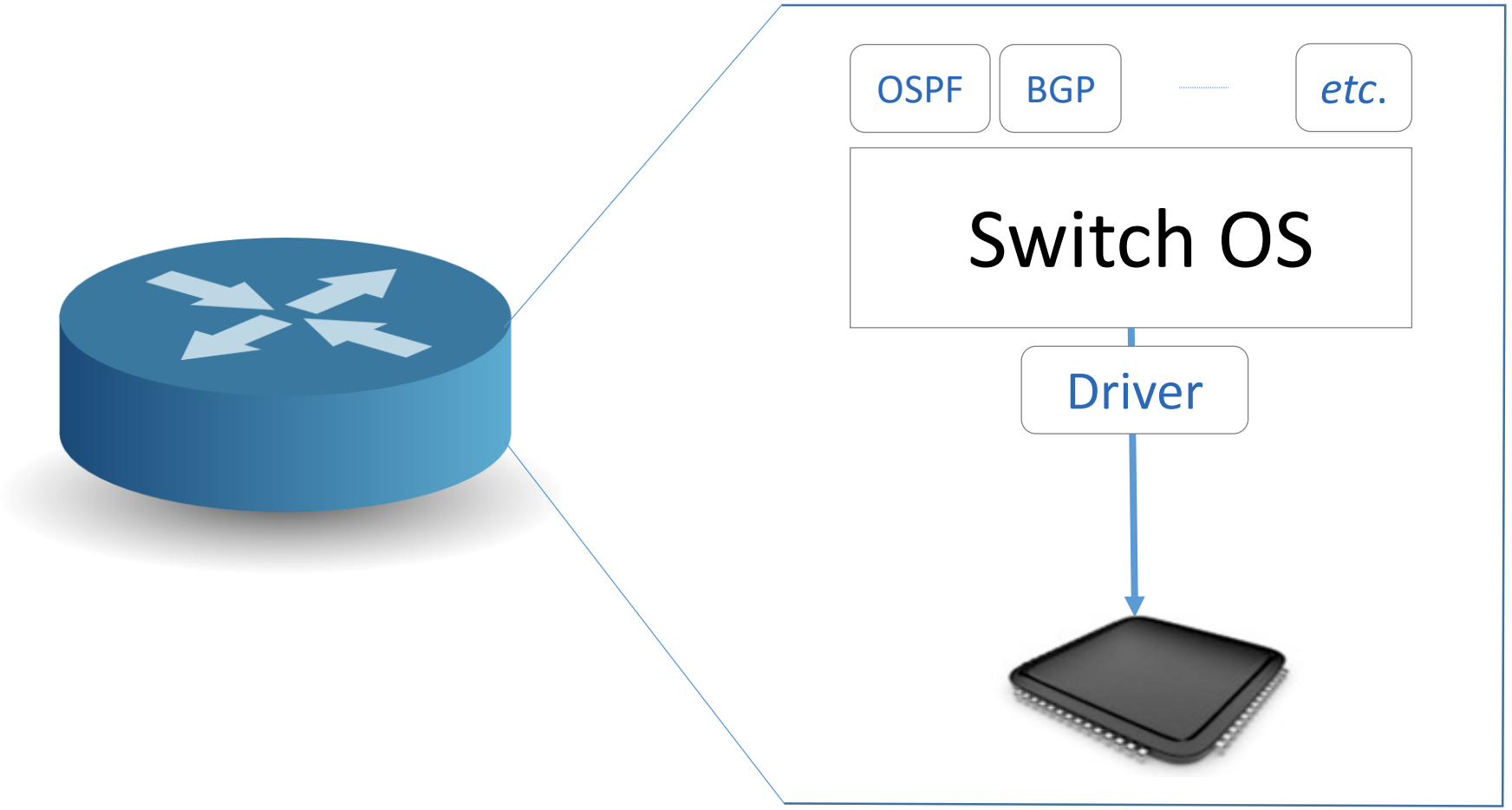
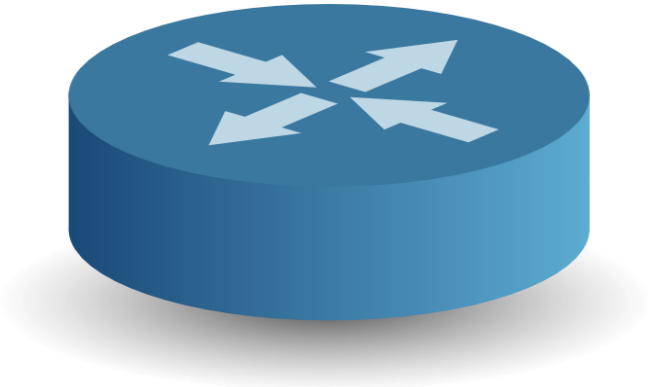
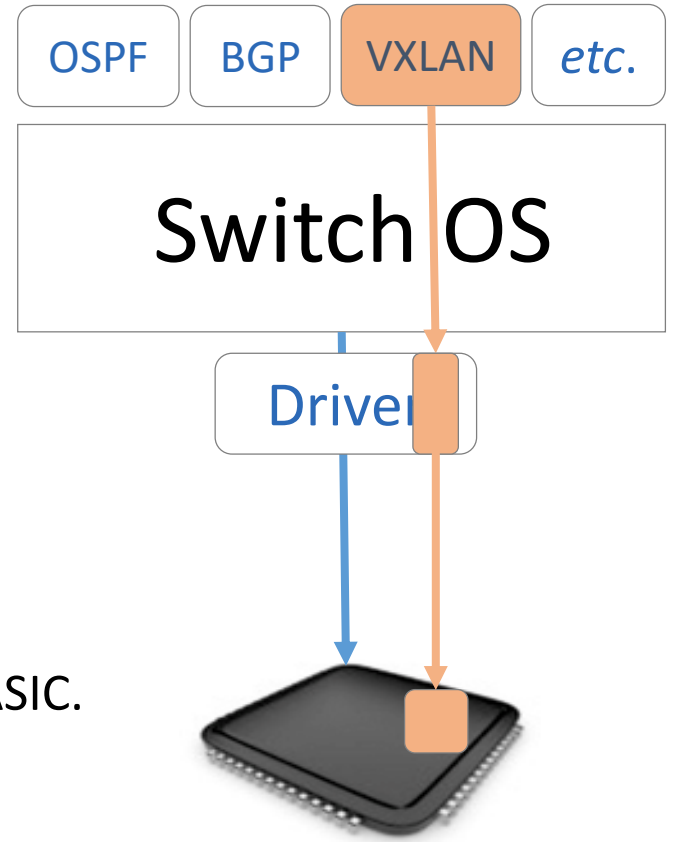
Programmable Forwarding

