Programmable Networks
Lecture 1 - Introduction

Sándor Laki, PhD

Communication Networks Laboratory

Dept. of Information Systems, Faculty of Informatics

ELTE Eötvös Loránd University

lakis@elte.hu

http://lakis.web.elte.hu

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

Programmable Networks

- Networking is on the verge of a paradigm shift towards deep programmability
- Huge industrial interest

VMware Buys Nicira For $1.26 Billion And Gives More Clarity About Cloud Strategy

7 years ago

nicira

VMware has acquired Nicira, a startup known for its software defined networking technology in a deal pegged at $1.26 billion. It's an

With $600M Invested in SDN Startups, the Ecosystem Builds

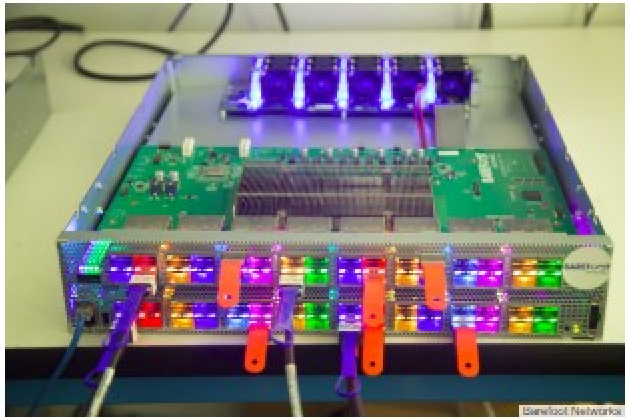
More than $600 million has been invested
according to Rayno Report research. You
take hold with a broad range of alliances a

NEWS

This startup may have built the world's fastest networking switch chip

Barefoot Networks is also making its switch platform completely programmable.

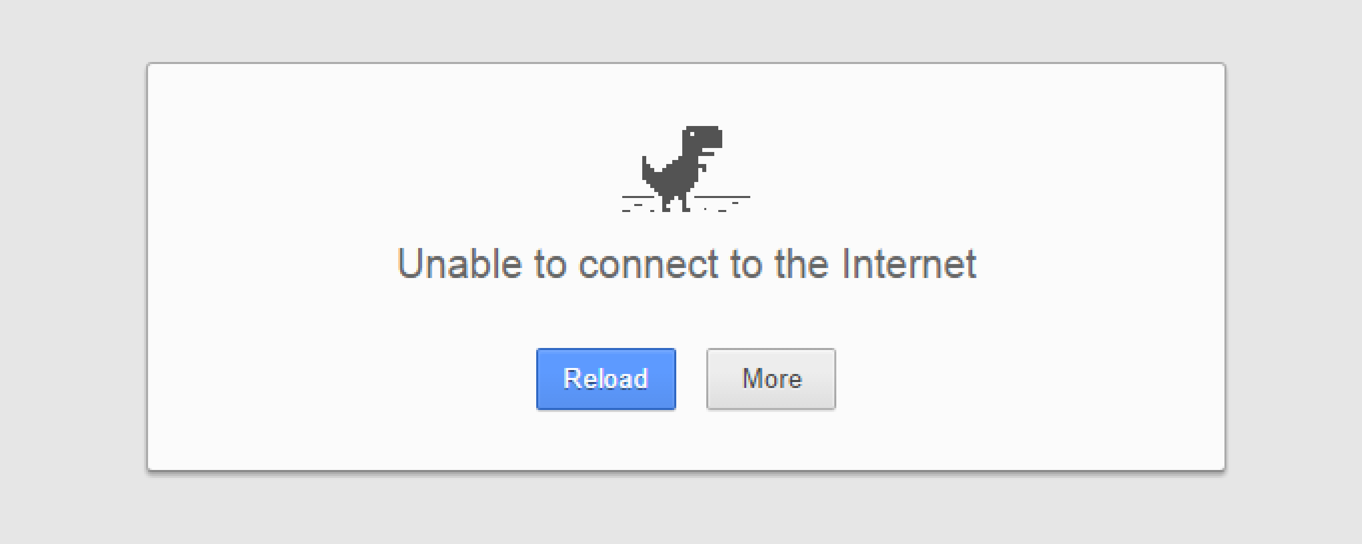
By Stephen Lawson
Senior U.S. Correspondent, IDG News Service | JUN 14, 2016 1:29 PM PT



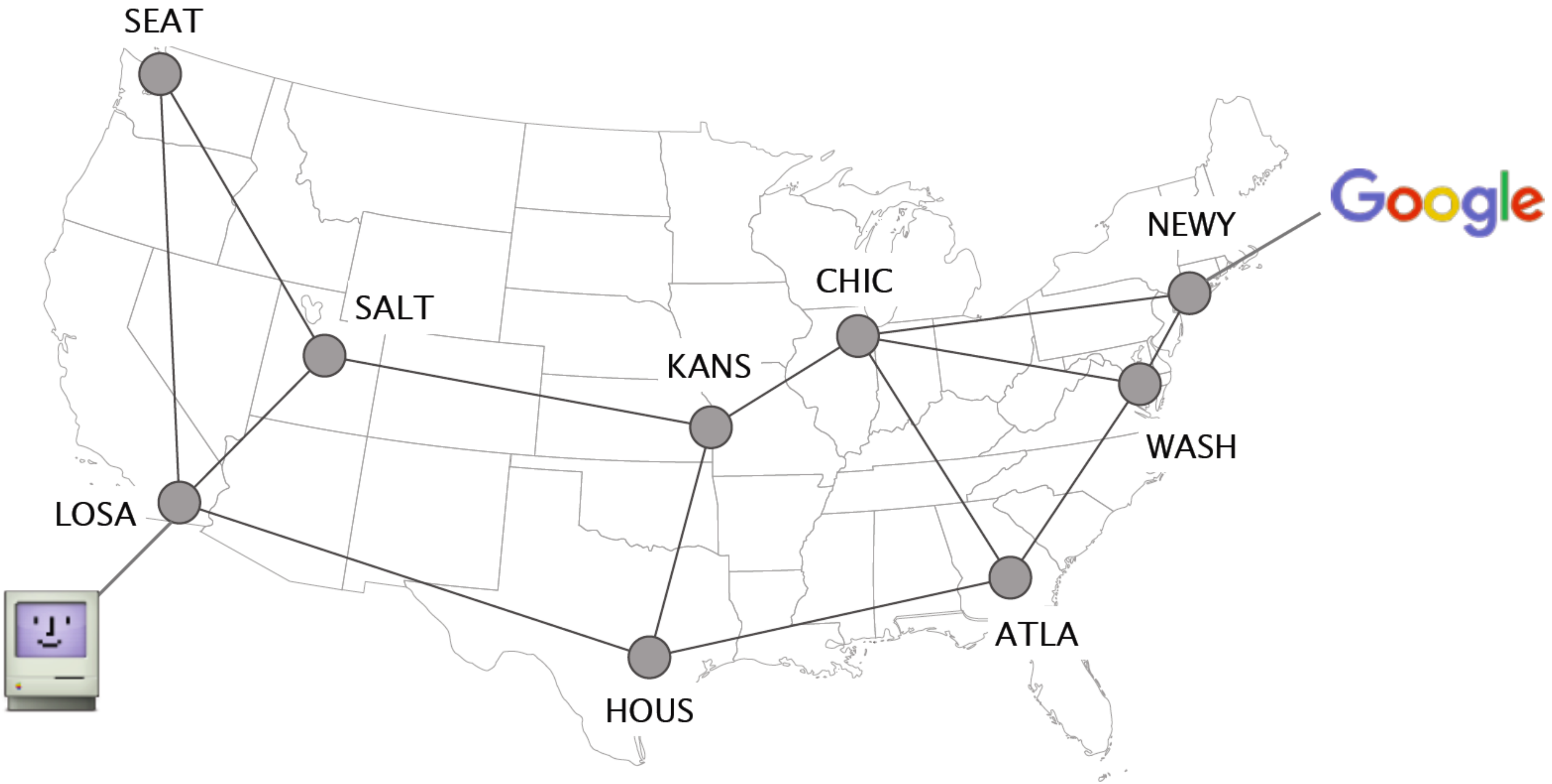
MORE LIKE THIS

- Will software-defined networking doom the command line interface?
- Nvidia GeForce GTX 1080 review: The most badass graphics card ever created
- Network heavy hitters to pool SDN efforts in OpenDaylight project
- VIDEO Intel shows off wireless VR on the HTC Vive

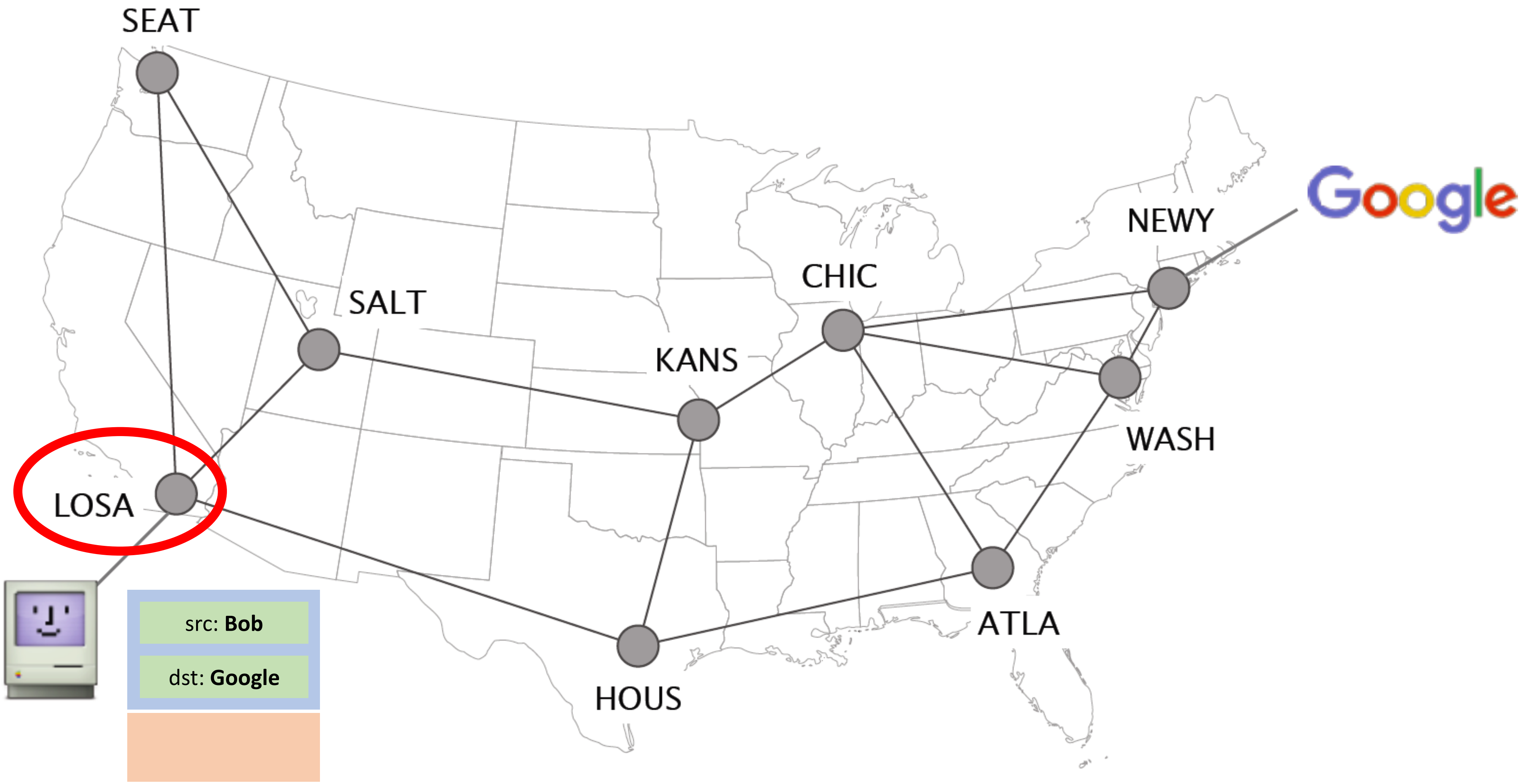
Network management crisis



Networks are large distributed systems

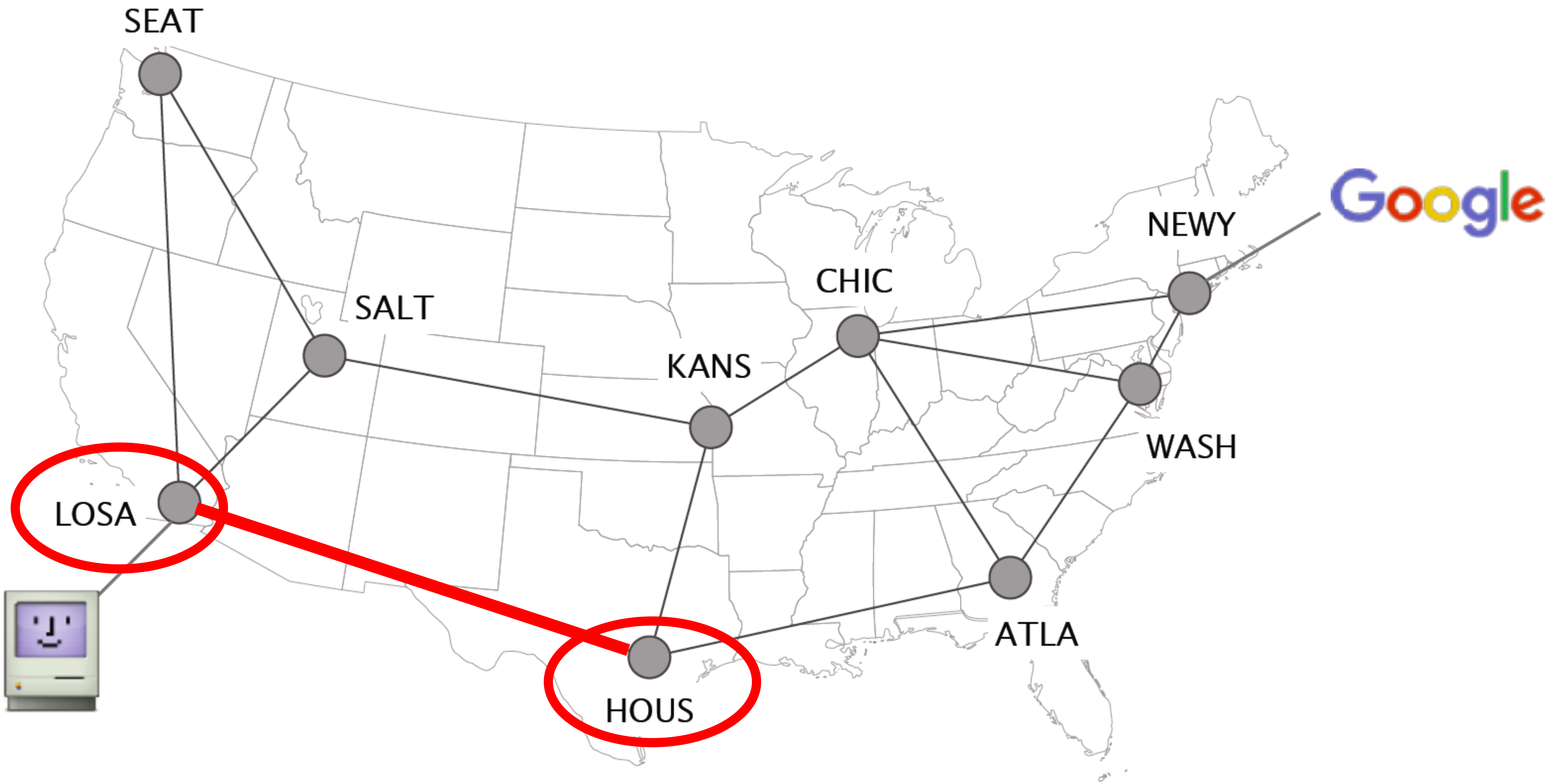


Running distributed algorithms

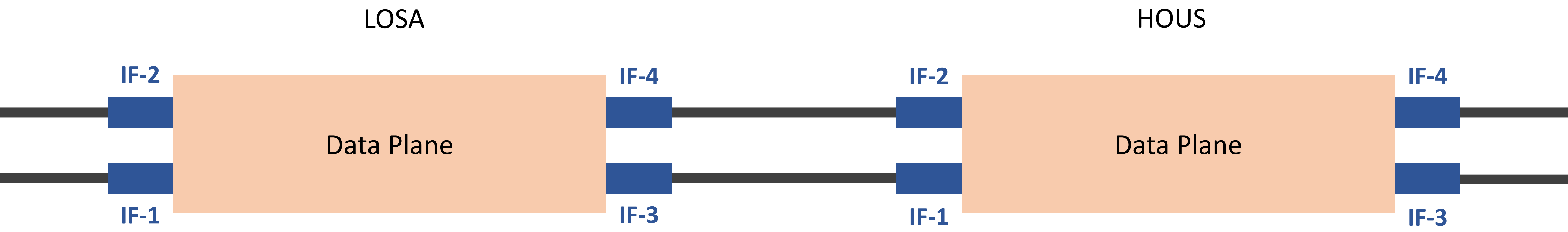


Routers forward IP packets hop-by-hop towards their destination

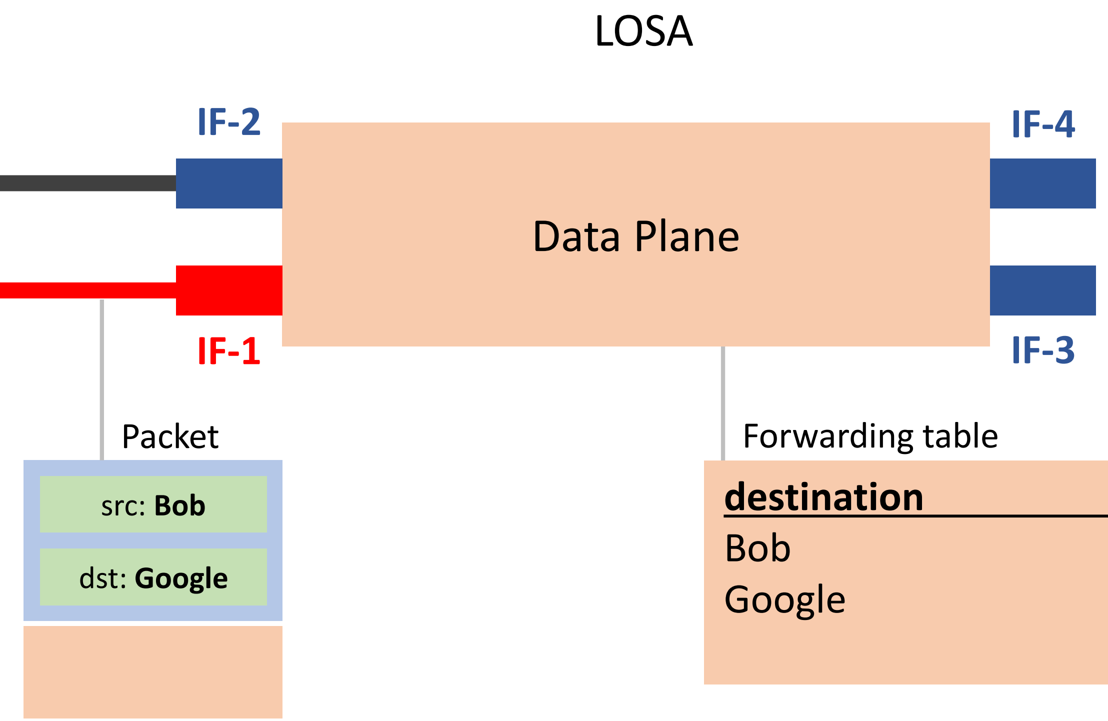
Let's check what is going on between two neighboring routers



Two neighboring routers

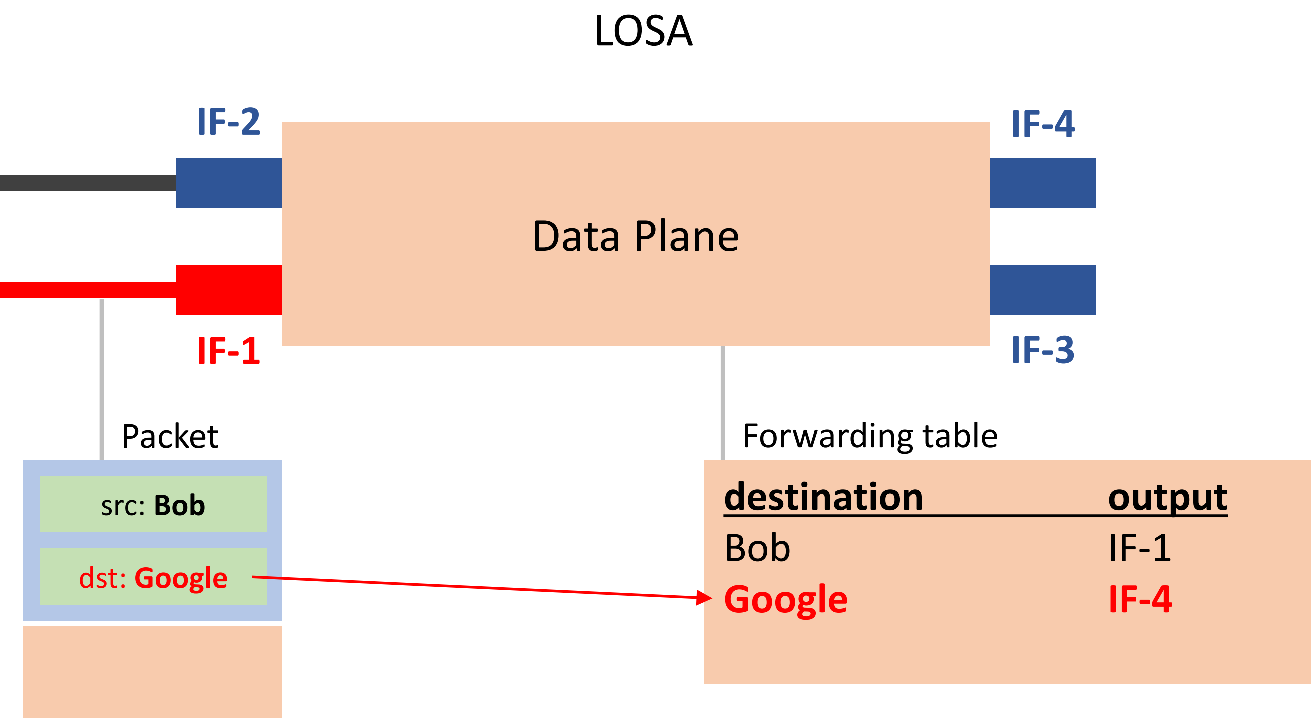


Upon packet reception, routers locally lookup their forwarding table to know where to send it next

