An Introduction to Software-Defined Networking (SDN)

Zhang Fu

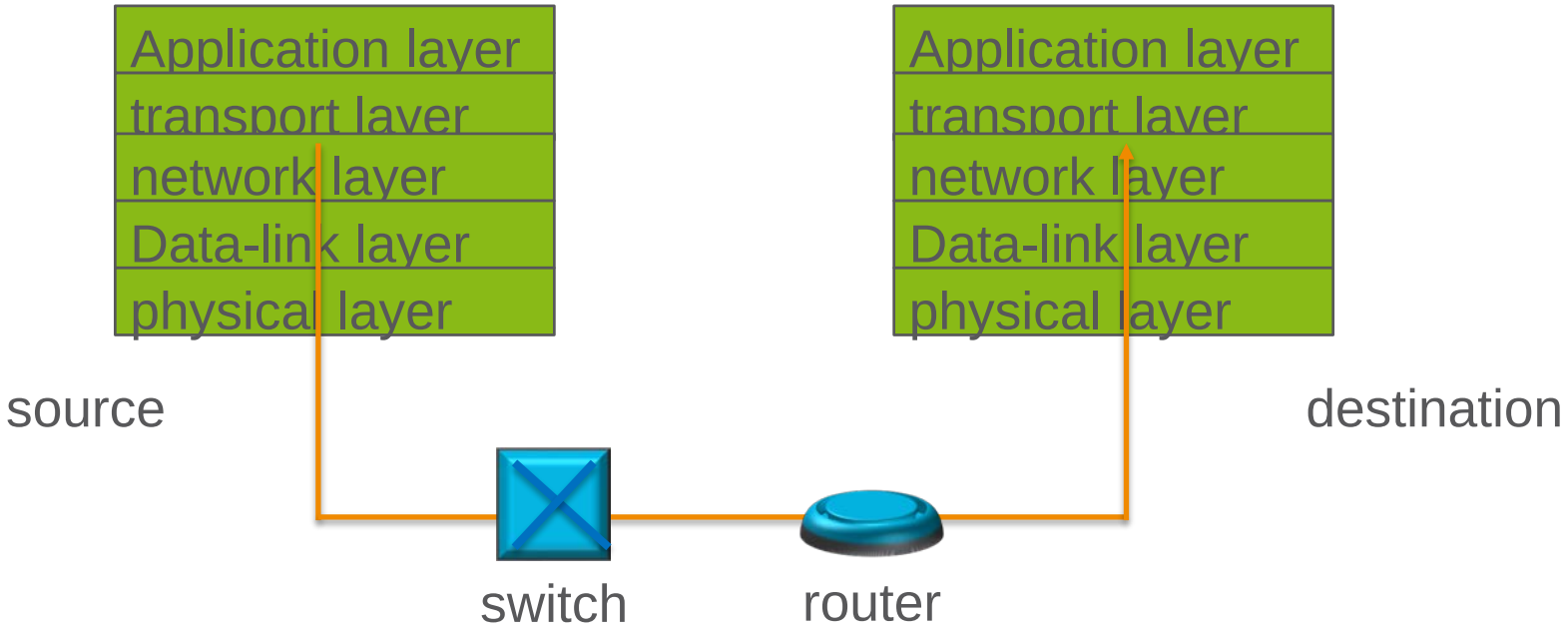
Ericsson Research

Roadmap

- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little touch on Openflow

Reviewing traditional networking

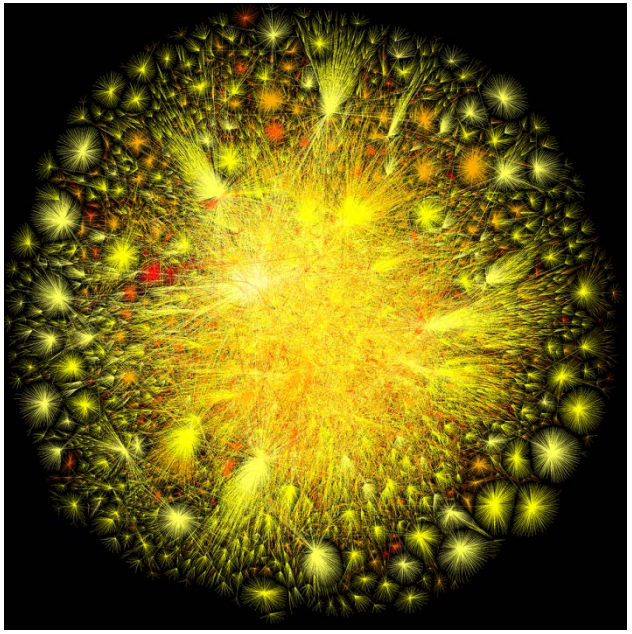
› Network layers



Why layers? Good abstraction, transparency…

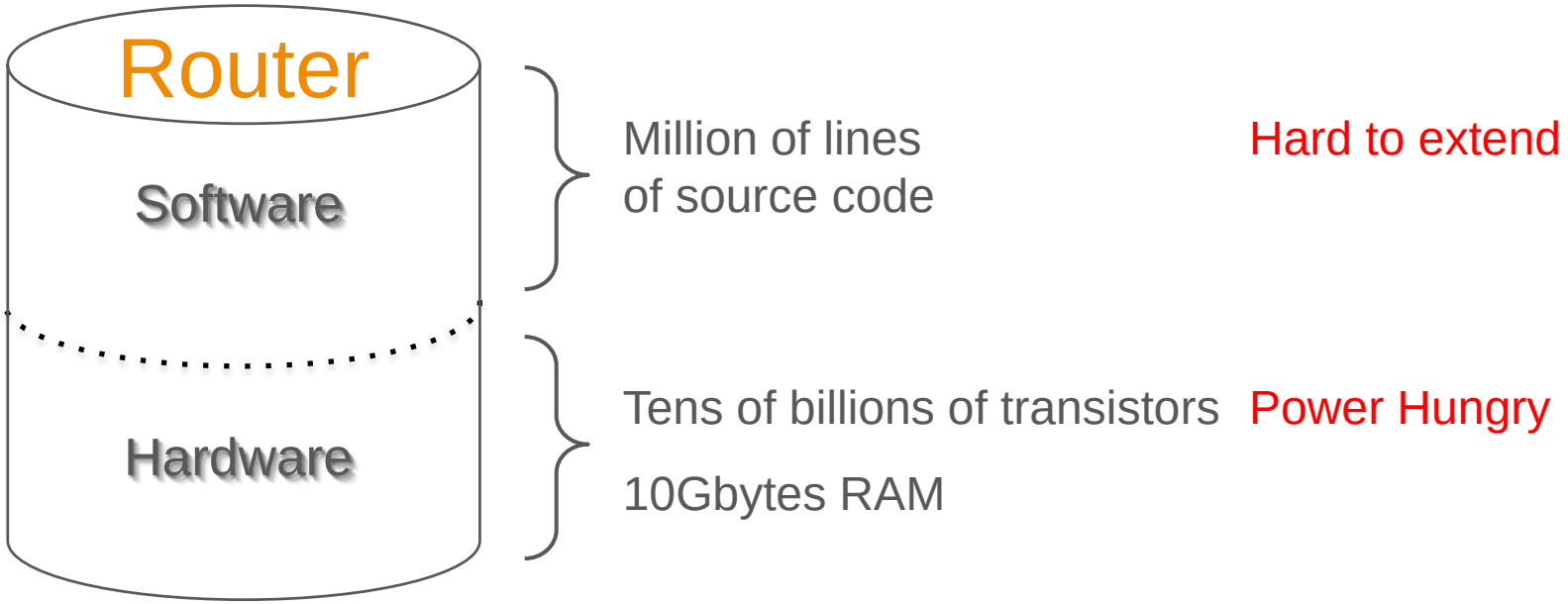
Reviewing traditional networking

- Design principles of Internet
  - Simple
  - Intelligent end-points
  - Distributed control
- Resulting in huge complex network and hard to manage
  - Billions of computers
  - Tens of thousands of ASes
  - Great business for selling routers



Reviewing traditional networking

› Complex routers



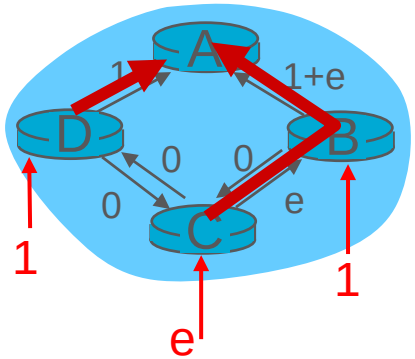
Vertically integration with many complex functions: *OSPF, BGP, multicast, QoS, Traffic Engineering, NAT, firewalls, MPLS…*

Roadmap

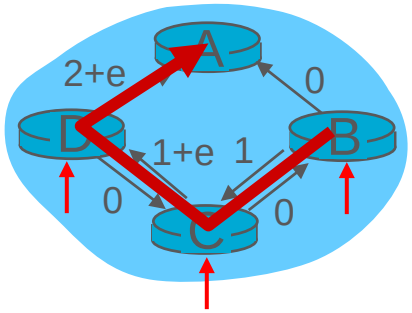
- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little bite of Openflow

Example: oscillation problem

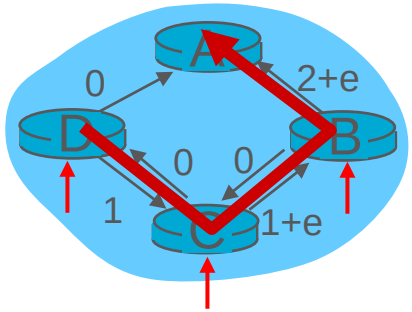
- Link cost equals the amount of carried traffic



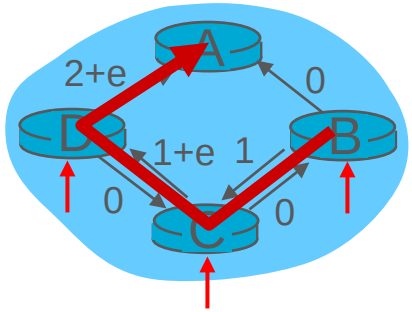
initially



given these costs,
find new routing….
resulting in new costs



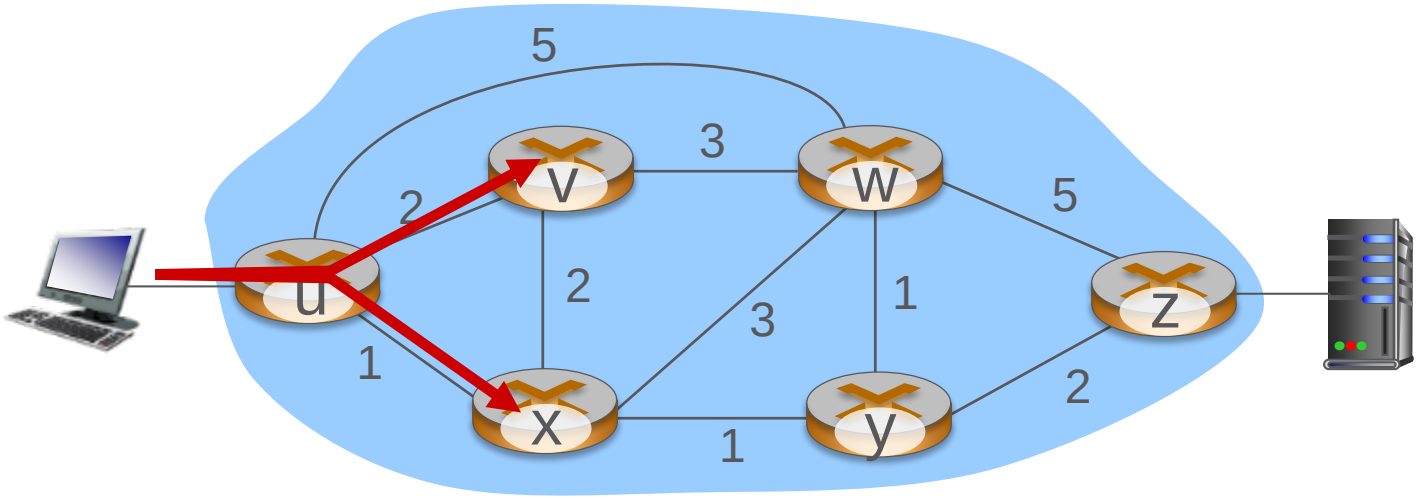
given these costs,
find new routing….
resulting in new costs



given these costs,
find new routing….
resulting in new costs

How to achieve optimal routing dynamically?