Nick McKeown

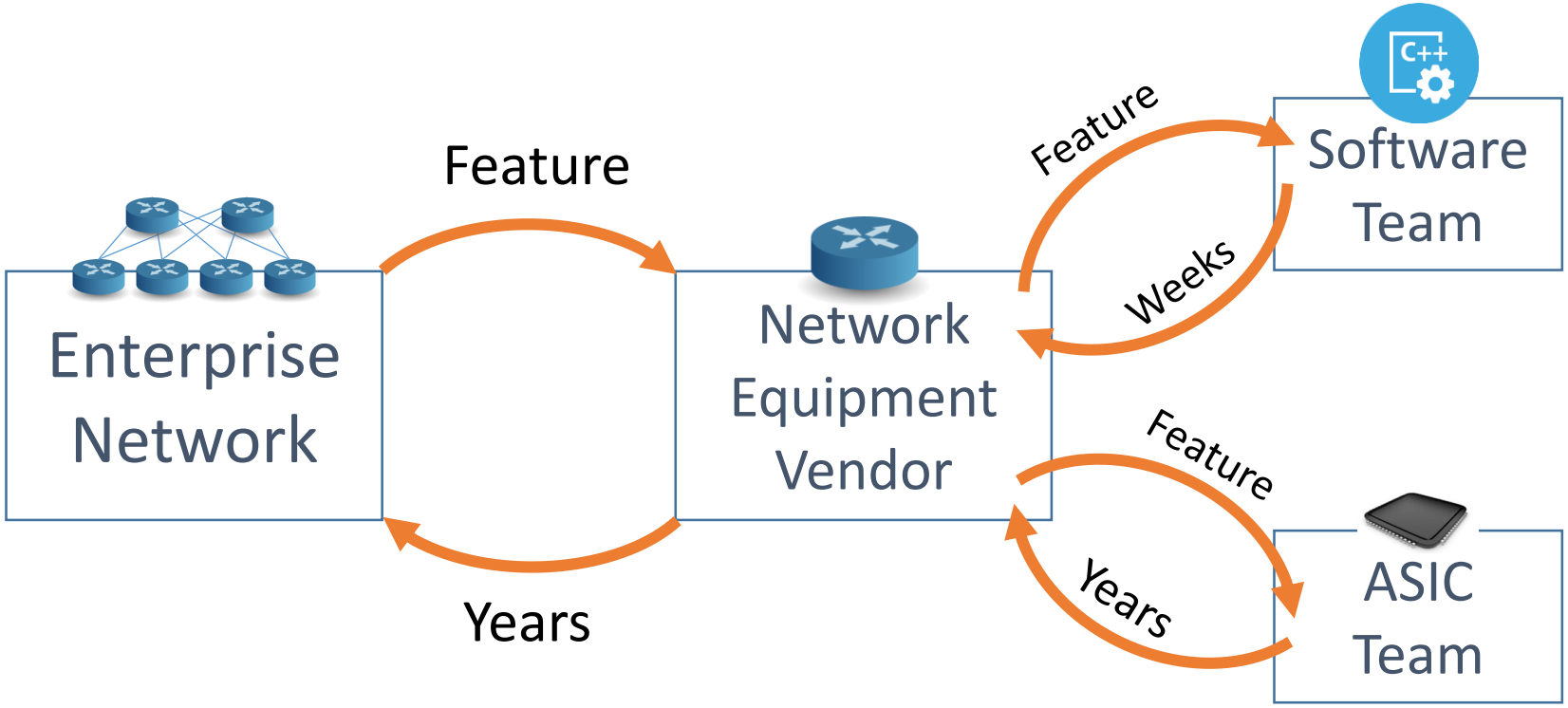
Professor of Electrical Engineering and Computer Science


OSPF
BGP
etc.
Switch OS
Driver





Today, it takes 4 years to add a new feature to a switch ASIC.
And that's the fast-path for urgent features!

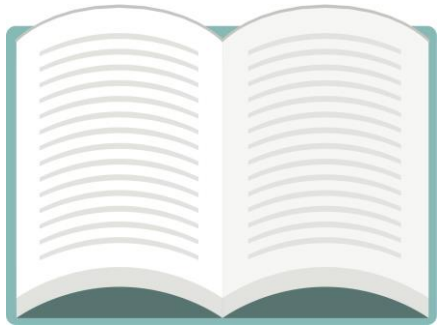

Feature
Enterprise Network
Years
Network Equipment Vendor
Feature
Weeks
Software Team
Feature
Years
ASIC Team

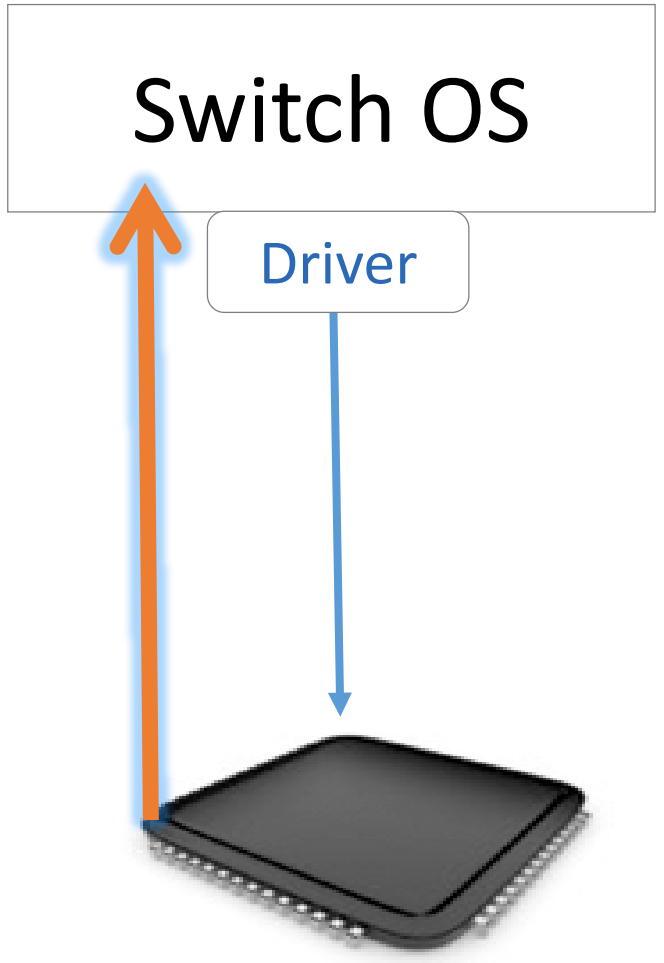
When you need an upgrade

1. A switch vendor can’t just send a software upgrade
2. It takes years to add new features
3. By then, you’ve figured out a kludge to work around it
4. Your network gets more complicated, more brittle.
5. Eventually, when the upgrade is available, it either
   - No longer solves your problem, or
   - You need a fork-lift upgrade, at huge expense.

