Cloud e Datacenter Networking

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI
Laurea Magistrale in Ingegneria Informatica

Prof. Roberto Canonico

Software Defined Networking

Lesson outline

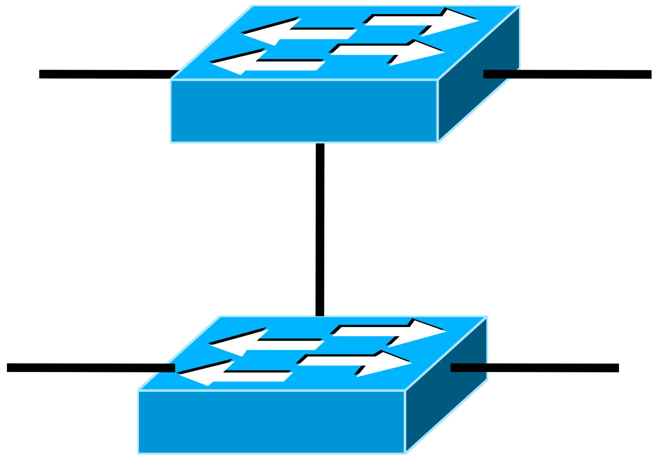
- Software Defined Networks (SDN)
- Credits for the material:
  - Jennifer Rexford
  - Nick McKeown
  - Scott Shenker

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 network: a different story…

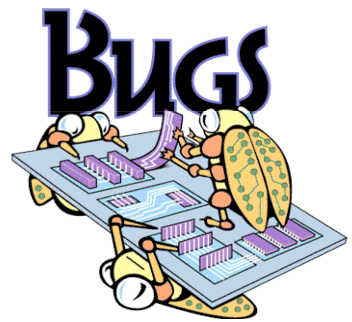
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features



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

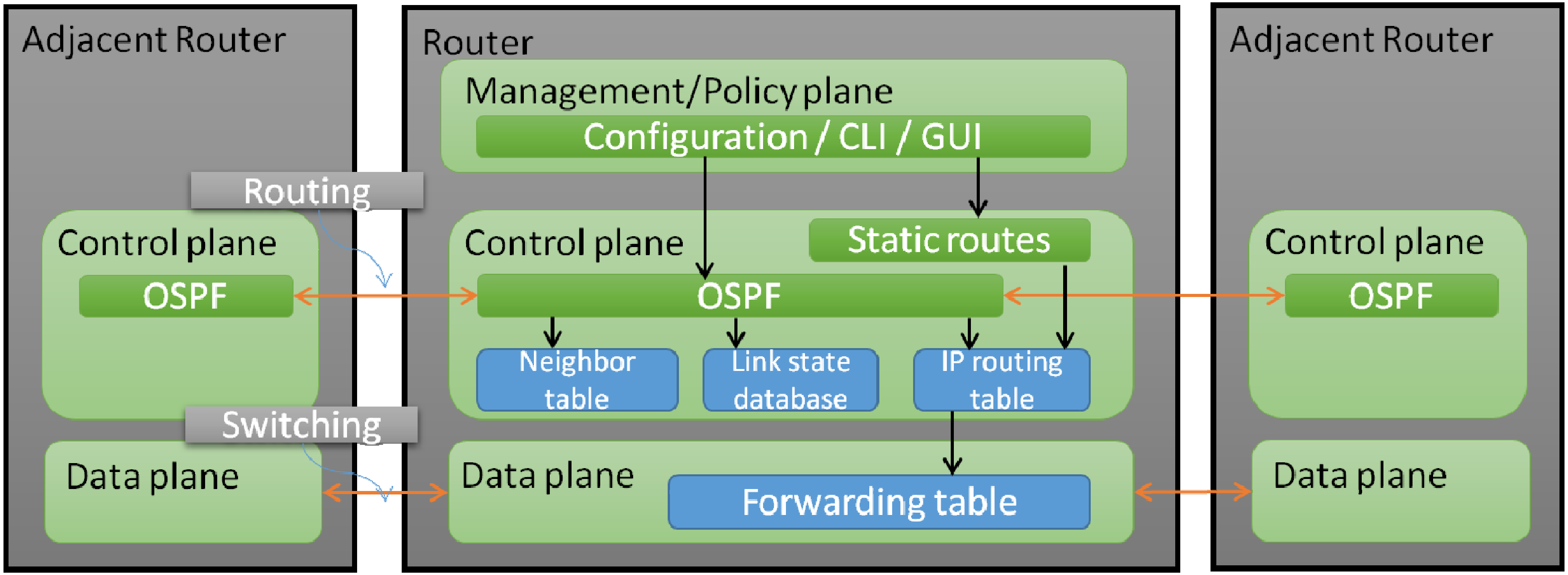


Traditional networking

- Each networking device operates at three different planes
- Management plane: configuration – responsible of general device behavior
  - Determines how the control plane should be configured
  - Slow time-scales (manual configuration by network administrators)
- Control Plane: decision – responsible of establishing the state in routers
  - Determines how and where packets are forwarded
  - Routing, traffic engineering, firewall state, …
  - Slow time-scales (per control event)
- Data Plane: responsible of processing and forwarding of packets
  - Based on state in routers and endpoints
  - E.g., IP, TCP, Ethernet, etc.
  - Fast timescales (per-packet)

Traditional networking: a router's view

- **In an IP router, the control plane role is played by dynamic routing protocols and the associated state**
  - **E.g. OSPF and the Link State Database**



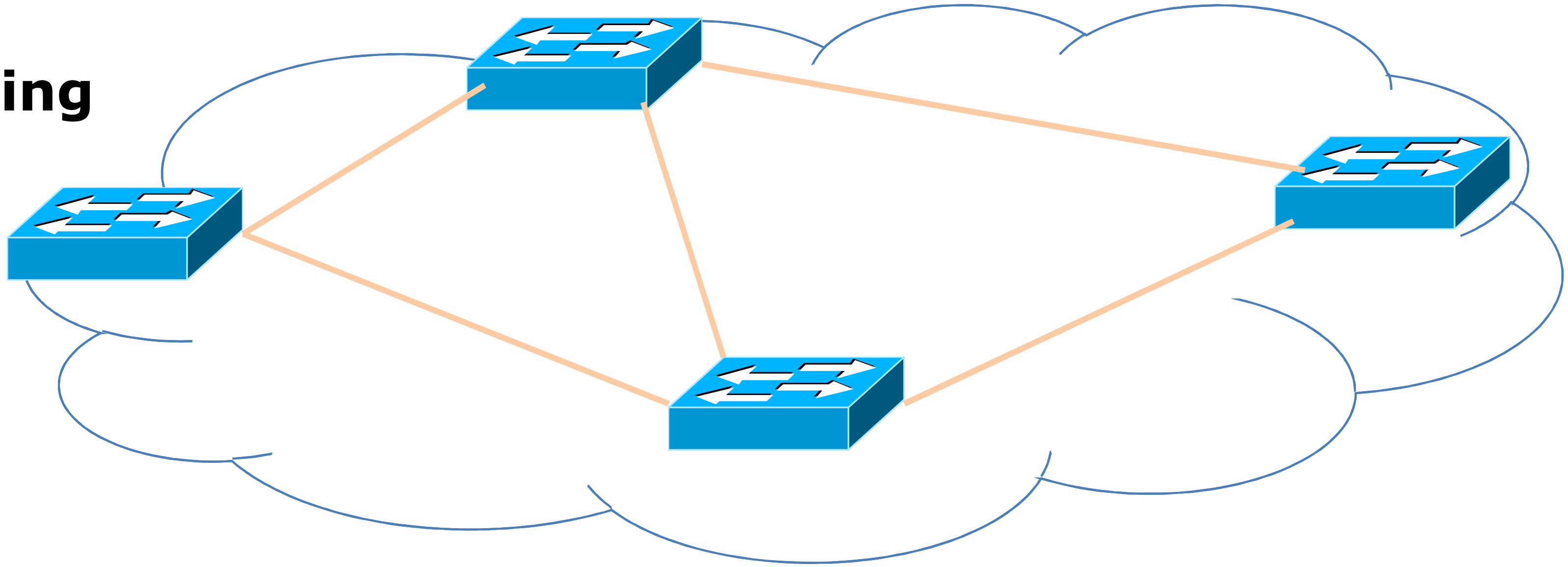
Tradional planes and time scales

	Data	Control	Management
Time scales	Packets	Events	Humans
Task	Forwarding/ buffering/ filtering/ scheduling	Routing, circuit set-up	Analysis, configuration
Location	Hardware • Specialized hardware • Processes at line rate • Every packet • Very fast	Router software • Uses CPU • Can only process a small number of packets • Very slow	Human or perl scripts

