SAPIENZA
UNIVERSITÀ DI ROMA

SDN

What's Software Defined Networking?

Angelo Capossele

Outline

- Introduction to SDN
- OpenFlow
- Network Functions Virtualization
- Some examples
- Opportunities
- Research problems
- Security
- Case study: LTE
- (Mini)Tutorial on Controller programming

The Problem

Networks are complicated

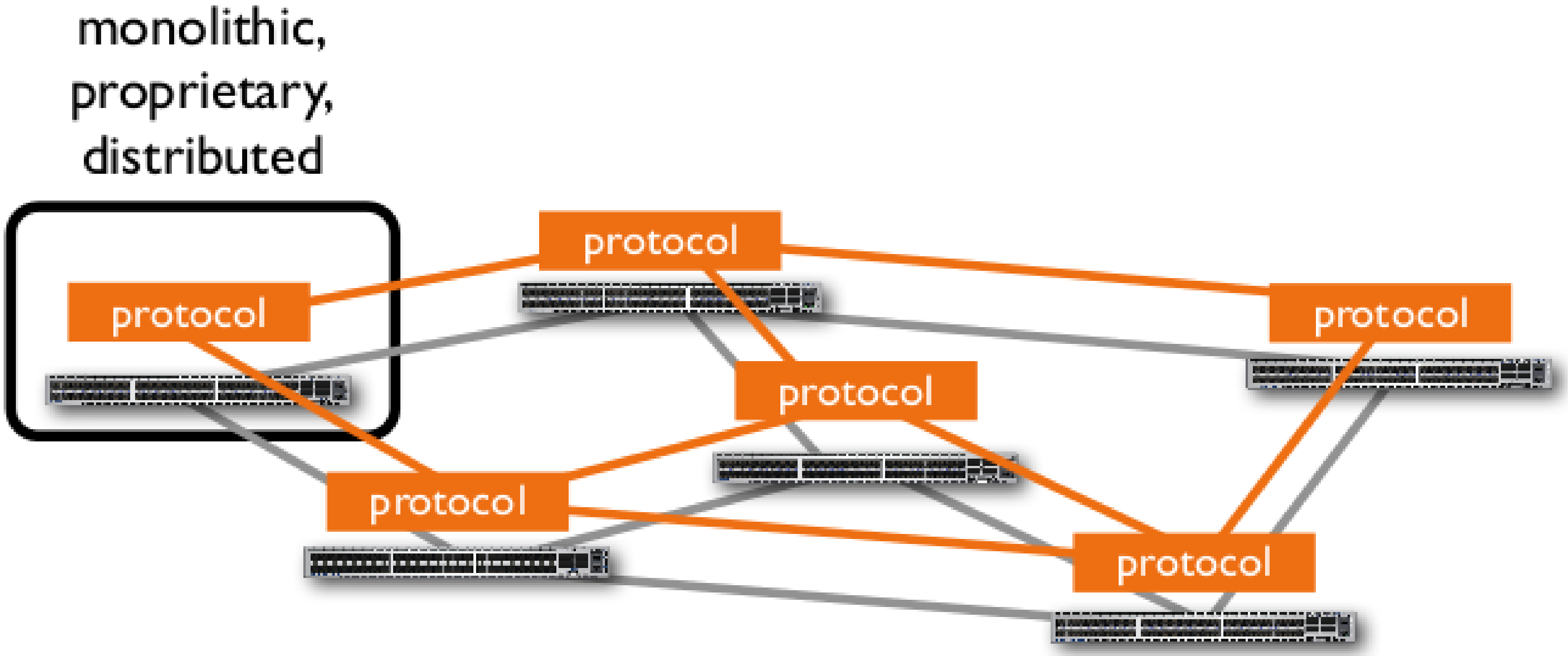
- Just like any computer system
- Worse: it’s distributed
- Even worse: no clean programming APIs, only “knobs and dials”

Network equipment is proprietary

- Integrated solutions (software, configuration, protocol implementations, hardware) from major vendors (Cisco, Juniper, etc.)

