UCL
Université catholique de Louvain

INGI2145: CLOUD COMPUTING (Fall 2014)

Cloud Networks

12 December 2014

Certain lecture slides adapted judiciously from Duke CPS590 by T. Benson and Cornell CS5413 by H. Weatherspoon

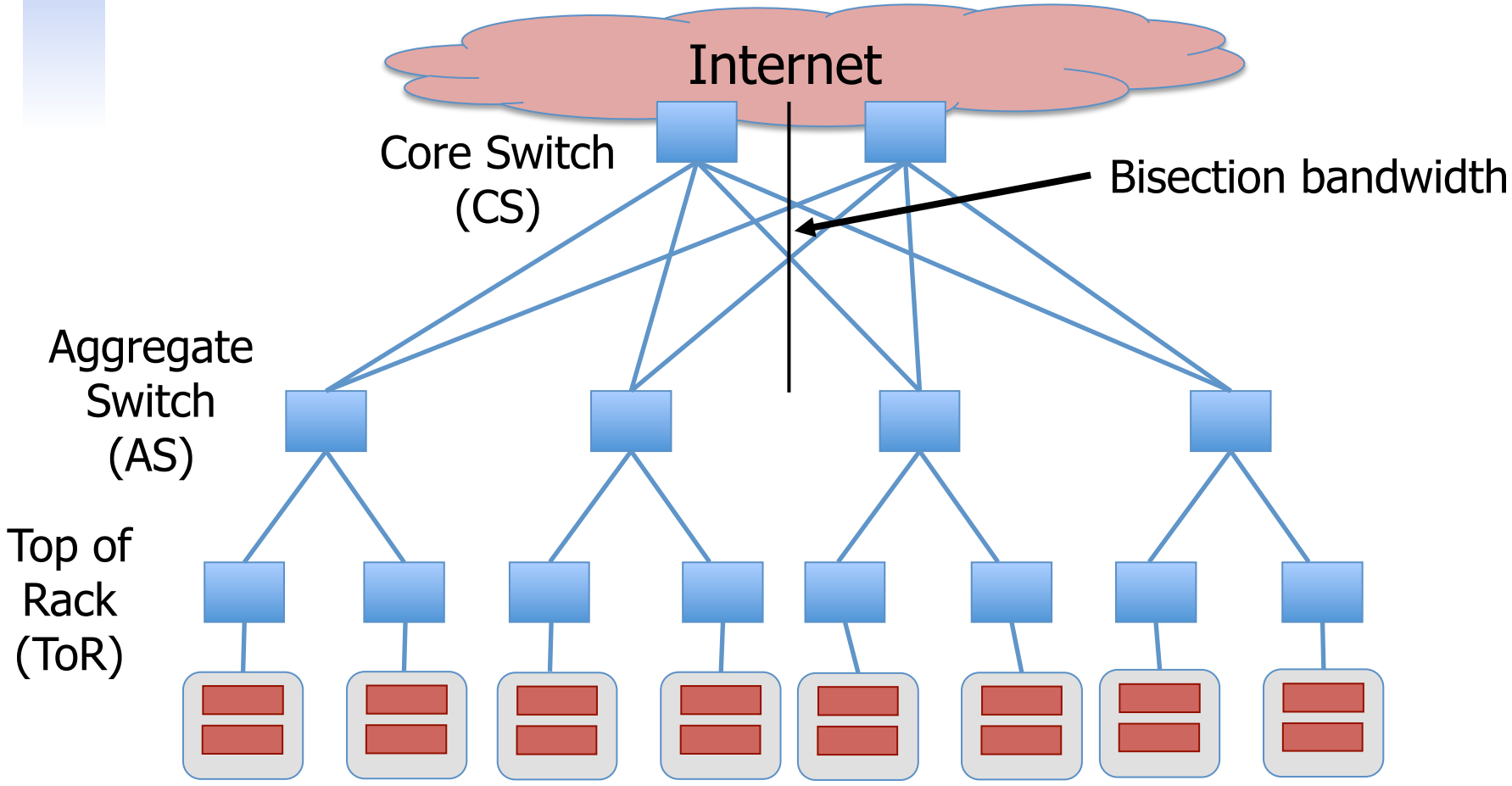
Today

- Modern data centers need modern networking!
- FatTree-based DC fabric
- Software-Defined Networking

Datacenter!

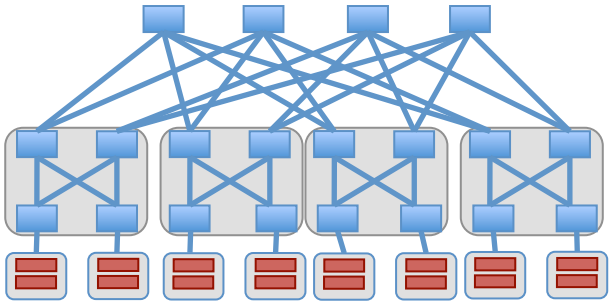
- 10s to 100s of thousands of servers
- Clusters of homogeneous servers
  - 10s of racks with 10s of serves in each rack
  - Each server: many cores, memory, network interfaces, local storage (HDD and/or SDD)
- Individual requests may be served by 10s to even 1000s of serves
  - Performance based on slowest response
  - Network congestion affects performance
  - Overprovisioning large-scale networks is too costly

Networking in Data Centers

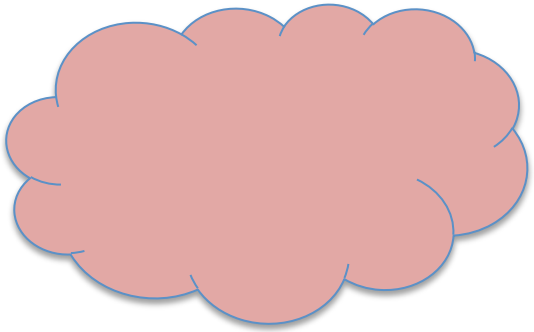


Goals of Data Center Networks

- Support diverse application
  - High throughput/low latency
  - Utilize multiple paths



- Scale to cloud size
  - Support large number of addresses
  - 5-10 million VMs

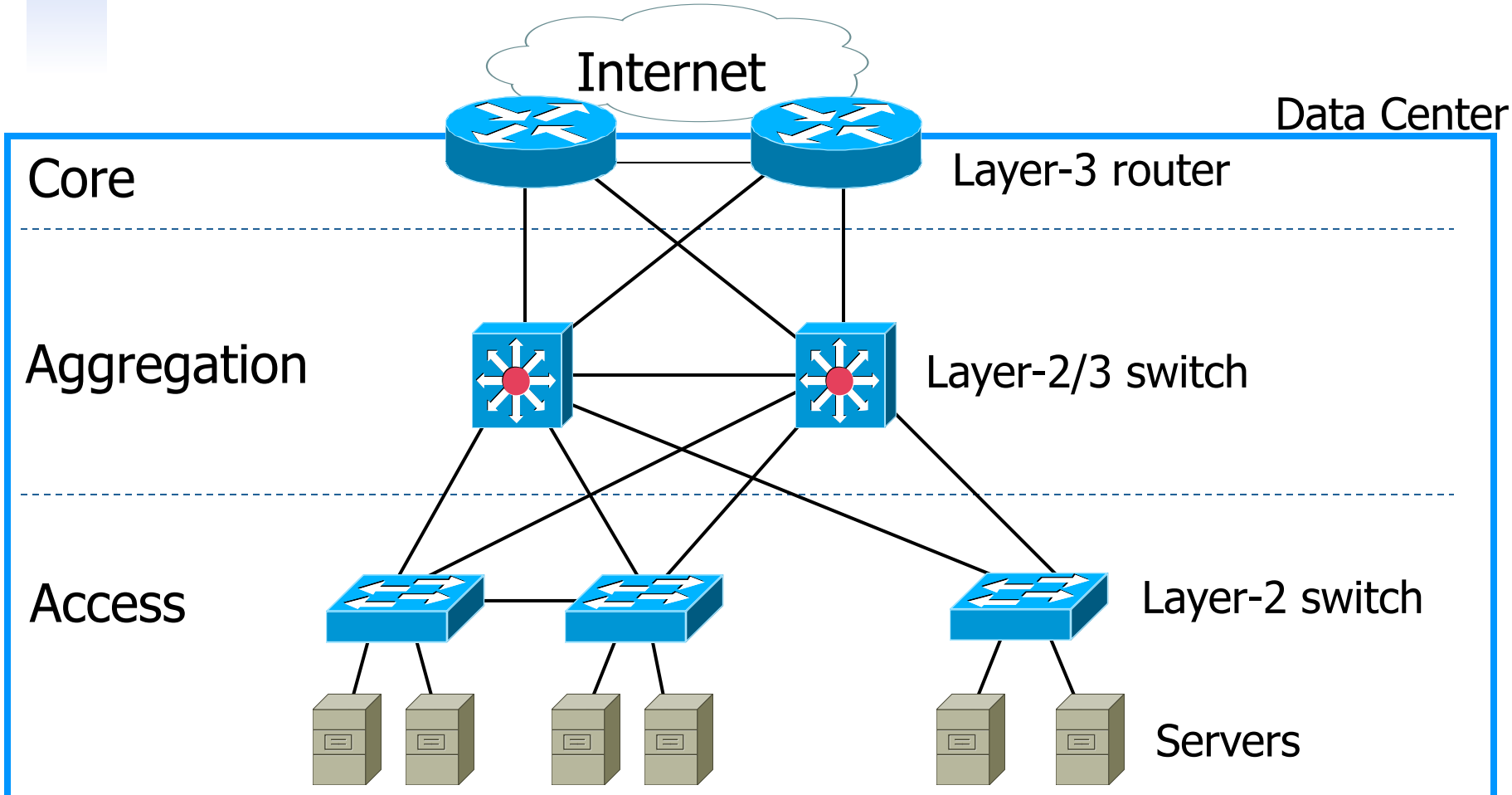


- Support flexible resource utilization
  - Support seamless VM mobility

Goals of Data Center Networks

- Uniform high capacity:
  - Maximum rate of server to server traffic flow should be limited only by capacity on network cards
  - Assigning servers to service should be independent of network topology
- Performance isolation:
  - Traffic of one service should not be affected by traffic of other services
- Layer-2 semantics:
  - Easily assign any server to any service
  - Configure server with whatever IP address the service expects
  - VM keeps the same IP address even after migration

Common data center topology



Problem with traditional DC topology

- Single point of failure
- Over subscript of links higher up in the topology
  - Trade off between cost and provisioning

Problem with traditional DC topology

- Oversubscription:
  - Ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology
  - Lower the total cost of the design
  - Typical designs: factor of 2:5:1 (400 Mbps) to 8:1 (125 Mbps)
- Cost:
  - Edge: $7,000 for each 48-port GigE switch
  - Aggregation and core: $700,000 for 128-port 10GigE switches
  - Cabling costs are not considered!

