Software Defined Networking

Based on slides from

Jennifer Rexford (Princeton)

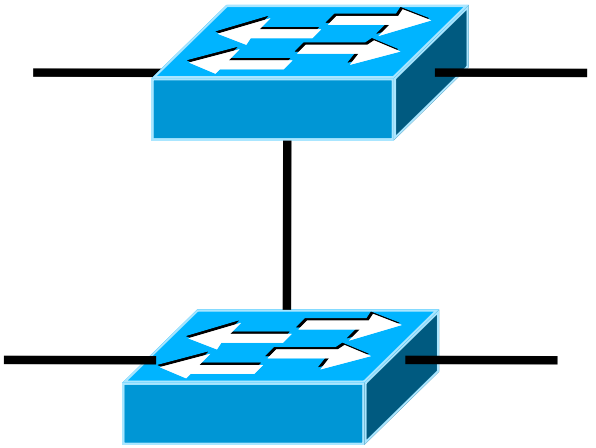
and Nick McKeown (Stanford)

The Internet: A Remarkable Story

- Tremendous success
  - From research experiment to global infrastructure
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Hosts: arbitrary applications
- Enables innovation in applications
  - Web, P2P, VoIP, social networks, virtual worlds
- But, change is easy only at the edge… ☹

Inside the ‘Net: A Different Story…

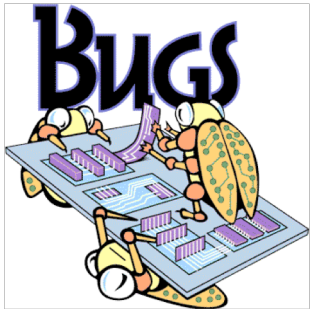
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features



Impacts performance, security, reliability, cost…

Networks are Hard to Manage

- Operating a network is expensive
  - More than half the cost of a network
  - Yet, operator error causes most outages
- Buggy software in the equipment
  - Routers with 20+ million lines of code
  - Cascading failures, vulnerabilities, etc.
- The network is "in the way"
  - Especially a problem in data centers
  - … and home networks



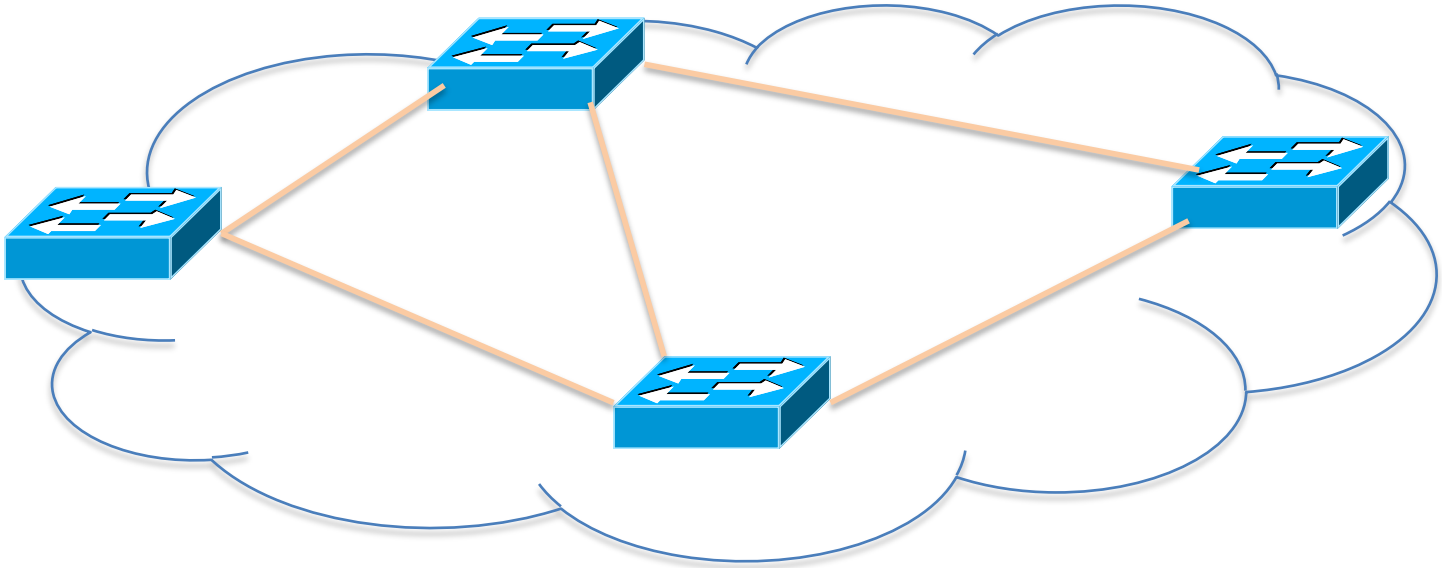
New Foundation for Networking

- A domain, not (yet?) a discipline
  - Alphabet soup of protocols
  - Header formats, bit twiddling
  - Preoccupation with artifacts
- From practice, to principles
  - Intellectual foundation for networking
  - Identify the key abstractions
  - … and support them efficiently
- To build networks worthy of society's trust

Rethinking the “Division of Labor”

Traditional Computer Networks

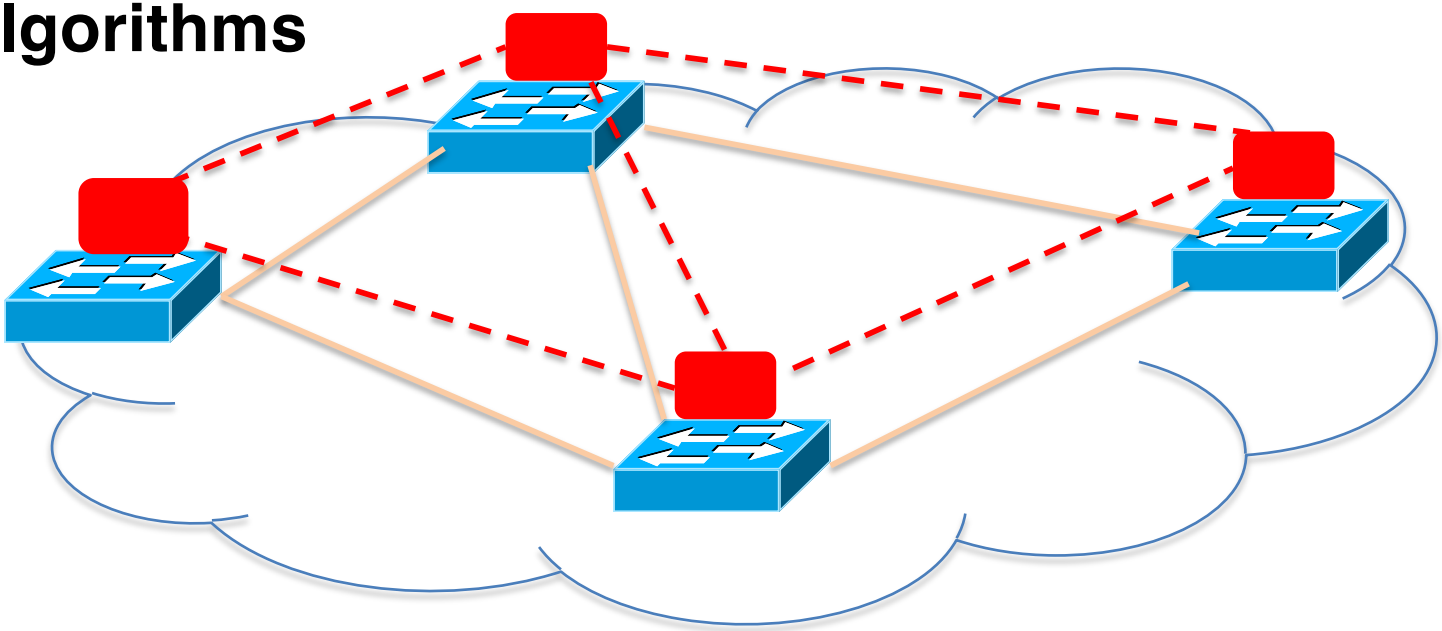
Data plane:
Packet streaming



Forward, filter, buffer, mark, rate-limit, and measure packets

Traditional Computer Networks

Control plane:
Distributed algorithms

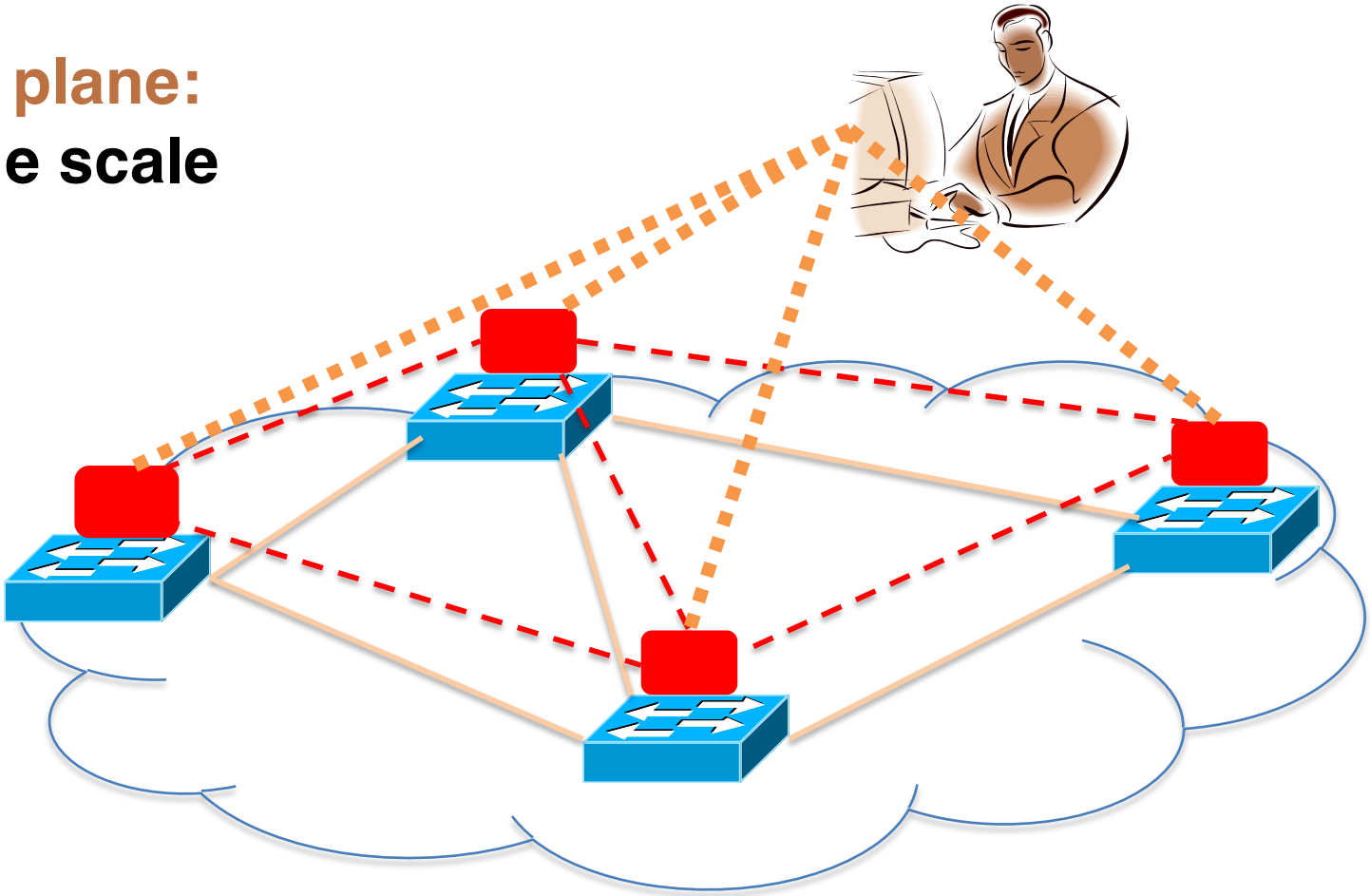


Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks

Management plane:

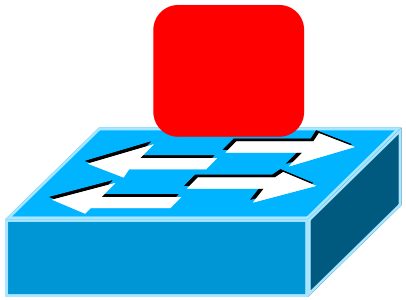
Human time scale



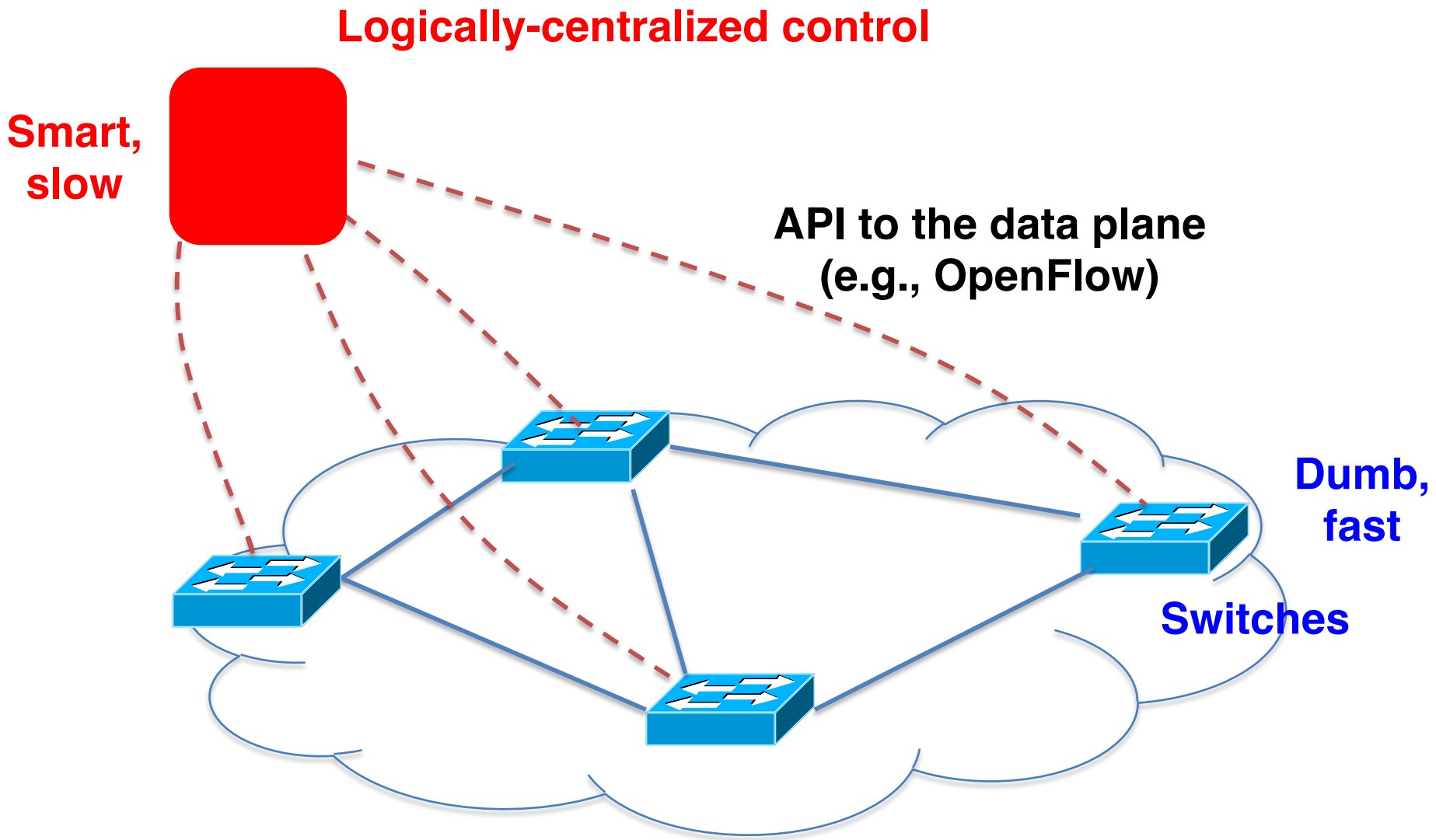
Collect measurements and configure the equipment

Death to the Control Plane!

