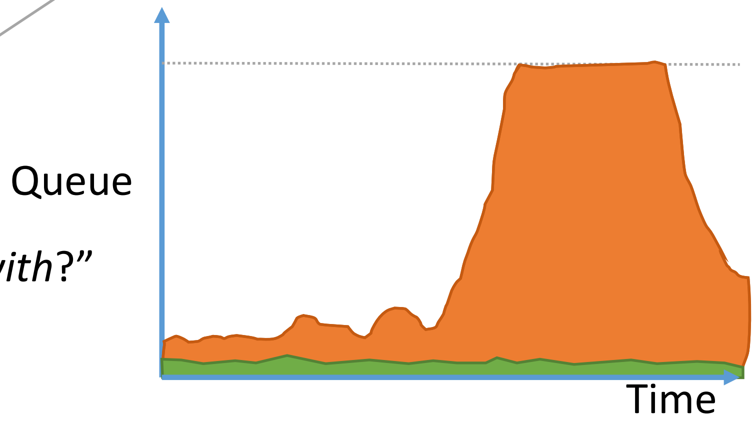
3 *"How long did my packet queue at each switch?"*



4 *"Who did my packet share the queue with?"*

3 *"How long did my packet queue at each switch?"*

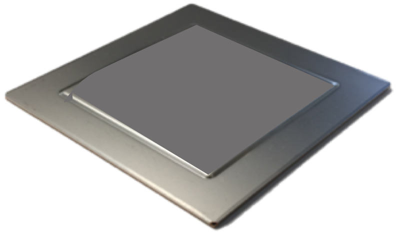




4 *"Who did my packet share the queue with?"*

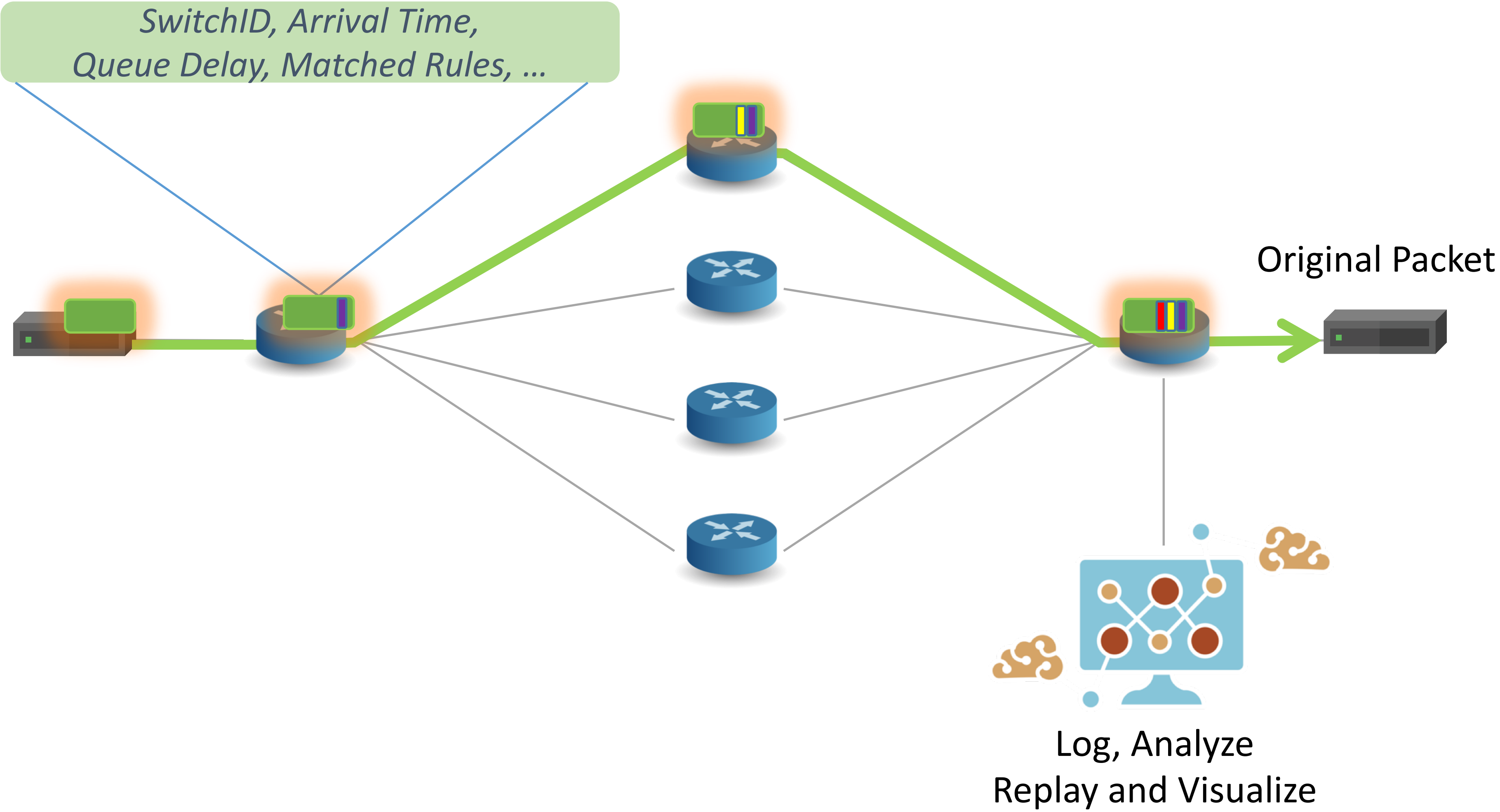
Today, basic information is hard to find

