Software Defined Networking

MOBILE AND CYBER-PHYSICAL SYSTEMS

FEDERICA PAGANELLI

federica.paganelli@unipi.it

Learning Objectives

- Requirements and current network technology limitations

- Software Defined Networking (SDN)
  - Main concepts and definitions
  - SDN Data Plane
  - SDN Control Plane
  - Some Applications of SDN

Requirements and current network technology limitations

Evolving Network Requirements

Why redesigning traditional approaches to network architecture?

from distributed to centralized logical approach

Demand is increasing

- Cloud computing
- Big Data
- Mobile traffic
- Internet of Things (IoT)

… just to mention main trends demanding for network architecture redesign => improved technologies

=> not just matter of # traffic

Evolving Network Requirements

But… also supply is increasing

the capacity of network technologies is capable of absorbing rising loads

performance of network devices has increased

- Faster processor speed, greater buffer capacity and buffer access speed (thanks to larger, faster memories)

So what's the problem?

It is not just a matter of supply and demand

1. **traffic patterns have changed and become more dynamic and complex**

Traffic patterns measure how traffic varies in time and space

Complex and dynamic traffic patterns 1/3

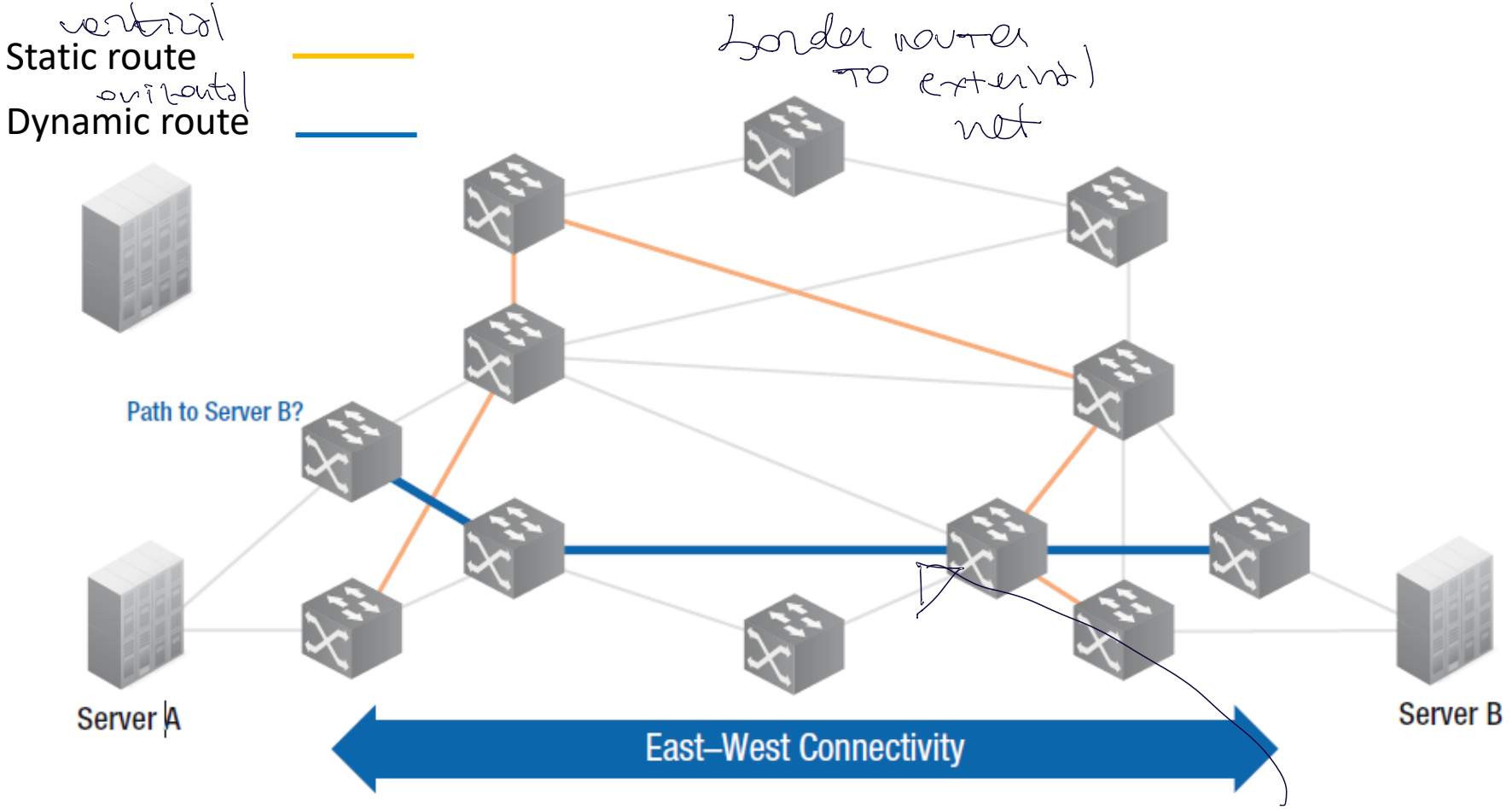
- **Modern distributed applications** typically access multiple databases and servers that must communicate with each other
  - "horizontal" traffic between servers in addition to the "vertical" client/server one
- **Unified communications** (UC) services are increasingly used within enterprises.
  - integration of communication services such as instant messaging (chat), presence information, voice, mobility, audio, web and video traffic into a unique service delivery platform
  - applications have thus to access multiple servers

Complex and dynamic traffic patterns 2/3

4. **Widespread use of clouds**
   - a significant amount of traffic that was previously local to enterprise networks has moved to remote clouds
   - creating increased and often very **unpredictable loads on enterprise routers**.

5. **Server virtualization**. It allows partitioning a single hardware machine into multiple virtual ones (e.g. Virtual Machines).
   - considerable amount of **traffic among virtual servers** (e.g., migration, switchover).
   - such **horizontal traffic** flow **changes in location and intensity over time**, demanding a flexible approach to managing network resources.

Traffic between servers



From: OPEN DATA CENTER ALLIANCE Master USAGE MODEL: Software-Defined Networking Rev. 1.0

By some estimates, **over 70 percent of traffic is now East-West** (traffic between servers within the data center).

The traditional three-layer data center network design (core-aggregation-access) is ill-equipped to handle this kind of traffic

Complex and dynamic traffic patterns 3/3

- The increasing use of **mobile devices** anywhere and any time
  - → fast-changing and unpredictable large traffic loads on the network
  - → the network attachment point can rapidly change

Different QoS requirements demanded by a variety of applications

- → differentiated treatment of granular traffic flows

Limitations of current networks

All these emerging trends require a **flexible management of network resources**

… but traditional networks show some limitations…

1. **Static, complex architecture:** It consists in a number of independently defined protocols each of which addresses a portion of networking requirements -> **complexity that is difficult to master**!

2. **Inconsistent policies**: manual procedures needed to change configuration in multiple switches, routers, firewalls -> error prone procedures

3. **Vendor dependence:** enterprises and carriers need to deploy new capabilities and services rapidly in response to changing business needs and user demands. A lack of open interfaces for network functions leaves the enterprises limited by the relatively slow product cycles of vendor equipment

from: Open Networking Foundation (ONF)

Key to Internet Success: Layers

Applications

…built on…

Reliable (or unreliable) transport

…built on…

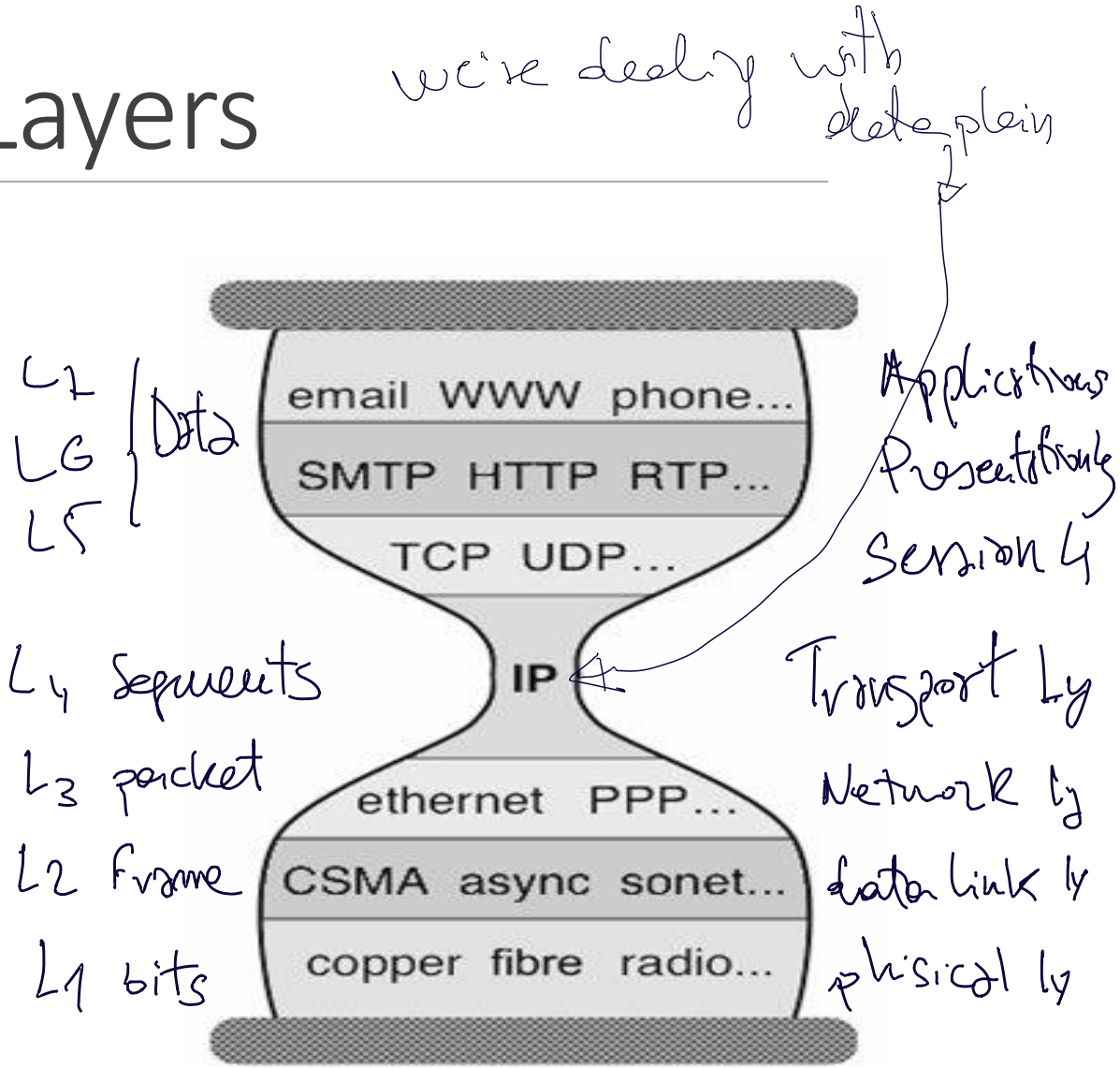
Best-effort global packet delivery

…built on…

Best-effort local packet delivery

…built on…

Physical transfer of bits



Why Is Layering So Important?