- Simpler management
  - No need to "invert" control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
- Easier interoperability
  - Compatibility only in "wire" protocols
- Simpler, cheaper equipment
  - Minimal software



Software Defined Networking (SDN)



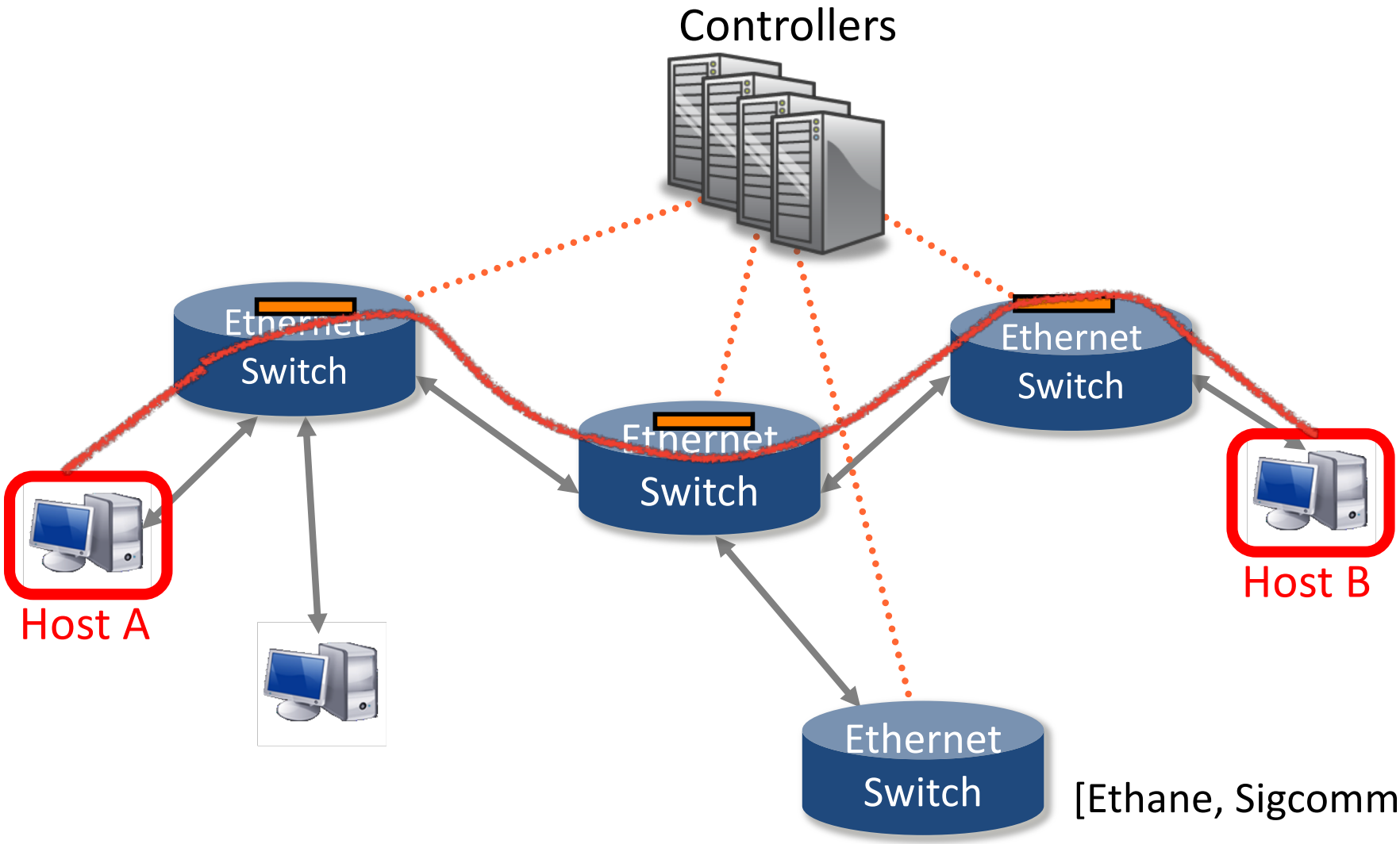
How difficult is it to define all network operations in software, outside the datapath?

2006

35,000 users
10,000 new flows/sec
137 network policies

2,000 switches
2,000 switch CPUs

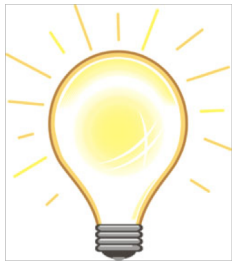
Extreme: What if software decides whether to accept each flow, and how to route it?



[Ethane, Sigcomm '07]

How many $400 servers do we need for 35,000 users?

Answer: less than one



If we can define network operation outside the datapath, then eventually we will.

With replication for
fault-tolerance and performance scaling.

It was already starting…

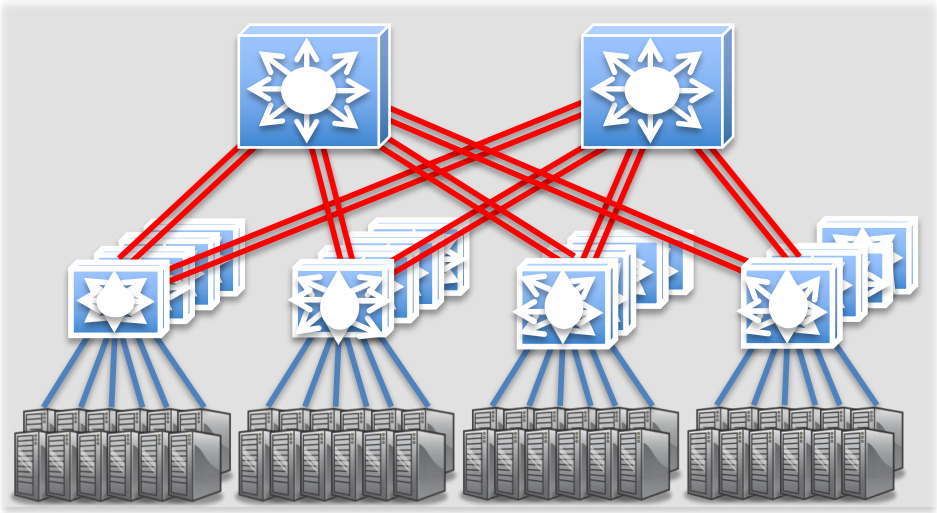
- Public WANs: Route reflectors decide routes centrally, and download to datapath
  - AT&T Backbone
- Enterprise WiFi: CAPWAP and Meraki
- Cable TV: Docsis

A bigger problem emerging

Grumbling Public Internet Providers

- Global IP traffic growing 40-50% per year
- End-customer monthly bill remains unchanged
- Therefore, CAPEX and OPEX need to reduce 40-50% per Gb/s per year
- But in practice, reduces by ~20% per year

Grumbling Data Center Owners



Cost

200,000 servers

Fanout of 20 ➔ 10,000 switches

$5k vendor switch = $50M

$1k commodity switch = $10M

Savings in 10 data centers = $400M

Control

More flexible control

Tailor network for services

Quickly improve and innovate

By 2007, Google and Amazon starting to write their own software

What a switch or router looks like

Routing, management, mobility management, access control, VPNs, …

Feature · · · · · · Feature

Operating System

Custom Forwarding Hardware

Million of lines of source code

7000 Internet RFCs

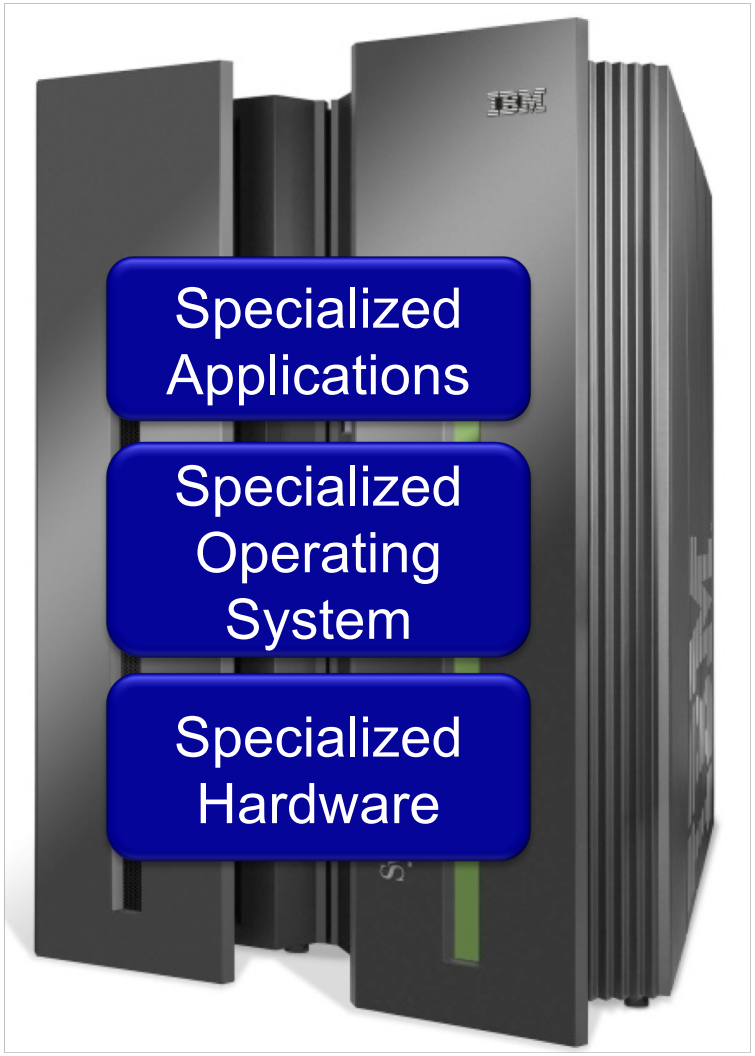
Billions of gates

Bloated

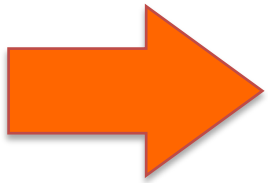
Power Hungry

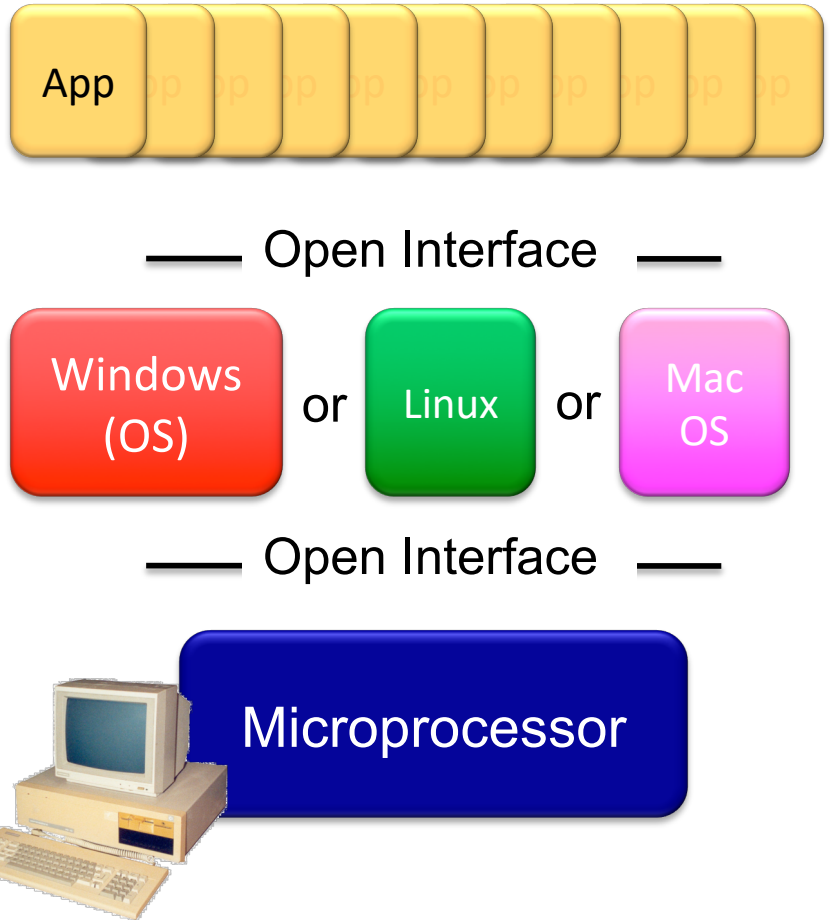
Overly complex
Mainframe mentality
Too expensive

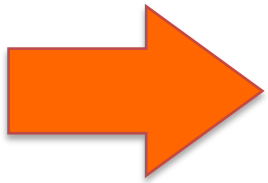
Part of an industry disruption



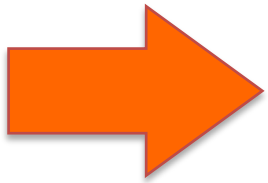
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

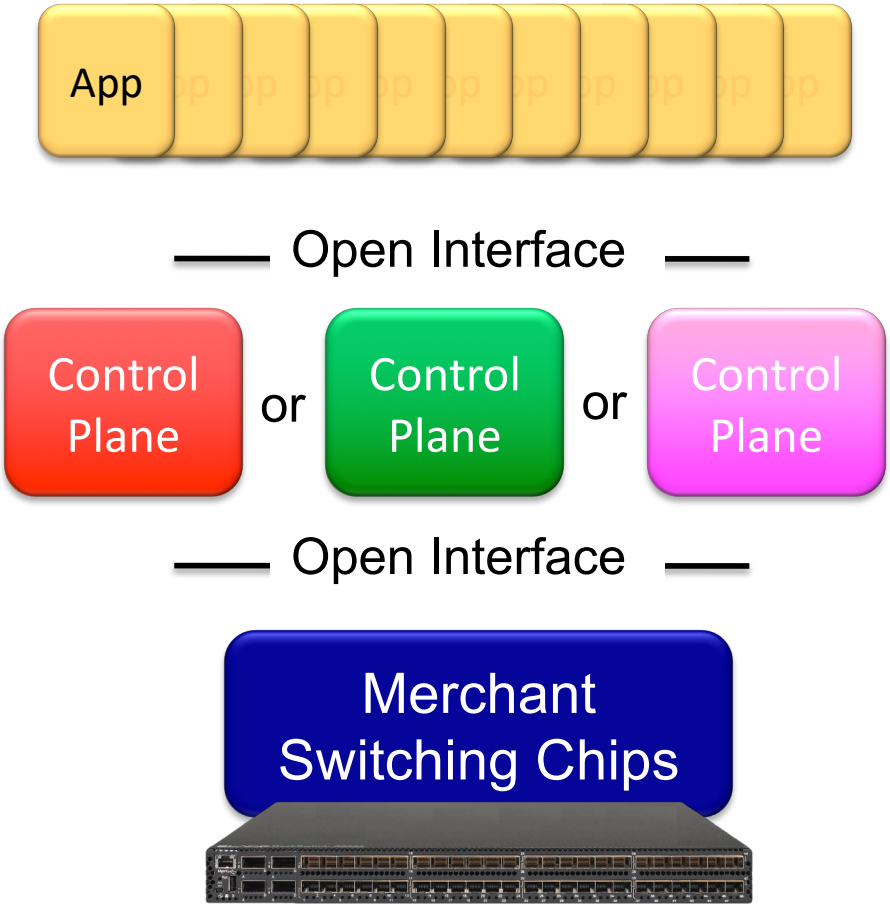




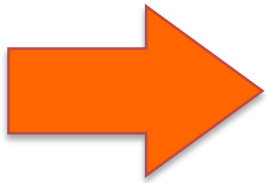


Horizontal
Open interfaces
Rapid innovation
Huge industry





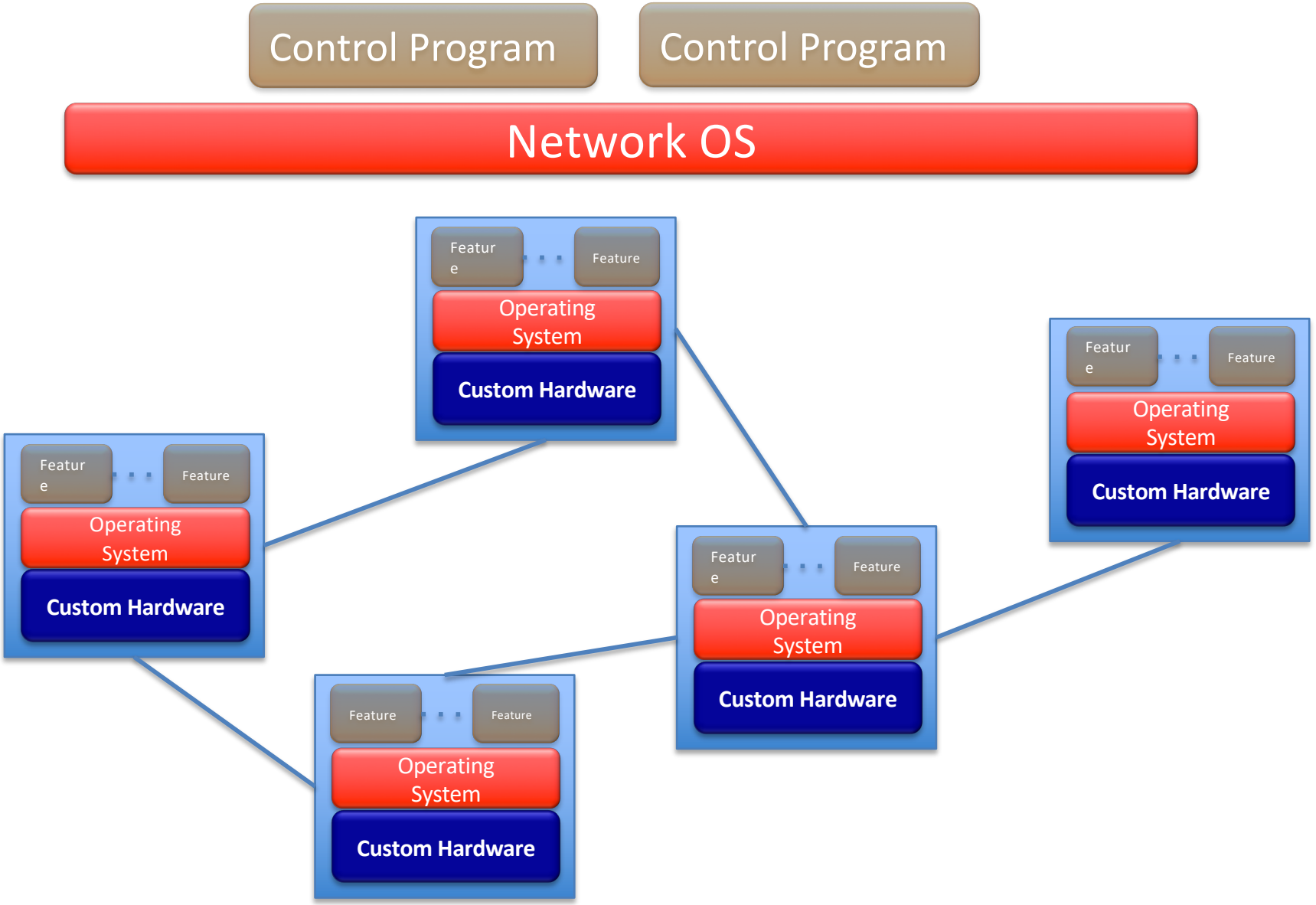
Vertically integrated
Closed, proprietary
Slow innovation



