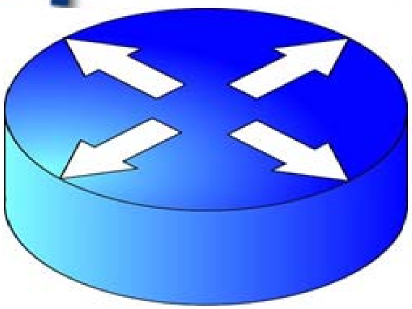
Software Defined Networking

SDN Relatives and OpenFlow

David Hausheer

Department of Electrical Engineering
and Information Technology
Technische Universität Darmstadt

E-Mail: hausheer@ps.tu-darmstadt.de
http://www.ps.tu-darmstadt.de/teaching/sdn



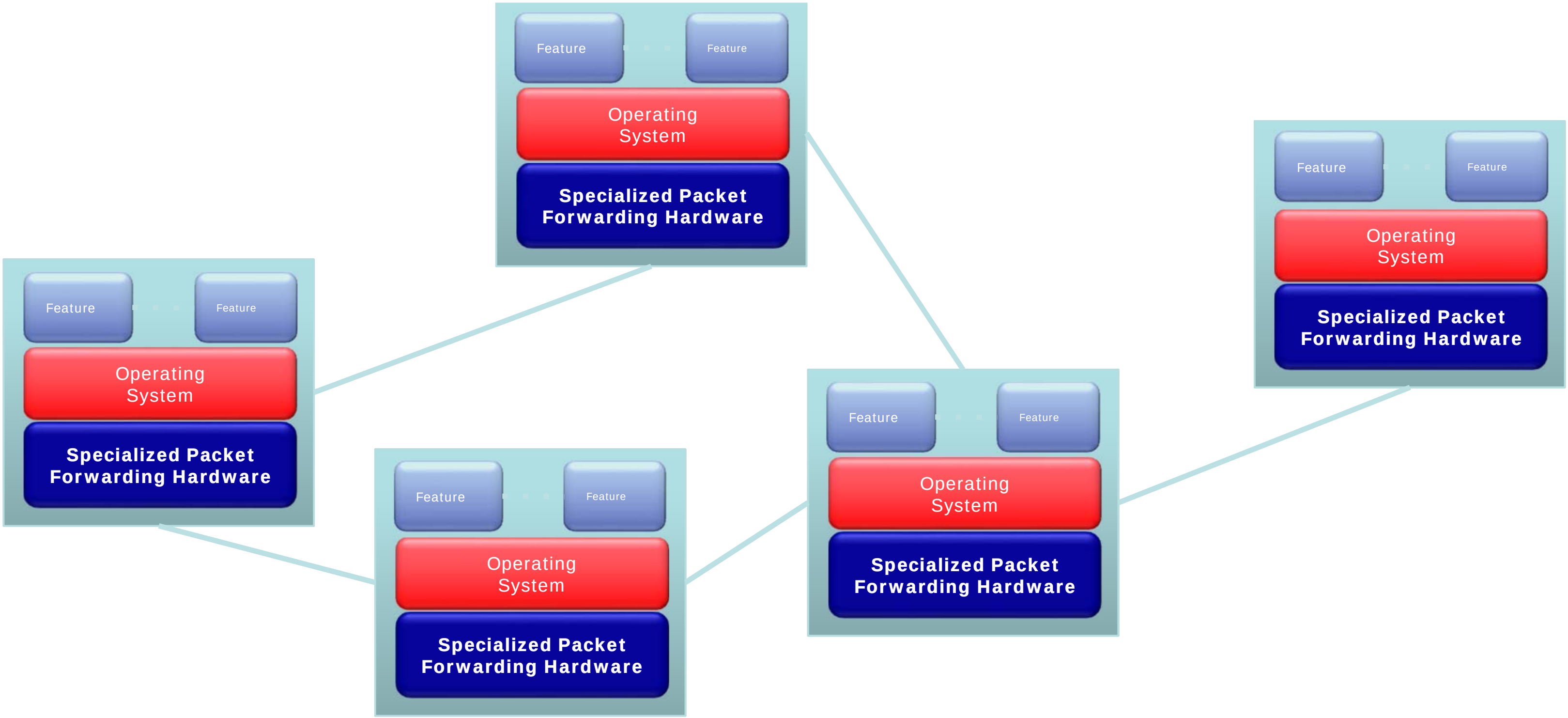
*Original slides for this lecture provided by Bernhard Plattner and Xenofontas Dimitropoulos (ETH Zürich)

Lecture Overview

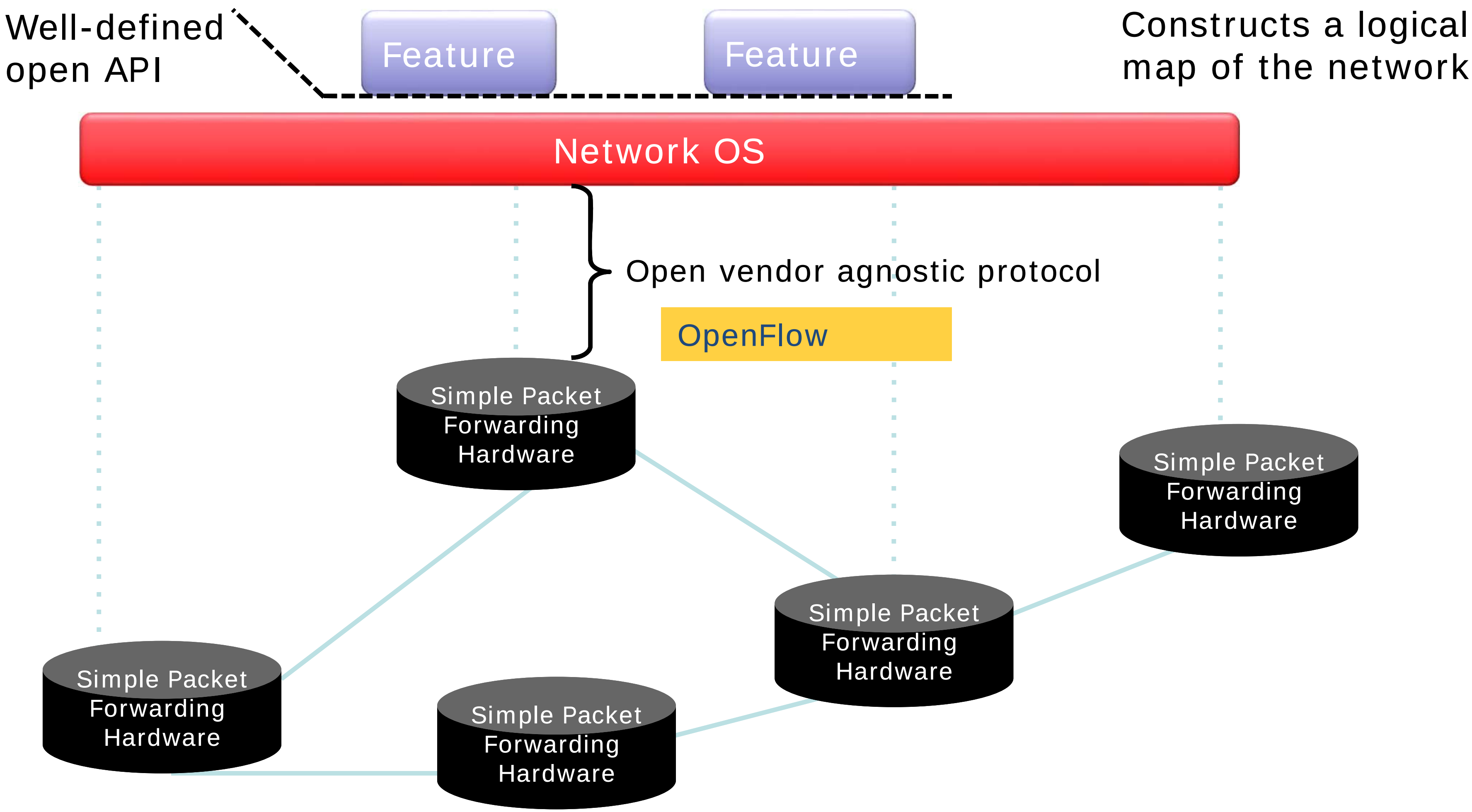
- SDN and its Relatives
- How does OpenFlow work?
- SDN/OpenFlow Controllers

Reminder: Classical Network Architecture

- Distributed control plane
- Distributed routing protocols: OSPF, IS-IS, BGP, etc.