Properties of desired solutions

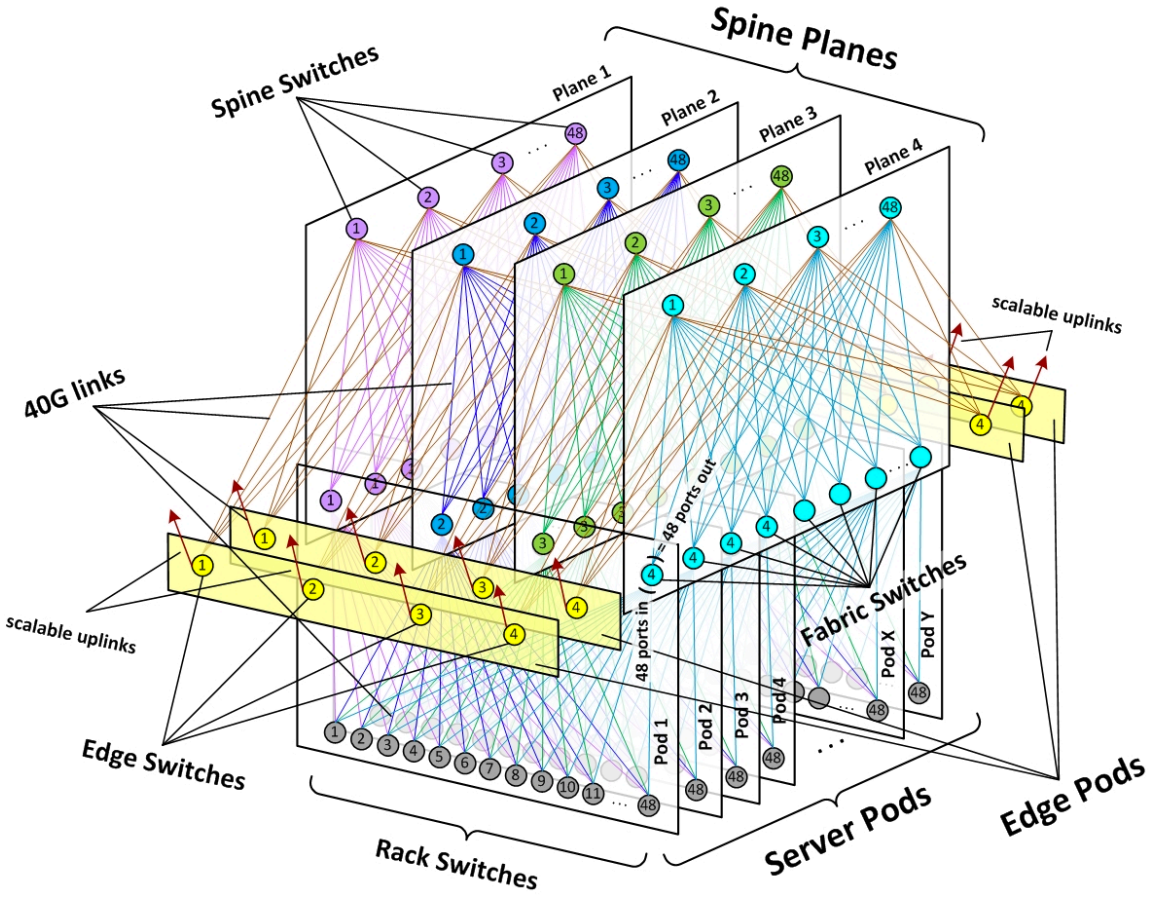
- Backwards compatible with existing infrastructure
  - No changes in application
  - Support of layer 2 (Ethernet)
- Cost effective
  - Low power consumption & heat emission
  - Cheap infrastructure
- Allows host communication at line speed

Backwards compatibility tradeoffs

- Specialized hardware and communication protocols, such as InfiniBand, Myrinet
  - Can scale to clusters of 1000s of nodes with high bandwidth
  - Expensive infrastructure, incompatible with TCP/IP applications
- Leverage commodity Ethernet switches and routers to interconnect cluster machines
  - Backwards compatible with existing infrastructures, low-cost
  - Aggregate cluster bandwidth scales poorly with cluster size, and achieving the highest levels of bandwidth incurs non-linear cost increase with cluster size

Facebook Example

- Introduction to Facebook's data center fabric



Today

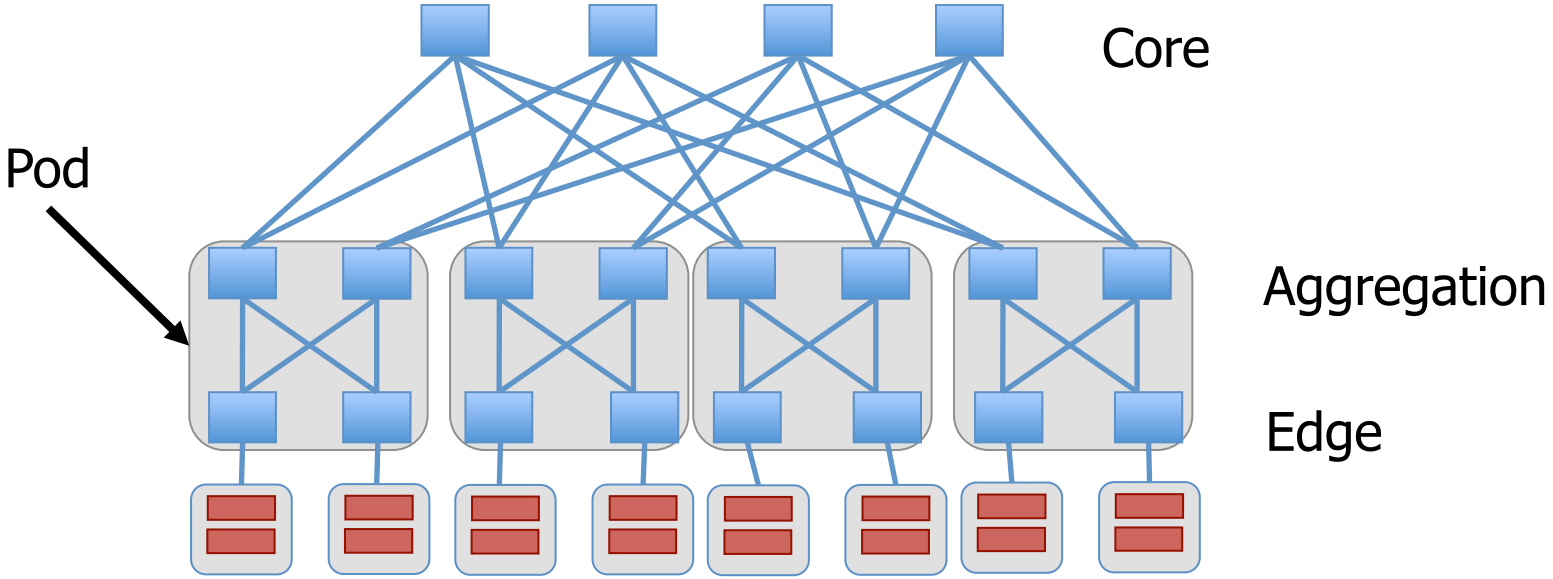
- Modern data centers need modern networking!
- FatTree-based DC fabric
- Software-Defined Networking

FatTree based solution

- Adopt a special instance of a Clos topology
- Similar trends in telephone switches led to designing a topology with high bandwidth by interconnecting smaller commodity switches

FatTree based solution

- Connect servers using a fat tree topology
  - K-ary fat tree: 3-layer topology (edge, aggregation and core)
    - each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches
    - each edge switch connects to k/2 servers & k/2 aggr. switches
    - each aggr. switch connects to k/2 edge & k/2 core switches
    - $(k/2)^2$ core switches: each connects to k pods



FatTree based solution

- Why Fat Tree?
  - Fat tree has identical bandwidth at any bisections
  - Each layer has the same aggregated bandwidth
- Built out of cheap devices with uniform capacity
  - Each port supports same speed as server
  - All devices can transmit at line speed if packets are distributed uniform along available paths
- Great scalability
  - A k-port fat tree supports $k^3/4$ servers

Problems with a vanilla Fat Tree

- Layer 3 will only use one of the existing equal cost paths
  - Bottlenecks up and down the fat-tree
- Packet re-ordering occurs if layer 3 blindly takes advantage of path diversity
  - Further load may not necessarily be well-balanced
- Wiring complexity in large networks
  - Packing and placement technique

FatTree modified

- Enforce a special (IP) addressing scheme in DC
  - Allows host attached to same switch to route only through switch
  - Allows inter-pod traffic to stay within pod
  - unused.PodNumber.switchnumber.Endhost

FatTree modified

- Use two level look-ups to distribute traffic and maintain packet ordering
- First level is prefix lookup
  - used to route down the topology to servers
- Second level is a suffix lookup
  - used to route up towards core
  - maintain packet ordering by using same ports for same server
  - diffuses and spreads out traffic

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

→

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

Optimization: Flow classification

- Denote a flow as a sequence of packets
- pod switches forward subsequent packets of the same flow to same outgoing port
- And periodically reassign a minimal number of output ports

- Eliminates local congestion
- Assign traffic to ports on a per-flow basis instead of a per-host basis
  - Ensure fair distribution on flows

Optimization: Flow scheduling

- Pay attention to routing large flows, edge switches detect any outgoing flow whose size grows above a predefined threshold, and then send notification to a central scheduler
- The central scheduler tries to assign non-conflicting paths for these large flows

- Eliminates global congestion
- Prevent long lived flows from sharing same links
- Assign long lived flows to different links

Fault Tolerance

- Each switch in the network maintains a BFD (Bidirectional Forwarding Detection) session with each of its neighbors to determine when a link or neighboring switch fails
- Failure between upper layer and core switches
  - Outgoing inter-pod traffic, local routing table marks the affected link as unavailable and chooses another core switch
  - Incoming inter-pod traffic, core switch broadcasts a tag to upper switches directly connected signifying its inability to carry traffic to that entire pod, then upper switches avoid that core switch when assigning flows destined to that pod