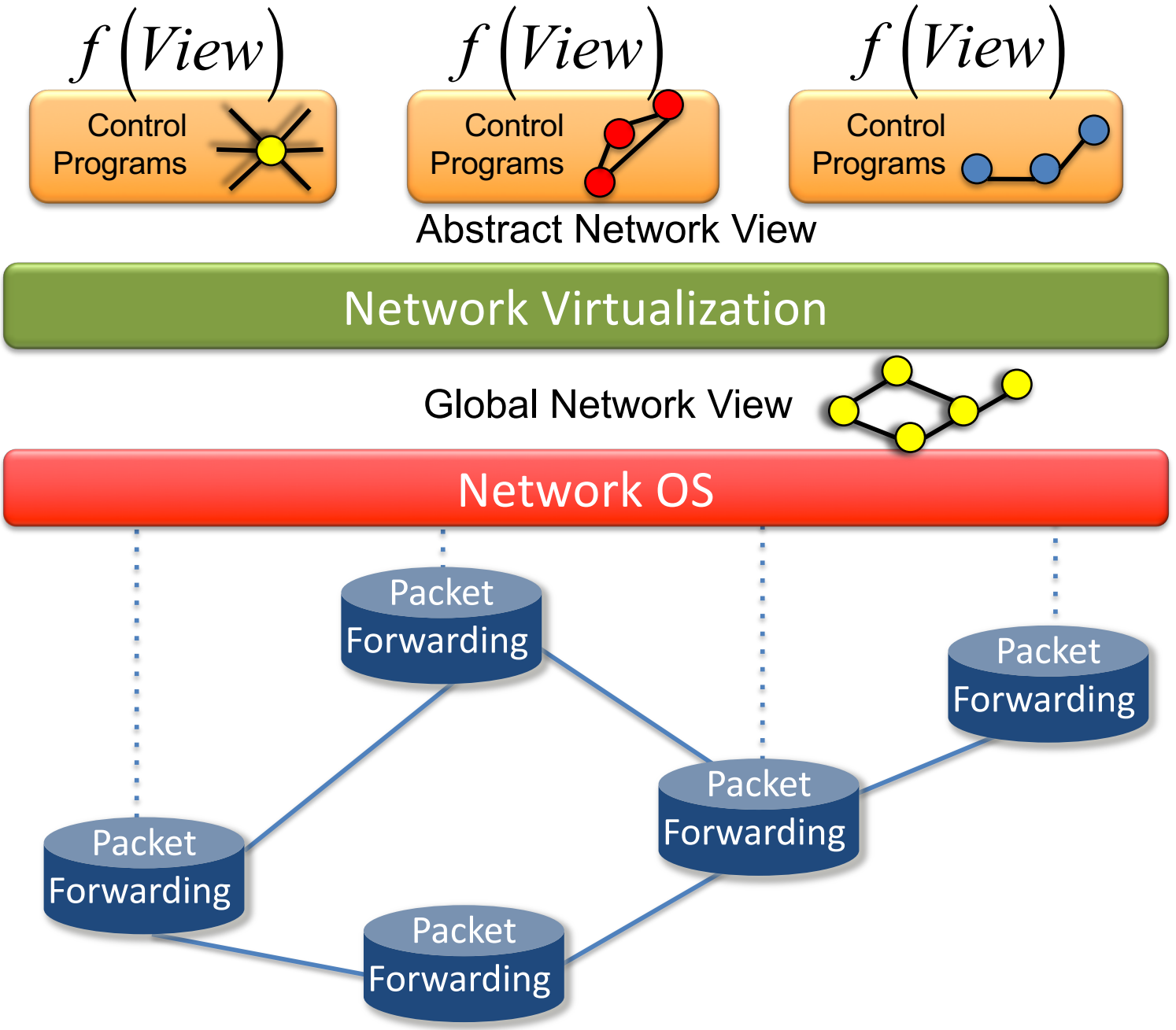
Horizontal
Open interfaces
Rapid innovation

Today’s SDN

Restructuring Networks



Software Defined Network (SDN)



Early adoption

Multi-tenant data centers:

eBay, AT&T, NTT,

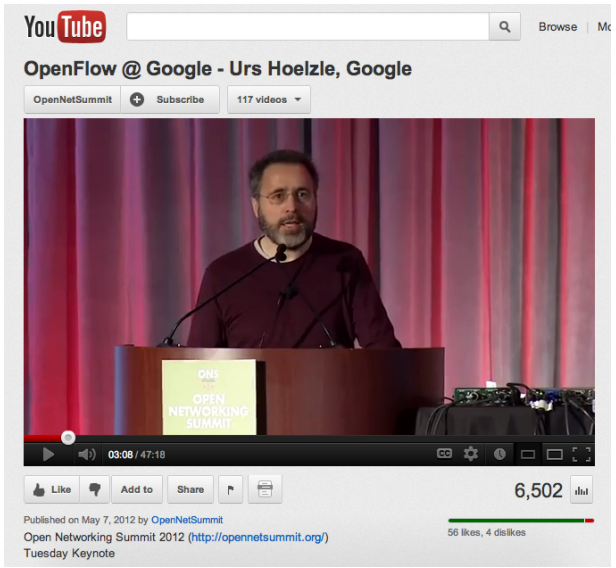
Rackspace, Fidelity, …

Large WANs

Google.

R&E WANs

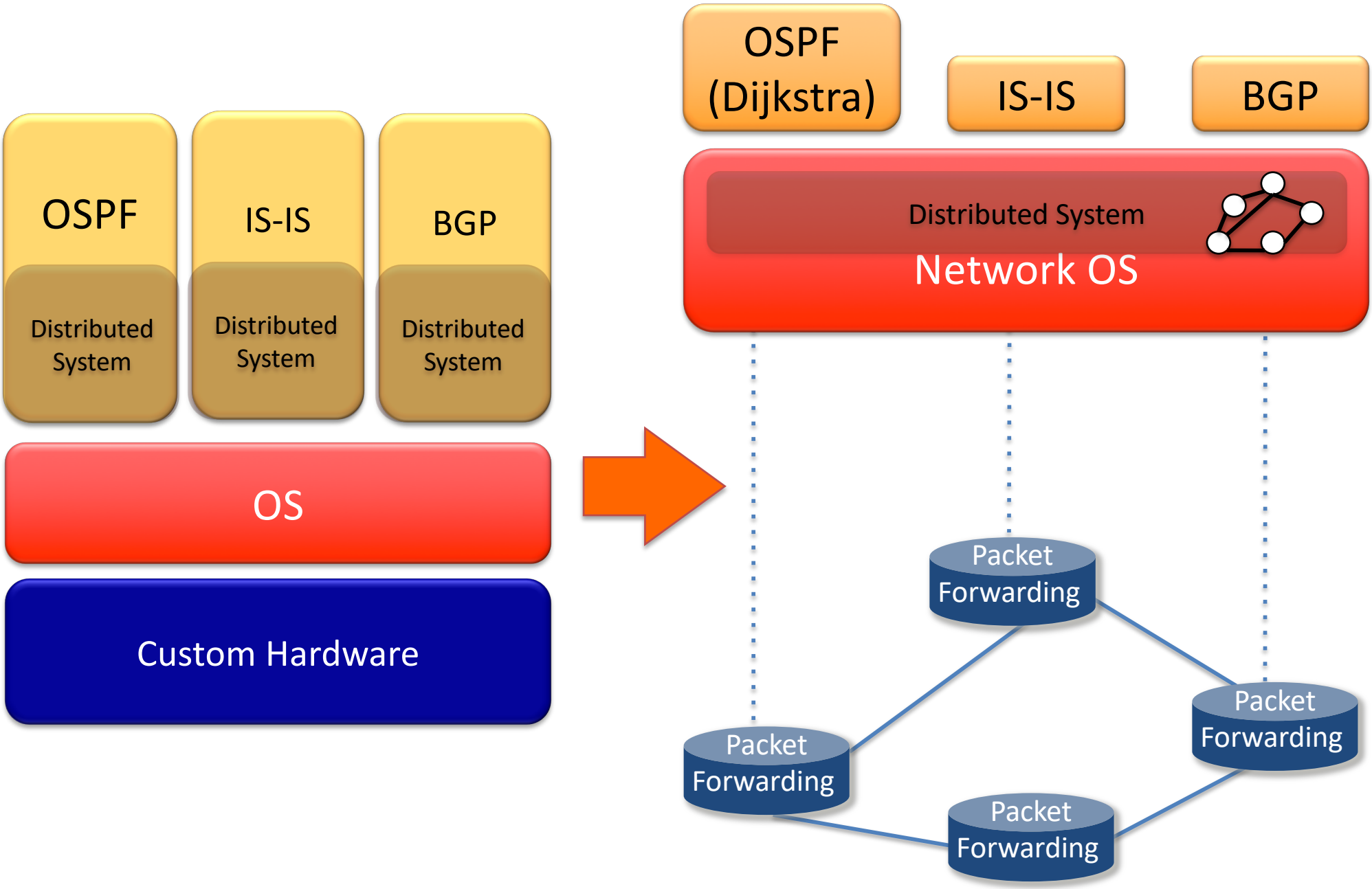
US, Brazil, Japan, (EU)



Urs Hoelzle (Google) at
Open Networking Summit 2012

Is it just refactoring code?

Not quite


OSPF
IS-IS
BGP
Distributed System
Distributed System
Distributed System
OS
Custom Hardware
OSPF (Dijkstra)
IS-IS
BGP
Distributed System
Network OS
Packet Forwarding
Packet Forwarding
Packet Forwarding
Packet Forwarding

Example: OSPF

RFC 2328: **245 pages**

Distributed System

- Builds consistent, up-to-date map of the network: **101 pages**

Dijkstra's Algorithm: **4 pages**

The consequences of well-defined abstractions

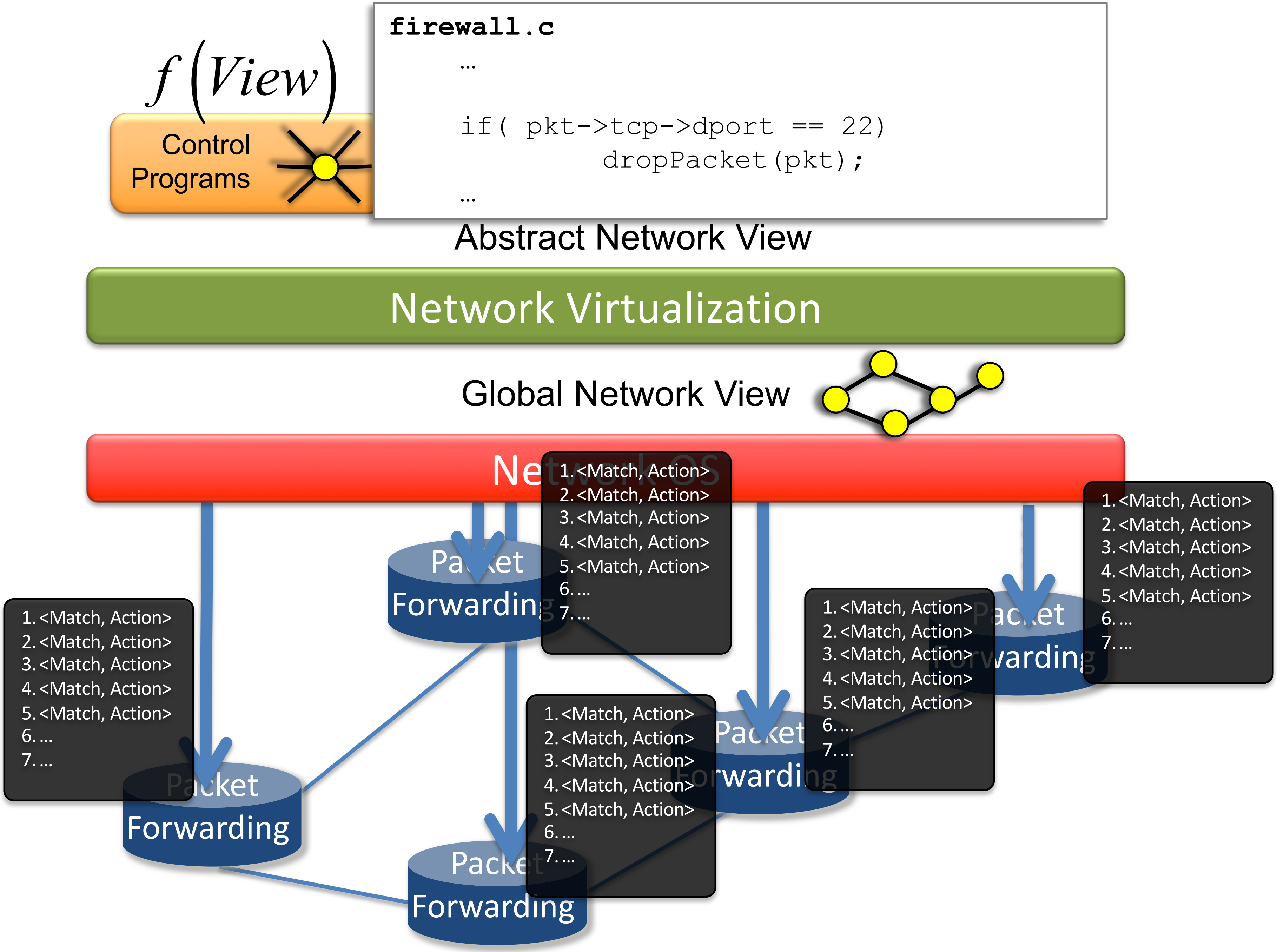
An intellectual foundation for verifying, troubleshooting and debugging networks

With SDN it seems like we should be able to:

1. Formally verify that our networks are behaving correctly.
2. Identify bugs, then systematically track down their root cause.