Reminder:
Software Defined Network



SDN and its Relatives

Bernhard Plattner

SDN has a Few Relatives

- Key ideas present in some form in
  - Active Networks (1996 -): Capsules and programmable networks approaches
  - IETF ForCES Working Group (2003): Forwarding and Control Element Separation
  - 4D (2005): A new Approach to Network Control and Management, proposing a network-wide view
  - Sane (2006): Protection layer, logically centralized
  - Signaling System No. 7 (SS7)
  - Ethane (2007): The ancestor of OpenFlow

See also: Nick Feamster, Jennifer Rexford, Ellen Zegura: The Road to SDN: An Intellectual History of Programmable Networks

ForCES Framework, RFC 3746 (2004)

```
   -------------------------     -------------------------
   |   Control Blade A     |     |   Control Blade B     |
   |        (CE)           |     |        (CE)           |
   -------------------------     -------------------------
           ^   |                         ^   |
           |   |                         |   |
           |   V                         |   V
   ---------------------------------------------------------
   |                 Switch Fabric Backplane               |
   ---------------------------------------------------------
          ^   |           ^   |                ^   |
          |   |           |   |    . . .       |   |
          |   V           |   V                |   V
      ------------   ------------          ------------
      |Router    |   |Router    |          |Router    |
      |Blade #1  |   |Blade #2  |          |Blade #N  |
      |   (FE)   |   |   (FE)   |          |   (FE)   |
      ------------   ------------          ------------
          ^   |           ^   |                ^   |
          |   |           |   |    . . .       |   |
          |   V           |   V                |   V
```

Figure 1. A router configuration example with separate blades.

ForCES: CE and FE Implemented Separately

```
     -------         -------
     | CE1 |         | CE2 |
     -------         -------
        ^               ^
        |               |
        V               V
============================================ Ethernet
   ^       ^     . . .       ^
   |       |                 |
   V       V                 V
  -------  -------         --------
  | FE#1|  | FE#2|         | FE#n |
  -------  -------         --------
   ^  |     ^  |             ^  |
   |  |     |  |             |  |
   |  V     |  V             |  V
```

Figure 2. A router configuration example with separate boxes.

ForCES: Example CE / FE Functions

```
-------------------------------------------------
|       |       |       |       |       |       |
|OSPF   |RIP    |BGP    |RSVP   |LDP    |. . .  |
|       |       |       |       |       |       |
-------------------------------------------------
|               ForCES Interface                |
-------------------------------------------------
                        ^   ^
                ForCES  |   |data
                control |   |packets
                messages|   |(e.g., routing packets)
                        v   v
-------------------------------------------------
|               ForCES Interface                |
-------------------------------------------------
|       |       |       |       |       |       |
|LPM Fwd|Meter  |Shaper |NAT    |Classi-|. . .  |
|       |       |       |       |fier   |       |
-------------------------------------------------
|               FE resources                    |
-------------------------------------------------
```

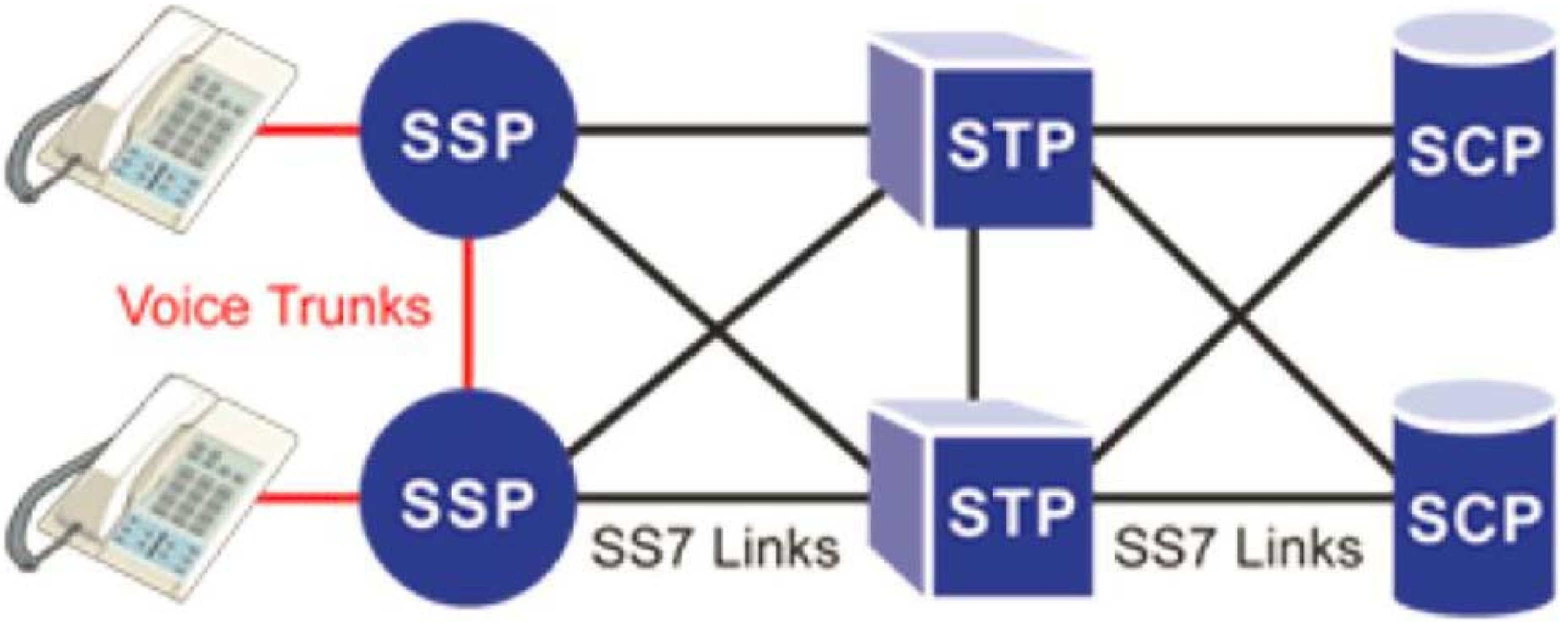
Figure 3. Examples of CE and FE functions.

Signaling System No. 7

- Basic call setup, management, and tear down
- Wireless services such as personal communications services (PCS), wireless roaming, and mobile subscriber authentication
- Toll-free (800/888) and toll (900) wireline services
- Enhanced call features such as call forwarding, calling party name/number display…
- Efficient and secure worldwide telecommunications