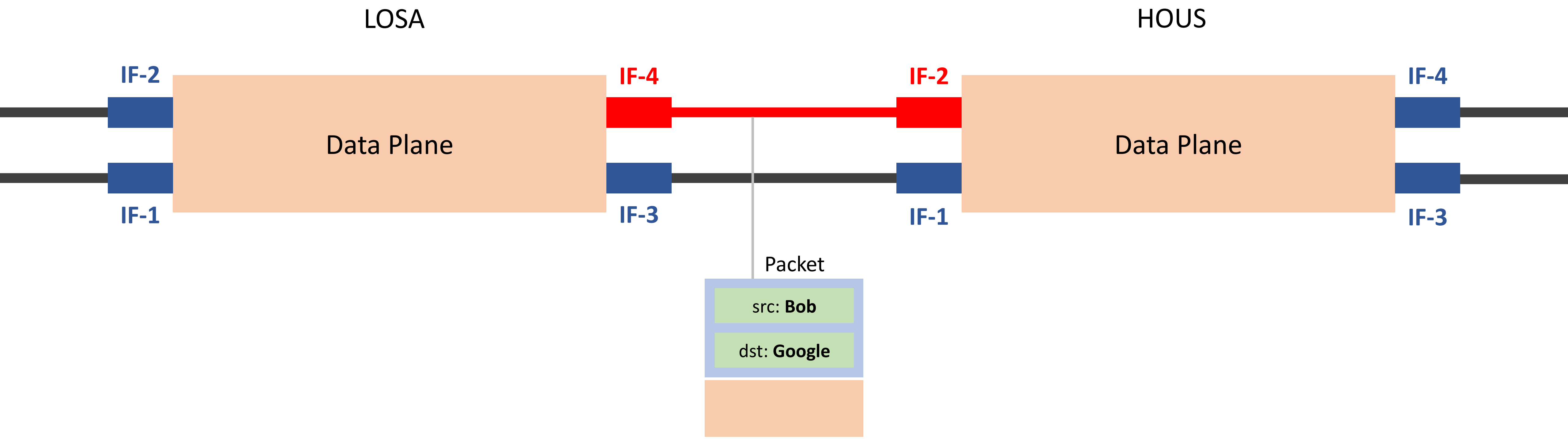


destination	**output**
Bob	IF-1
Google	IF-4

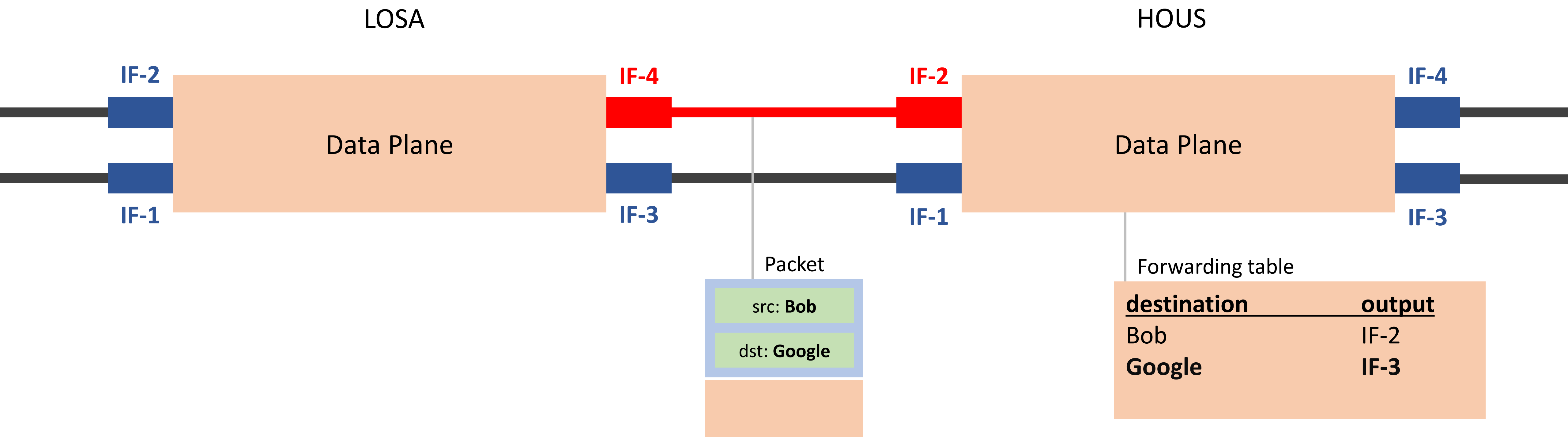
According to the fwd table, the packet should be directed to IF-4



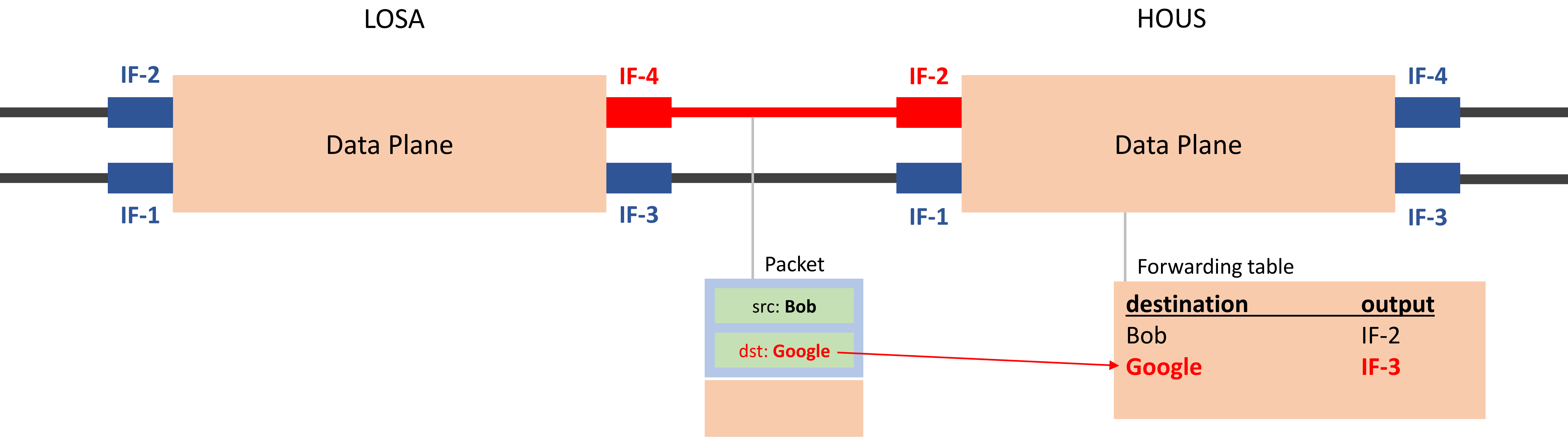
According to the fwd table, the packet should be directed to IF-4



Forwarding is repeated at each router until the destination is reached

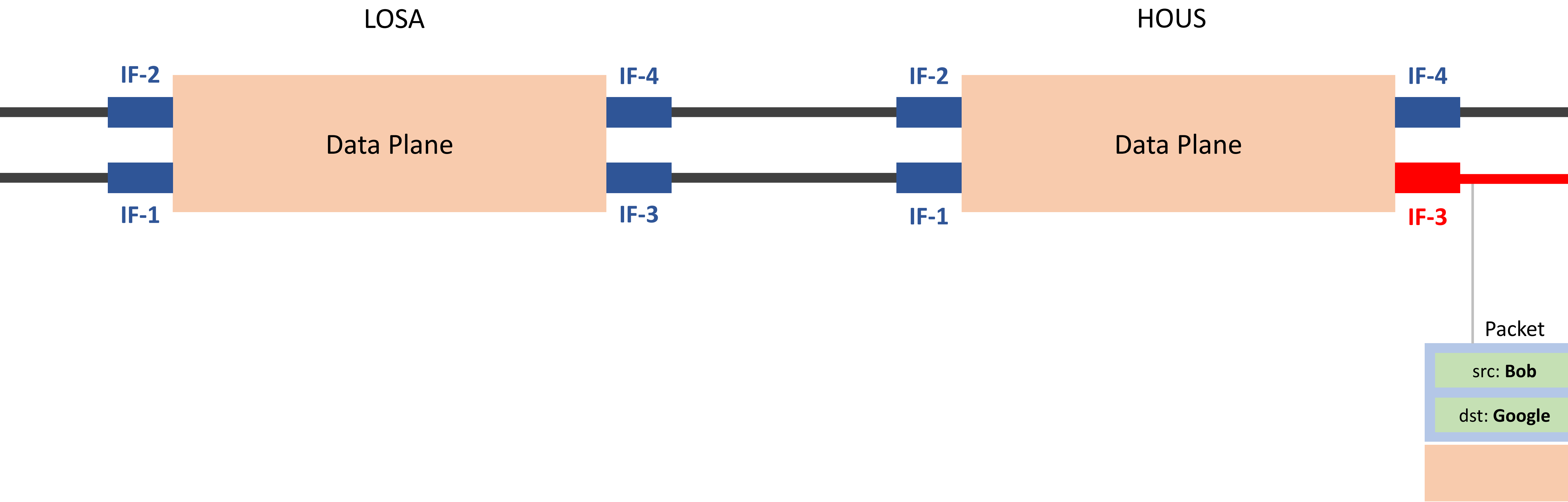


Forwarding is repeated at each router until the destination is reached



destination	output
Bob	IF-2
Google	**IF-3**

Forwarding is repeated at each router until the destination is reached



Network management crisis

- These distributed algorithms produce the forwarding state which drives IP traffic to its destination
- Forwarding behavior is implemented by configuring each forwarding device individually
- Moving to a new network behavior requires the reconfiguration of one or multiple devices

Configuring each element is often done manually, using arcane low-level, vendor-specific “languages”

Cisco IOS

```
!
ip multicast-routing
!
interface Loopback0
 ip address 120.1.7.7 255.255.255.255
 ip ospf 1 area 0
!
!
interface Ethernet0/0
 no ip address
!
interface Ethernet0/0.17
 encapsulation dot1Q 17
 ip address 125.1.17.7 255.255.255.0
 ip pim bsr-border
 ip pim sparse-mode
!
!
router ospf 1
 router-id 120.1.7.7
 redistribute bgp 700 subnets
!
router bgp 700
 neighbor 125.1.17.1 remote-as 100
 !
 address-family ipv4
  redistribute ospf 1 match internal external 1 external 2
  neighbor 125.1.17.1 activate
 !
 address-family ipv4 multicast
  network 125.1.79.0 mask 255.255.255.0
  redistribute ospf 1 match internal external 1 external 2
```

Juniper JunOS

```
interfaces {
   so-0/0/0 {
        unit 0 {
            family inet {
                address 10.12.1.2/24;
            }
            family mpls;
        }
    }
   ge-0/1/0 {
        vlan-tagging;
        unit 0 {
            vlan-id 100;
            family inet {
                address 10.108.1.1/24;
            }
            family mpls;
        }
        unit 1 {
            vlan-id 200;
            family inet {
                address 10.208.1.1/24;
            }
        }
    }
…
}
protocols {
    mpls {
        interface all;
    }
    bgp {
```

Source: slides of Laurent Vanbever (ETHZ)

A single mistyped line is enough to bring down the entire network

Cisco IOS

```
!
ip multicast-routing
!
interface Loopback0
 ip address 120.1.7.7 255.255.255.255
 ip ospf 1 area 0
!
!
interface Ethernet0/0
 no ip address
!
interface Ethernet0/0.17
 encapsulation dot1Q 17
 ip address 125.1.17.7 255.255.255.0
 ip pim bsr-border
 ip pim sparse-mode
!
!
router ospf 1
 router-id 120.1.7.7
 redistribute bgp 700 subnets
!
router bgp 700
 neighbor 125.1.17.1 remote-as 100
 !
 address-family ipv4
  redistribute ospf 1 match internal external 1 external 2
  neighbor 125.1.17.1 activate
 !
 address-family ipv4 multicast
  network 125.1.79.0 mask 255.255.255.0
  redistribute ospf 1 match internal external 1 external 2
```

Anything else than 700 creates blackholes

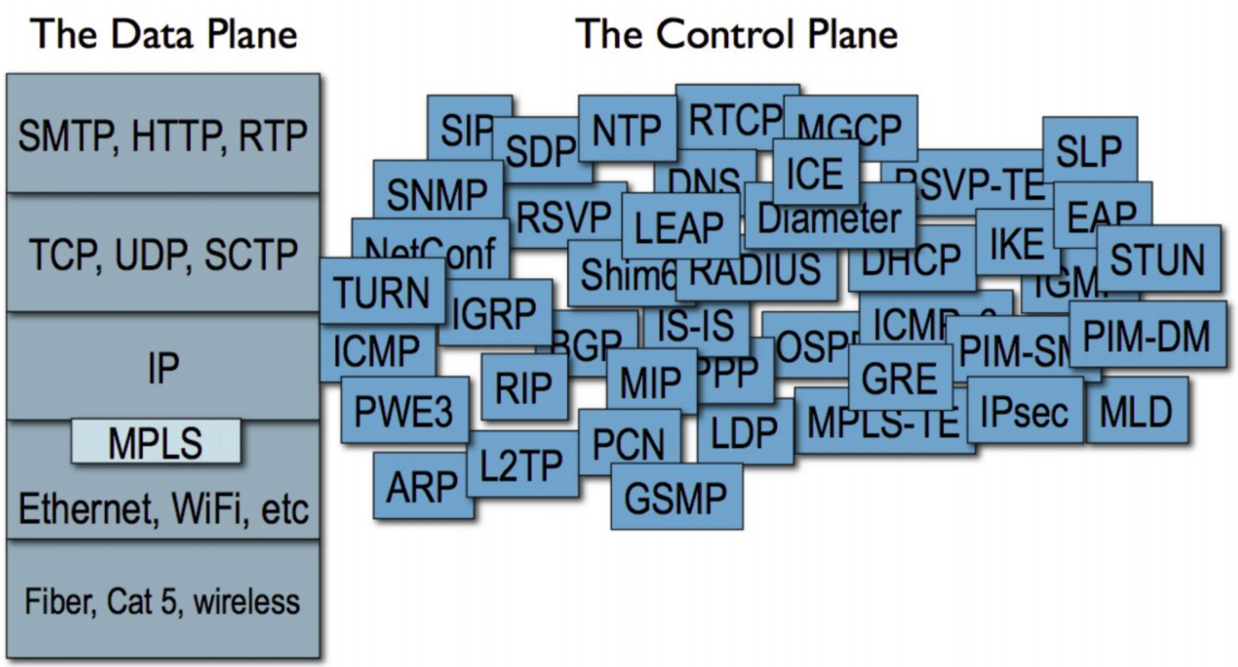
Juniper JunOS

```
interfaces {
   so-0/0/0 {
        unit 0 {
            family inet {
                address 10.12.1.2/24;
            }
            family mpls;
        }
    }
   ge-0/1/0 {
        vlan-tagging;
        unit 0 {
            vlan-id 100;
            family inet {
                address 10.108.1.1/24;
            }
            family mpls;
        }
        unit 1 {
            vlan-id 200;
            family inet {
                address 10.208.1.1/24;
            }
        }
    }
…
}
protocols {
    mpls {
        interface all;
    }
    bgp {
```

Source: slides of Laurent Vanbever (ETHZ)

Network management crisis

- It's not only about the problem of configuring the network
- but the high level of complexity in networks



High Complexity

+

Low-level Management

=

Problems

Source: Mark Handley. Re-thinking the control architecture of the internet. Keynote talk. REARCH. December 2009.

ORACLE + Dyn   DNS   Security   Other Products   Company   Support   SIGN IN   CONNECT

VANTAGEPOINT
IN: RESEARCH

Widespread impact caused by Level 3 BGP route leak

Research // Nov 7, 2017 // Doug Madory

For a little more than 90 minutes yesterday, internet service for millions of users in the U.S. and around the world slowed to a crawl. Was this widespread service degradation caused by the latest botnet threat? Not this time. The cause was yet another BGP routing leak — a router misconfiguration directing internet traffic from its intended path to somewhere else.

* https://dyn.com/blog/widespread-impact-caused-by-level-3-bgp-route-leak/

We have a little problem here…

- A little outage
  – **for more than 90 mins**
- **Affected millions of users from the US and world-wide**
- Cause: BGP route leaking
  - **A misconfigured router** directed Internet traffic from its intended path to somewhere else.

August 2017

The Register®
Biting the hand that feeds IT

DATA CENTRE SOFTWARE SECURITY DEVOPS BUSINESS PERSONAL TECH SCIENCE E

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35 40 SHARE ▼

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory "leaked" a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

Since Google doesn't provide transit services, as BGP Mon explains, that traffic either filled a link beyond its capacity, or hit an access control list, and disappeared.

The outage in Japan only lasted a couple of hours, but was so severe

* https://www.theregister.co.uk/2017/08/27/google_routing_blunder_sent_japans_internet_dark/

Human factor

- People also often mistakenly destroy their own infrastructure

United Airlines Blames Router for Grounded Flights

Alexandra Talty Senior Contributor
Personal Finance

After a computer problem caused nearly two hours of grounded flights for United Airlines this morning and ongoing delays throughout the day, the airline announced the culprit: a faulty router.

Spokeswoman Jennifer Dohm said that the router problem caused "degraded network connectivity," which affected various applications.

A computer glitch in the airline's reservations system caused the Federal Aviation Administration to impose a groundstop at 8:26 a.m. E.T. Planes that were in the air continued to operate, but all planes on the ground were held. There were reports of agents writing tickets by hand. The ground stop was lifted around 9:47 a.m. ET.

Traders work on the floor of the New York Stock Exchange (NYSE) in July 2015. (Photo by Spencer Platt/Getty Images)

DOWNTIME

UPDATED: "Configuration Issue" Halts Trading on NYSE

The article has been updated with the time trading resumed.

A second update identified the cause of the outage as a "configuration issue."

A third update added information about a software update that created the configuration issue.

Human factor

- People also often mistakenly destroy their own infrastructure

11,353 views | Jul 8, 2015, 03:36pm

United Airlines Blames Router for Grounded Flights

Alexandra Talty Senior Contributor
Personal Finance

After a computer problem caused nearly two hours of grounded flights for United Airlines this morning and ongoing delays throughout the day, the airline announced the culprit: a faulty router.

Spokeswoman Jennifer Dohm said that the router problem caused "degraded network connectivity," which affected various applications.

A computer glitch in the airline's reservations system caused the Federal Aviation Administration to impose a groundstop at 8:26 a.m. E.T. Planes that were in the air continued to operate, but all planes on the ground were held. There were reports of agents writing tickets by hand. The ground stop was lifted around 9:47 a.m. ET.

„Human factors are responsible for 50% to 80% of network outages."

Jupiter Networks, What's Behind Network Downtime?, 2008

Traders work on the floor of the New York Stock Exchange (NYSE) in July 2015. (Photo by Spencer Platt/Getty Images)

DOWNTIME

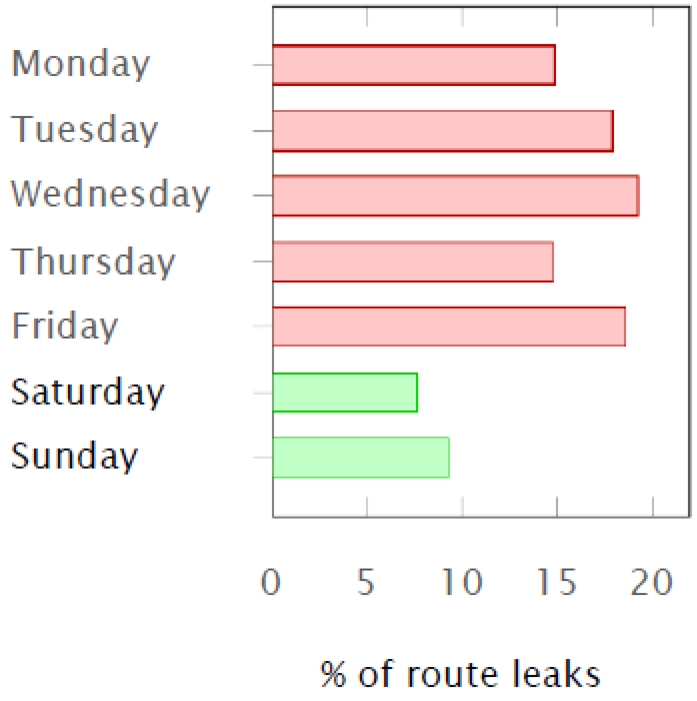
UPDATED: "Configuration Issue" Halts Trading on NYSE

The article has been updated with the time trading resumed.

A second update identified the cause of the outage as a "configuration issue."

A third update added information about a software update that created the configuration issue.