Decomposed delivery into fundamental components

Independent but compatible **innovation** at each layer

A practical success of unprecedented proportions…

Network Control/Management Plane

no general principles or abstractions guiding the design of network *control/management* plane

- in contrast, e.g., to distributed systems or database systems

Control plane is basically composed of various distributed "control" protocols

- **For each new feature/functionality needed, a new protocol is designed**

Network boxes are "closed" equipment

- (mostly proprietary) software bundled with hardware
- vendor-specific interfaces

- Big bag of protocols
- Notoriously difficult to manage
- Slow protocol standardization
- Slow innovation!

Software Defined Networking

MAIN CONCEPTS AND DEFINITIONS

Network-layer functions

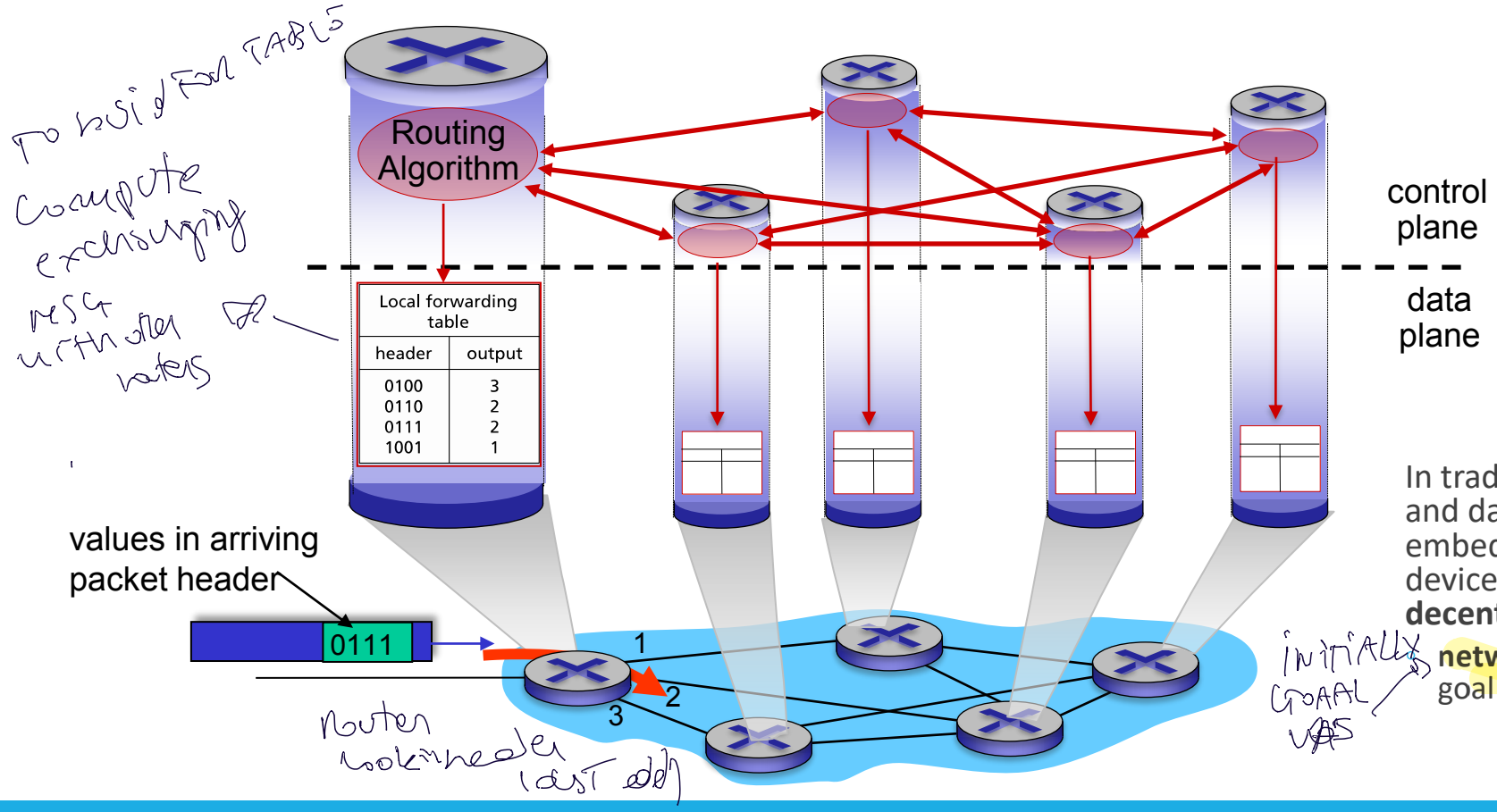
- **forwarding:** move packets from router's input to appropriate router output — *data plane* — *Fast timescales (per-packet)*
- **routing:** determine route taken by packets from source to destination — *control plane* — *Slow time-scales (per control event)*

Two approaches to structuring network control plane:

1) per-router control (traditional)
2) logically centralized control (software defined networking)

Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane to compute forwarding tables

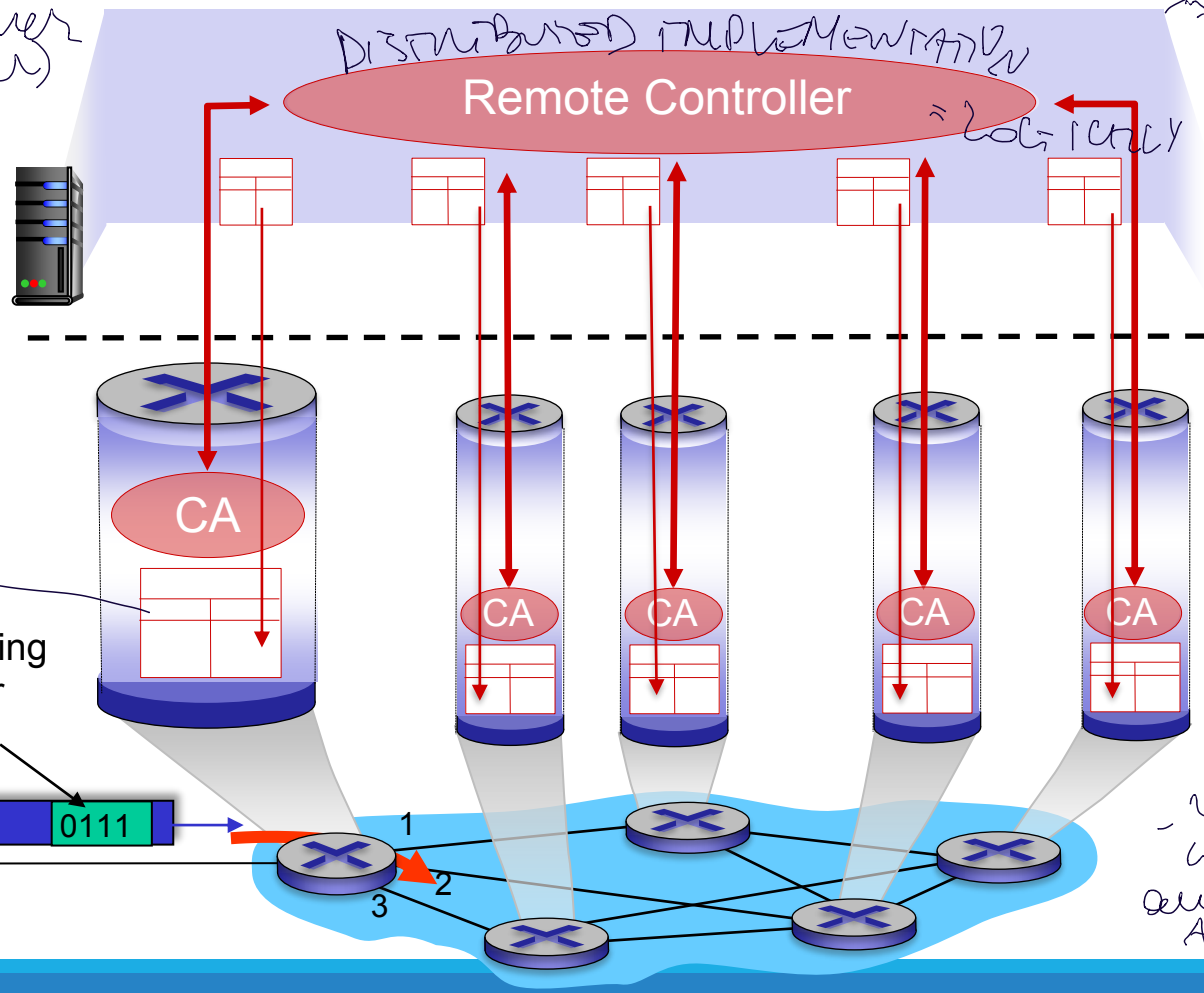


In traditional IP networks, the control and data planes are **tightly coupled**, embedded in the same networking devices, and the whole structure is highly **decentralized**.

network resilience was a crucial design goal in the early days of Internet.