Network systems are built “bottom-up”

“This is how I process packets …”





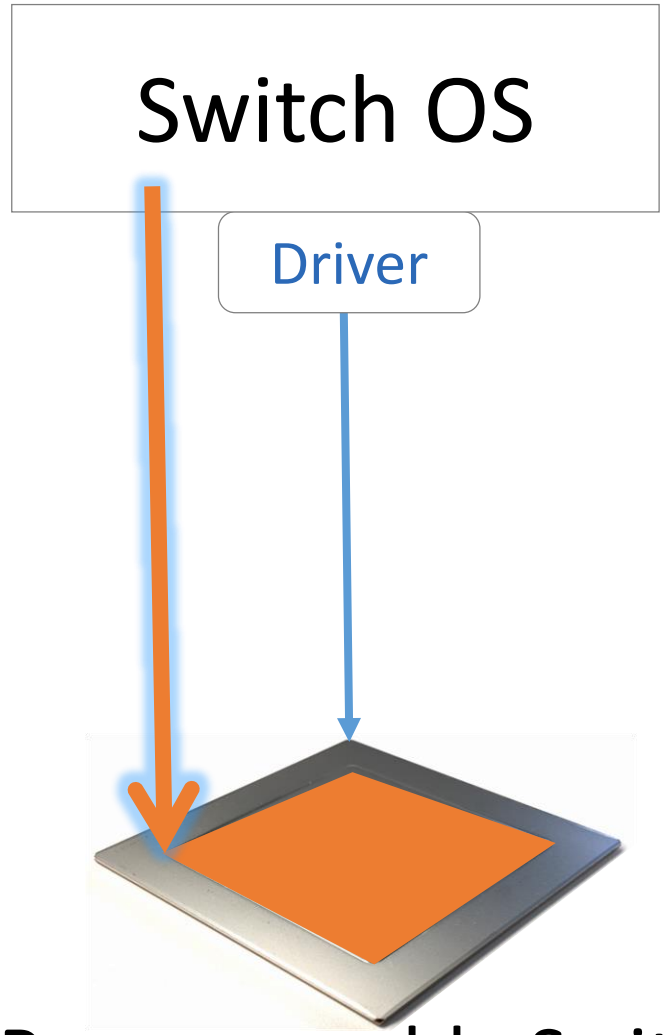
Fixed-function switch

Network systems will be programmed “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);
}
```



Why aren’t all network systems built this way?

“Programmable switches are 10-100x slower than fixed-function switches. They cost more and consume more power.”

Conventional wisdom in networking

This is changing…

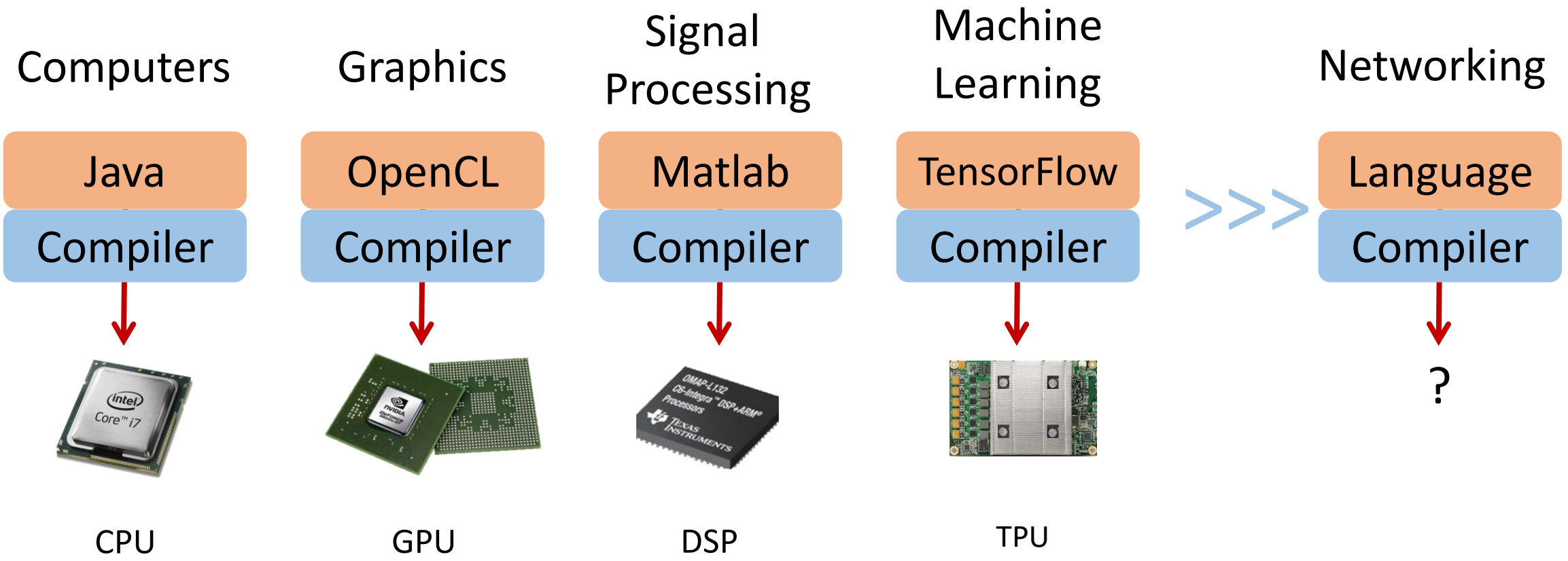
Performance:
At 6.5Tb/s Tofino is the fastest switch chip in the world.

Cost and power:
Same as fixed-function chips.

Easy to program:
Programmers learn to write P4 programs in less than a day.

Why is programmability happening now?

Domain Specific Processors