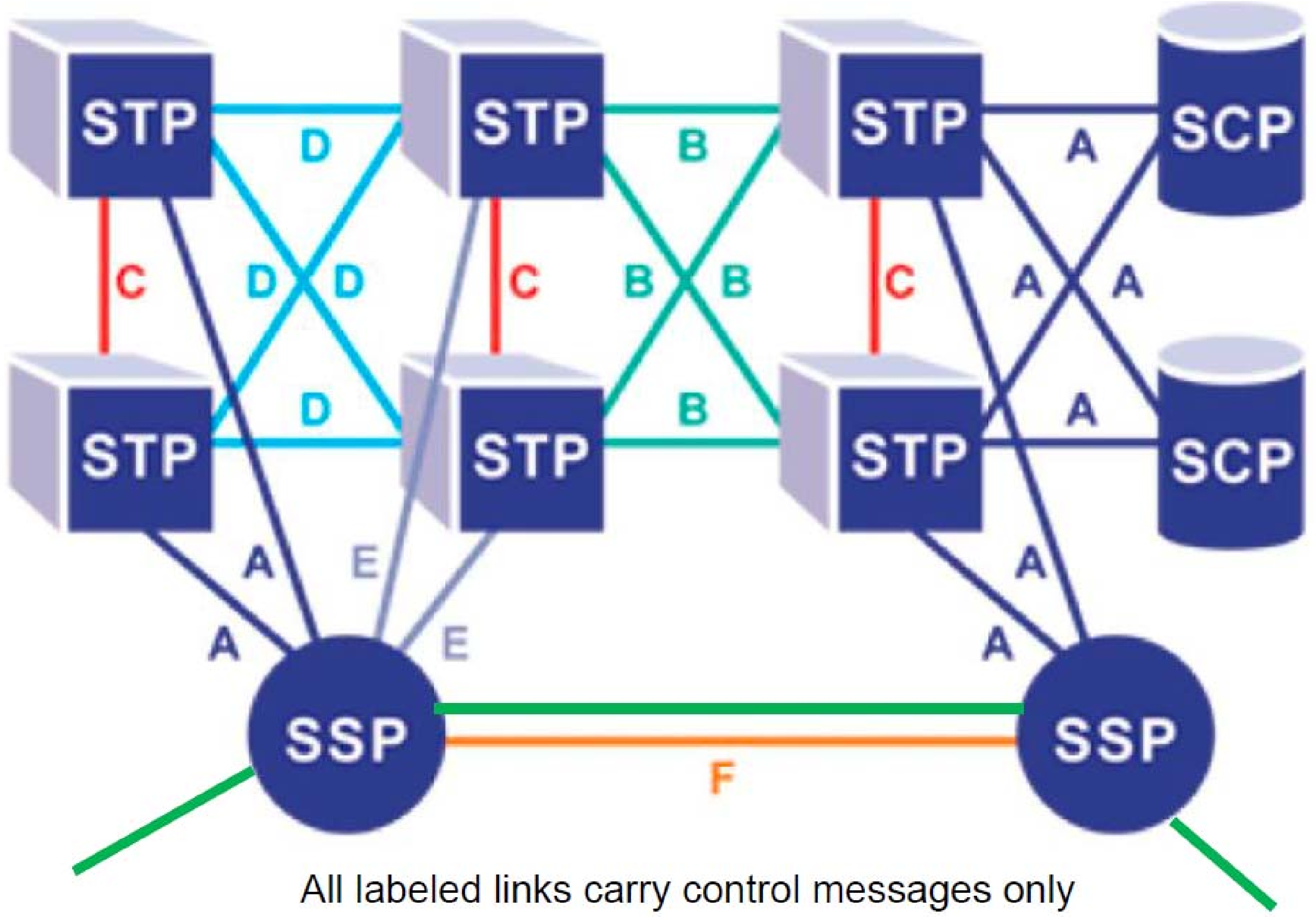
Source: www.pt.com, SS7 tutorial

SS7 Basic Architecture



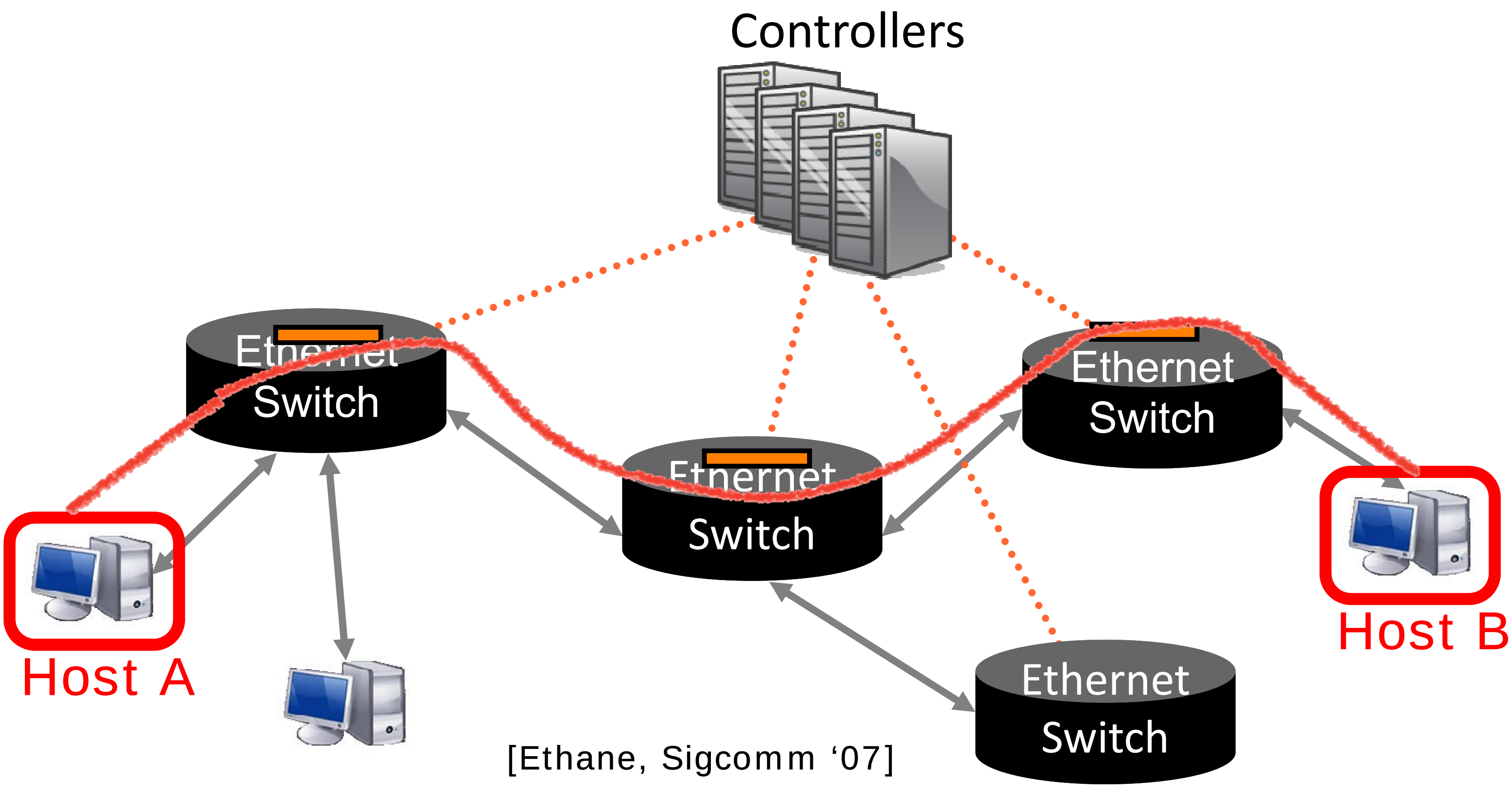
SSP (Service Switching Point)
STP (Signal Transfer Point)
SCP (Service Control Point)

SS7 Link Types and Redundant Layout



Ethane: A Precursor to OpenFlow

- Centralized, reactive, per-flow control



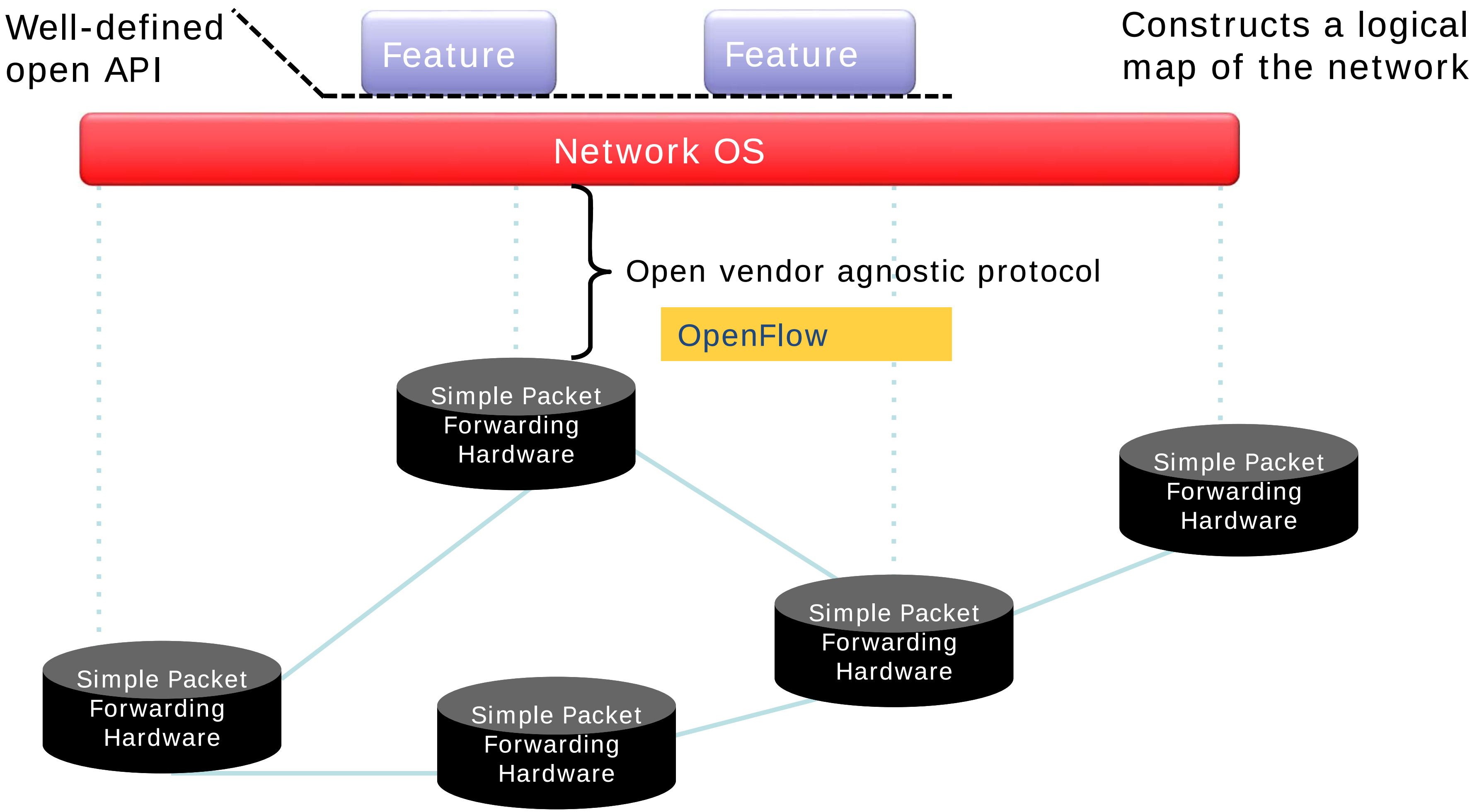
[Ethane, Sigcomm '07]

How does OpenFlow work?

