

Management and Operations of Networks, Services, and Systems

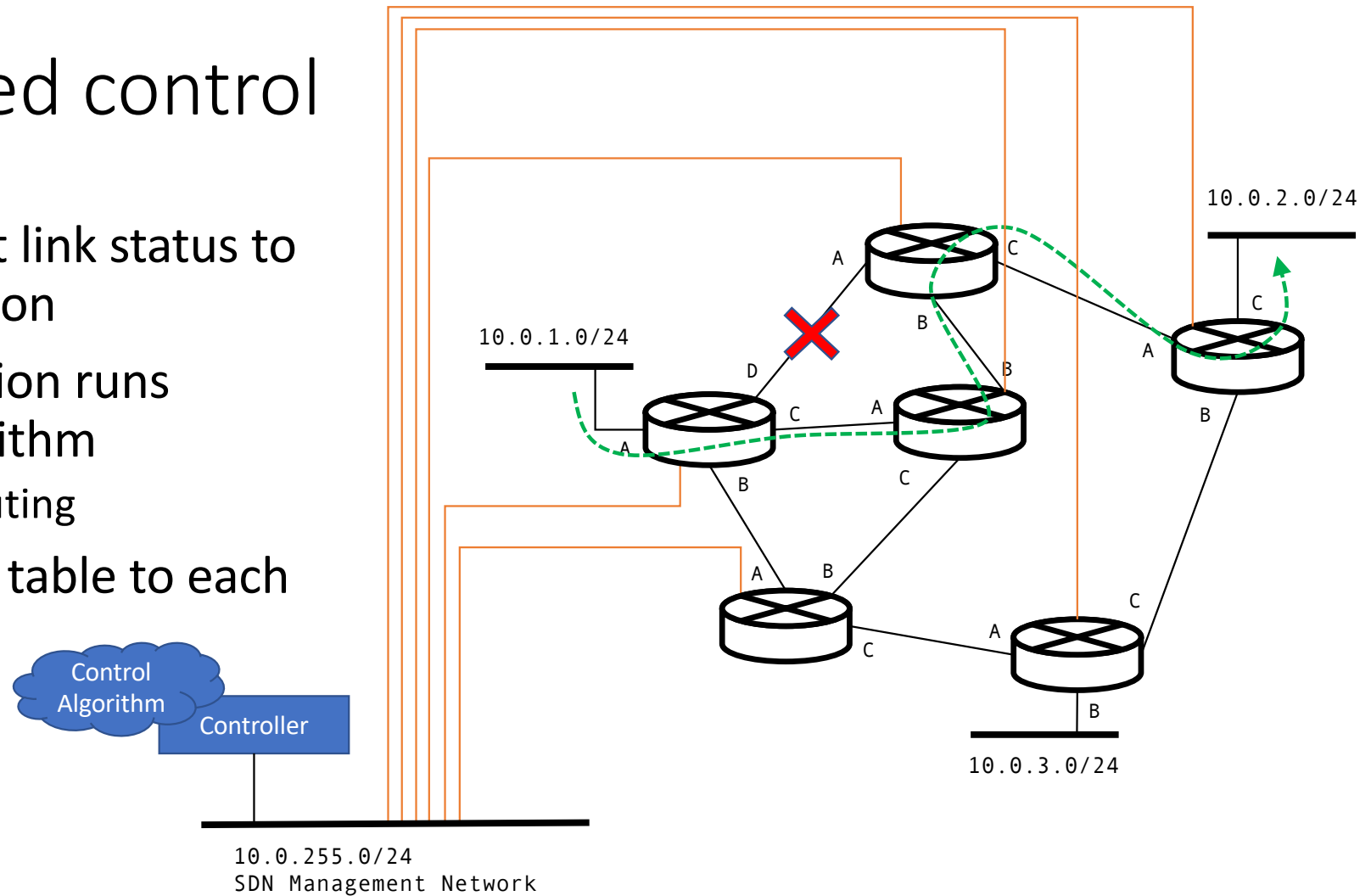
Software Defined Networking

Ricardo Morla

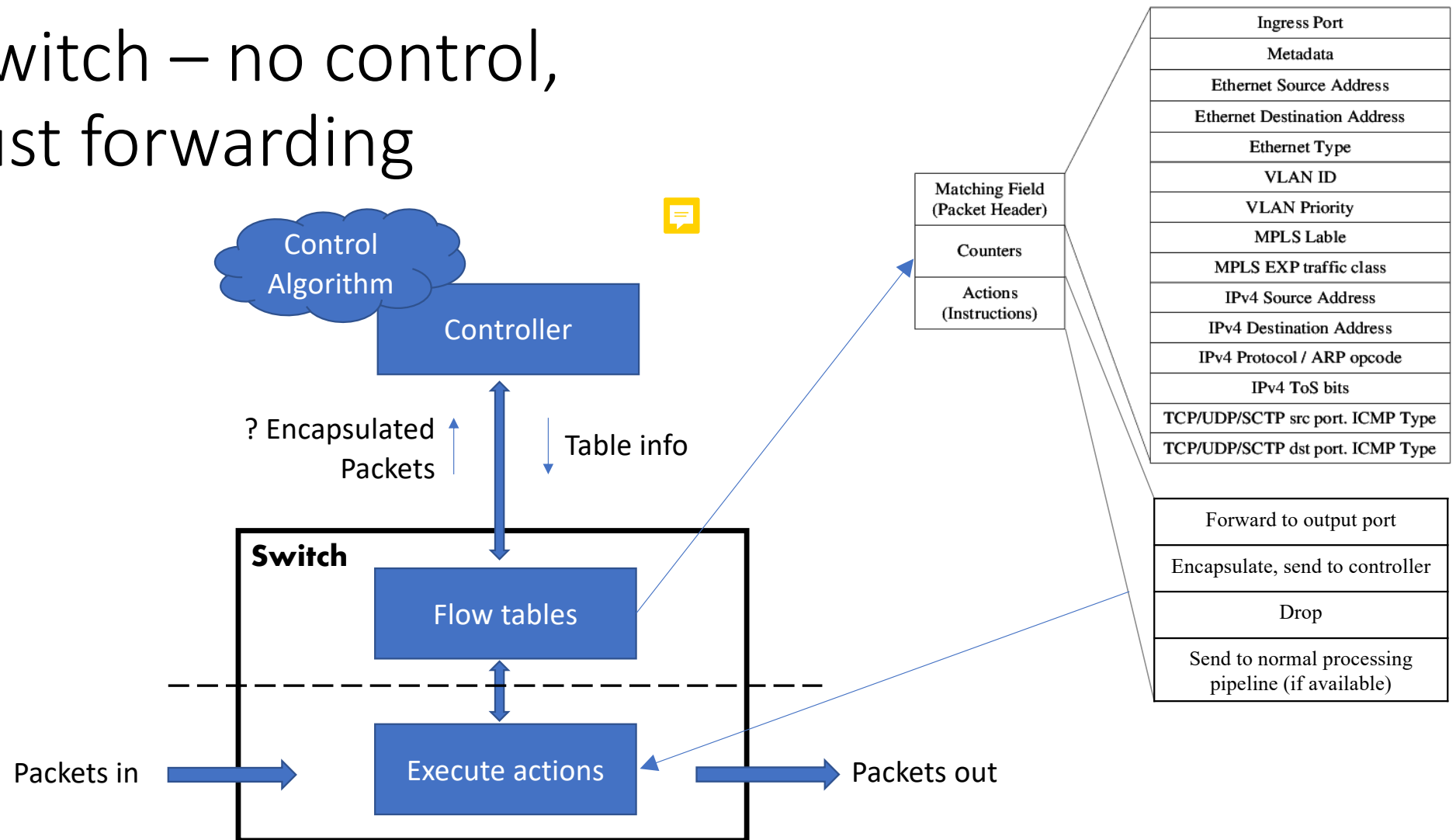
FEUP – GORS/M.EEC, GRS/M.EIC

Centralized control

- Nodes report link status to central location
- Central location runs control algorithm
 - Namely routing
- Returns flow table to each switch




Switch – no control, just forwarding



<https://en.wikipedia.org/wiki/OpenFlow>

Access control example

- Access control 
 - 10.0.0.1 can ping 10.0.0.2




Matching fields			Default: drop
Source	Destination	Protocol	Action
10.0.0.1/32	10.0.0.2/32	ICMP	Forward port 2
10.0.0.2/32	10.0.0.1/32	ICMP	Forward port 1