Data networks work better during week-ends…☺



source: Job Snijders (NTT)

Network management crisis

"Cost per network outage can be as high as 750 000$"

Source: Smart Management for Robust Carrier Network Health and Reduced TCO!, NANOG54, 2012

Root of the problem

- Networking devices are completely closed
  - Closed software
  - Closed hardware

Course goals & organization

Goals

- Learn the principles of network programmability
  - Both data and control planes
- Learn P4 language
- Get insights into hot research problems

Logistics

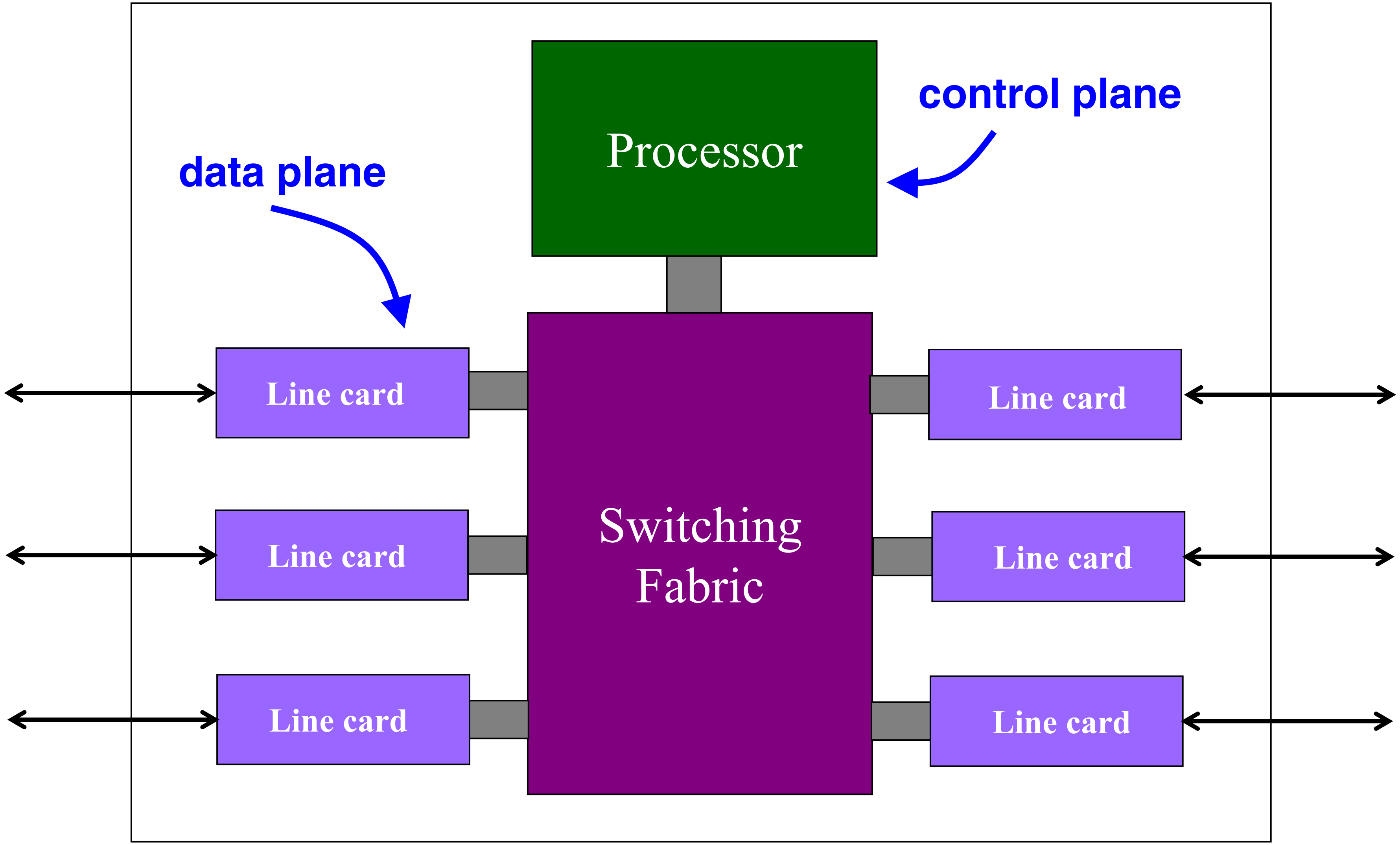
- Two 7-8 weeks blocks
  - Lectures/Excercises
    - Principles of SDN and data plane programmability
    - Learn how to program in P4
  - Group project
    - In teams of 2-3 person
    - 15 min presentation + report at the end
    - Code available on GitHub
- Final grade
  - 50% EXAM
  - 50% Group project (code, report, presentation)

Data, Control and Management planes

Timescales

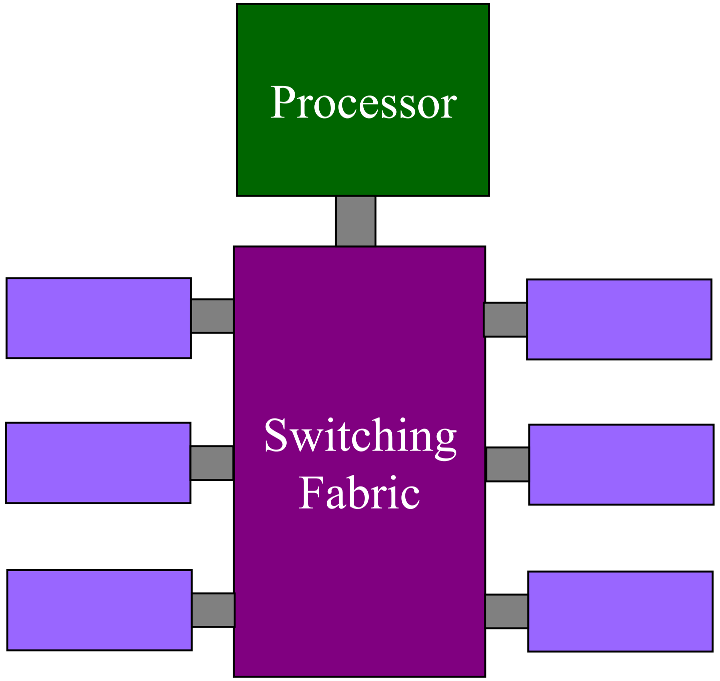
	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, circuit set-up	Analysis, configuration
Location	Line-card hardware	Router software	Humans or scripts

Data and Control Planes



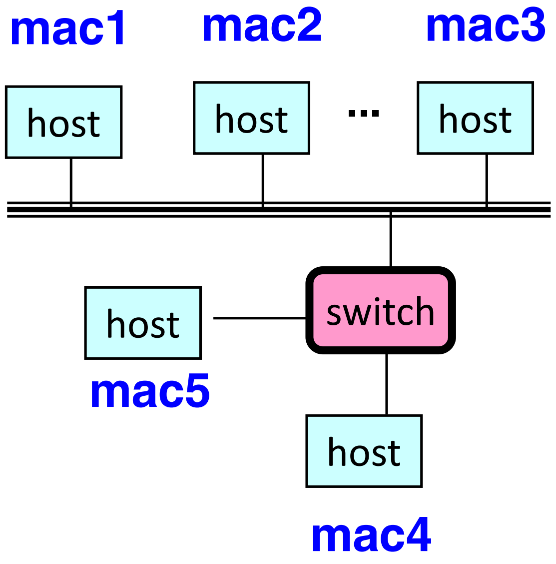
Data Plane

- Streaming algorithms on packets
  - Matching on some bits
  - Perform some actions
- Wide range of functionality
  - Forwarding
  - Access control
  - Mapping header fields
  - Traffic monitoring
  - Buffering and marking
  - Shaping and scheduling
  - Deep packet inspection



Switch: Match on Destination MAC

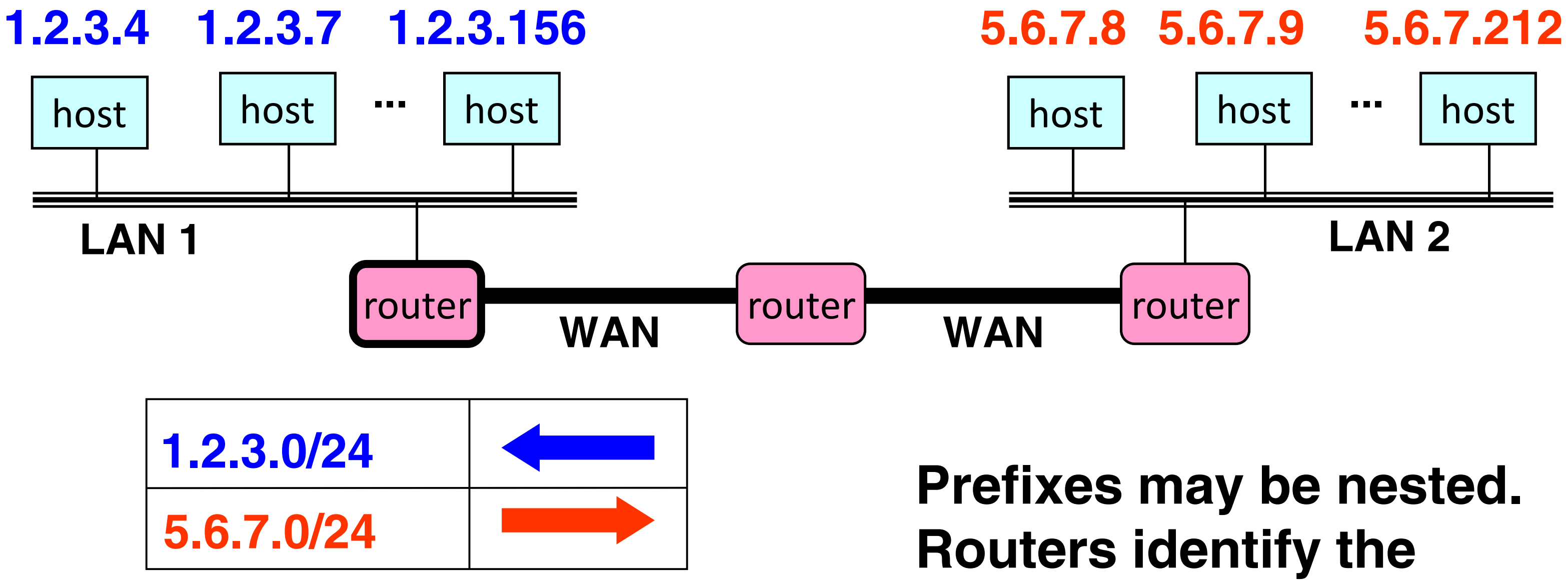
- MAC addresses are location independent
  - Assigned by the vendor of the interface card
  - Cannot be aggregated across hosts in LAN



mac1	↑
mac2	↑
mac3	↑
mac5	←
mac4	↓

Router: Match on IP Prefix

- IP addresses grouped into common subnets
  - Allocated by ICANN, regional registries, ISPs, and within individual organizations
  - Variable-length prefix identified by a mask length

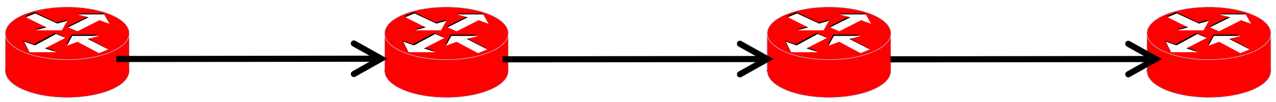


forwarding table

Prefixes may be nested. Routers identify the *longest matching* prefix.

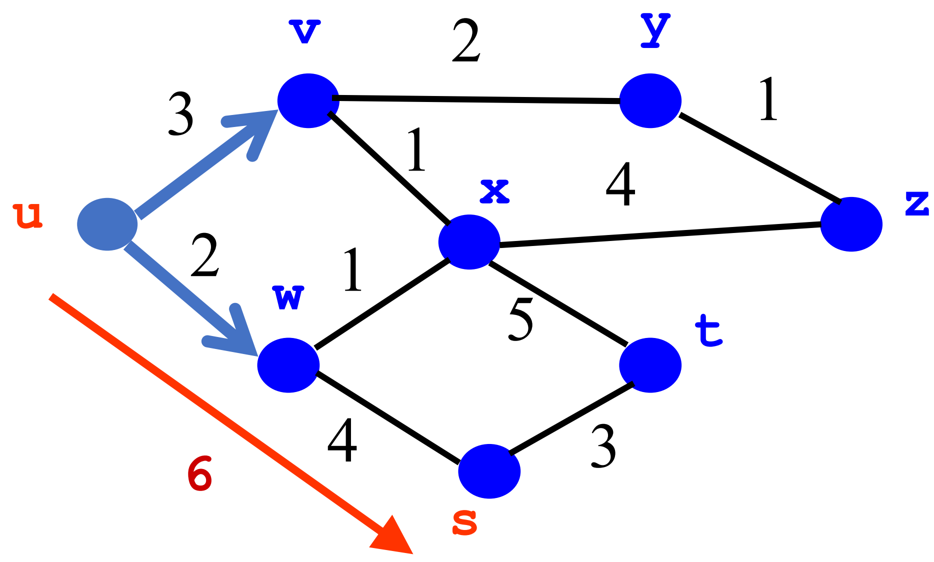
Forwarding vs. Routing

- **Forwarding**: data plane
  - Directing a data packet to an outgoing link
  - Individual router *using* a forwarding table
- **Routing**: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Individual router ***creating*** a forwarding table



Example: Shortest-Path Routing

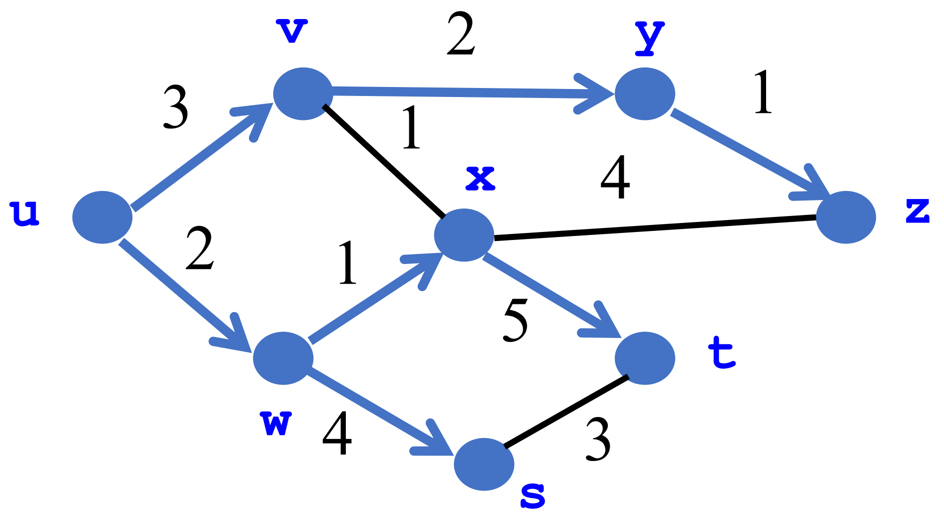
- Compute: *path costs* to all nodes
  - From a source u to all other nodes
  - Cost of the path through each link
  - Next hop along least-cost path to s



	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Distributed Control Plane

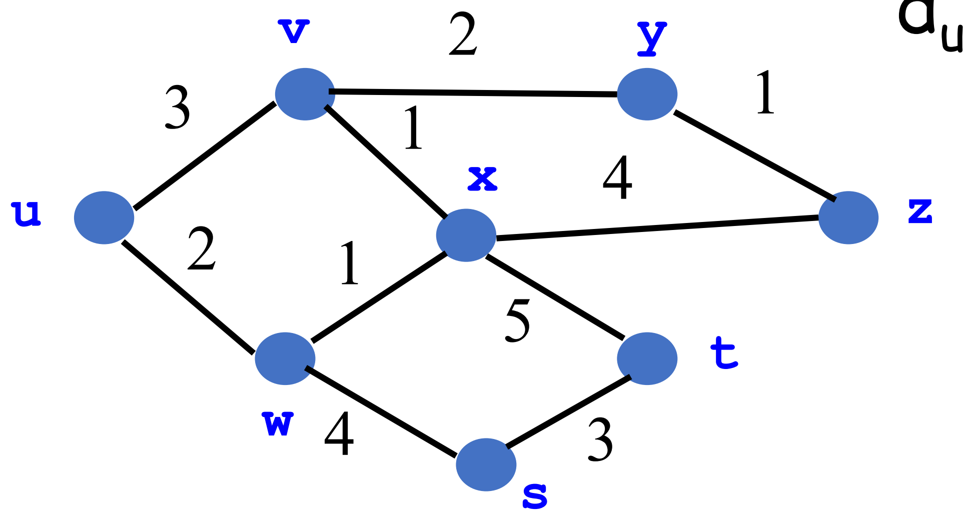
- Link-state routing: OSPF, IS-IS
  - Flood the entire topology to all nodes
  - Each node computes shortest paths
  - Dijkstra's algorithm



	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Distributed Control Plane

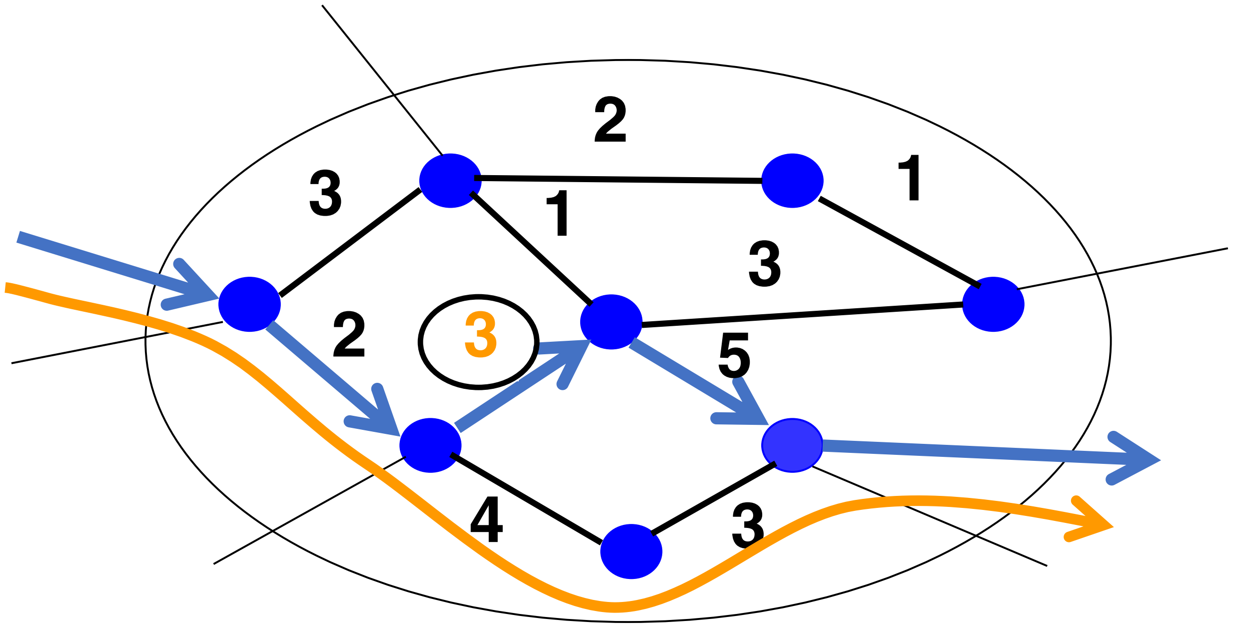
- Distance-vector routing: RIP, EIGRP
  - Each node computes path cost
  - … based on each neighbors' path cost
  - Bellman-Ford algorithm



$$d_u(z) = \min\{c(u,v) + d_v(z),\ c(u,w) + d_w(z)\}$$

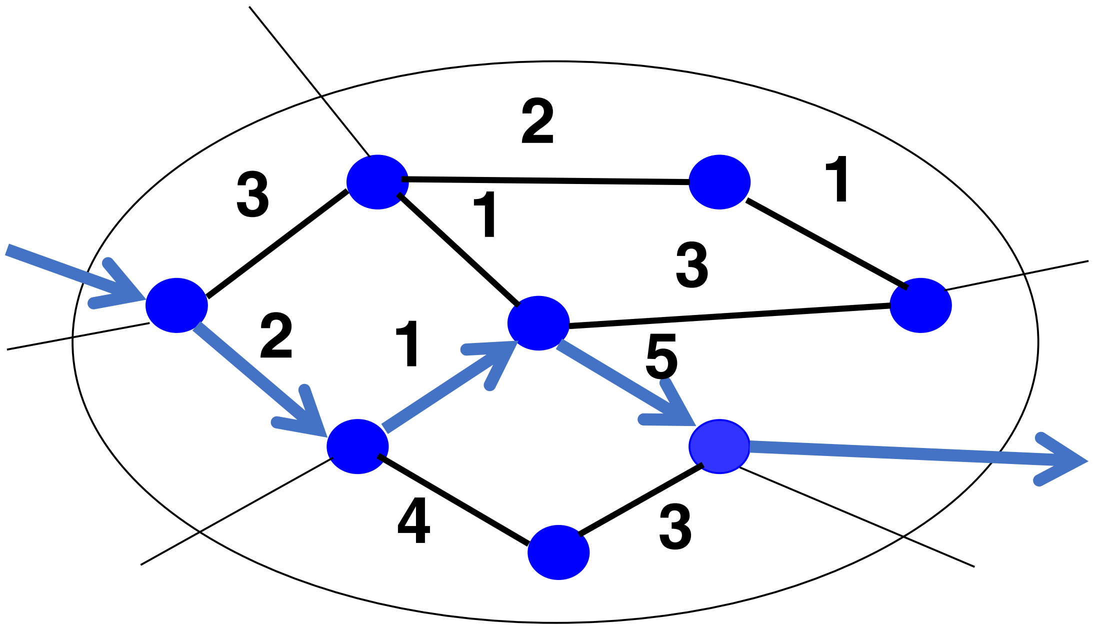
Traffic Engineering Problem

- Management plane: setting the weights
  - Inversely proportional to link capacity?
  - Proportional to propagation delay?
  - Network-wide optimization based on traffic?