Traditional Computer Networks: data plane

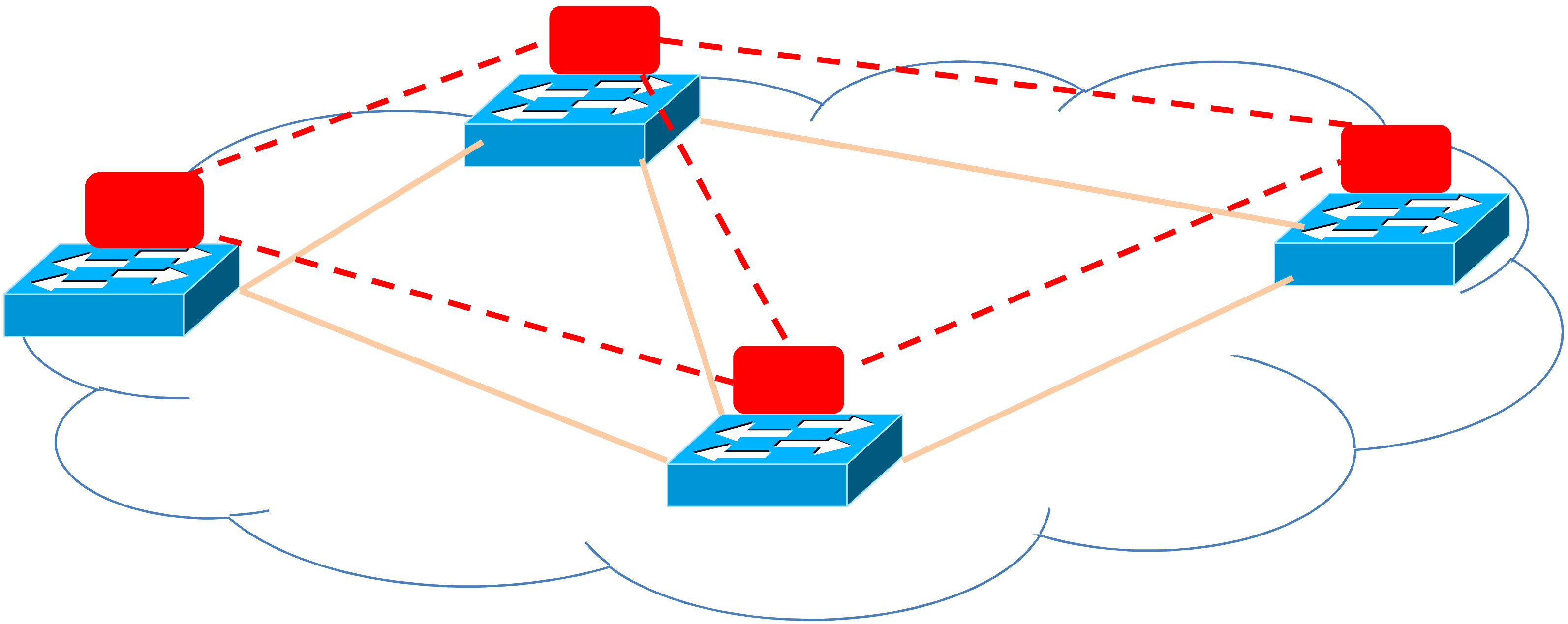
Data plane:

Packet streaming



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

Traditional Computer Networks: control plane

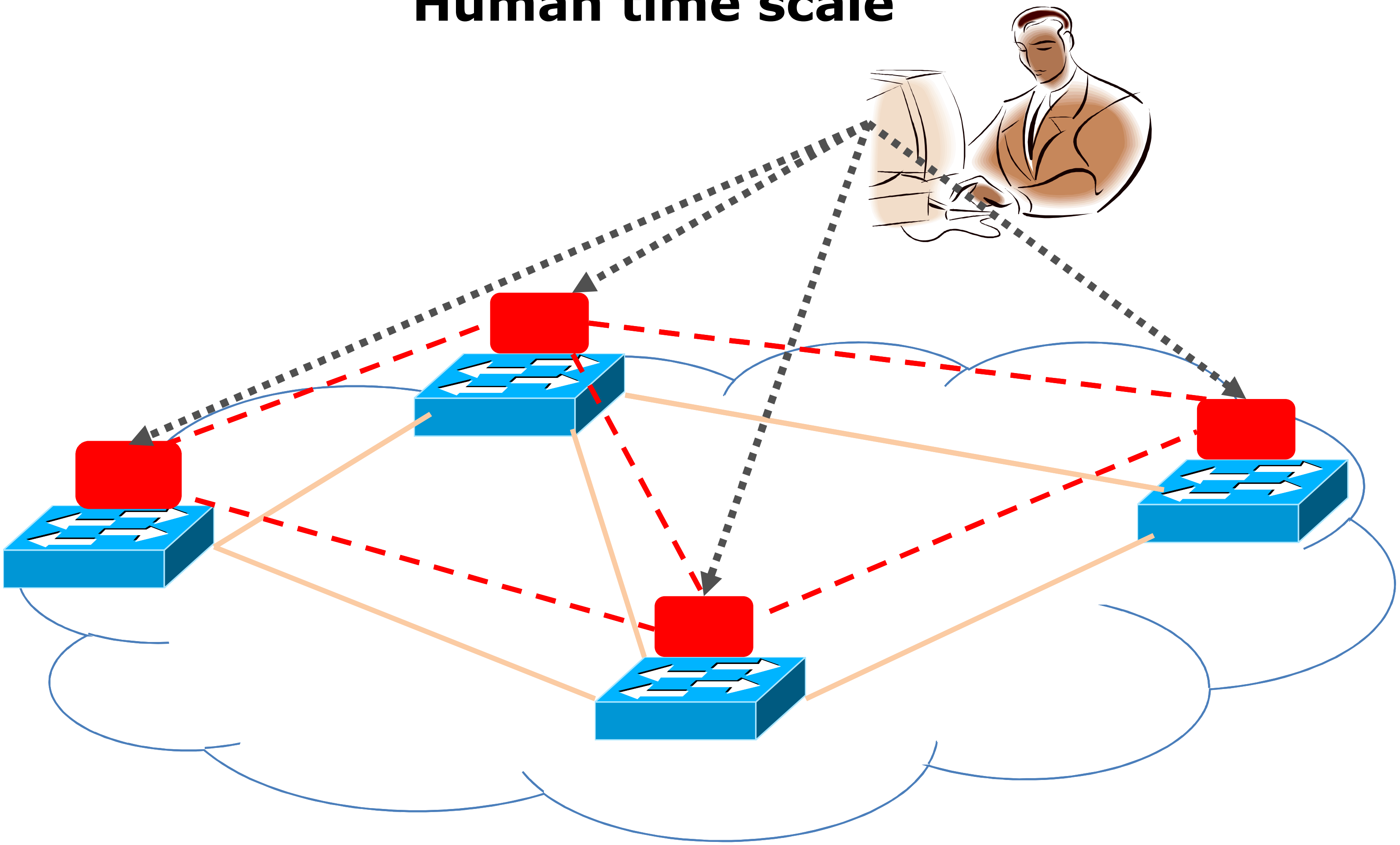


Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks: management plane

Management plane:

Human time scale

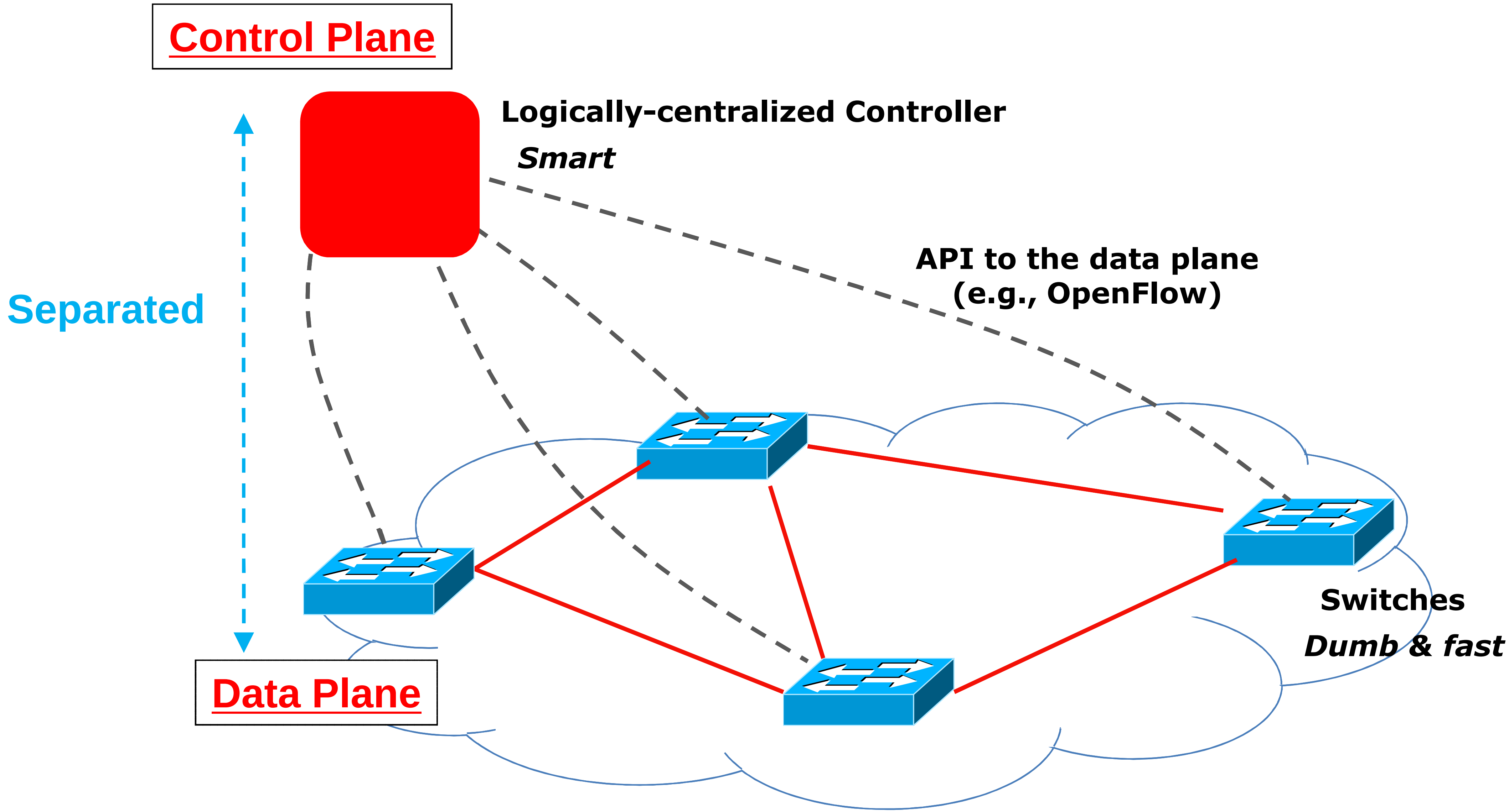


Collect measurements and configure the equipment

SDN Concept

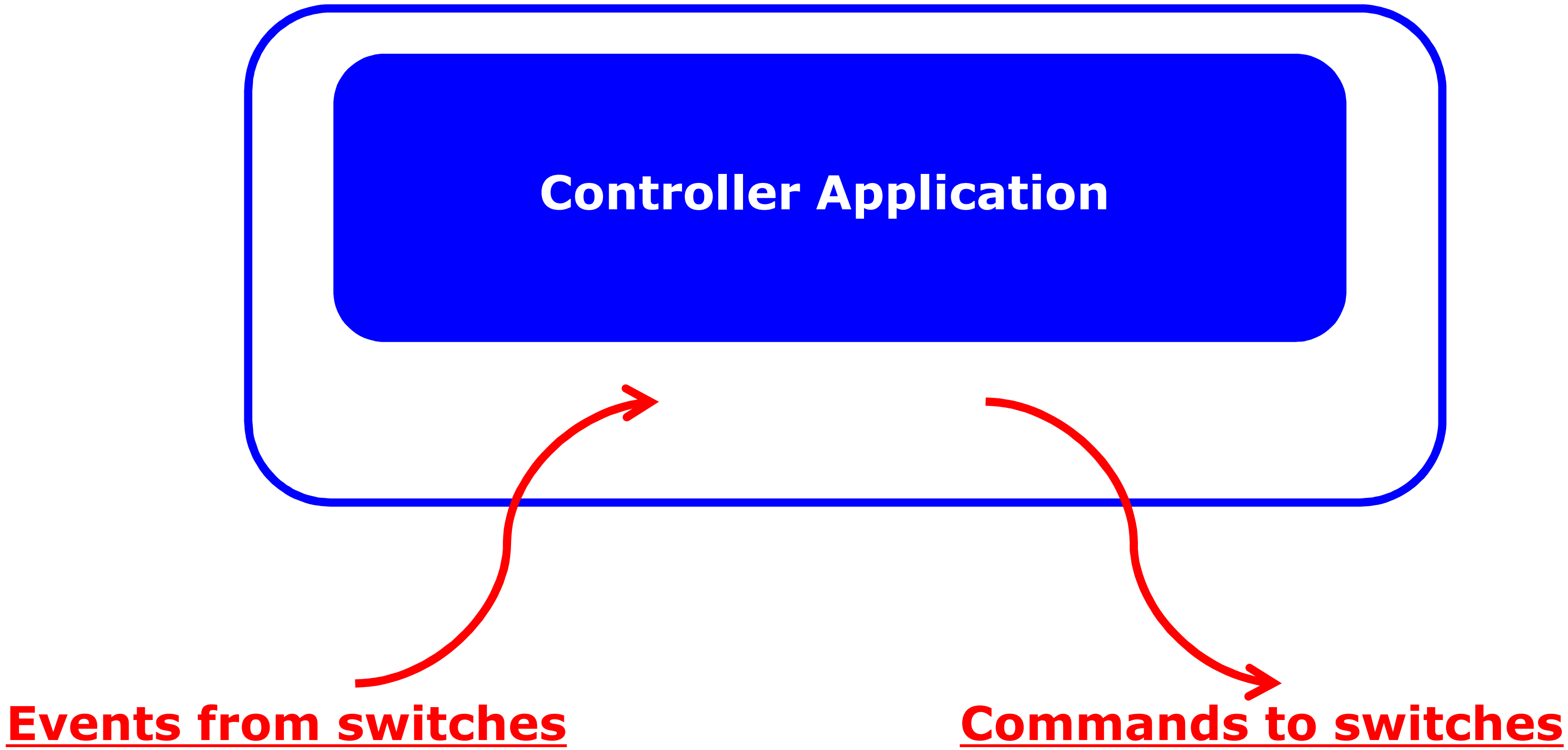
- Separate control plane and data plane entities
  - Network intelligence and state are logically centralized
  - The underlying network infrastructure is abstracted from the applications
- Remotely control network devices from a central entity
- Execute or run control plane software on general purpose hardware
  - Decouple from specific networking hardware
  - Use commodity servers
- An architecture to control
  - not just a networking device …
  - but an entire network
- Expected advantages:
  - Ability to innovate through software
    - Overcome the “Internet ossification problem”
  - Cost reductions through increased competition, hardware commoditization and open-source software

Software Defined Networking (SDN)



- A logically centralized "Controller" uses an open protocol (e.g. OpenFlow) to:
  - Get state information from forwarding elements (i.e. switches)
  - Give controls and directives to forwarding elements

SDN: controller programmability



SDN allows to unify different kinds of boxes

- Router
  - Match: longest destination IP prefix
  - Action: decrement TTL, re-compute header checksum, 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

All the above boxes may be replaced by "generic" SDN switches whose behaviour is programmed in the controller

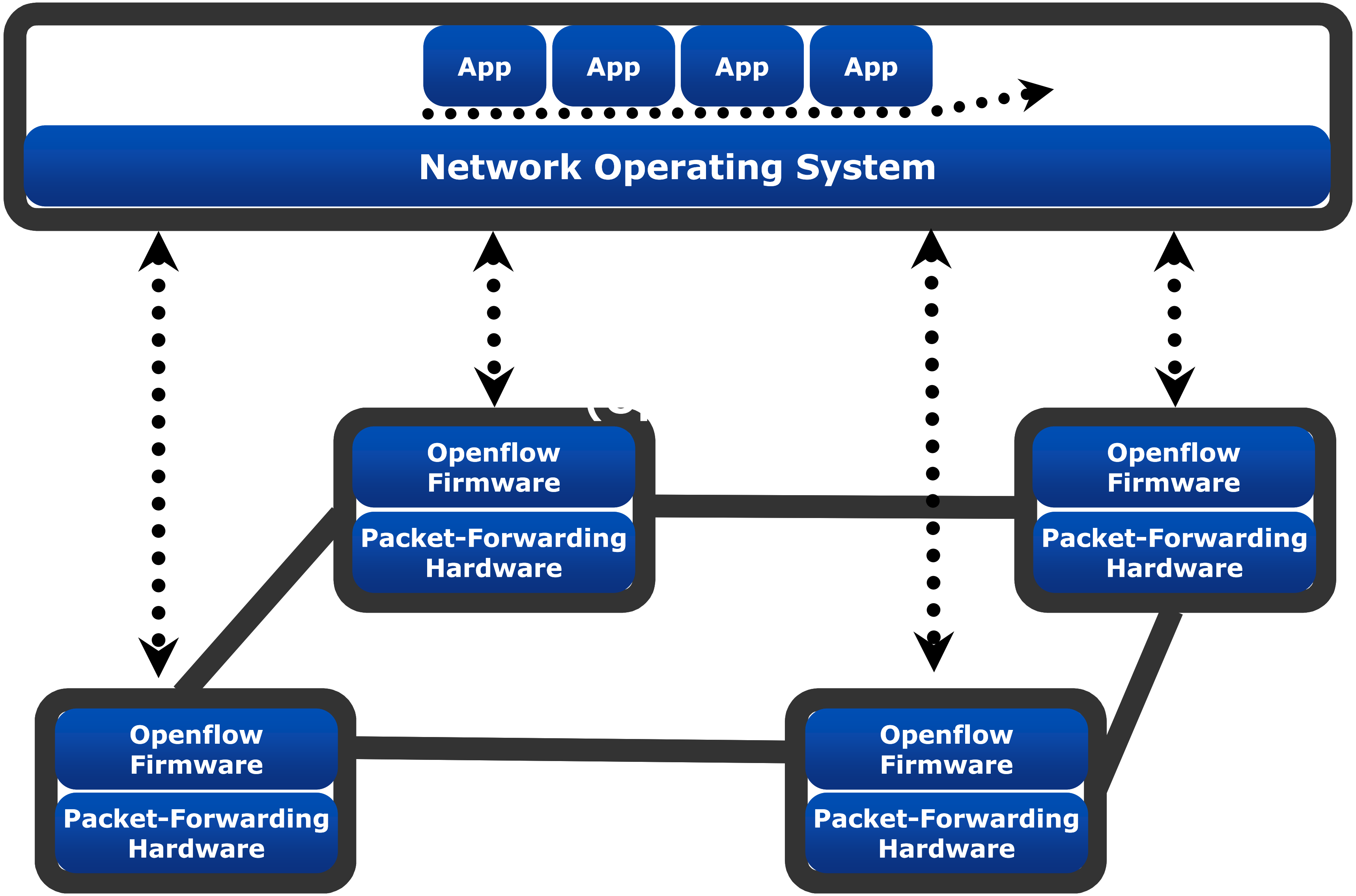
By decoupling the network function from the physical infrastructure, the SDN approach is also useful to second another emerging trend in the telecommunications industry: *Network Function Virtualization* (NFV)

