Software-defined Networking
Session I

Introduction to Software-defined Networking

Block Course – 14-18 March 2016

David Koll

Course overview

Block Course Software-defined Networking

Introduction: basics (this week)

Advanced: more cool stuff (next week)

Formalities:

- 5 ECTS for this week
- Requirements to pass:
  - See Wiki page

- Exercise submission:
  - All exercises together with report
  - Deadline: April 30th, 23:59 CET
  - Submit code as well!

IMPORTANT:
FlexNow registration deadline:
TODAY, 23:59 CET

Course overview

Daily Schedule:

Session	Time Slot
Morning Lecture	09.15-10.45
Morning Exercise	11.15-12.45
Afternoon Lecture	14.00-15.30
Afternoon Exercise	15.30-open end

We will deviate from this schedule at some points throughout the course

Detailed Course overview

Day	Morning Session 1	Morning Session 2	Afternoon Session 1	Afternoon Session 2
Mon	Introducing SDN	Exercise	OpenFlow	Exercise
Tue	Network Virtualization	Exercise & Quiz	SDN Controllers	Exercise
Wed	SDN Security	Exercise & Quiz	Python Recap	Exercise
Thu	Mininet I	Exercise	Mininet II	Exercise
Fri	Mininet III	Exercise	Mininet IV	Exercise

Detailed Course overview

Day	Morning Session 1	Morning Session 2	Afternoon Session 1	Afternoon Session 2
Mon	**Introducing SDN**	Exercise	**OpenFlow**	Exercise
Tue	**Network Virtualization**	Exercise & Quiz	**SDN Controllers**	Exercise
Wed	**SDN Security**	Exercise & Quiz	**Python Recap**	Exercise
Thu	**Mininet I**	Exercise	**Mininet II**	Exercise
Fri	**Mininet III**	Exercise	**Mininet IV**	Exercise

Lectures: Basic concepts you need to know about SDN and programming for SDN

Detailed Course overview

Day	Morning Session 1	Morning Session 2	Afternoon Session 1	Afternoon Session 2
Mon	Introducing SDN	**Exercise**	OpenFlow	**Exercise**
Tue	Network Virtualization	**Exercise** & Quiz	SDN Controllers	**Exercise**
Wed	SDN Security	**Exercise** & Quiz	Python Recap	**Exercise**
Thu	Mininet I	**Exercise**	Mininet II	**Exercise**
Fri	Mininet III	**Exercise**	Mininet IV	**Exercise**

Exercises: Review and/or implement the concepts taught in the lectures

Detailed Course overview

Day	Morning Session 1	Morning Session 2	Afternoon Session 1	Afternoon Session 2
Mon	Introducing SDN	Exercise	OpenFlow	Exercise
Tue	Network Virtualization	Exercise & **Quiz**	SDN Controllers	Exercise
Wed	SDN Security	Exercise & **Quiz**	Python Recap	Exercise
Thu	Mininet I	Exercise	Mininet II	Exercise
Fri	Mininet III	Exercise	Mininet IV	Exercise

Quizzes: short (30 minute) tests

Detailed Course overview

Day		
April 21st/22nd	**Presentations (tentative)**	
April 30th	**Final submission day**	

Presentations: train your presentation skills based on hot topics in SDN (15+5)

Grading

- Earn 50% of the points of the exercises.
- **Grading**:
  - Final presentation (25%)
  - A review of an SDN research paper (25%)
  - Two graded quizzes (25%)
  - Two graded exercises (25%)

Detailed Course overview

Day	Morning Session 1	Morning Session 2	Afternoon Session 1	Afternoon Session 2
Mon	Introducing SDN	**Exercise**	OpenFlow	**Exercise**
Tue	Network Virtualization	**Exercise & Quiz**	SDN Controllers	**Exercise**
Wed	SDN Security	**Exercise & Quiz**	Python Recap	**Exercise**
Thu	Mininet I	**Exercise**	Mininet II	**Exercise**
Fri	Mininet III	**Exercise**	Mininet IV	**Exercise**

Exercise/Quiz: graded

Final report

Final submission has to contain all exercises, paper review(s) and presentation slides, send by email to:

skulkarni~at~cs.uni-goettingen.de

Introduction to Software-defined Networks – Session I

Partly based on slides of Nick McKeown, Scott Shenker, Nick Feamster, and Jennifer Rexford

Why this course?

„Software-Defined Networks **– the counter model of the internet“**
– *heise.de*

“**November 2014: Cisco** declares “game over” for SDN competitors […], prompting reaction from two industry groups that the game has just begun; **Alcatel-Lucent** and **Juniper** also virtualize their routers […]; **AT&T** and other unveil ONOS […] as an alternative […].”
– *networkworld.com*

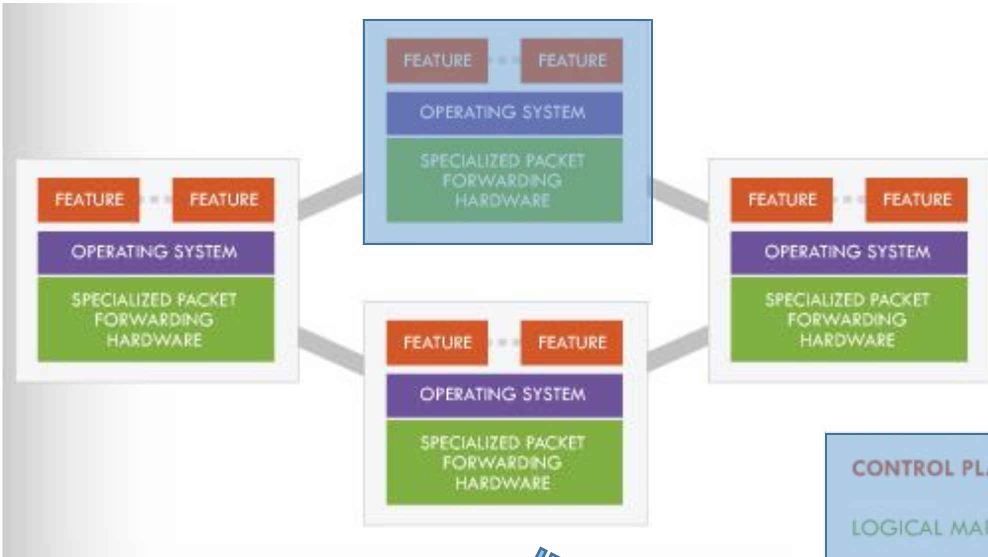
“Many solution providers believe 2016 is the year that **SDN will truly begin to reshape the networking landscape**”
– *crn.com*

What is Software-defined Networking?

"The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices."
– The Open Networking Foundation*

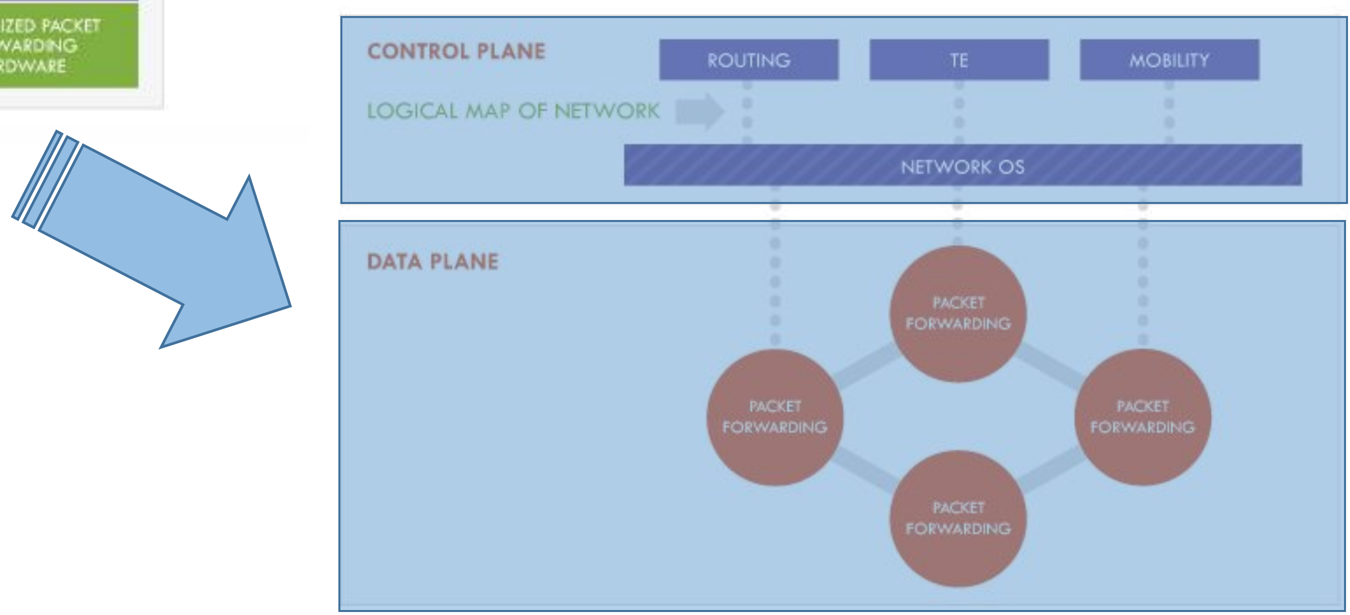
*** Google, Facebook, Microsoft, Deutsche Telekom, Verizon, Yahoo, Cisco, Citrix, Dell, Ericsson, HP, IBM, Juniper Networks, NEC, Netgear, VMWare, …**
…and various institutions from academia (e.g., Stanford, Berkeley)

SDN in a Nutshell



"The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices."

– **The Open Networking Foundation**



Taken from: http://www.opennetsummit.org/archives/apr12/site/why.html

The History behind the Hype

Going to talk about…

What are the origins of SDN?

Why do we need SDN?

Where are we now?