Fault Tolerance

- Failure between lower and upper layer switches
  - Outgoing inter- and intra pod traffic from lower-layer,
    - the local flow classifier sets the cost to infinity and does not assign it any new flows, chooses another upper layer switch
  - Intra-pod traffic using upper layer switch as intermediary
    - Switch broadcasts a tag notifying all lower level switches, these would check when assigning new flows and avoid it
  - Inter-pod traffic coming into upper layer switch
    - Tag to all its core switches signifying its ability to carry traffic, core switches mirror this tag to all upper layer switches, then upper switches avoid affected core switch when assigning new flows

Evaluation

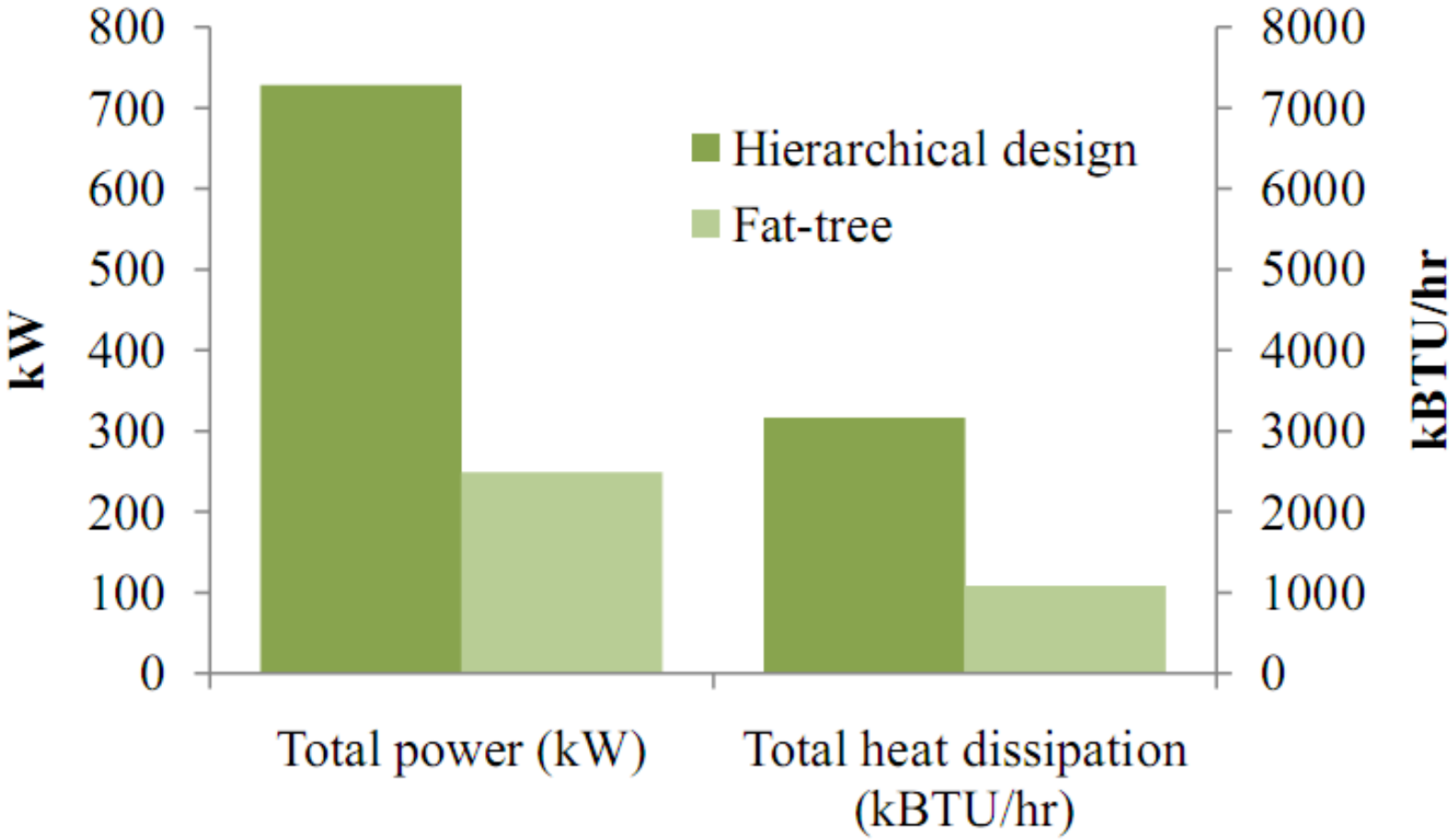
- Benchmark suite of communication mappings to evaluate the performance of the 4-port fat-tree using the TwoLevelTable switches, FlowClassifier ans the FlowScheduler and compare to hierarchical tree with 3.6:1 oversubscription ratio

Results: Network Utilization

Test	Tree	Two-Level Table	Flow Classification	Flow Scheduling
Random	53.4%	75.0%	76.3%	93.5%
Stride (1)	100.0%	100.0%	100.0%	100.0%
Stride (2)	78.1%	100.0%	100.0%	99.5%
Stride (4)	27.9%	100.0%	100.0%	100.0%
Stride (8)	28.0%	100.0%	100.0%	99.9%
Staggered Prob (1.0, 0.0)	100.0%	100.0%	100.0%	100.0%
Staggered Prob (0.5, 0.3)	83.6%	82.0%	86.2%	93.4%
Staggered Prob (0.2, 0.3)	64.9%	75.6%	80.2%	88.5%
Worst cases:				
Inter-pod Incoming	28.0%	50.6%	75.1%	99.9%
Same-ID Outgoing	27.8%	38.5%	75.4%	87.4%

Table 2: Aggregate Bandwidth of the network, as a percentage of ideal bisection bandwidth for the Tree, Two-Level Table, Flow Classification, and Flow Scheduling methods. The ideal bisection bandwidth for the fat-tree network is 1.536Gbps.

Results: Heat & Power Consumption



Perspective

- Bandwidth is the scalability bottleneck in large scale clusters
- Existing solutions are expensive and limit cluster size
- Fat-tree topology with scalable routing and backward compatibility with TCP/IP and Ethernet
- Large number of commodity switches have the potential of displacing high end switches in DC the same way clusters of commodity PCs have displaced supercomputers for high end computing environments

Other Data Center Architectures

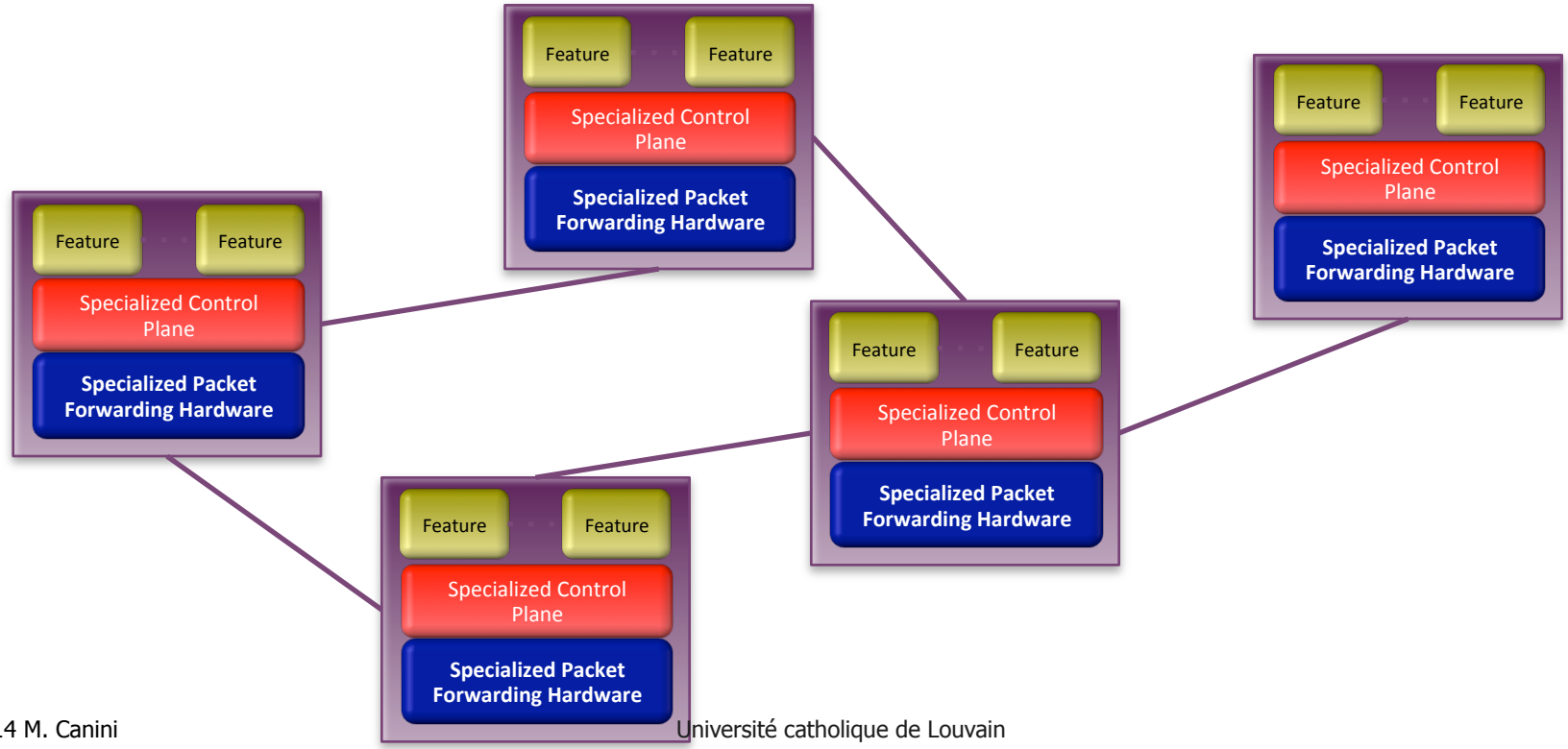
- A Scalable, Commodity Data Center Network Architecture
  - a new Fat-tree "inter-connection" structure (topology) to increases "bi-section" bandwidth
  - needs "new" addressing, forwarding/routing
- VL2: A Scalable and Flexible Data Center Network
  - consolidate layer-2/layer-3 into a "virtual layer 2"
  - separating "naming" and "addressing", also deal with dynamic load-balancing issues
- Other Approaches:
  - PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric
  - BCube: A High-Performance, Server-centric Network Architecture for Modular Data Centers