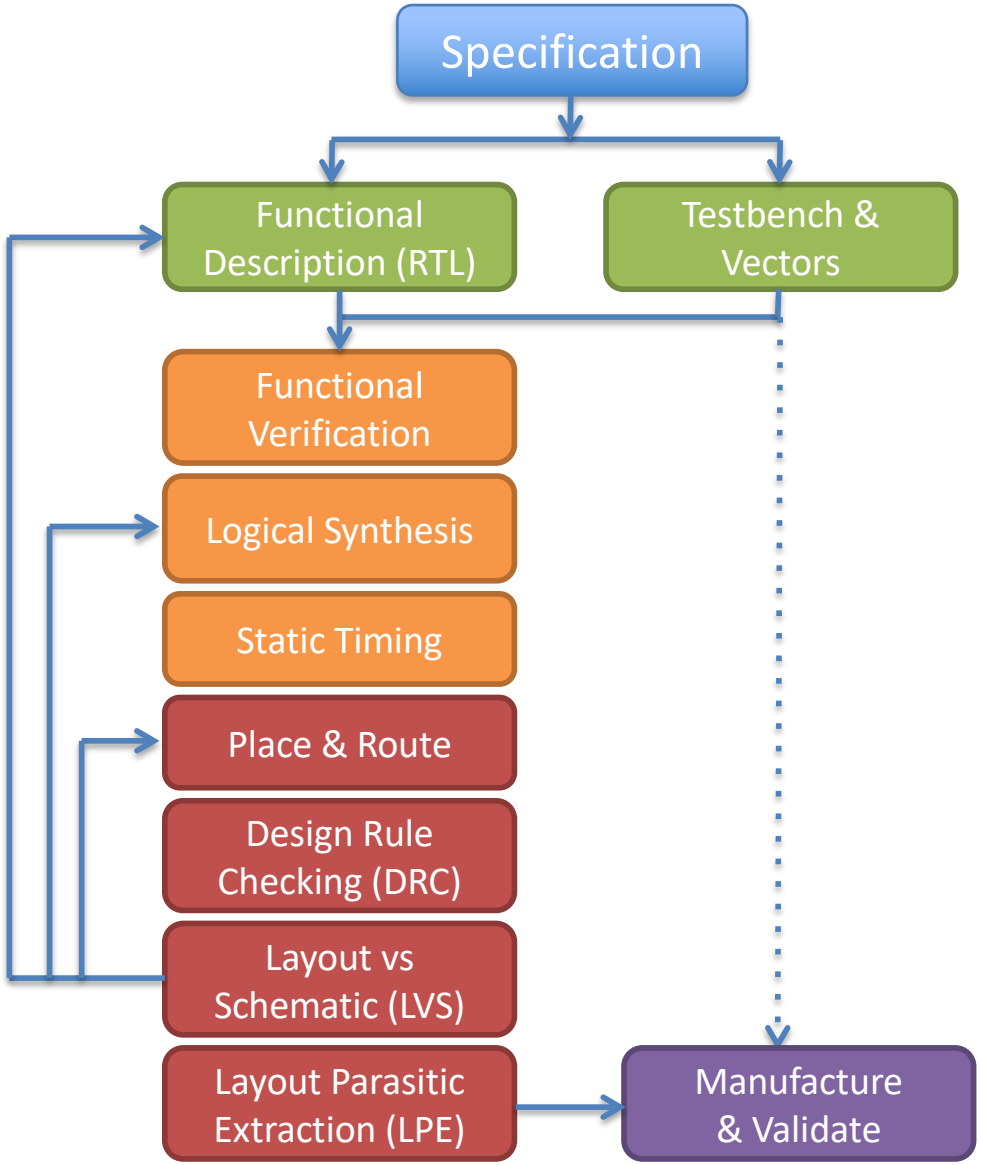
- Ensuring correctness [Frenetic][HFT][Netcore]
  Nate Foster, Andrew Ferguson, Mike Freedman, Jen Rexford, Rob Harrison, Dave Walker, ++
- Software Fault Localization [W3]
  Scott Shenker, Colin Scott, Kyriakos Zarifis, Andreas Wundsam.
- Checking behavior [NICE]
  Marco Canini, Daniele Venzano, Peter Peresini, Dejan Kostic, Jen Rexford.
- Checking Invariants [VeriFlow]
  Ahmed Khurshid, Wenxuan Zhou, Matthew Caesar, P. Brighten Godfrey
- Consistent updates
  Mark Reitblatt, Rick McGeer, ++
- Troubleshooting [OFRewind]
  Andreas Wundsam, Dan Levin, Srini Seetharaman, Anja Feldman
- …

Software Defined Network (SDN)



How do other industries do it?

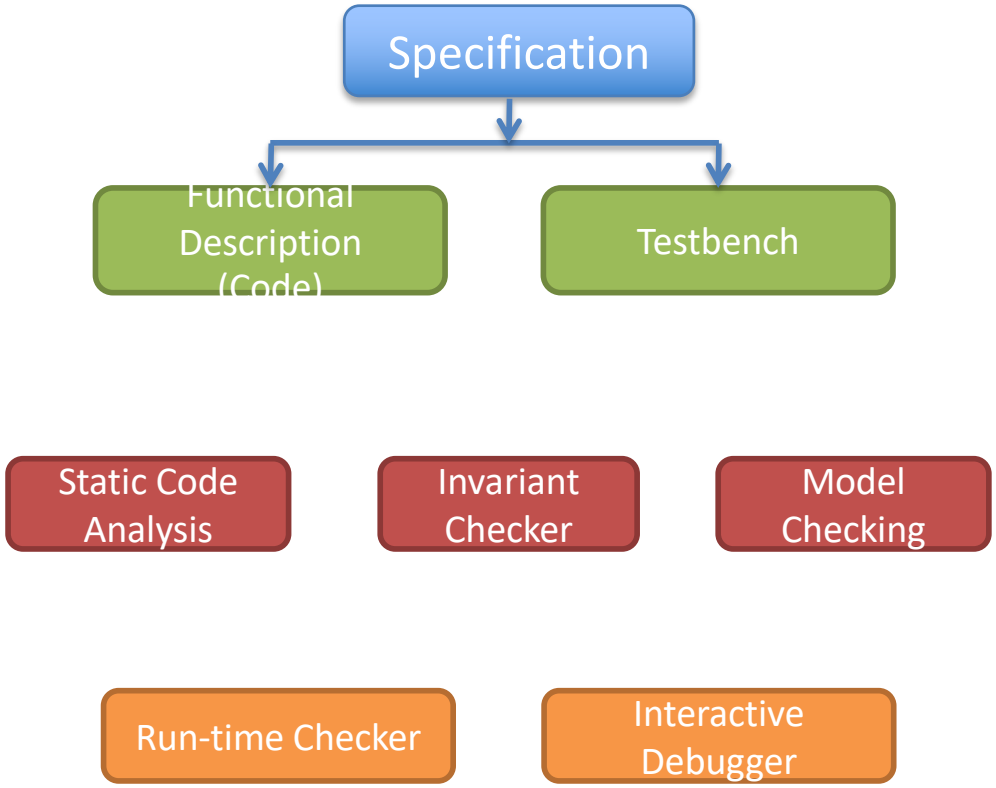
Making ASICs Work



$10B tool business supports a $250B chip industry

100s of Books
>10,000 Papers
10s of Classes

Making Software Work



$10B tool business supports a $300B S/W industry

100s of Books

>100,000 Papers

10s of Classes

Making Networks Work (Today)

`traceroute, ping, tcpdump, SNMP, Netflow`

…. er, that's about it.

Why debugging networks is hard

Complex interaction

- Between multiple protocols on a switch/router.
- Between state on different switches/routers.

Multiple uncoordinated writers of state.

Operators can't…

- Observe all state.
- Control all state.

Networks are kept working by

“Masters of Complexity”

A handful of books

Almost no papers

No classes

OpenFlow Networks

Data-Plane: Simple Packet Handling

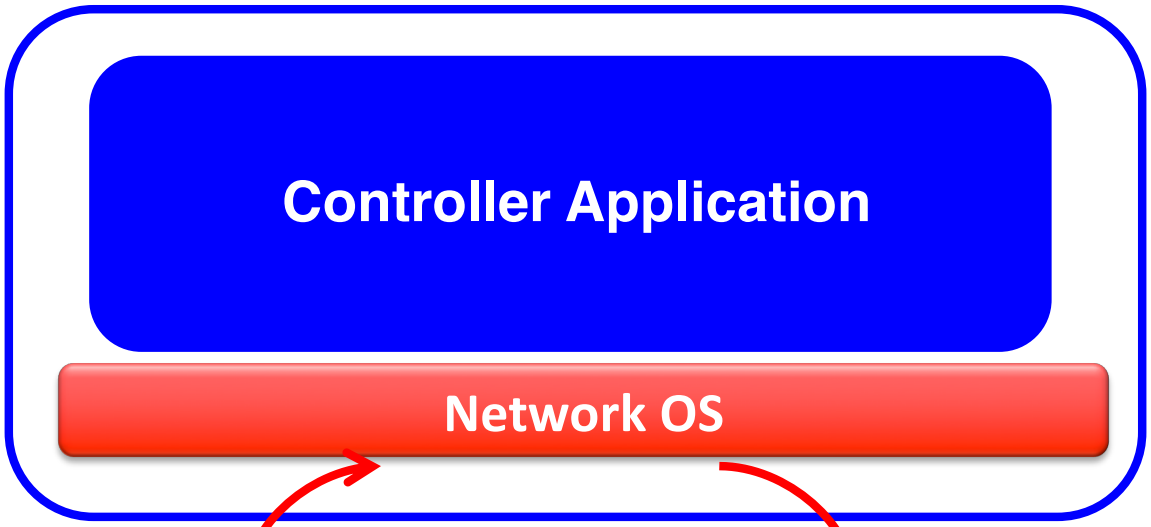
- Simple packet-handling rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

1. **src=1.2.*.*, dest=3.4.5.* → drop**
2. **src = *.*.*.*, dest=3.4.*.* → forward(2)**
3. **src=10.1.2.3, dest=*.*.*.* → send to controller**

Unifies Different Kinds of Boxes

- Router
  - Match: longest destination IP prefix
  - Action: forward out a link
- Switch
  - Match: destination MAC address
  - Action: forward or flood
- Firewall
  - Match: IP addresses and TCP/UDP port numbers
  - Action: permit or deny
- NAT
  - Match: IP address and port
  - Action: rewrite address and port

Controller: Programmability



Events from switches
Topology changes,
Traffic statistics,
Arriving packets

Commands to switches
(Un)install rules,
Query statistics,
Send packets

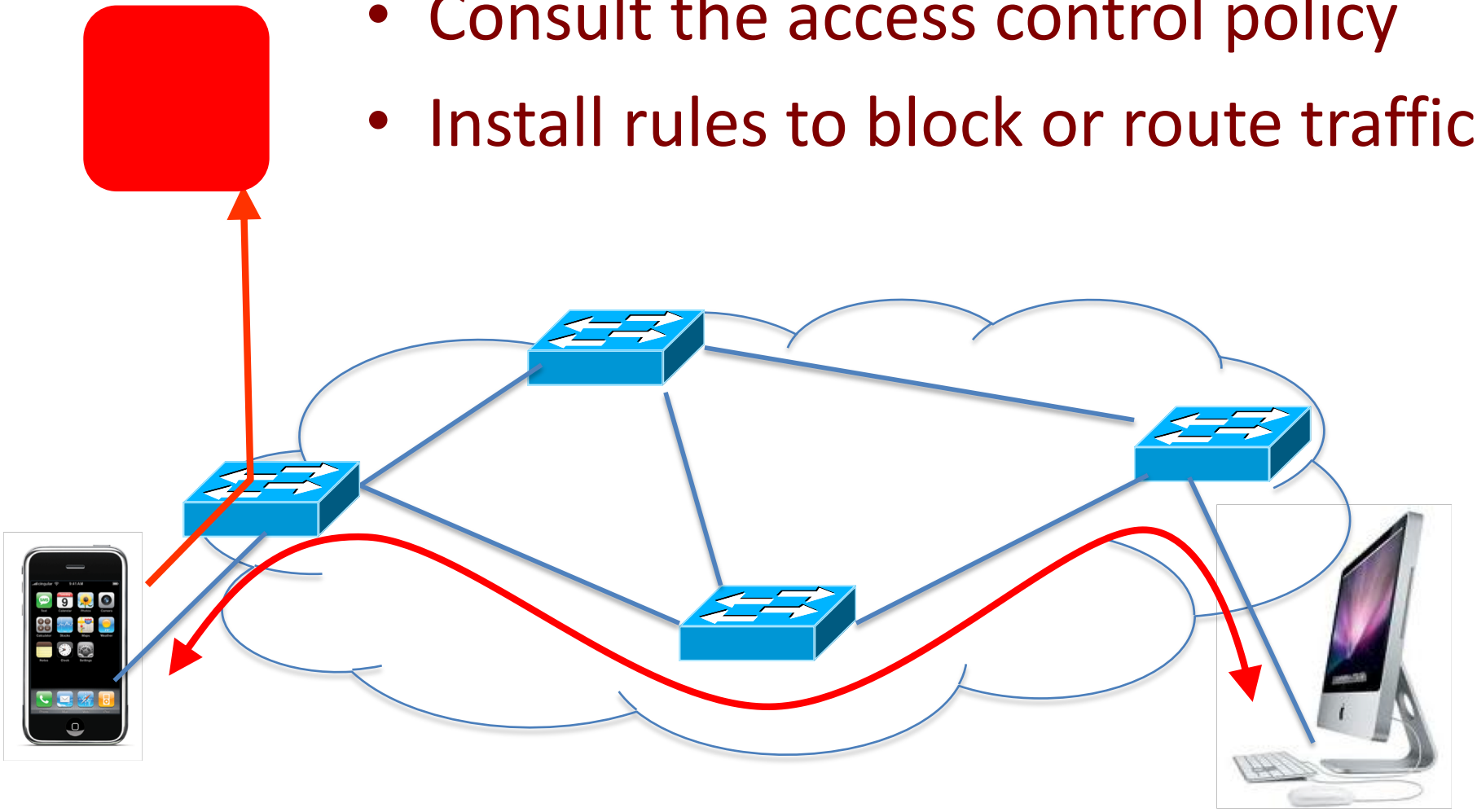
Example OpenFlow Applications

- **Dynamic access control**
- **Seamless mobility/migration**
- **Server load balancing**
- **Network virtualization**
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection

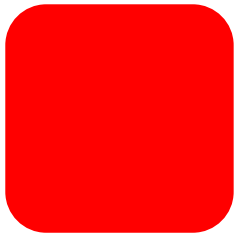
See http://www.openflow.org/videos/

E.g.: Dynamic Access Control

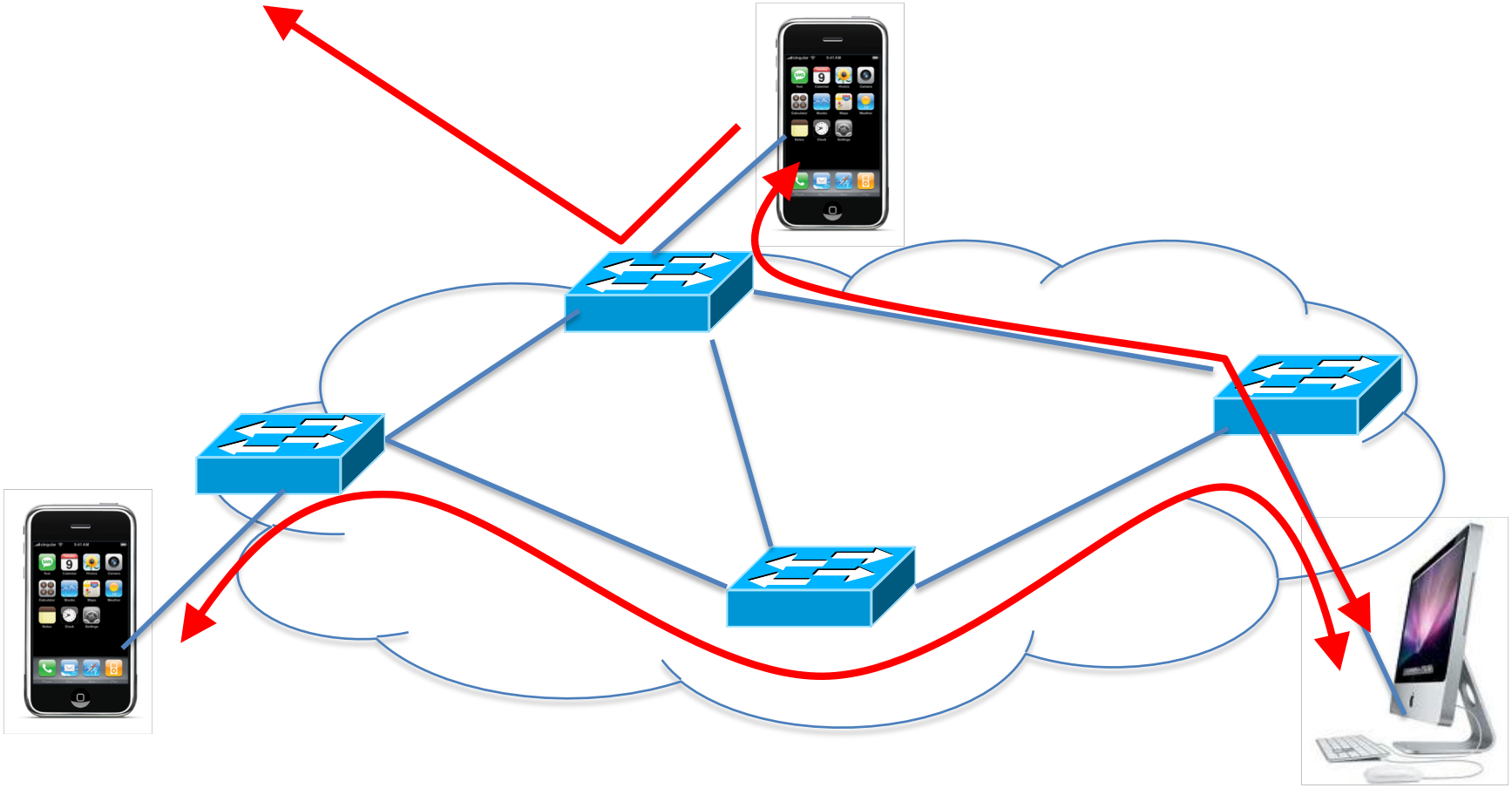
- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