Domain Specific Processors

Computers

Java

Compiler

CPU

Graphics

OpenCL

Compiler

GPU

Signal Processing

Matlab

Compiler

DSP

Machine Learning

TensorFlow

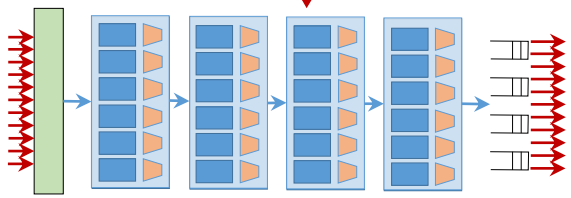
Compiler

TPU

Networking

P4

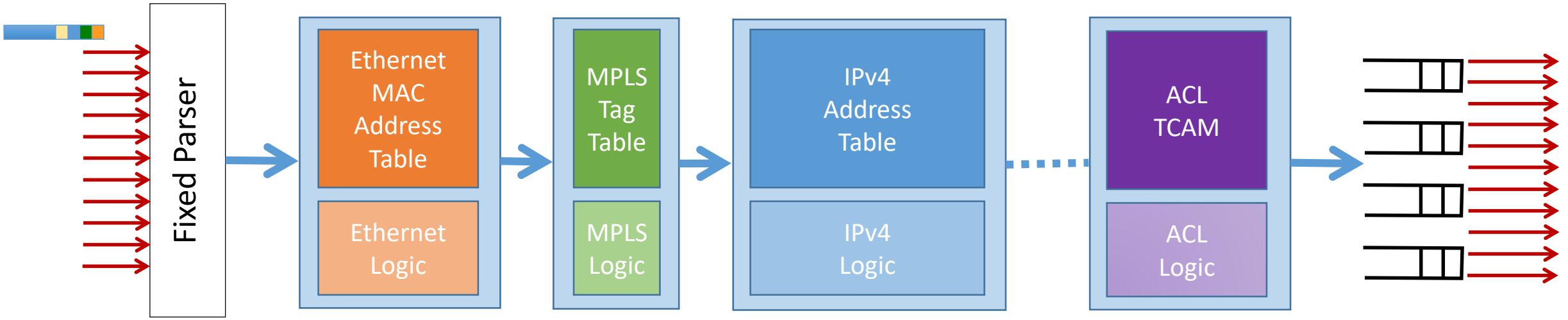
Compiler



PISA
(Tofino)

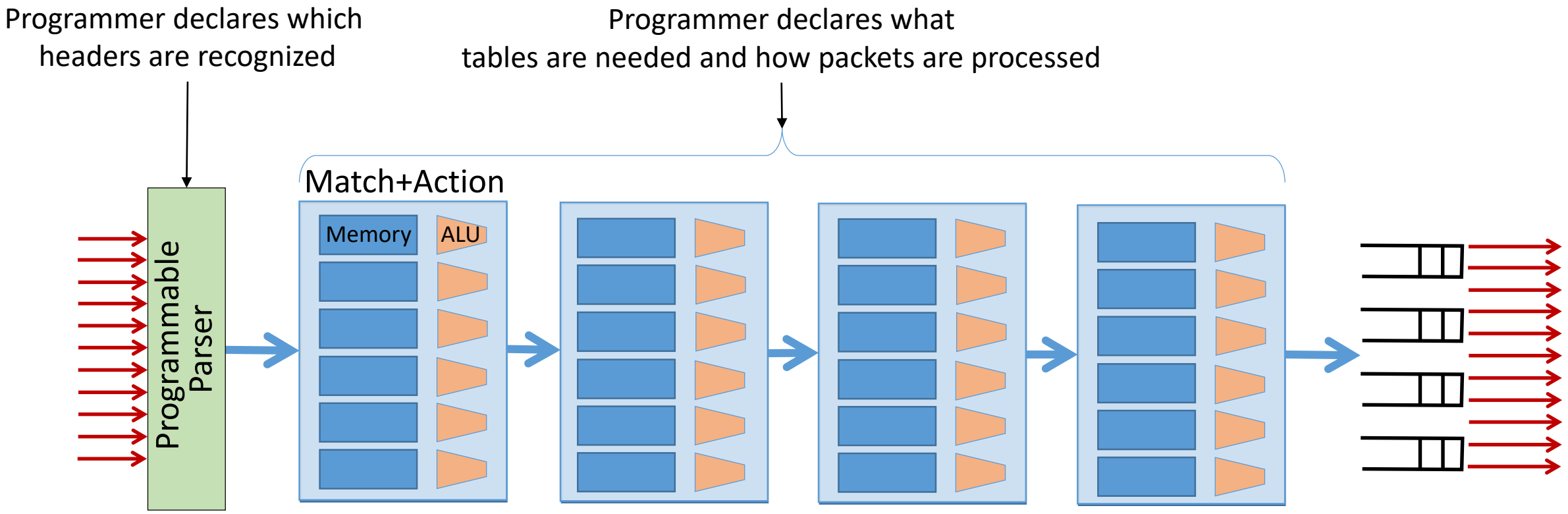
Fixed-Function Switch

Same architecture for 20 years



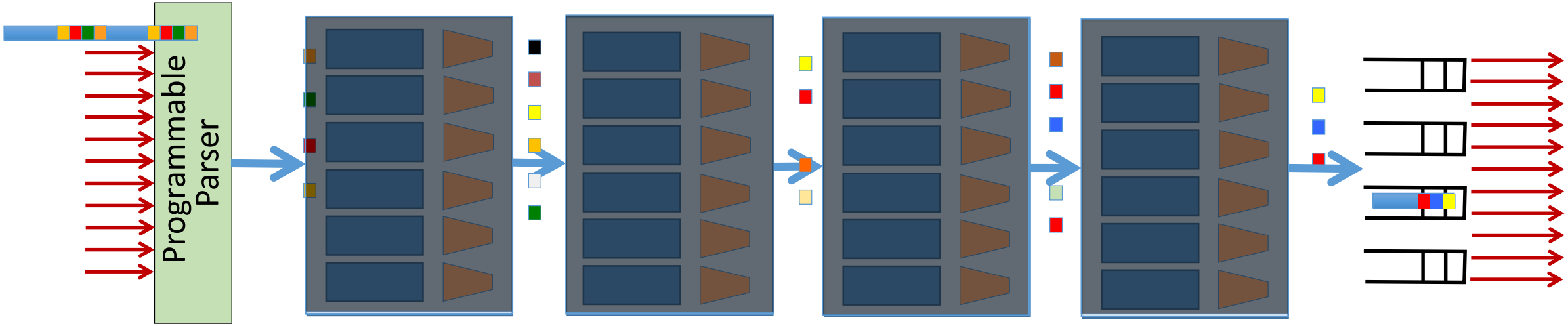
Features and table-sizes are baked in at design time

PISA: Protocol Independent Switch Architecture



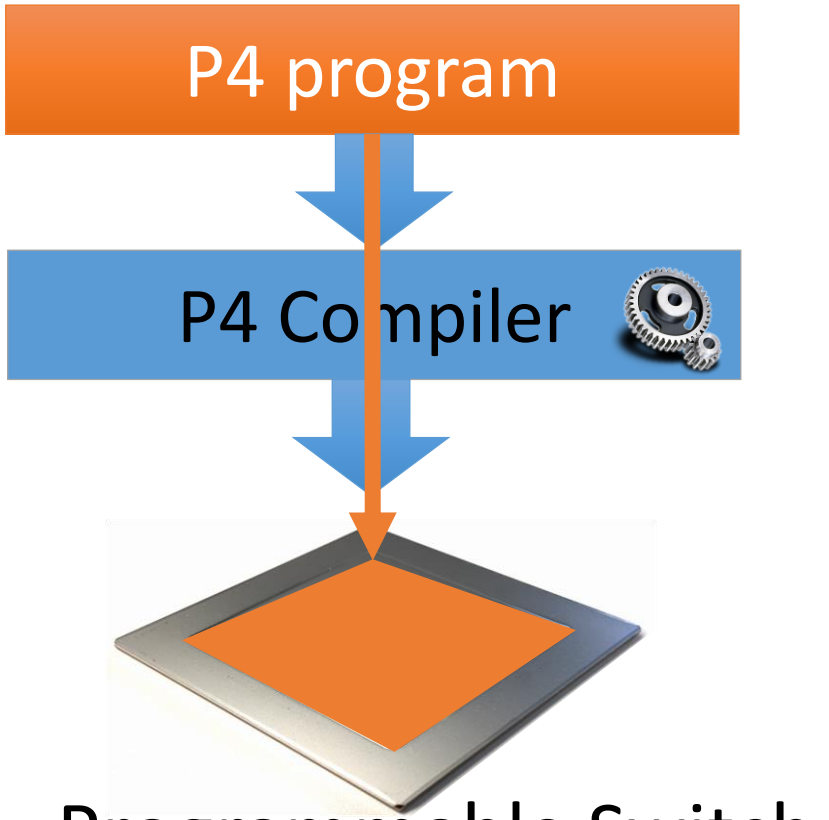
All stages are identical – makes PISA a good “compiler target”

PISA: Protocol Independent Switch Architecture

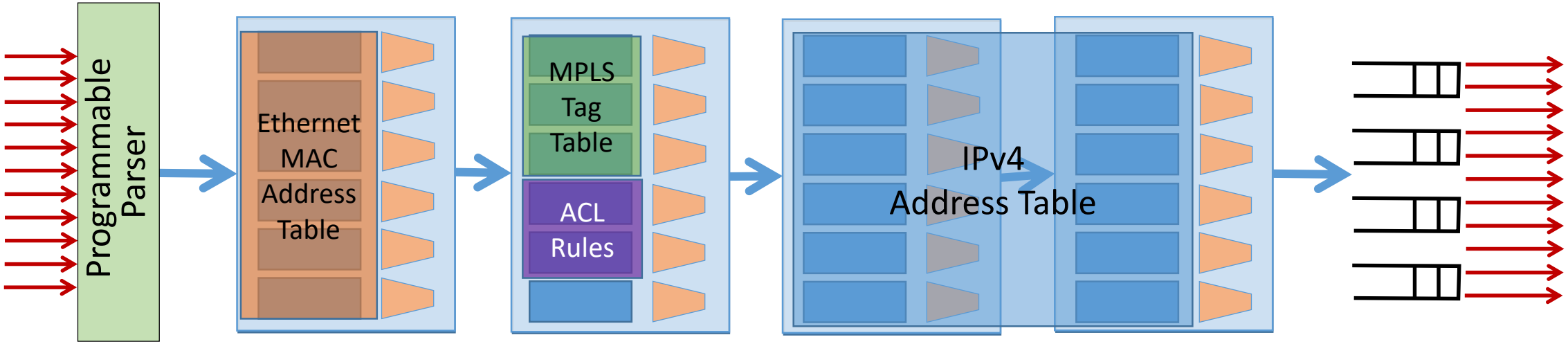