Today

- Modern data centers need modern networking!
- FatTree-based DC fabric
- Software-Defined Networking

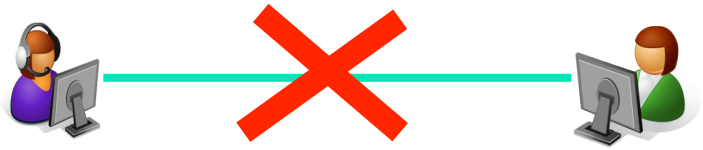
Classical network architecture

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

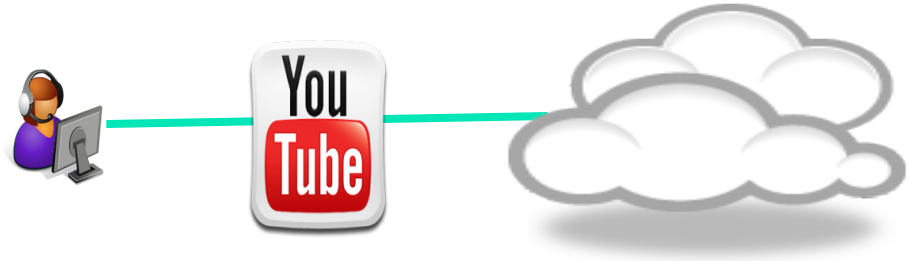


Network Policies

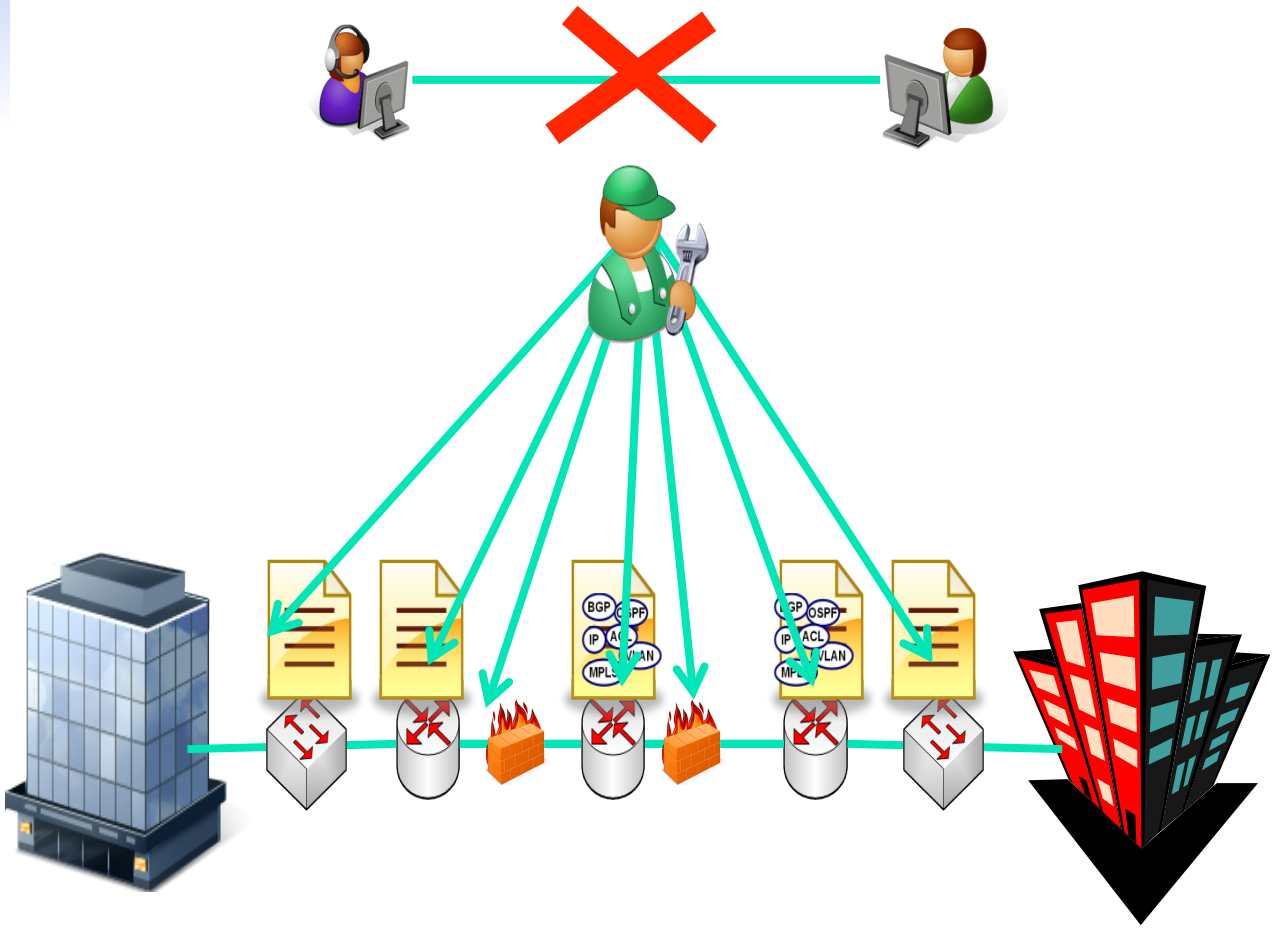
- Access control: reachability
  - Alice can not send packets to Bob



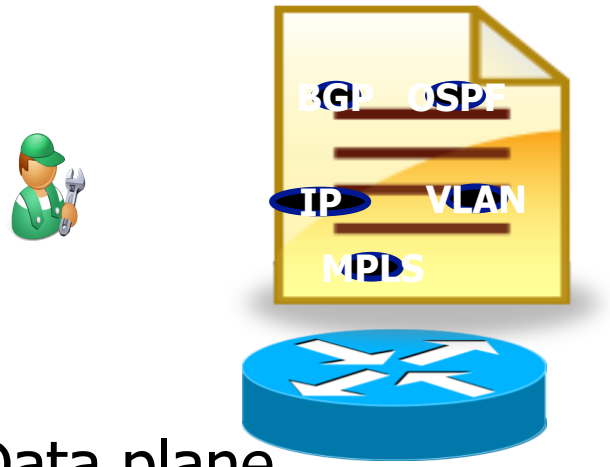
- Application classification
  - Place video traffic in the gold queue



Network Management: Past



Networking Yesterday



Prefix	Port
10.10.2.10	1
2.3.4.23	3

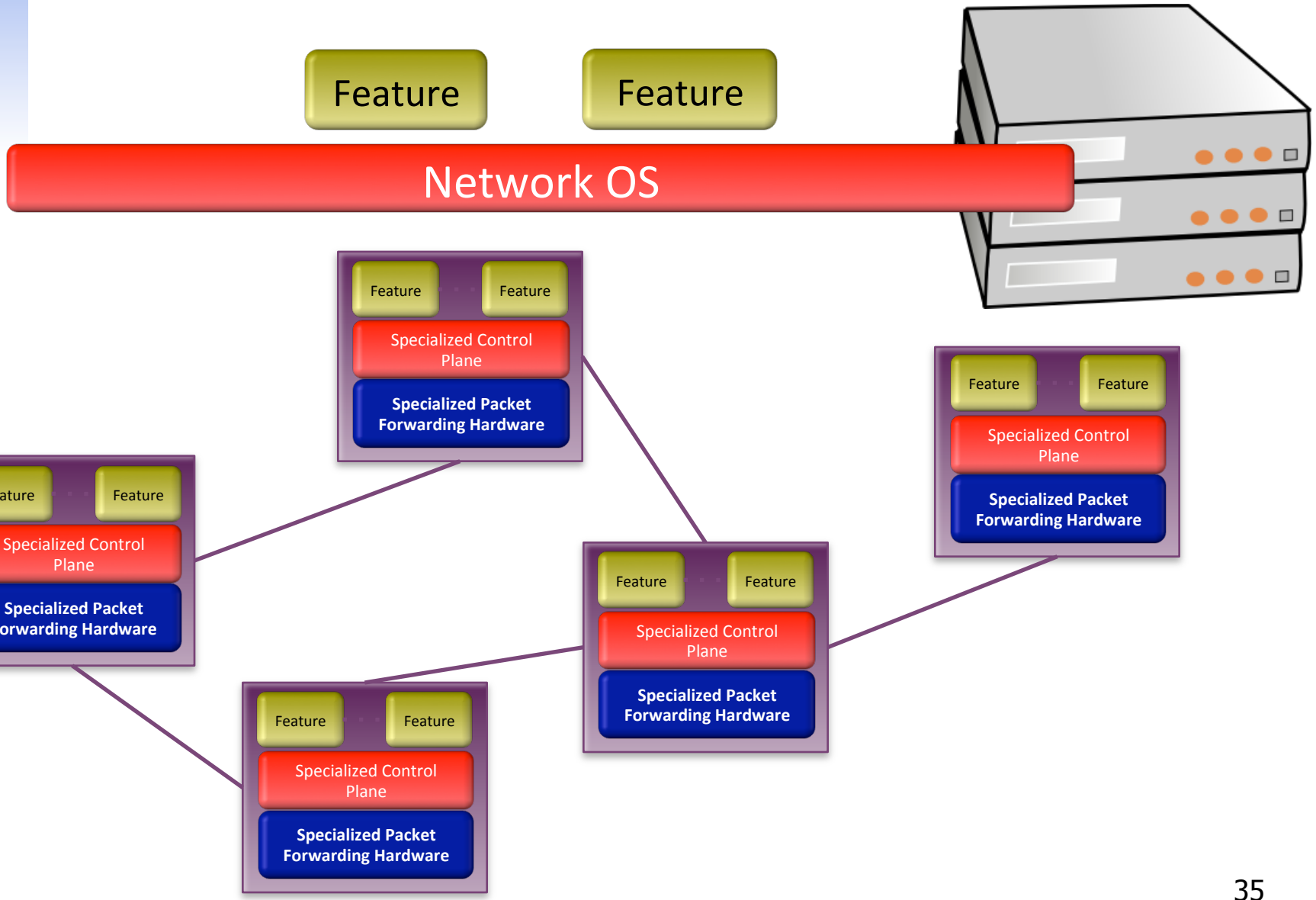
MPLS Label	Path
23	1
45	3

- Data plane
  - Determines how to forward a packet
  - Looks up the forwarding table to determine output port for a packet
- Control plane
  - Determines how to populate the forwarding tables
  - Translate user commands into hardware
    - ACLs, MPLS
  - Runs a bunch of routing protocols
    - IGPs: OSPF, IS-IS, RIP, & EGPs: BGP

Networking Yesterday

- Control + Data-plane on each device
  - Network is a Distributed systems
  - Built to avoid failure (ArpaNet)
- A network is supported by an infinite number of protocols
  - New protocols developed to support new functionality
  - Takes time to standardize and to change the hardware
- Think: writing a distributed program in Perl
  - Error prone (Over 50% of errors caused by misconfig)
  - Time consuming
    - IT Operators are the most costly portion of IT
    - Takes up to 6 months for ISPs to roll out services for new costumers
    - Very difficult to add new functionality into the network

Separation of control and data planes



Requirements for SDN

Network Operating System

- Operate without Guarantees
  - Need abstraction for distributed state
  - Want to deal with information without worrying about the fact that the state is from a distribution
  - Logically Centralized

Network Operating System

- Compute configuration of each device
  - Need abstraction that simplifies configuration
  - Want to specify your intent → desired goal; the what
  - NOT: how to do it.