1. *“Which path did my packet take?”*
2. *“Which rules did my packet follow?”*
3. *“How long did it queue at each switch?”*
4. *“Who did it share the queues with?”*



With P4 + INT we can answer all four questions for the first time.
At full line rate. Without generating additional packets.

INT: In-band Network Telemetry



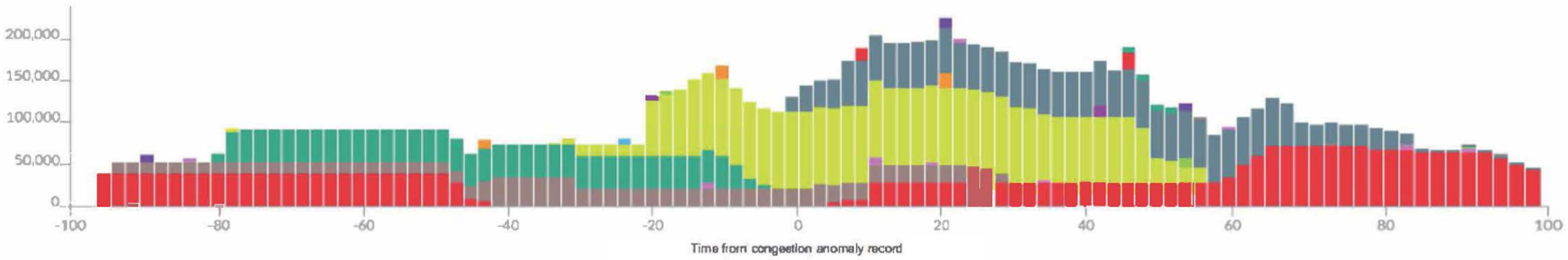
+ SONATA [Sigcomm '18], Sketches [Sigcomm '12] …

Viewing Microbursts (to the nanosecond)

Anomaly Records

Timestamp	Switch Id	Queue
July 25, 2017 - 18:17:51.513 UTC		

Queue Occupancy Over Time (bytes)