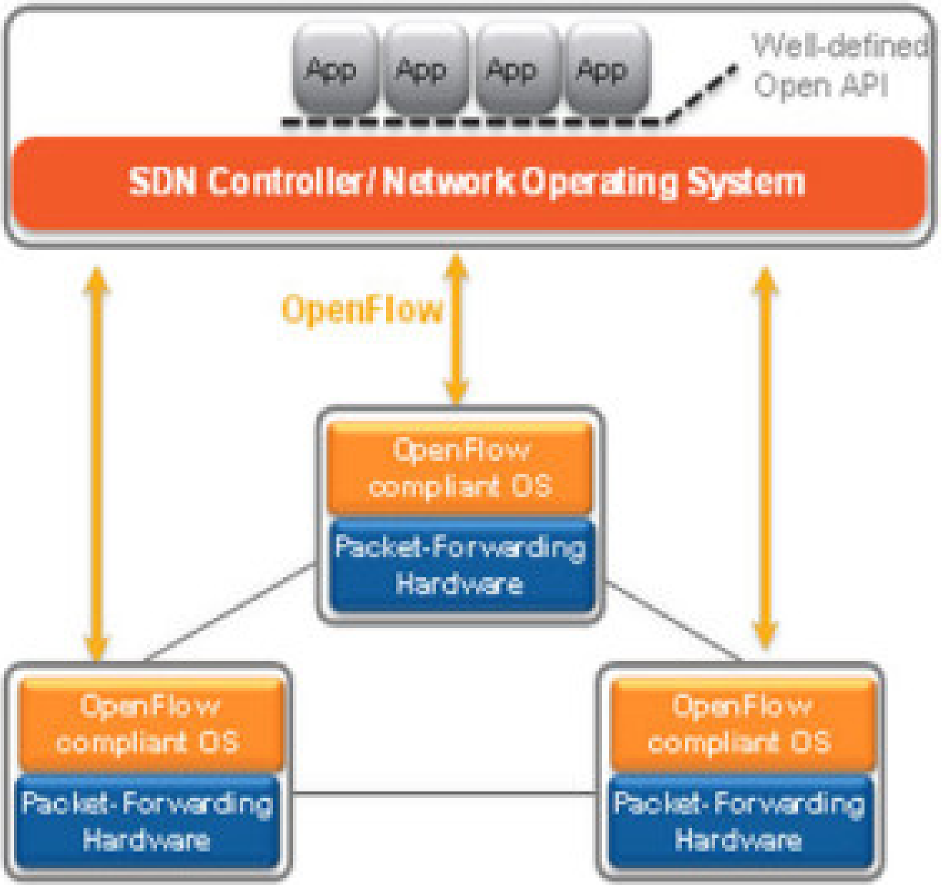
Result: Hard to innovate and modify networks