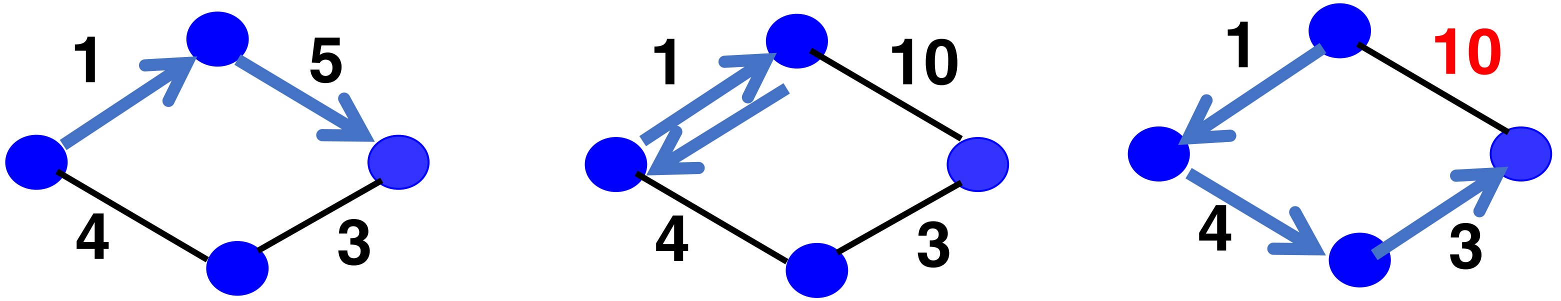
Traffic Engineering: Optimization

- Inputs
  - Network topology
  - Link capacities
  - Traffic matrix
- Output
  - Link weights
- Objective
  - Minimize max-utilized link
  - Or, minimize a sum of link congestion



Transient Routing Disruptions

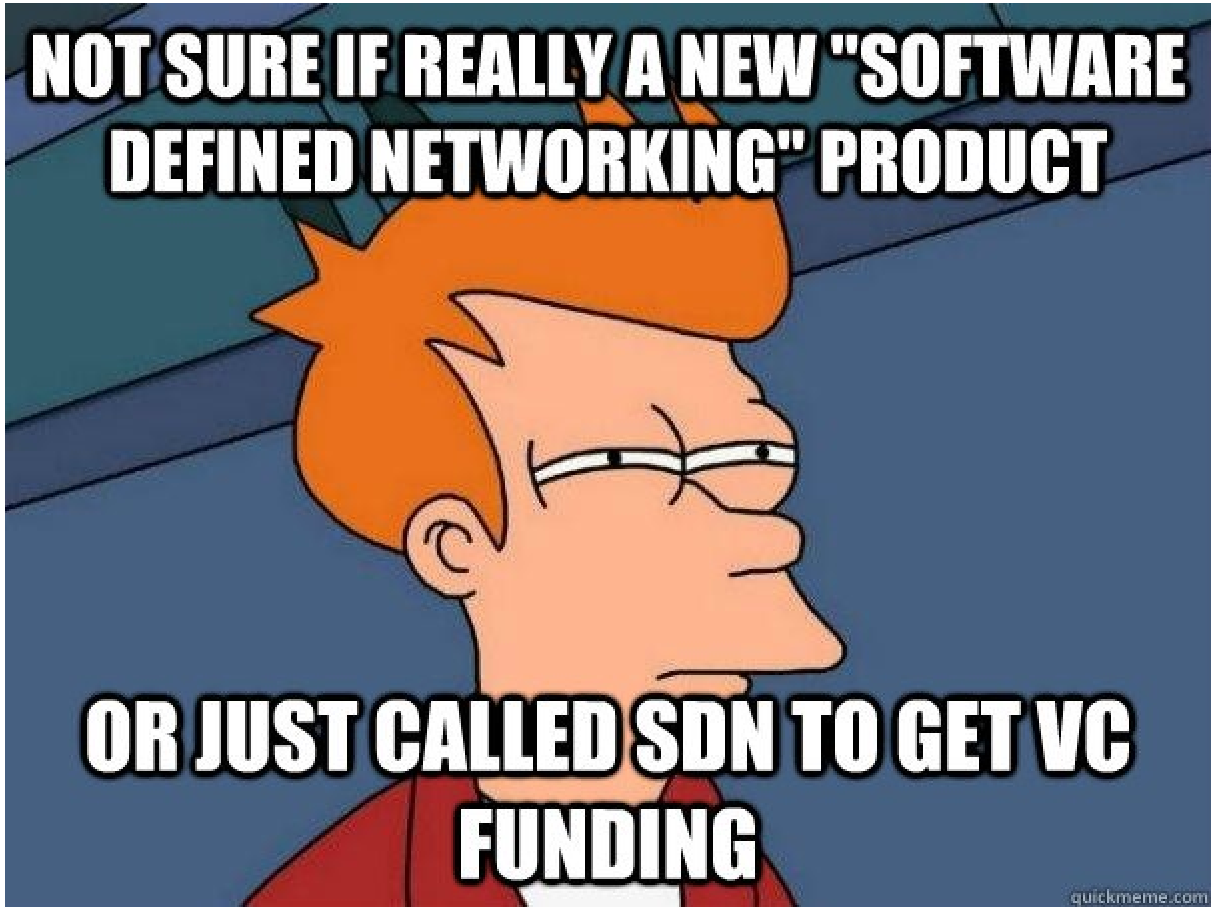
- Topology changes
  - Link weight change
  - Node/link failure or recovery
- Routing convergence
  - Nodes temporarily disagree how to route
  - Leading to transient loops and blackholes



Management Plane Challenges

- Indirect control
  - Changing weights instead of paths
  - Complex optimization problem
- Uncoordinated control
  - Cannot control which router updates first
- Interacting protocols and mechanisms
  - Routing and forwarding
  - Naming and addressing
  - Access control
  - Quality of service
  - …

SDN – Software Defined Networking

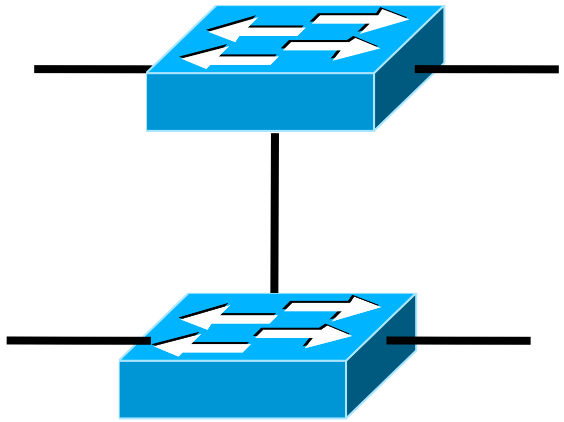


The Internet: A Remarkable Story

- Tremendous success
  - From research experiment to global infrastructure
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Hosts: arbitrary applications
- Enables innovation in applications
  - Web, P2P, VoIP, social networks, virtual worlds
- But, change is easy only at the edge… ☹

Inside the 'Net: A Different Story…

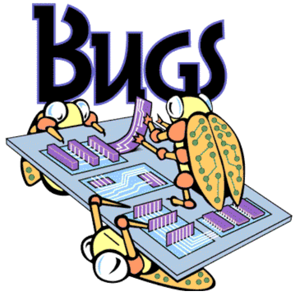
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features



Impacts performance, security, reliability, cost…

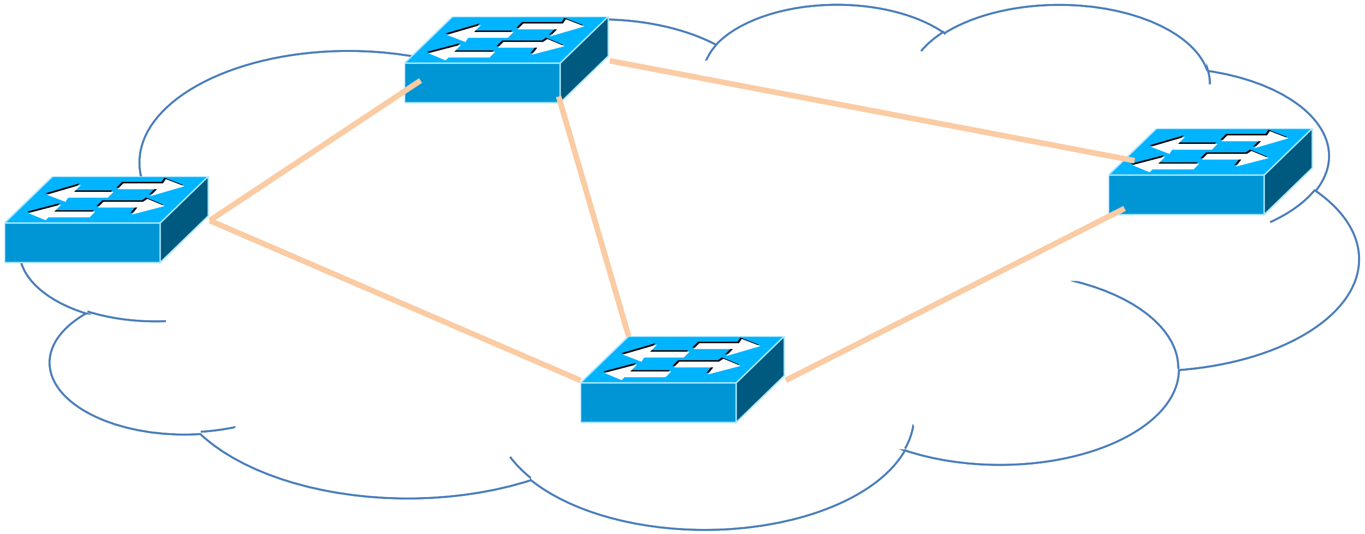
Networks are Hard to Manage

- Operating a network is expensive
  - More than half the cost of a network
  - Yet, operator error causes most outages
- Buggy software in the equipment
  - Routers with 20+ million lines of code
  - Cascading failures, vulnerabilities, etc.
- The network is “in the way”
  - Especially a problem in data centers
  - … and home networks



Rethinking the “Division of Labor”

Traditional Computer Networks

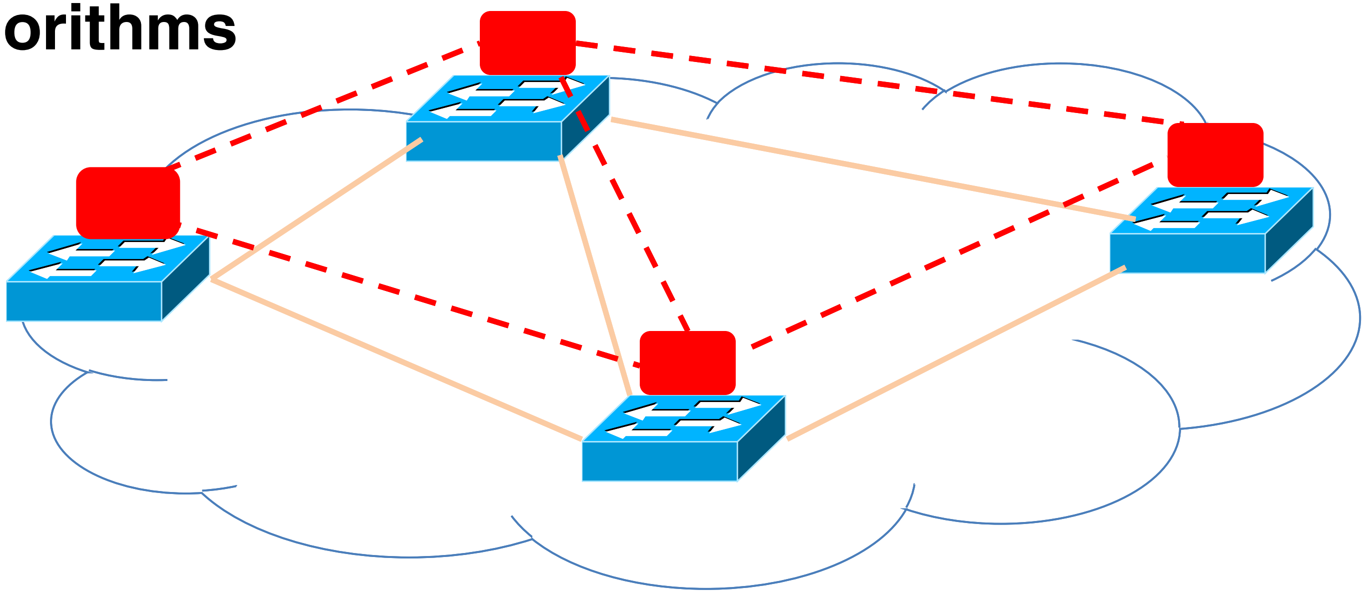


Data plane:
Packet streaming

Forward, filter, buffer, mark, rate-limit, and measure packets

Traditional Computer Networks

Control plane:
Distributed algorithms

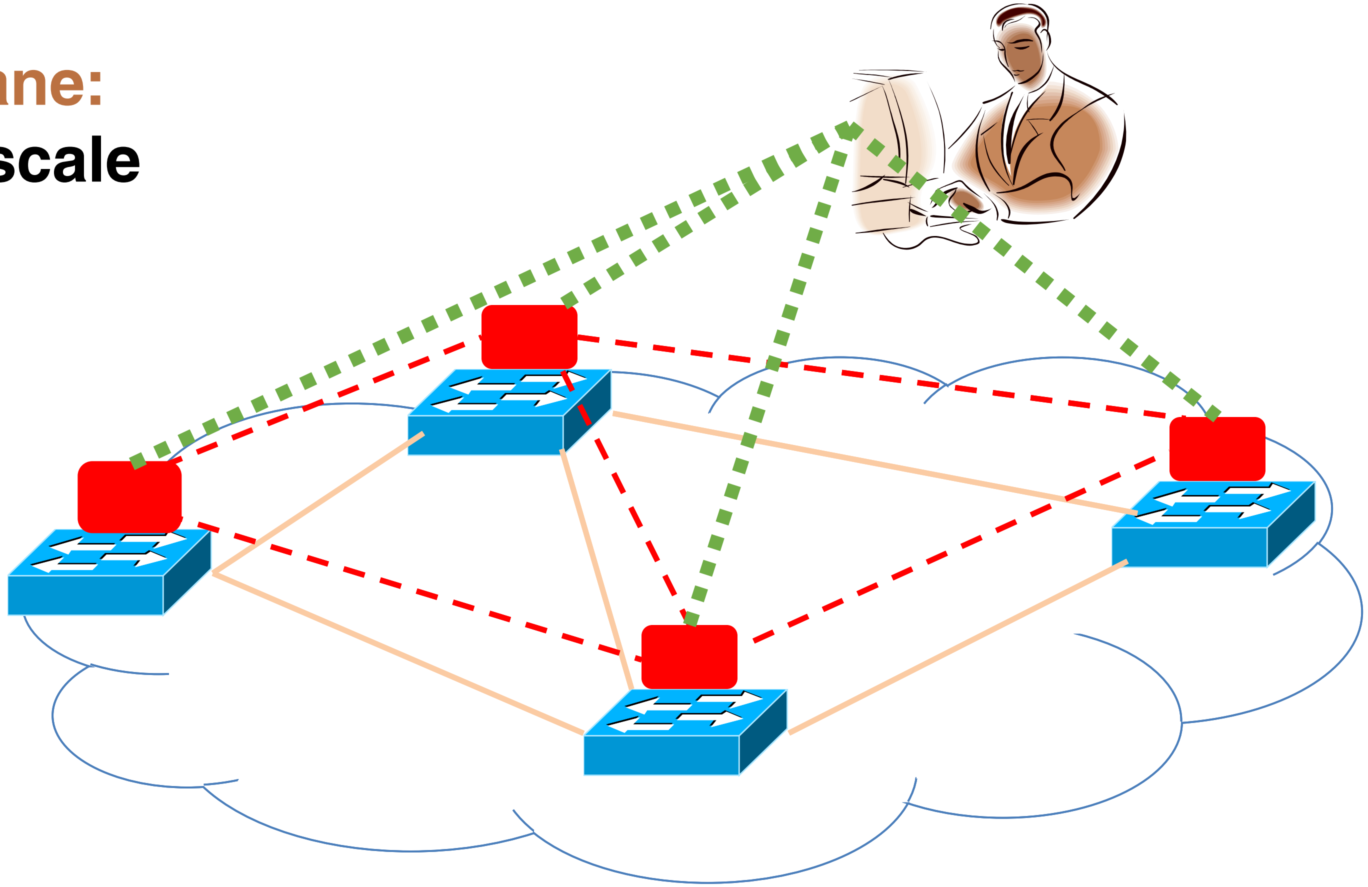


Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks

Management plane:

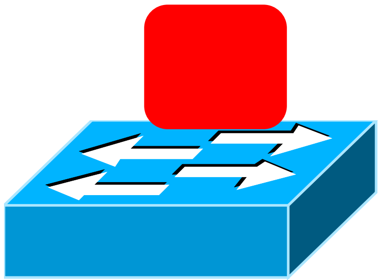
Human time scale



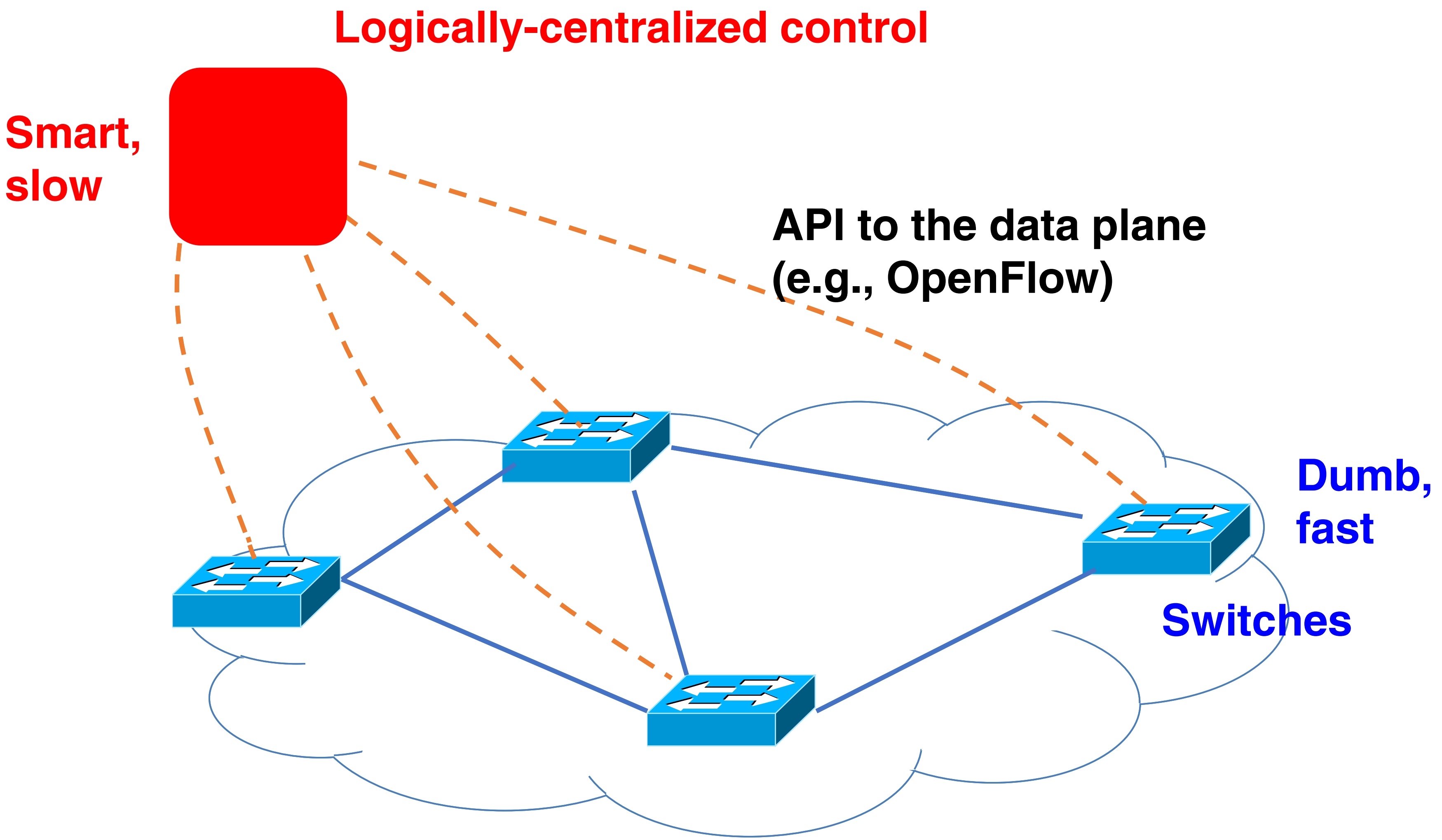
Collect measurements and configure the equipment

Death to the Control Plane!