… before we dive into the technical details of SDN

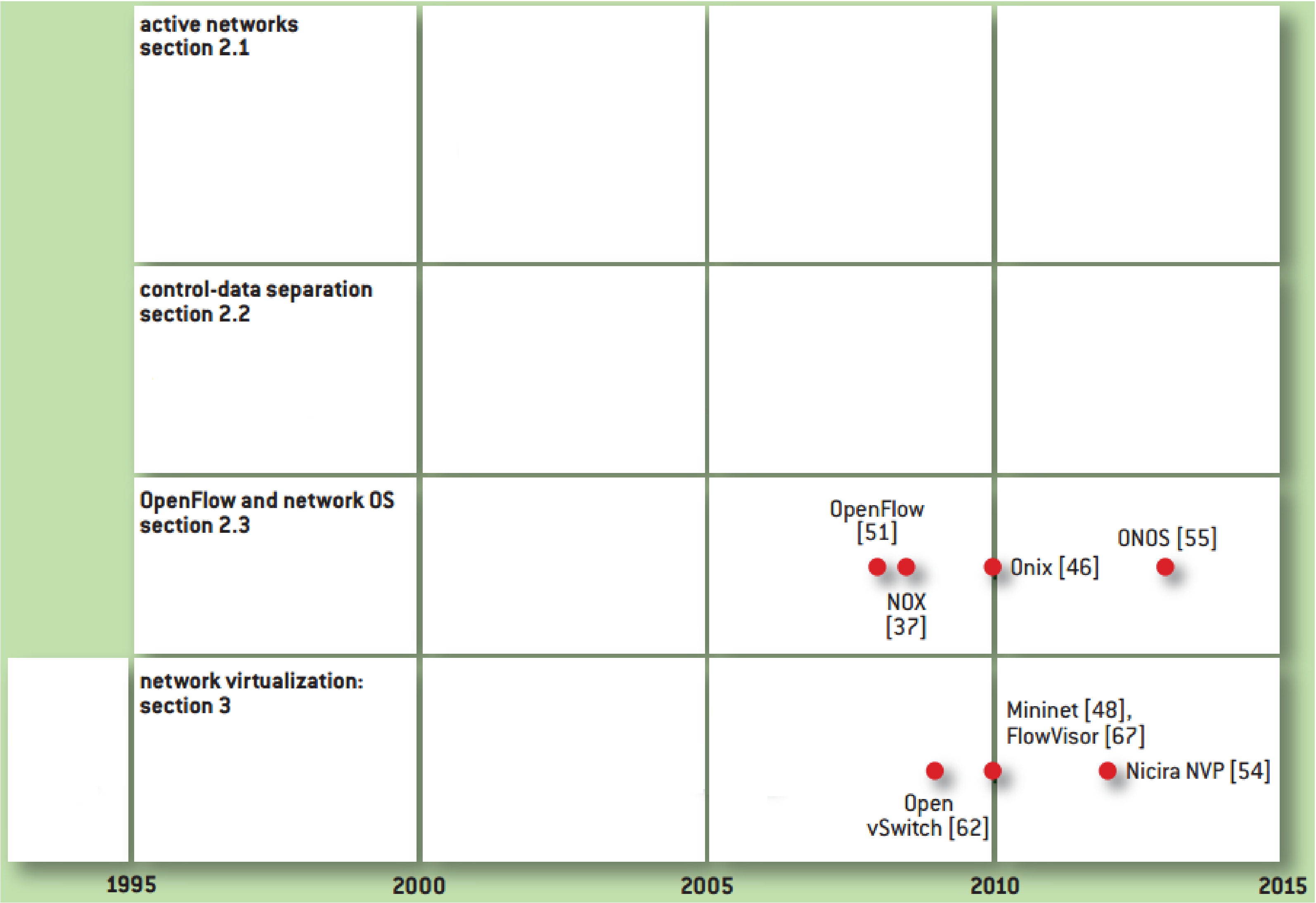
The History behind the Hype

The concepts behind SDN are not really new!

Scott Shenker: *„[SDN is] not a revolutionary technology, [it is] just a way of organizing network functionality."[1]*

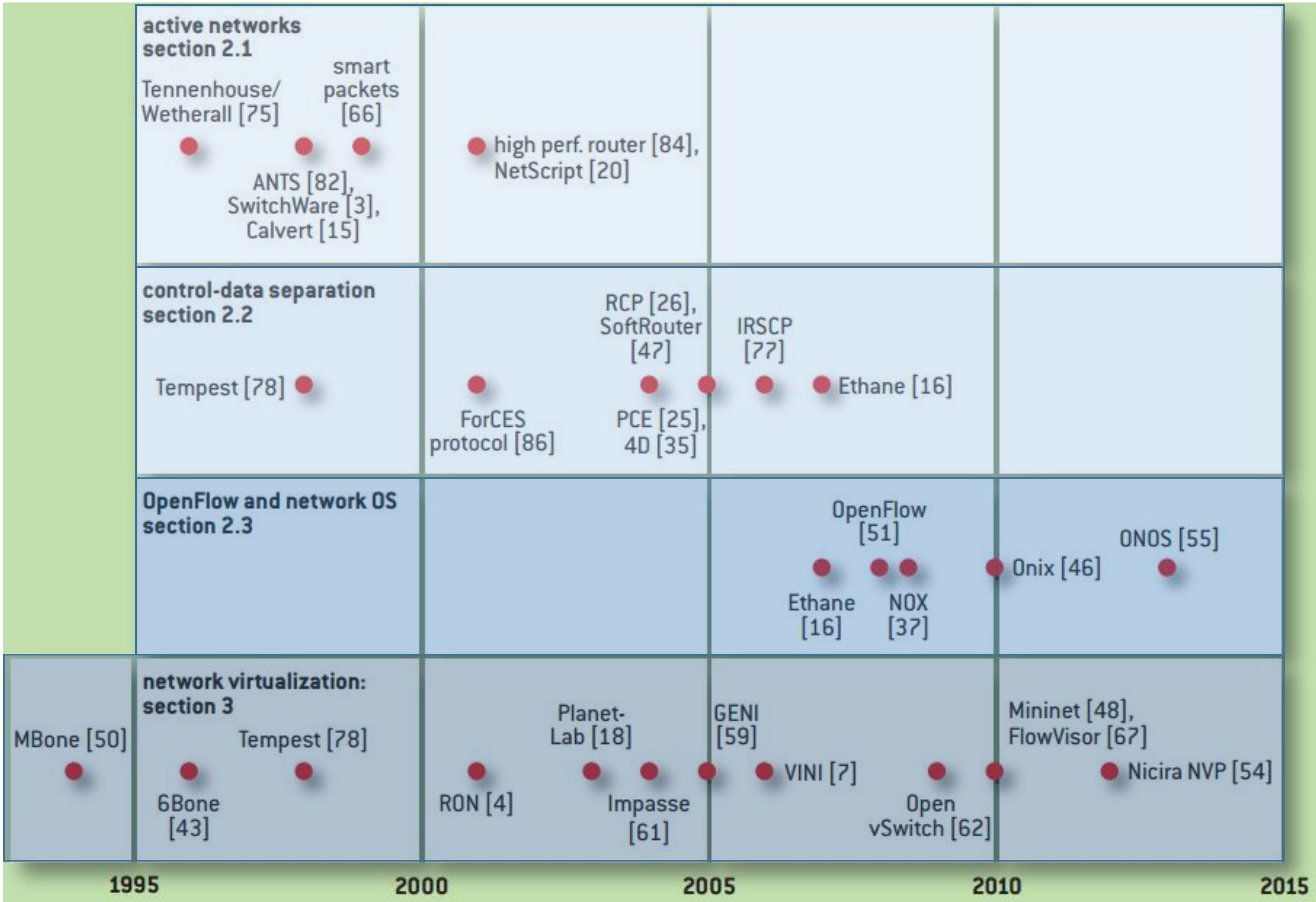
[1] S. Shenker in his talk „A Gentle Introduction to Software-defined Networks"

The History behind the Hype



N. Feamster et al.: „The Road to SDN – An intellectual history of programmable networks" *ACM SIGCOMM Computer Communication Review* 44.2 (2014): 87-98.

The History behind the Hype



N. Feamster et al.: „The Road to SDN – An intellectual history of programmable networks" *ACM SIGCOMM Computer Communication Review* 44.2 (2014): 87-98.

A brief history of programmable networks:

Active Networks

Active Networks?

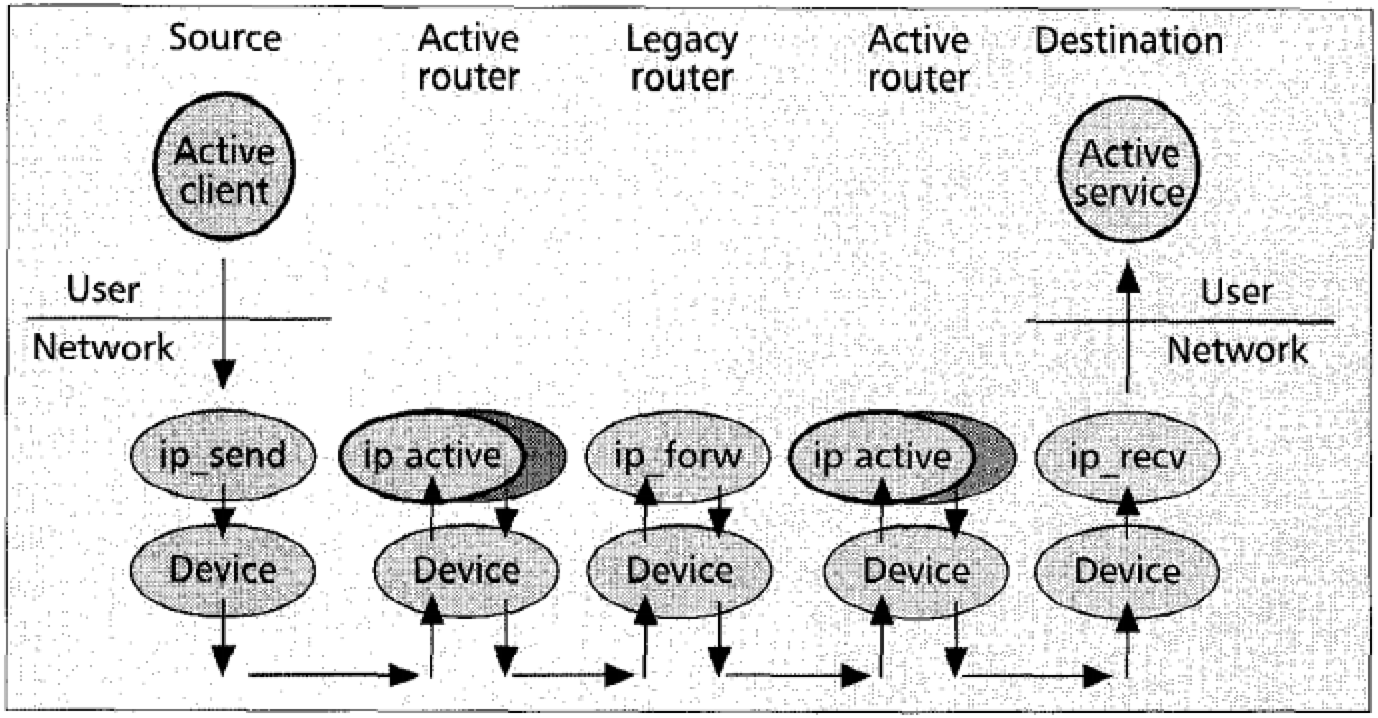
- End of 1990s: network ossification (idea->deployment: 10 years!)