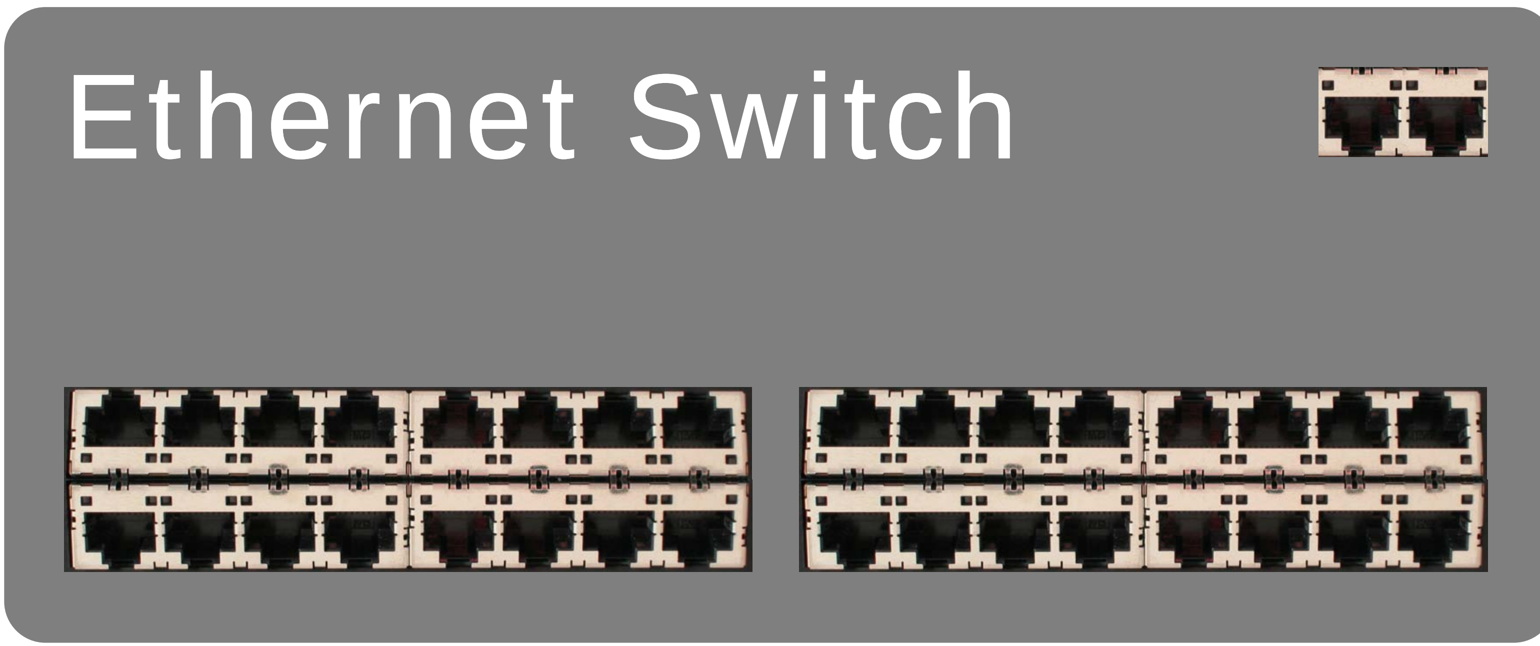
Xenofontas Dimitropoulos

Including open material from: Brandon Heller, Nick McKeown, Guru Parulkar, Scott Shenker, Yashar Ganjali, Rob Sherwood