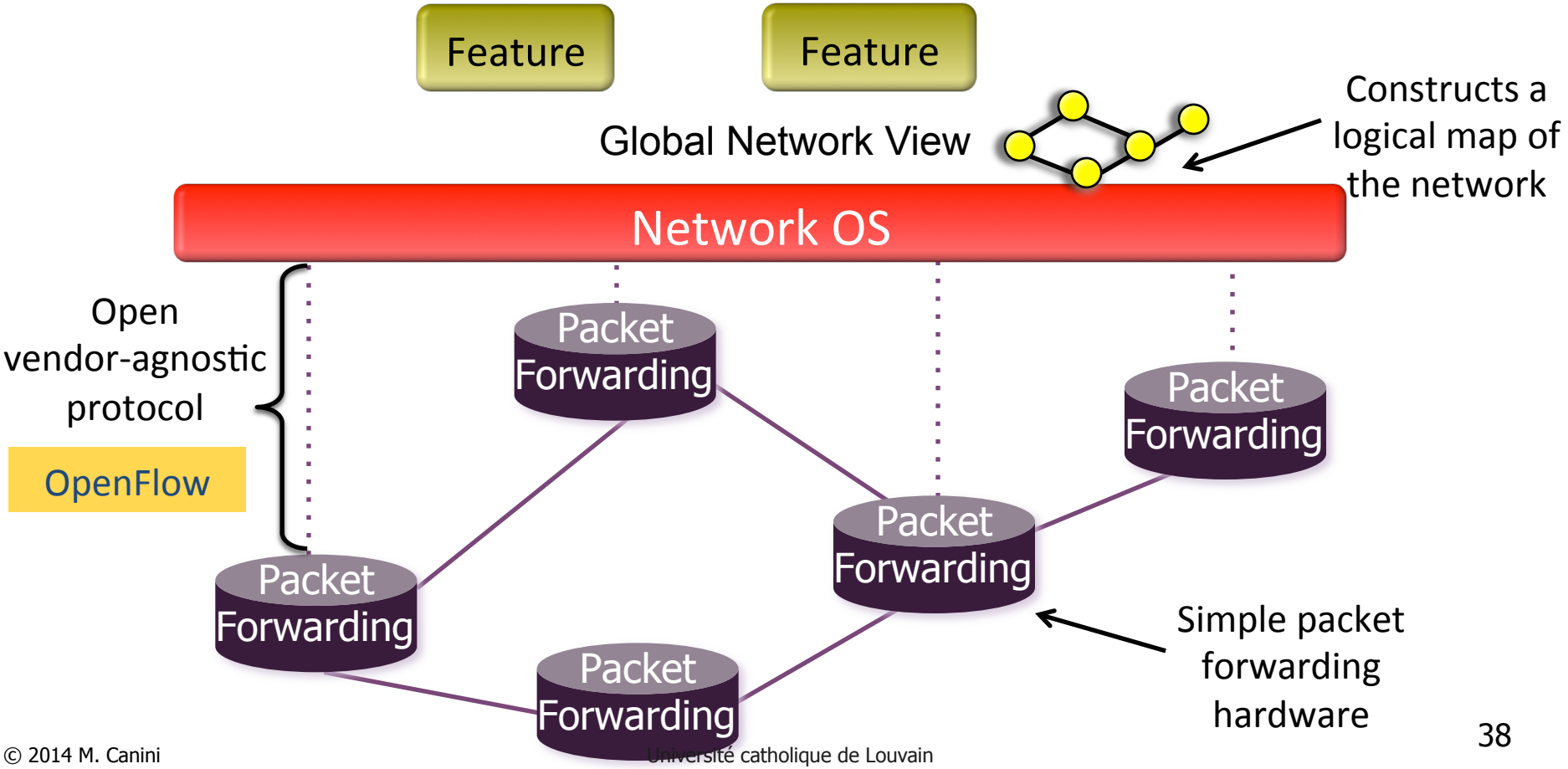
OpenFlow Protocol

- Operate within given network-level protocol
  - Need abstraction for forwarding model
  - Hide details about hardware specifics
  - No need to worry about he exact hardware

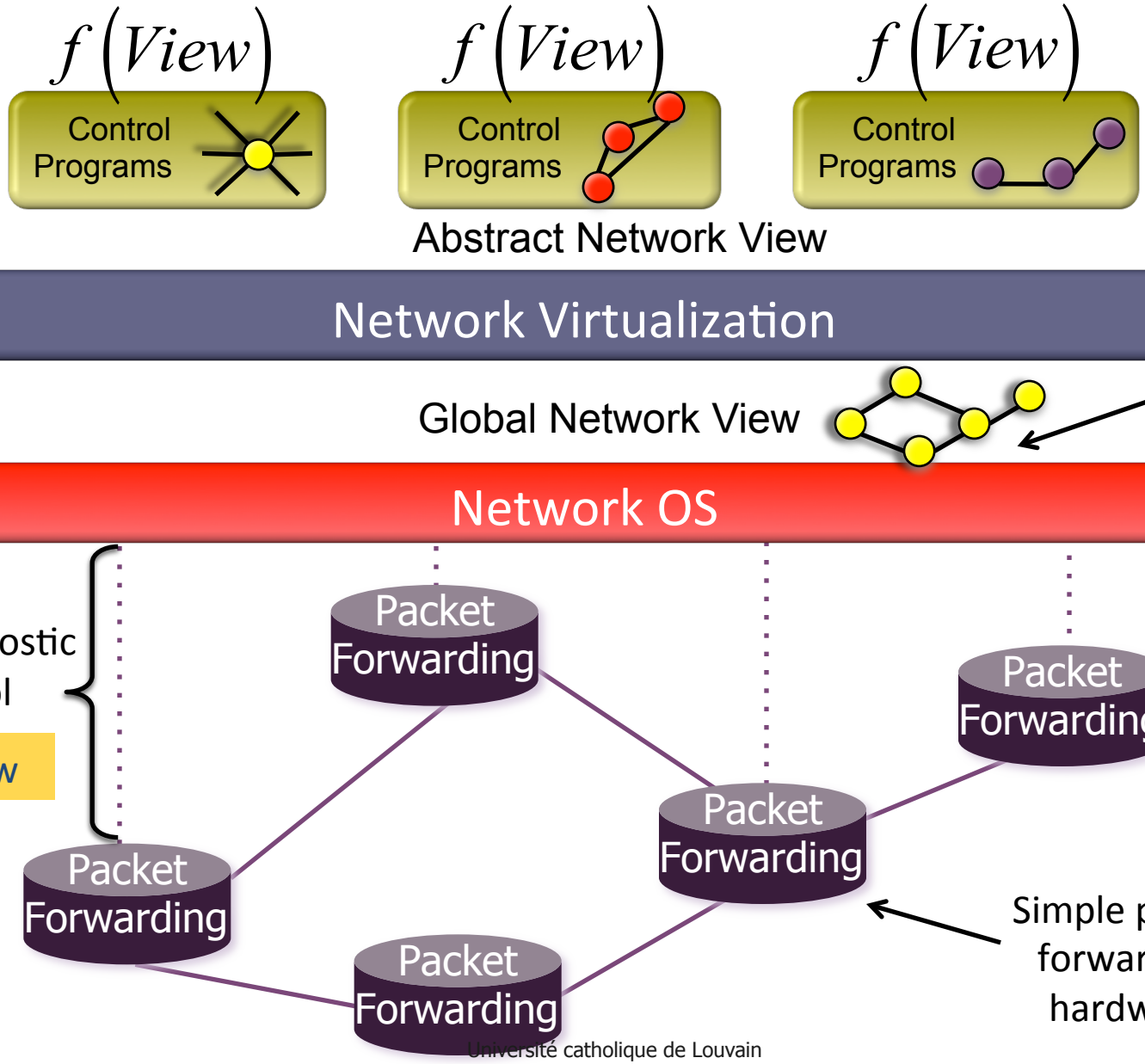
Enter Software Defined Networking: Separation of concerns

- Network operator
  - Specify behavior on a model
  - Behavior == network policies
- Network runtime
  - Provides abstract view of the network
  - Maps abstract view to global view
  - Function of the types of network policies to be supported
- Network Operating System
  - Maps global view to physical view
  - Translate abstract commands to device configuration
  - Device interface: forwarding abstractions