- Simpler management
  - No need to “invert” control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
- Easier interoperability
  - Compatibility only in “wire” protocols
- Simpler, cheaper equipment
  - Minimal software



Software Defined Networking (SDN)

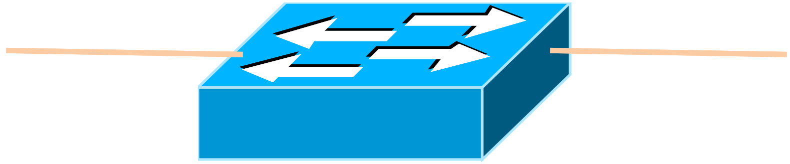


communication networks laboratory
Eötvös Loránd University

OpenFlow Networks

Data-Plane: Simple Packet Handling

- Simple packet-handling rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

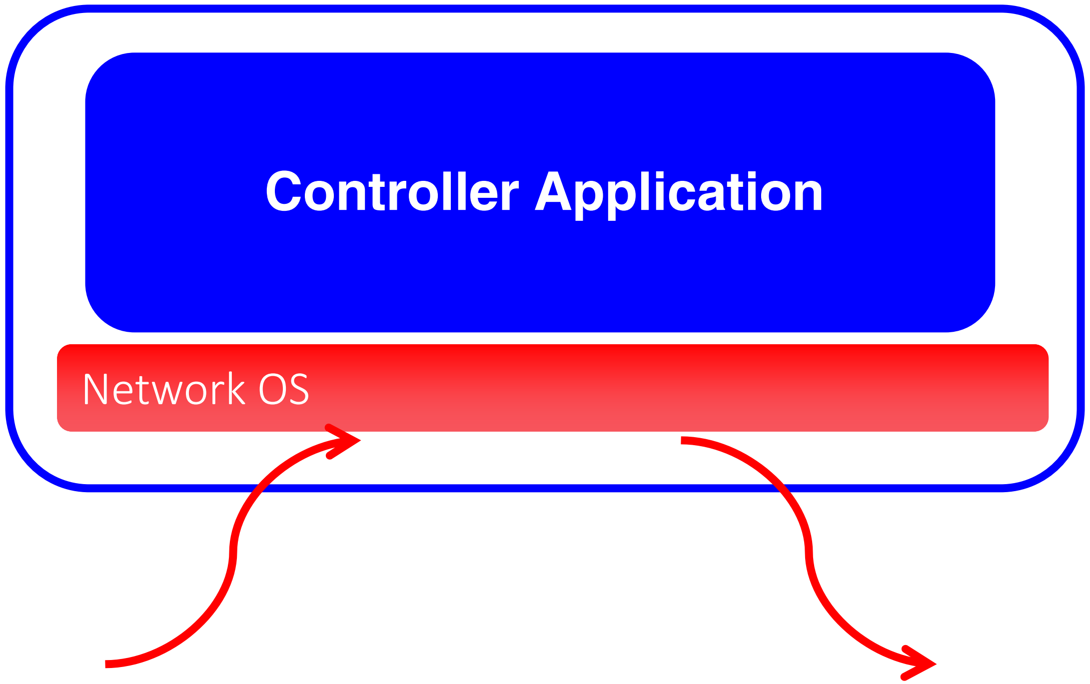


1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

Unifies Different Kinds of Boxes

- Router
  - Match: longest destination IP prefix
  - Action: forward out a link
- Switch
  - Match: destination MAC address
  - Action: forward or flood
- Firewall
  - Match: IP addresses and TCP/UDP port numbers
  - Action: permit or deny
- NAT
  - Match: IP address and port
  - Action: rewrite address and port

Controller: Programmability



Events from switches
Topology changes,
Traffic statistics,
Arriving packets

Commands to switches
(Un)install rules,
Query statistics,
Send packets

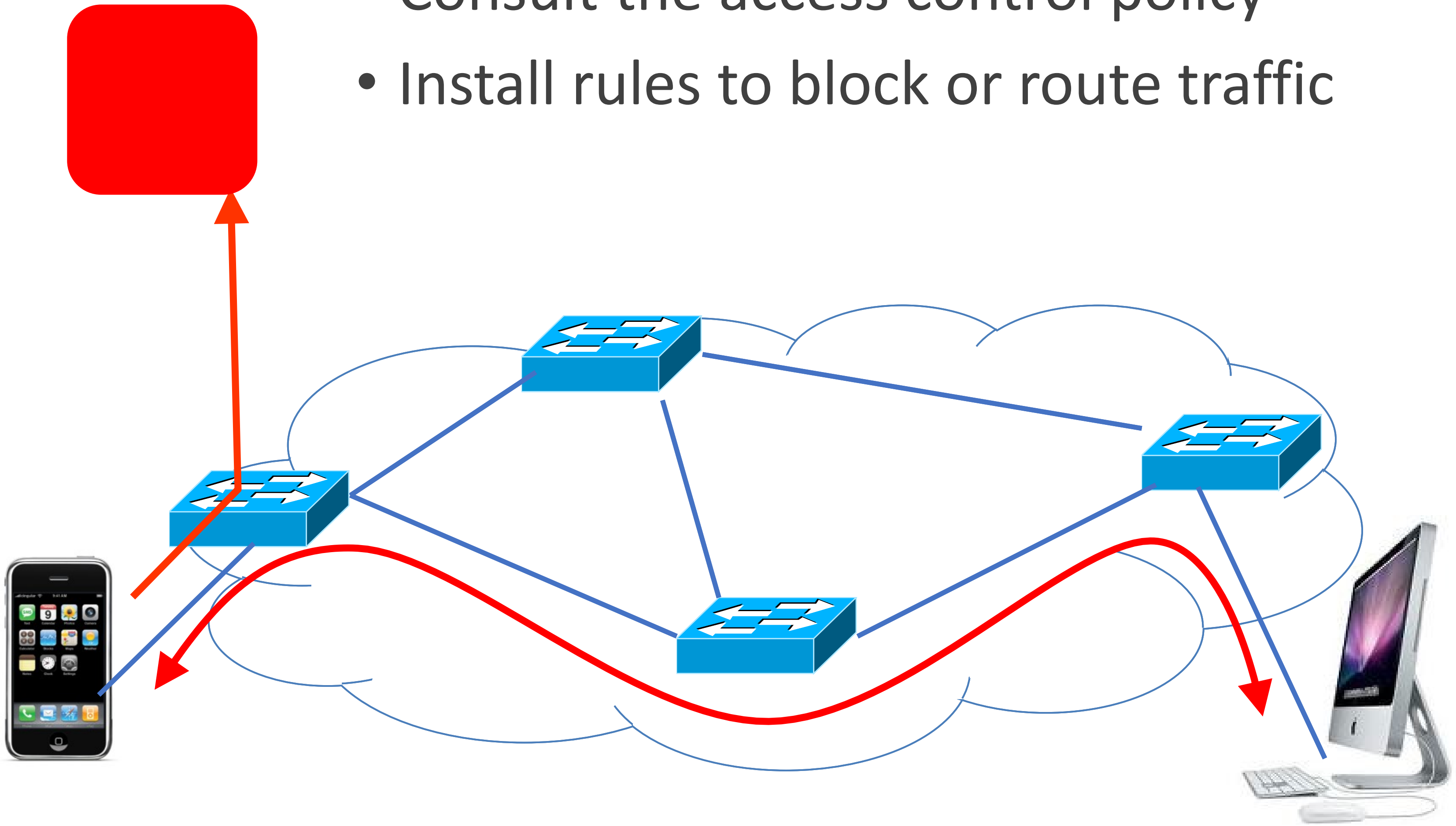
Example OpenFlow Applications

- **Dynamic access control**
- **Seamless mobility/migration**
- **Server load balancing**
- **Network virtualization**
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection

See http://www.openflow.org/videos/

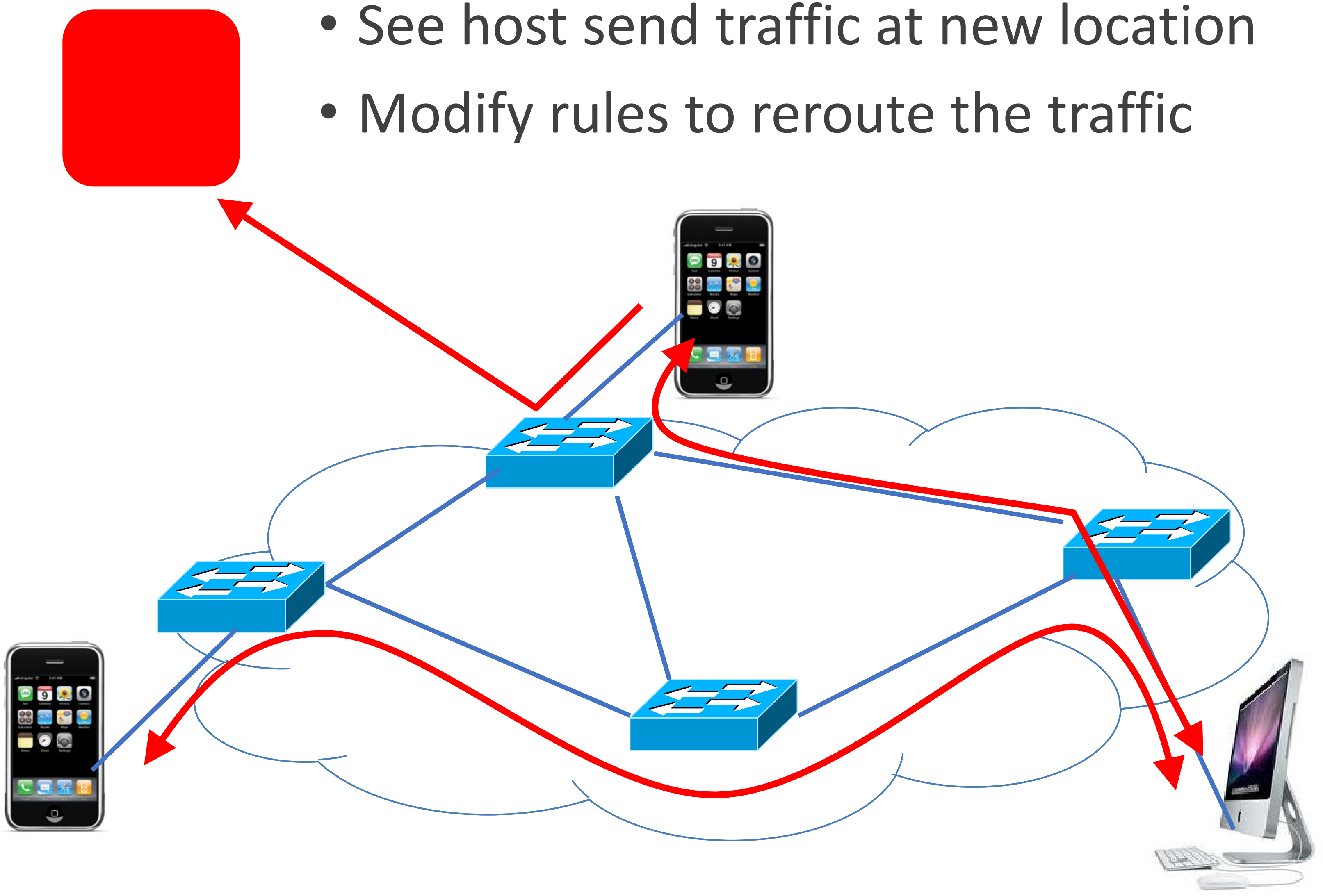
E.g.: Dynamic Access Control

- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



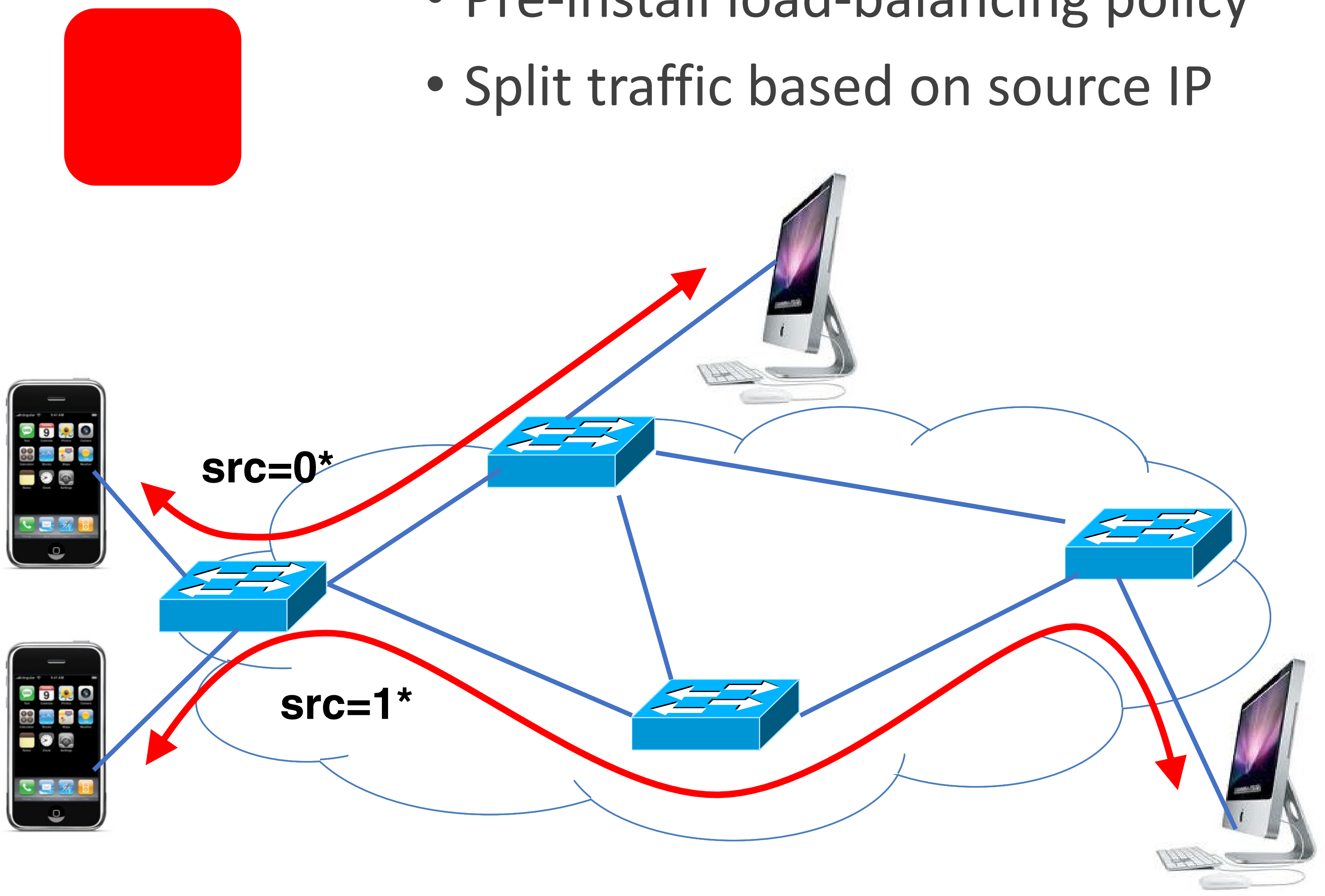
E.g.: Seamless Mobility/Migration

- See host send traffic at new location
- Modify rules to reroute the traffic



E.g.: Server Load Balancing

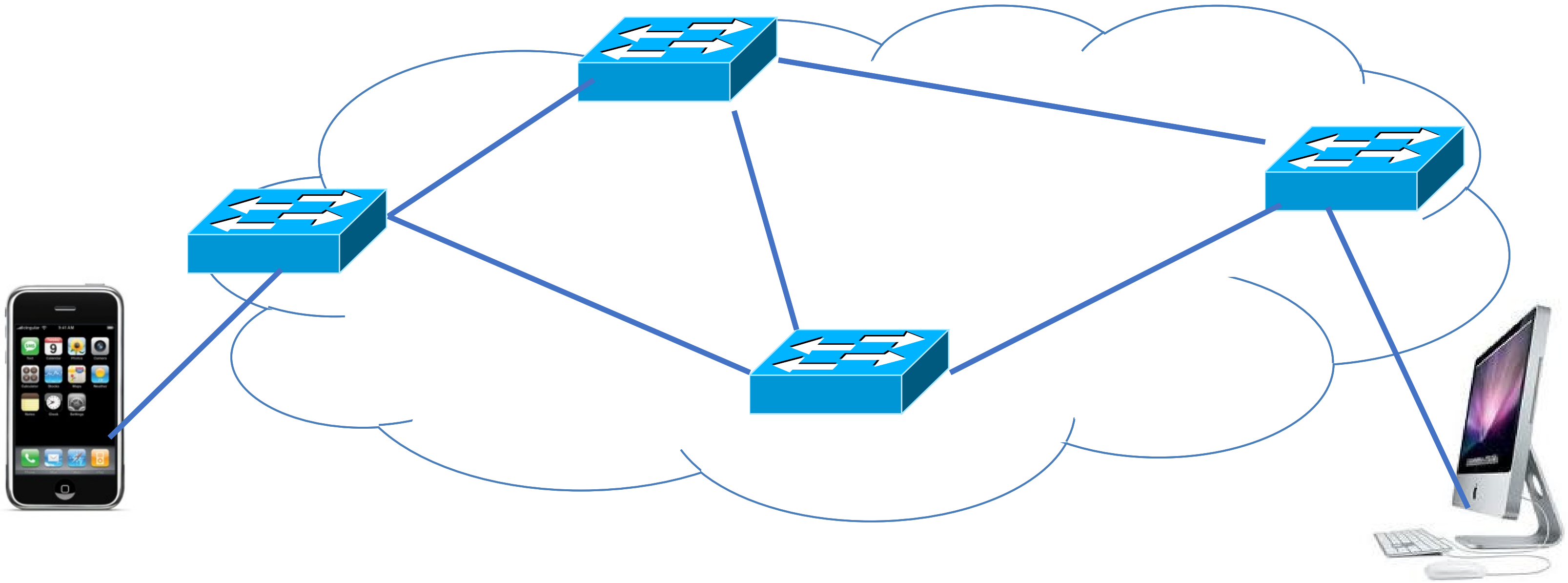
- Pre-install load-balancing policy
- Split traffic based on source IP



E.g.: Network Virtualization

Controller #1 | **Controller #2** | **Controller #3**

Partition the space of packet headers



OpenFlow in the Wild

- Open Networking Foundation
  - Google, Facebook, Microsoft, Yahoo, Verizon, Deutsche Telekom, and many other companies
- Commercial OpenFlow switches
  - HP, NEC, Quanta, Dell, IBM, Juniper, …
- Network operating systems
  - NOX, Beacon, Floodlight, Nettle, ONIX, POX, Frenetic
- Network deployments
  - Eight campuses, and two research backbone networks
  - Commercial deployments (e.g., Google backbone)

Eötvös Loránd University

A Helpful Analogy

From Nick McKeown’s talk “Making SDN Work” at the Open Networking Summit, April 2012

Mainframes

Specialized Applications

Specialized Operating System

Specialized Hardware

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

App

Open Interface

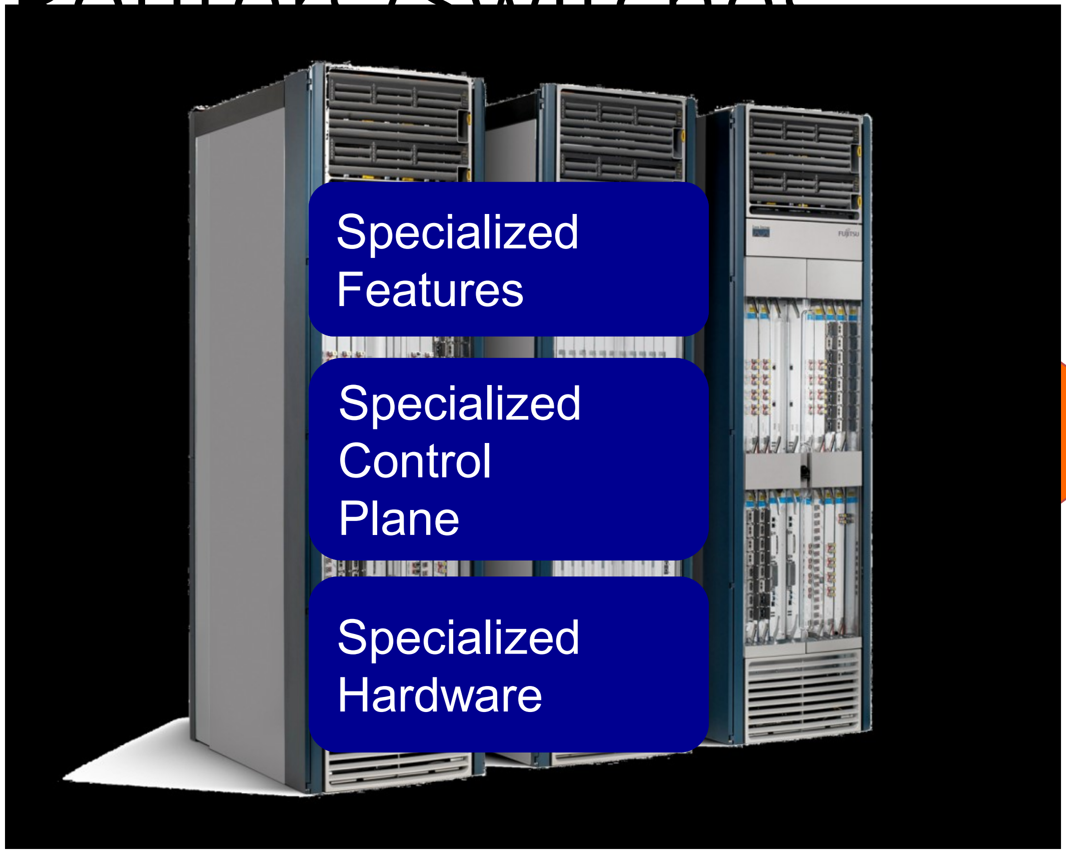
Windows (OS) **or** Linux **or** Mac OS

Open Interface

Microprocessor

Horizontal
Open interfaces
Rapid innovation
Huge industry

Routers/Switches



App

— Open Interface —

Control Plane or Control Plane or Control Plane

— Open Interface —

Merchant Switching Chips

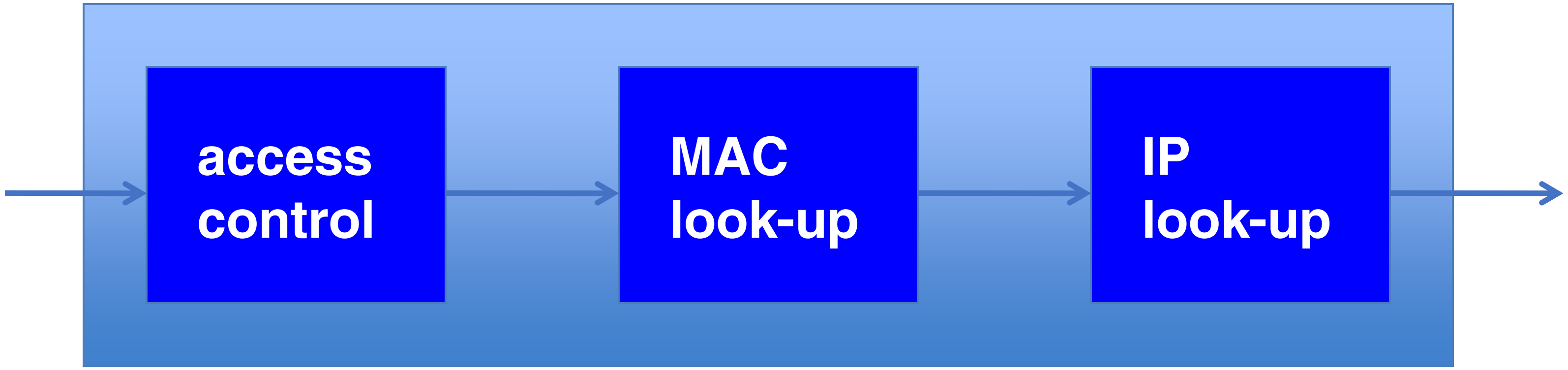
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Challenges

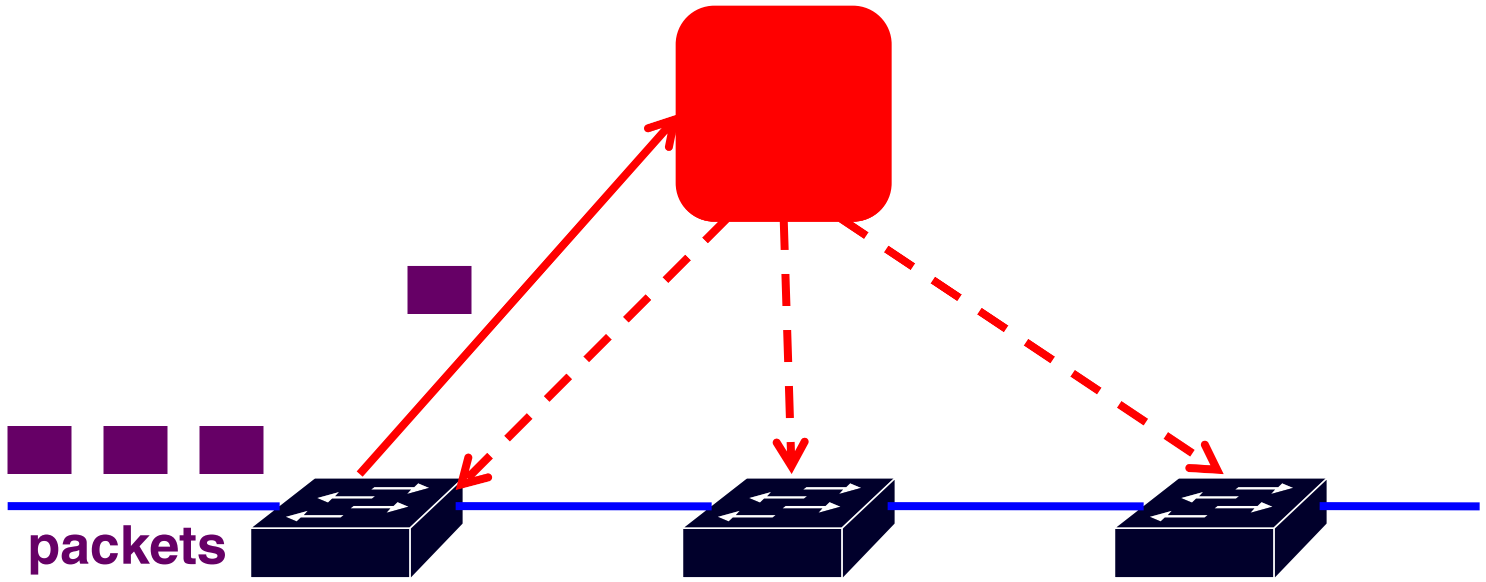
Heterogeneous Switches

- Number of packet-handling rules
- Range of matches and actions
- Multi-stage pipeline of packet processing
- Offload some control-plane functionality (?)