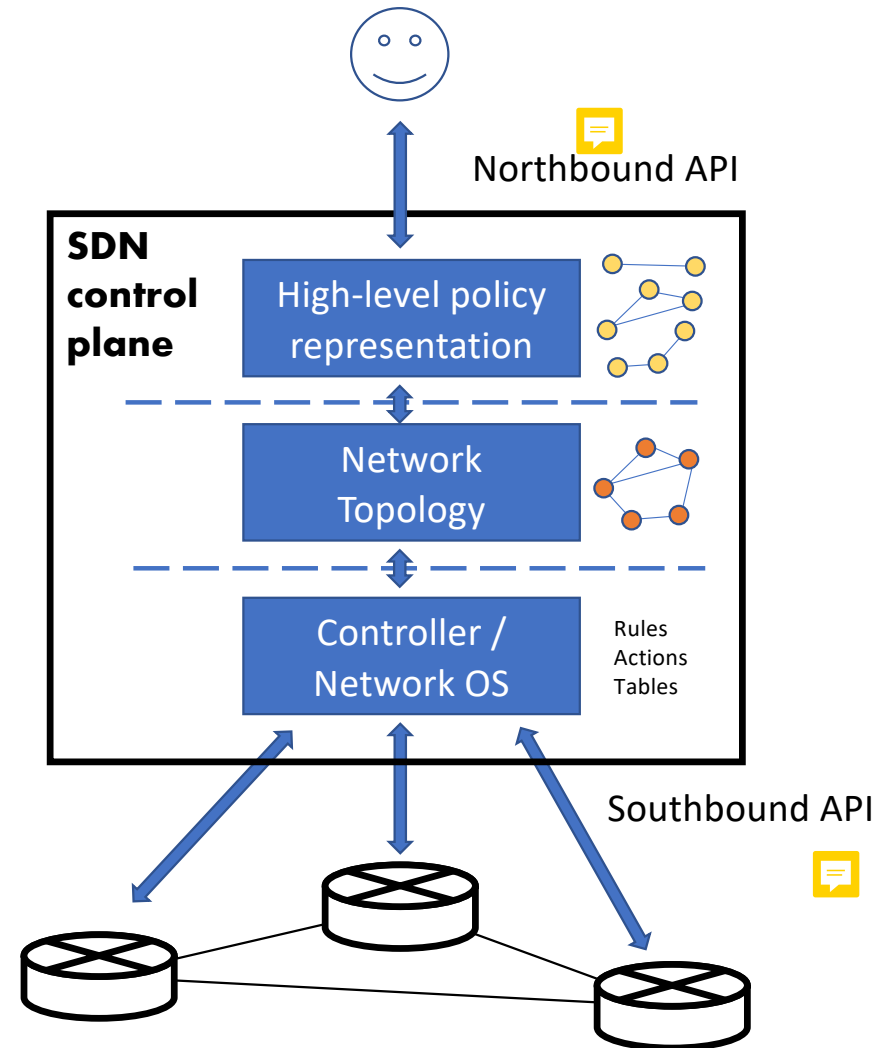
- Routing
 - Routing algorithm sets up destination address and forwarding port
 - Destination IP address – matching field
 - Forward to port – action

Features of SDN

- Hardware abstraction
 - Control works on any hardware that implements API
- Programmable
 - Hardware/software bundles only allow configuration of existing control algorithms
 - SDN allows us to develop our own algorithms
- Centralized control of policies
 - Security policies “nodes of type X can only talk with nodes of type Y”
 - Routing policies “route guest traffic through the firewall”
 - Quality of service policies “prioritize voice traffic”