- Goal: opening up network control
- Envisioned method: make network devices programmable via an API

- API could be accessed via two models:
  - **Capsule model:** code included in data packets transmitted in-band [1]
  - **Programmable router/switch model:** code transmitted out-of-band [2]

[1] Wetherall, et al.: *"ANTS: a toolkit for building and dynamically deploying network protocols."* In Proceedings of IEEE OpenArch 1998.
[2] Bhattacharjee, S., Calvert, K.L. et al.: *"An architecture for active networks".* In Proceedings of High-Performance Networking 1997.

Active Networks



■ **Figure 1.** *Application-specific processing within the nodes of an active network.*

Co-existence of legacy routers with active routers

[1] Tennenhouse, et al.: *"A survey of active network research." IEEE Communications Magazine,* 35.1 (1997): 80-86.

Why did active networks fail?

- Timing was off
  - End of 1990s: no data-centers/clouds yet
  - Hardware was expensive (compared to 2015)

- Conceptual mistakes:
  - Programmable by end-users (security?)
  - Limited interoperability

The Legacy of Active Networks

- Intellectual contributions of Active Networks:
  - Programmable network functions
  - Network virtualizations (de-multiplexing of packets according to their header)

The concepts behind SDN are not really new!

(we see both contributions in today‘s SDN)

A brief history of programmable networks:
Control and data plane separation

Control and Data Plane Separation

- Early 2000s: increasing traffic volumes, network sizes
  - need for traffic engineering
- But: conventional routers/switches: tight integration of data and control planes
  - Problem: Hard to debug and control router behaviour
- Goal: Traffic control and configuration should be easier
- Envisioned method: decouple control and data plane

Control and Data Plane Separation

- Mainly two innovations:
  - Open interface between the control and data plane (e.g., ForCES [1])
  - *Logically* centralized control of the network(e.g., RCP [2])

- Compared to active networks:
  - **Targeted at network administrators** rather than end-users
  - **Programmability in control plane** rather than in data plane
  - **Network wide control** rather than device-level configuration

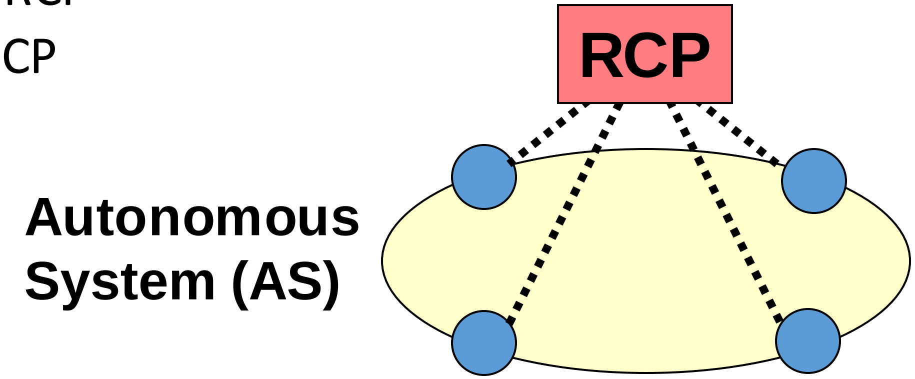
[1] Yang, et al. "*Forwarding and control element separation (ForCES) framework*." RFC 3746, April, 2004.
[2] Caesar, et al. "*Design and implementation of a routing control platform.*" Proceedings of Usenix NSDI, 2005.

RCP - Separating *Inter*domain Routing

- Compute interdomain routes for the routers
  - Input: BGP-learned routes from neighboring ASes
  - Output: forwarding-table entries for each router

- Backwards compatibility with legacy routers
  - RCP speaks to routers using BGP protocol

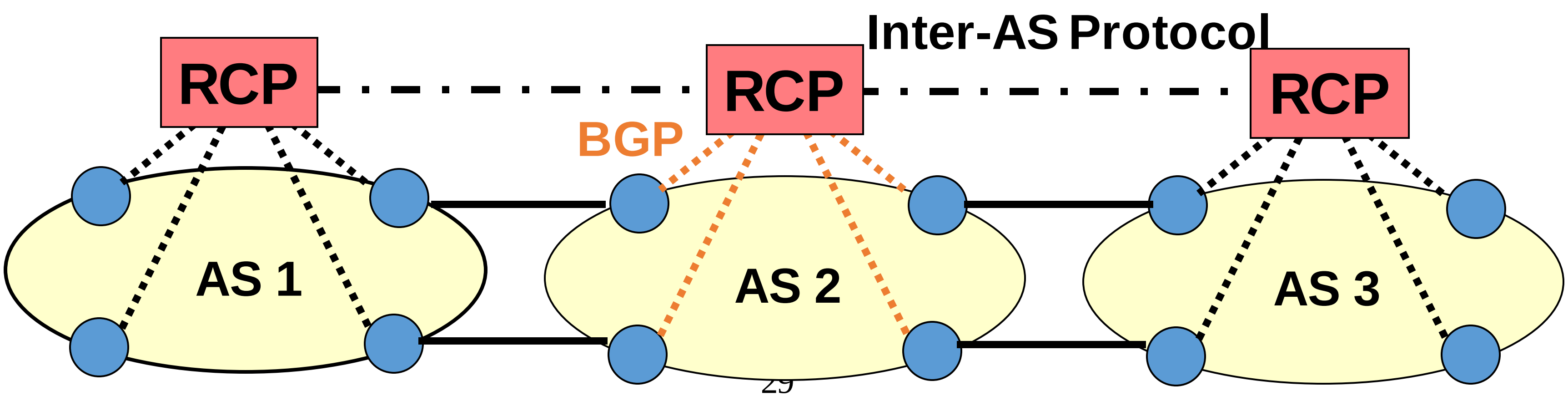
- Routers still run intradomain routing protocol
  - So the routers can reach the RCP
  - To reduce overhead on the RCP



Incremental Deployability

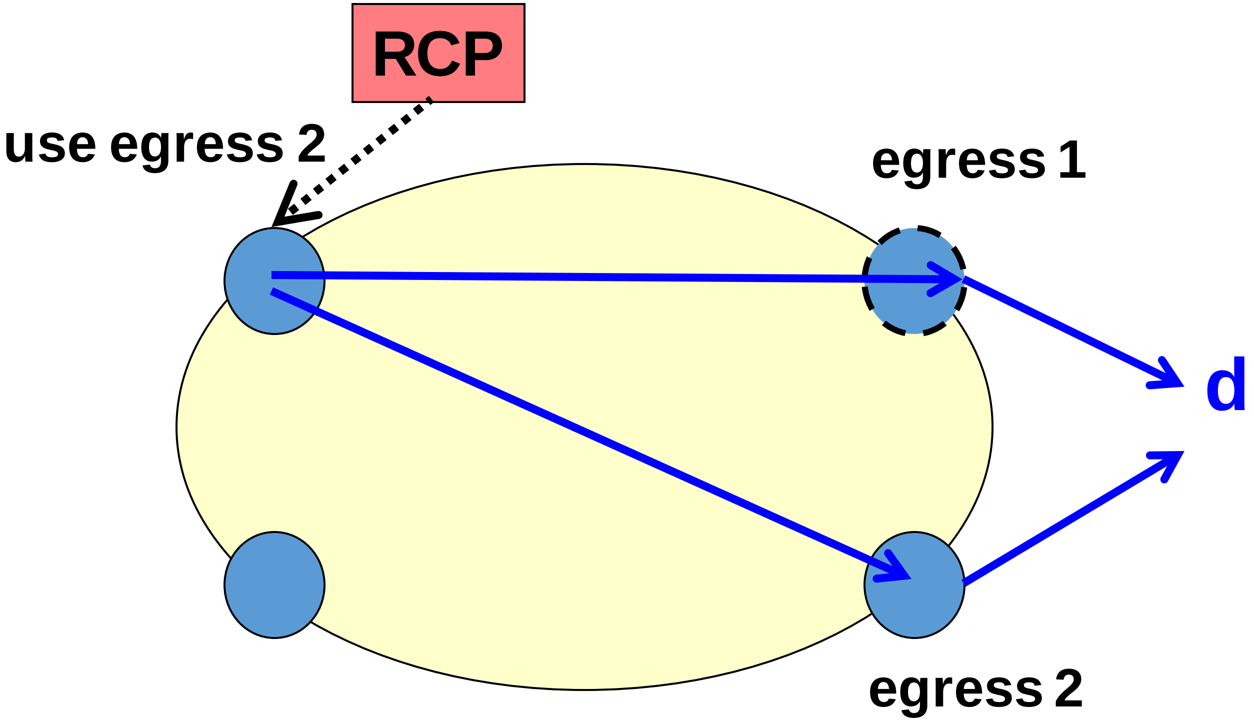
- Backwards compatibility
  - Work with existing routers and protocols
- Incentive compatibility
  - Offer significant benefits, even to the first adopters