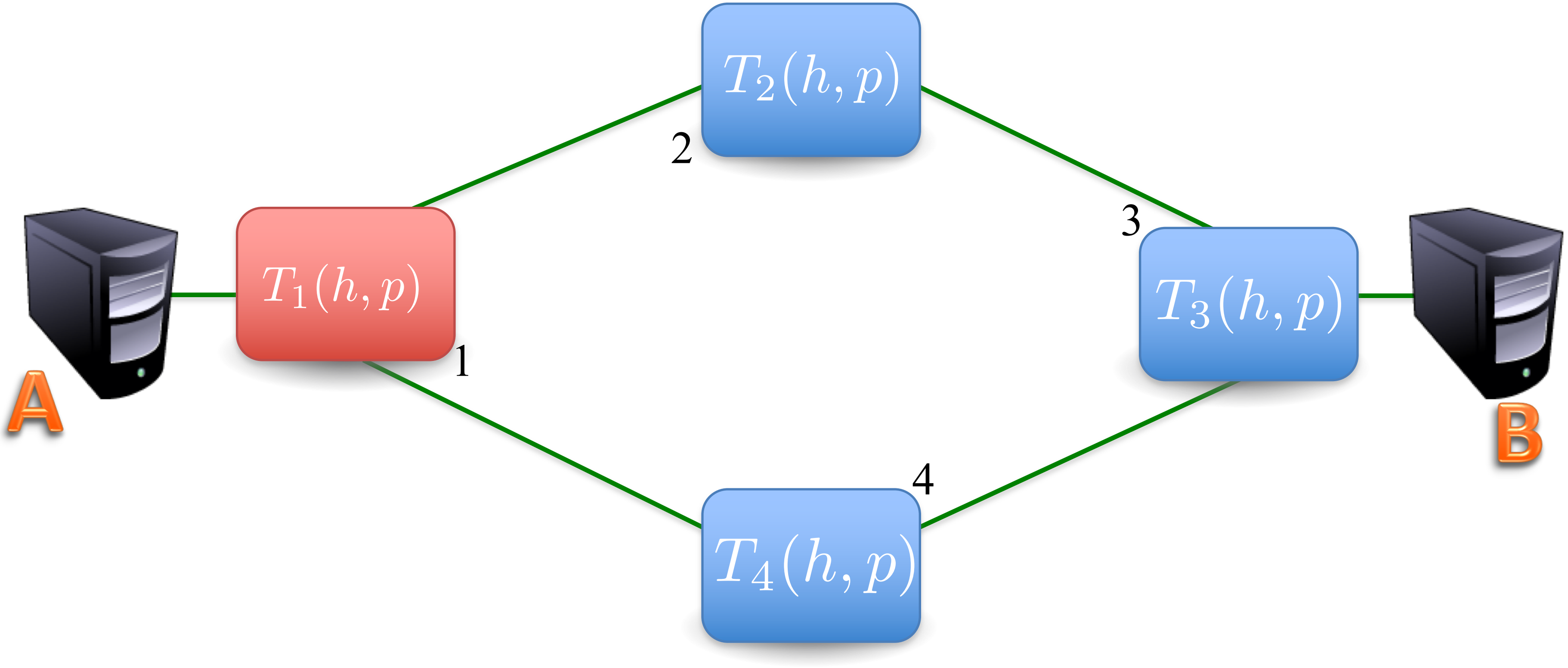
17 Affected Flows

Flow	kB in Queue	% of Queue Buildup	Packet Drops
10.32.2.2:46380 -> 10.36.1.2:5101 TCP	3282	29	0
10.32.2.2:46374 -> 10.36.1.2:5101 TCP	3073.5	27	25
10.32.2.2:46386 -> 10.36.1.2:5101 TCP	2092.5	18	27
10.32.2.2:46388 -> 10.36.1.2:5101 TCP	1456.5	13	0
10.32.2.2:46390 -> 10.36.1.2:5101 TCP	1227	11	36
10.32.2.2:46372 -> 10.36.1.2:5101 TCP	45	0	0
10.32.2.2:46392 -> 10.36.1.2:5101 TCP	37.5	0	39
10.35.1.2:34256 -> 10.36.1.2:5102 TCP	34.5	0	0