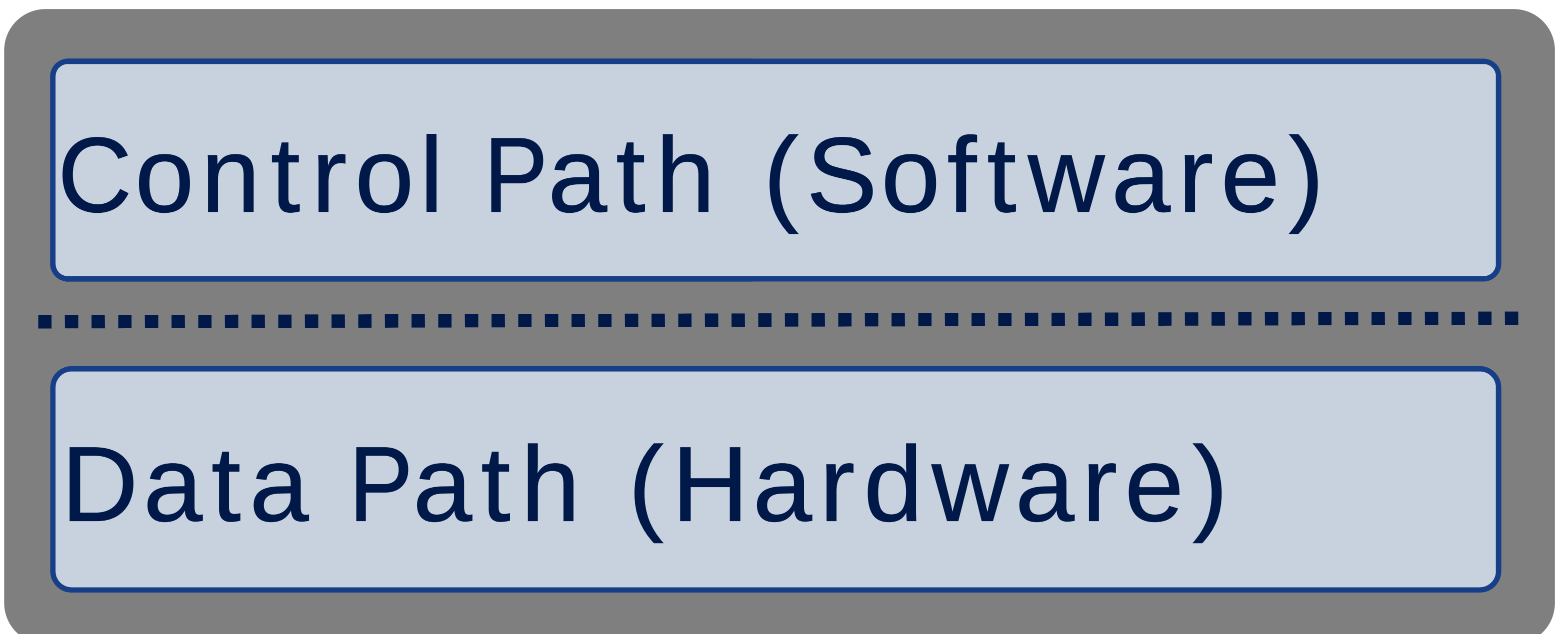
OpenFlow: A “southbound” API



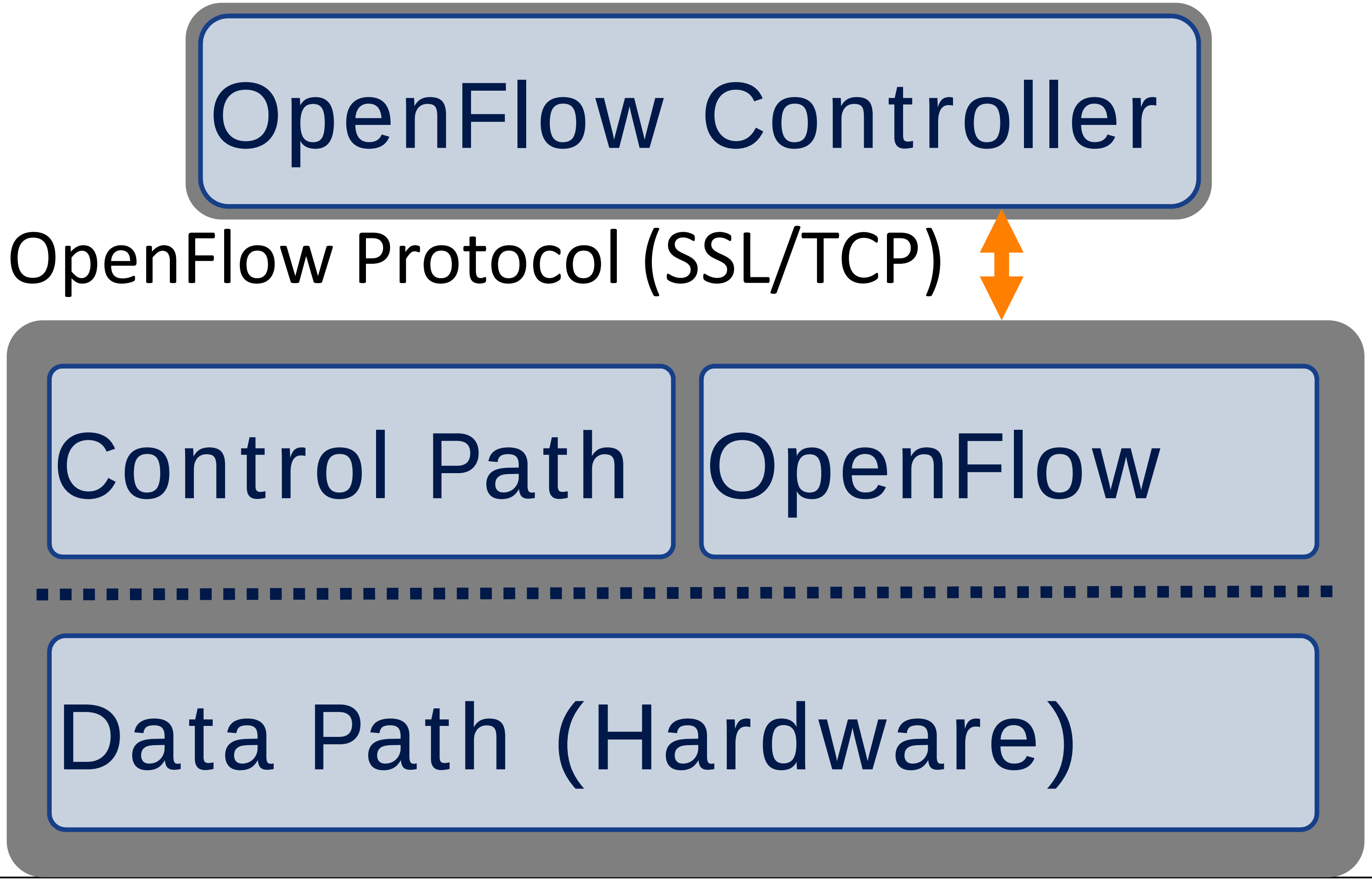
Ethernet Switch (Revisited)



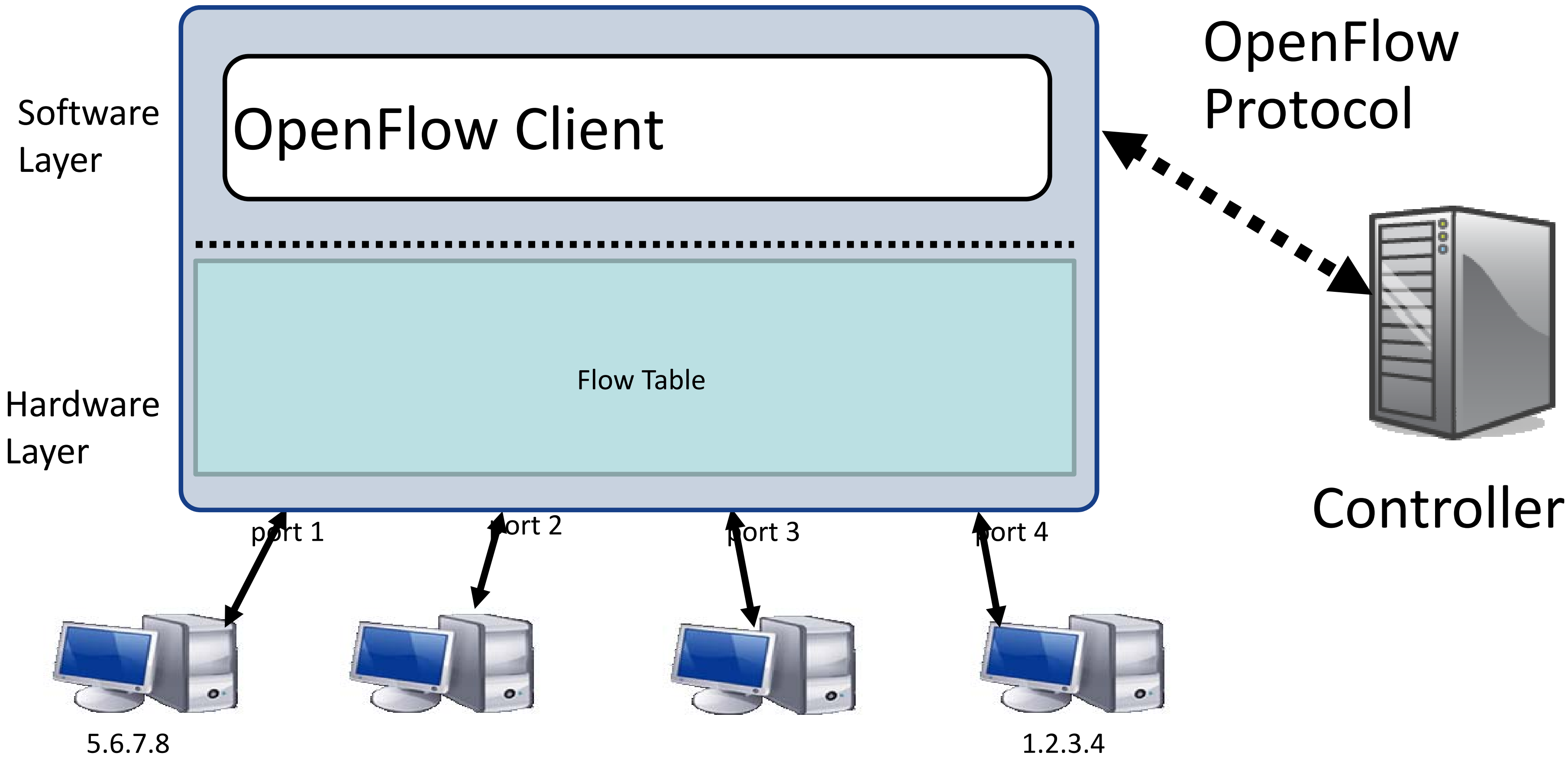
Control Path vs. Data Path (Revisited)



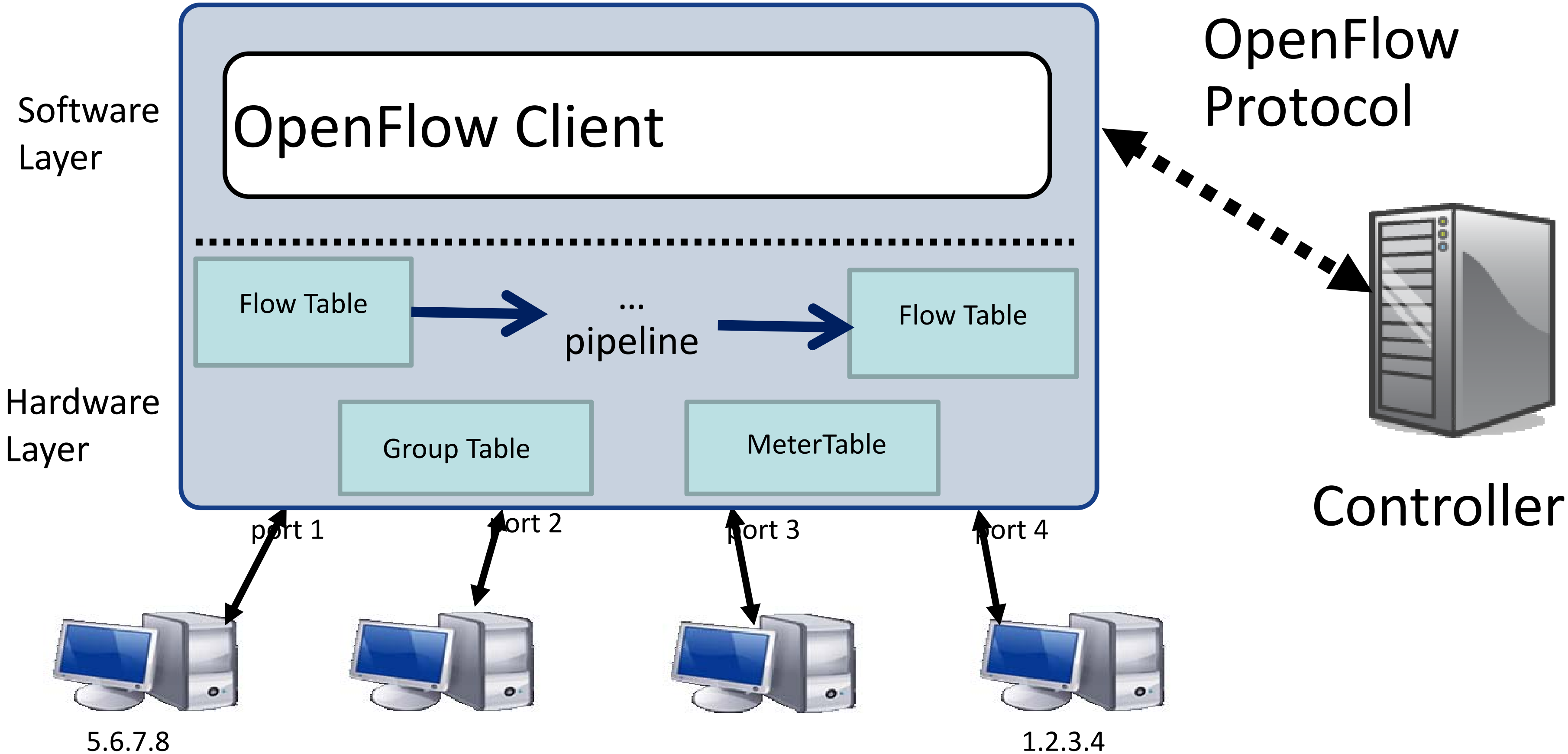
OpenFlow Protocol (Revisited)