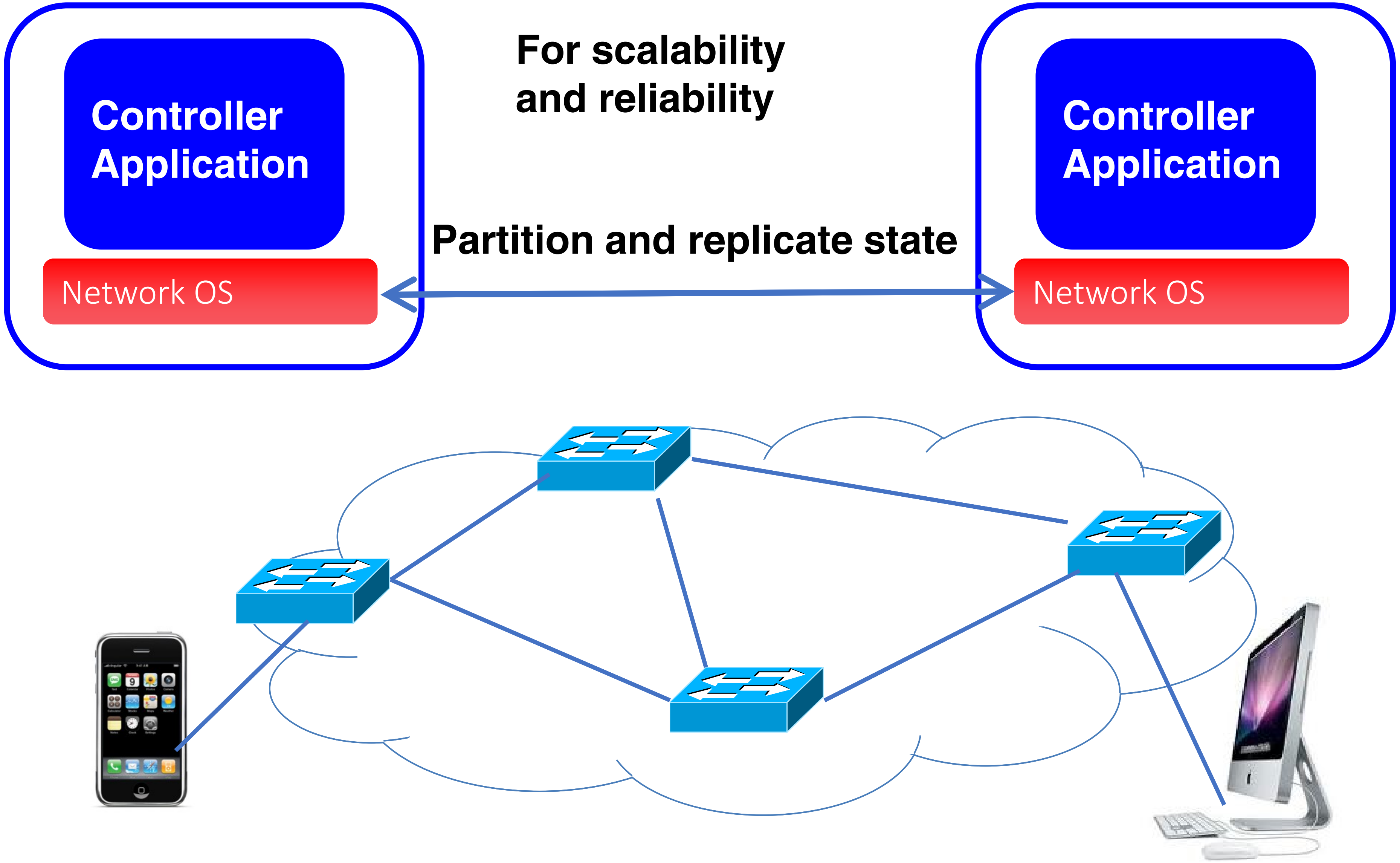


Controller Delay and Overhead

- Controller is much slower the the switch
- Processing packets leads to delay and overhead
- Need to keep most packets in the "fast path"



Distributed Controller

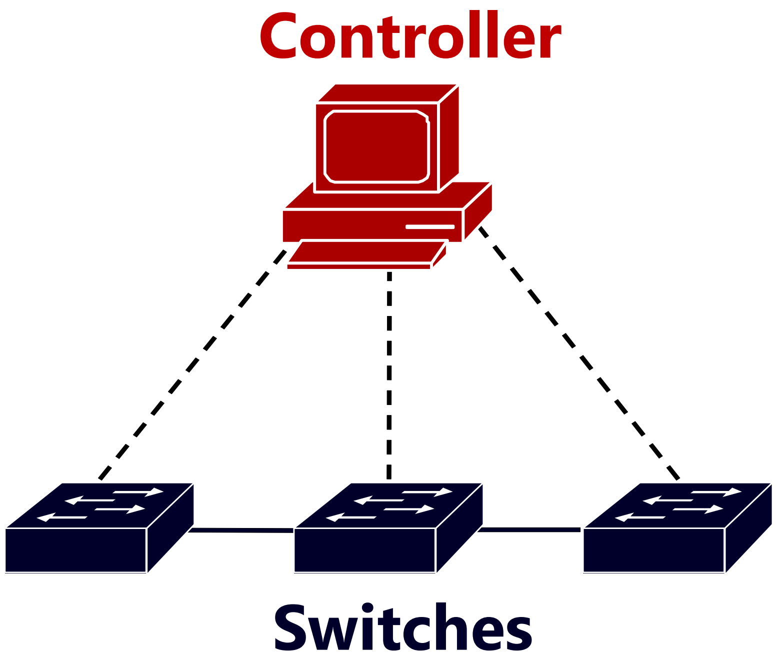


Testing and Debugging

- OpenFlow makes programming possible
  - Network-wide view at controller
  - Direct control over data plane
- Plenty of room for bugs
  - Still a complex, distributed system
- Need for testing techniques
  - Controller applications
  - Controller and switches
  - Rules installed in the switches

Programming Abstractions

- Controller APIs are low-level
  - Thin veneer on the underlying hardware
- Need better languages
  - Composition of modules
  - Managing concurrency
  - Querying network state
  - Network-wide abstractions
- Ongoing at Princeton
  - http://www.frenetic-lang.org/



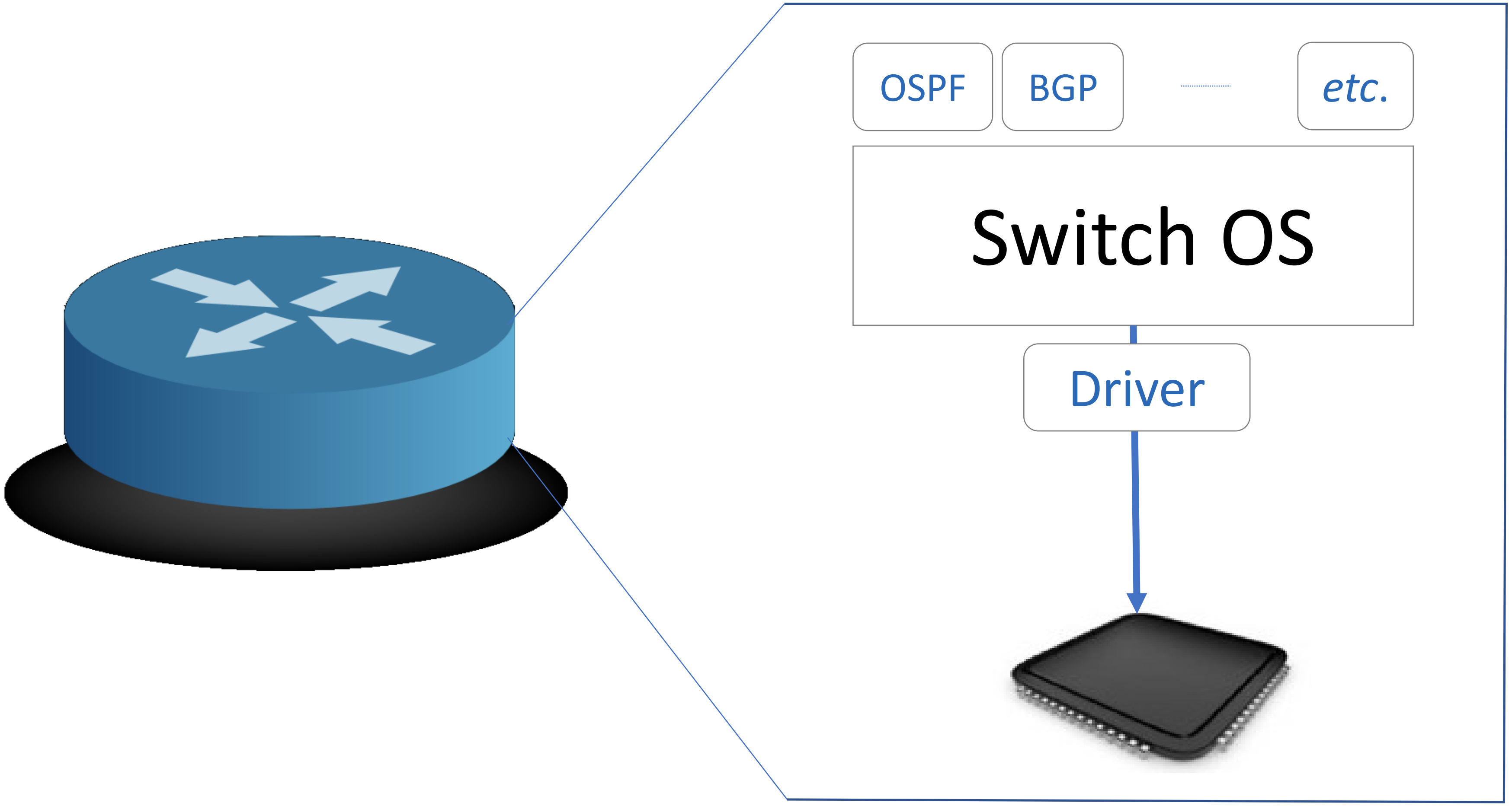
<u>Deep</u> programmability

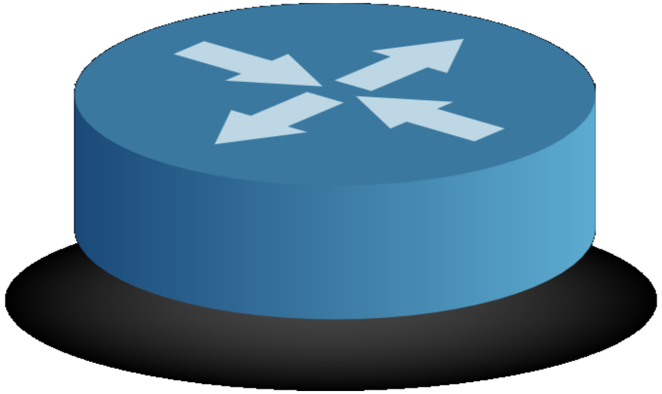
Can OpenFlow solve all the problems of networks?

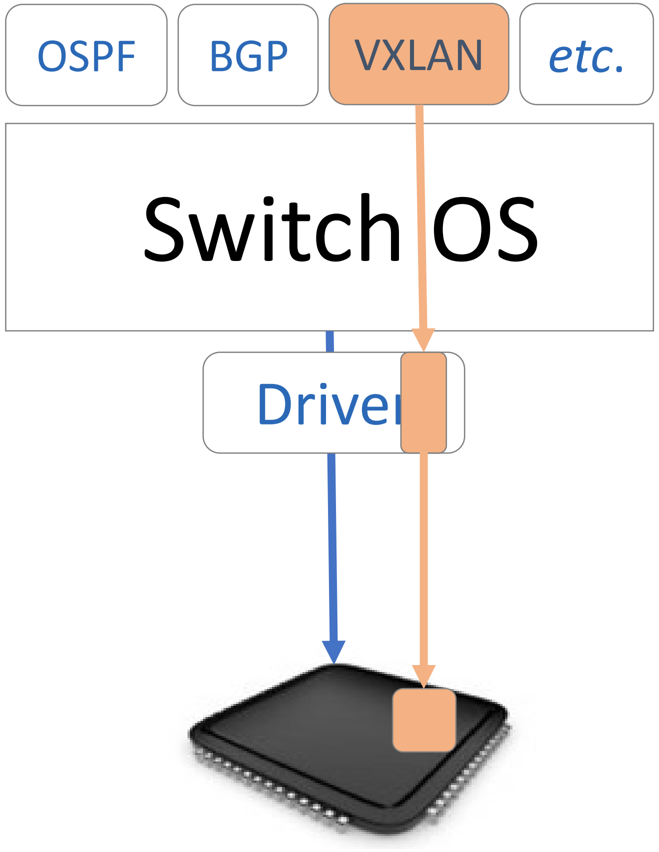
Well… no.

OpenFlow is only the first step…

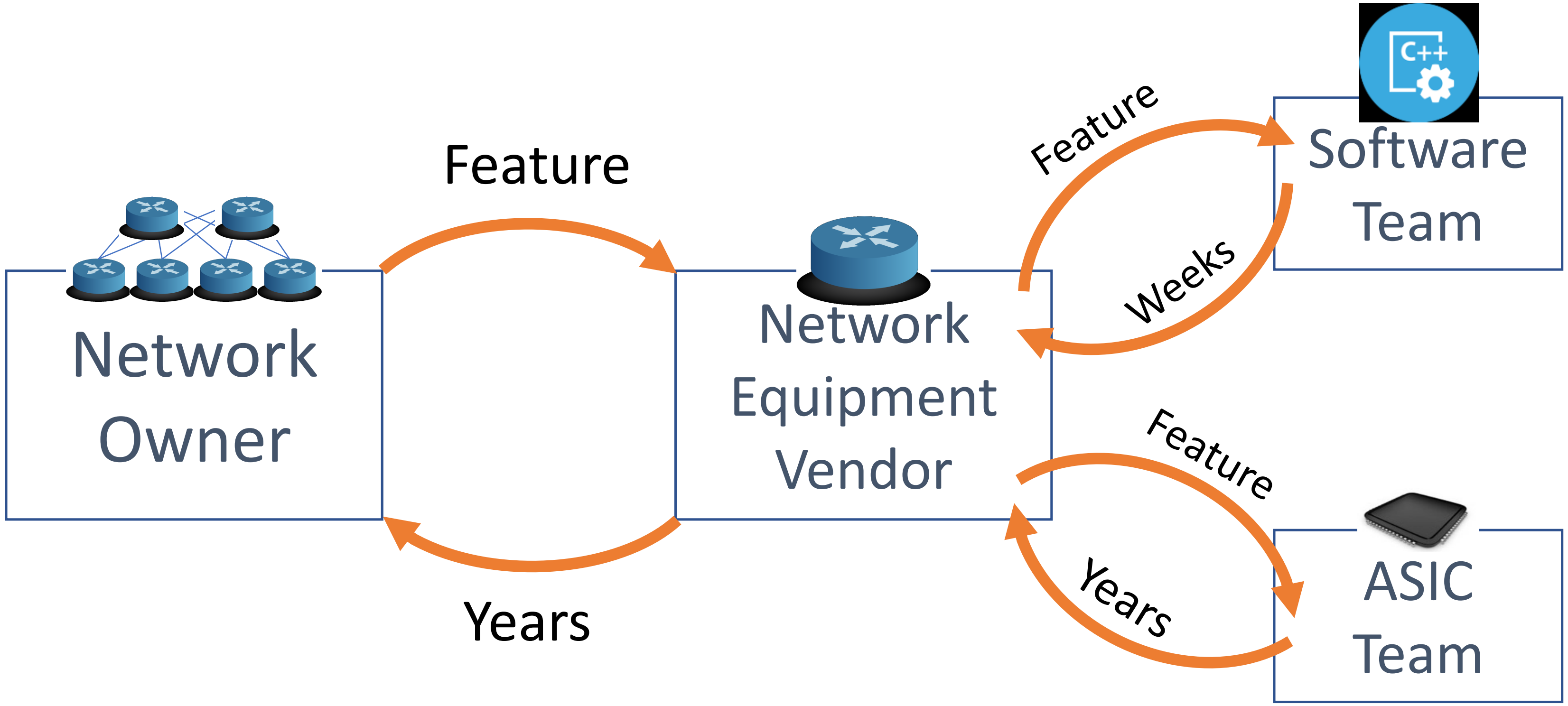
- Advantages
  - Opening up the data planes by providing an open vendor-independent API
  - Control plane can manage data plane devices through this API
- Disadvantages
  - The protocol and the specification are too complex
    - Switches must support complicated parsers and pipelines
    - Extra features make the software agent more complicated
  - Only supports a set of existing protocols
    - Not protocol independent
- Consequences
  - Parts of spec are implemented by switch vendors
    - **Breaking the abstraction of** one API to ***rule-them-all***


OSPF
BGP
etc.
Switch OS
Driver




OSPF
BGP
VXLAN
etc.
Switch OS
Driver

Development cycle of a new network feature

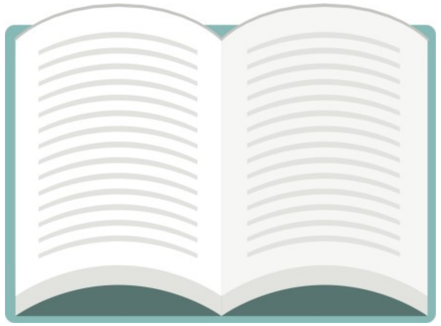


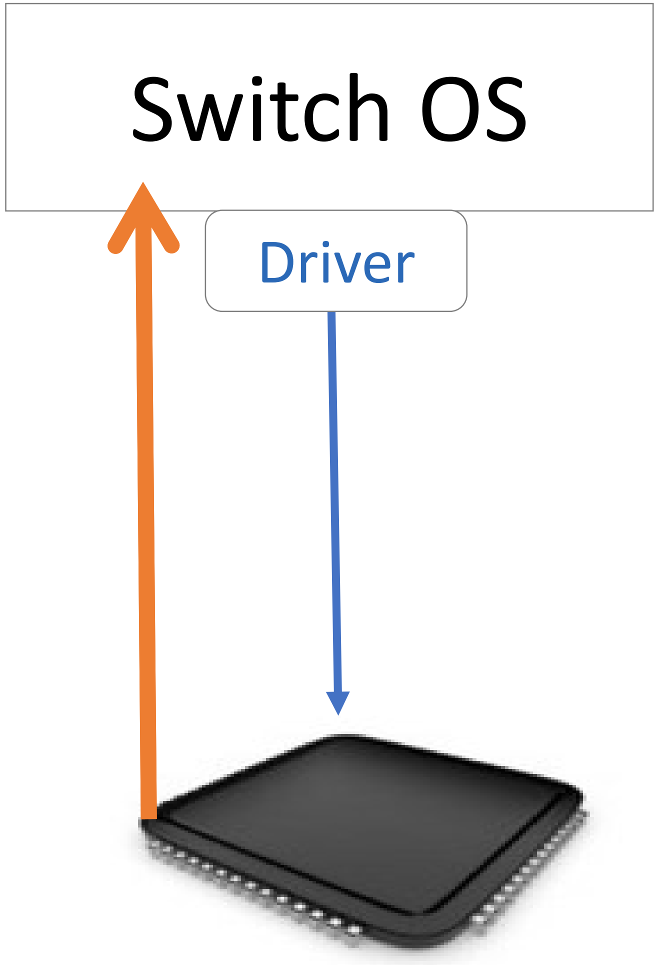
When you need a new feature…

1. Equipment vendor can’t just send you a software upgrade
2. New forwarding features take years to develop
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 “bottoms-up”

“This is how I process packets …”