RCP tells routers how to forward traffic



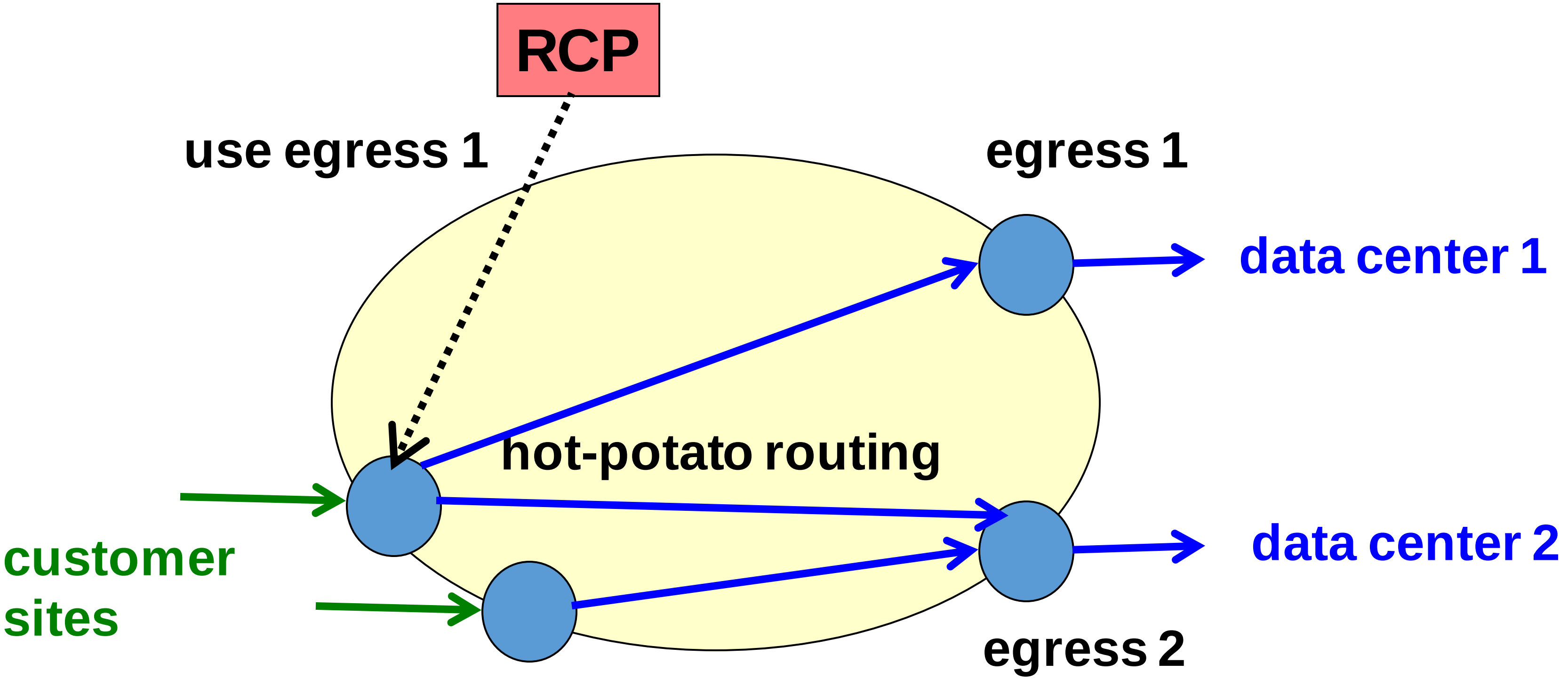
Example: Maintenance Dry-Out

- Planned maintenance on an edge router
  - Drain traffic off of an edge router
  - Before bringing it down for maintenance



Example: Egress Selection

- Customer-controlled egress selection
  - Multiple ways to reach the same destination
  - Giving customers control over the decision



RCP – The big "BUT"

- RCP still uses BGP, a single routing protocol
  - This is not what we need
  - However, we can learn from it!

The Legacy of the Separation

- Recall the two innovations:
  - Open interface between the control and data plane (e.g., ForCES [1])
  - *Logically* centralized control of the network(e.g., RCP [2])

The concepts behind SDN are not really new!

(we see both contributions in today's SDN)

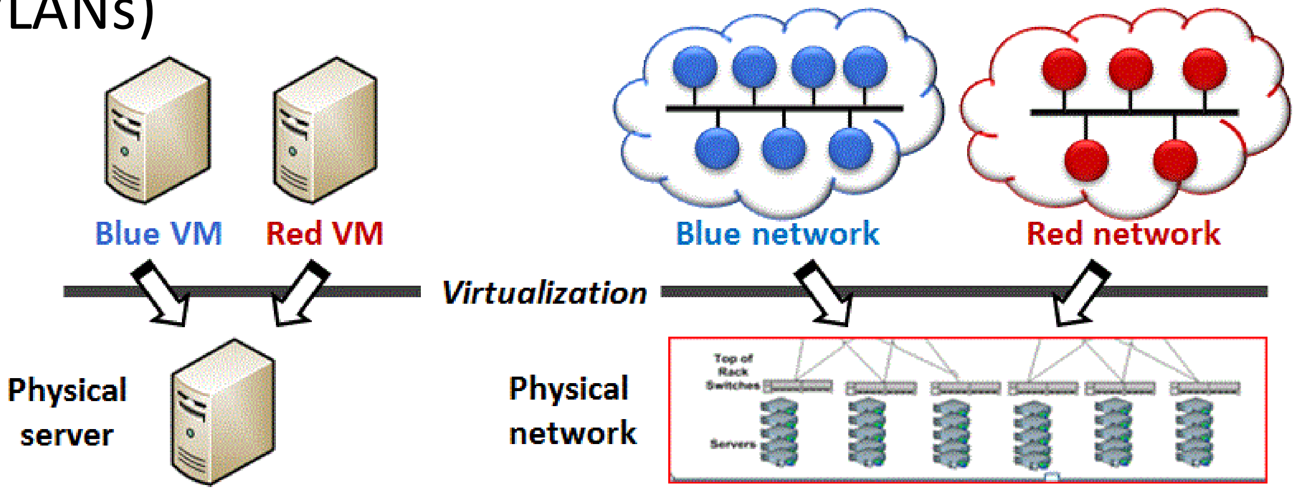
[1] Yang, et al. "*Forwarding and control element separation (ForCES) framework*." RFC 3746, April, 2004.
[2] Caesar, et al. *"Design and implementation of a routing control platform."* Proceedings of Usenix NSDI, 2005.

A brief history of programmable networks:

Network virtualization

Network Virtualization?

- Abstraction of a network that is decoupled from theunderlying physical network (e.g., VLANs)



Server virtualization

- Run multiple virtual servers on a physical server
- Each VM has illusion it is running as a physical server

Network virtualization

- Run multiple virtual networks on a physical network
- Each virtual network has illusion it is running as a physical network

Microsoft Technet.: https://gallery.technet.microsoft.com/scriptcenter/Simple-Hyper-V-Network-d3efb3b8

Network Virtualization

First steps:

- Overlay networks as virtual networks on top of legacy technology
  - own control protocol, encapsulation over legacy network (tunneling)
  - MBone [1] (for multicast), 6Bone (for IPv6)

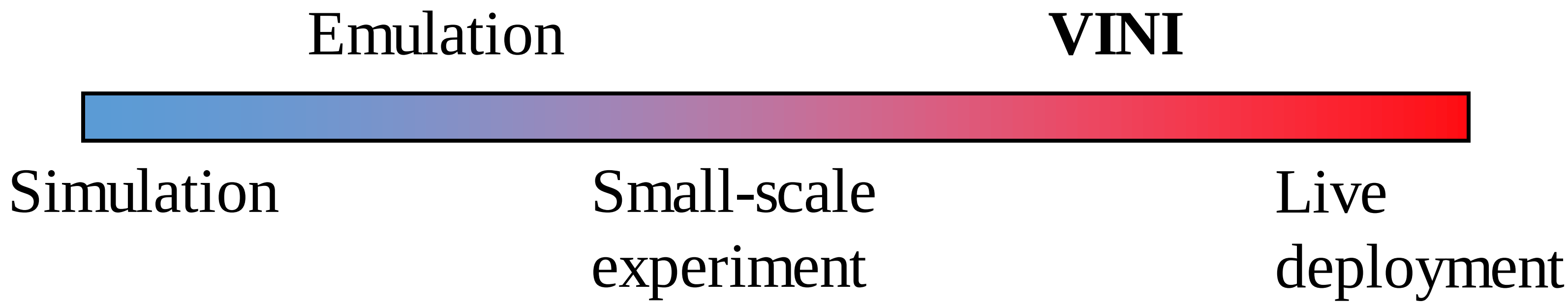
In contrast to active networks, overlay networks do not require any support from network equipment

Later:

Virtual networks inside the underlying network (e.g., VINI [2])

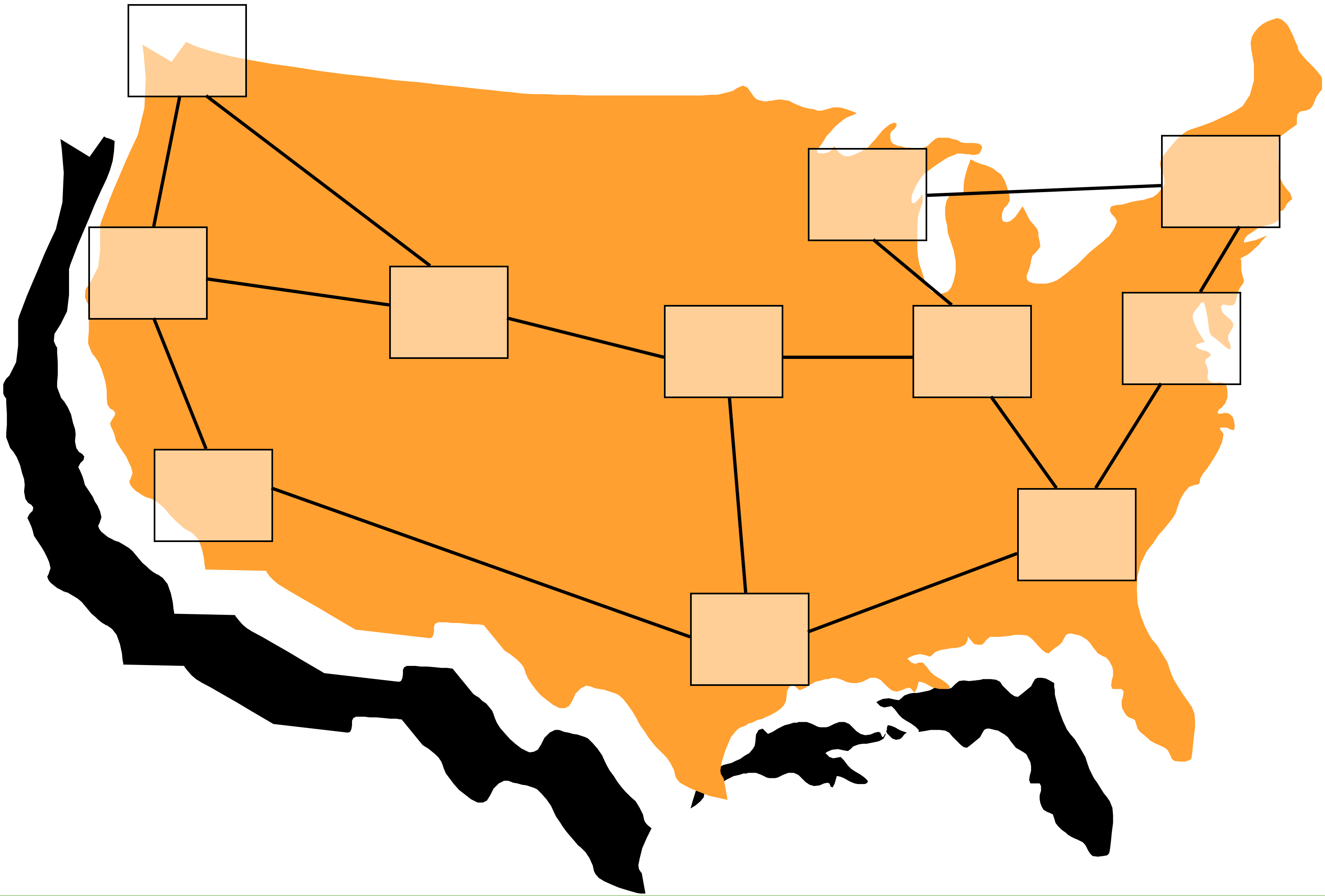
[1] http://tools.ietf.org/wg/mboned/
[2] Bavier, et al. *"In VINI veritas: realistic and controlled network experimentation."* ACM CCR. Vol. 36. No. 4. ACM, 2006.