A Short History of SDN

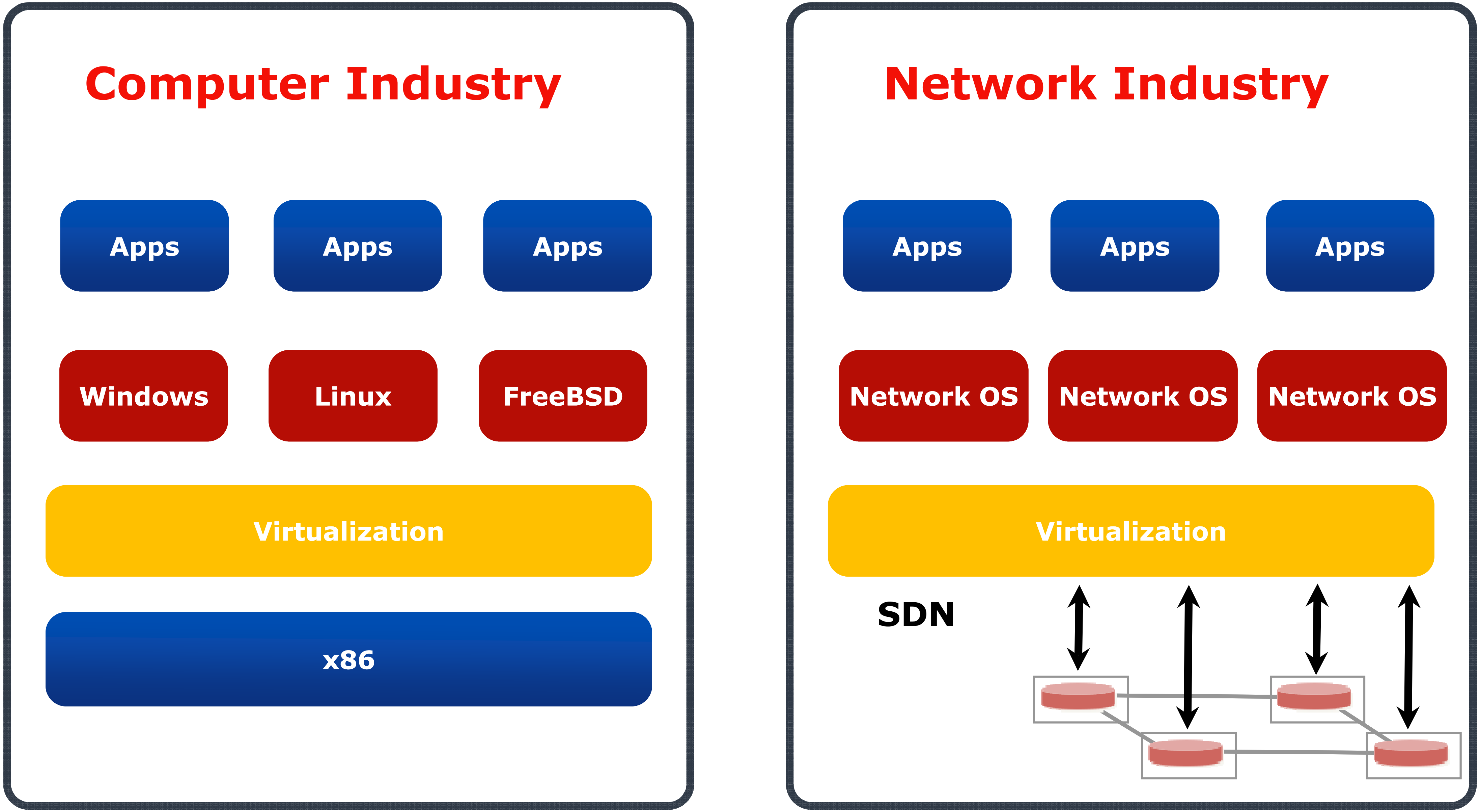
- ~2004: Research on new management paradigms
  - RCP, 4D [Princeton, CMU,….]
- 2006: Martin Casado, a PhD student at Stanford and team propose a clean-slate security architecture (SANE) which defines a centralized control of security (instead of at the edge as normally done)
- 2008: the idea of *Software Defined Network* is originated from OpenFlow project (*ACM SIGCOMM 2008*)
- 2009: Stanford publishes OpenFlow V1.0.0 specs
- June 2009: Martin Casado co-founds Nicira
- 2011: Open Networking Foundation (~69 members)
  - Board: Google, Yahoo, Verizon, DT, Msoft, F'book, NTT
  - Members: Cisco, Juniper, HP, Dell, Broadcom, IBM,…..
- 2012: Latest Open Networking Summit
  - Almost 1000 attendees, Google: SDN used for their WAN
  - Commercialized, in production use (few places)
- July 2012: VMware buys Nicira for $1.26B

An OS for networks thanks to SDN

- **The whole network as one big machine**
- **The key is to have a standardized control interface that speaks directly to hardware**



The “Software-Defined Networking” (r)evolution



Computers evolution: from mainframes to microprocessors

From Nick McKeown's talk "Making SDN Work" at the Open Networking Summit, April 2012

App

Windows (OS)