Main Components of OpenFlow v1.0 Switch



Main Components of OpenFlow v1.4 Switch



Flow Table Entries

Main components of a flow entry in a flow table	
Match fields	To match against packets. These consist of the ingress port and packet headers
Priority	Matching precedence of the flow entry
Counters	e.g. packet and byte counters
Instructions	Determine action set or pipeline processing
Timeouts	Maximum amount of time or idle time before flow is expired by the switch
Cookies	Opaque data value chosen by the controller. Not used when processing packets.

Switch Port	VLAN ID	VLAN Pcp	MAC Src	MAC Dst	Eth Type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport

The match field contains either a specific value or a "wildcard"

Match / Action Examples (Revisited)

TECHNISCHE UNIVERSITÄT DARMSTADT

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

Examples (Revisited)

TECHNISCHE UNIVERSITÄT DARMSTADT

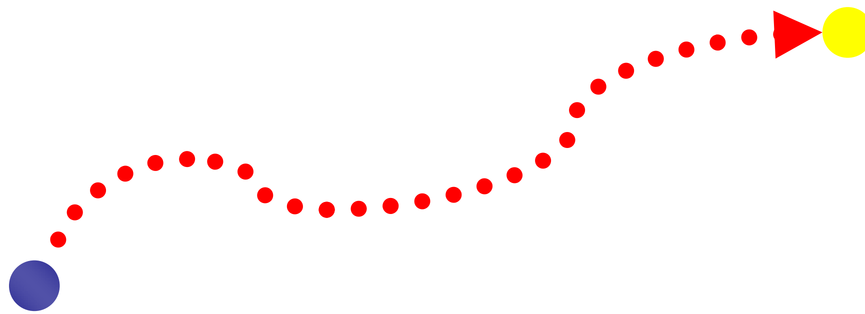
Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

Flowspace Revisited



What is a flow?

- Application flow
- All http
- Jim's traffic
- All packets to Canada
- …

Types of action

- Allow/deny flow
- Route & re-route flow
- Isolate flow
- Remove flow

Properties of a Flow-based Substrate

- We need flexible definitions of a flow
  - Unicast, multicast, multipath, waypoints
  - Different aggregations
- We need direct control over flows
  - Flow as an entity we program: To route, to move, …
- Exploit the benefits of packet switching
  - It works and is universally deployed
  - It is efficient (when kept simple)