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Software defined networking (SDN)

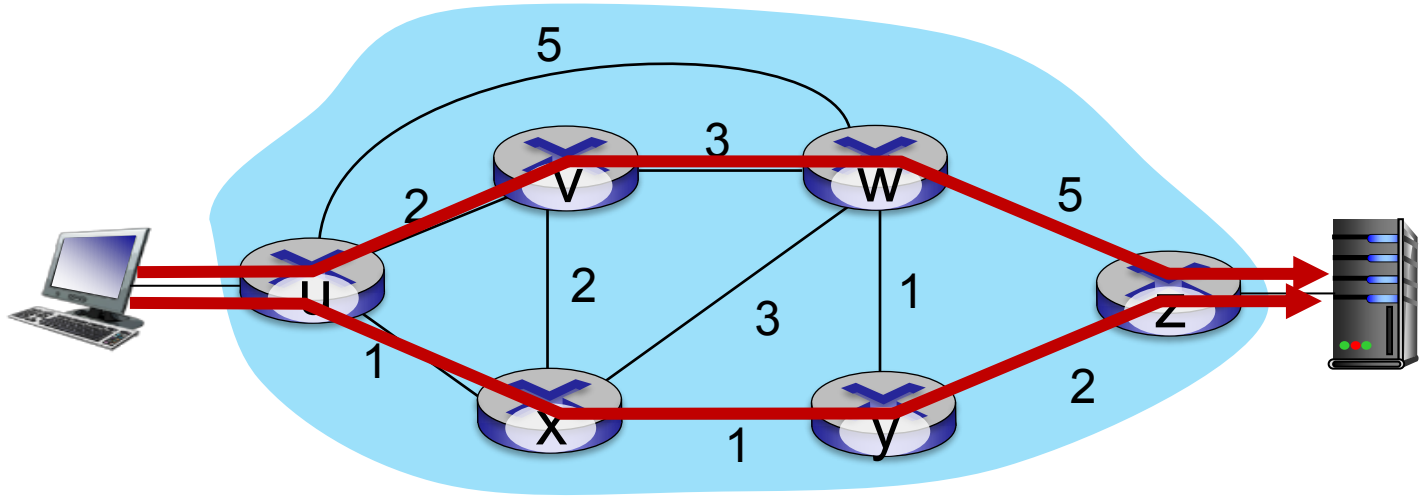
Why a *logically centralized* control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming" more difficult: compute tables as result of distributed algorithm (protocol) implemented in each-and-every router

open (non-proprietary) implementation of control plane

- foster innovation: let 1000 flowers bloom

Traffic engineering: difficult with traditional routing

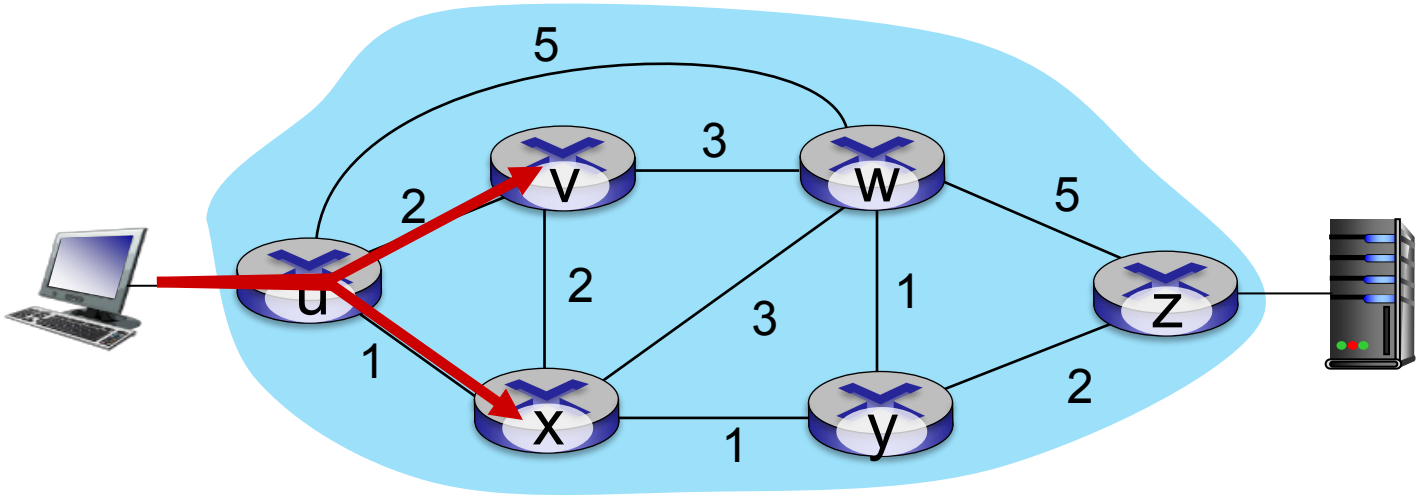


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

A: need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

link weights are only control "knobs": not much control!

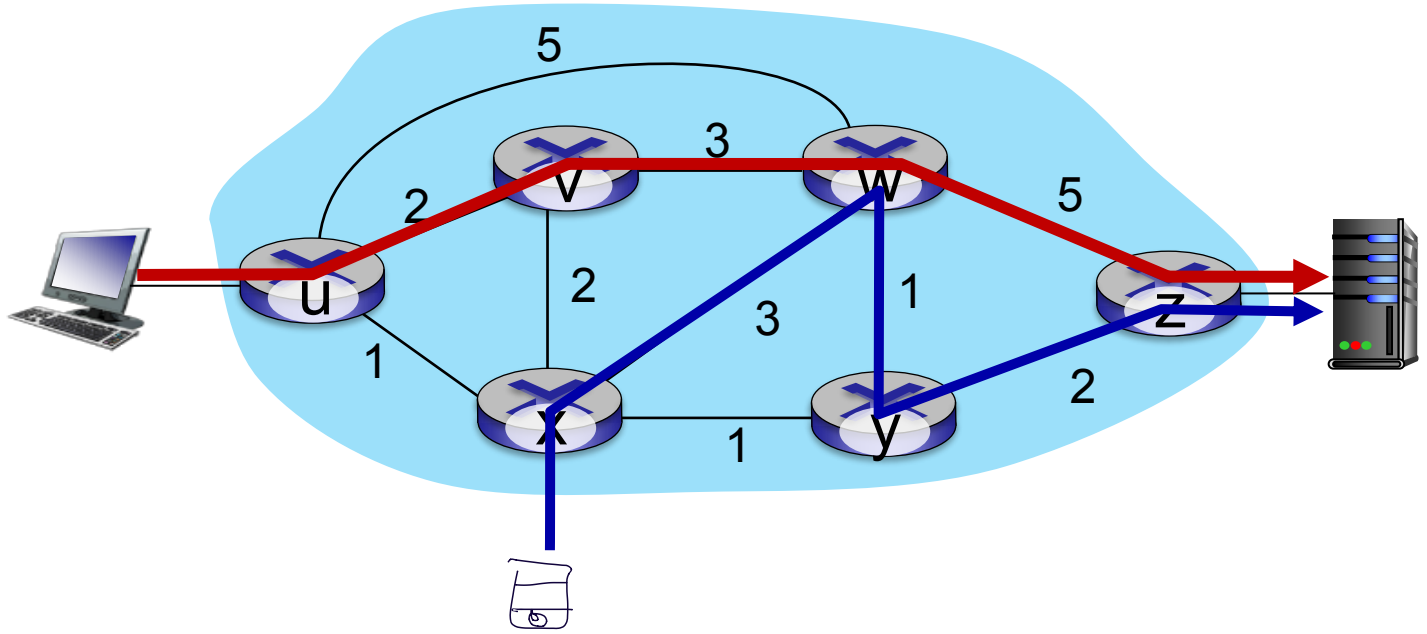
Traffic engineering: difficult with traditional routing



Q: what if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

A: can't do it (or need a new routing algorithm)

Traffic engineering: difficult with traditional routing



Q: what if w wants to route blue and red traffic differently from w to z?

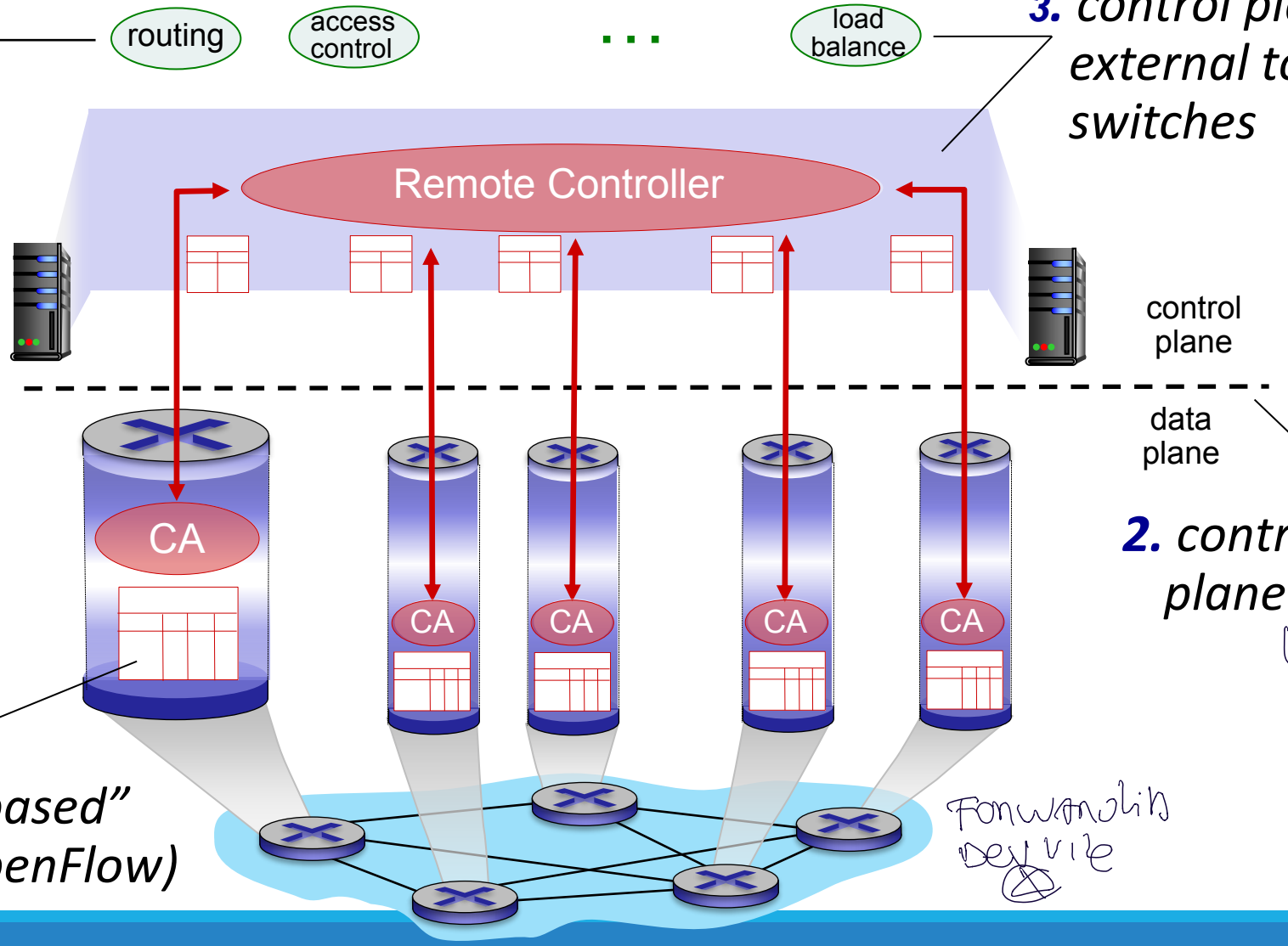
A: can't do it (with destination-based forwarding, and LS, DV routing)