How to Validate an Idea?

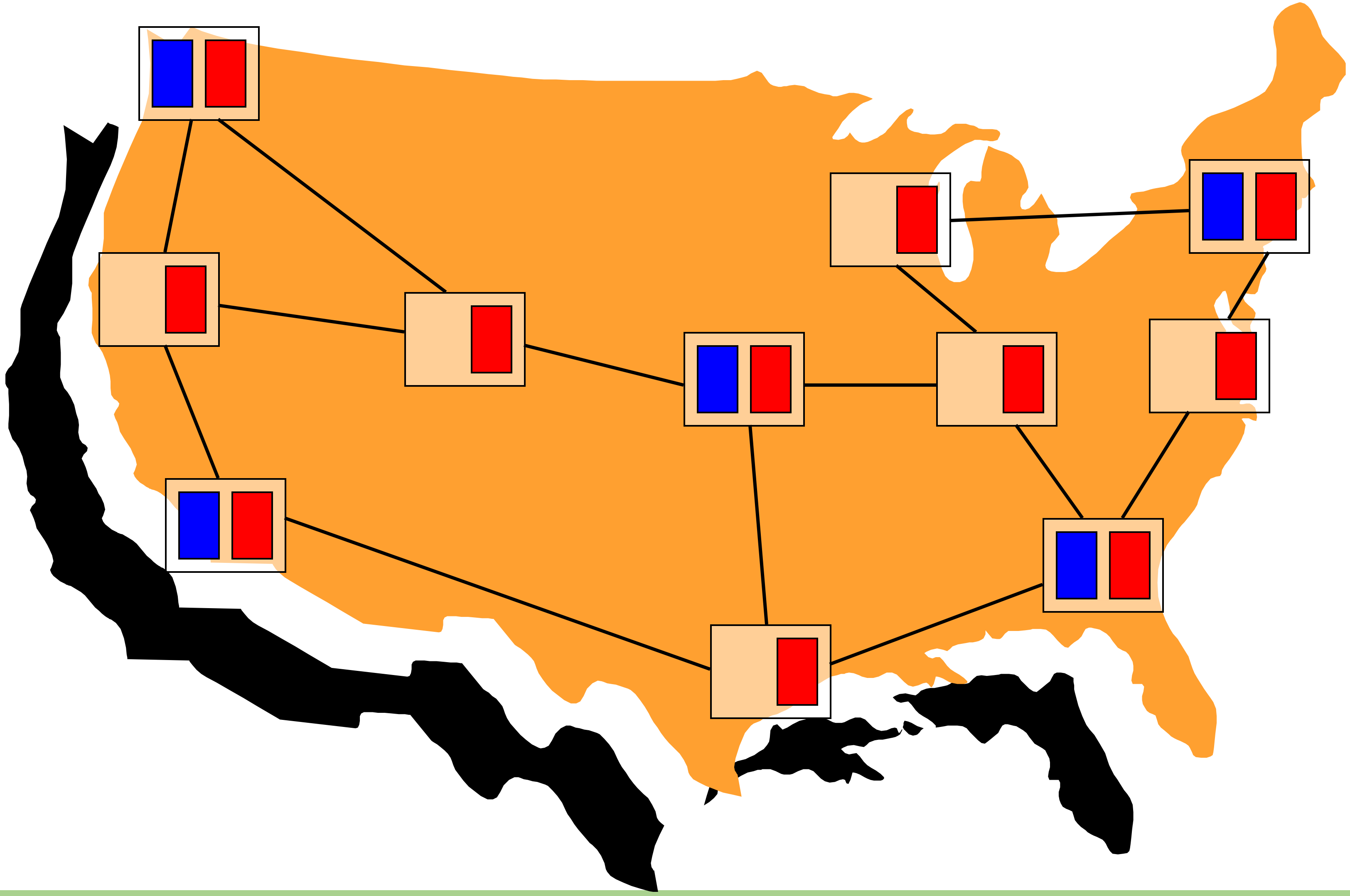


- Fixed infrastructure, shared among many experiments
- Runs real routing software
- Exposes realistic network conditions
- Gives control over network events
- Carries traffic on behalf of real users