Substrate: “Flowspace”: Headers as a protocol-agnostic collection of bits

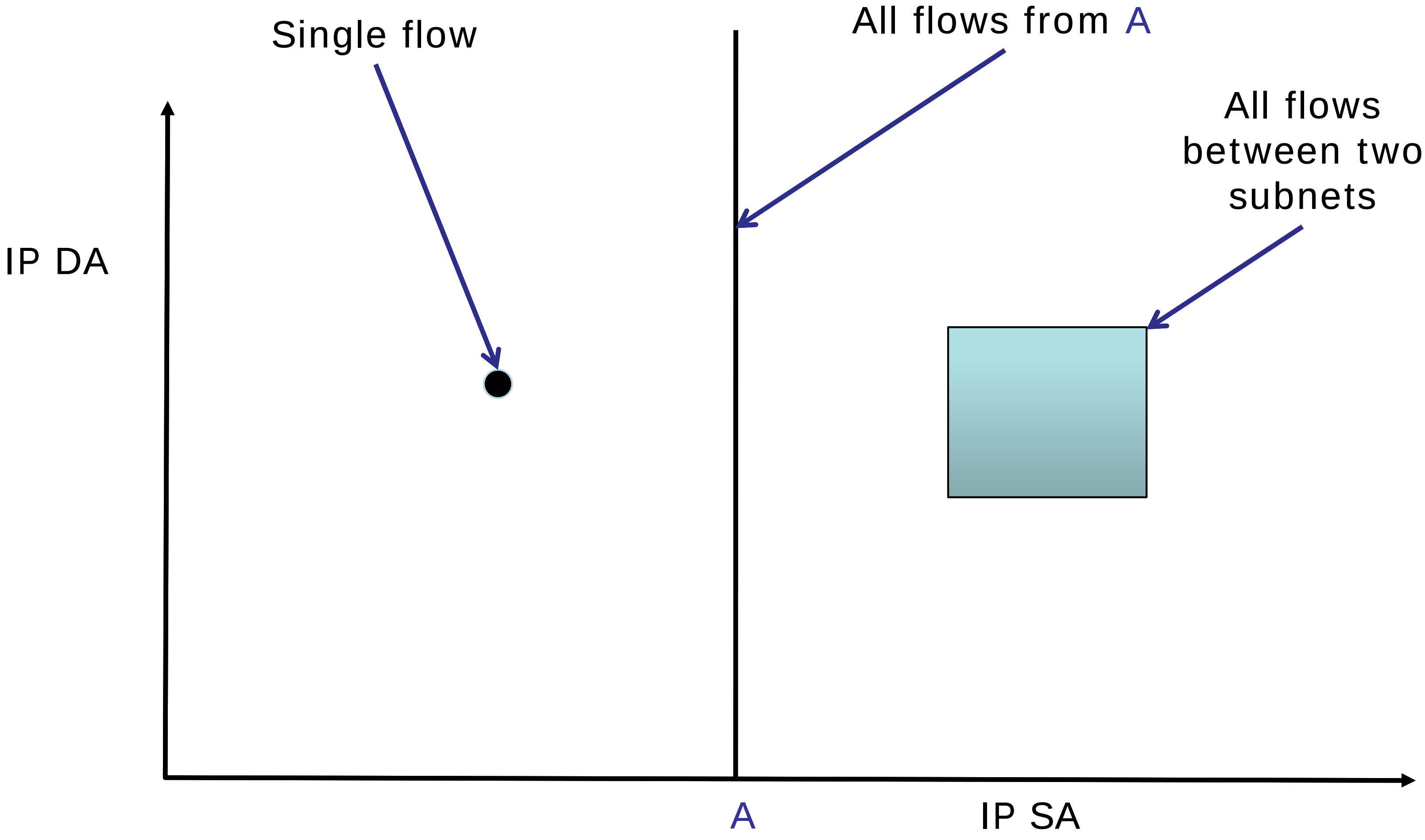
TECHNISCHE UNIVERSITÄT DARMSTADT

Ethernet DA, SA, etc	**IP** DA, SA, etc	**TCP** DP, SP, etc	Payload

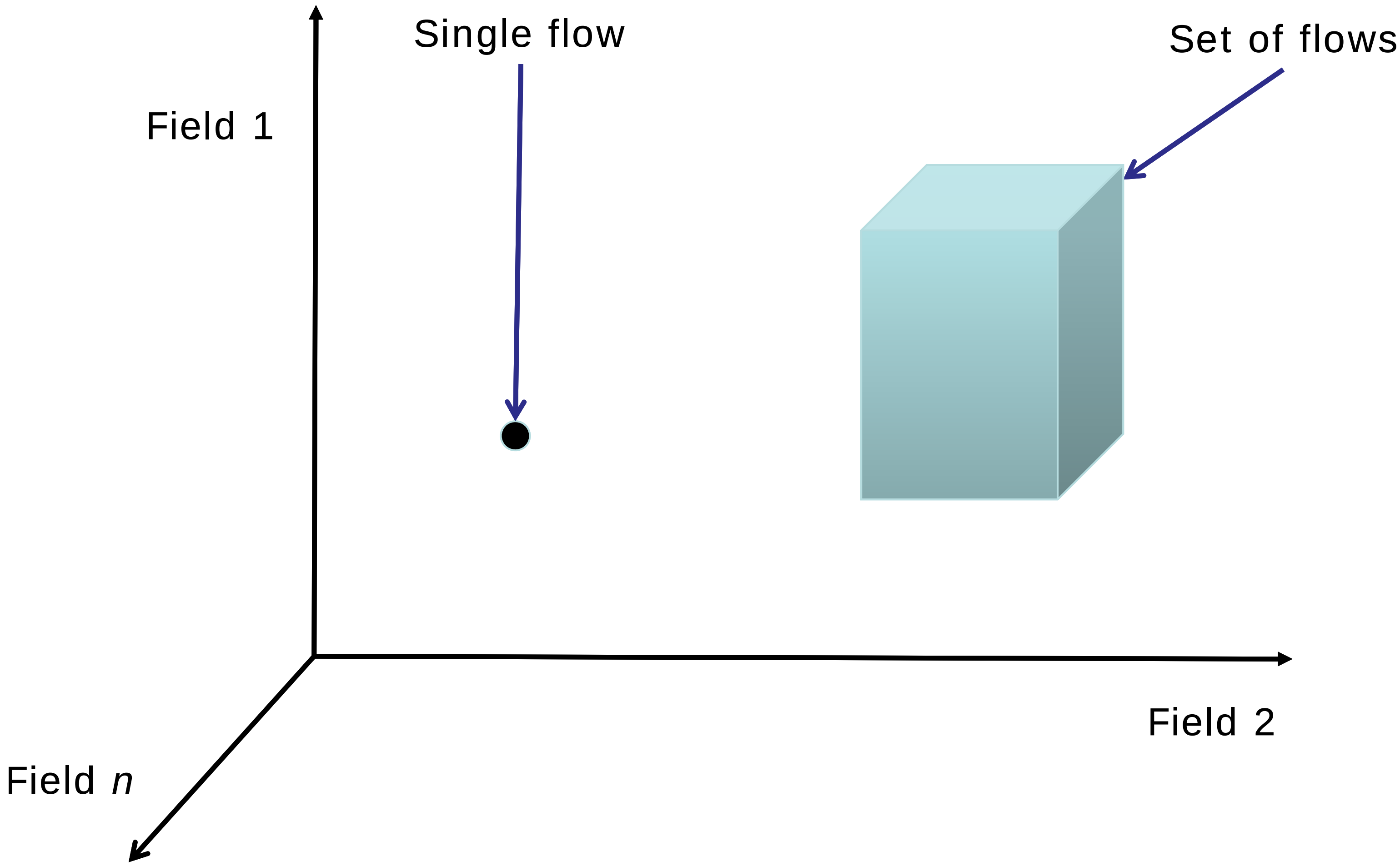
Collection of bits to plumb flows
(of different granularities)
between end points

Header User-defined flowspace	Payload

“Flowspace”: A way to think about flows defined by match fields



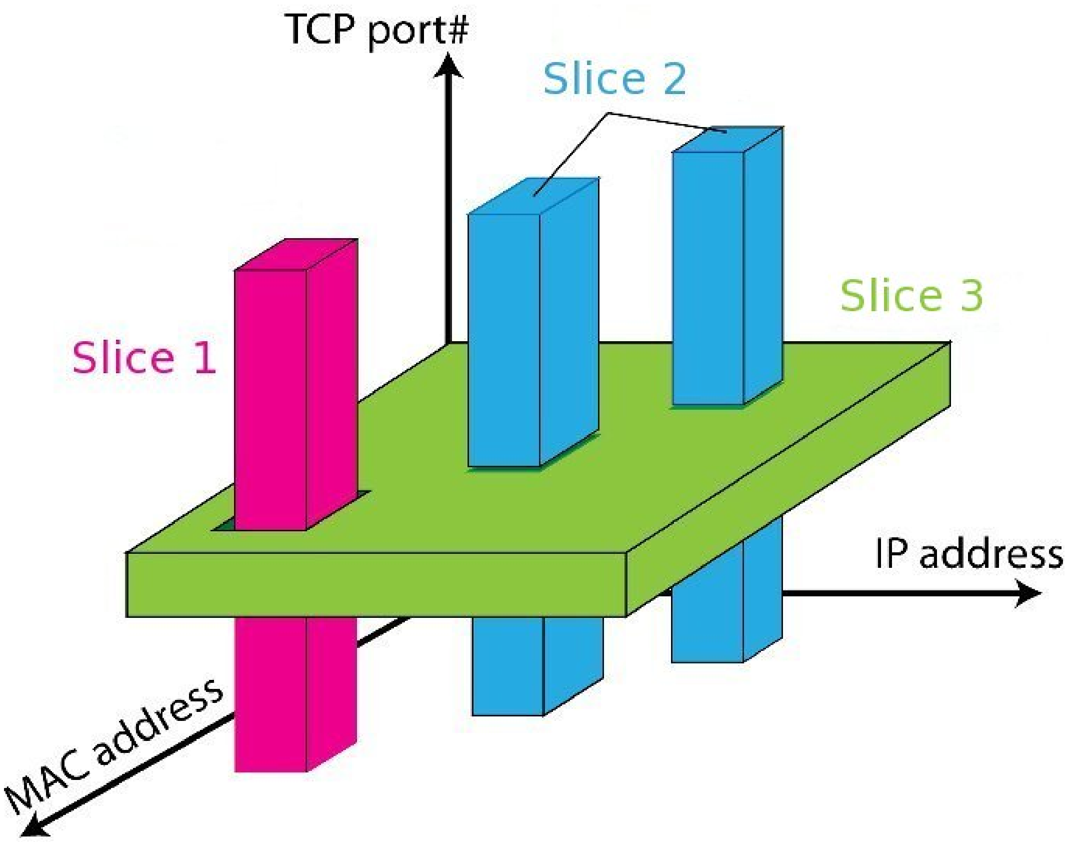
Flowspace: Generalization



Features of Flowspace

- Backwards compatible
  - Current layers are a special case
  - No end points need to change
- Easily implemented in hardware
  - e.g. TCAM flow-table in each switch
- Strong isolation of flows
  - Simple geometric construction
  - Can prove which flows can/cannot communicate

FlowSpace: Maps Packets to Slices



Different network slices occupy different parts of the flowspace

Table Miss

- Packets for which no flow has been defined are normally sent to the controller
- The controller then defines a new flow for that packet and creates one or more flow table entries
- The packet is then processed as determined by the newly created flow entries
- By default packets unmatched by flow entries are dropped

Instructions

- Carry out an action on the packet or add actions to be carried out later.

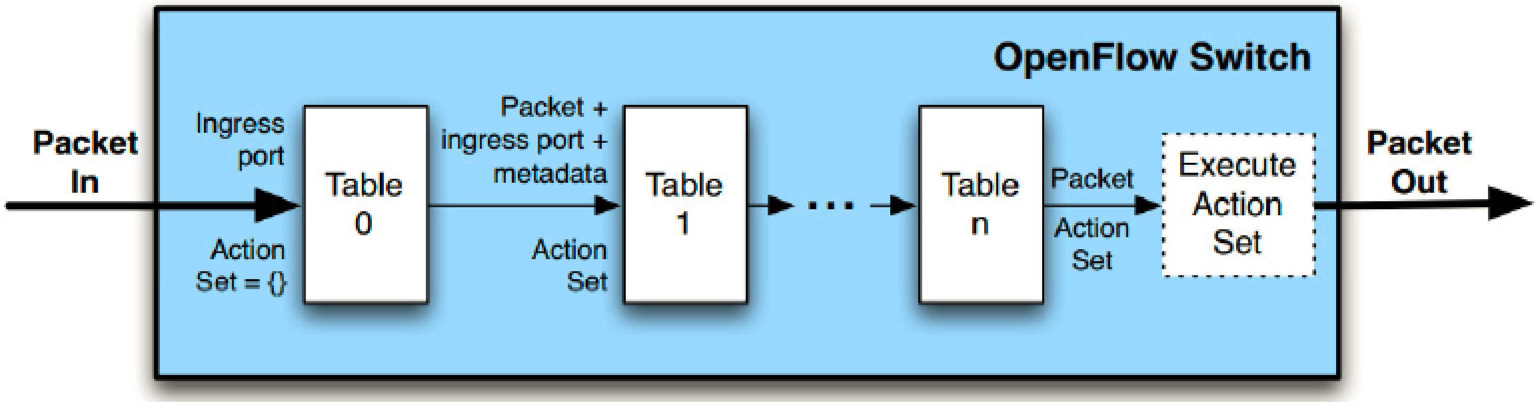
- Instructions may also direct a packet to another flow table.

- An instruction may specify a group or a meter identifier.