We will see how that generalized forwarding and SDN can be used to achieve *any* routing desired

CA: control agent in routers

Software defined networking (SDN)

4. *programmable control applications*



3. *control plane functions external to data-plane switches*

2. *control, data plane separation*

(real separation)

1: *generalized "flow-based" forwarding (e.g., OpenFlow)*

(D) implements

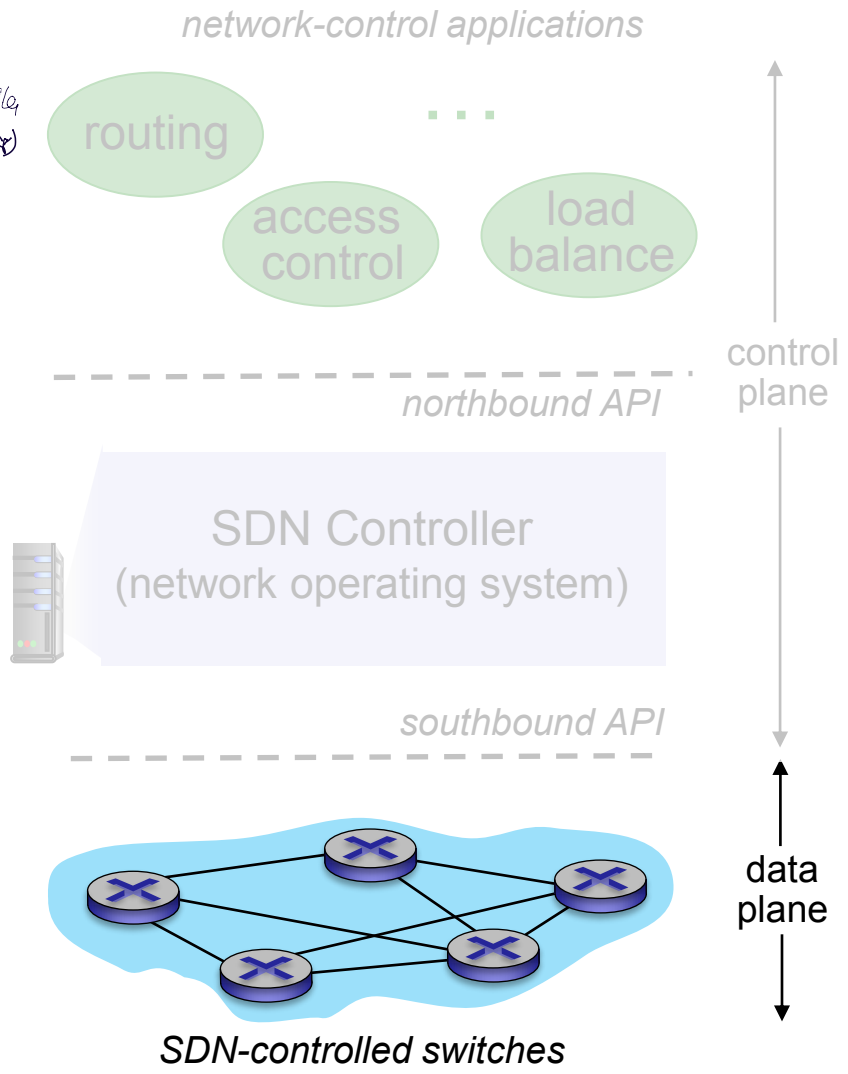
Not SDN

Forwarding device

SDN Architecture

Data-plane switches:

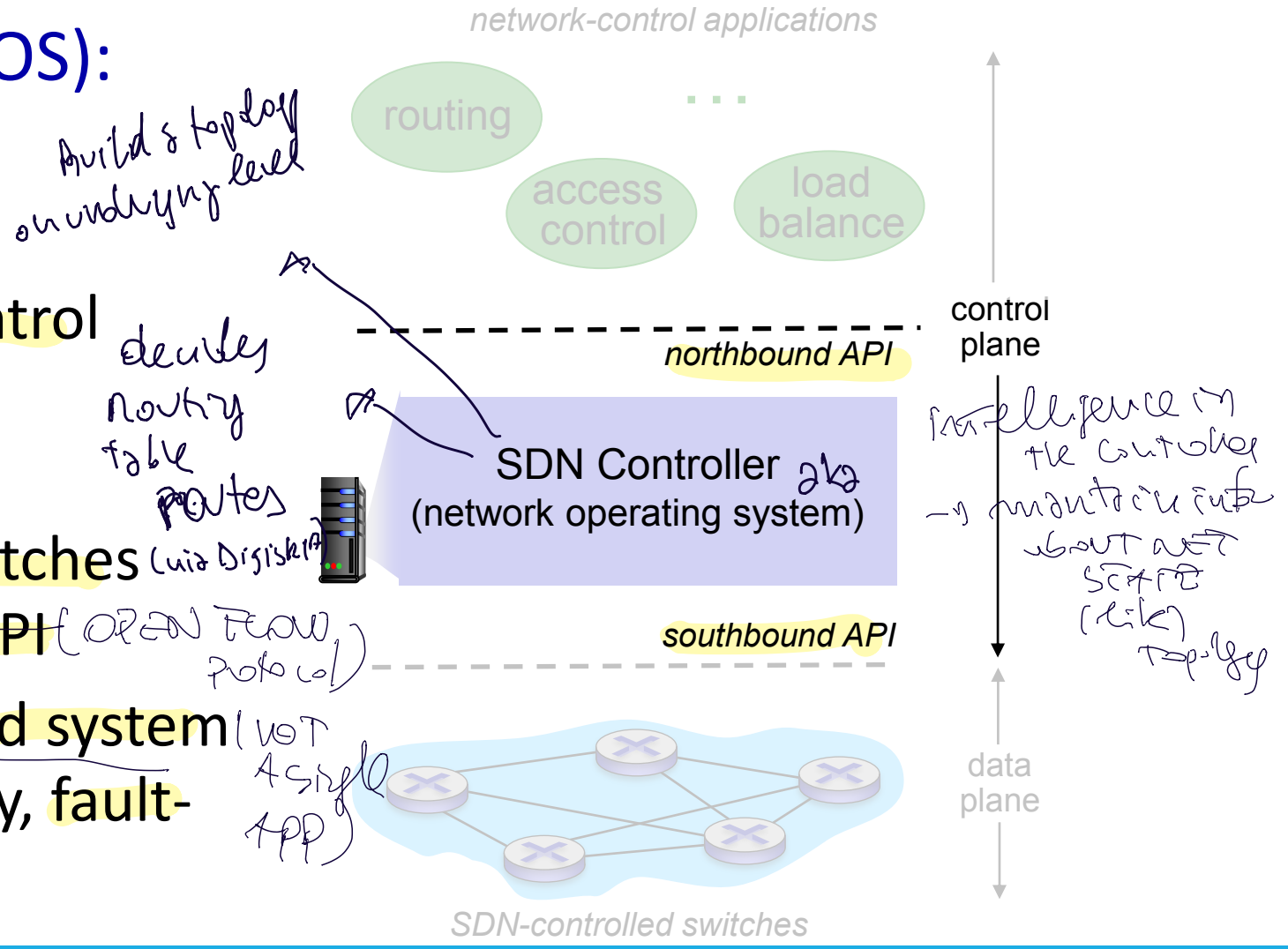
- fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating with controller (e.g., OpenFlow)



Software defined networking (SDN)

SDN controller (network OS):

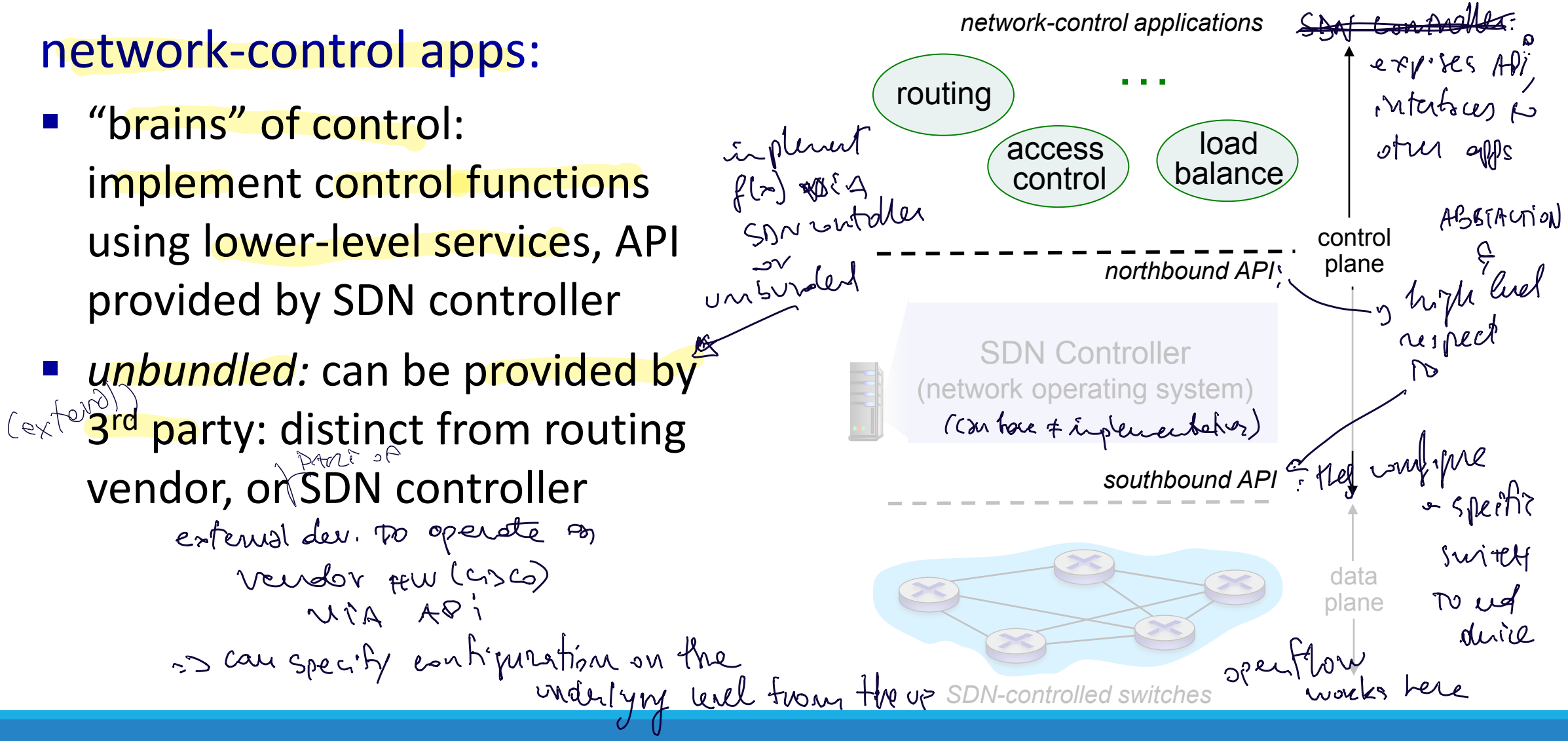
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