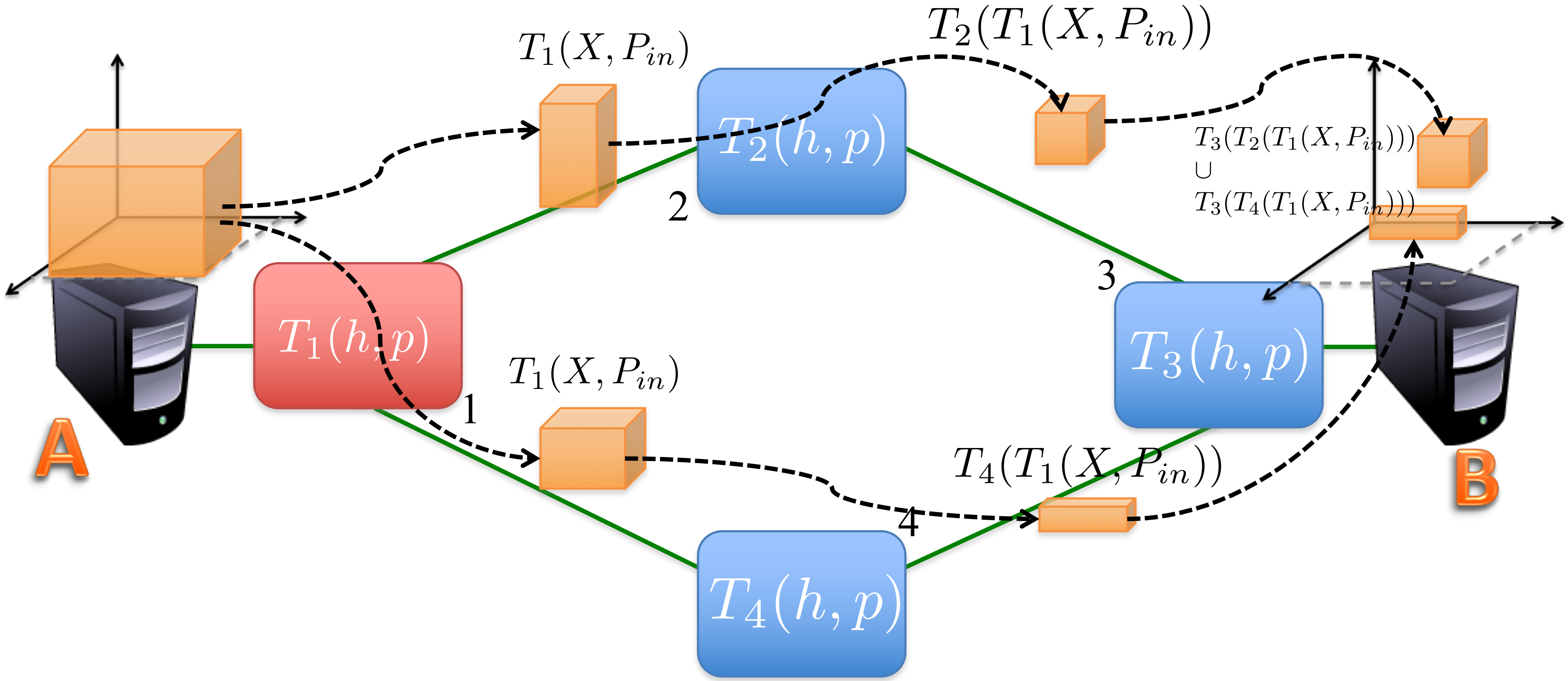
Three pieces

1. The ability to observe packets, network state and code, in real-time.
2. The ability to generate new control and forwarding behaviors, on the fly, to correct errors.
3. The ability to verify newly generated code and deploy it quickly.

Header Space Analysis



Example: Can A talk to B?