E.g.: Seamless Mobility/Migration

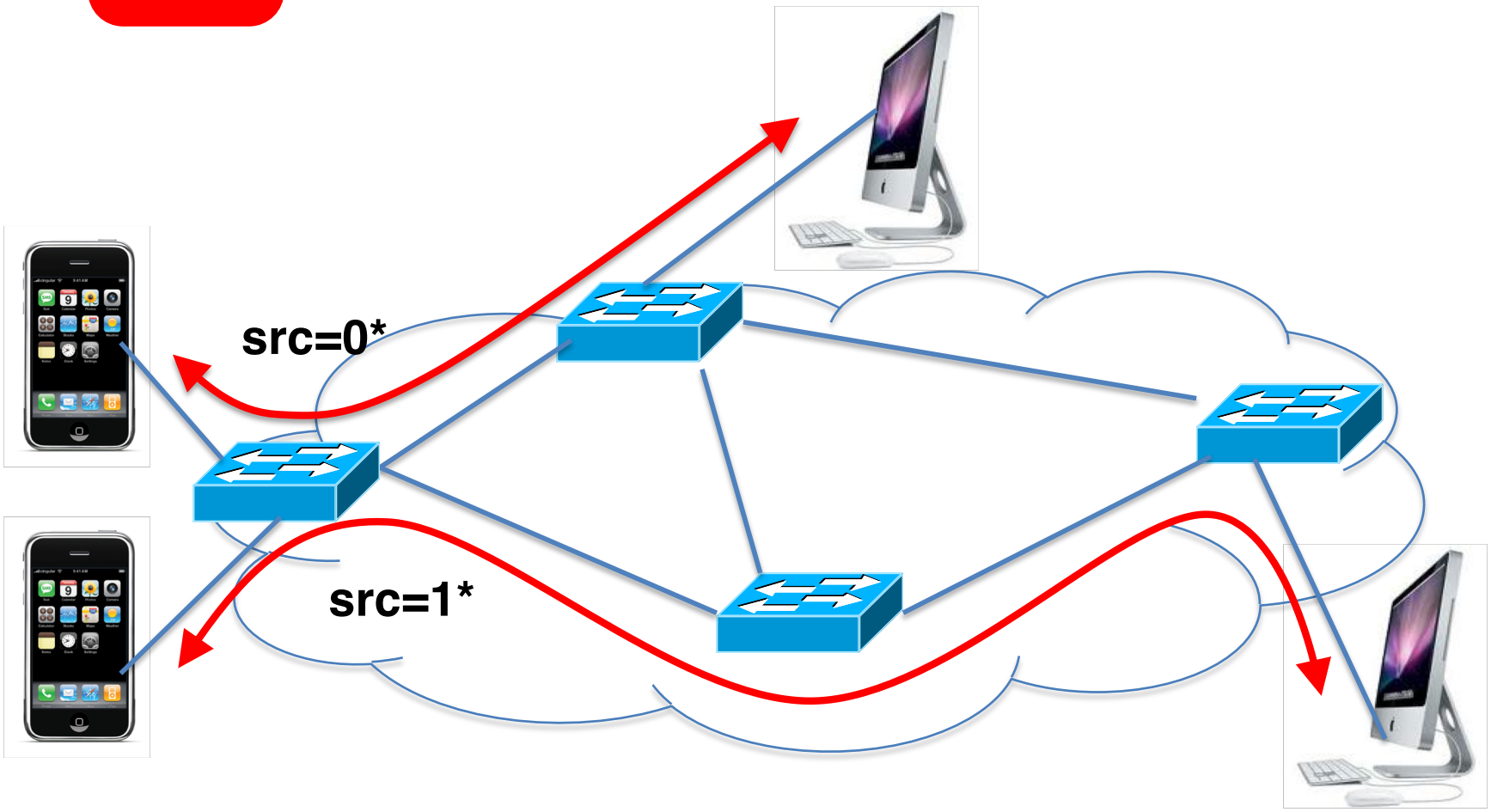


- See host send traffic at new location
- Modify rules to reroute the traffic



E.g.: Server Load Balancing

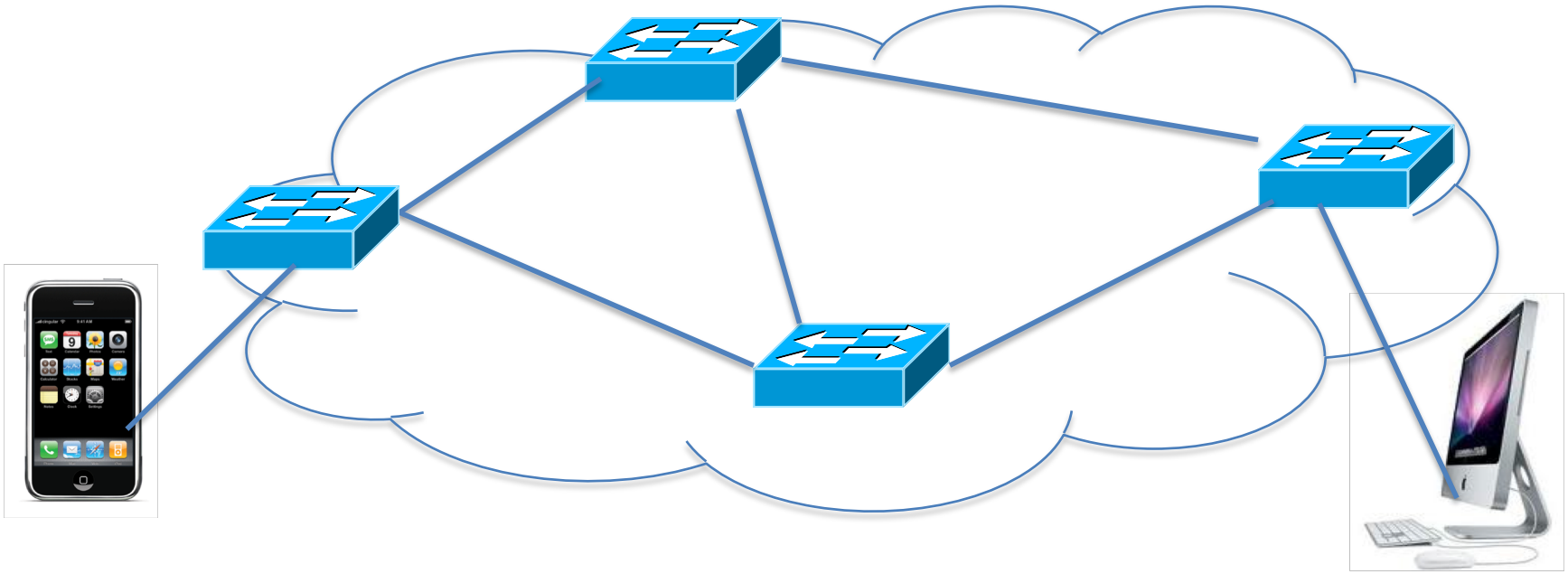
- Pre-install load-balancing policy
- Split traffic based on source IP



E.g.: Network Slicing

Controller #1 **Controller #2** **Controller #3**

Partition the space of packet headers



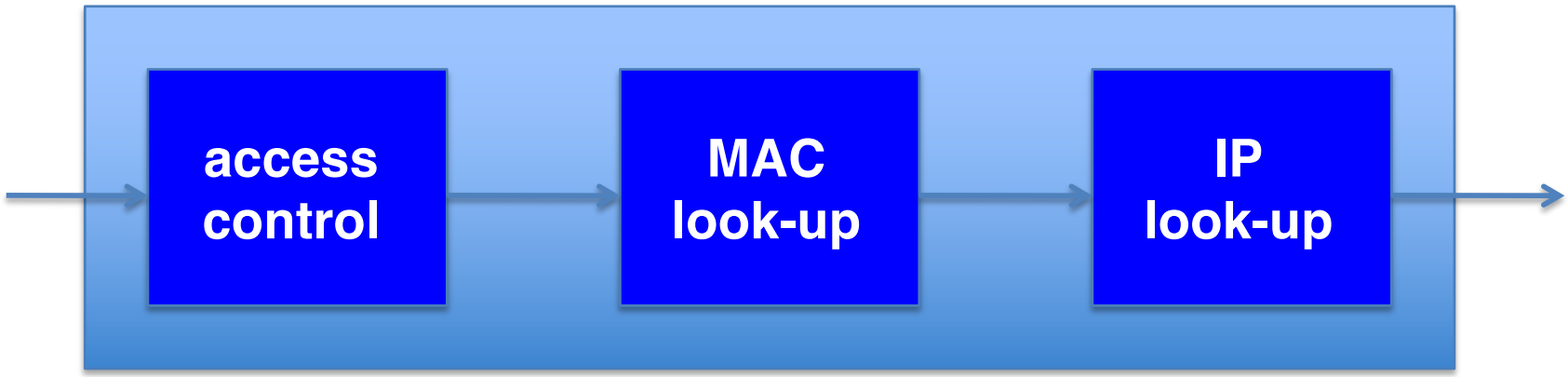
OpenFlow in the Wild

- Open Networking Foundation
  - Google, Facebook, Microsoft, Yahoo, Verizon, Deutsche Telekom, and many other companies
- Commercial OpenFlow switches
  - HP, NEC, Quanta, Dell, IBM, Juniper, …
- Network operating systems
  - NOX, Beacon, Floodlight, Nettle, ONIX, POX, Frenetic
- Network deployments
  - Eight campuses, and two research backbone networks
  - Commercial deployments (e.g., Google backbone)

Challenges

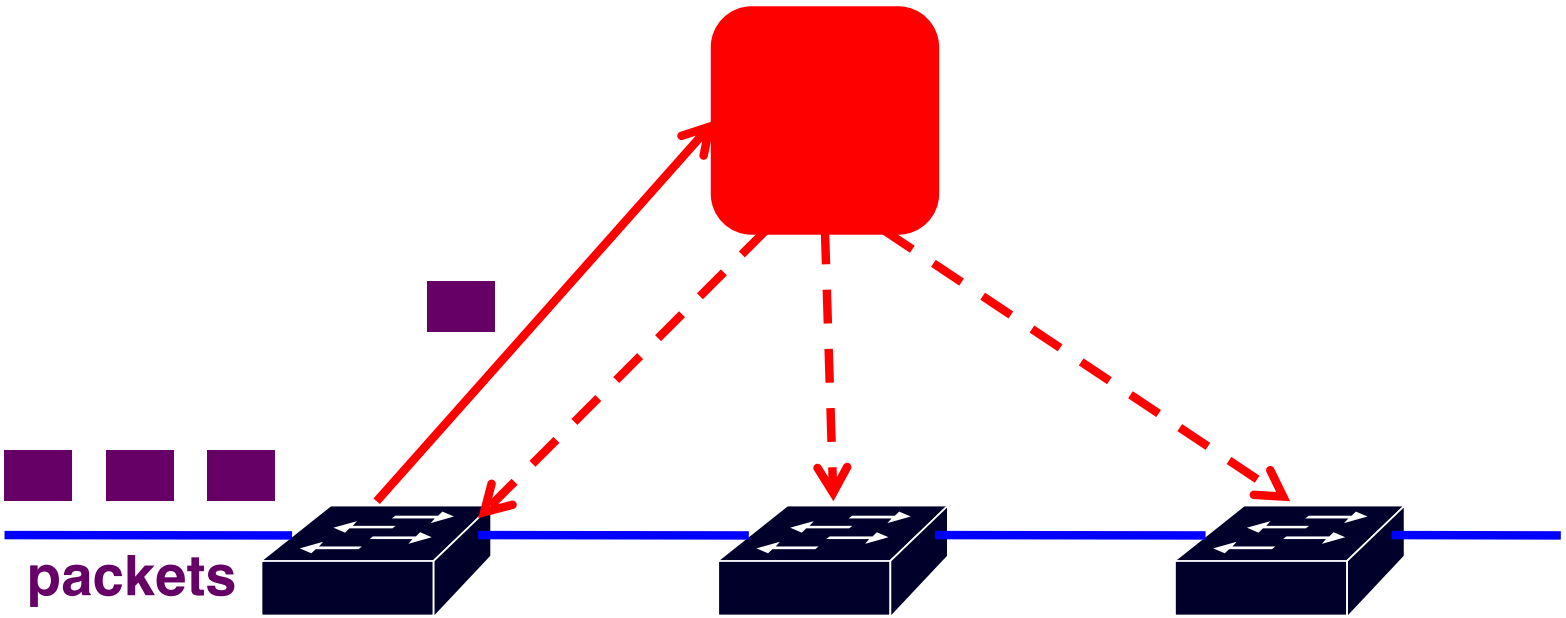
Heterogeneous Switches

- Number of packet-handling rules
- Range of matches and actions
- Multi-stage pipeline of packet processing
- Offload some control-plane functionality (?)

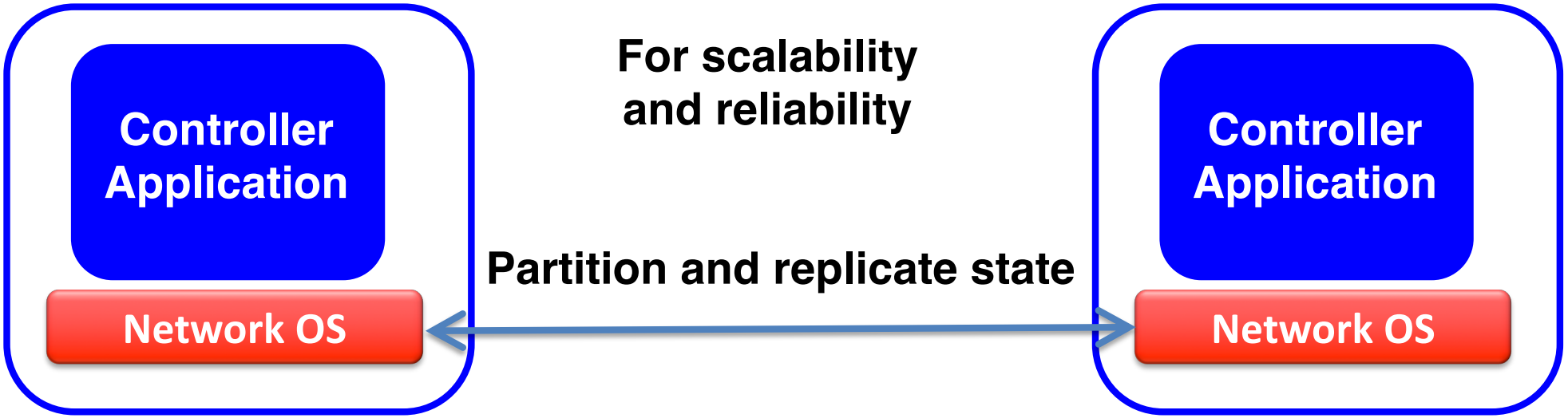


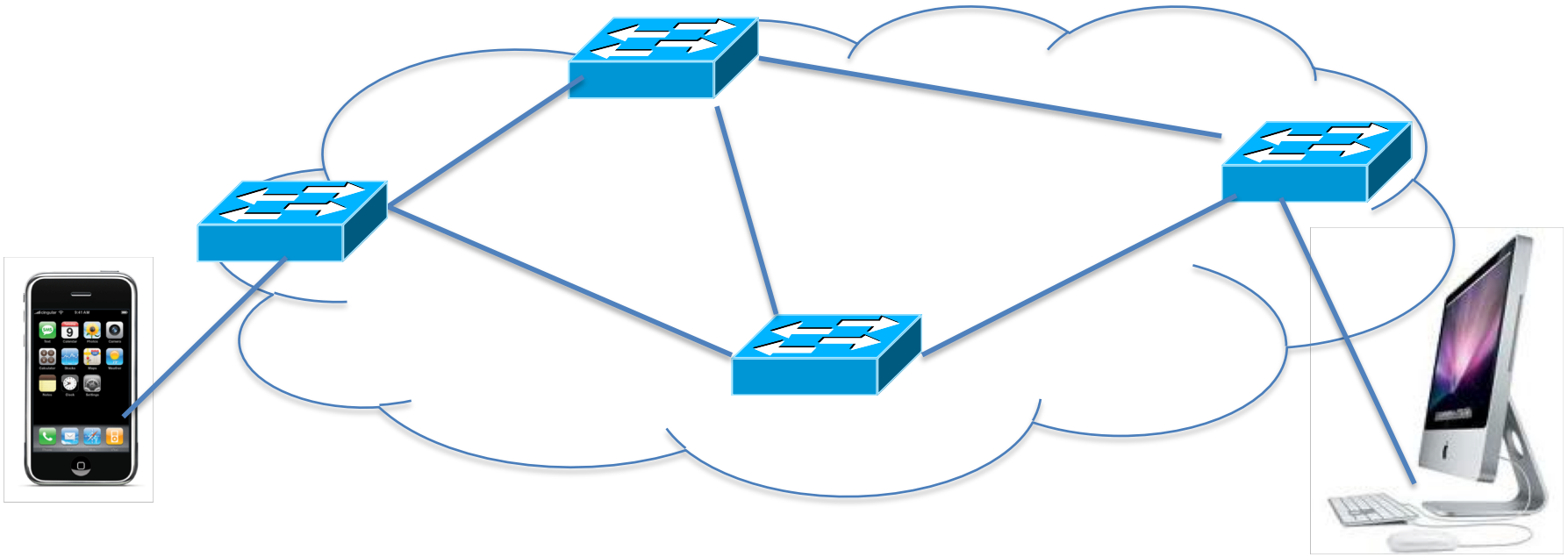
Controller Delay and Overhead

- Controller is much slower the the switch
- Processing packets leads to delay and overhead
- Need to keep most packets in the “fast path”