Scaling up

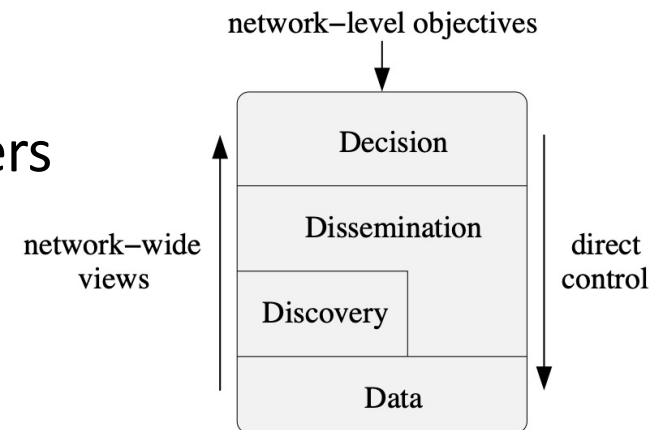
- Control multiple switches 
- Abstract network topology
- Support high-level representation of policies (e.g. virtualization, intents)  



4D clean slate

*Albert Greenberg, Gisli Hjalmtýsson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. 2005. A clean slate 4D approach to network control and management. SIGCOMM Comput. Commun. Rev. 35, 5 (October 2005), 41–54.
DOI:<https://doi.org/10.1145/1096536.1096541>*

- 2005 Proposal
- Dissemination: connects switches and controllers
- Discovery: devices and neighborhoods



The road to SDN

Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The road to SDN: an intellectual history of programmable networks. SIGCOMM Comput. Commun. Rev. 44, 2 (April 2014), 87–98. DOI:<https://doi.org/10.1145/2602204.2602219>

- Active networking
- Control and data plane separation
- Openflow and network OS's
- Network virtualization

NOX – a network operating system

Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. 2008. NOX: towards an operating system for networks. SIGCOMM Comput. Commun. Rev.38, 3 (July 2008), 105–110. DOI:<https://doi.org/10.1145/1384609.1384625>

- NOX between switches and control applications
- Programmatic interface for control applications
 - Event handlers
 - Network view
 - Library (routing, DHCP, etc)

B4 – Google's private WAN (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. 2013. B4: experience with a globally-deployed software defined wan. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13). Association for Computing Machinery, New York, NY, USA, 3–14. DOI:<https://doi.org/10.1145/2486001.2486019>

- Merchant silicon, lower reliability
- Control plane in software
- Target 100% link utilization
- Centralized traffic engineering

References

- <http://www.cse.wustl.edu/~jain/cse570-18/>
- <https://people.csail.mit.edu/alizadeh/courses/6.888/schedule.html>

Ryu controller example

- Component-based software defined networking framework
- <https://ryu-sdn.org>
- <https://book.ryu-sdn.org/en/Ryubook.pdf>
- Mininet setup: 1 switch, 3 hosts
- Ryu openflow controller rest API (`ryu.app.ofctl_rest`)
- Add flow table entries to allow h1 ping h2

Setup docker with openvswitch and ryu

```
docker run -it --name ryu --rm --privileged  
-v /lib/modules:/lib/modules osrg/ryu-book
```

How to start a shell

```
docker exec -it ryu /bin/bash
```

@Mininet shell

```
mn --topo single,3 --mac --switch ovsk --  
controller remote -x
```

```
mininet> h1 ping h2
```

@Ryu controller shell

```
ryu-manager --verbose ryu.app.ofctl_rest  
wsgi starting up on http://0.0.0.0:8080
```

@Another shell use Ryu REST to configure the switch

List switches / datapaths

```
curl -X GET  
http://localhost:8080/stats/switches
```

List ports (on switch 1)

```
curl -X GET  
http://localhost:8080/stats/port/1
```

Delete all flow entries

```
curl -X DELETE  
http://localhost:8080/stats/flowentry/clear/  
1
```

List all flow entries

```
curl -X GET  
http://localhost:8080/stats/flow/1
```

Add flow entry

```
curl -X POST -d '{ "dpid": 1, "cookie": 1,  
"cookie_mask": 1, "table_id": 0,  
"idle_timeout": 30, "hard_timeout": 30,  
"priority": 1111, "flags": 1,  
"match": { "in_port": 1 },  
"actions": [ { "type": "OUTPUT", "port": 2 } ] }'  
http://localhost:8080/stats/flowentry/add
```

Can h1 ping h2 yet?

Debug with tcpdump. What flow entry is missing?

Increase cookie count for new entry

Management and Operations of Networks, Services, and Systems

Software Defined Networking

Ricardo Morla

FEUP – GORS/M.EEC, GRS/M.EIC