```
table int_table {
  reads {
    ip.protocol;
  }
  actions {
    export_queue_latency;
  }
}
```

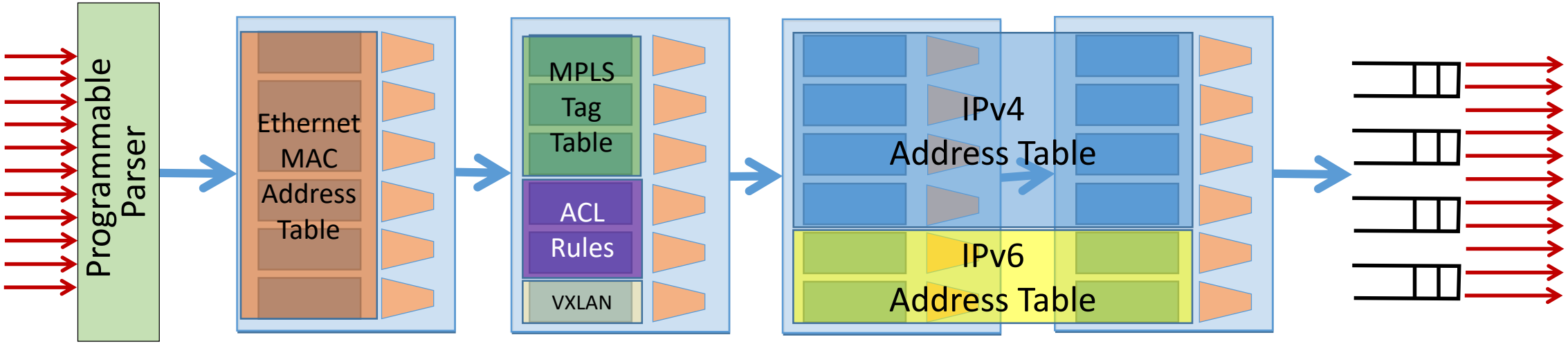
```
action export_queue_latency (sw_id) {
  add_header(int_header);
  modify_field(int_header.kind, TCP_OPTION_INT);
  modify_field(int_header.len, TCP_OPTION_INT_LEN);
  modify_field(int_header.sw_id, sw_id);
  modify_field(int_header.q_latency,
               intrinsic_metadata.deq_timedelta);
  add_to_field(tcp.dataOffset,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



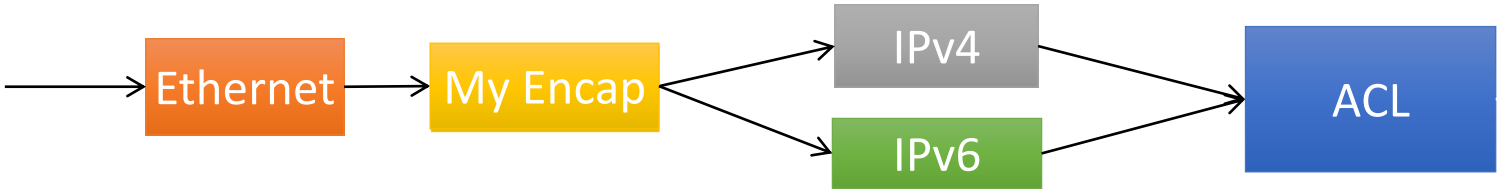
PISA: Protocol Independent Switch Architecture