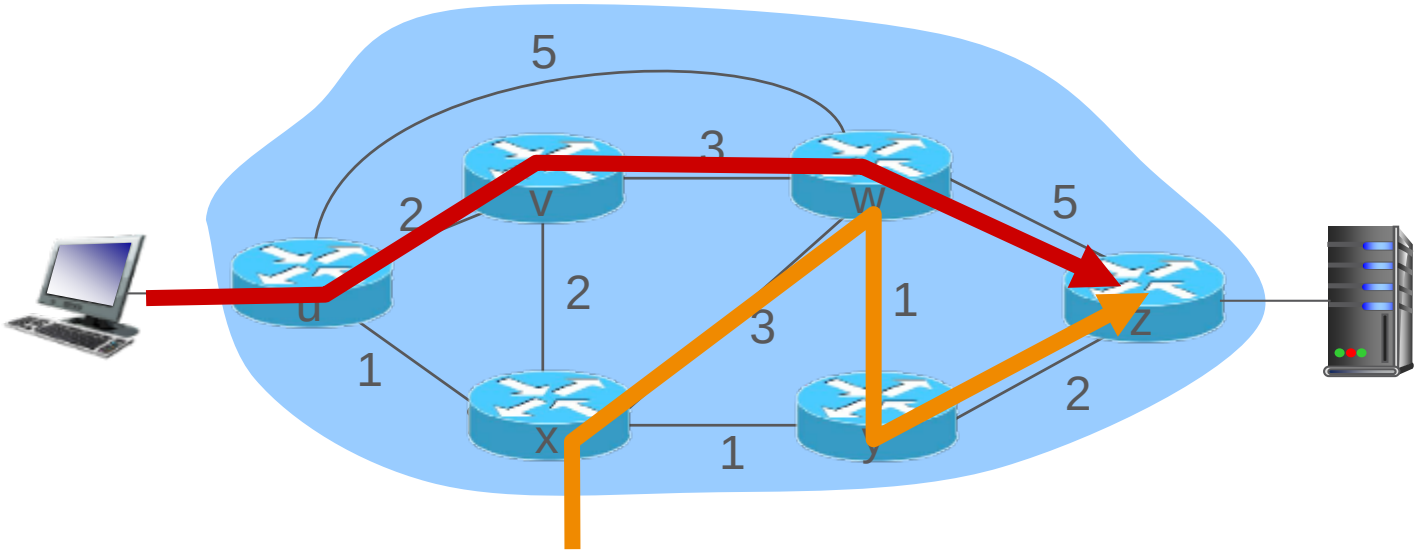
Traffic engineering: difficult



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



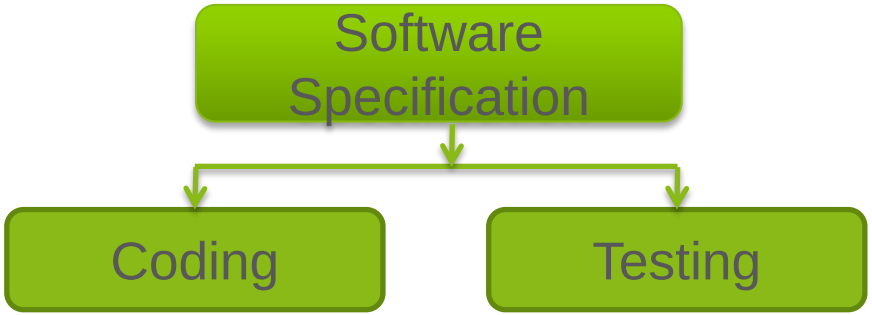
Q: what if w wants to route yellow and red traffic differently?

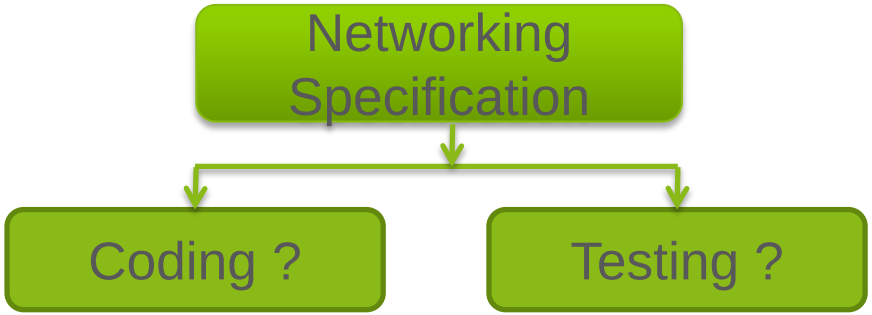
A: can't do it (with destination based forwarding, and LS, DV routing)

Roadmap

- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little touch on Openflow

Software development VS Network diagnosing





Various tools for code analysis, verification, debugging

Diagnosing tools? Ping, traceroute, SNMP

- The life cycle for network protocols is much longer than that for software
- Timely research does not find its way into practice

Network substrate

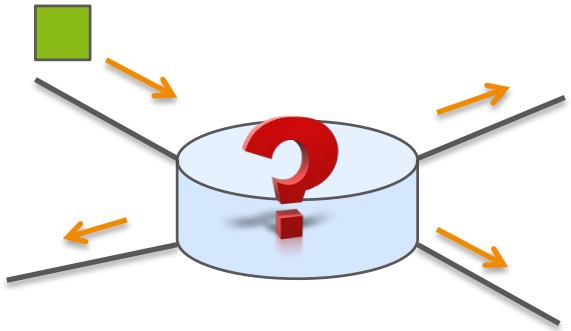
› We want to mimic the success in software industry
  – Has simple common substrate
  – Building OS on top the hardware, which enables easy deployment of networking applications

SDN

- A network in which the control plane is physically separate from the data plane.
- A single control plane controls several forwarding devices.