Software defined networking (SDN)

network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor, or SDN controller



Brief history

~ 2004 Research on new management paradigms

- Princeton, Stanford/Berkeley

2008 Software defined networking

- NOX Network Operating System [NICIRA]
- OpenFlow switch interface [Stanford/NICIRA]

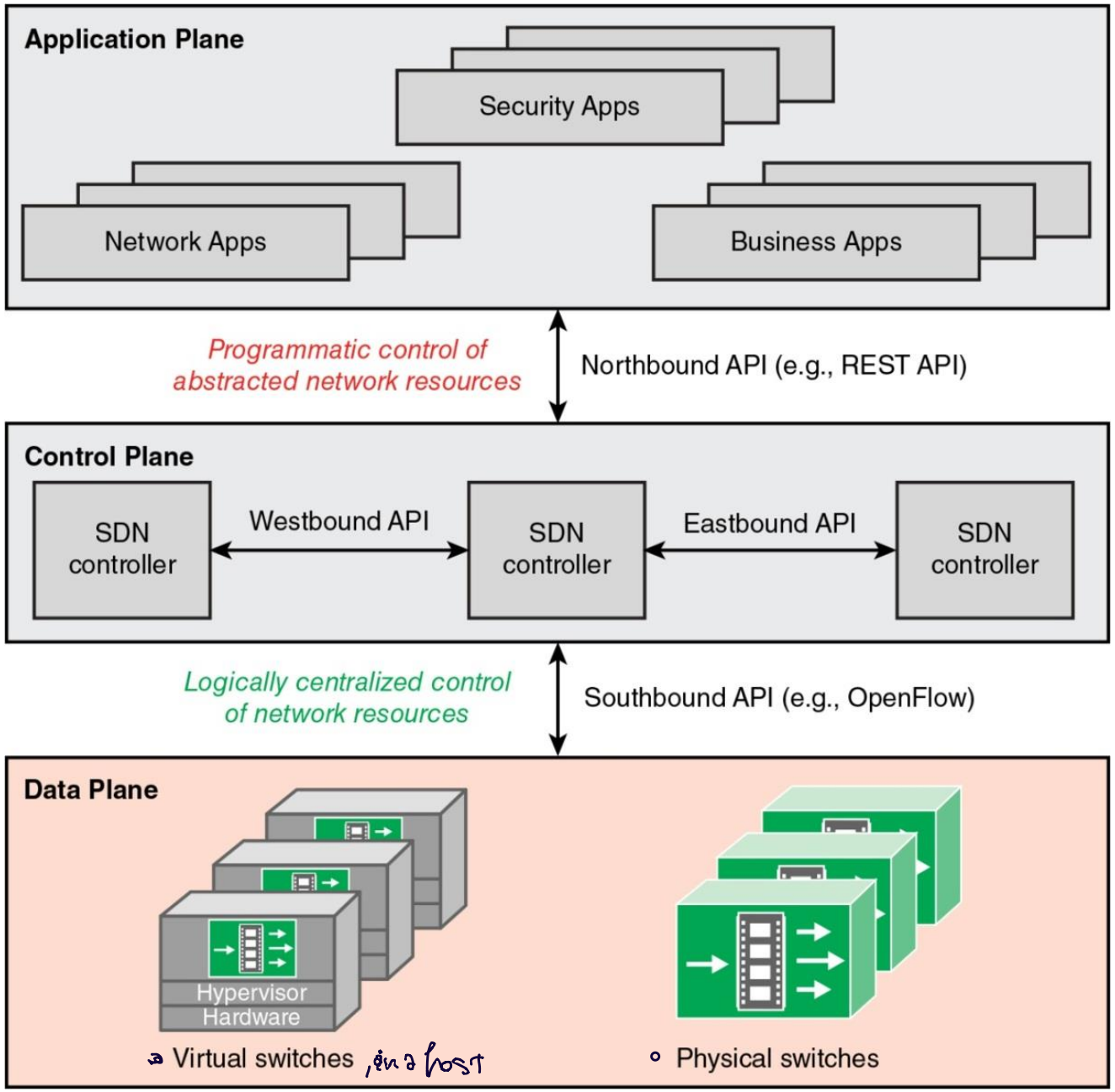
2011 Open Networking Foundation

- Board: Google, Yahoo, Verizon, ecc.
- Members: Cisco, HP, Dell,..

2013

- 1600 attendees at the Open Networking Summit
- SDN used in Google WAN

SDN Data Plane



Data Plane

Resource or infrastructure layer made by forwarding devices

- perform the transport and processing of data according to **decisions made by the SDN control plane**.
- without embedded software to make autonomous decisions.

-> **Forwarding Abstraction**

Data Plane Network Device

Data forwarding function:

- Accepts incoming data flows from other network devices and end systems
- forwards them along the data forwarding paths computed and established according to the rules defined by the SDN applications

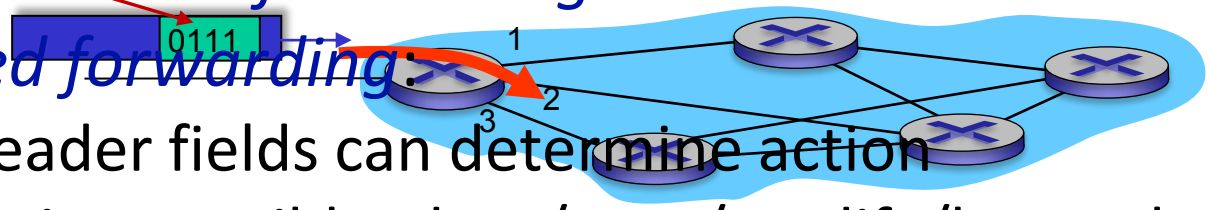
Control support function

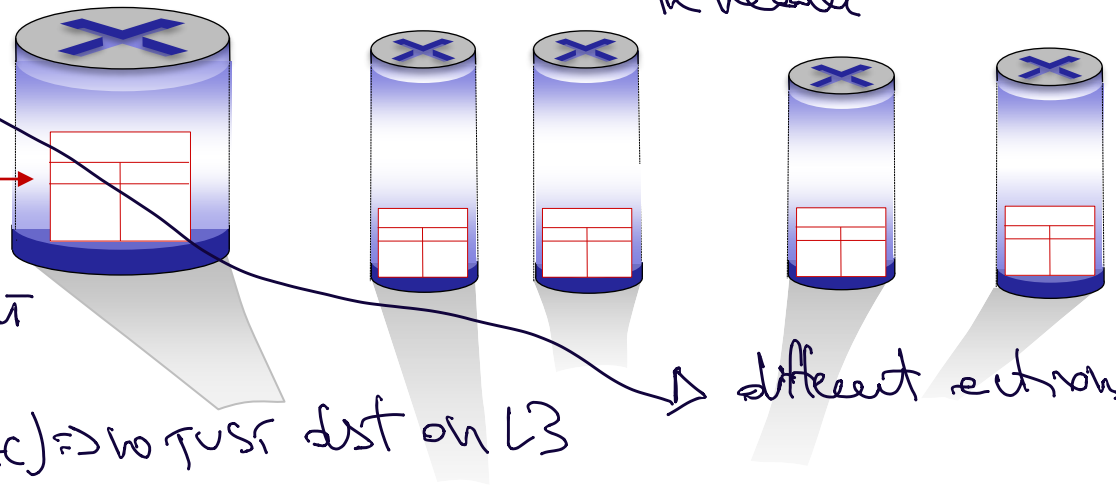
- Interaction with the SDN controller for management of forwarding rules
- Reference standard specifications: **OpenFlow switch protocol**.

Generalized forwarding: match plus action

Review: each router contains a forwarding table (aka: flow table)

- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet



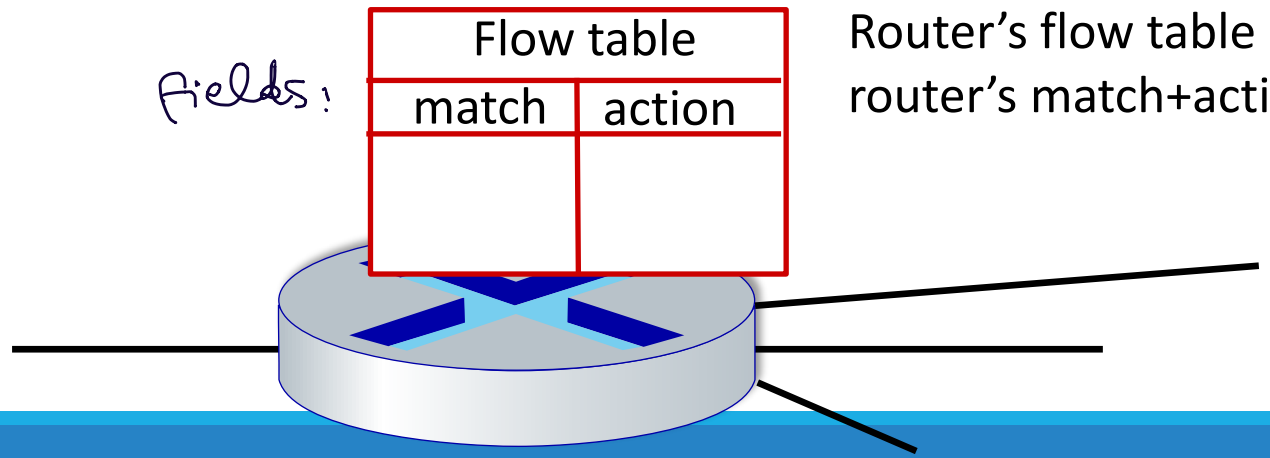


Flow table abstraction

flow: defined by header field values (in link-, network-, transport-layer fields)

generalized forwarding: simple packet-handling rules

- **match:** patterns values in packet header fields
- **actions:** for matched packet: drop, forward, modify, matched packet or send matched packet to controller
- **priority:** disambiguate overlapping patterns
- **counters:** #bytes and #packets