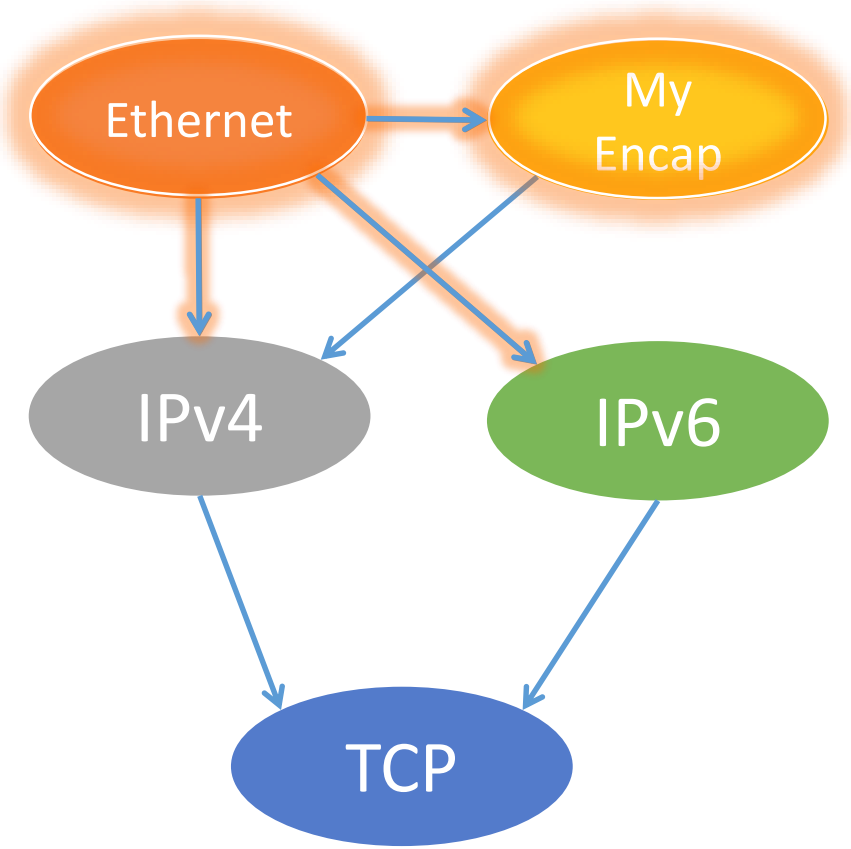


PISA: Protocol Independent Switch Architecture



P4 program example: Parsing Headers



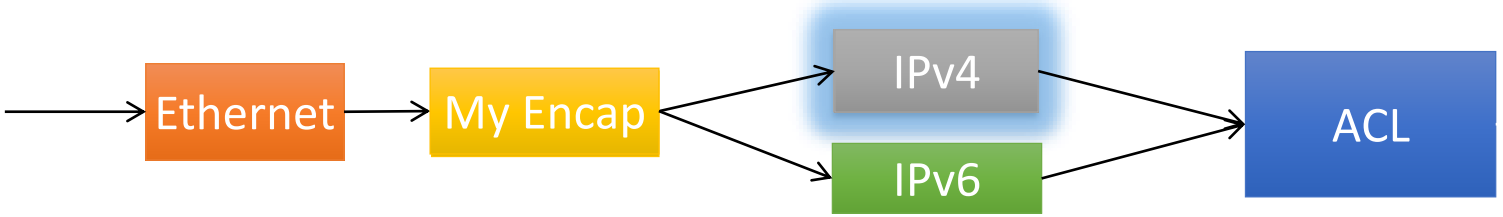



```
header_type ethernet_t {
        fields {
                dstAddr : 48;
```

```
parser parse_ethernet {
  extract(ethernet);
  return select(latest.etherType) {
    0x8100 : parse_my_encap;
    0x800  : parse_ipv4;
    0x86DD : parse_ipv6;
  }
}
```

```
                baz : 4;
                qux : 4;
                next_protocol : 4;
    }
}
```