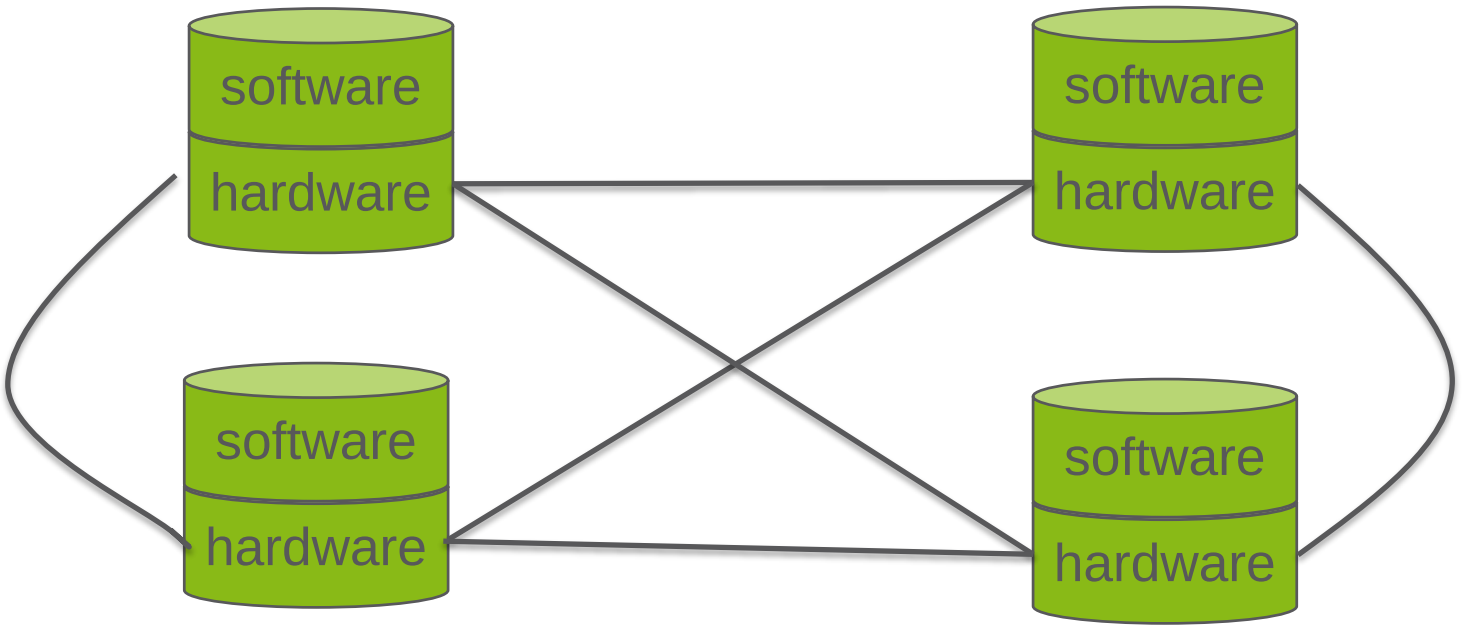
Network substrate

› Router Example

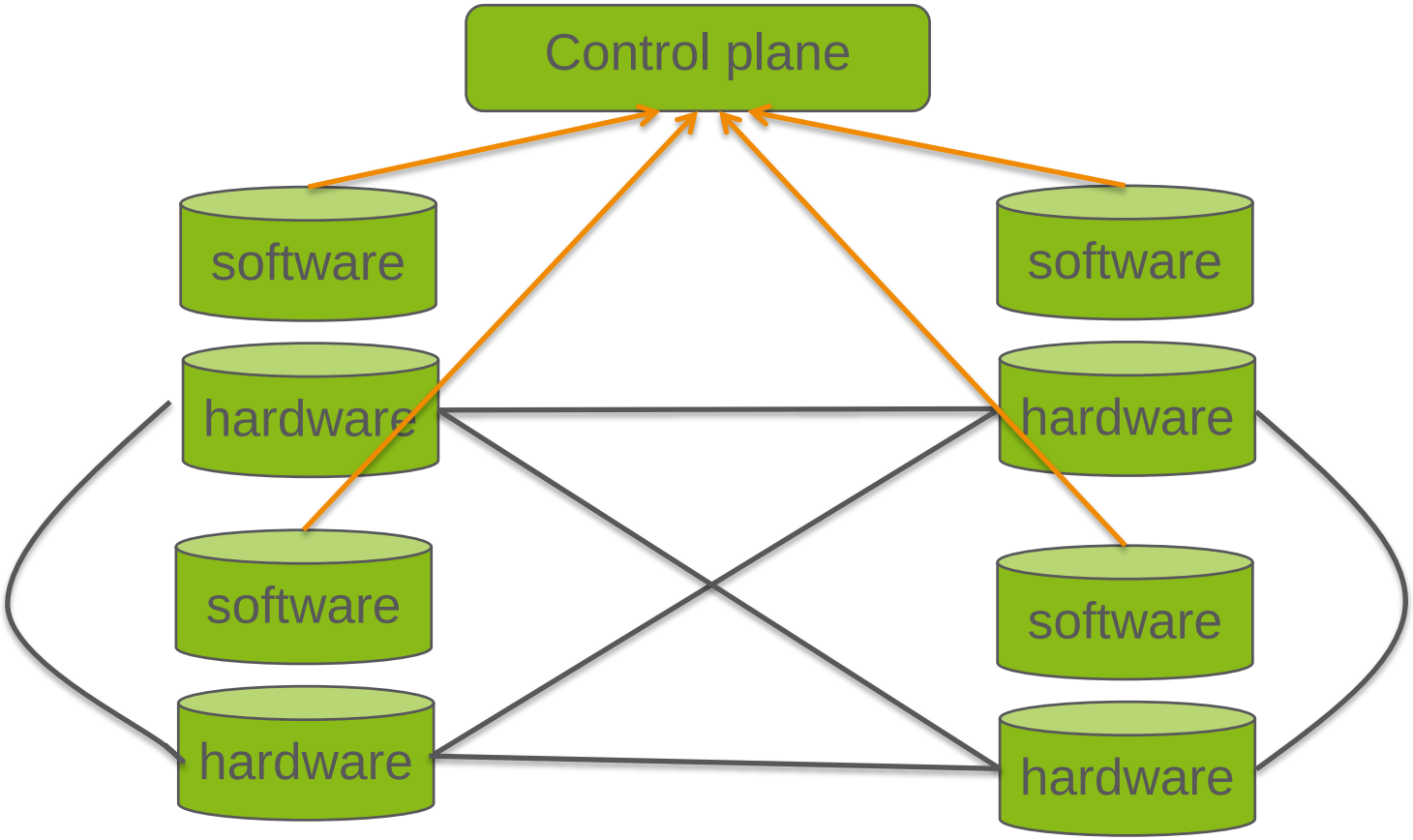


– Basic job of the router: *receiving packets, checking the routing table, forwarding the packets out*
– In order to build the routing table, the router has to understand BGP, OSPF, RIP, etc.
– What about getting the routing table from somewhere else?

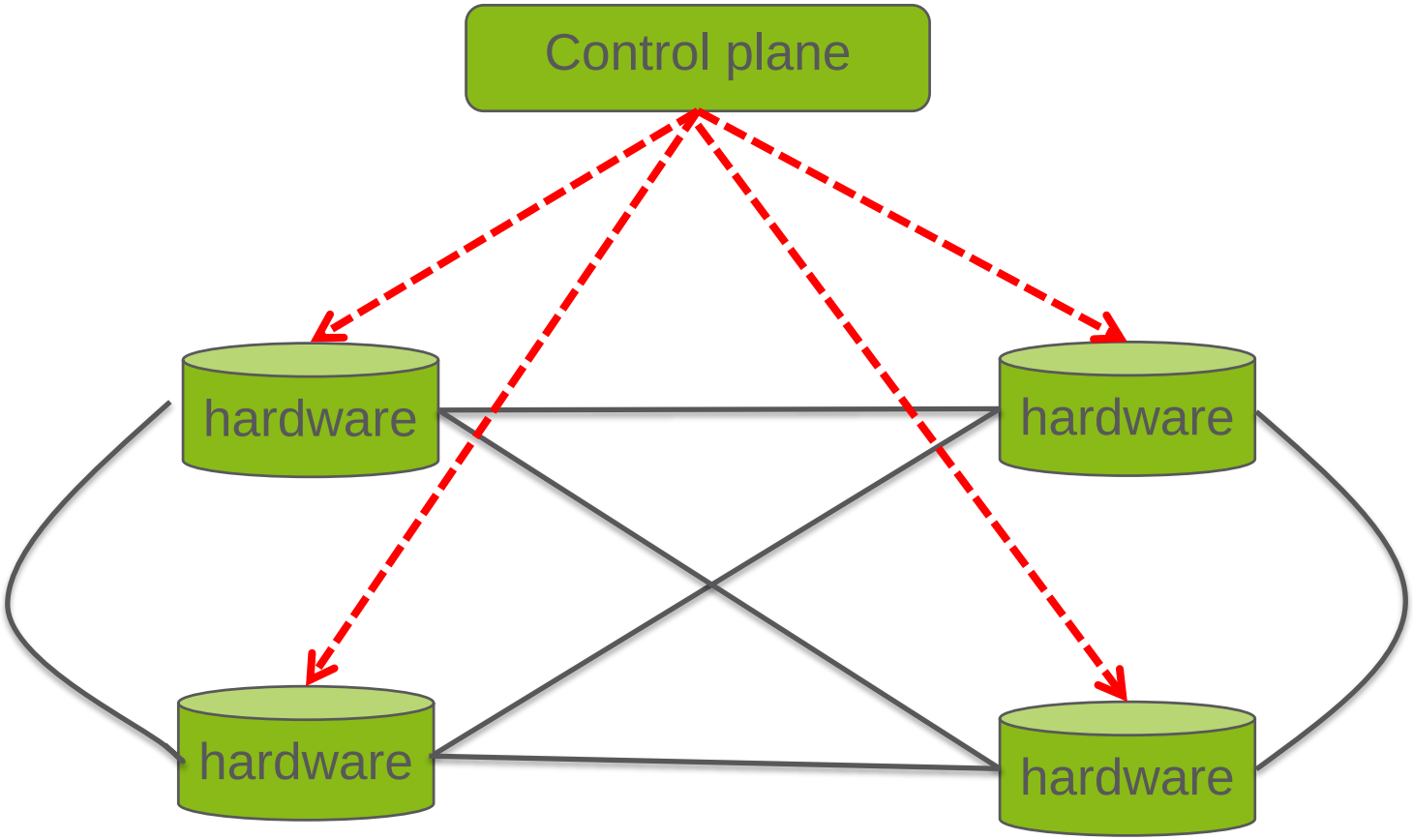
Separate data and control plane



Separate data and control plane



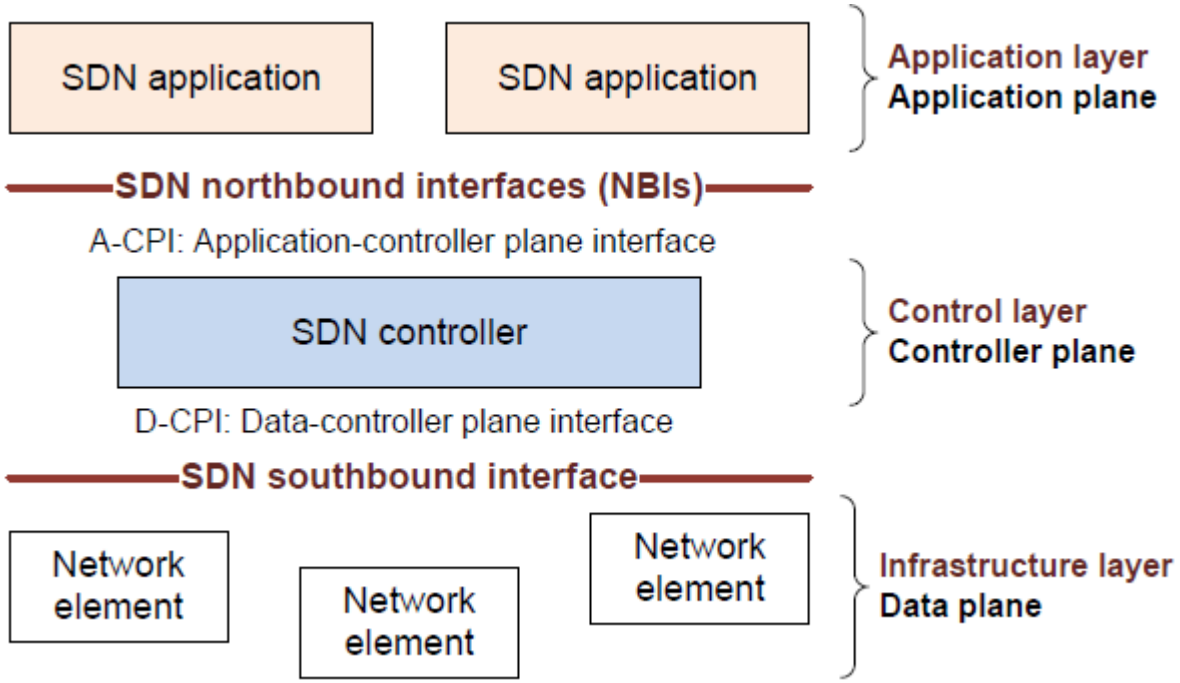
Separate data and control plane



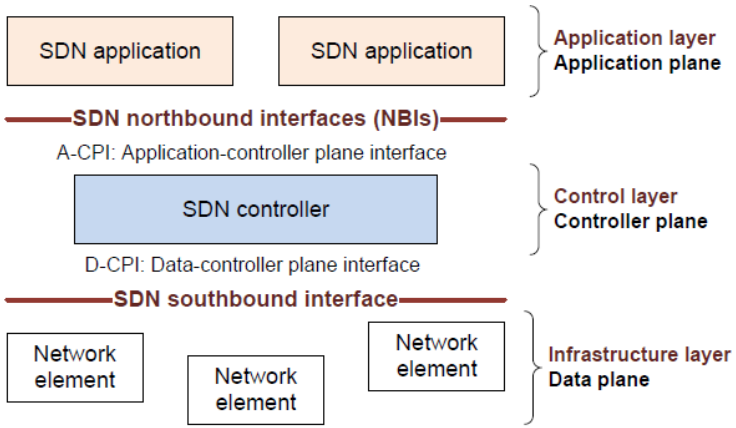
Roadmap

- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little touch on Openflow

SDN architecture



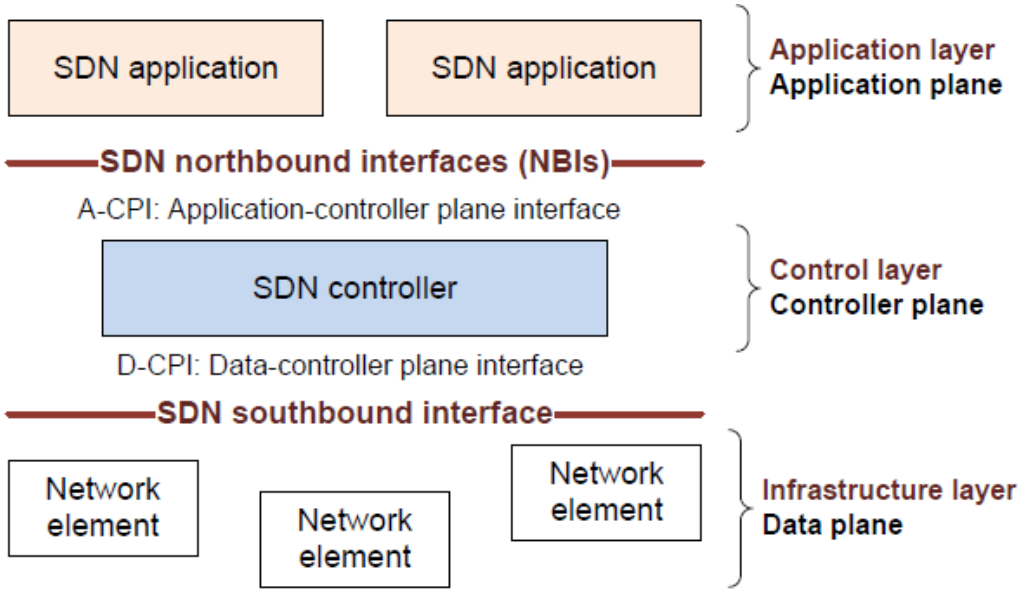
SDN architecture



- The data plane consists of network elements, which expose their capabilities to the control plane via southbound interface
- The SDN applications are in the application plane and communicate their network requirements toward the control plane via northbound interface
- The control plane sits in the middle
  - translate the applications' requirements and exerts low-level control over the network elements
  - Provide network information to the applications
  - Orchestrate different applications