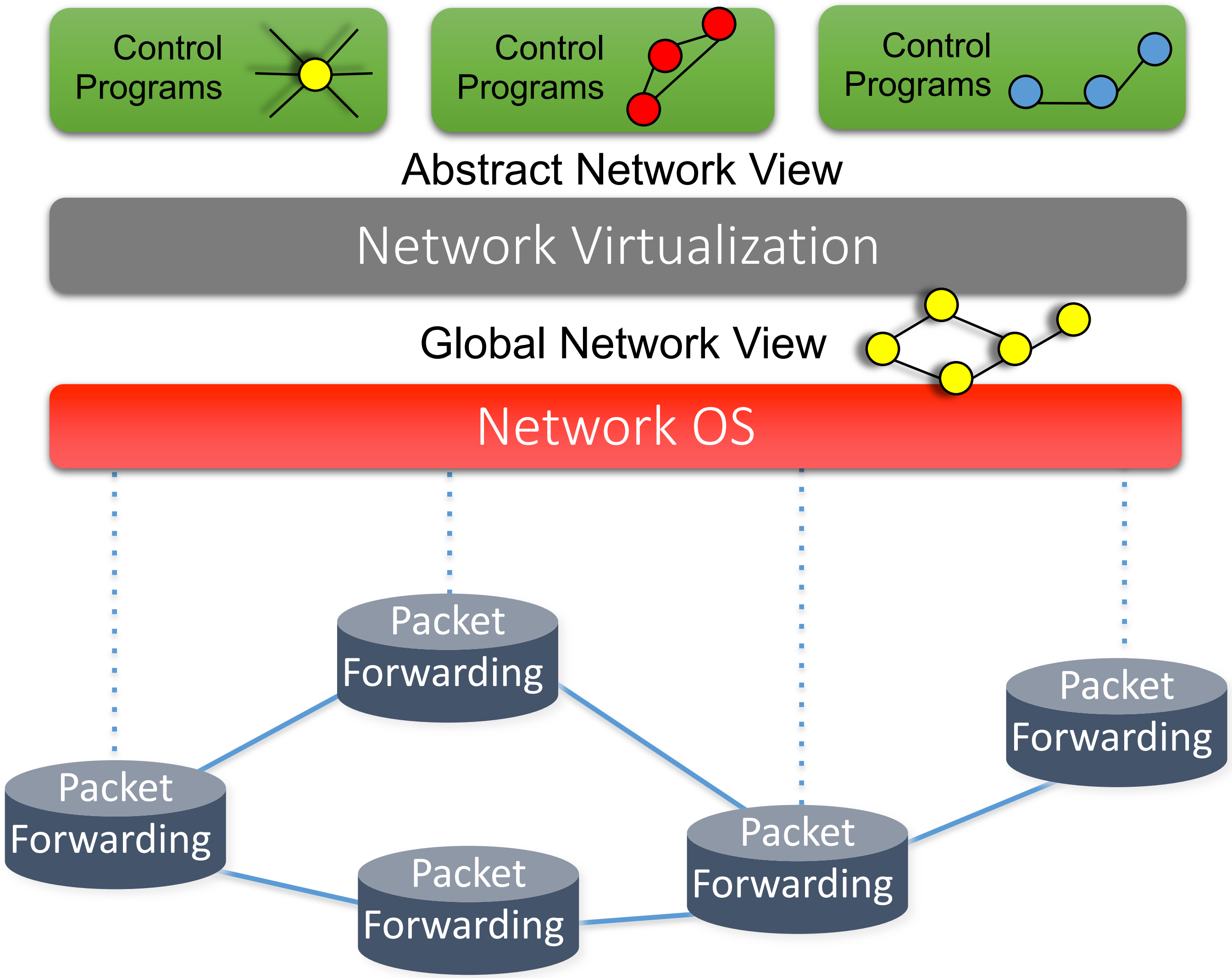
Three pieces

1. The ability to observe packets, network state and code, in real-time.
2. The ability to generate new control and forwarding behaviors, on the fly, to correct errors.