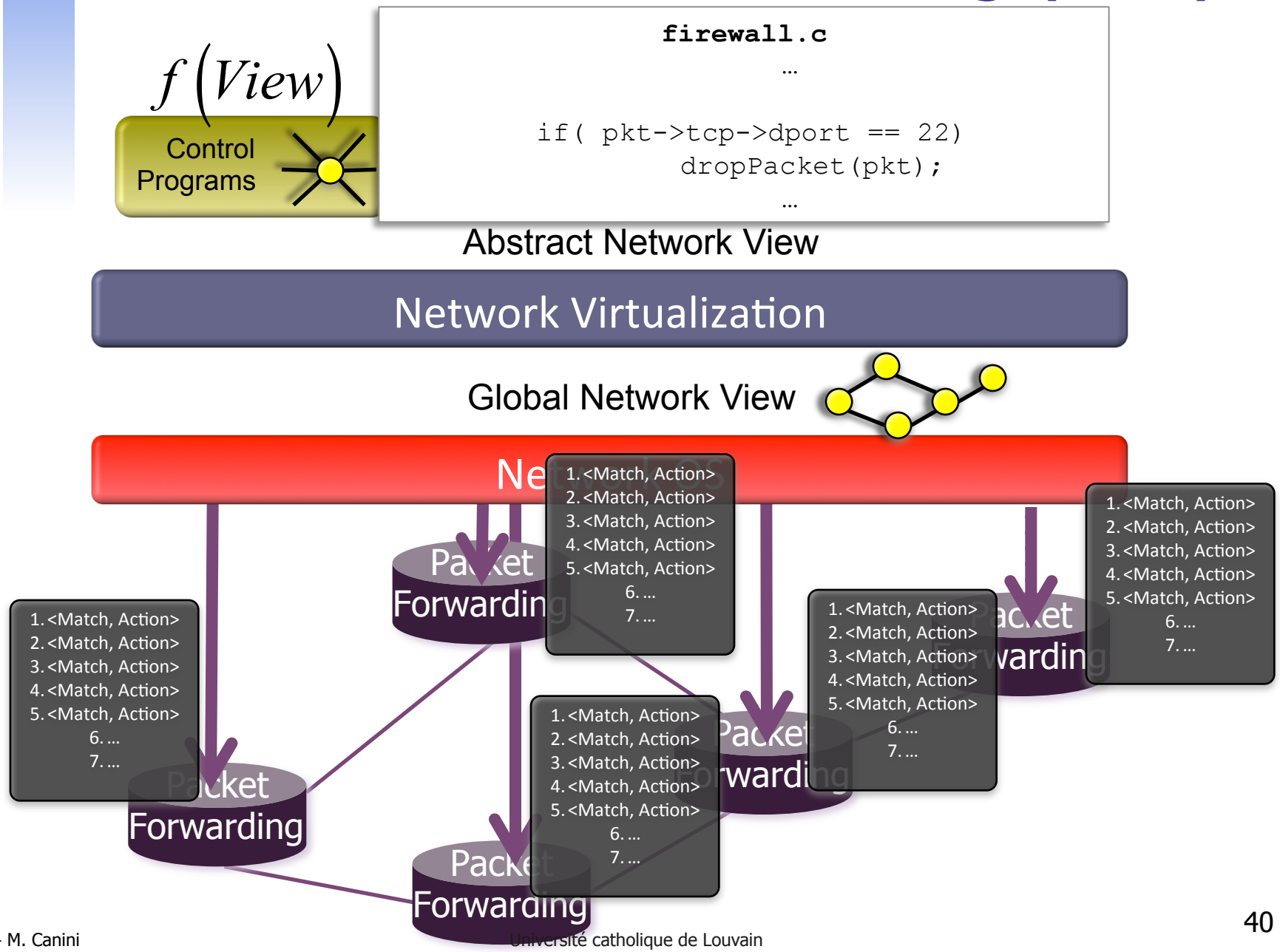
Software Defined Networking (SDN)



Software Defined Networking (SDN)



Software Defined Networking (SDN)



What problem does SDN solve?

- Great tool to enable innovation in network control
- Platform to help solve longstanding problems in managing networks and deploying new functionality

SDN applications

- Network management
  - Enterprise access, middleboxes waypointing
- Monitoring and measurement
- Network virtualization
- User mobility and VM migration
- Server load balancing
- Traffic engineering
  - Energy efficiency, high WAN utilization
- Exposing an API to host applications

OpenFlow

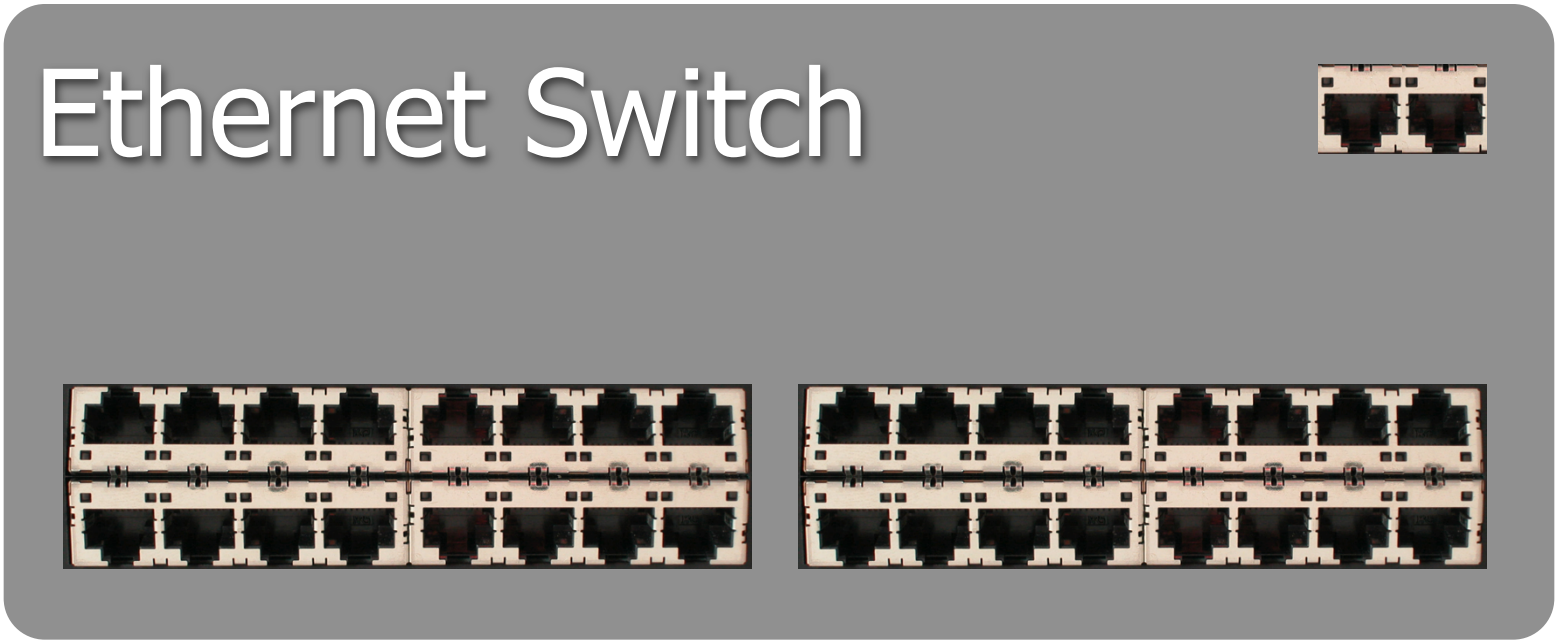
- is a protocol for remotely controlling the forwarding table of a switch or router
- is one element of SDN

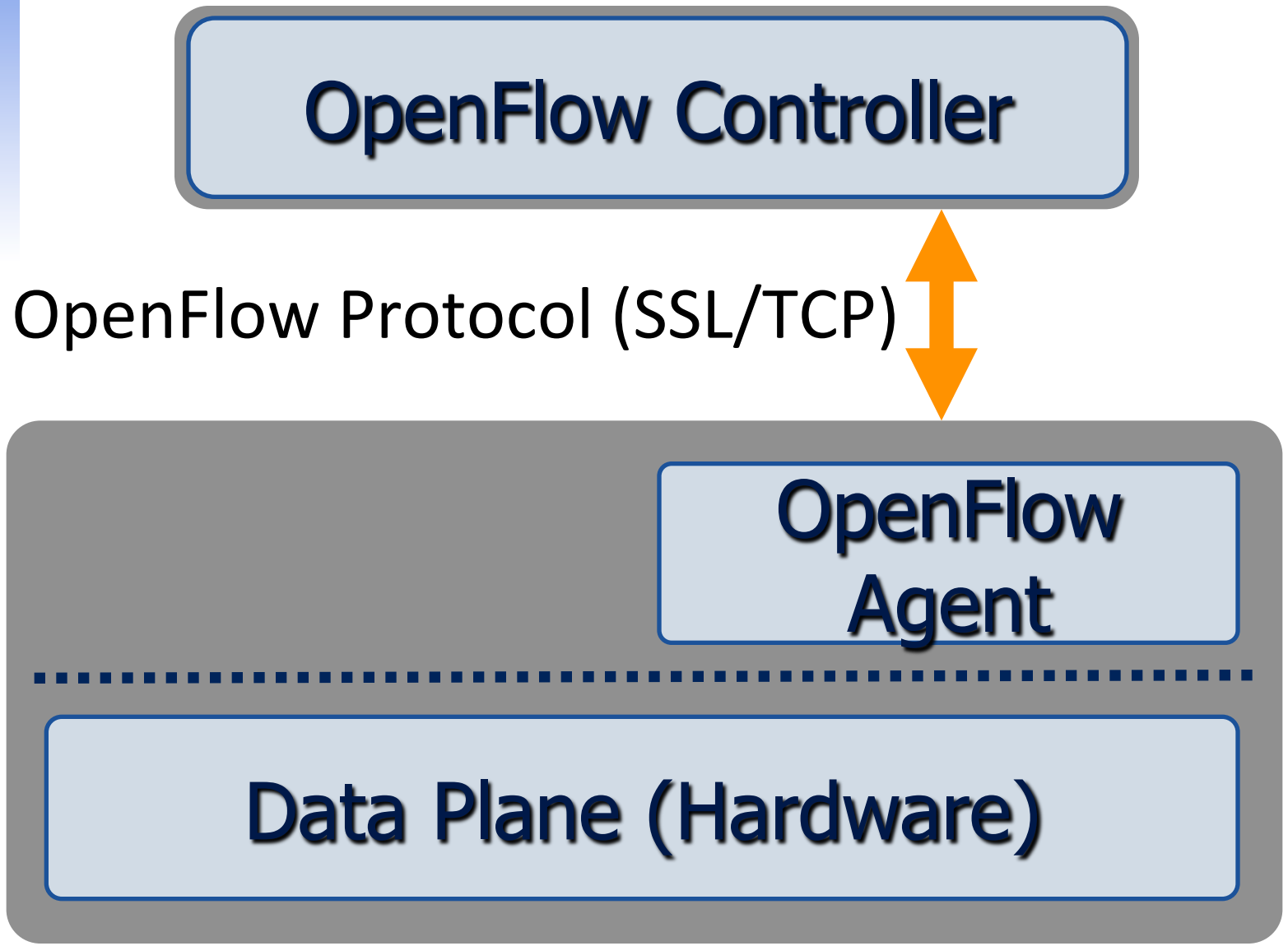
OpenFlow Protocol

- Message between controller and switches
  - Synchronous
    - Stats, Flow-mods
  - Asynchronous
    - Packet-in
- Abstract hardware details
- Allows direct control over forwarding table