Group and Meter Tables

- Groups represent sets of actions for more complex forwarding semantics
  - E.g. flooding, multipath
- A meter table consists of per-flow meters.
  - A meter measures and enables controlling the rate of packets assigned to it and allows to implement simple QoS operations
  - E.g., rate limit packets to controller

OpenFlow Pipeline



Packets are matched against multiple tables in the pipeline

OpenFlowSwitch Specification Version > 1.1.0

OpenFlow Message Types

- Controller-to-switch messages
  - Manage flow entries
  - Request info on switch capabilities and counters
  - Send a packet back to a switch
- Asynchronous messages
  - Send to controller a packet that does not match
  - Inform the controller that a timer has expired or that an error has occurred
- Symmetric messages
  - Hello and echo messages

OpenFlow Key Messages

Message	Direction	Description
Packet-In	Switch->Controller	Transfer the control of a packet to the controller. Packet-in events can be configured to buffer packets
Packet-Out	Controller->Switch	Instruct switch to send a packet out of a specified port. Send in response to Packet-in messages.
Modify-State	Controller->Switch	Add, delete and modify flow/group entries in the flow tables and to set switch port properties
Flow-Removed	Switch->Controller	Inform the controller about the removal of a flow entry from a flow table

Openflow's Short History

- OF v1.0 (end of 2009):
  - Single table, L2+IPv4 focused matching
- OF v1.1 (Mar 2011):
  - Multiple tables, MPLS + VLAN matching, multipath forwarding: ECMP, groups
- OF v1.2 (Dec 2011): "Extensible Protocol"
  - Extensible match & actions (TLV), IPv6, multiple controllers
- OF v1.3 (June 12):
  - Better expression of capabilities of a switch, meters, multiple parallel channels between switch and controller.
- OF v1.4 (Aug 13):
  - Improve extensibility, better support for optical ports, many other incremental improvements.

OpenFlow Support

- Open Networking Foundation was founded in 2011 to develop and standardize OpenFlow.
  - Members include Cisco, Facebook, Google, HP, IBM and Juniper Networks.
- Juniper and start-ups Nicira and Big Switch are warm supporters of OpenFlow
- Vendors, such as Cisco, IBM, NEC and HP, have implemented OpenFlow in existing products
- Cisco's SDN initiative is called Open Network Environment (ONE)

Summary of Key SDN/Openflow Features

TECHNISCHE UNIVERSITÄT DARMSTADT

- Separate data from control
- Open control API
- Define a generalized flow table
  - Flexible and generalized flow abstraction
  - Unified view of layers1-7
- Backward compatible
  - Though allows completely new header
- Virtualization of the data and control plane

OpenFlow: A Pragmatic Compromise

TECHNISCHE UNIVERSITÄT DARMSTADT

- \+ Speed, scale, fidelity of vendor hardware
- \+ Flexibility and control of software and simulation
- \+ Vendors don't need to expose implementation
- \+ Leverages hardware inside most switches today (ACL tables)

- \- Least-common-denominator interface may prevent using all hardware features
- \- Limited table sizes
- \- Switches not designed for this
- \- New failure modes to understand
- \- Security?

Research Questions

- How to design the interfaces
  - To the hardware («southbound»)
  - API of the network operating system(«northbound»)
- Design of the virtualization layer
- Design of the network operating system
- How to achieve perfect isolation between different slices
- How to develop applications (network programming language?)
- What about security? Attack surface increased/decreased? Secure app development?

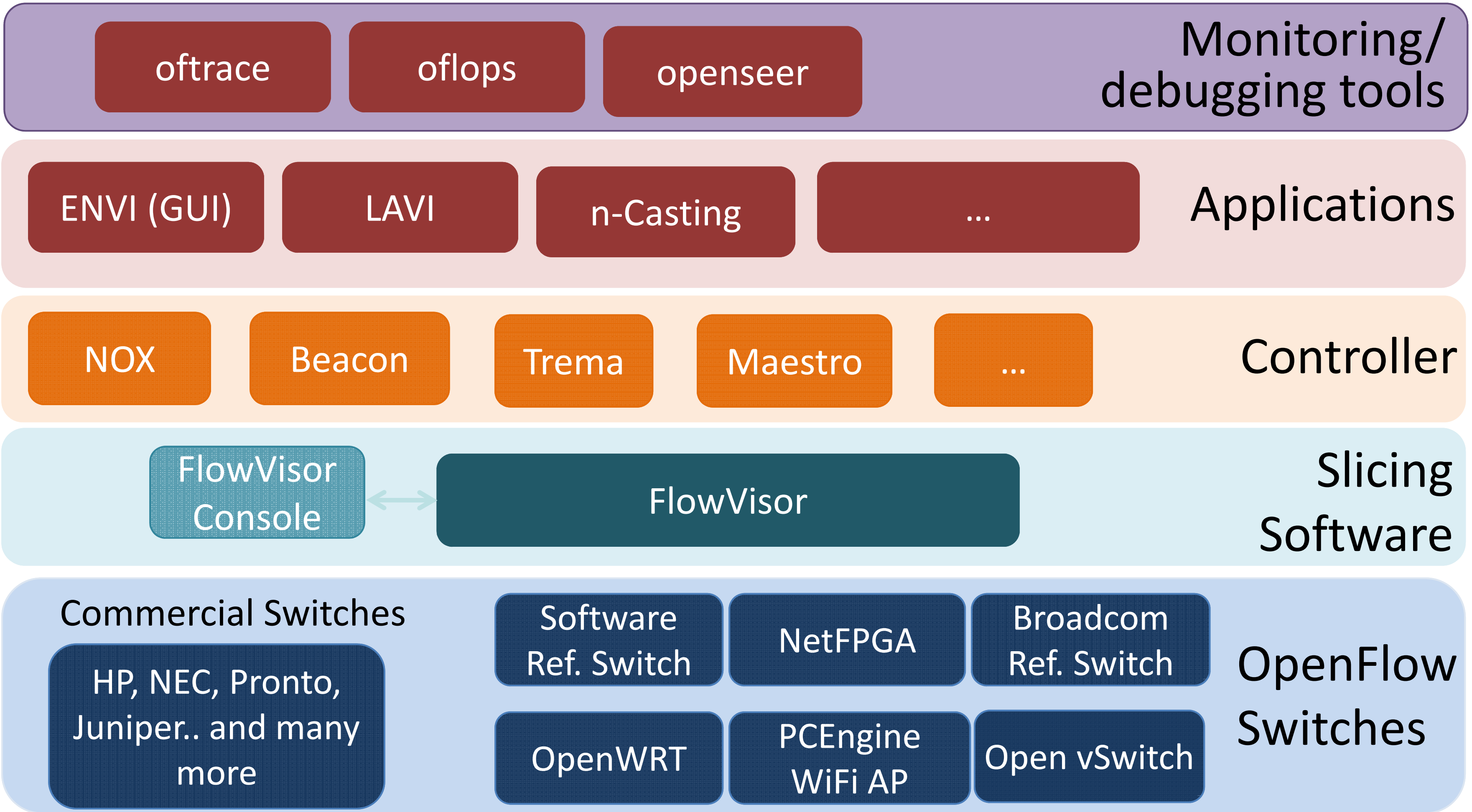
Further Reading

- OpenFlow Switch Specification
  - https://www.opennetworking.org/images/stories/downloads/sdnresources/onfspecifications/openflow/openflowspec-v1.4.0.pdf
- OpenFlow Hands-On Tutorial
  - http://www.openflow.org/wk/index.php/OpenFlow_Tutorial
- OpenFlow Apps Research Demos
  - http://www.openflow.org/videos/
- OpenFlow White Paper
  - http://archive.openflow.org/documents/openflow-wp-latest.pdf
- SDN/OpenFlow Reading List
  - http://www.nec-labs.com/~lume/sdn-reading-list.html

SDN / OpenFlow Controllers

Xenofontas Dimitropoulos

The SDN Stack



Open Controllers

Name	Lang	Platforms	License	Original Author	Notes
OpenFlow Reference	C	Linux	OpenFlow License	Stanford/ Nicira	not designed for extensibility
NOX	Python, C++	Linux	GPL	Nicira	actively developed
Beacon	Java	Win, Mac, Linux, Android	GPL (core), FOSS Licenses for your code	David Erickson (Stanford)	runtime modular, web UI framework, regression test framework
Maestro	Java	Win, Mac, Linux	LGPL	Zheng Cai (Rice)	
Trema	Ruby, C	Linux	GPL	NEC	includes emulator, regression test framework
RouteFlow	?	Linux	Apache	CPqD (Brazil)	virtual IP routing as a service

Open Controllers (2)

Name	Lang	Platforms	License	Original Author	Notes
OpenFaucet	Python				Library
Mirage	OCaml				
POX	Python	Any			
Floodlight	Java	Any		BigSwitch, based on Beacon	
OpenDay Light	Java	Any	Eclipse Public License (EPL)	Linux Foundation Collaborative Project	

Too many to easily list of keep track of...