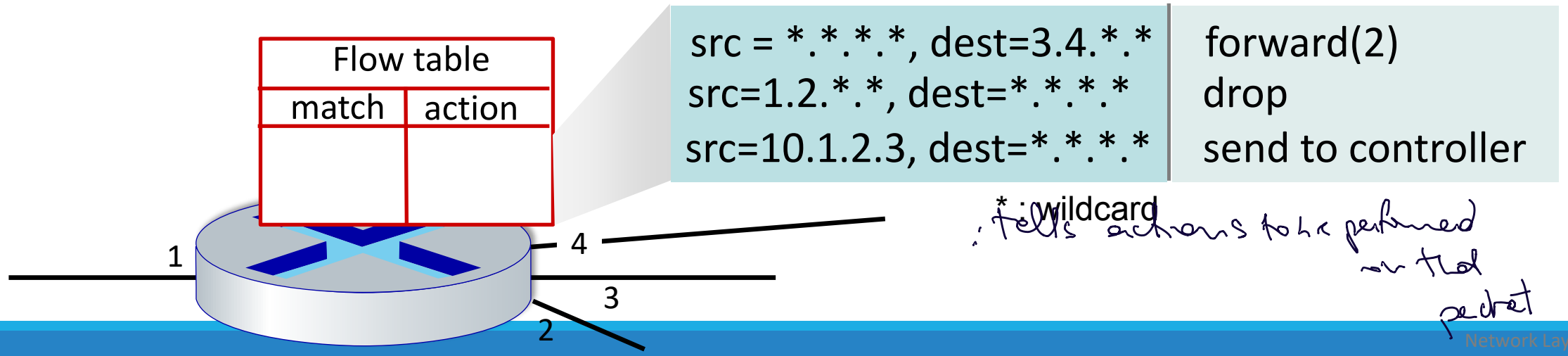
Router's flow table define router's match+action rules

Flow table abstraction

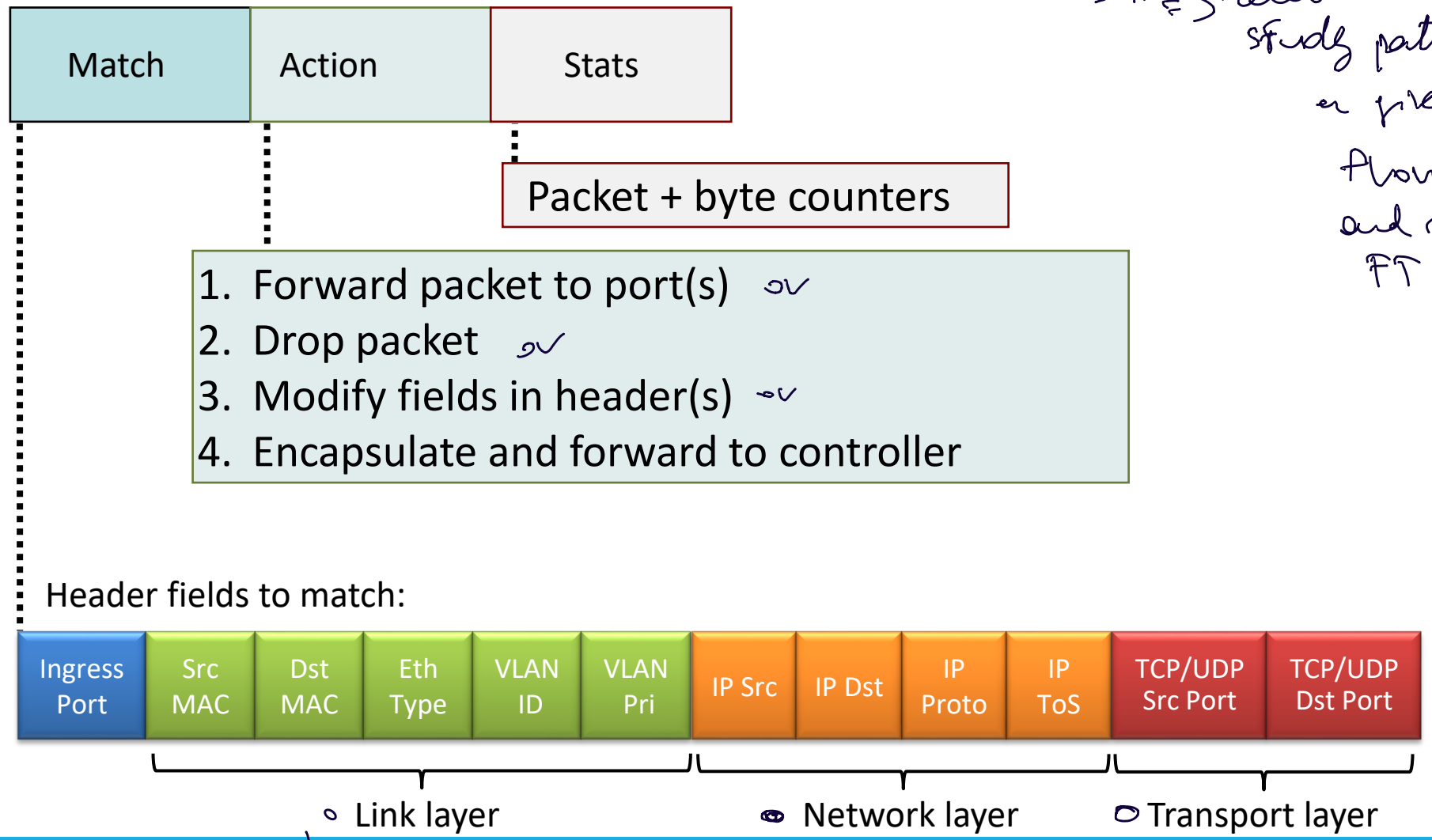
flow: defined by header fields

generalized forwarding: simple packet-handling rules

- **match:** pattern values in packet header fields
- **actions:** for matched packet: drop, forward, modify, matched packet or send matched packet to controller
- **priority:** disambiguate overlapping patterns
- **counters:** #bytes and #packets



OpenFlow: flow table entries



~~match~~ way to generalized forwarding Fields in the header

OpenFlow: examples

Destination-based forwarding: switch specific dest address to forward at port: X

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1 (suspicious host)

OpenFlow: examples

behaviour of a switch

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OpenFlow abstraction behaviors implemented:

match+action: abstraction unifies different kinds of devices

Router

- *match:* longest destination IP prefix
- *action:* forward out a link

Switch

- *match:* destination MAC address
- *action:* forward or flood: more ports

To port #

Firewall

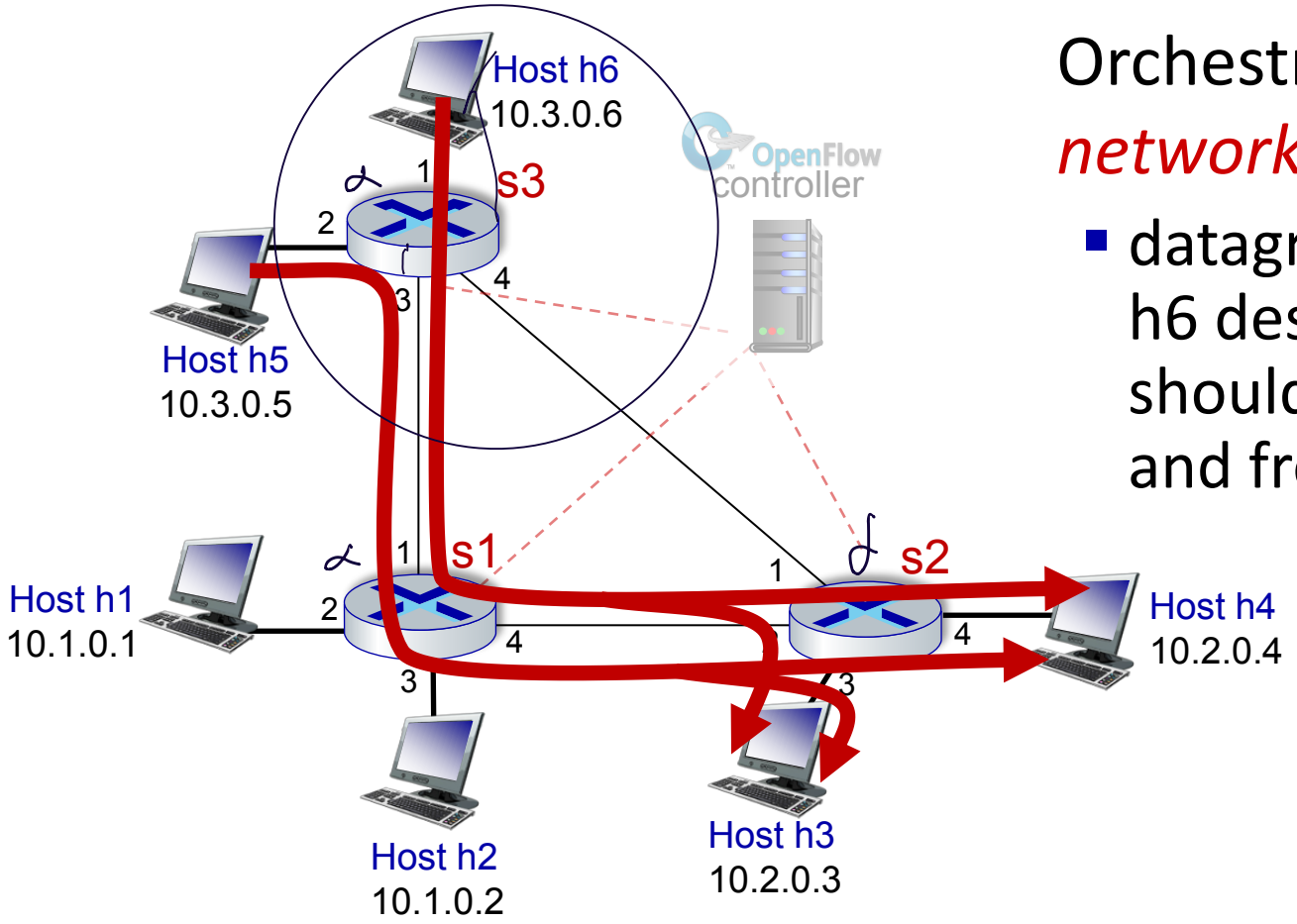
- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny (allow/drop)

NAT

- *match:* IP address and port
- *action:* rewrite address and port

device can modify the header:
1st match a given ip port (match)
2nd rewrite (source) ip address with the public one