Linux

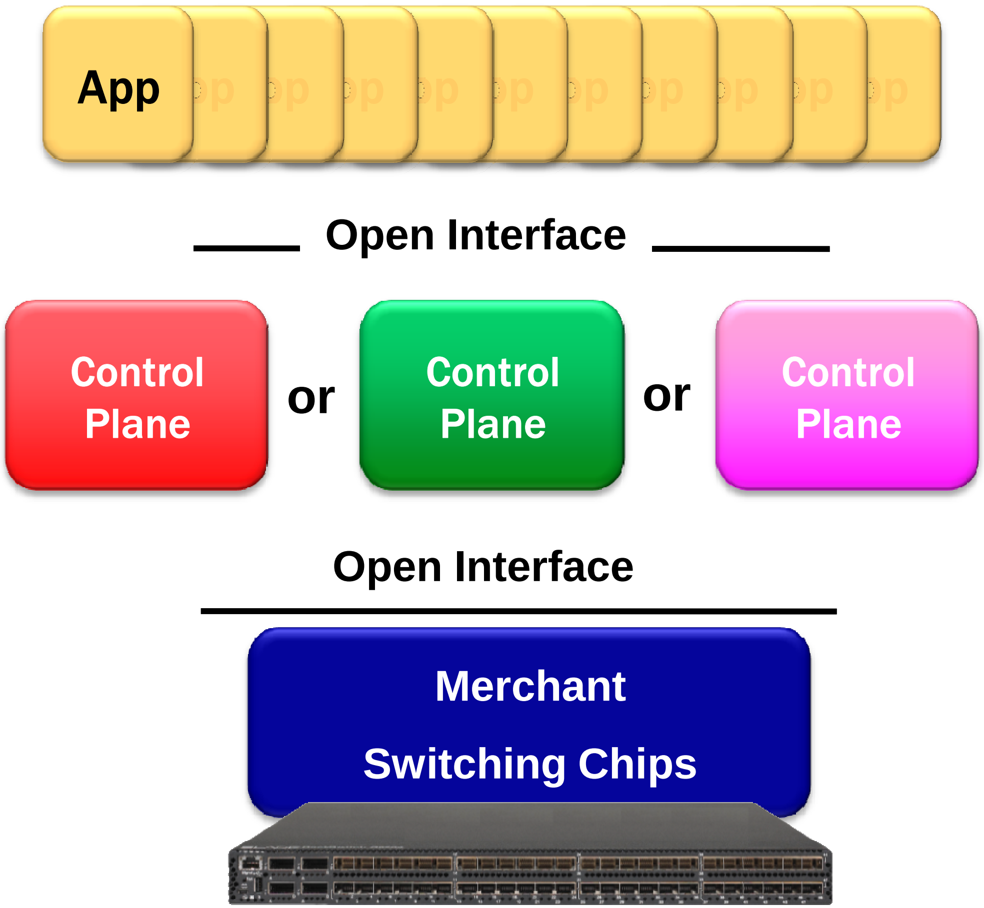
Mac OS

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

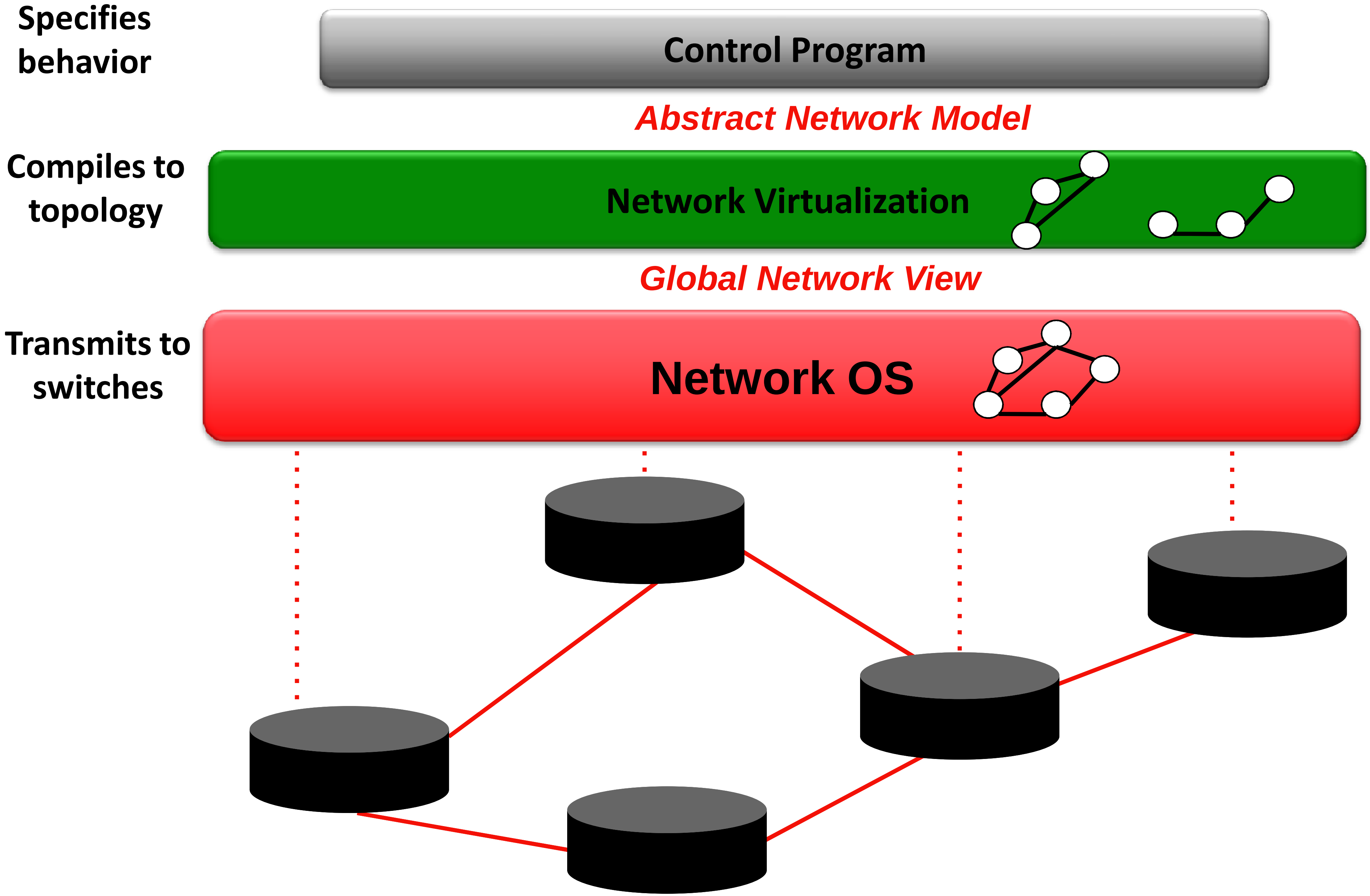
SND (r)evolution



Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

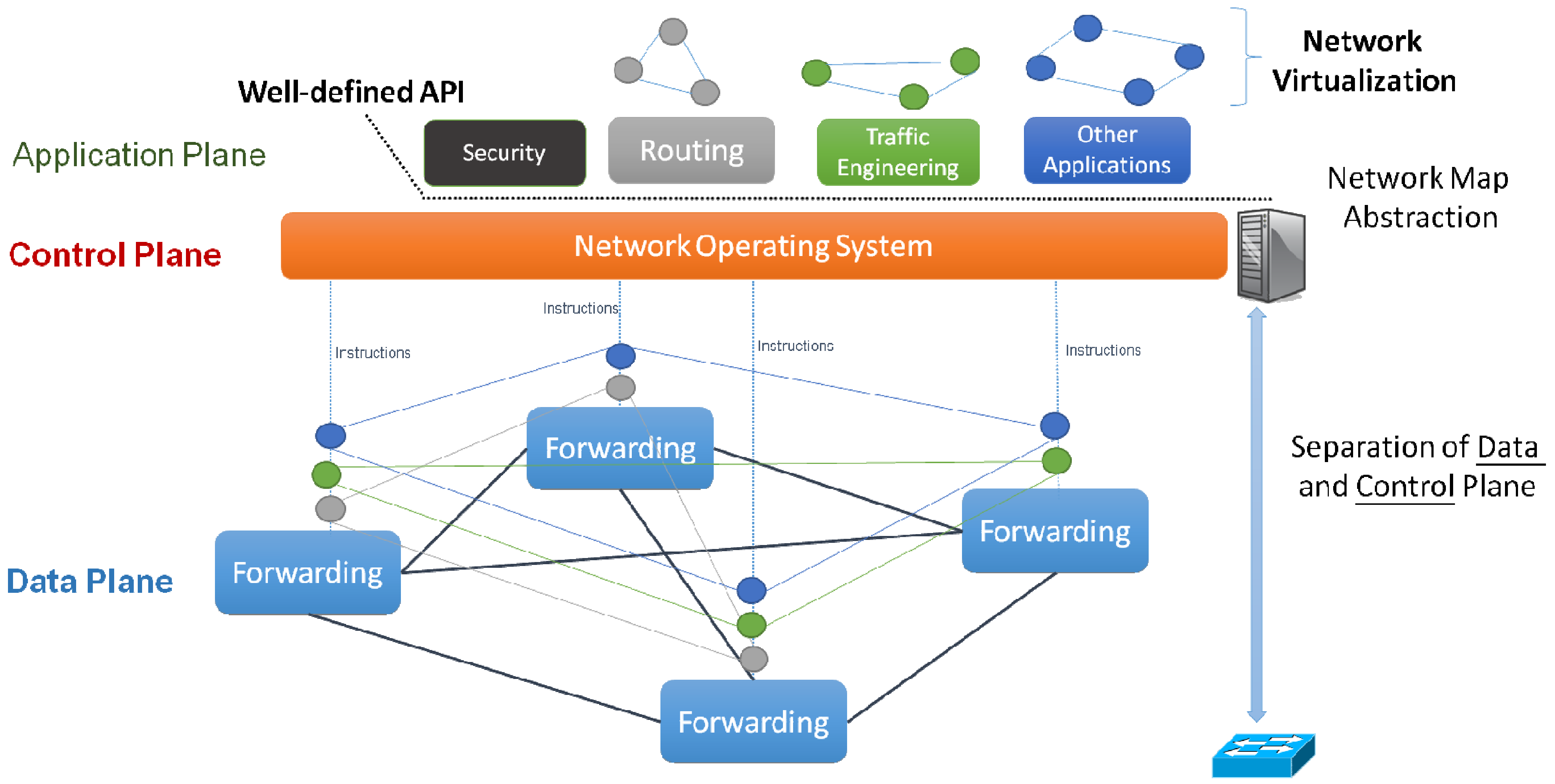
The “Software-Defined Network”



What does this picture mean?