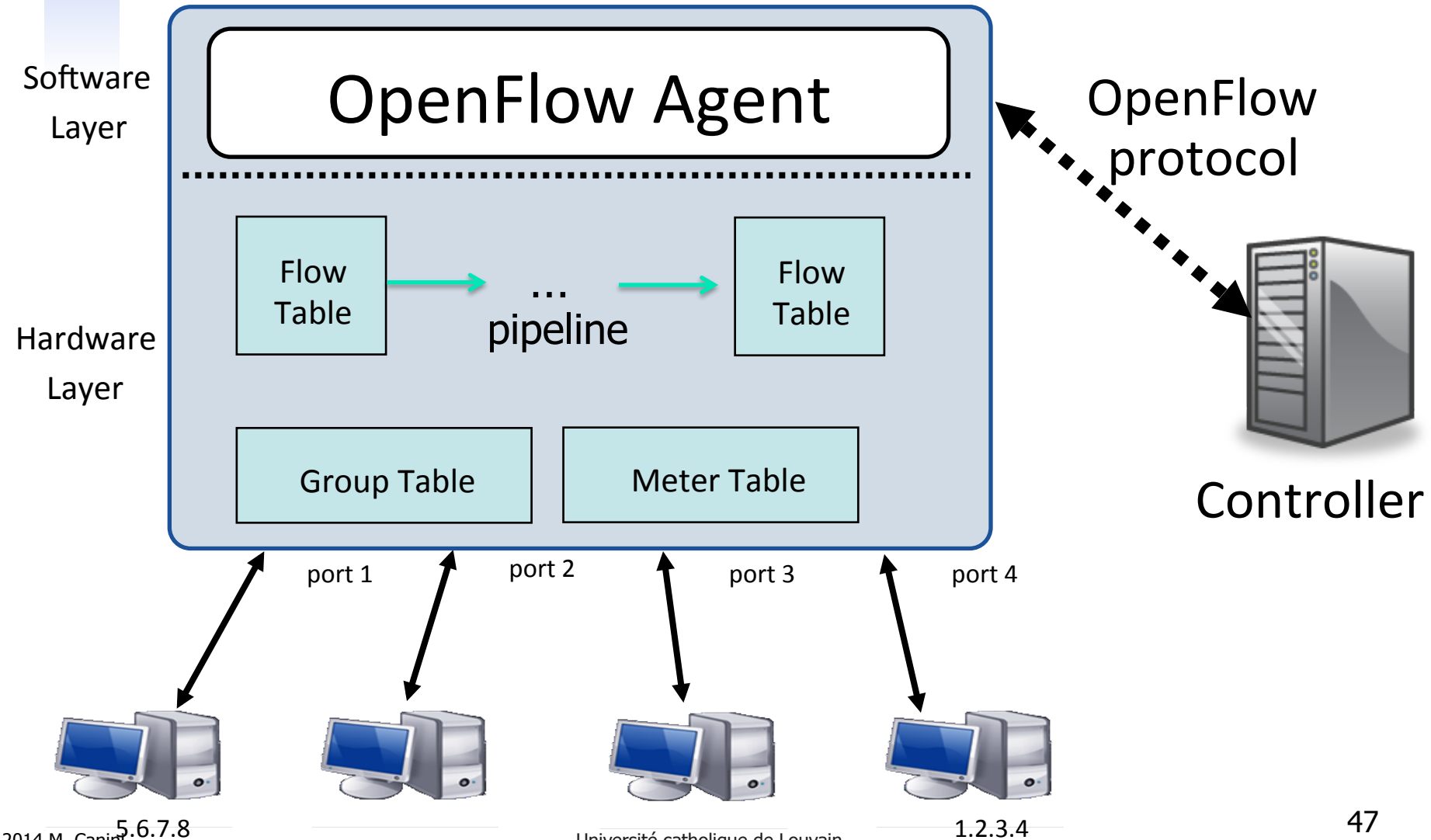
Match	Action
10.2.3.4:10.2.3.3	Fwd Port 1
A2:e3:f1:ba:ea:23:*	Drop

How does OpenFlow work?




OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
OpenFlow Agent
Data Plane (Hardware)

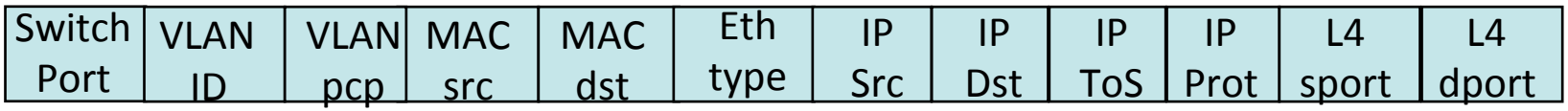
Main components of an OpenFlow switch



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.