Distributed Controller



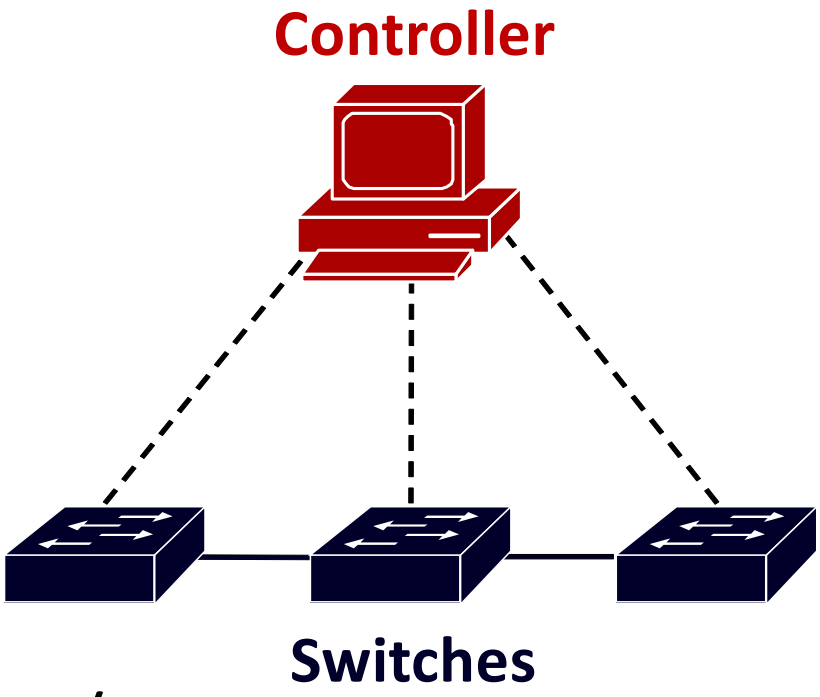


Testing and Debugging

- OpenFlow makes programming possible
  - Network-wide view at controller
  - Direct control over data plane
- Plenty of room for bugs
  - Still a complex, distributed system
- Need for testing techniques
  - Controller applications
  - Controller and switches
  - Rules installed in the switches

Programming Abstractions

- Controller APIs are low-level
  - Thin veneer on the underlying hardware
- Need better languages
  - Composition of modules
  - Managing concurrency
  - Querying network state
  - Network-wide abstractions
- Ongoing at Princeton
  - http://www.frenetic-lang.org/

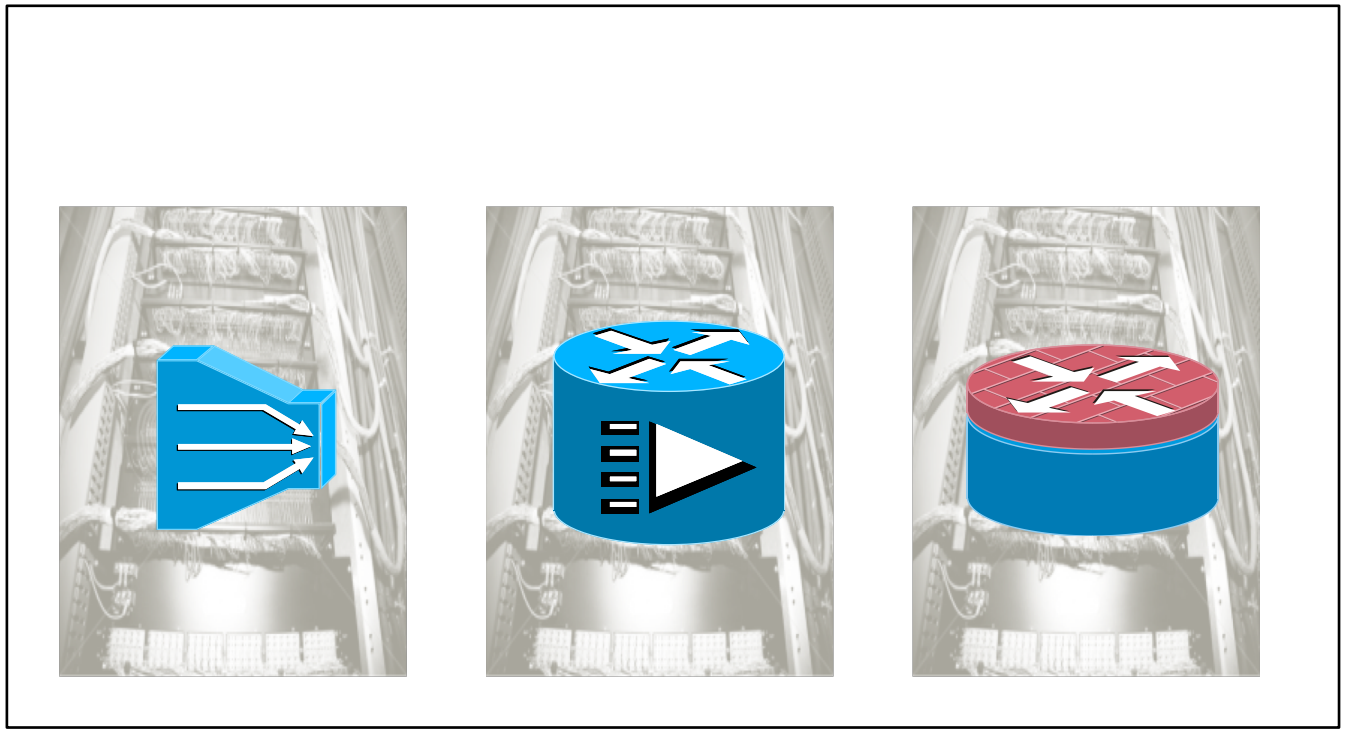


Conclusion on SDN

- Rethinking networking
  - Open interfaces to the data plane
  - Separation of control and data
  - Leveraging techniques from distributed systems
- Significant momentum
  - In both research and industry

Network Function Virtualization

Traditional network model



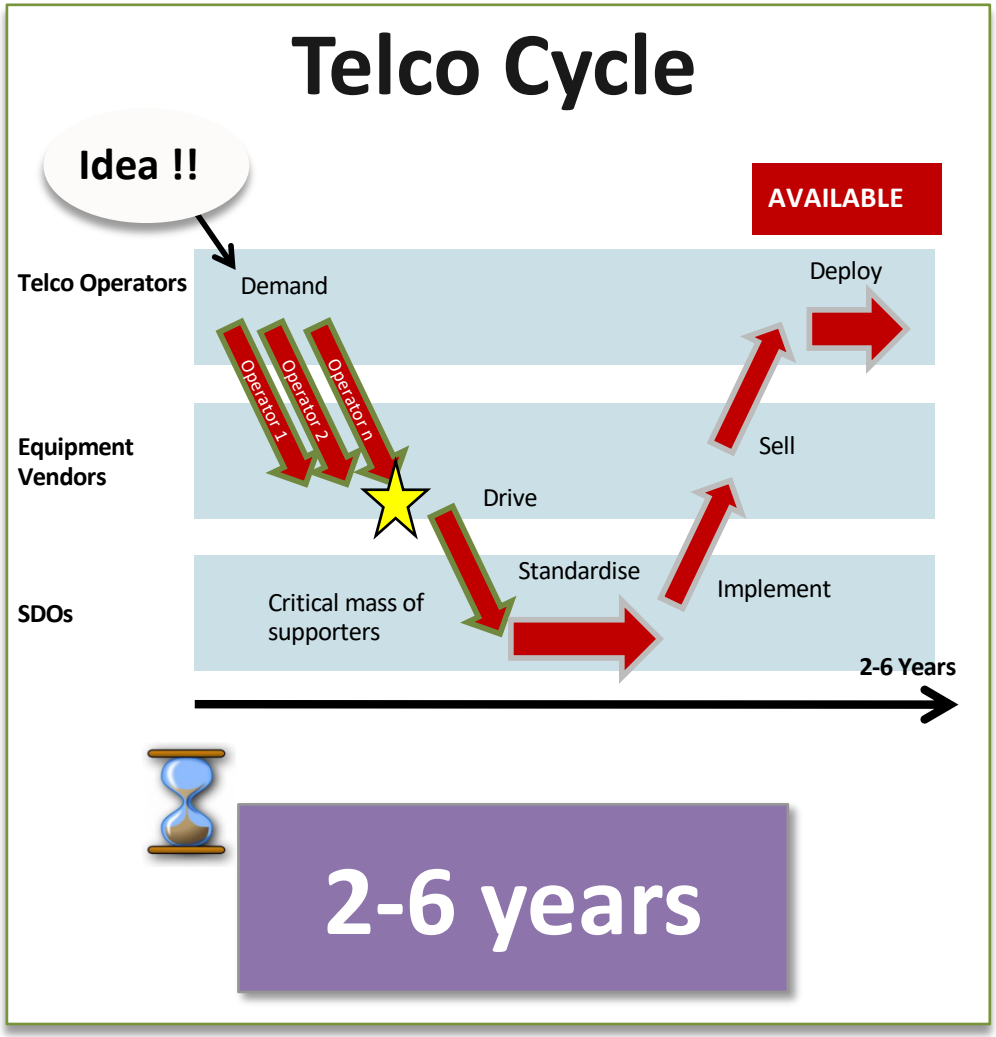
- Network functionalities are **based on specific HW&SW**
- **One physical node per role**

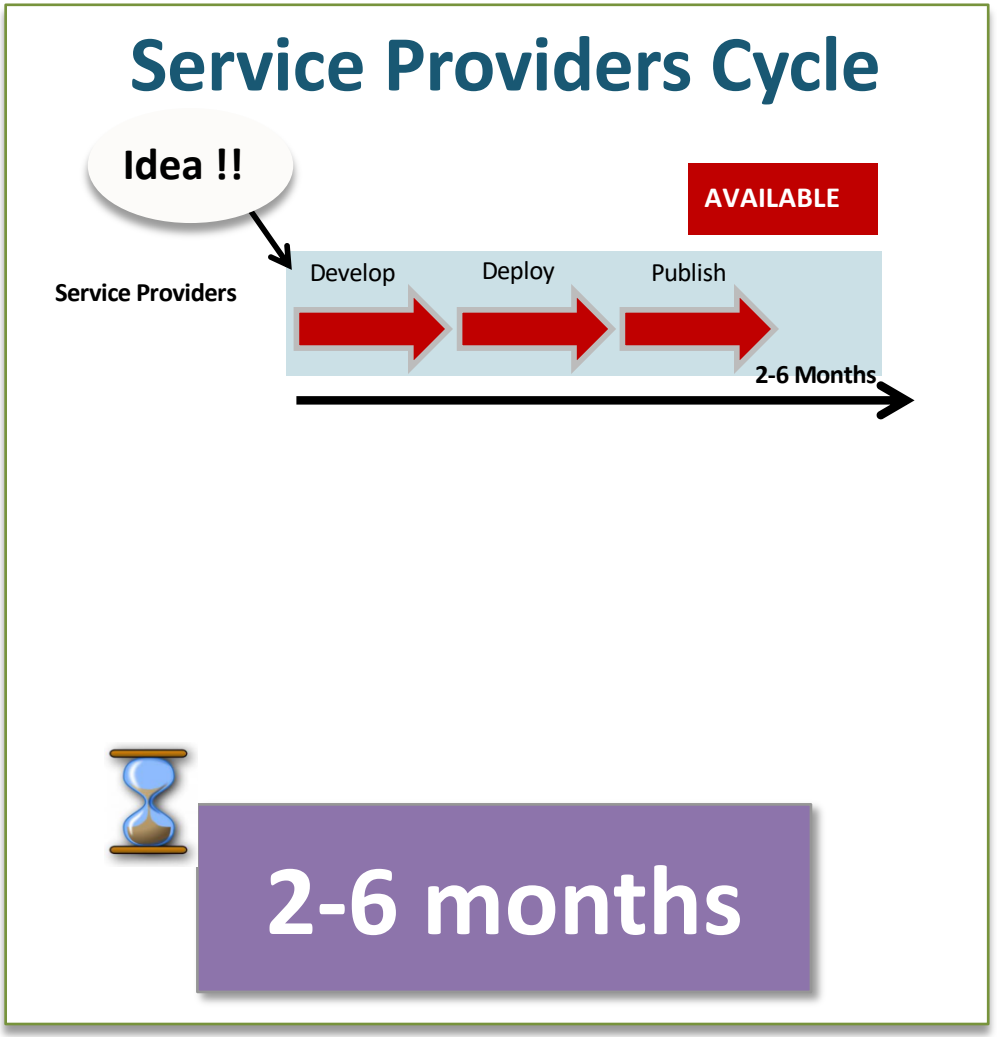
Motivation

Problem Statement

- Complex carrier networks
  - with a large variety of proprietary nodes and hardware appliances.
- Launching new services is difficult and takes too long
  - **Space and power to accommodate**
  - requires just another variety of box, which needs to be integrated.
- Operation is expensive
  - **Rapidly reach end of life**
  - due to existing procure-design-integrate-deploy cycle.