- Write a simple program to configure a simple model
  - Configuration is merely a way to specify what you want
- Examples
  - ACLs: who can talk to who
  - Isolation: who can hear my broadcasts
  - Routing: only specify routing to the degree you care
    - Some flows over satellite, others over landline
  - TE: specify in terms of quality of service, not routes
- Virtualization layer “compiles” these requirements
  - Produces suitable configuration of actual network devices
- NOS then transmits these settings to physical boxes

SDN: the role of the Network Operating System



Two examples uses

- **Scale-out router:**
  - Abstract view is single router
  - Physical network is collection of interconnected switches
  - Allows routers to "scale out, not up"
  - Use standard routing protocols on top

- **Multi-tenant networks:**
  - Each tenant has control over their "private" network
  - Network virtualization layer compiles all of these individual control requests into a single physical configuration

- **Hard to do without SDN, easy *(in principle)* with SDN**

Does SDN work?

- Is it scalable? **Yes**
- Is it less responsive? **No**
- Does it create a single point of failure? **No**
- Is it inherently less secure? **No**
- Is it incrementally deployable? **Yes**

SDN: clean separation of concerns

- **Control program: specify behavior on abstract model**
  - Driven by **Operator Requirements**

- **Network Virtualization: map abstract model to global view**
  - Driven by **Specification Abstraction**

- **NOS: map global view to physical switches**
  - API: driven by **Distributed State Abstraction**
  - Switch/fabric interface: driven by **Forwarding Abstraction**

Where SDN is and will be deployed

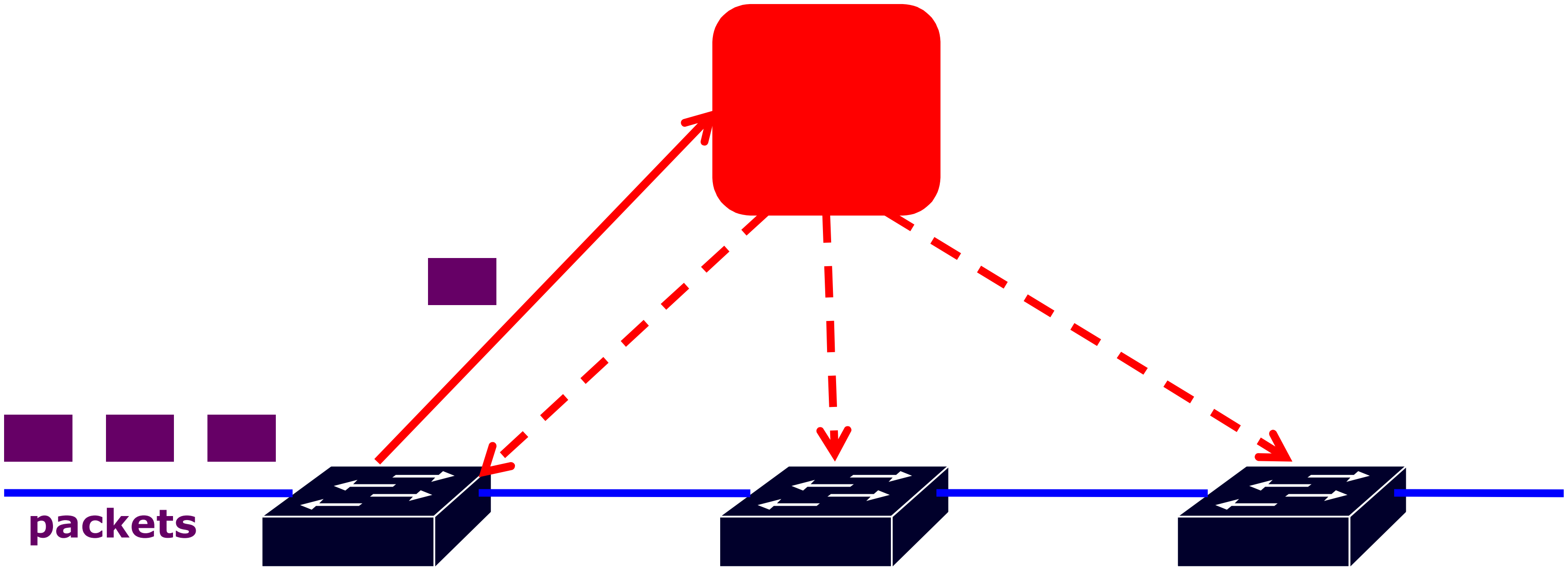
- Multi-tenant “virtualized” data centers
  - Public and private clouds
- WANs
  - Google WAN
  - Eventually, public WANs
- Enterprise networks
  - Greater control, fewer middleboxes

Where SDN is and will be deployed (2)

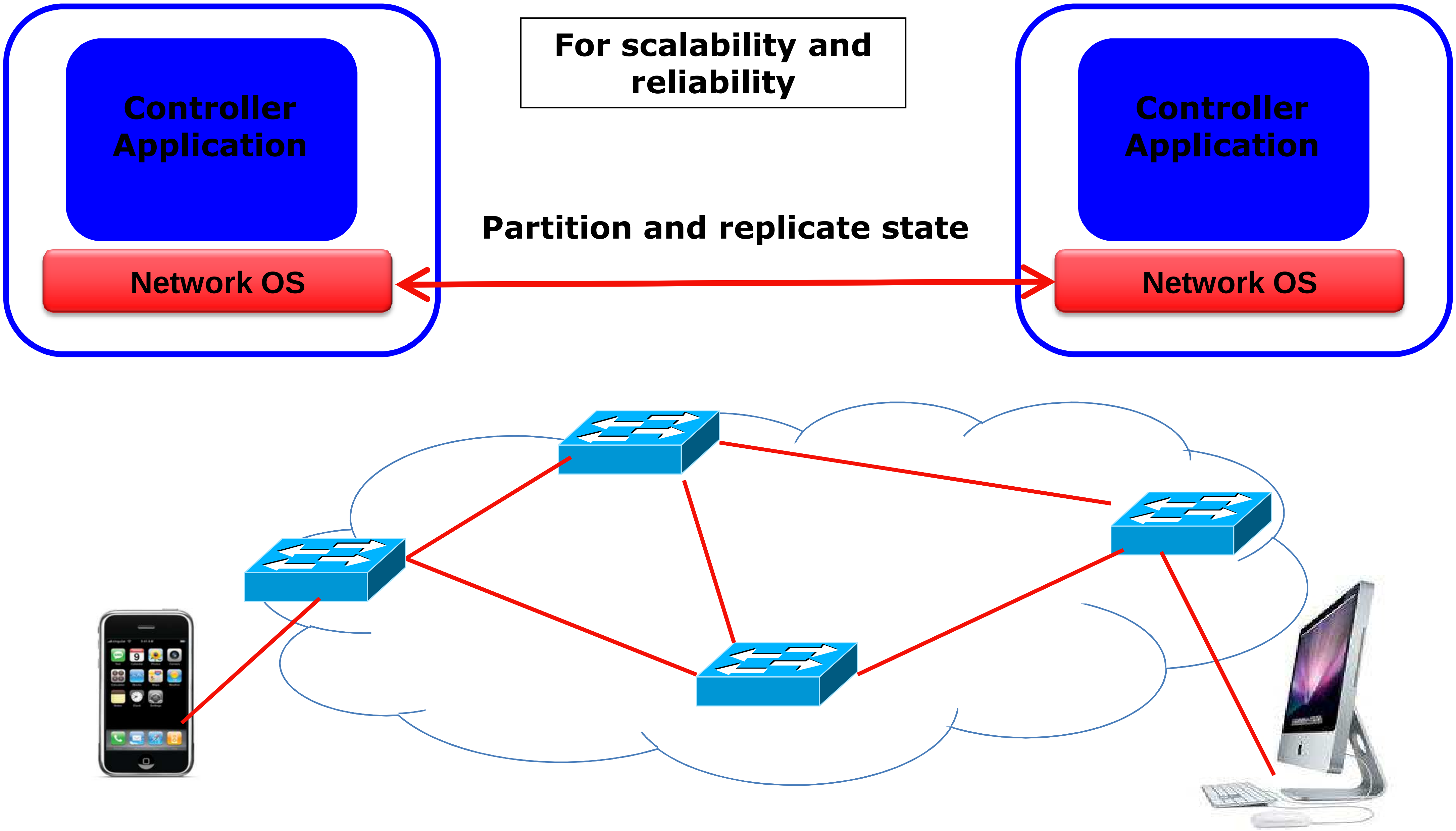
- Home networks
  - Outsourced management
- Cellular Networks
  - Separation of service from physical infrastructure
- Research and Education Networks
  - National backbones
  - College campus networks

Challenge: 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”