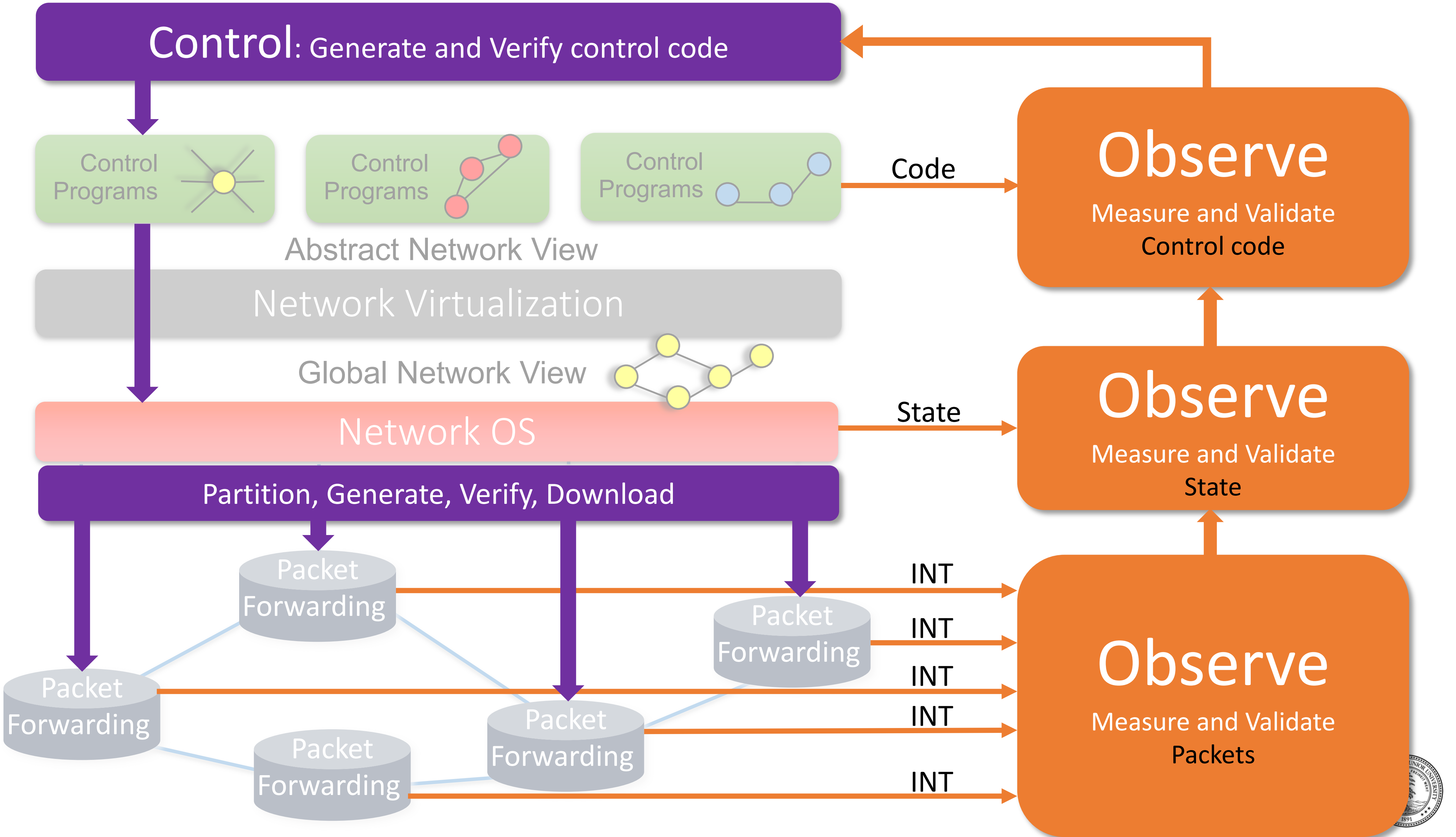
?

3. The ability to verify newly generated code and deploy it quickly.

Software Defined Network (SDN)



Software Defined Network (SDN)