Data-plane

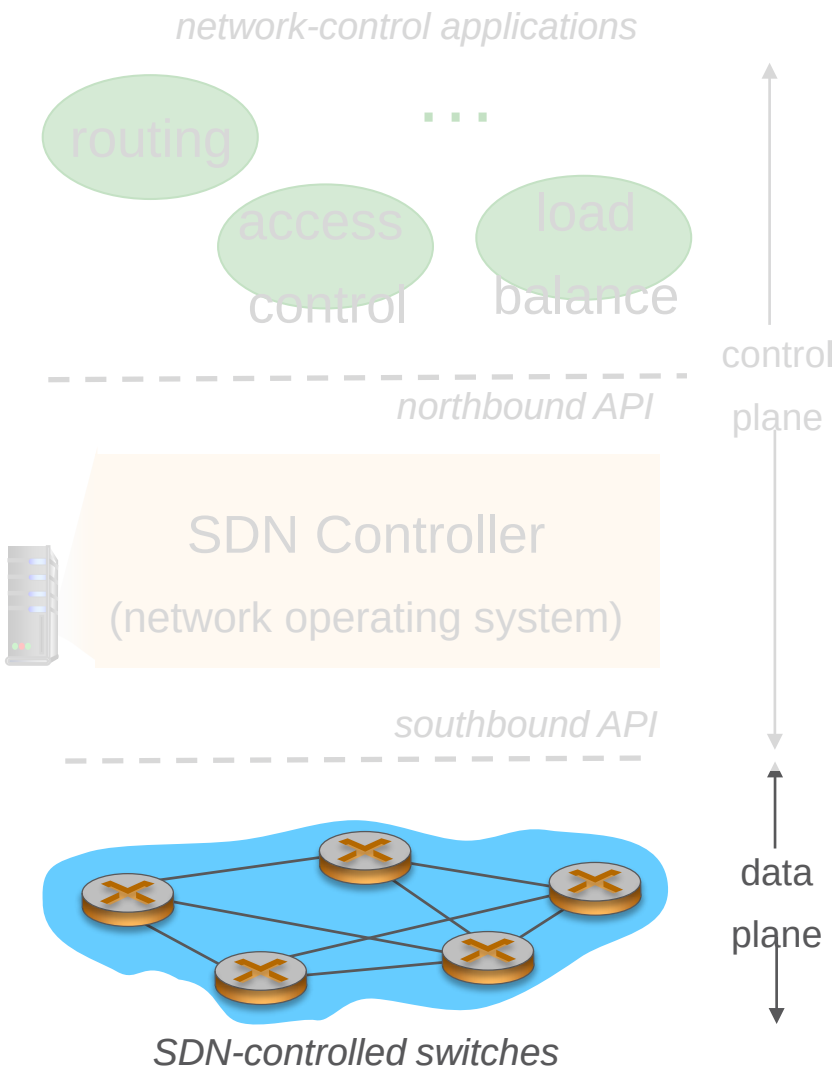
- Data sources and sinks
- Traffic forwarding/processing engine
  - May have the ability to handle some types of protocol, e.g. ARP, LLDP.
- Provide interfaces communicating to the control plane
  - Programmatic control of all functions offered by the network element
  - Capability advertisement
  - Event notification



SDN perspective: data plane switches

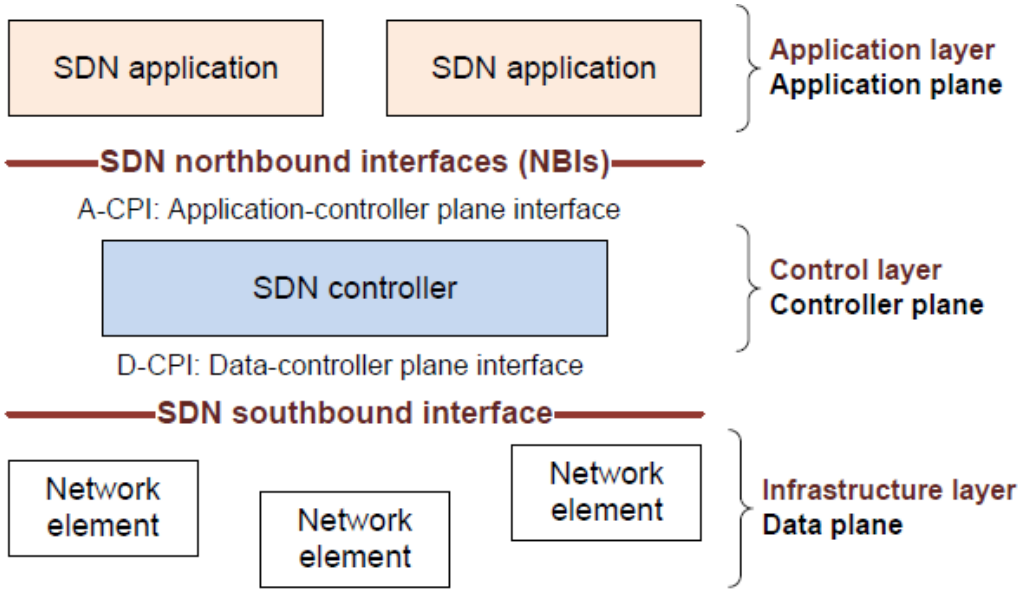
Data plane switches

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



Control-plane

- Logically centralized
- Core functionality
  - Topology and network state information
  - Device discovery
  - Path computation
  - Security mechanism
- Coordination among different controllers
- Interfaces to the application plane

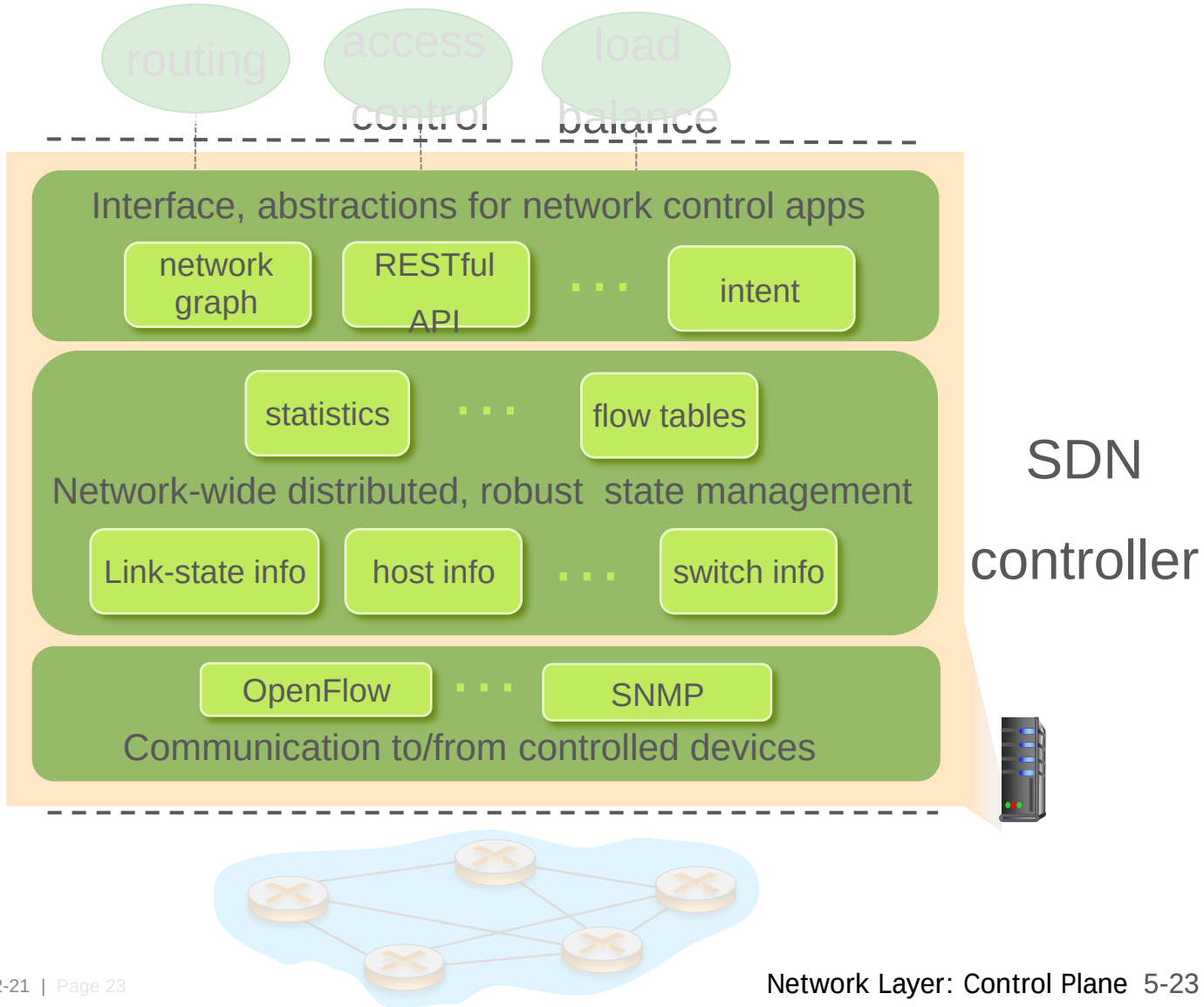


Components of SDN controller

Interface layer to network control apps: abstractions API

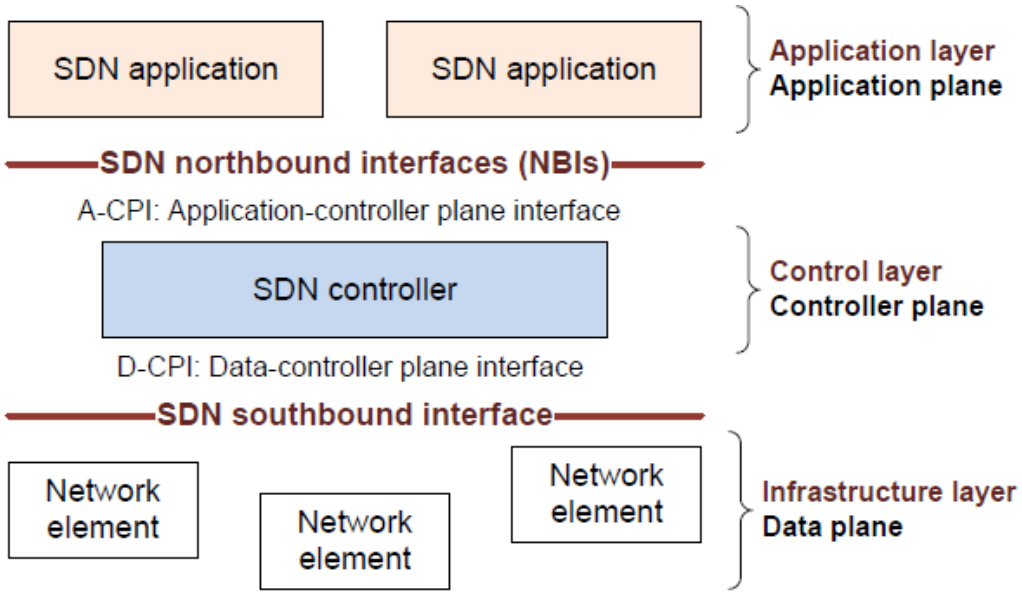
Network-wide state management layer: state of networks links, switches, services: a *distributed database*