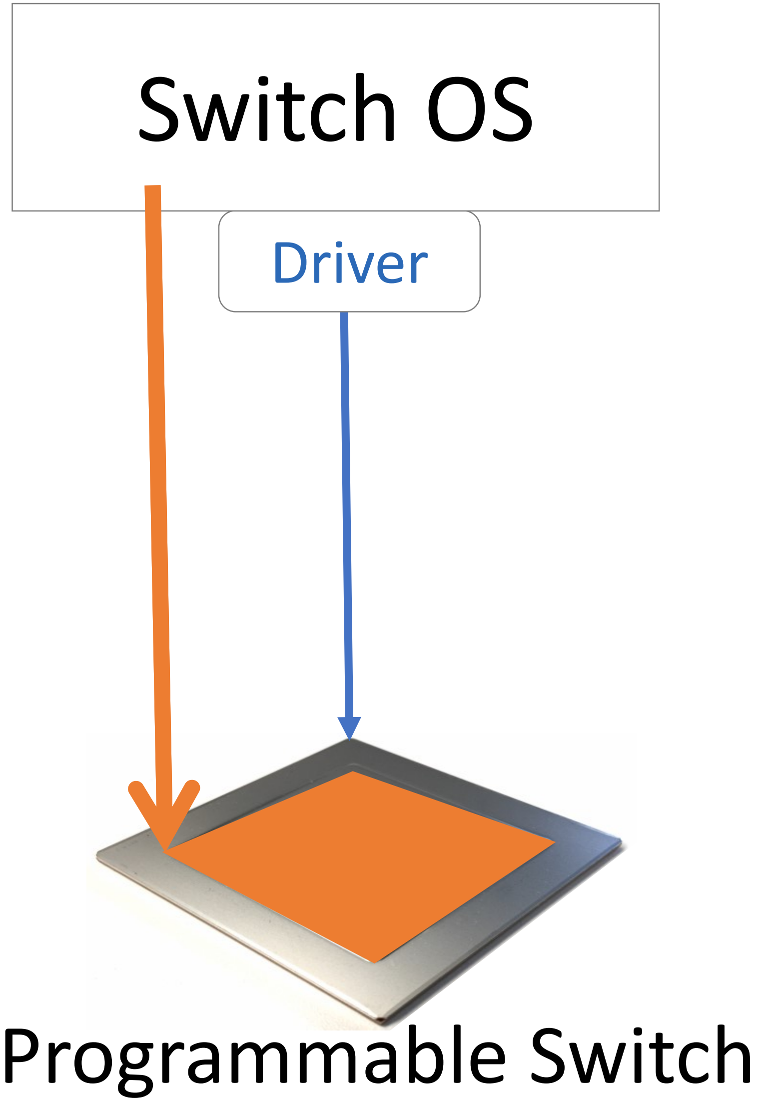
Fixed-function switch

Network systems are starting to be programmed "top-down"

"This is precisely how you must process packets"

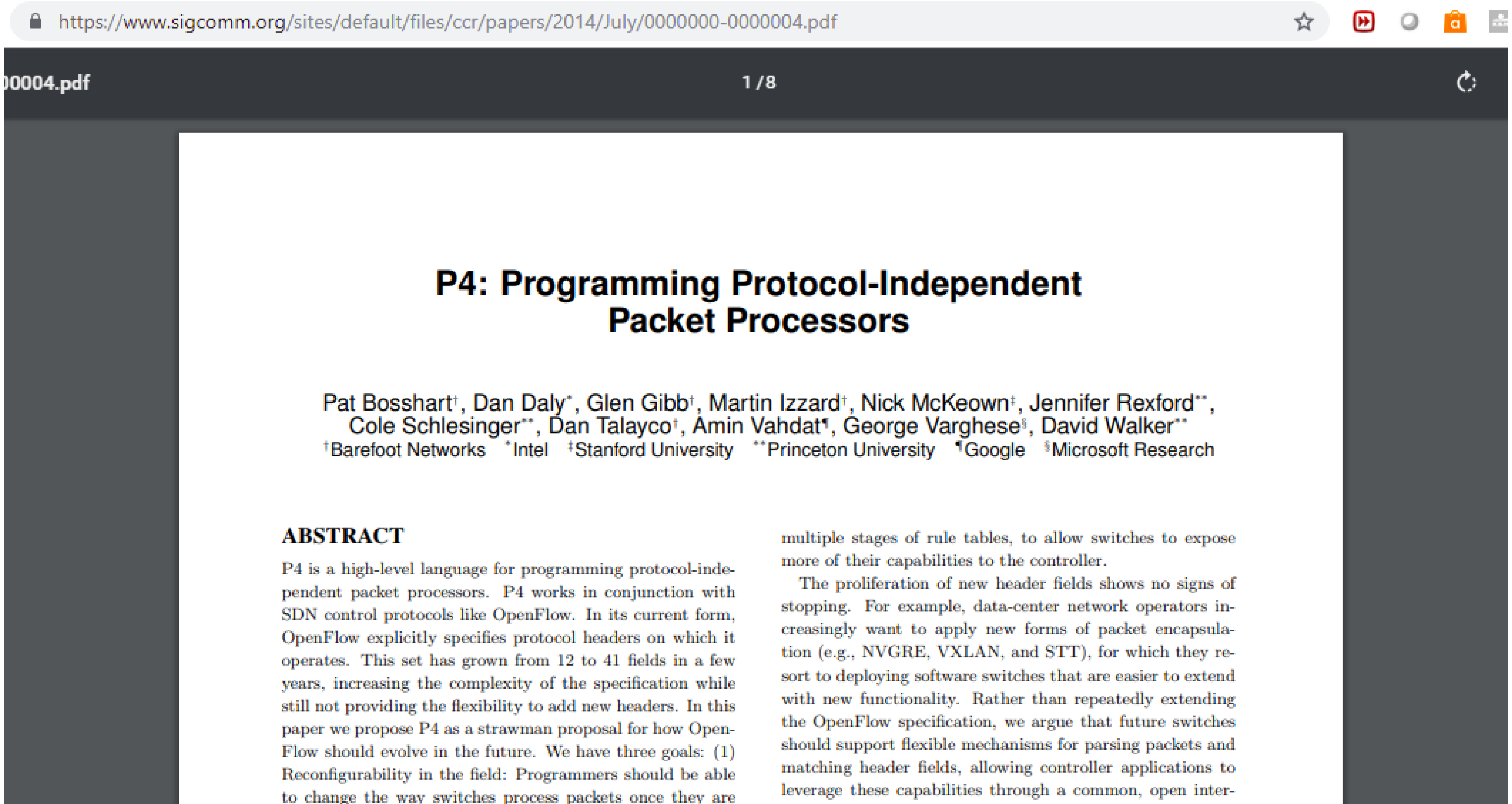
```
table int_table {
  reads {
    ip.protocol;
  }
  actions {
    export_queue_latency;
  }
}
```

```
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);
}
```



P4.org

https://www.sigcomm.org/sites/default/files/ccr/papers/2014/July/0000000-0000004.pdf



P4: Programming Protocol-Independent Packet Processors

Pat Bosshart†, Dan Daly*, Glen Gibb†, Martin Izzard†, Nick McKeown‡, Jennifer Rexford**, Cole Schlesinger**, Dan Talayco†, Amin Vahdat¶, George Varghese§, David Walker**
†Barefoot Networks *Intel ‡Stanford University **Princeton University ¶Google §Microsoft Research

ABSTRACT

P4 is a high-level language for programming protocol-independent packet processors. P4 works in conjunction with SDN control protocols like OpenFlow. In its current form, OpenFlow explicitly specifies protocol headers on which it operates. This set has grown from 12 to 41 fields in a few years, increasing the complexity of the specification while still not providing the flexibility to add new headers. In this paper we propose P4 as a strawman proposal for how OpenFlow should evolve in the future. We have three goals: (1) Reconfigurability in the field: Programmers should be able to change the way switches process packets once they are

multiple stages of rule tables, to allow switches to expose more of their capabilities to the controller.

The proliferation of new header fields shows no signs of stopping. For example, data-center network operators increasingly want to apply new forms of packet encapsulation (e.g., NVGRE, VXLAN, and STT), for which they resort to deploying software switches that are easier to extend with new functionality. Rather than repeatedly extending the OpenFlow specification, we argue that future switches should support flexible mechanisms for parsing packets and matching header fields, allowing controller applications to leverage these capabilities through a common, open inter-

Domain Specific Processors

Computers

Java

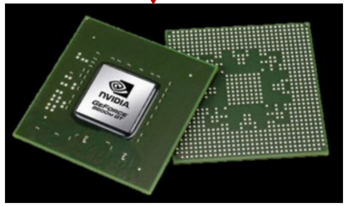
Compiler

CPU

Graphics

OpenCL

Compiler



GPU

Signal Processing

Matlab

Compiler

DSP

Machine Learning

TensorFlow

Compiler

TPU

>>>

Networking

Language

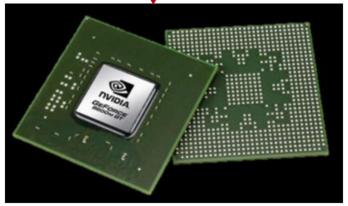
Compiler

?

Domain Specific Processors

Computers	Graphics	Signal Processing	Machine Learning		Networking
Java	OpenCL	Matlab	TensorFlow	>>>	P4
Compiler	Compiler	Compiler	Compiler		Compiler

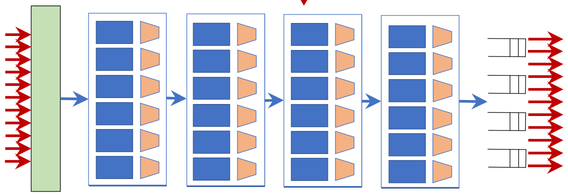
CPU



GPU

DSP

TPU

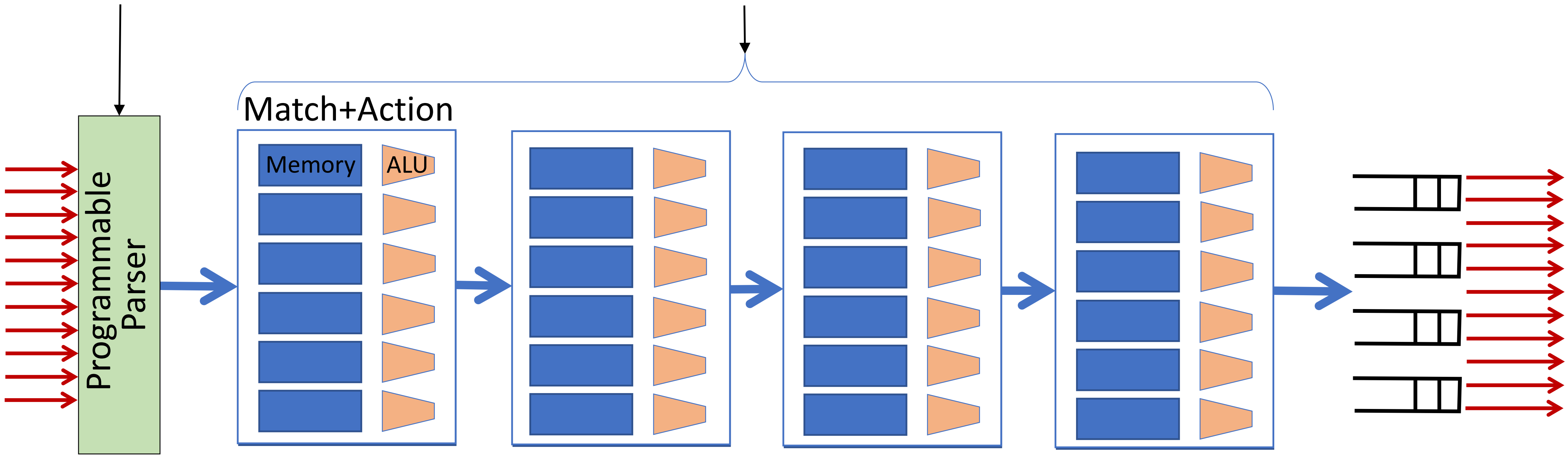


PISA

PISA: Protocol Independent Switch Architecture

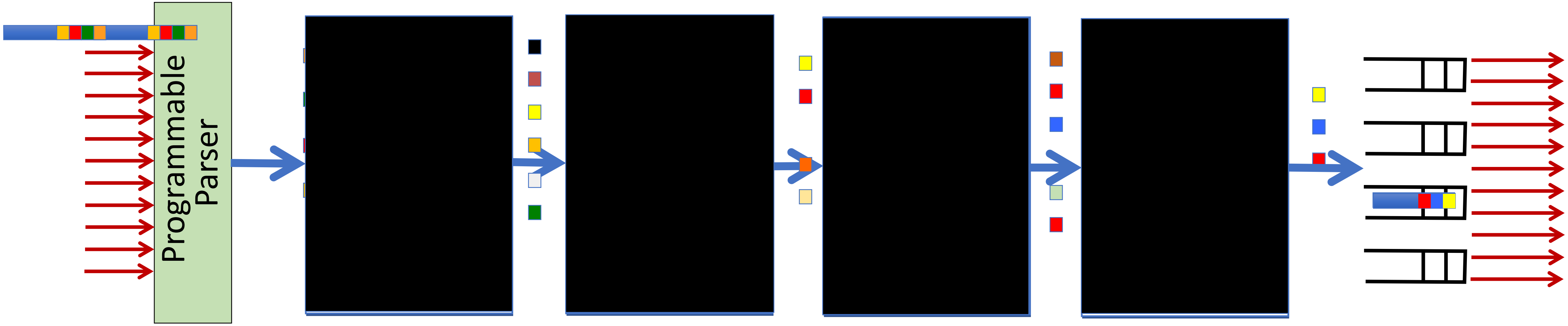
Programmer declares which headers are recognized

Programmer declares what tables are needed and how packets are processed



All stages are identical – makes PISA a good "compiler target"

PISA: Protocol Independent Switch Architecture



Tofino 6.5Tb/s Switch

December 2016

65 x 100GE (or 260 x 25GE)

Same power and cost as fixed-function switches.

How programmability is being used

❶ Reducing complexity

Reducing complexity

switch.p4

Switch OS

IPv4 and IPv6 routing
- Unicast Routing
  - Routed Ports & SVI
  - VRF
- Unicast RPF
  - Strict and Loose
- ~~Multicast~~
  - ~~PIM-SM/DM & PIM-Bidir~~

Ethernet switching
- ~~VLAN Flooding~~
- MAC Learning & Aging
- STP state
- ~~VLAN Translation~~

Load balancing
- ~~LAG~~
- ECMP & WCMP
- Resilient Hashing
- ~~Flowlet Switching~~

Fast Failover
– LAG & ECMP

Tunneling
- IPv4 and IPv6 Routing & Switching
  - ~~IP in IP (6in4, 4in4)~~
  - VXLAN, NVGRE, GENEVE & GRE
  - ~~Segment Routing, ILA~~

~~MPLS~~
- ~~LER and LSR~~
- ~~IPv4/v6 routing (L3VPN)~~
- ~~L2 switching (EoMPLS, VPLS)~~
- ~~MPLS over UDP/GRE~~

ACL
- MAC ACL, IPv4/v6 ACL, RACL
- ~~QoS ACL, System ACL, PBR~~
- Port Range lookups in ACLs

QOS
- QoS Classification & marking
- ~~Drop profiles/WRED~~
- ~~RoCE v2 & FCoE~~
- CoPP (Control plane policing)

~~NAT and L4 Load Balancing~~

Security Features
- ~~Storm Control, IP Source Guard~~

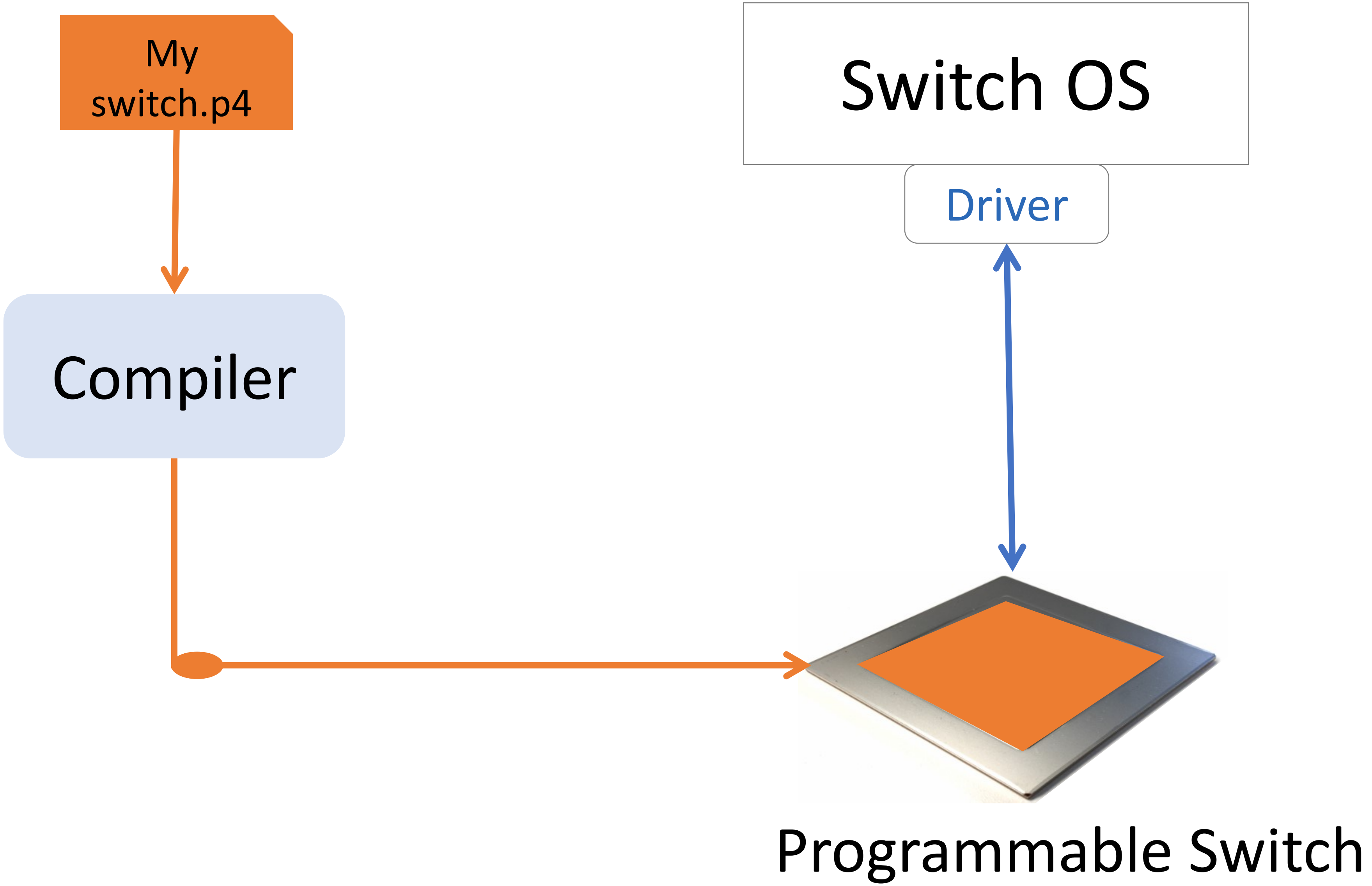
Monitoring & Telemetry
- ~~Ingress Mirroring and Egress Mirroring~~
- Negative Mirroring
- ~~Sflow~~
- INT

Counters
- Route Table Entry Counters
- ~~VLAN/Bridge Domain Counters~~
- Port/Interface Counters

Protocol Offload
- BFD, OAM

Multi-chip Fabric Support
- ~~Forwarding, QOS~~

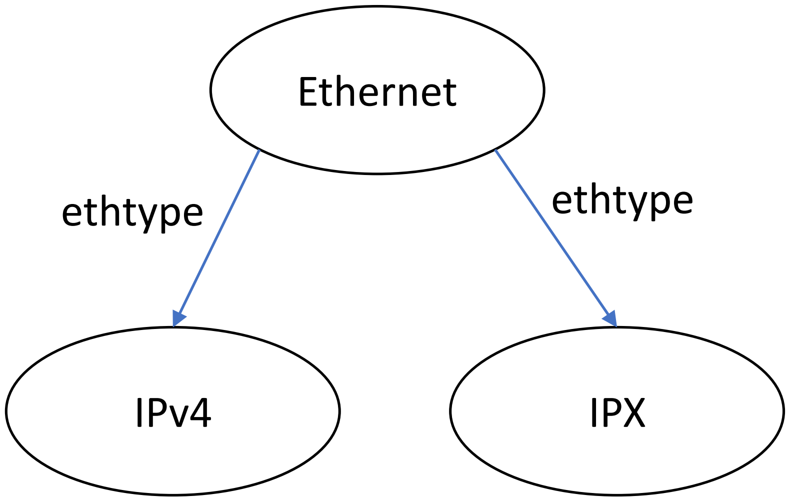
Reducing complexity



How programmability is being used

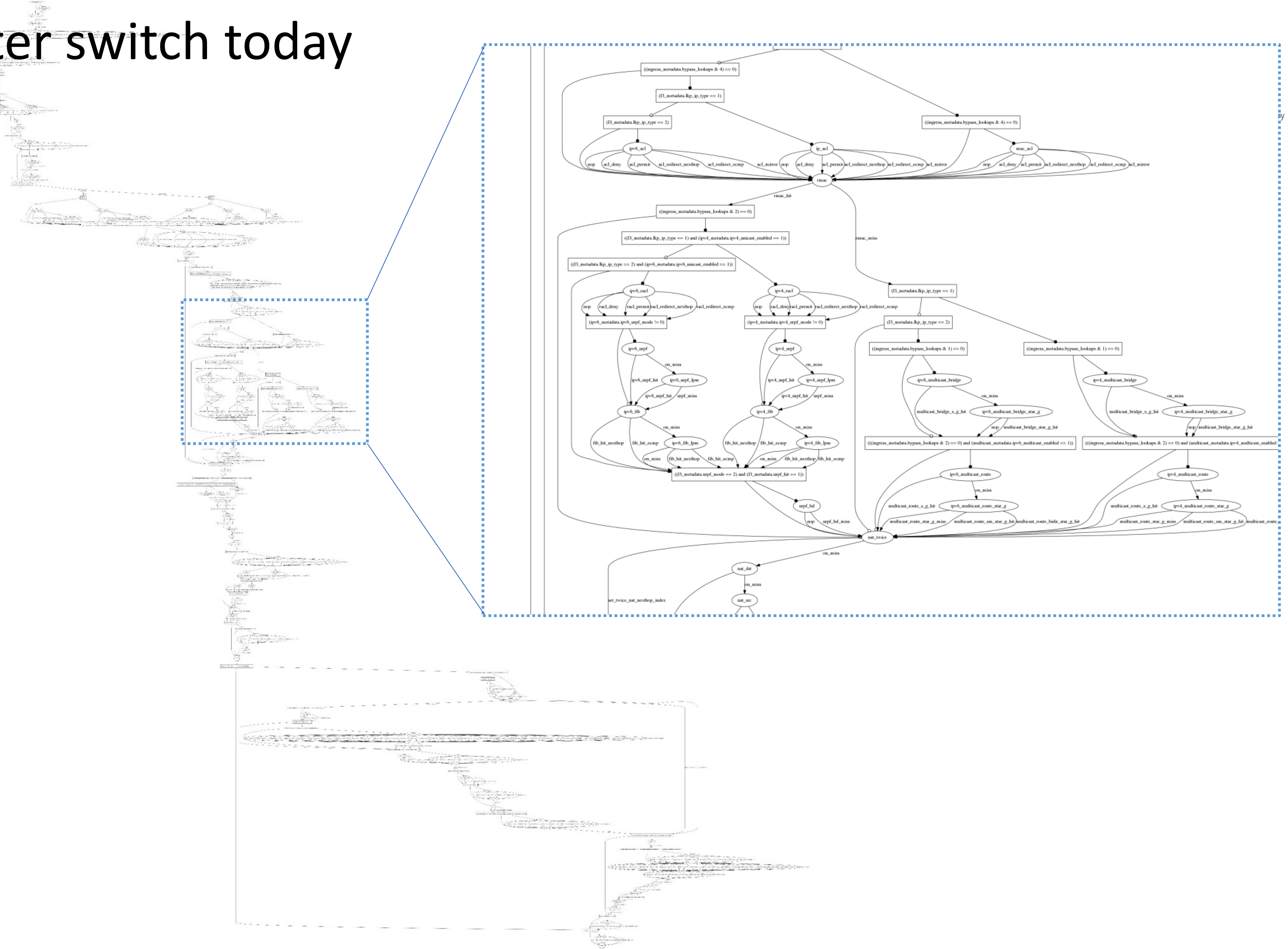
❷ Adding proprietary features

Protocol complexity 20 years ago



Datacenter switch today

switch.p4



Adding features: Some examples so far

1. New encapsulations and tunnels
2. New ways to tag packets for special treatment
3. New approaches to routing: e.g. source routing in MSDCs
4. New approaches to congestion control
5. New ways to process packets: e.g. processing ticker-symbols