Hardware vs. Software





Source: Adapted from Diego Lopez Telefonica I+D, NFV

Trends in IT

- **Commodity x86**
- **Convergence**
- **Virtualization**
- **Cloud**
- **Mobility**

Telco Challenges

- Huge **capital investment**
- **Disparity between costs and revenues**
- **Complexity**
- Reduced **hardware life cycles**
- **Lack of flexibility and agility**
- **Launching new services is difficult and takes too long.**

Source: Adapted from D. Lopez Telefonica I+D, NFV

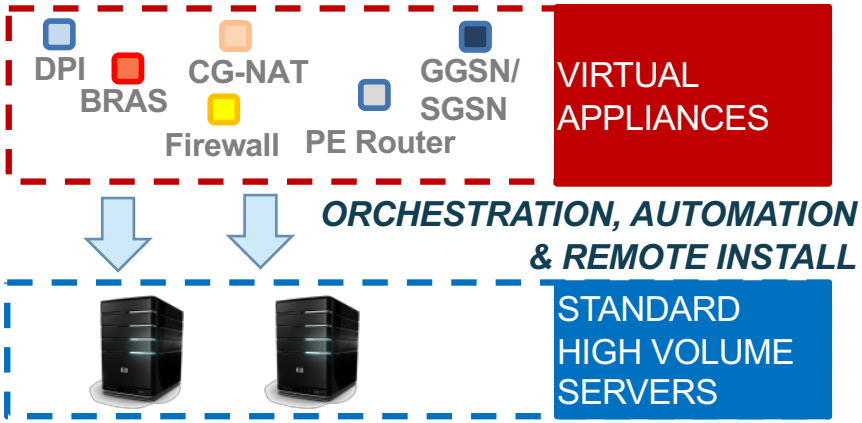
The NFV Concept

A means to make the network more flexible and simple by minimising dependence on HW constraints

Traditional Network Model:
APPLIANCE APPROACH

- Network Functions are **based on specific HW&SW**
- **One physical node per role**

Virtualised Network Model:
VIRTUAL APPLIANCE APPROACH



- Network Functions are **SW-based over well-known HW**
- **Multiple roles** over **same HW**

Source: Adapted from D. Lopez Telefonica I+D, NFV

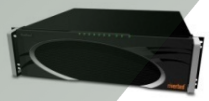
Target

Classical Network Appliance Approach


Message Router


CDN


Session Border Controller


WAN Acceleration


DPI


Firewall


Carrier Grade NAT


Tester/QoE monitor


SGSN/GGSN

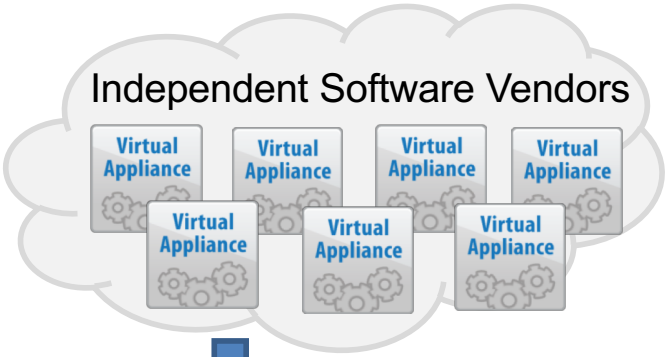

PE Router


BRAS


Radio Access Network Nodes

- Fragmented non-commodity hardware.
- Physical install per appliance per site.
- Hardware development large barrier to entry for new vendors, constraining innovation & competition.

Source: NFV



Network Virtualisation Approach

Benefits & Promises of NFV

- Network Functions Virtualization is about implementing network functions in software
  - that today run on proprietary hardware
  - leveraging commodity hardware and IT virtualization
- Provides opportunities for pure software players
  - Facilitates innovation
  - Faster time to market
- Supports multi-versioning and multi-tenancy of network functions, which allows use of a single physical platform for different applications, users and tenants

Source: Adapted from D. Lopez Telefonica I+D, NFV

Benefits & Promises of NFV

- Targeted service introduction based on geography or customer sets
  - More service differentiation & customization
- Services can be rapidly scaled up/down as required.
- IT-oriented skillset and talent
- Reduced equipment costs (CAPEX)
  - through consolidating equipment and economies of scale of IT industry.

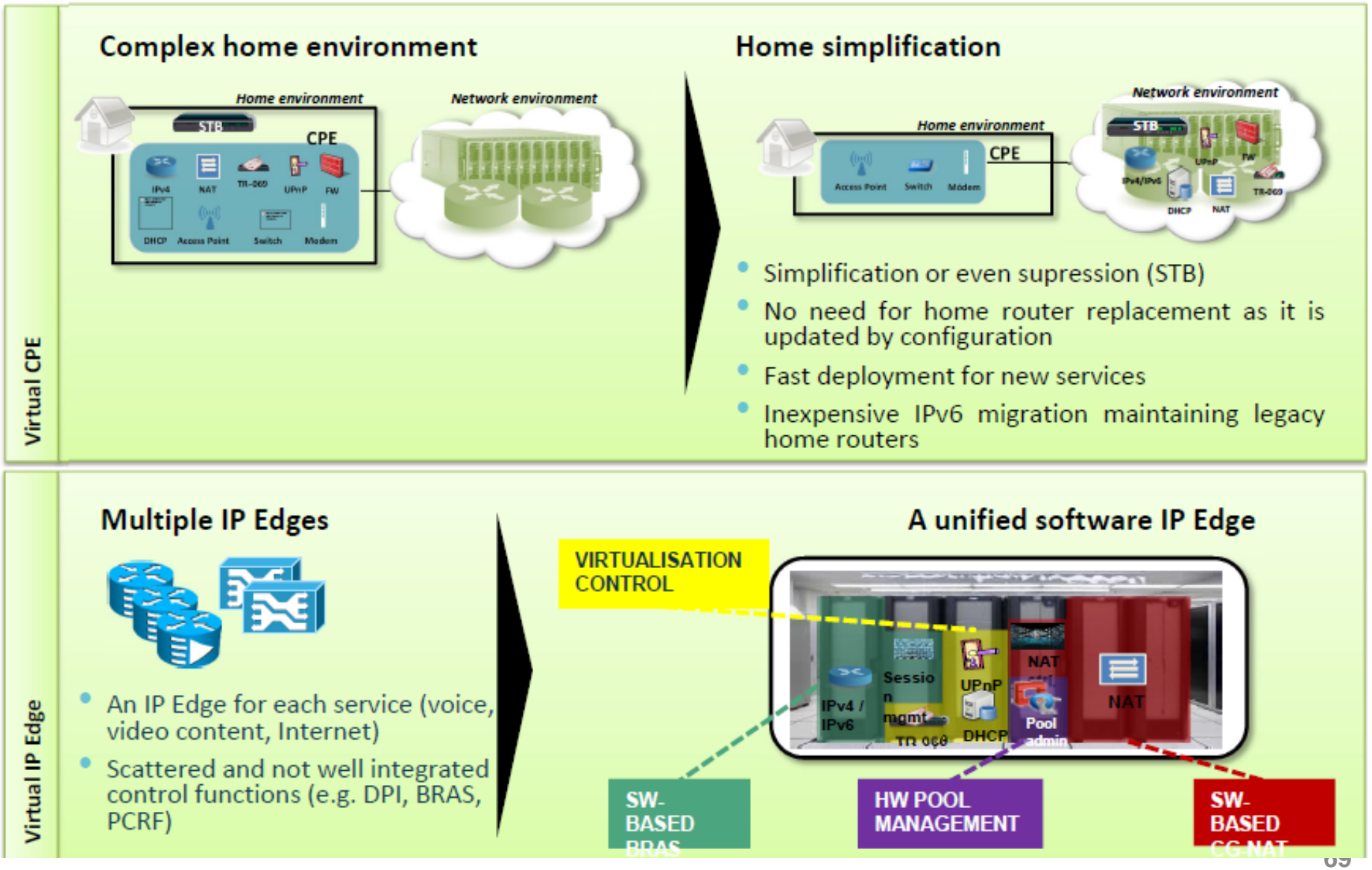
Source: NFV

Benefits & Promises of NFV

- Improved operational efficiency
  - by taking advantage of the higher uniformity of the physical network platform and its homogeneity to other support platforms.
- Reduced (OPEX) operational costs: reduced power, reduced space, improved network monitoring

Source: Adapted from D. Lopez Telefonica I+D, NFV

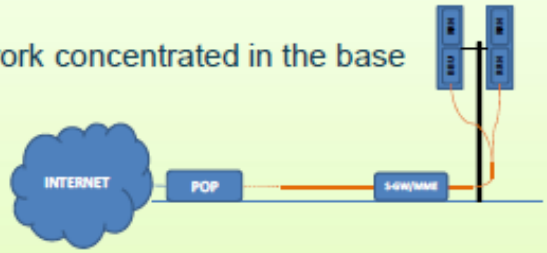
Use cases


Virtual CPE
Complex home environment
Home environment
Network environment
STB
CPE
IPv4
NAT
TR-069
UPnP
FW
DHCP
Access Point
Switch
Modem
Home simplification
Network environment
Home environment
CPE
Access Point
Switch
Modem
STB
UPnP
FW
IPv4/IPv6
TR-069
DHCP
NAT
Simplification or even supression (STB)
No need for home router replacement as it is updated by configuration
Fast deployment for new services
Inexpensive IPv6 migration maintaining legacy home routers
Virtual IP Edge
Multiple IP Edges
An IP Edge for each service (voice, video content, Internet)
Scattered and not well integrated control functions (e.g. DPI, BRAS, PCRF)
A unified software IP Edge
VIRTUALISATION CONTROL
IPv4 / IPv6
Session mgmt
TR 0c0
UPnP
DHCP
NAT
Pool admin
NAT
SW-BASED BRAS
HW POOL MANAGEMENT
SW-BASED CG-NAT

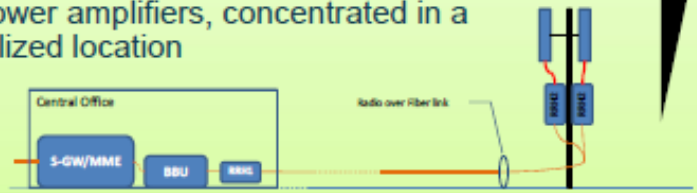
Other use cases

Mobile Network Virtualisation

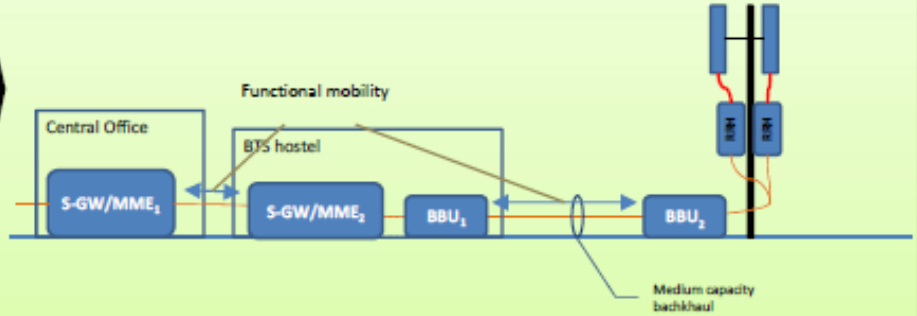
- All the network concentrated in the base station



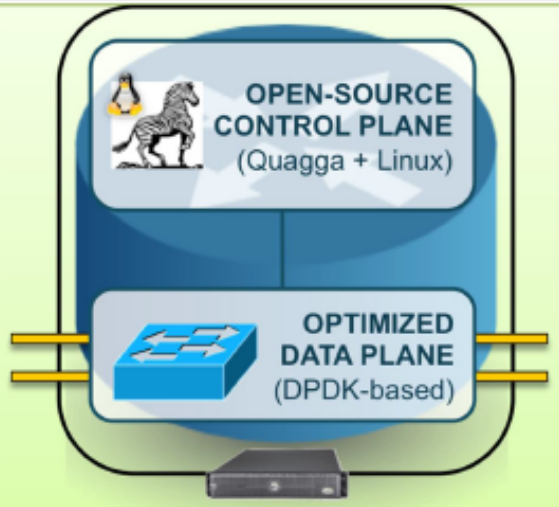
- C-RAN: All the base station functionalities, except for the antennas and power amplifiers, concentrated in a centralized location



Having the flexibility of **moving functionalities between different locations** may help to network to adopt the best option in each case



Optimized Quagga data plane



- Leverage on open source routing project as rich and widely tested protocol suite while assuring data plane performance
  - Common routing protocols supported and extended by open source project
  - High-performance line-rate data plane
  - Running in separate process, does not lead to licensing issues

Source: Adapted from D. Lopez Telefonica I+D, NFV

Many others…

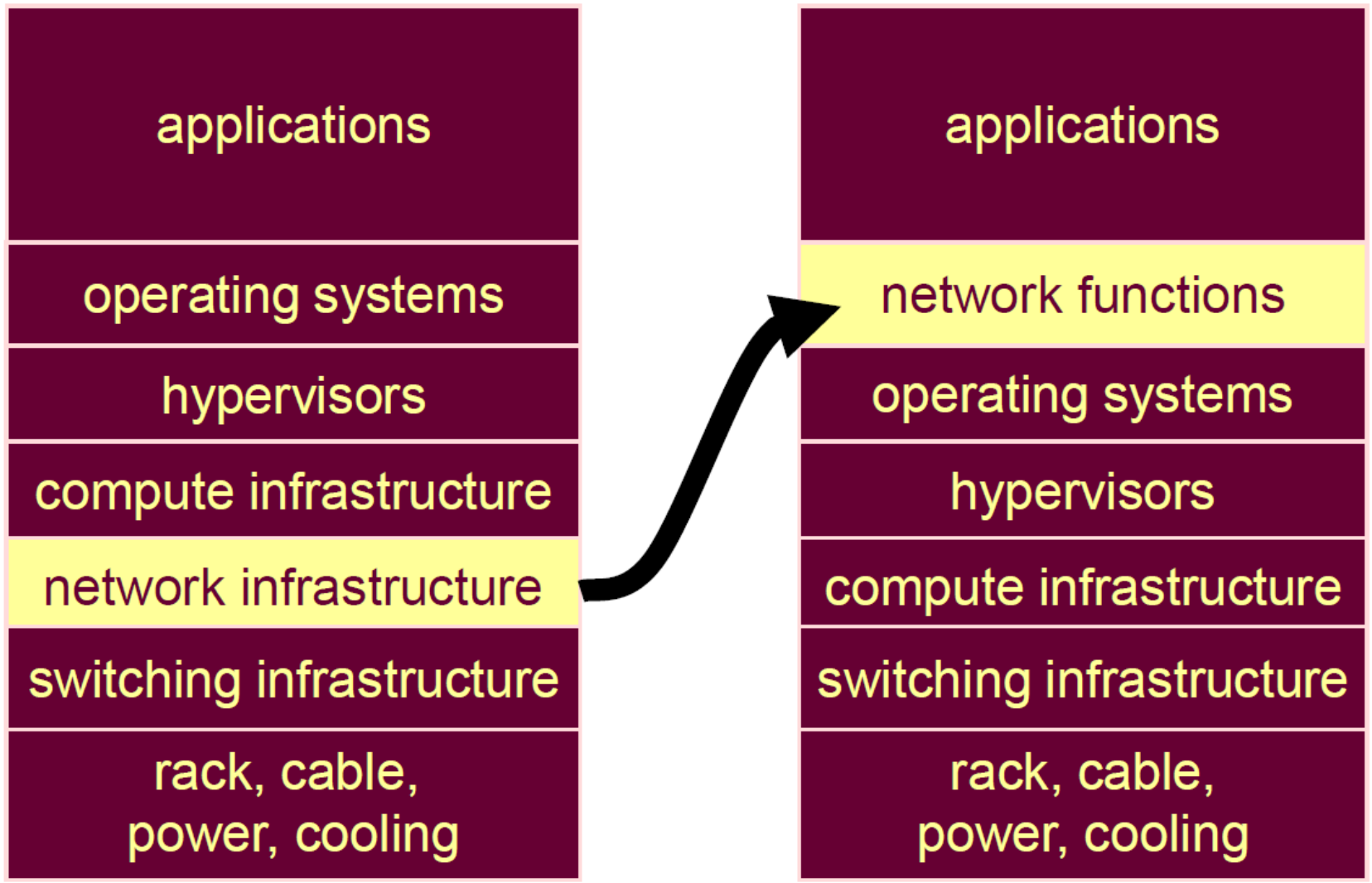
- Switching elements: BNG, CG-NAT, routers.
- Mobile network nodes: HLR/HSS, MME, SGSN, GGSN/PDN-GW.
- Home networks: Functions contained in home routers and set top boxes to create virtualised home environments.
- Tunnelling gateway elements: IPSec/SSL VPN gateways.
- Traffic analysis: DPI, QoE measurement.
- Service Assurance: SLA monitoring, Test and Diagnostics.
- NGN signalling: SBCs, IMS.
- Converged and network-wide functions: AAA servers, policy control and charging platforms.
- Application-level optimisation: CDNs, Cache Servers, Load Balancers, Application Accelerators.
- Security functions: Firewalls, virus scanners, intrusion detection systems, spam protection.