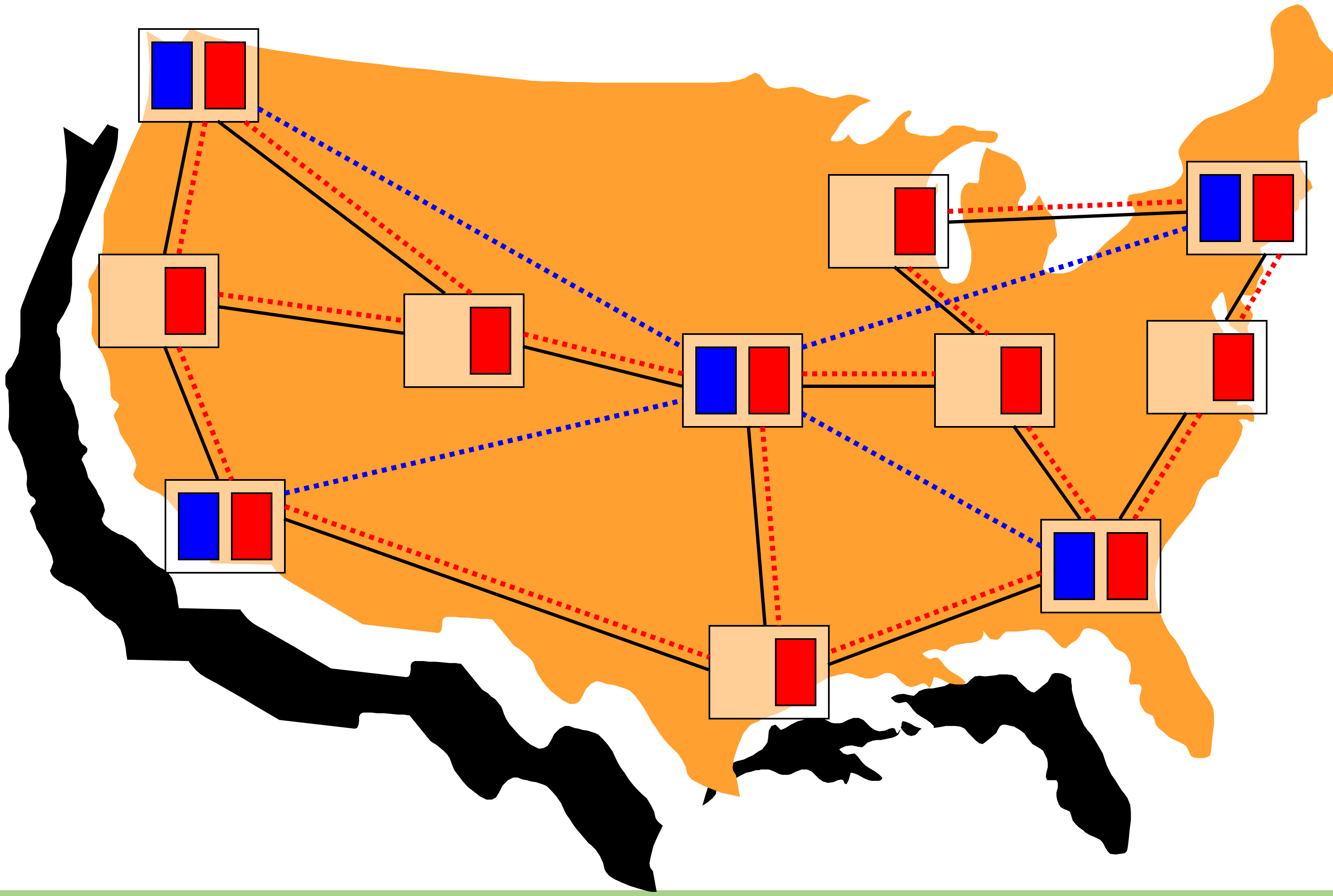
Fixed Infrastructure



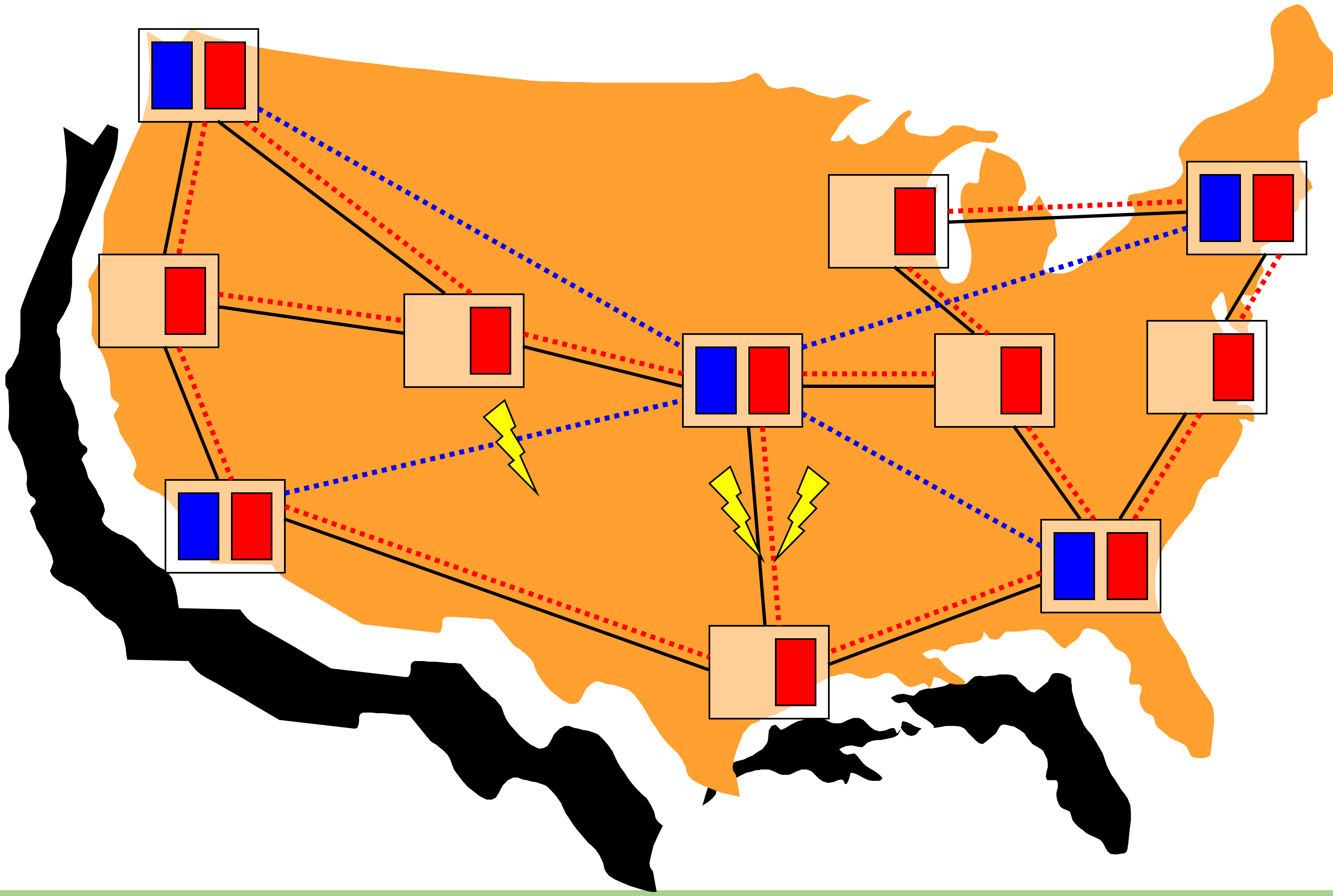
Shared Infrastructure



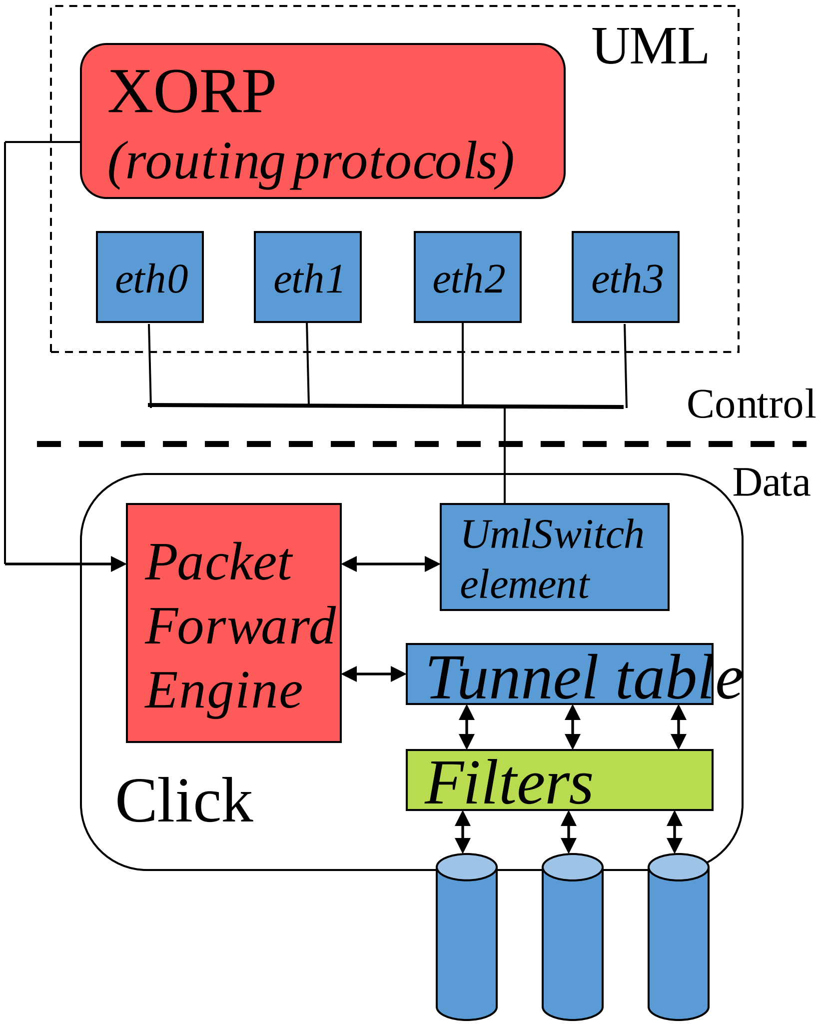
Flexible Topology



Network Events



VINI: Control/Data Plane Separation



- Filters
  - “Fail a link” by blocking packets at tunnel

The Legacy of Network Virtualization

- Three main ideas
  - Separate service from infrastructure
  - Have multiple controllers (virtual networks) for the same switch
  - Logical network topologies

The concepts behind SDN are not really new!

(we see these contributions in today‘s SDN)

Control and Data Plane Separation – What *is* SDN actually?

SDN: Control and Data Plane Separation

Control Plane

logic for controlling the forwarding elements

routing protocols (e.g., BGP, OSPF), middlebox (e.g., firewall) configuration, etc.

Data Plane

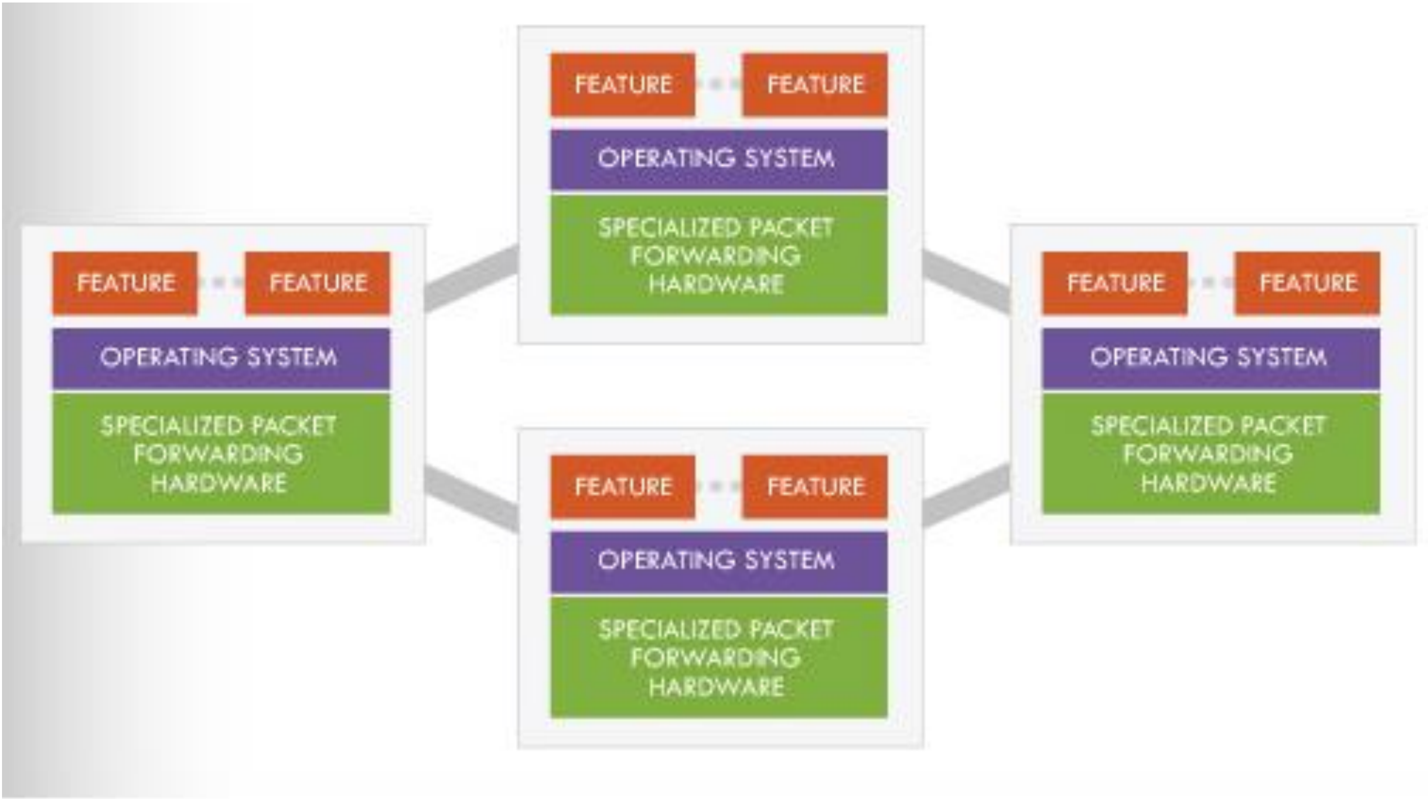
forward data based on rules set by the control logic

IP forwarding, layer 2 switching, etc.