communication layer: communicate between SDN controller and controlled switches



Application-plane

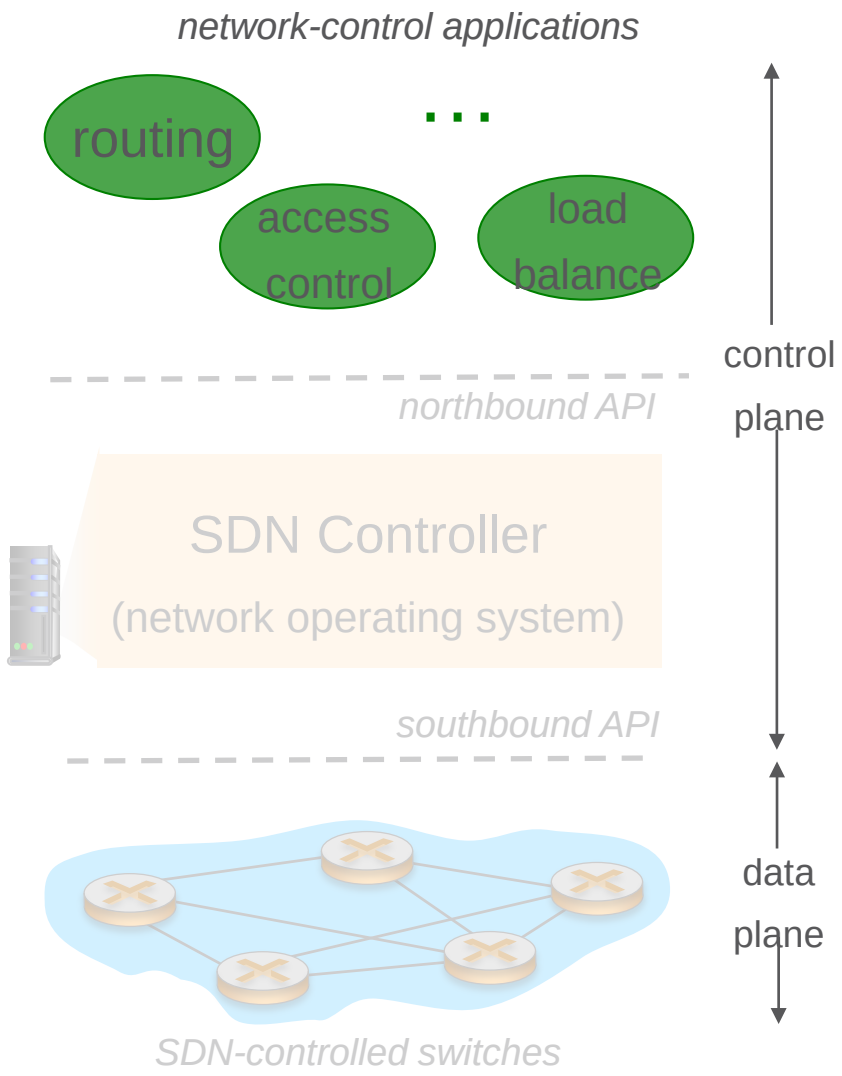
- Applications specify the resources and behaviors required from the network, with the context of business and policy agreement
- It may need to orchestrate multiple-controllers to achieve the objectives, (Cloudify, Unify)
- Programming languages help developing applications.



SDN perspective: control applications

network-control apps:

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



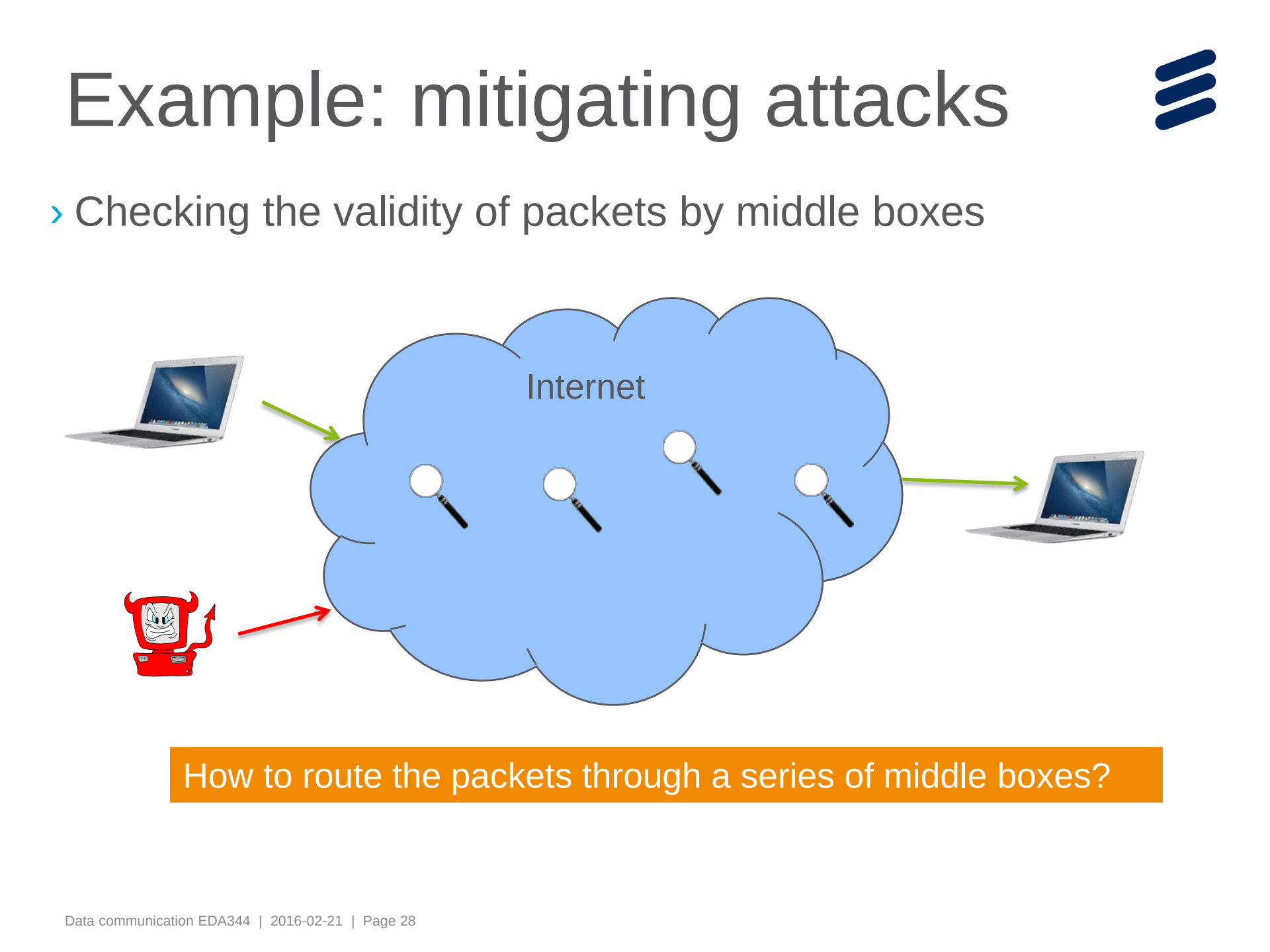
Roadmap

- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little touch on Openflow

Use cases

- Traffic engineering
  - Avoid congestion
  - Adaptive to different policies, QoS
- Mobility and wireless
  - Seamless mobility
  - SDN based Core network
- Security
  - Packets going through a set checking boxes
- Data center networking
  - Enhancing link utilization
  - Saving energy