SDNs with distributed controllers

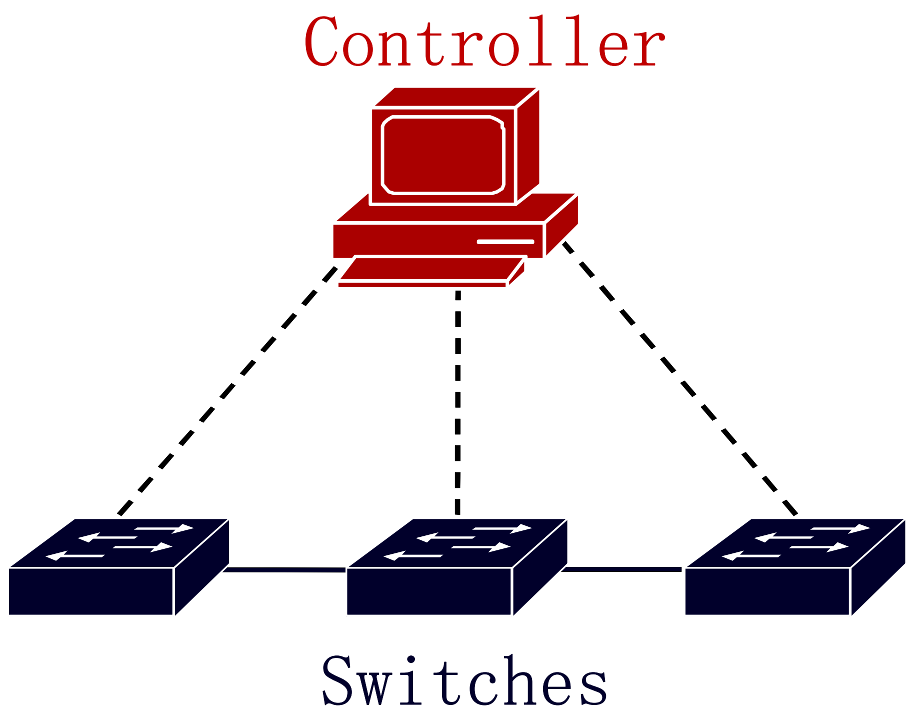


Challenge: testing and debugging

- 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

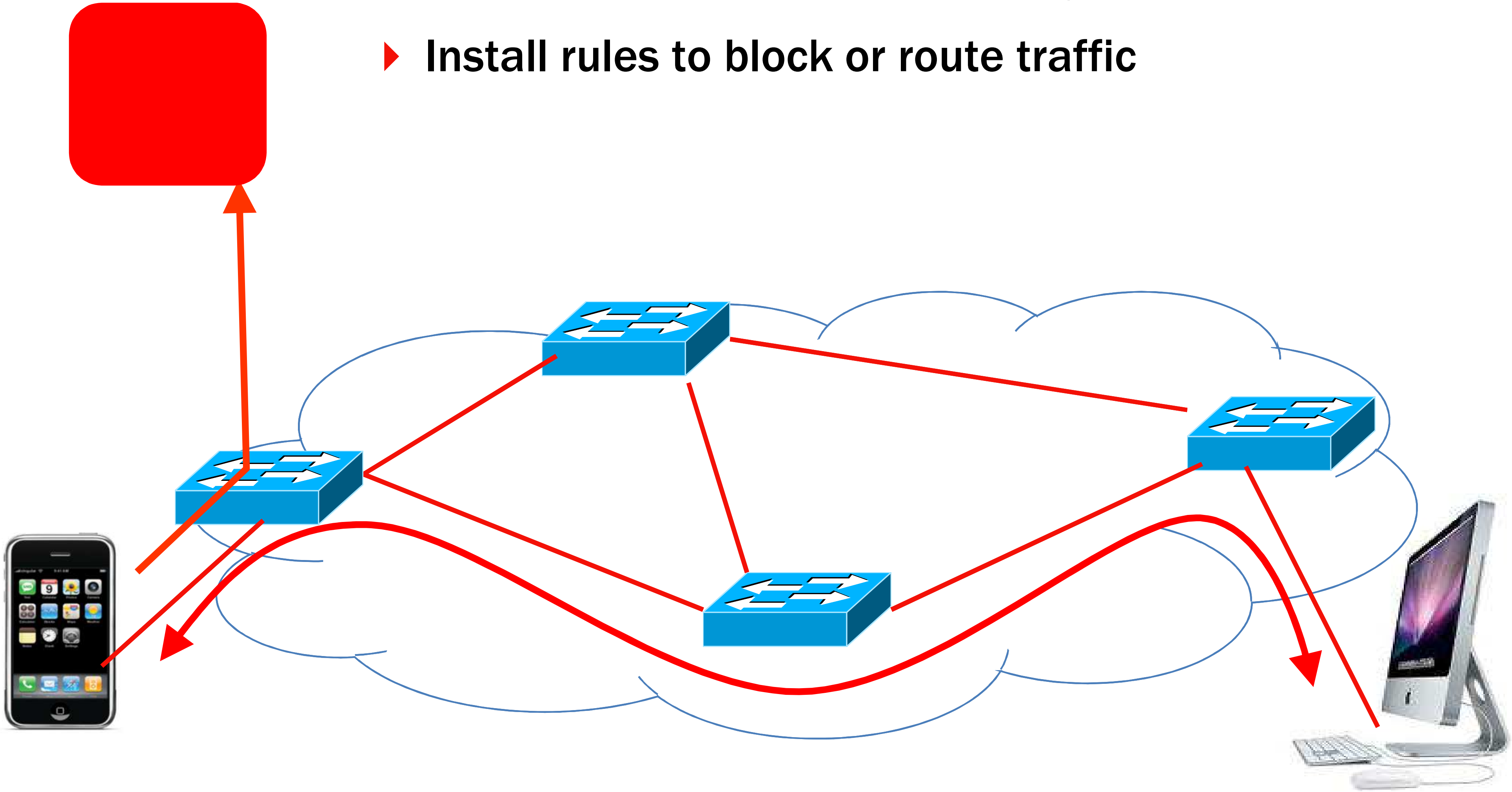
Challenge: programming abstractions

- Controller APIs are low-level
  - Thin layer on top of the underlying hardware
- Need better languages
  - Composition of modules
  - Managing concurrency
  - Querying network state
  - Network-wide abstractions