P4 program example

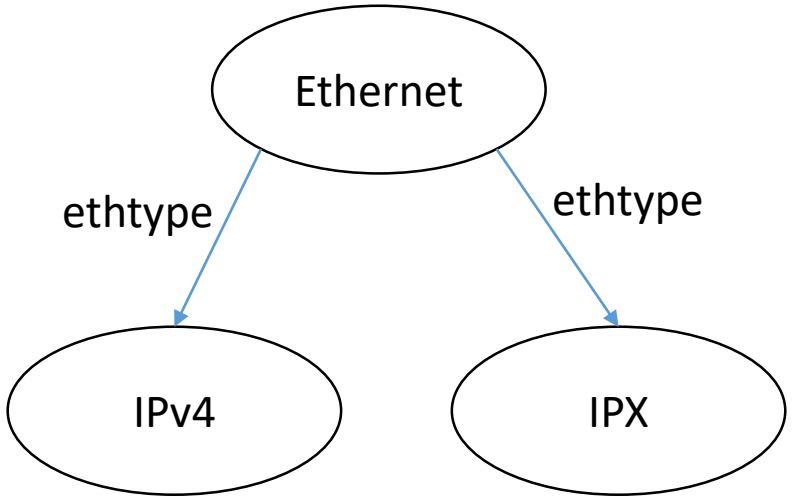


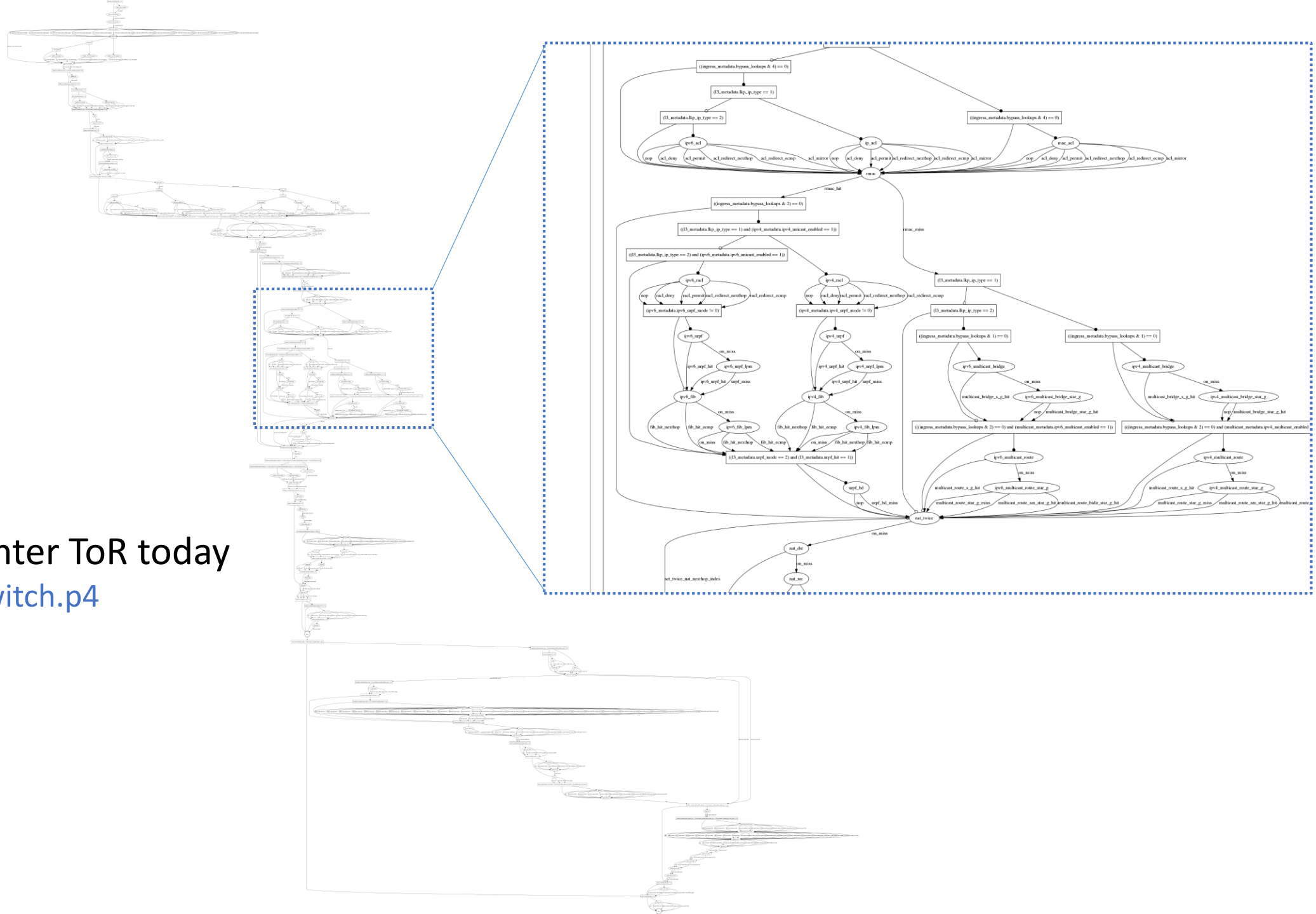

```
table ipv4_lpm
{
        reads {
        ipv4.dstAddr
        }
        actions {
        set_next_hop;
        drop;
        }
}
```

```
control ingress
{
        apply(l2);
        apply(my_encap);
        if (valid(ipv4) {
                apply(ipv4_lpm);
        } else {
                apply(ipv6_lpm);
        }
        apply(acl);
}
```

```
action set_next_hop(nhop_ip
{
        modify_field(metadata.nhop_ipv4_addr, nhop_ipv4_addr);
        modify_field(standard_metadata.egress_port, port);
        add_to_field(ipv4.ttl, -1);
}
```