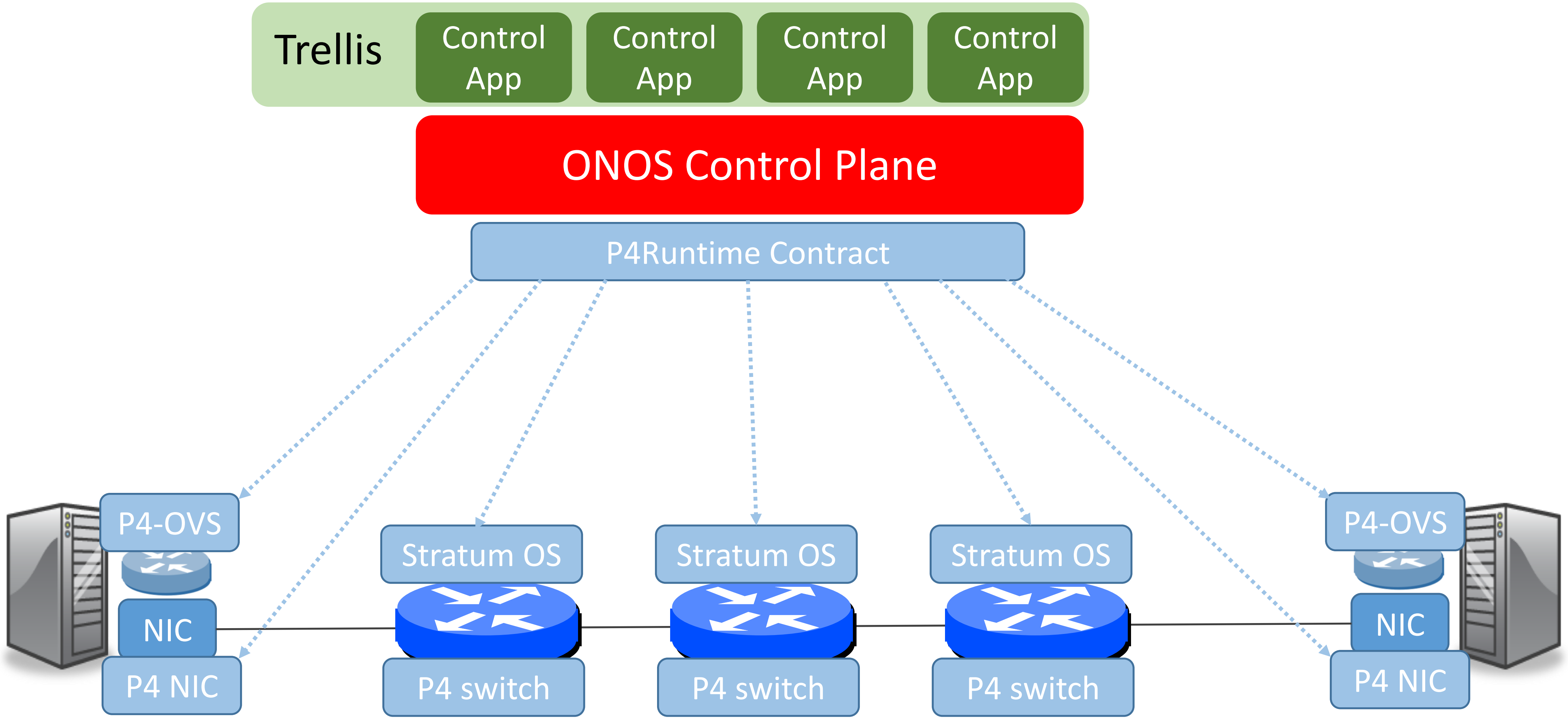
Getting humans out of the way

SDN with Verifiable Closed-Loop Control

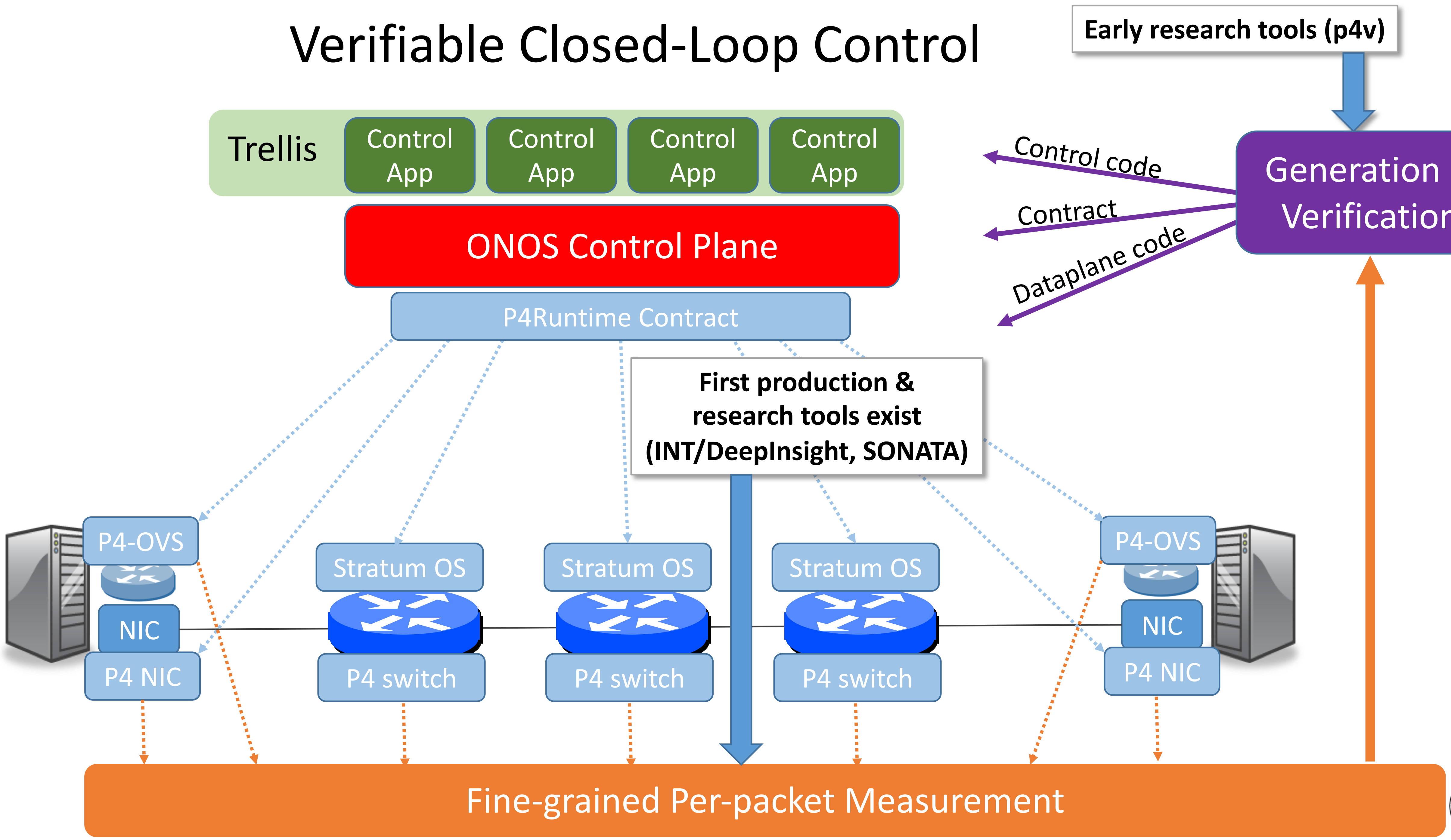
Network owners and operators will use
fine-grain measurement and formal verification
to automate network control at scale.

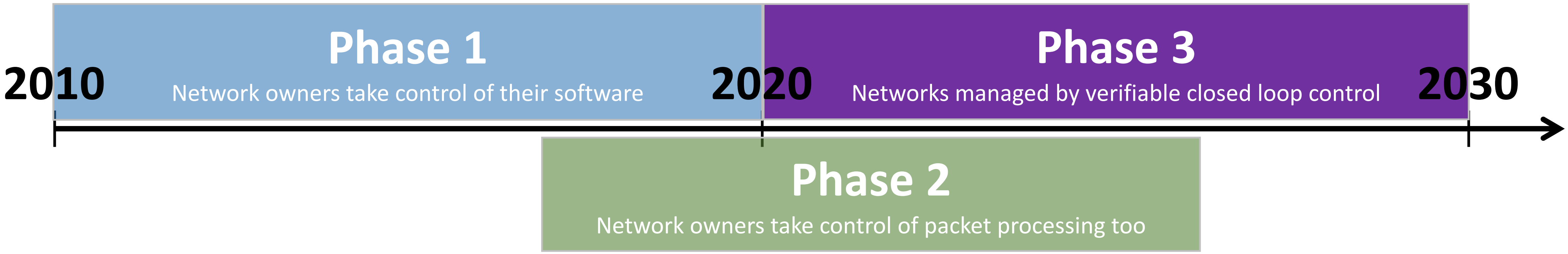
Joint work with: Nate Foster (Cornell), Guru Parulkar (ONF), Larry Peterson (ONF), Jennifer Rexford (Princeton)

ONF Open-source Software Today



Verifiable Closed-Loop Control




Phase 1
Network owners take control of their software
Phase 3
Networks managed by verifiable closed loop control
2010
2020
2030
Phase 2
Network owners take control of packet processing too

"Making SDNs Work" ~~ONS 2012~~ ONF Connect 2019

With SDN we will:

1. Formally verify that our networks are behaving correctly.
2. Identify bugs, then systematically track down their root cause.
3. Measure and validate correctness, then generate and verify code fix. Download to correct the bug.
4. Goto beach….?