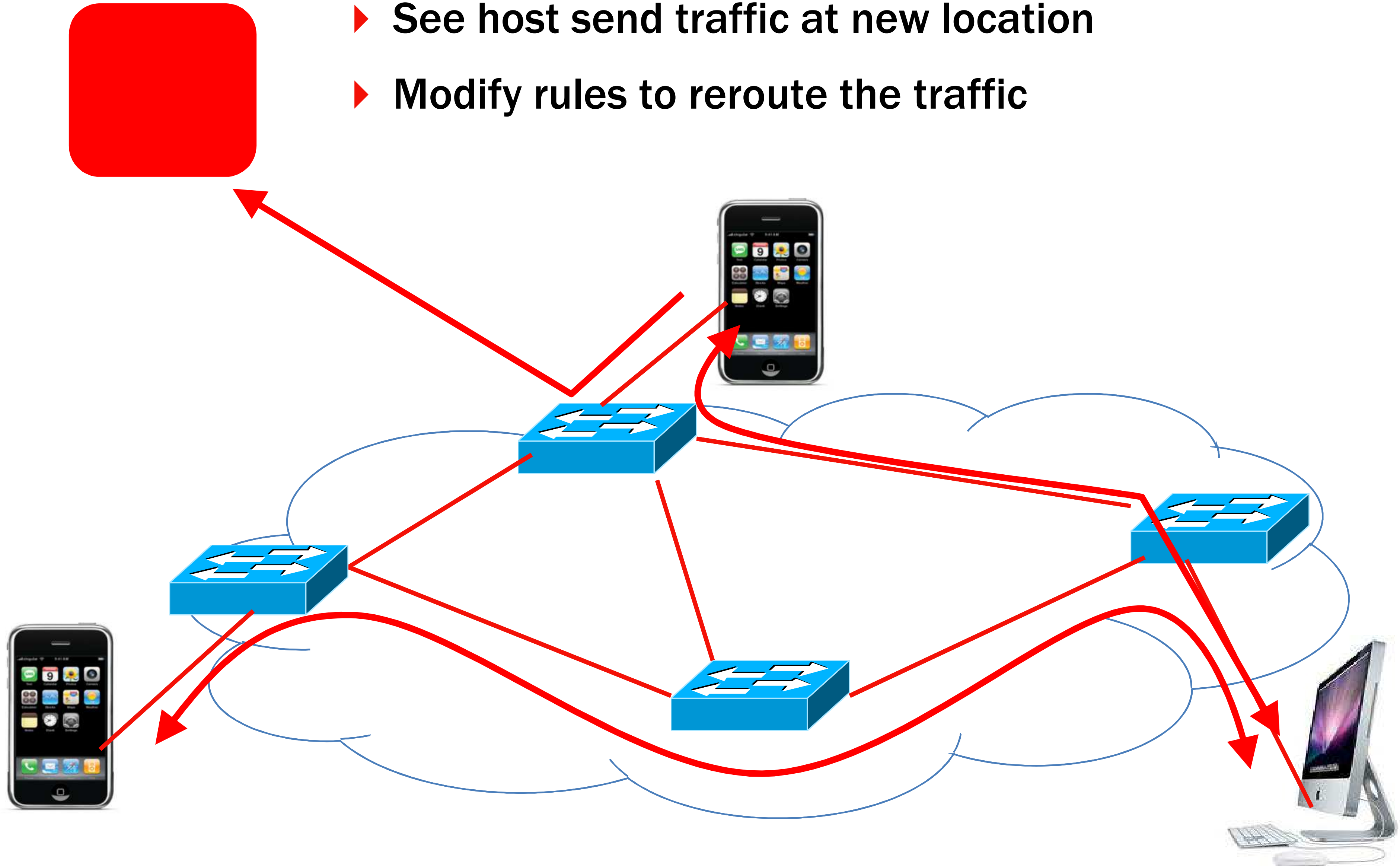
SDN use case: Dynamic Access Control

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



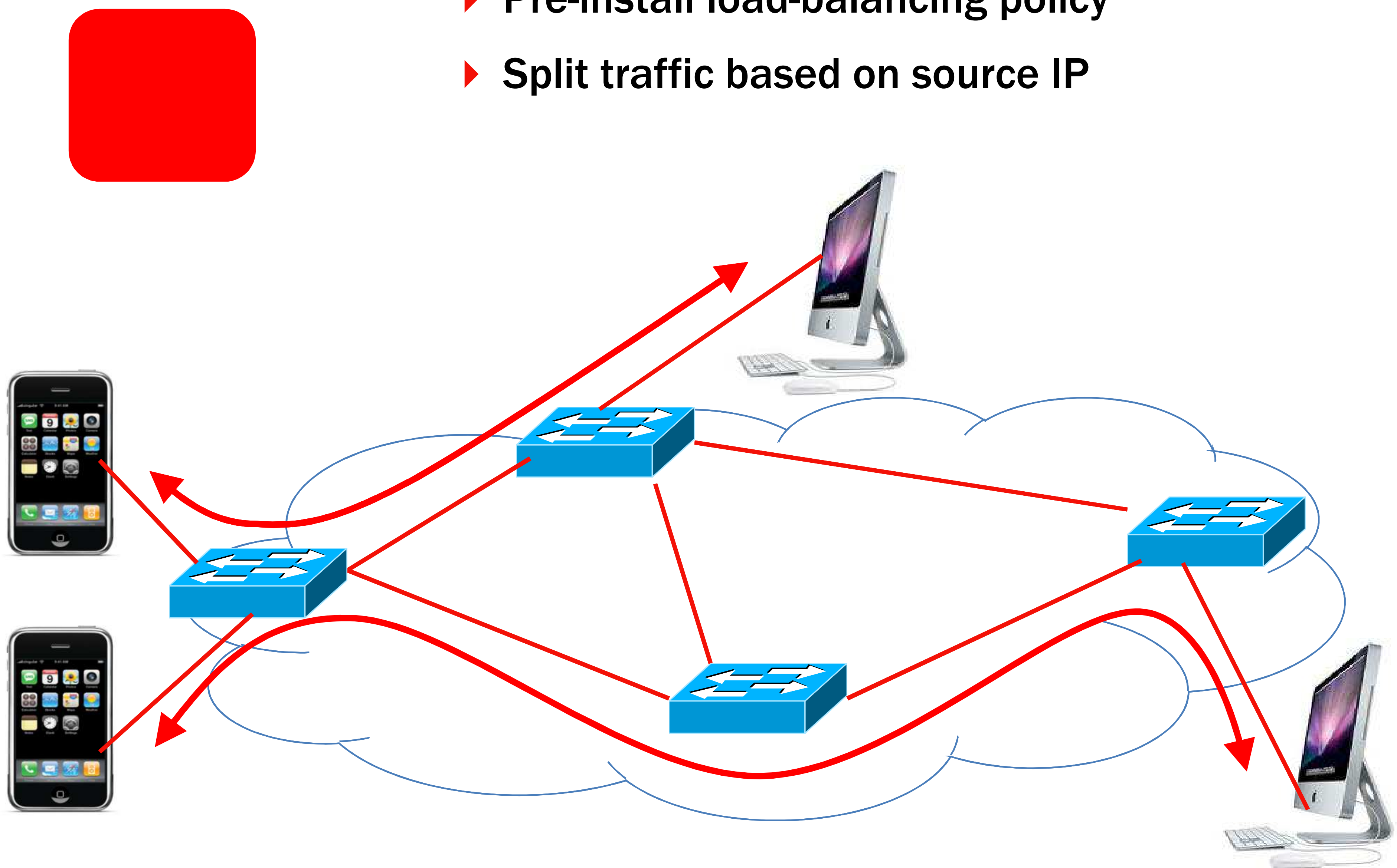
SDN use case: Seamless Mobility/Migration

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



SDN use case: Server Load Balancing

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



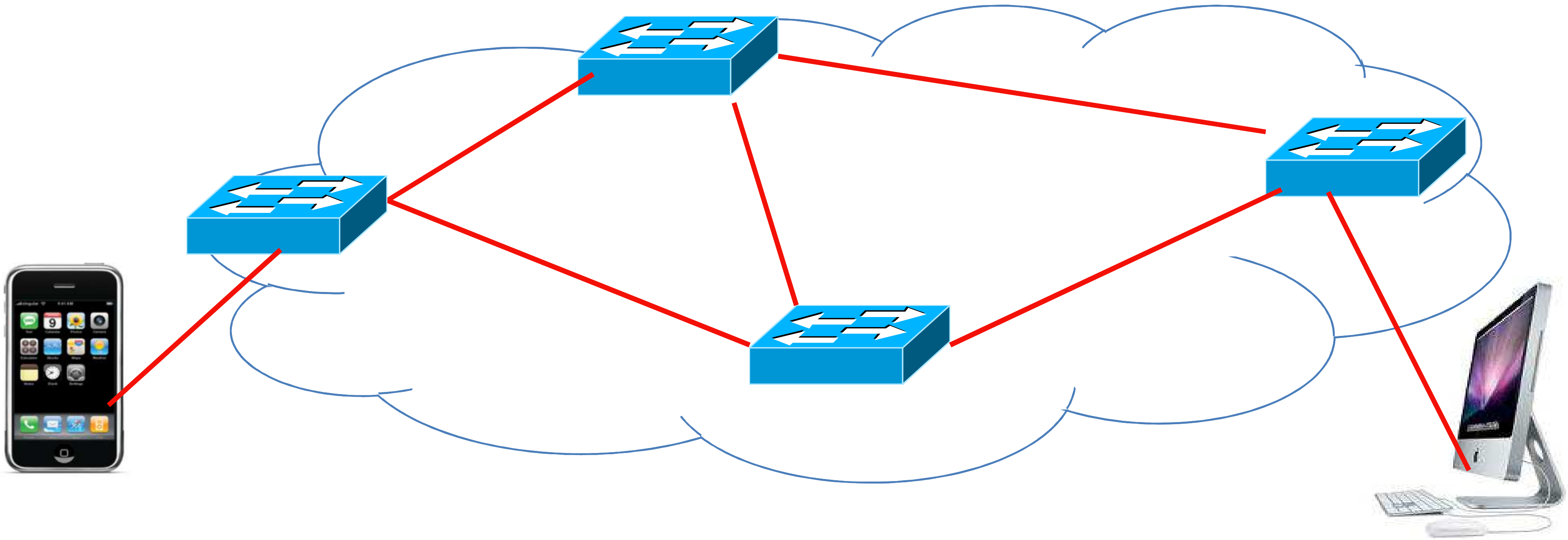
SDN use case: Network Virtualization

Controller #1

Controller #2

Controller #3

Partition the space of packet headers



SDN in Real World – Google's Story

- The industries were skeptical whether SDN was possible
- Google had big problems:
  - High financial cost managing their datacenters: Hardware and software upgrade, over provisioning (fault tolerant), manage large backup traffic, time to manage individual switch, and a lot of men power to manage the infrastructure
  - Delay caused by rebuilding connections after link failure
    - Slow to rebuild the routing tables after link failure
    - Difficult to predict what the new network may perform
- Google went ahead and implemented SDN
  - Built their hardware and wrote their own software for their internal datacenters
  - Surprised the industries when Google announced SDN was possible in production
- How did they do it?
  - "*B4: Experience with a Globally-Deployed Software Defined WAN*", ACM SIGCOMM 2013

References

- *OpenFlow: Enabling innovation in campus networks*.
  Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner.
  ACM SIGCOMM Computer Communication Review. Volume 38 Issue 2, April 2008
- *Origins and Evolution of OpenFlow/SDN*.
  Martin Casado.
  Open Networking Summit, Stanford, CA, October 2011. (Video available on YouTube)
- *Software-Defined Networking: A Comprehensive Survey*.
  Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig.
  Proceedings of the IEEE, vol.103, no.1, pp.14–76, Jan. 2015
- *B4: Experience with a Globally-Deployed Software Defined WAN*.
  Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart and Amin Vahdat.
  ACM SIGCOMM Computer Communication Review. Volume 43 Issue 4, October 2013
- Open Network Foundation.
  http://opennetworking.org
- IEEE SDN Technical Committee.
  http://sdn.ieee.org/education