New applications: Some examples so far

1. Layer-4 Load Balancer[1]
   - Replace 100 servers or 10 dedicated boxes with one programmable switch
   - Track and maintain mapping for 5-10 million http flows
2. Fast stateless firewall
   - Add/delete and track 100s of thousands of new connections per second
3. Cache for Key-value store[2]
   - Memcache in-network cache for 100 servers
   - 1-2 billion operations per second

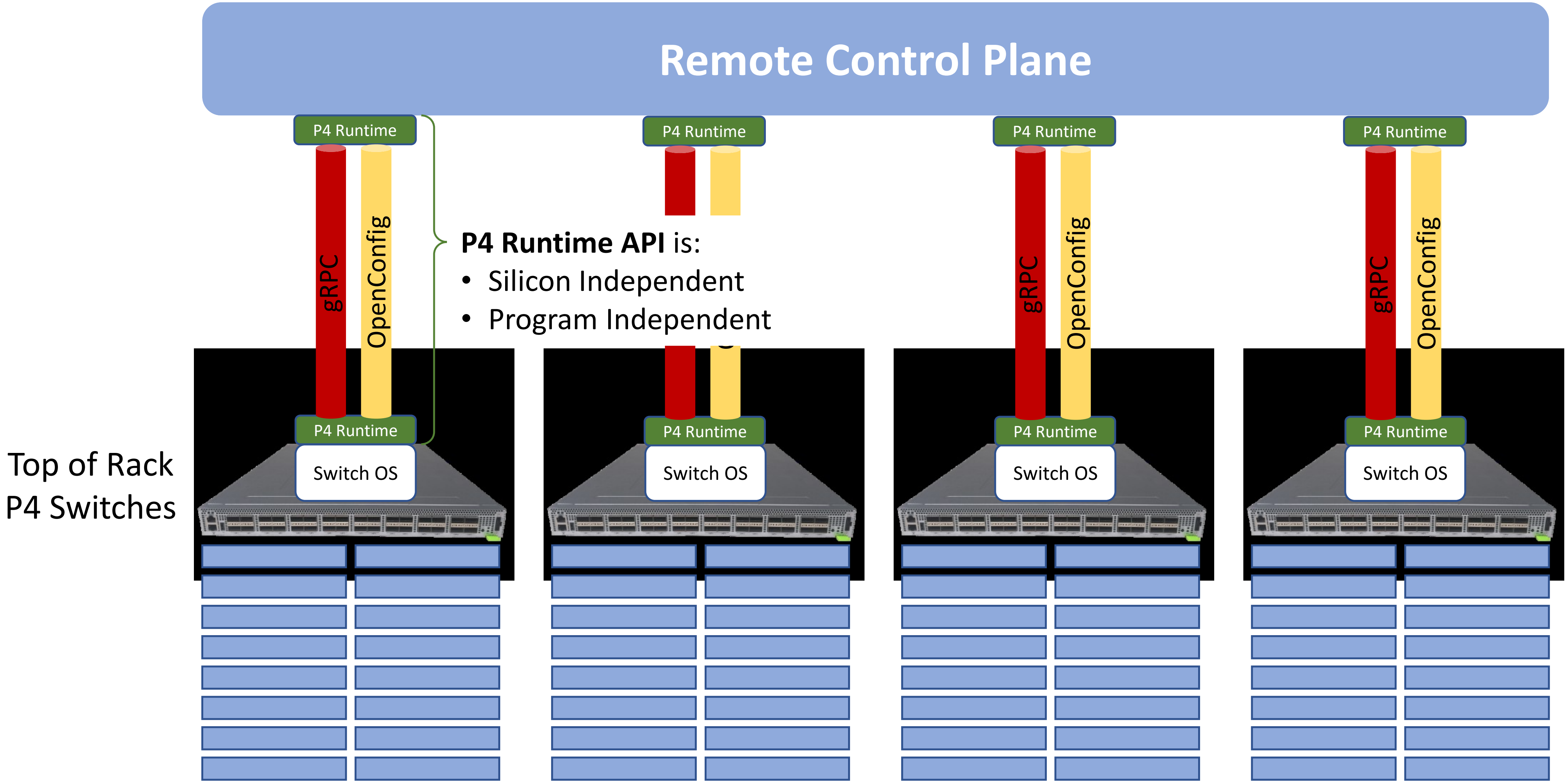
[1] “SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs.” Rui Miao et al. Sigcomm 2017.
[2] “NetCache: Balancing Key-Value Stores with Fast In-Network Caching”, Xin Jin et al. To appear at SOSP 2017

Eötvös Loránd University

How programmability is being used

❸ Silicon independence

P4 Runtime Open-source project to remotely control P4 switches[1]



P4 Runtime API is:
- Silicon Independent
- Program Independent

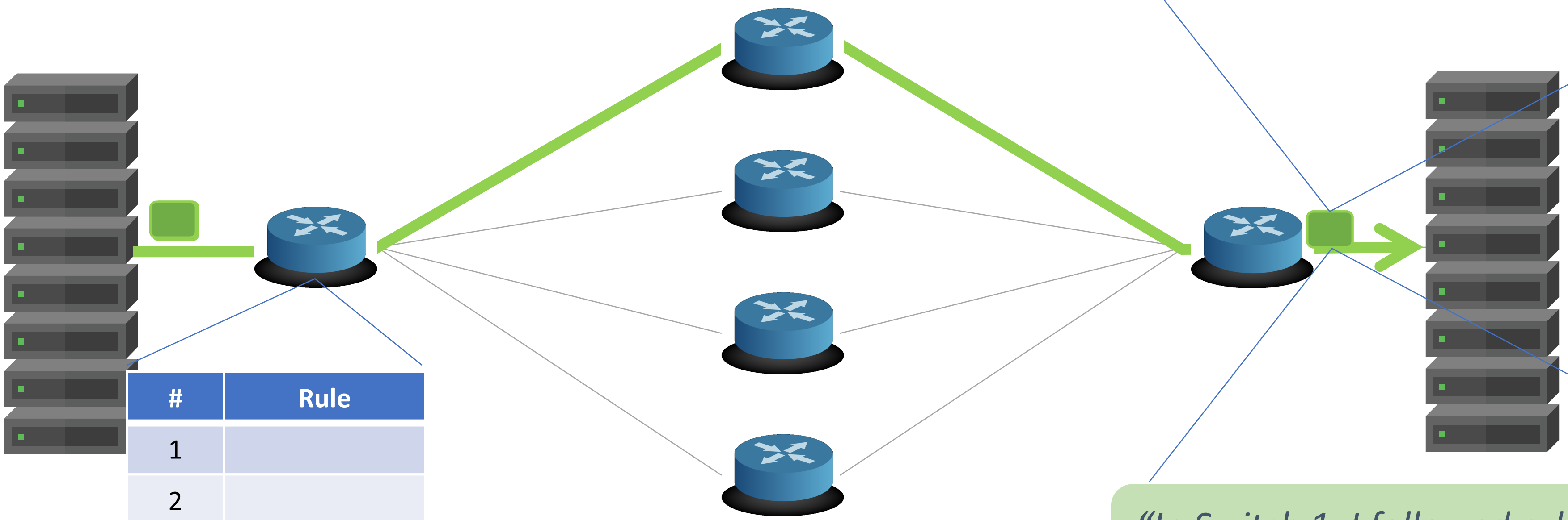
[1] "P4 Program-dependent Controller Interface for SDN Applications", Samar Abdi et al (Google), P4 Workshop 2017

How programmability is being used

❹ Network telemetry

❶ *“Which path did my packet take?”*

“I visited Switch 1 @780ns,
Switch 9 @1.3µs, Switch 12 @2.4µs”

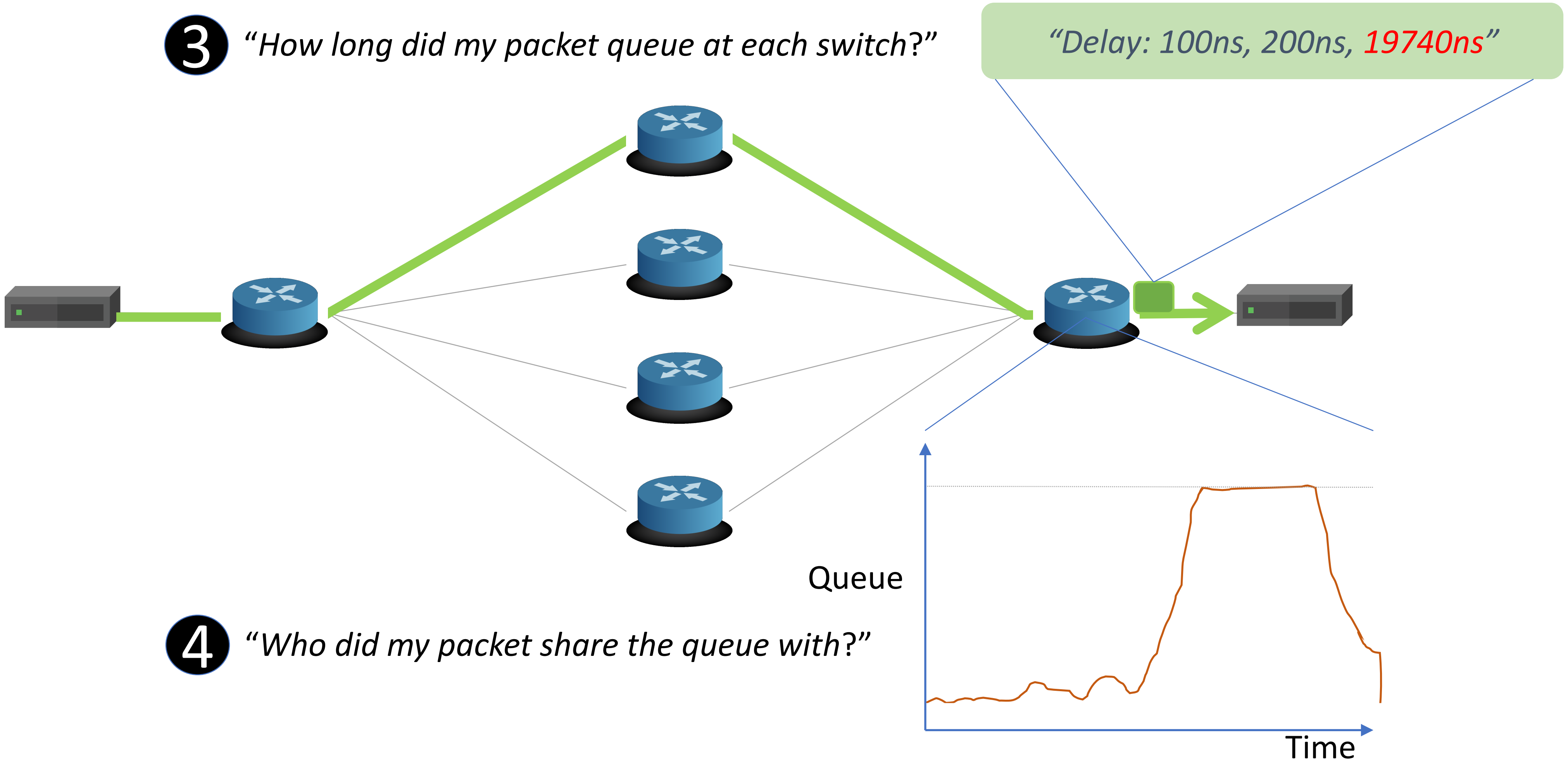


#	Rule
1	
2	
3	
…	
75	192.168.0/24
…	

“In Switch 1, I followed rules 75 and 250.
In Switch 9, I followed rules 3 and 80. ”

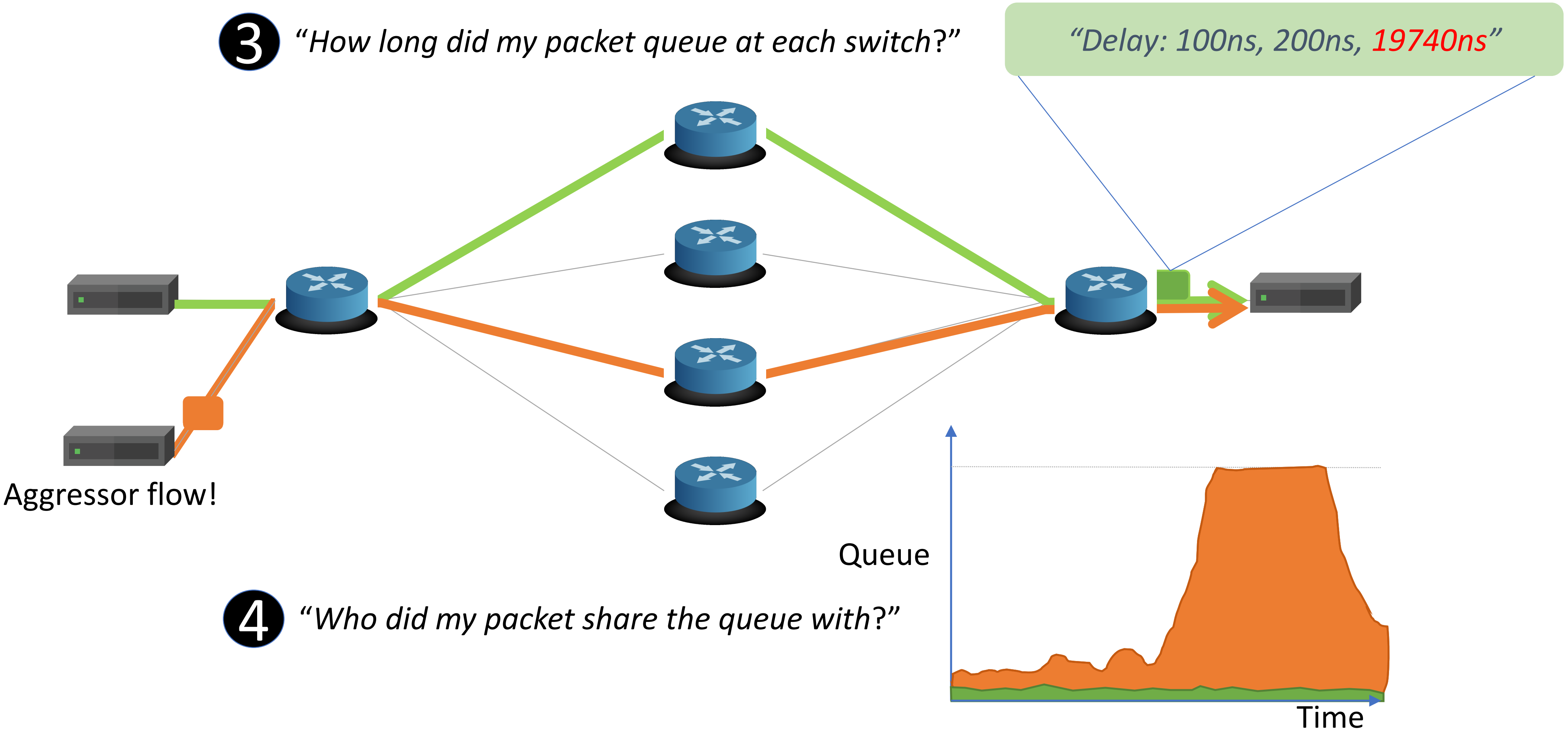
❷ *“Which rules did my packet follow?”*

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



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

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



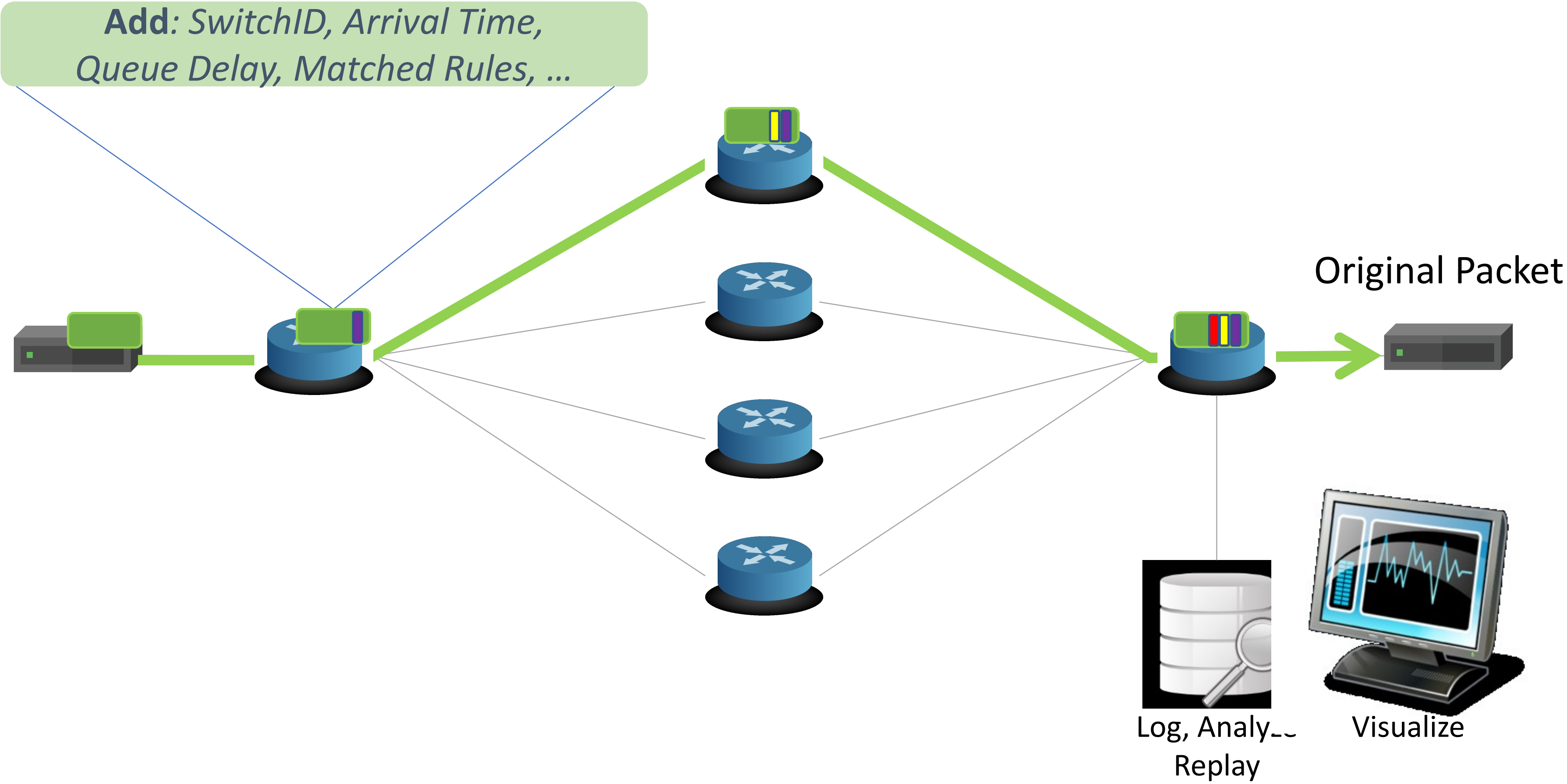
❹ *“Who did my packet share the queue with?”*

We'd like the network to 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?"*

A programmable device can potentially answer all four questions at line rate.

INT: Inband Network Telemetry



```
/* INT: add switch id */
action int_set_header_0() {
  add_header(int_switch_id_header);
  modify_field(int_switch_id_header.switch_id,
               global_config_metadata.switch_id);
}

/* INT: add ingress timestamp */
action int_set_header_1() {
  add_header(int_ingress_tstamp_header);
  modify_field(int_ingress_tstamp_header.ingress_tstamp, i2e_metadata.ingress_tstamp);
}

/* INT: add egress timestamp */
action int_set_header_2() {
  add_header(int_egress_tstamp_header);
  modify_field(int_egress_tstamp_header.egress_tstamp,
               eg_intr_md_from_parser_aux.egress_global_tstamp);
}
```

P4 code snippet: Insert switch ID, ingress and egress timestamp