Protocols and table complexity 20 years ago

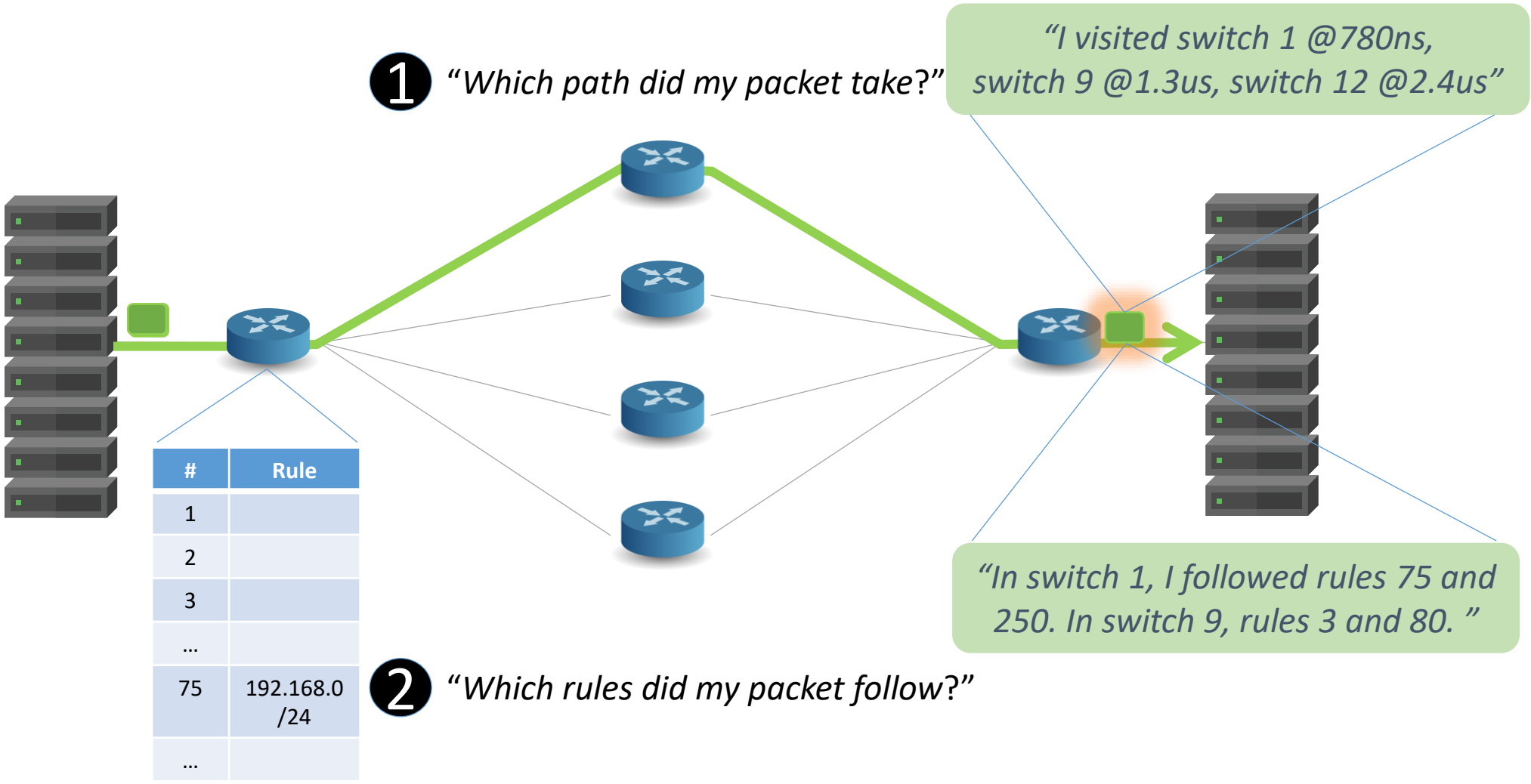




Datacenter ToR today

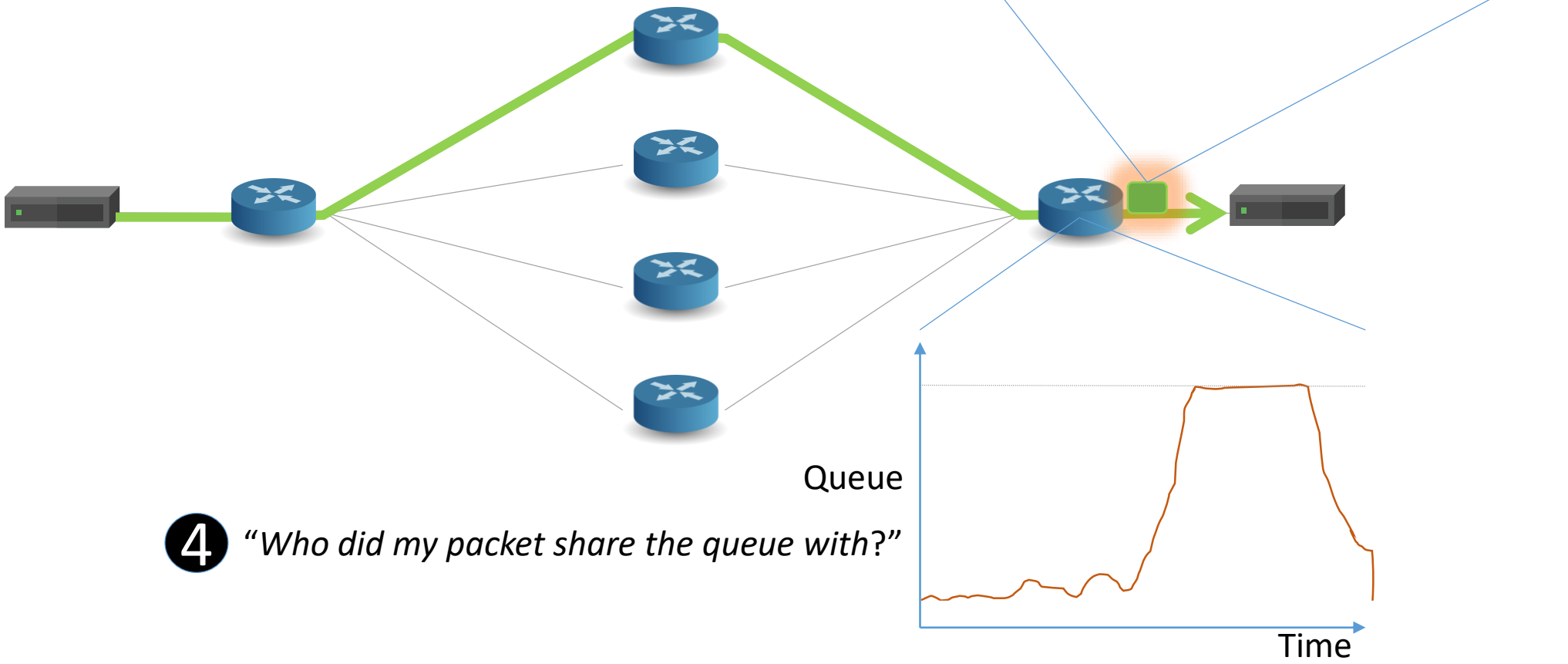
public switch.p4

An example: Telemetry


1
"Which path did my packet take?"
"I visited switch 1 @780ns, switch 9 @1.3us, switch 12 @2.4us"
Rule
1
2
3
...
75 192.168.0 /24
...
2
"Which rules did my packet follow?"
"In switch 1, I followed rules 75 and 250. In switch 9, rules 3 and 80. "

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

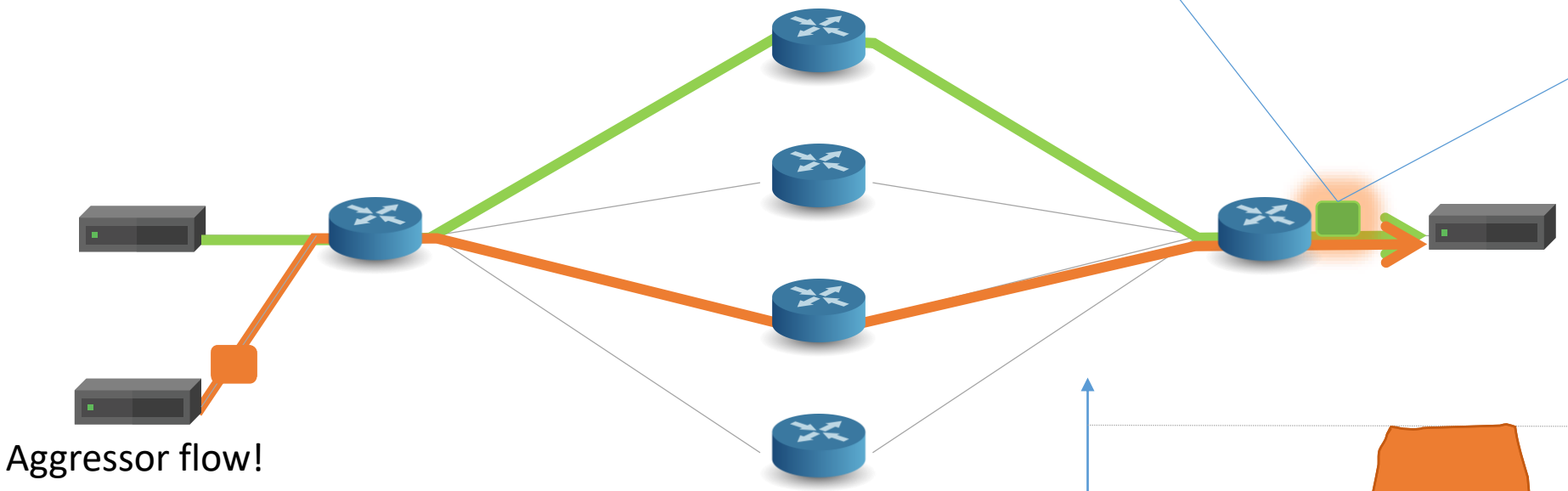
"Delay: 100ns, 200ns, 19740ns"



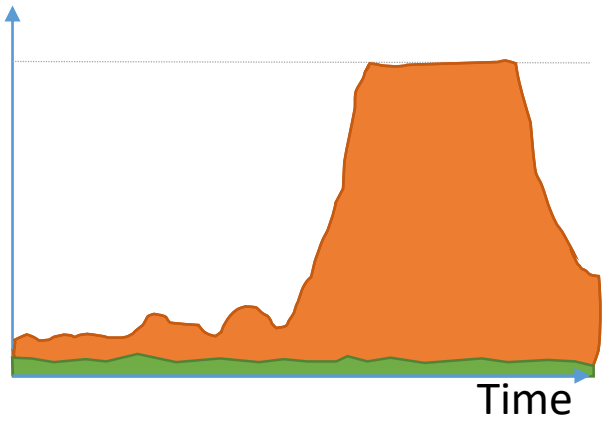
❹ *"Who did my packet share the queue with?"*

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

"Delay: 100ns, 200ns, 19740ns"



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



The network should answer these questions

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?"*

INT (in P4) can answer all four questions for the first time.
At full line rate. Without generating any additional packets!

Thank you.