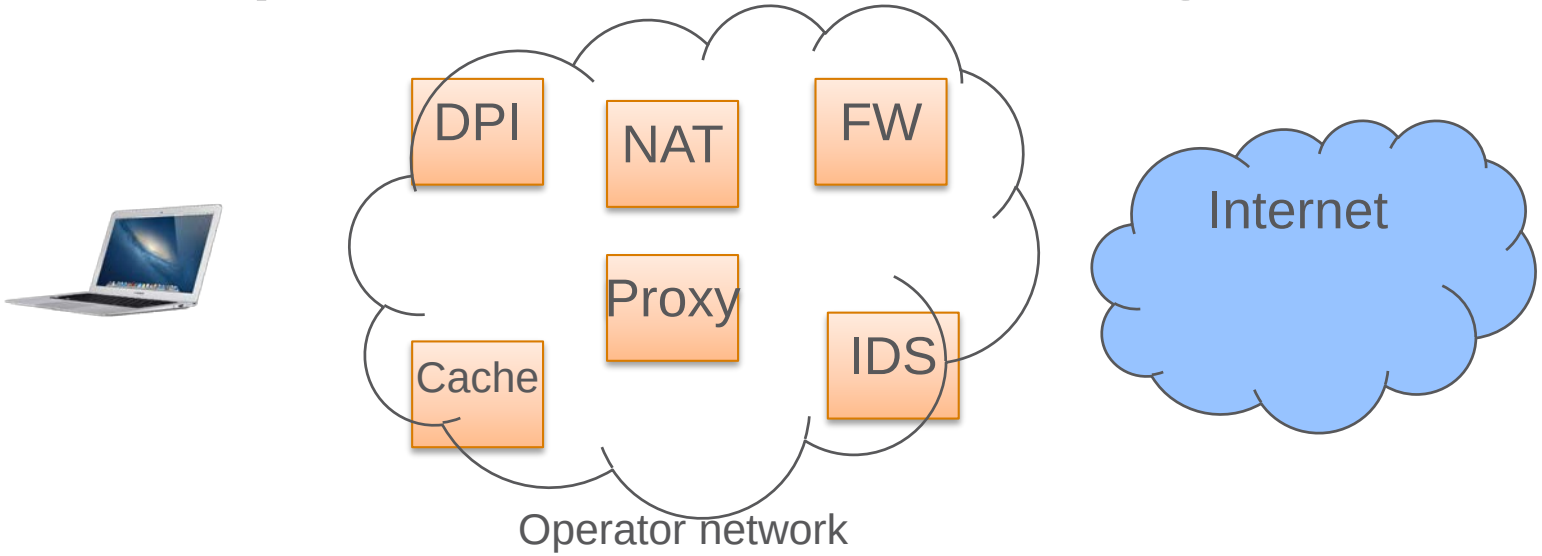
Example: mitigating attacks

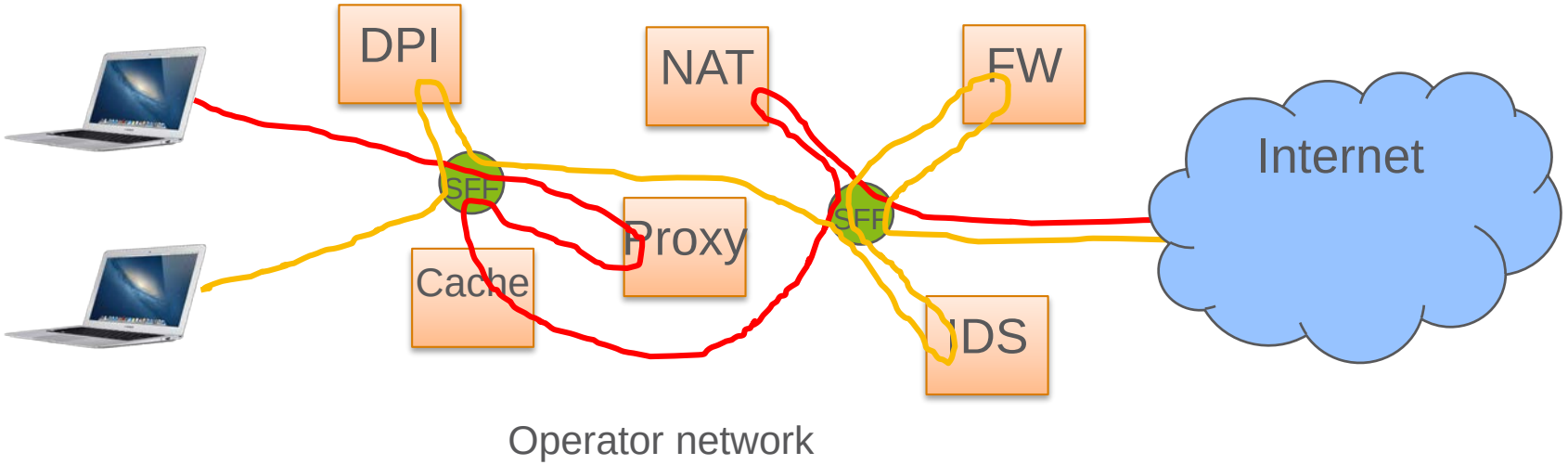


- Checking the validity of packets by middle boxes

How to route the packets through a series of middle boxes?

Example: service chaining





Roadmap

- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little touch on Openflow

Challenges and research problems

- Switch design
  - Find common abstraction
  - Flow table capacity
  - Throughput
- Controller platform
  - Distributed vs centralized
  - Flexibility
- Dependability and security
  - Attack to data plane
  - Attack to control plane
  - Trust, privacy issues

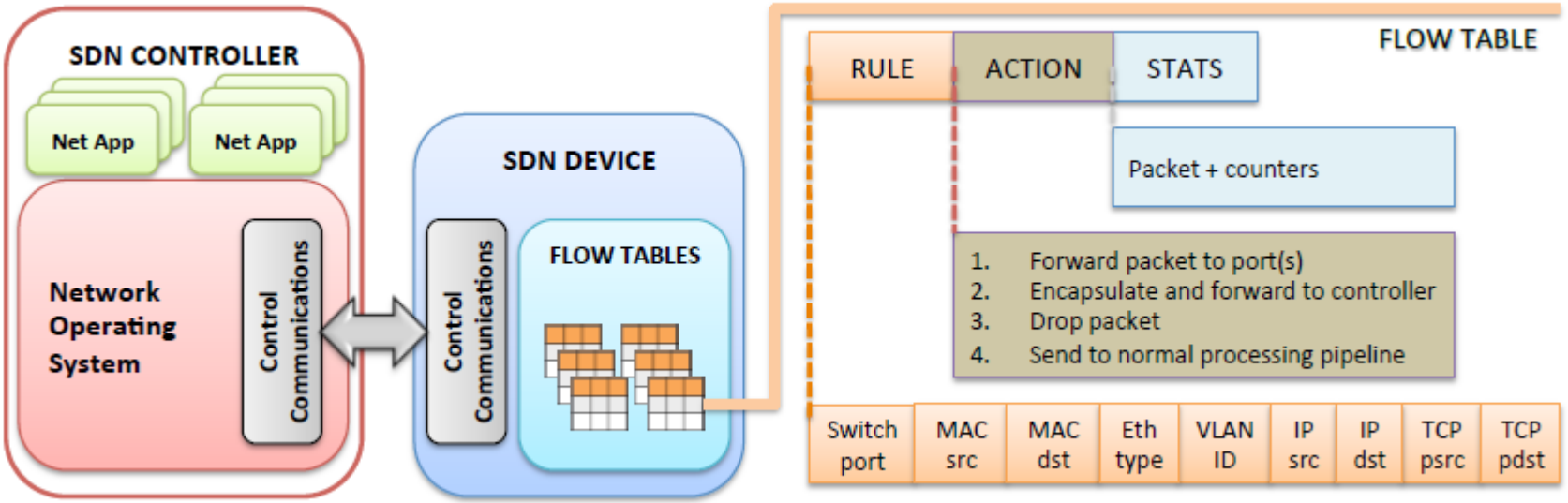
Roadmap

- Reviewing traditional networking
- Examples for motivating SDN
- Enabling networking as developing softwares
- SDN architecture
- Use cases
- Challenges and research problems
- Little touch on Openflow

Openflow

An southbound standard:

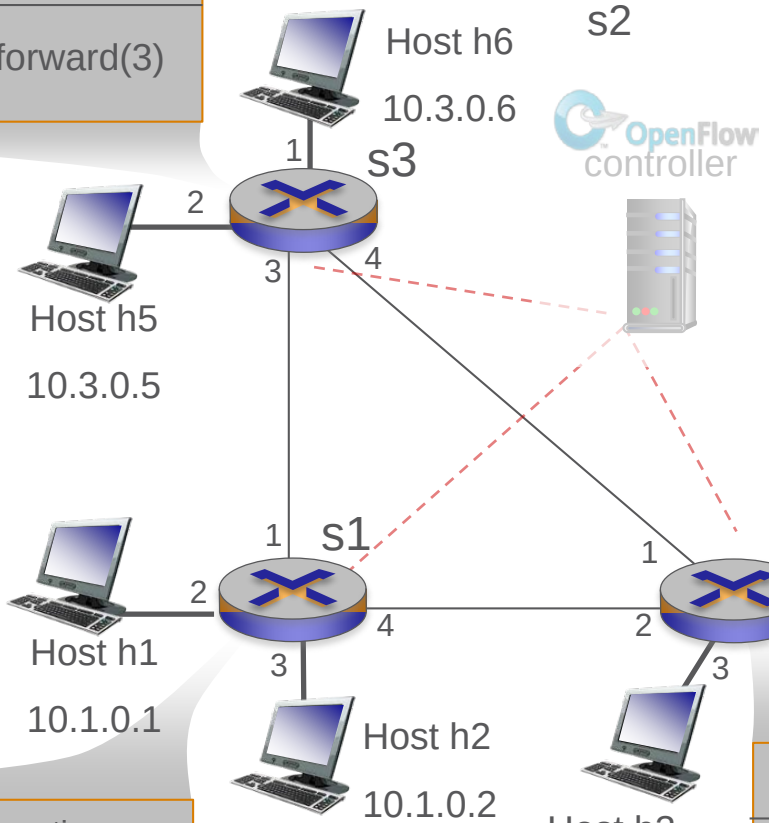
- Provide specification to implement Openflow-enabled forwarding devices
- Communication channel between data and control plane
  - TCP used to exchange messages
- New versions are keeping released



OpenFlow example

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



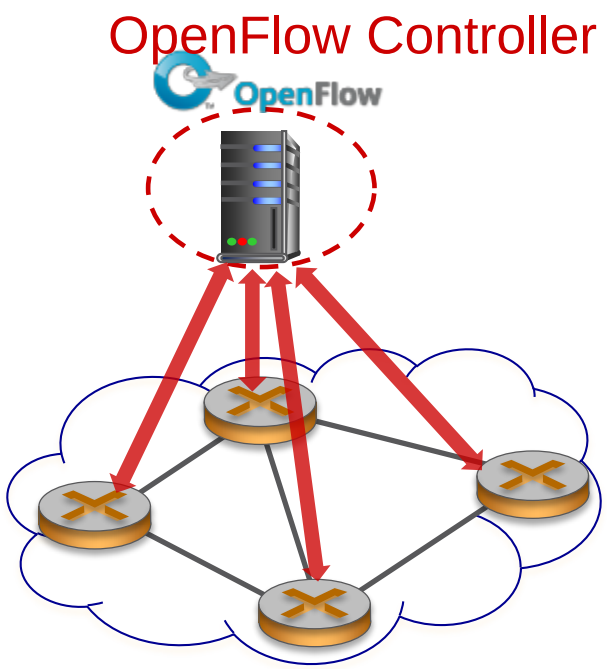
match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

OpenFlow: controller-to-switch messages

Key controller-to-switch messages

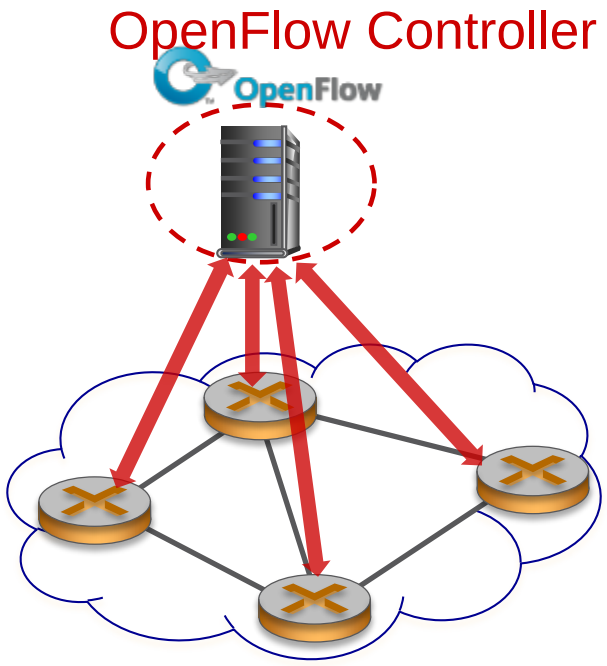
- *features:* controller queries switch features, switch replies
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *packet-out:* controller can send this packet out of specific switch port



OpenFlow: switch-to-controller messages

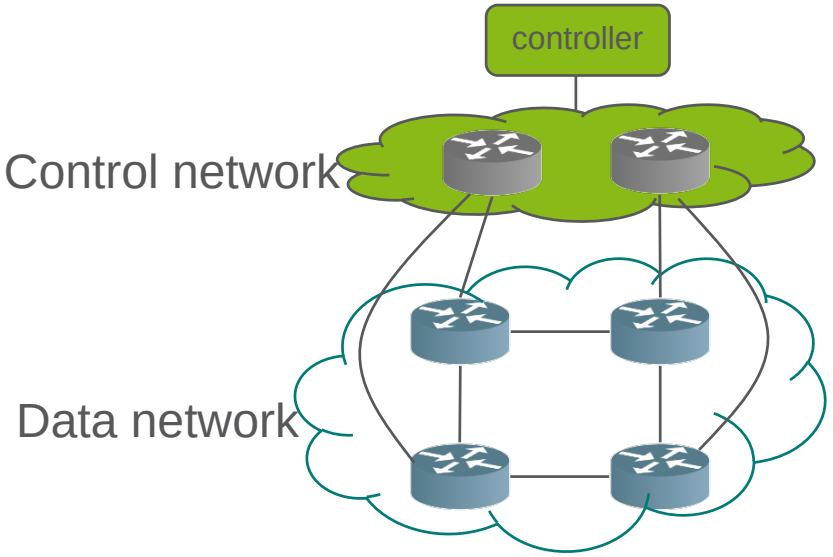
Key switch-to-controller messages

- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.