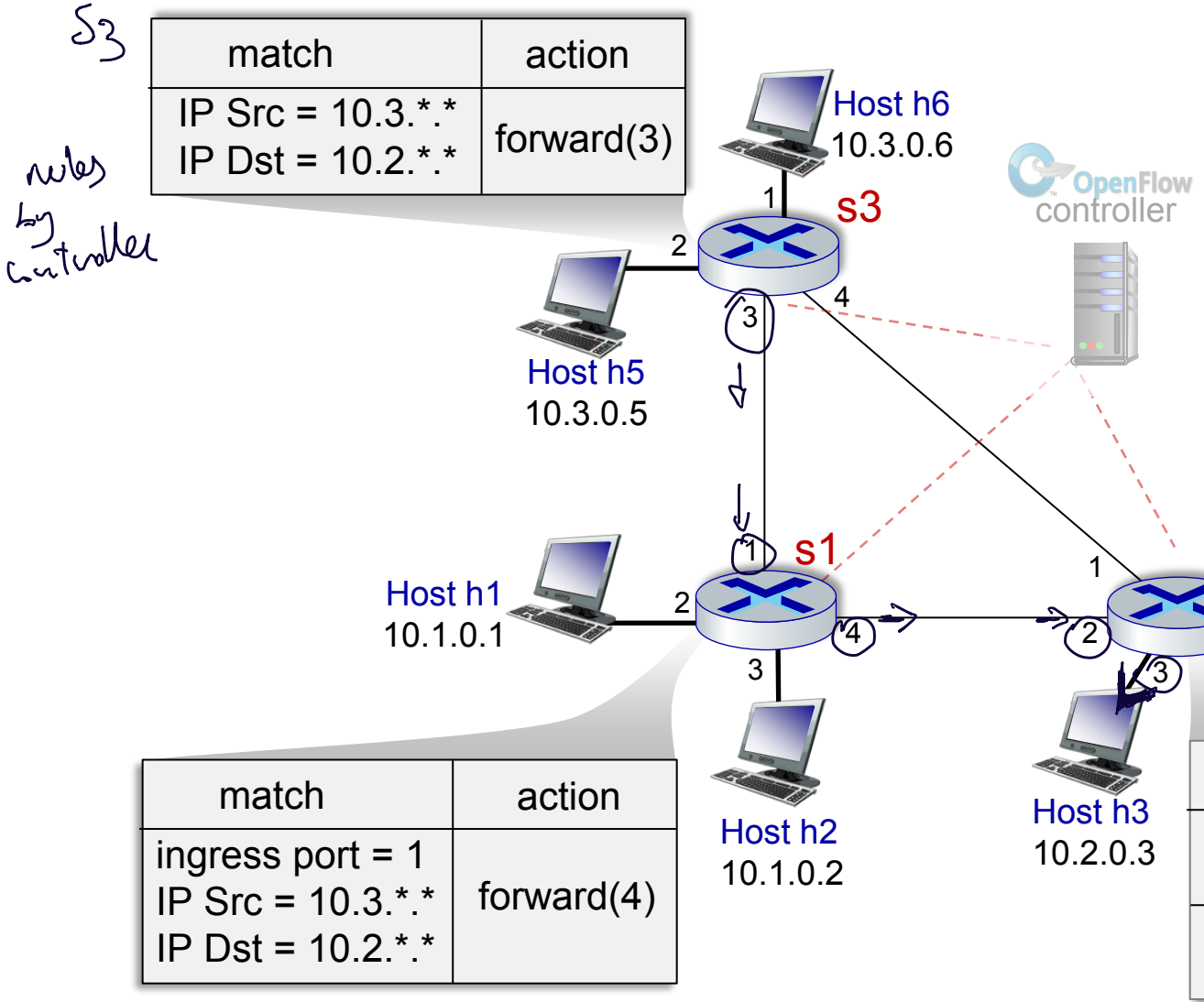
OpenFlow example



Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 destined to h3 or h4, should be forwarded via s1 and from there to s2

OpenFlow example



Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 destined to h3 or h4, should be forwarded via s1 and from there to s2

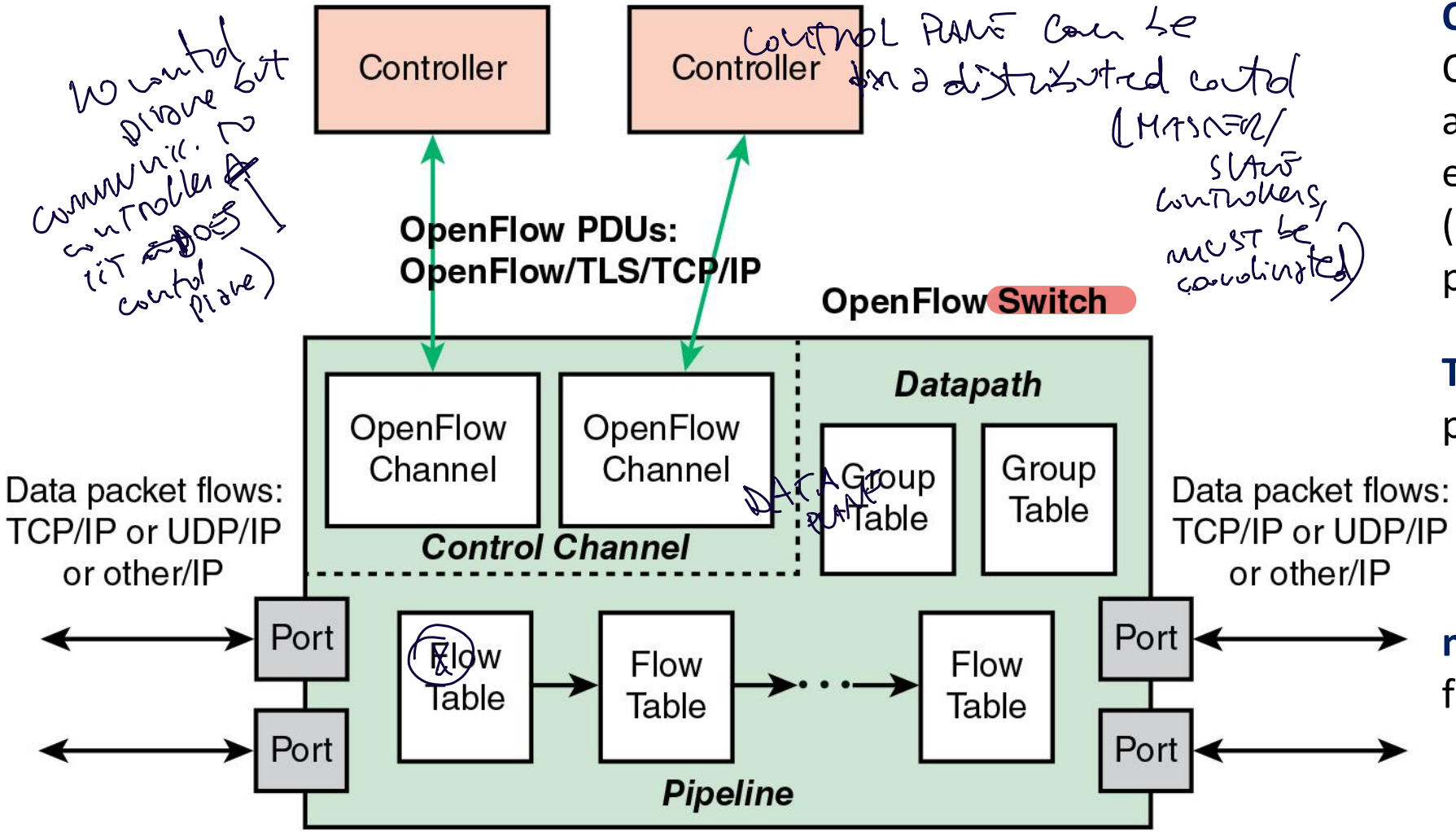
controller: orchestrate rules to implement behavior on the complete topology of the network

Generalized forwarding: summary

- "match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - "program" *network-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming:

    P4 (see p4.org)

OpenFlow Switch



OpenFlow channel: through OpenFlow, the controller can add, update, and delete flow entries in tables, both reactively (in response to packets) and proactively.

Tables for packet flow processing, i.e. forwarding

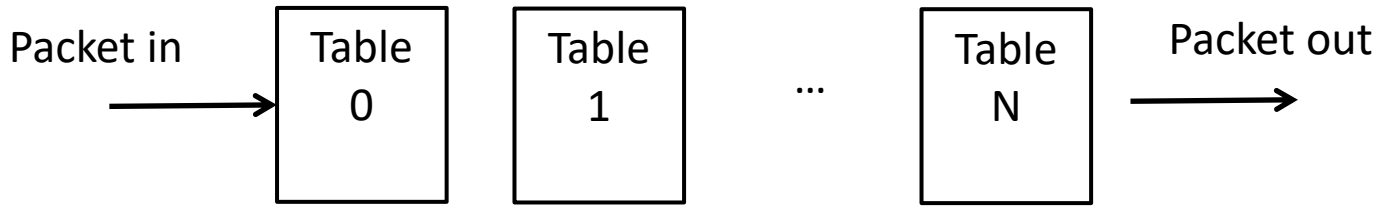
multiple I/O ports for packet flows into and out of the device.

Flow Table Pipeline

- PACKET UNIT goes matched firstly by TABLE 0:
  - according to IP can jump by the rules specified

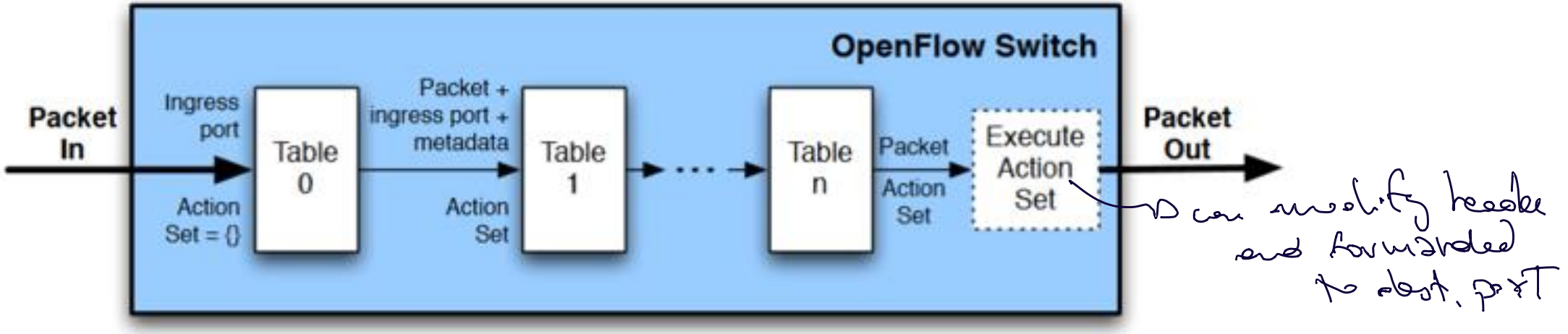
- A switch includes one or more flow tables
- If there is more than one flow table, they are organized as a pipeline
  - The flow tables of an OpenFlow switch are sequentially numbered, starting at 0.
  - The packet is first matched against flow entries of flow table 0. Other flow tables may be used depending on the outcome of the match in the first table.