In traditional networks, routers implement both

Why separate?

Currently, routers implement *both:*



What do we gain from separating control and data plane?

Key to Internet Success: Layers

Applications

...built on...

Reliable (or unreliable) transport

...built on...

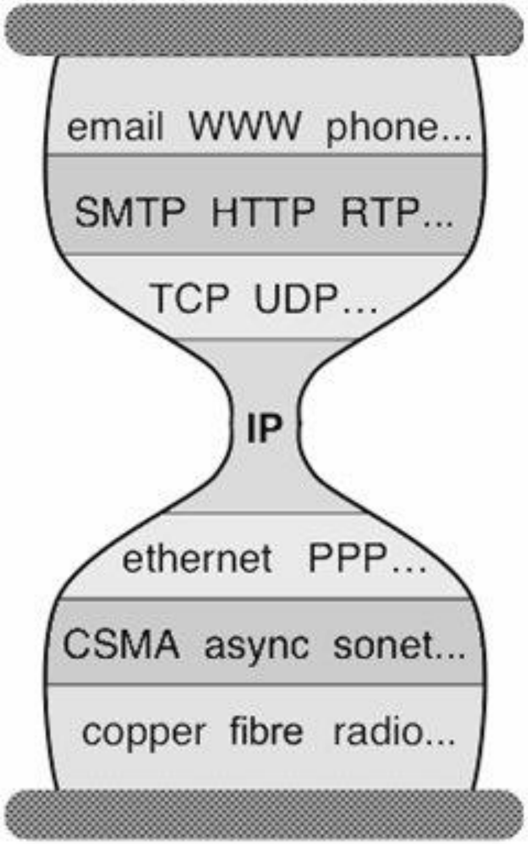
Best-effort global packet delivery

...built on...

Best-effort local packet delivery

...built on...

Physical transfer of bits



Why is Layering so Important?

- It provides ***abstraction:*** decomposed delivery into fundamental components
- Independent but compatible innovation at each layer
- A practical success (it still works!)

The Problem in Computer Networks

- Layers only deal with the **data plane**
- We have no powerful ***control plane*** ***abstractions!***

Control Plane Abstractions

?

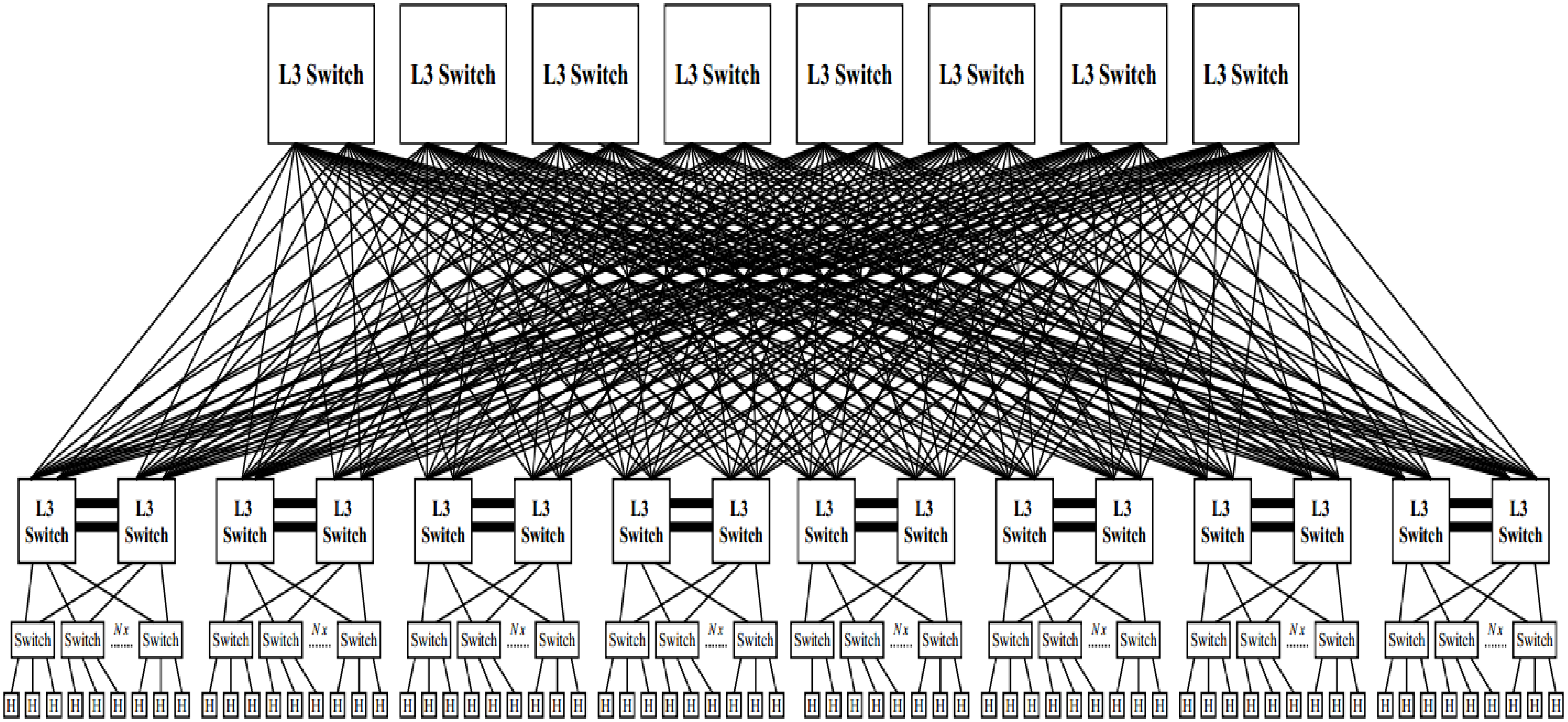
The Problem in Computer Networks

- Many different control plane mechanisms
- **Designed from scratch for specific goal**
- Variety of implementations
  - **Globally distributed:** routing algorithms
  - **Manual/scripted configuration:** ACLs, VLANs
  - **Centralized computation:** Traffic engineering
- Network control plane is a complicated mess!

The Problem in Computer Networks

- Complexity has increased to "unmanageable" levels
- Consider datacenters:
  - 100000s machines, 10000s switches
  - 1000s of customers
    - Each with their own logical networks: ACLs, VLANs, etc
- Way beyond what we can handle
  - Leads to brittle, ossified configurations
  - Probably inefficient too

Example: Datacenter Networks



Example: Datacenter Networks

- Complexity has increased to “unmanageable” levels

- **20k server cluster ~= 16k internal links**
  - This means upto **1024** distinct links between a pair of hosts
  - How do you troubleshoot this (for packetloss, etc)?
    - # of links to test = 1024/2 = 512
    - 30 seconds/test
    - **256 man-minutes for most-basic troubleshooting!**

The Problem is not only Complexity

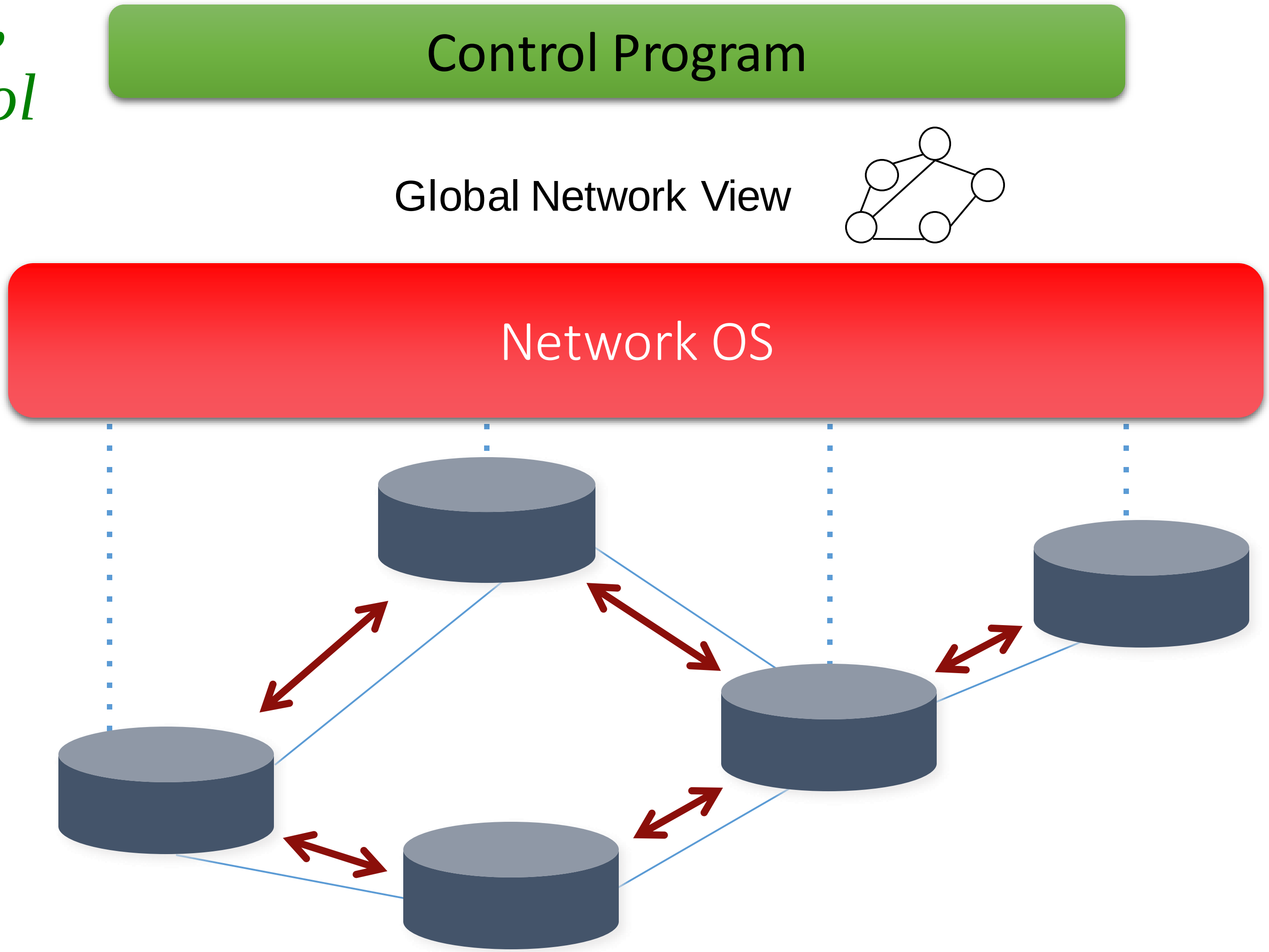
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features