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

Match/action examples

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

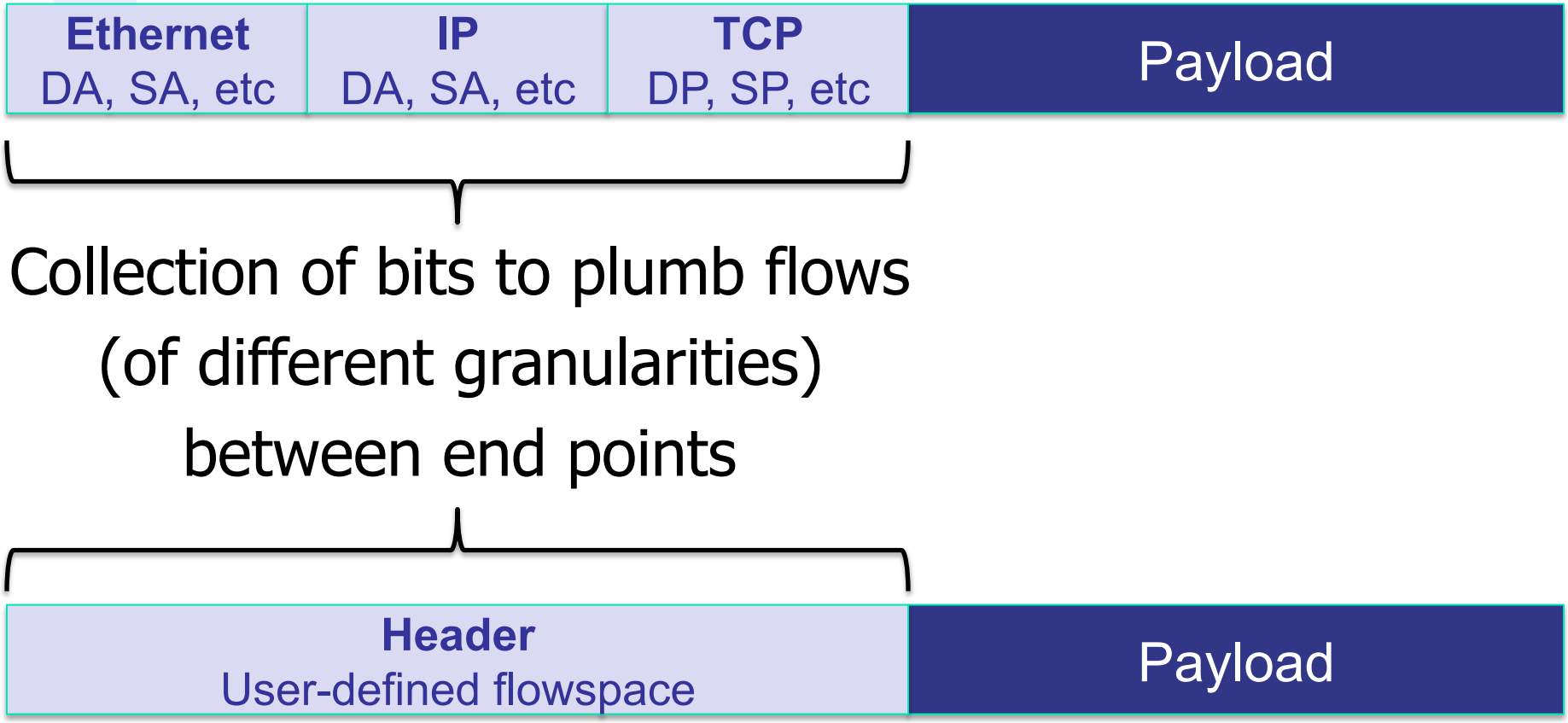
Routing

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

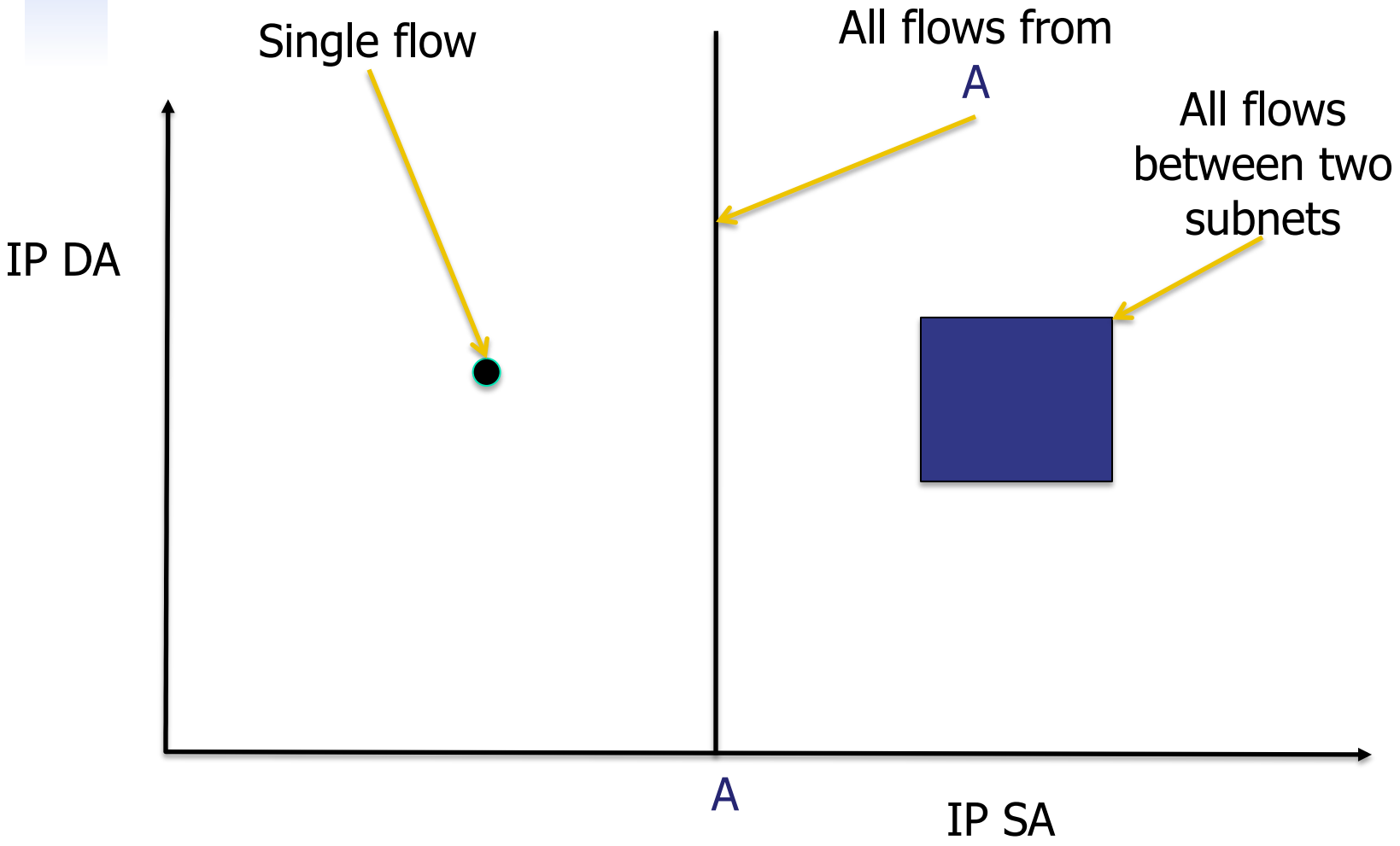
VLAN Switching

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

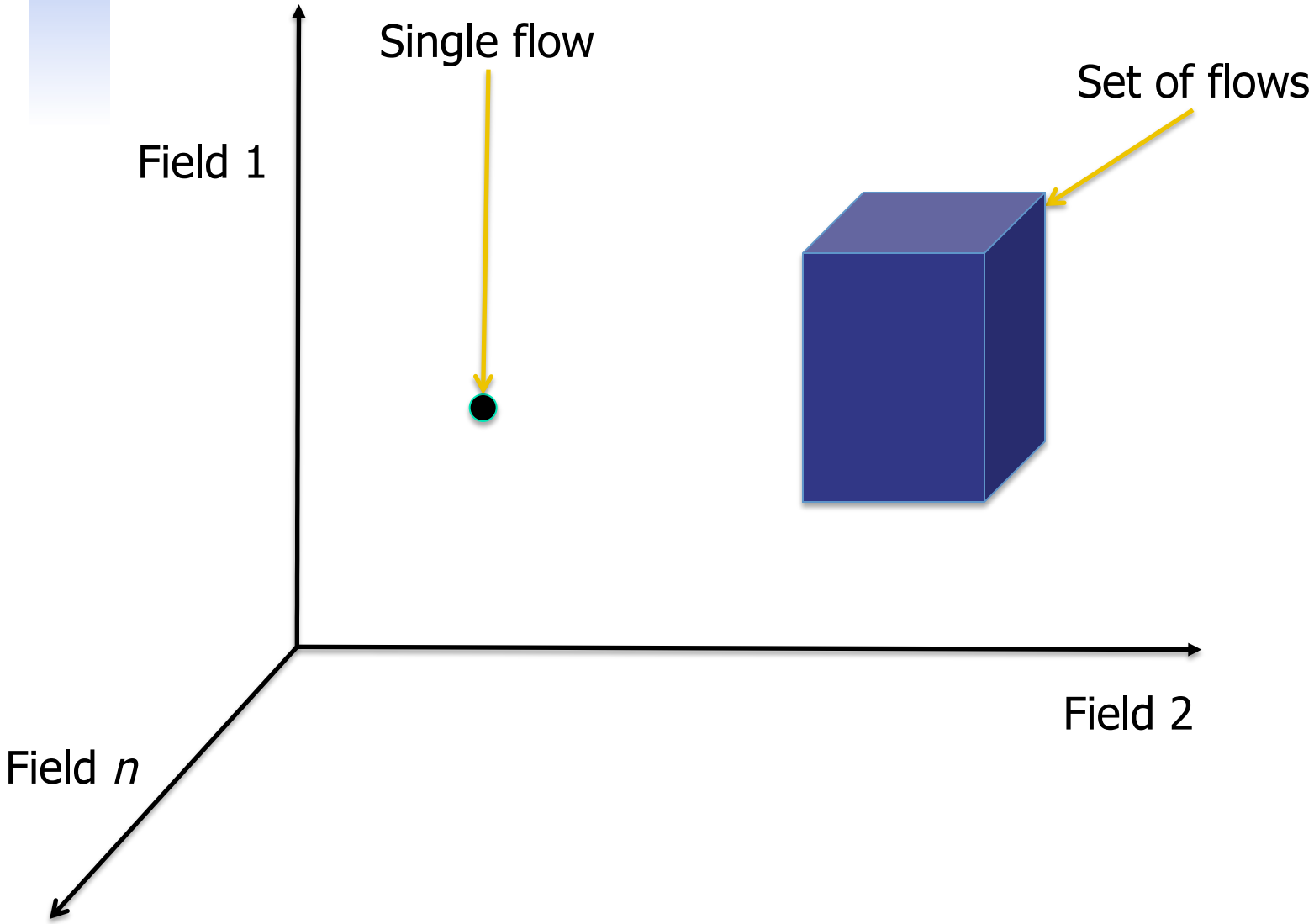
Headers as a protocol-agnostic collection of bits



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



Flowspace: Generalization



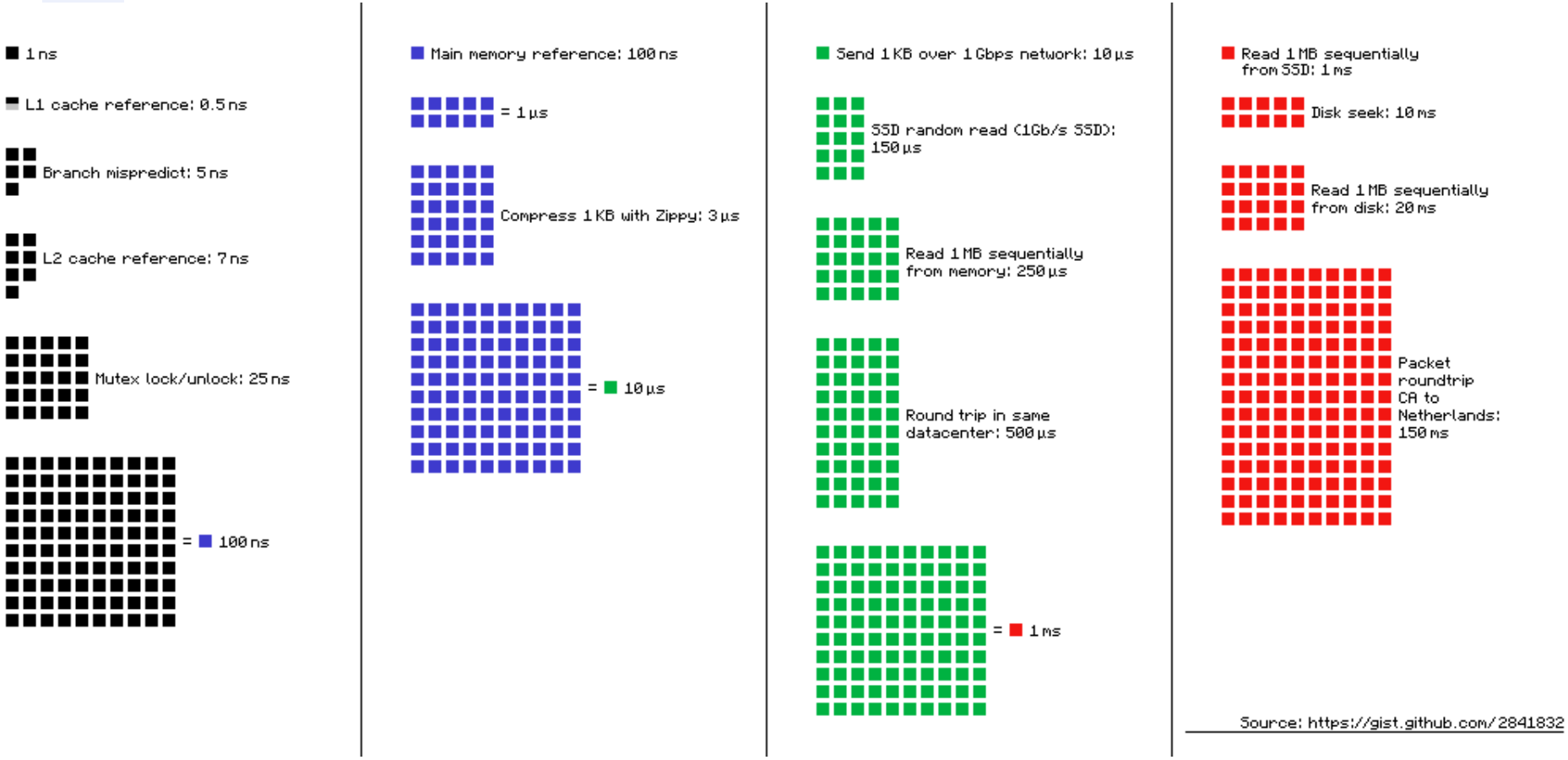
OpenFlow key messages

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

Further readings

- A Guided Tour Through Datacenter Networking
  Abts, D., Felderman, B. Communications of the ACM(CACM), 55(6), 2012
  http://queue.acm.org/detail.cfm?id=2208919
- The next-generation Facebook data center network
  http://bit.ly/1xp8fbi
- A scalable, commodity data center network architecture
  Al-Fares, M., Loukissas, A., & Vahdat, A. SIGCOMM 2008
- OpenFlow: Enabling Innovation in Campus Networks
  McKeown, N., Anderson, … Turner, J. ACM SIGCOMM Computer Communication Review, 38(2), 2008
- Software-Defined Networking at the Crossroads
  S. Shenker
  https://www.youtube.com/watch?v=WabdXYzCAOU

Numbers Everyone Should Know



Source: http://i.imgur.com/k0t1e.png

Numbers by Jeff Dean (Google)

That's all

- Hope you enjoyed INGI2145!
- Exam: 17 January 8:30 – BARB 91