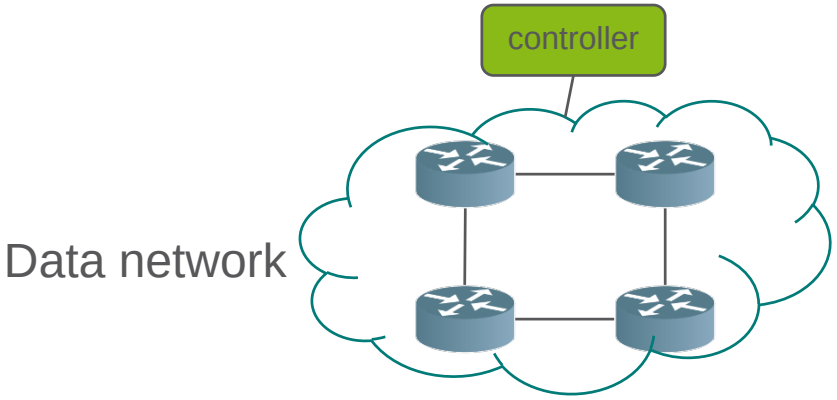


In-band VS out-band Control

- How to setup the communication channels between the SWs and the controller?
  - ***Out-band control:*** building a separate network connecting management ports of SDN switches with the controller. This network is out of the control by the SDN switches. It has to implement traditional L2/L3 routing protocols.
  - ***In-band control:*** the control network has overlap with the data network, in other words, the control traffic may share same network with the data traffic



out-band control



in-band control

Setup the communication channels between the SWs and the controller

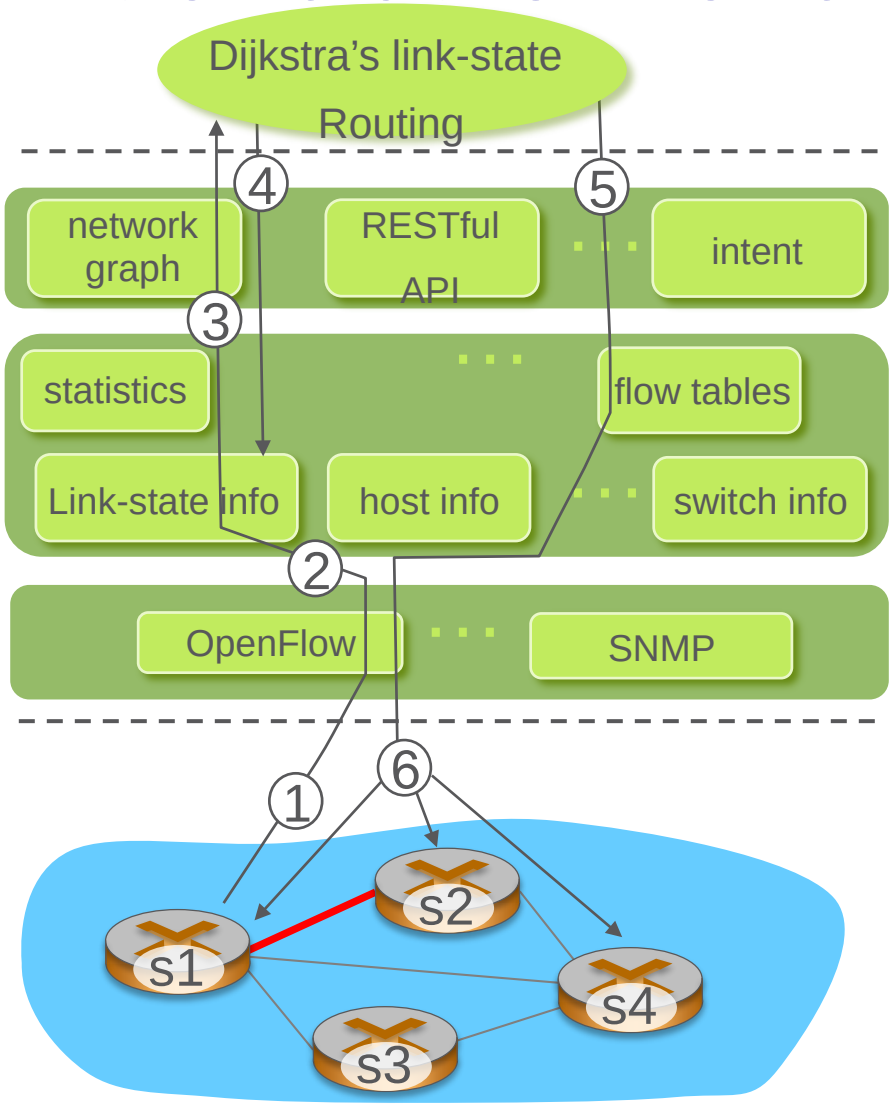
- **Out-band control**, the SWs just initiate the TCP connection through the dedicated port connecting to the control network. The address of the controller is preconfigured in a file or can be retrieved through e.g. DHCP [2]
- **In-band control**
  - If the data network is a complete SDN network, then some rules have to be preinstalled into the switch for bootstrapping. For instance, allowing ARP traffic for the MAC of the controller or gateway, allowing for DHCP packets and TCP traffic from/to the controller. These rules have highest priority then the rules from the controller
  - If the data network is a hybrid SDN network, then initially the connection can be setup like the out-band scenario. Later, the controller may setup rules for the control traffic in order to make the connection more reliable.

- Some examples of out-band control: google B4; in-band control: OVS

Topology discovery

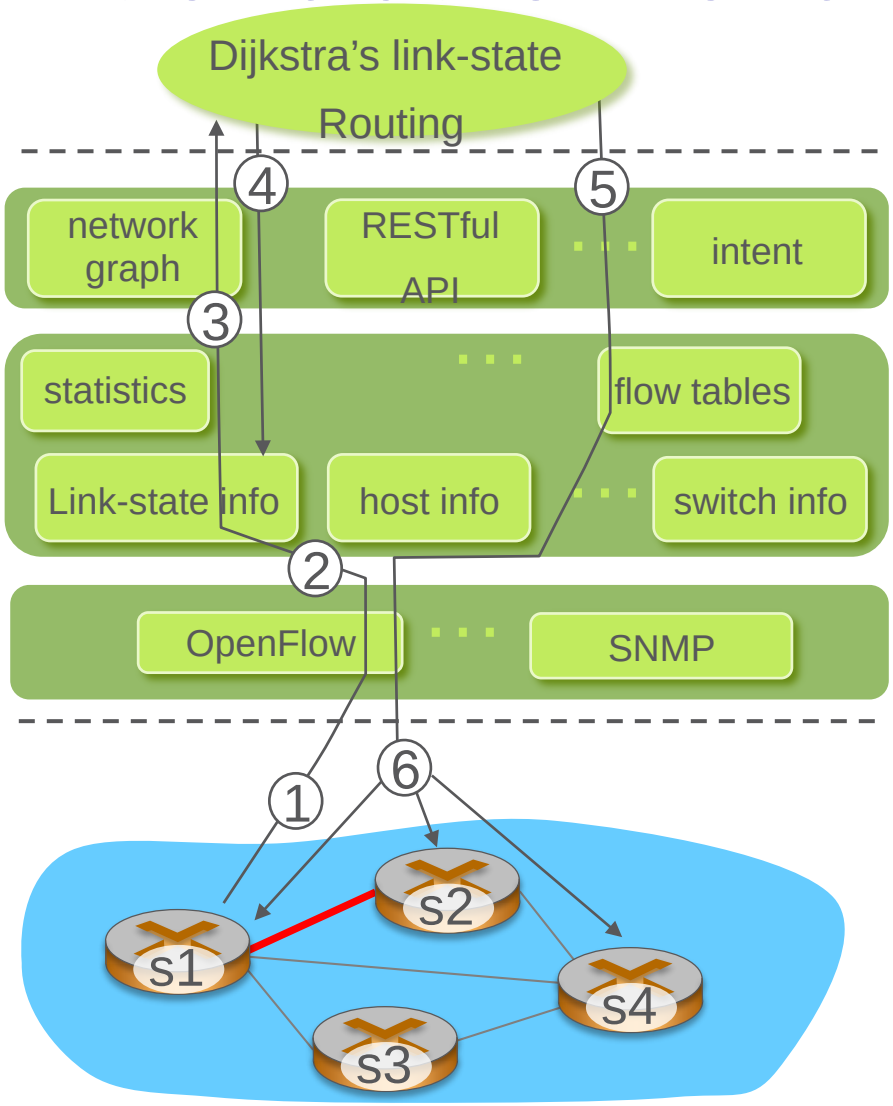
- How does the controller discover the network topology?
  - In the literature, LLDP was proposed to discover the links among switches, however that cannot help with host discovery
  - For the host discovery, SDN relies on that the hosts send packets, then the switches forward the first packet to the controller, therefore, the controller knows which switch connects which host via which port.
  - To get the information of which host connects to which SW, we may let the hosts to send some packets (e.g. LLDP, ICMP) to the switch, then the switch can report this to the controller.

SDN: control/data plane interaction example



1. S1, experiencing link failure using OpenFlow port status message to notify controller
2. SDN controller receives OpenFlow message, updates link status info
3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

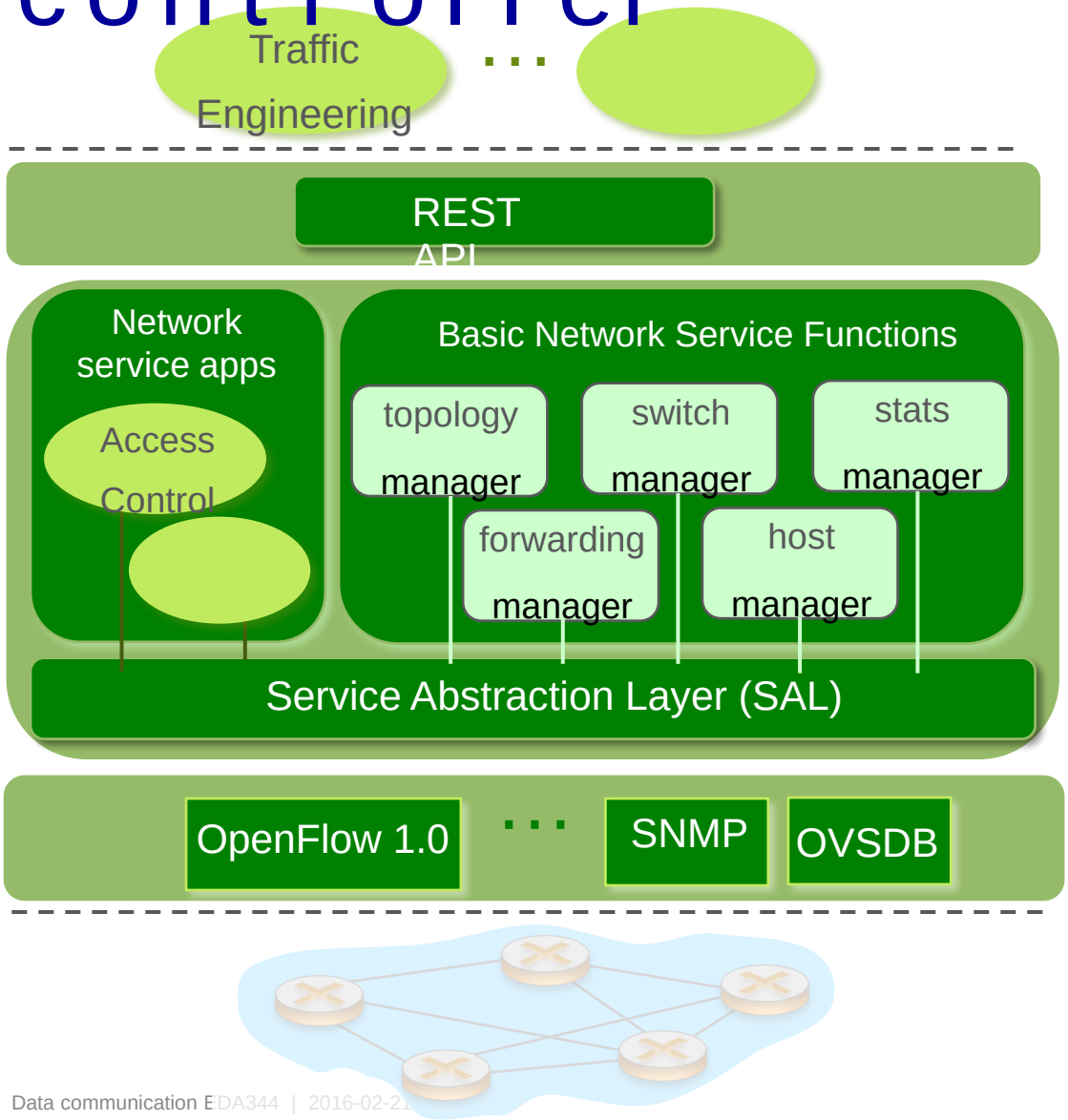
SDN: control/data plane interaction example