Enter SDN

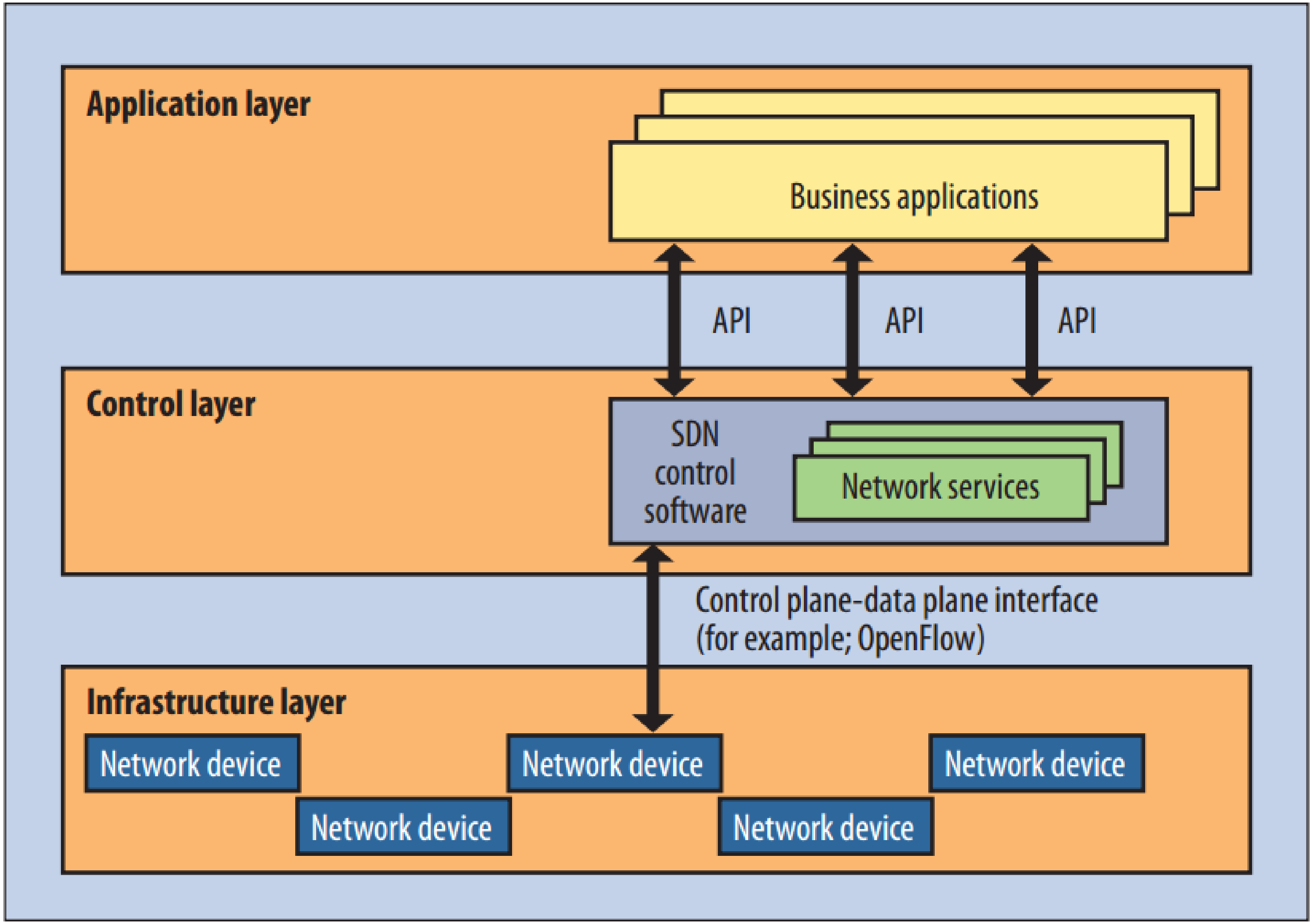
- Today, routers implement both planes
  - They forward packets
  - And run the control plane software
- SDN networks
  - Data plane implemented by switches
    - Switches act on local forwarding state
  - Control plane implemented by controllers
    - All forwarding state computed by SDN platform
  - Open protocols!
- This is a technical change, with broad implications

Enter SDN

e.g. routing, access control



Another View



Anology

- You are lost in a city and are trying to reach a destination

- Todays networks: ask other people you meet to obtain information (routing protocols)

- SDN: pull out your cellphone and start Google maps – it will calculate the route for you

Changes

- Less vendor lock-in
  - Can buy HW/SW from different vendors
    - Anology: you can use OSM or even Apple Maps ;)
- Changes are easier
  - Can test components separately
    - *HW has to forward*
    - *Can simulate controller*
    - *Can do verification on logical policy*
  - Can change topology and policy independently
  - Greater rate of innovation

Challenges of the Separation

- Talked a lot about the opportunities

- What about the challenges?

Practical Challenges

- Scalability
  - Decision elements responsible for many routers
- Response time
  - Delays between decision elements and routers
- Reliability
  - Surviving failures of decision elements and routers
- Consistency
  - Ensuring multiple decision elements behave consistently
- Security
  - Network vulnerable to attacks on decision elements
- Interoperability
  - Legacy routers and neighboring domains

Example - Scalability

- Take routing: the controller has to make routing decisions for a lot of routers
  - Potentially 1000s
  - Topology maintenance?

- Also has to store these routes
  - a lot of routing tables

- Single controller node for this task?
  - OSPF: distributed

Current Status of SDN

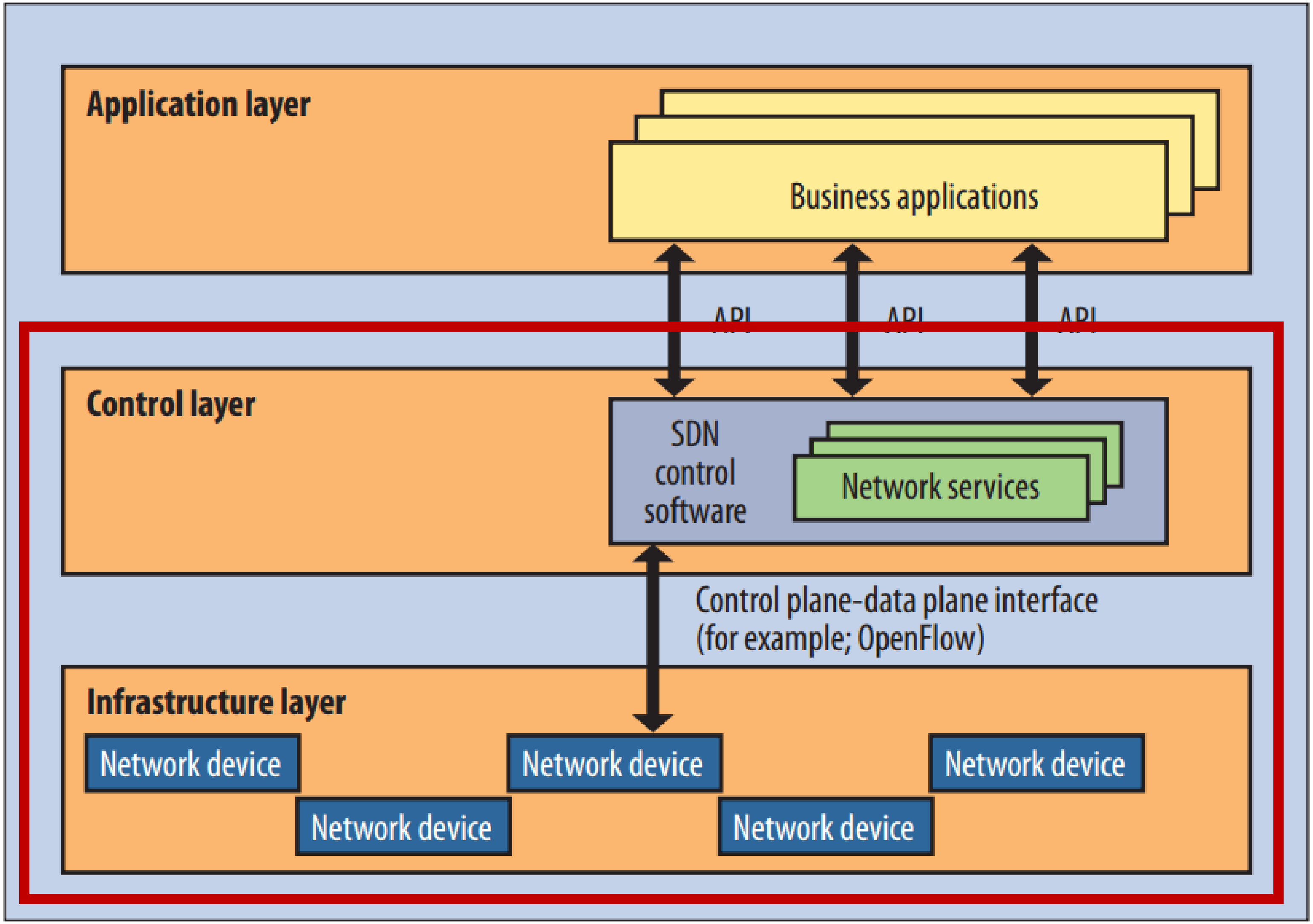
- SDN widely accepted as "**future of networking**"
  - ~1000 engineers at latest Open Networking Summit
  - *More acceptance in industry than in academia*
- Insane level of SDN hype, and still:
  - SDN doesn't work miracles, merely makes things easier

Current Status of SDN

- Most innovations in southbound interface, controllers, northbound interface, and applications
  - OpenFlow (as ONE example of the sb interface)
  - NOX, POX, ONOS, etc.
  - Pyretic, Frenetic, etc.

- But: also changes in network devices
  - Most global players offer SDN switches now

Up Next


Application layer
Business applications
API
API
API
Control layer
SDN control software
Network services
Control plane-data plane interface (for example; OpenFlow)
Infrastructure layer
Network device
Network device
Network device
Network device
Network device