The use of multiple tables in a pipeline, rather than a single flow table, provides the SDN controller with considerable flexibility



Processing pipeline

- When processed by a flow table, the packet is matched against the flow entries of the flow table.
- If there is a matching, any instructions associated with this entry is executed. This may include updating the action set, updating the metadata value, and performing actions.
- those instructions may explicitly direct the packet to another flow table (Goto Instruction), where the same process is repeated again, to the group table, to the meter table, or, finally, **to an output port for forwarding**



Processing pipeline

If there is a match on one or more entries in a table the match is defined to be with the **highest-priority** matching entry

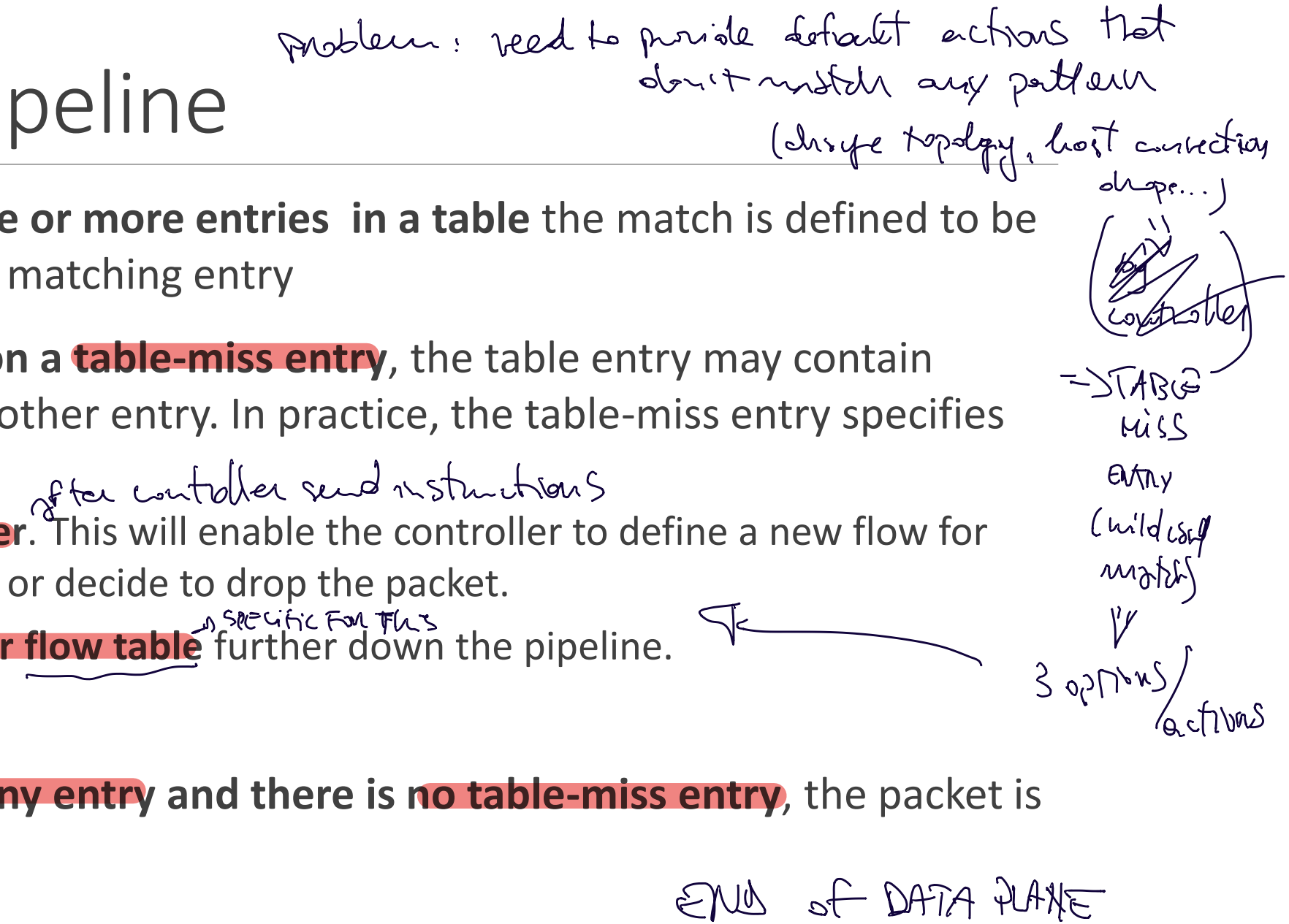
If there is a match only on a table-miss entry, the table entry may contain instructions, as with any other entry. In practice, the table-miss entry specifies one of three actions:

a) **Send packet to controller**. This will enable the controller to define a new flow for this and similar packets, or decide to drop the packet.

b) **Direct packet to another flow table** further down the pipeline.

c) **Drop the packet**.

If there is no match on any entry and there is no table-miss entry, the packet is dropped.



Questions

1. How does general forwarding differ from destination-based forwarding?

   MATCH & ACTION

   - actions not just based on IP address but any field of header frame, can abstract behaviours of: routers, switches, FIREWALL, NAT

2. What is meant by the "match-action" operation of a router or switch? In the case of destination-based forwarding, what is compared and what action is taken?

3. In the case of SDN, name three fields that can be compared and three actions that can be taken.

   - MAC source add / Dest
   - IP Sorce / Dest
   - TCP port / UDP port

   ACTIONS: copy, drop, modify, log, forward

Problem

Consider the OpenFlow SDN network in the figure.

Suppose the desired forwarding behavior for datagrams arriving at s2 is as follows:

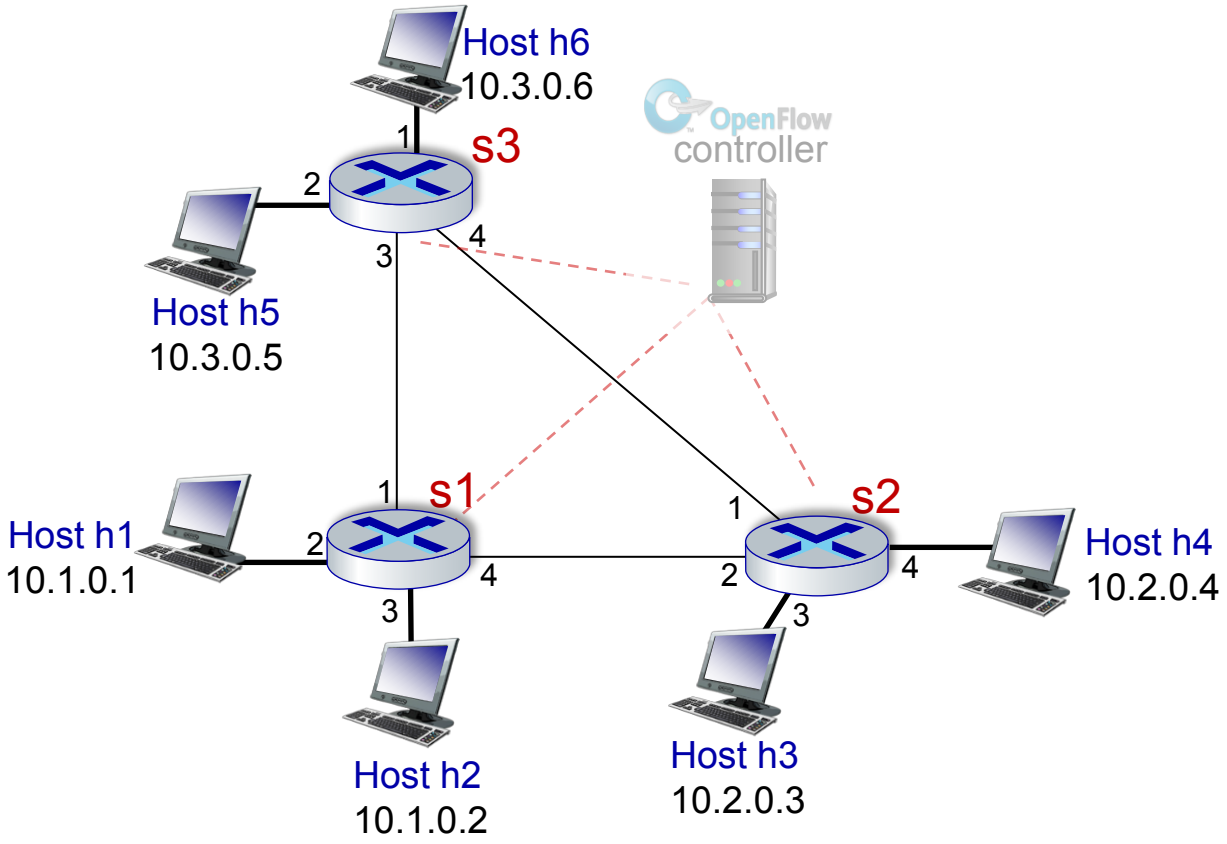
a) datagrams arriving at port 1 from hosts h5 or h6 and having hosts h1 or h2 as destination must be forwarded to output port 2;

b) datagrams arriving at port 2 from hosts h1 or h2 and having hosts h5 or h6 as destination must be forwarded to output port 1;

c) datagrams arriving at port 1 or 2 and having hosts h3 or h4 as destination must be forwarded to the specified host;

d) hosts h3 or h4 must be able to exchange datagrams.

Specify occurrences of the s2 flow table to implement this behavior



References

Prof. Scott Shenker of UC Berkeley, *"The Future of Networking, and the Past of Protocols"*

Stallings, W. (2015). Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud. Addison-Wesley Professional.

Shenker, S., Casado, M., Koponen, T., & McKeown, N. (2011). The future of networking, and the past of protocols. Open Networking Summit, 20, 1-30.

OpenFlow switch specifications, Open Networking, v 1.5.1 2015, https://www.opennetworking.org/software-defined-standards/specifications/

Sushant Jain et al. 2013. B4: experience with a globally-deployed software defined wan. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13). ACM, New York, NY, USA, 3-14

A Service-Oriented Deployment Policy of End-to-End Network Slicing Based on Complex Network Theory - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/5G-network-slicing-architecture_fig1_324175599 [accessed 13 Mar, 2019]

http://thenetworksherpa.com/data-control-plane-separation-sortof/)

Paul Goransson, Chuck Black, Software Defined Networks: A Comprehensive Approach, Morgan Kaufmann

Marco Cello, Talk@IEIIT CNR, Genova, 28 Marzo 2014

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat, B4: experience with a globally-deployed software defined wan. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13). ACM, New York, NY, USA, 3-14

From: SDN Apps (or "how I can use SDN in my daily life") Dr. Vasileios Kotronis vkotronis@ics.forth.gr