5 link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

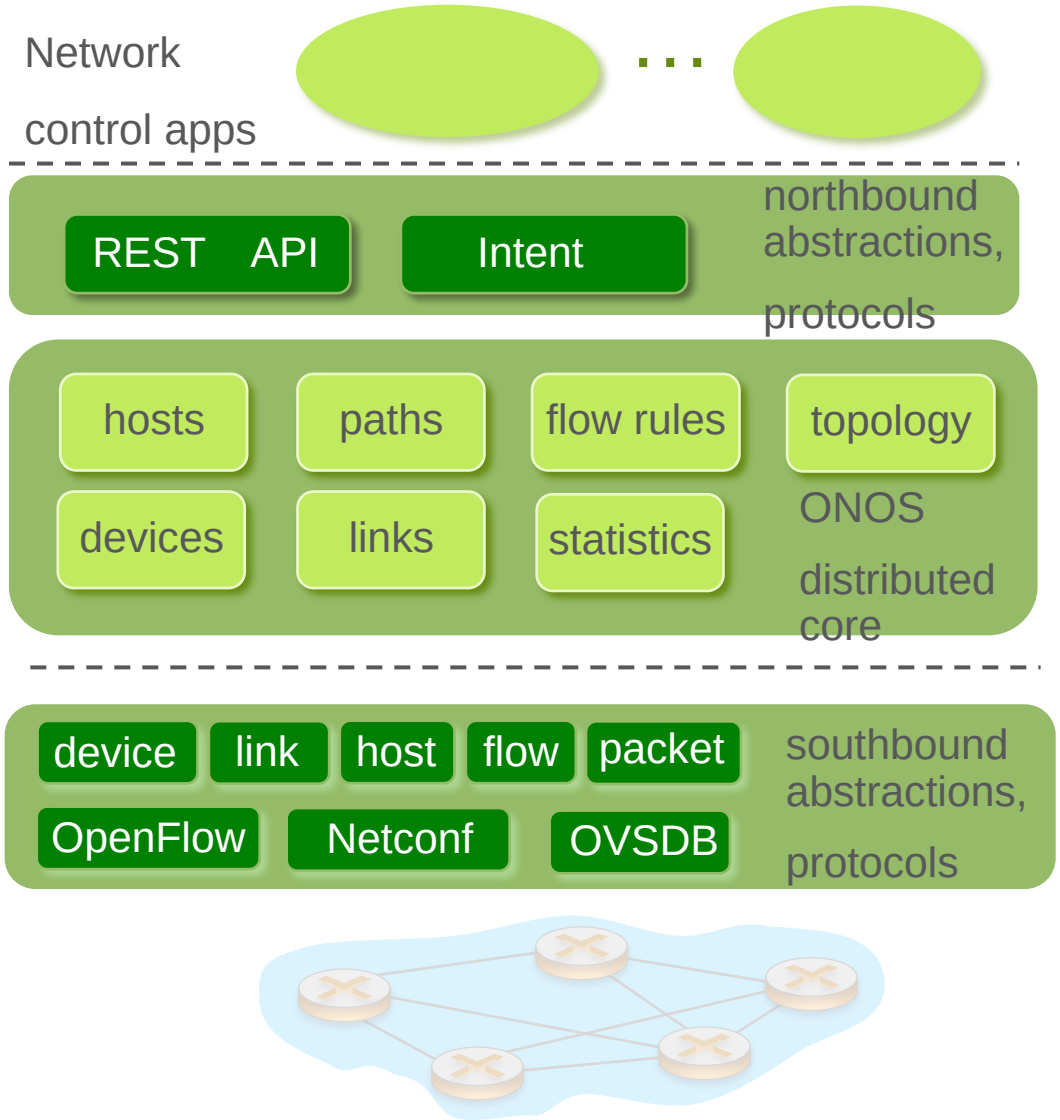
6 Controller uses OpenFlow to install new tables in switches that need updating

Open Daylight (ODL) controller



- ODL Lithium controller
- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

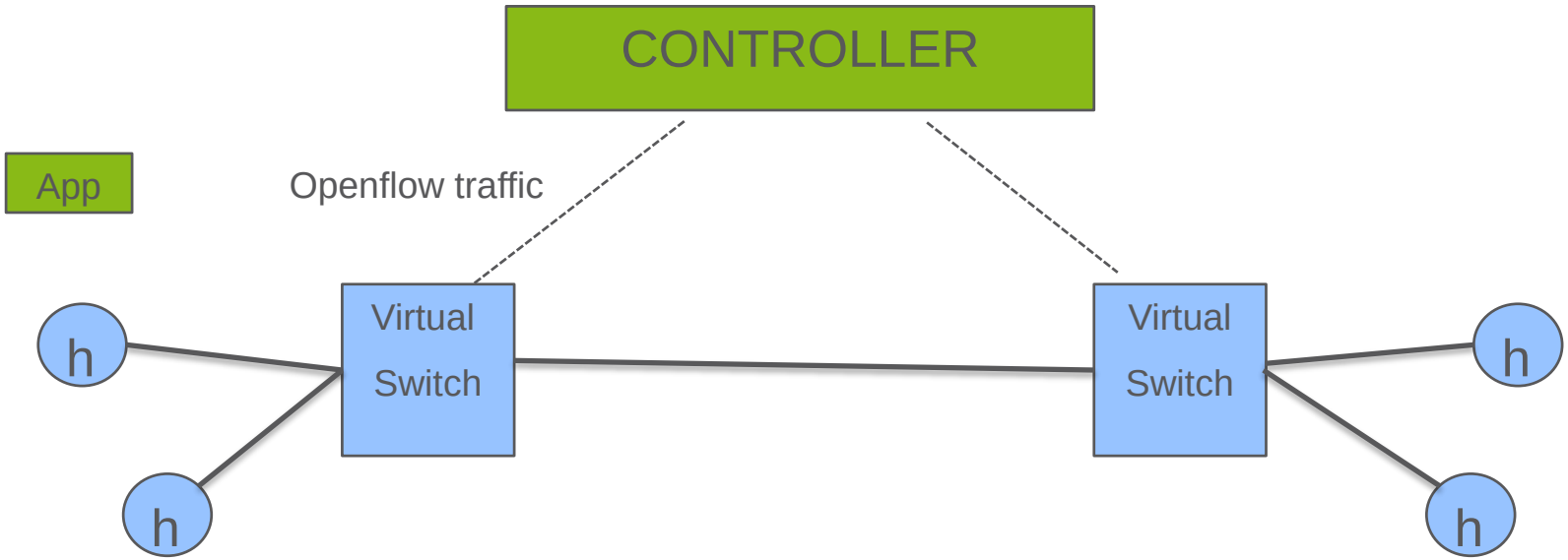
ONOS controller



- control apps separate from controller
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

Mininet

- Provide tools to create virtualized network with OVS
- CLI for manipulating network dynamically
- Virtualized hosts
- Controllers: POX, Ryu, Opendaylight, etc.



```
mininet$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=38.192s, table=0, n_packets=1, n_bytes=98, idle_timeout=60, idle_age=38,
priority=65535,icmp,in_port=2,vlan_tci=0x0000,dl_src=d6:13:79:41:63:43,dl_dst=7e:6c:76:0d:89:c9,nw_src=10.0.0.2,nw_dst=10
.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:1
 cookie=0x0, duration=38.190s, table=0, n_packets=1, n_bytes=98, idle_timeout=60, idle_age=38,
priority=65535,icmp,in_port=2,vlan_tci=0x0000,dl_src=d6:13:79:41:63:43,dl_dst=7e:6c:76:0d:89:c9,nw_src=10.0.0.2,nw_dst=10
.0.0.1,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:1
 cookie=0x0, duration=38.189s, table=0, n_packets=1, n_bytes=98, idle_timeout=60, idle_age=38,
priority=65535,icmp,in_port=1,vlan_tci=0x0000,dl_src=7e:6c:76:0d:89:c9,dl_dst=d6:13:79:41:63:43,nw_src=10.0.0.1,nw_dst=10
.0.0.2,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:2
 cookie=0x0, duration=38.192s, table=0, n_packets=1, n_bytes=98, idle_timeout=60, idle_age=38,
priority=65535,icmp,in_port=1,vlan_tci=0x0000,dl_src=7e:6c:76:0d:89:c9,dl_dst=d6:13:79:41:63:43,nw_src=10.0.0.1,nw_dst=10
.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:2
 cookie=0x0, duration=33.190s, table=0, n_packets=1, n_bytes=42, idle_timeout=60, idle_age=33,
priority=65535,arp,in_port=1,vlan_tci=0x0000,dl_src=7e:6c:76:0d:89:c9,dl_dst=d6:13:79:41:63:43,arp_spa=10.0.0.1,arp_tpa=10
.0.0.2,arp_op=2 actions=output:2
 cookie=0x0, duration=38.193s, table=0, n_packets=1, n_bytes=42, idle_timeout=60, idle_age=38,
priority=65535,arp,in_port=2,vlan_tci=0x0000,dl_src=d6:13:79:41:63:43,dl_dst=7e:6c:76:0d:89:c9,arp_spa=10.0.0.2,arp_tpa=10
.0.0.1,arp_op=2 actions=output:1
 cookie=0x0, duration=33.191s, table=0, n_packets=1, n_bytes=42, idle_timeout=60, idle_age=33,
priority=65535,arp,in_port=2,vlan_tci=0x0000,dl_src=d6:13:79:41:63:43,dl_dst=7e:6c:76:0d:89:c9,arp_spa=10.0.0.2,arp_tpa=10
.0.0.1,arp_op=1 actions=output:1
```

References

- "The road to SDN" http://queue.acm.org/detail.cfm?ref=rss&id=2560327
- Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey.*proceedings of the IEEE*, *103*(1), 14-76.
- “SDN Architecture 1.0 - Open Networking Foundation”, https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf
- Openflow specification v1.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf
- Mininet, mininet.org
- POX controller, http://www.noxrepo.org/

ERICSSON