Source: NFV

NFV challenges

- Achieving high performance virtualised network appliance
  - portable between different HW vendors, and with different hypervisors
- Co-existence with bespoke HW based network platforms
  - enabling efficient migration paths to fully virtualised network platforms
- Management and orchestration of virtual network appliances
  - ensuring security from attack and misconfiguration
- NFV will only scale if all of the functions can be automated
- Appropriate level of resilience to HW and SW failures
- Integrating multiple virtual appliances from different vendors
  - Network operators need to be able to “mix & match” HW
  - hypervisors and virtual appliances from different vendors
  - without incurring significant integration costs and avoiding lock-in

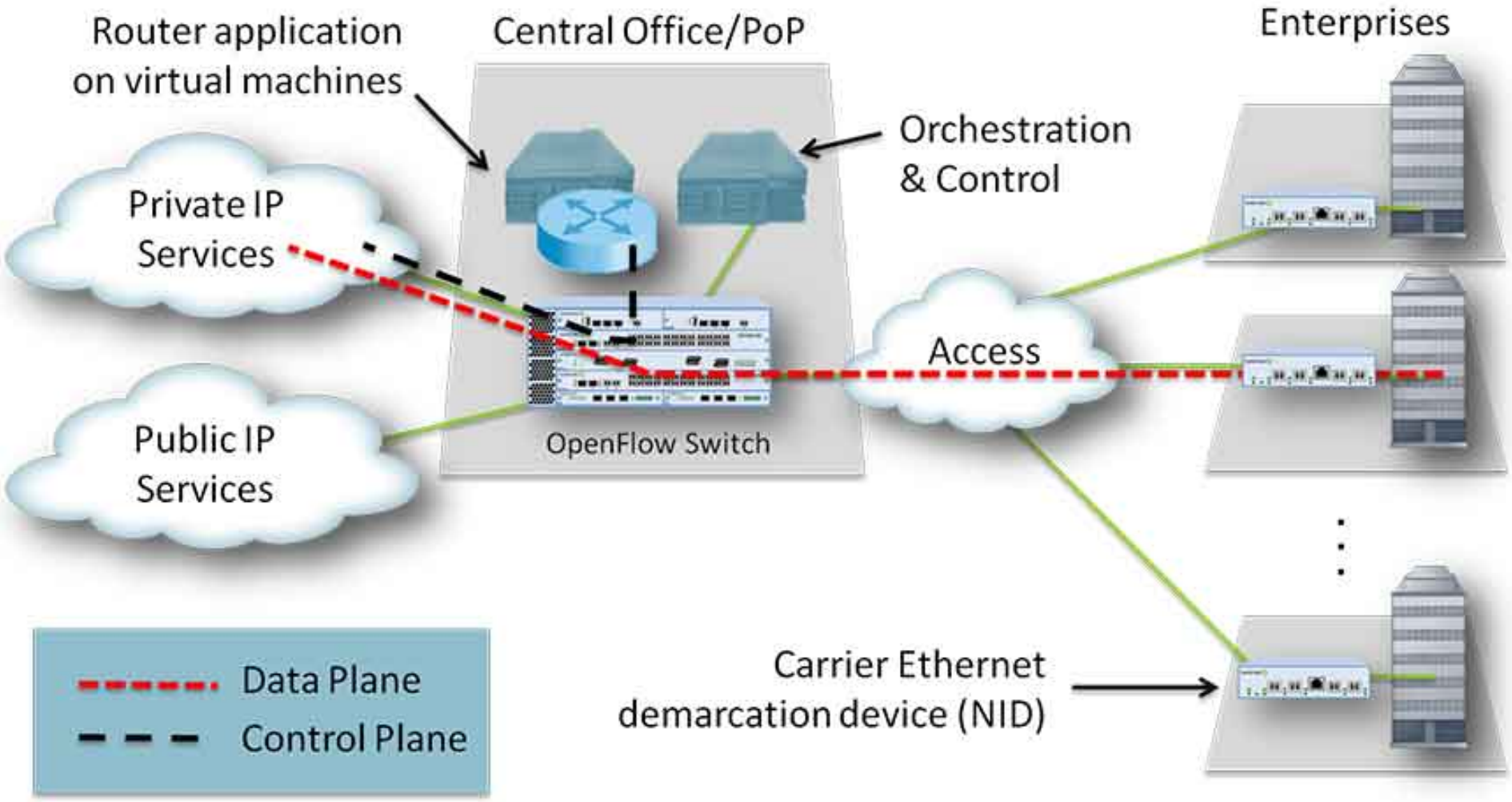
Rethinking relayering



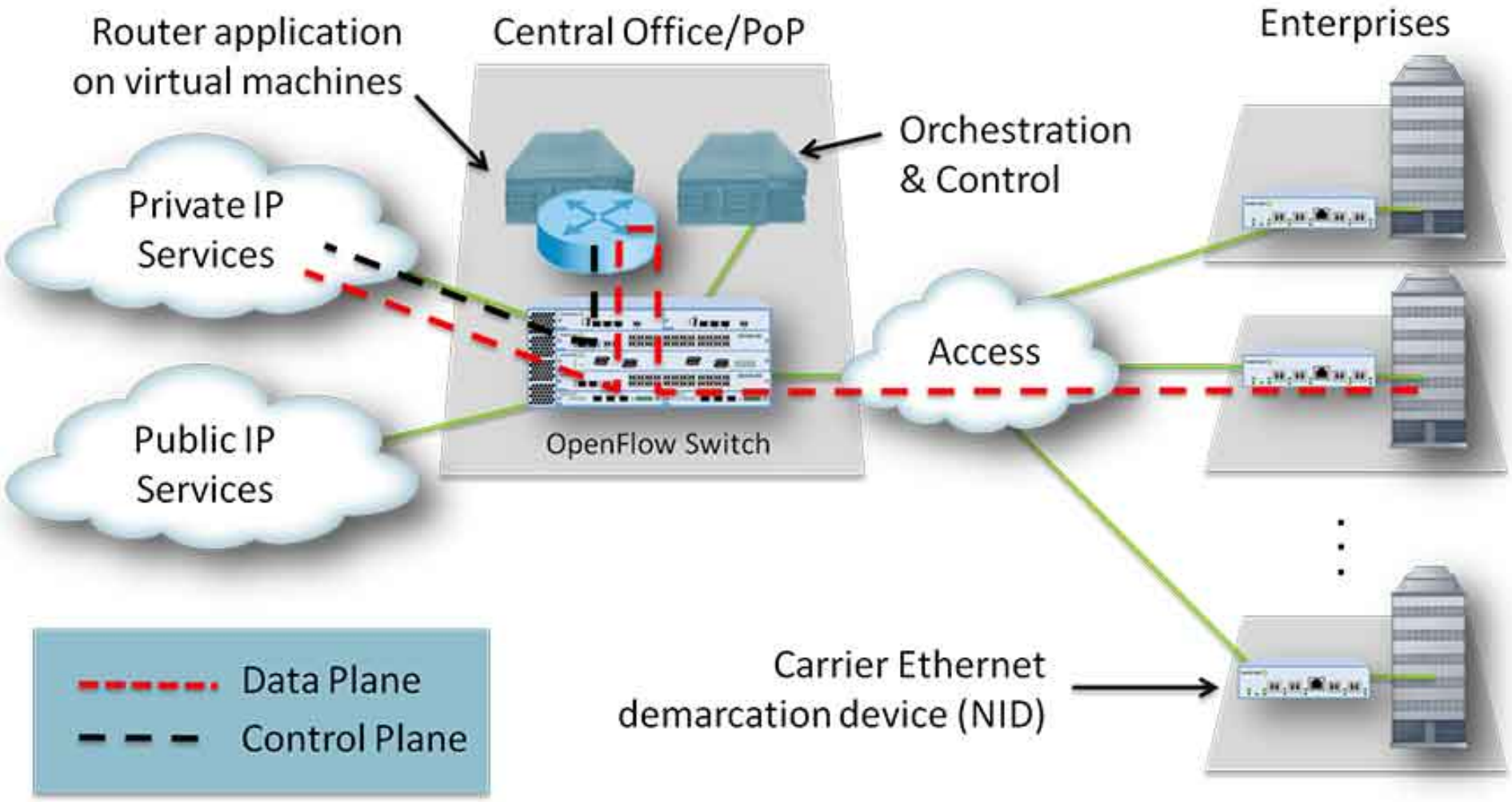
NFV and SDN

- NFV: re-definition of **network equipment architecture**
- NFV was born to meet Service Provider (SP) needs:
  - Lower CAPEX by reducing/eliminating proprietary hardware
  - Consolidate multiple network functions onto industry standard platforms
- SDN: re-definition of **network architecture**
- SDN comes from the IT world:
  - **Separate the data and control layers**, while centralizing the control
  - Deliver the ability to **program** network behavior using well-defined interfaces

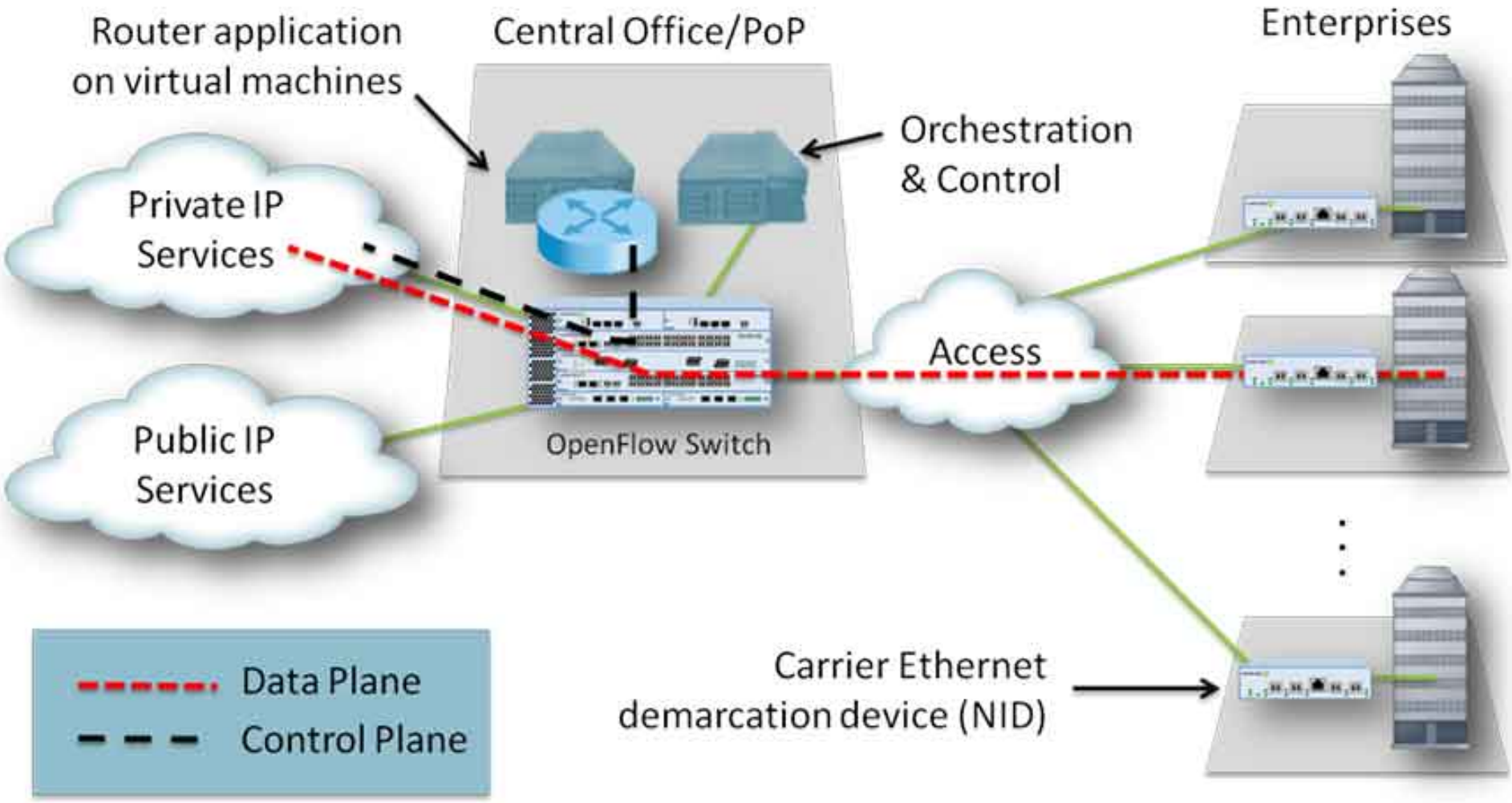
Manage router service today



Managed router using NFV



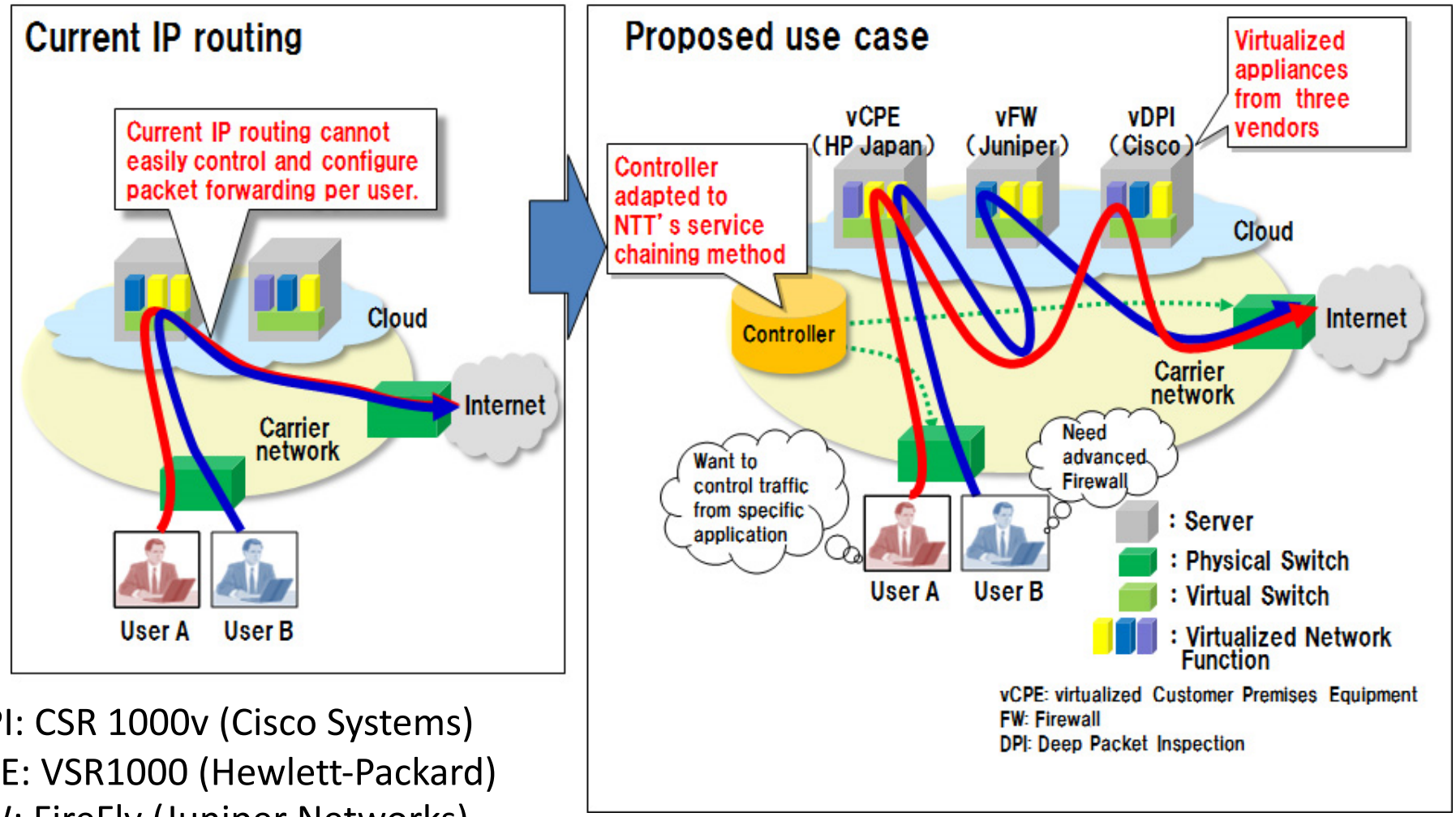
Managed router using NFV & SDN



Service Fonctions Chaining (SFC)

- Set of network services, such as firewalls or application delivery controllers interconnected through the network to support an application
- In the past
  - Required specialized hardware & (re)configuration
  - Unnecessary bandwidth and CPU consumption
- NFV/SDN enables to quickly and inexpensivelycreate, modify and remove service chains.
  - entire service chains can be provisioned and constantly reconfigured from the controller
  - No hardware purchase; NF execute as elastic VM

Service Chaining for NW Function Selection in Carrier Networks



vDPI: CSR 1000v (Cisco Systems)
vCPE: VSR1000 (Hewlett-Packard)
vFW: FireFly (Juniper Networks)
VIM (NW Controller): Service Chaining Function (prototype) + Ryu (NTT)

Source: ETSI Ongoing PoC

Take home messages

- SDN is about network programmability
- NFV is about software network functions
- Current vendors/operators should adapt to software development cycles