Traditional networking

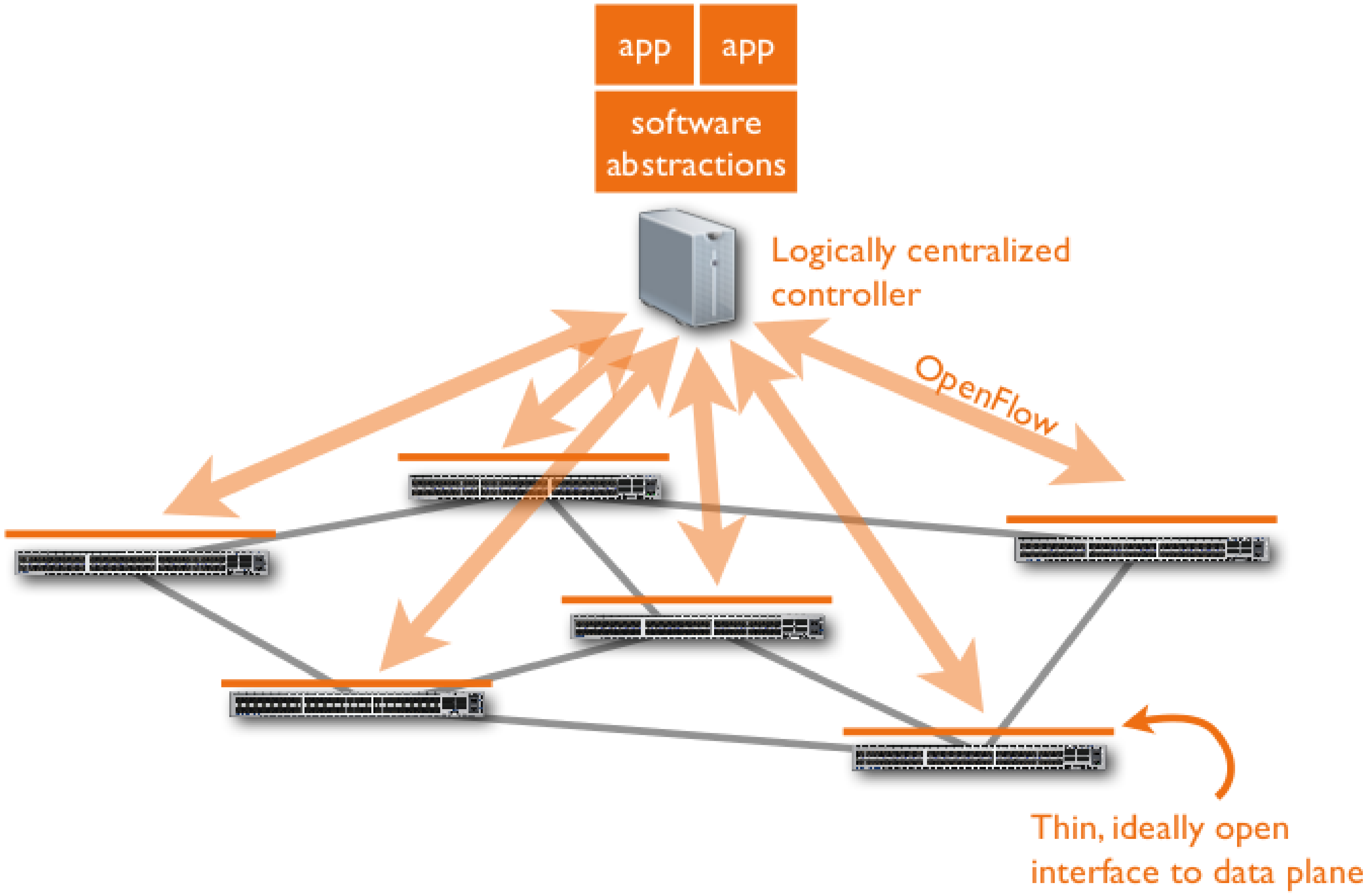


What is Software Defined Networking

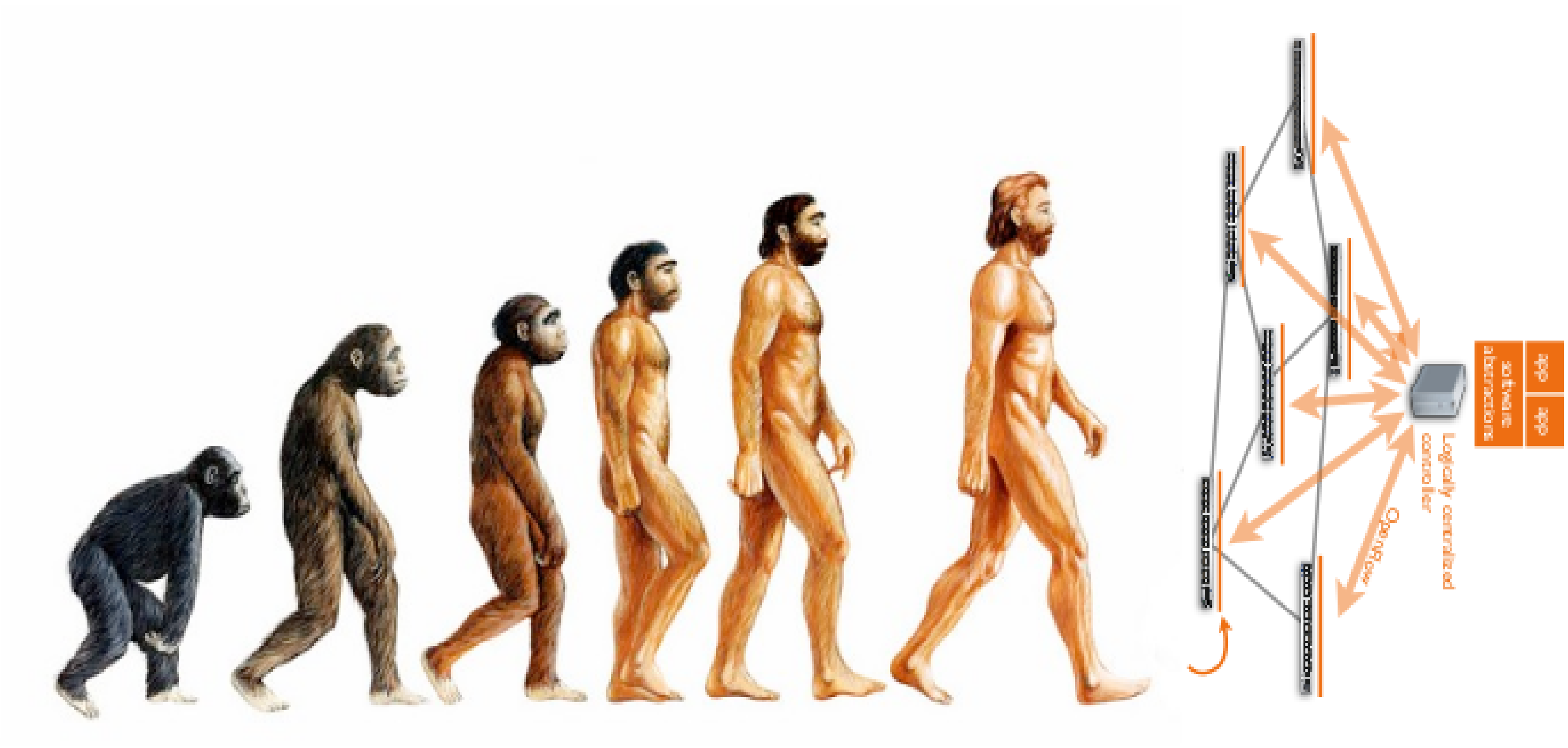
- A new approach to designing, building and managing networks
- The basic concept is that SDN separates the network's control (brains) and forwarding (muscle) planes to make it easier to optimize each.


App
App
App
App
Well-defined Open API
SDN Controller/ Network Operating System
OpenFlow
OpenFlow compliant OS
Packet-Forwarding Hardware
OpenFlow compliant OS
Packet-Forwarding Hardware
OpenFlow compliant OS
Packet-Forwarding Hardware

Software Defined Networking

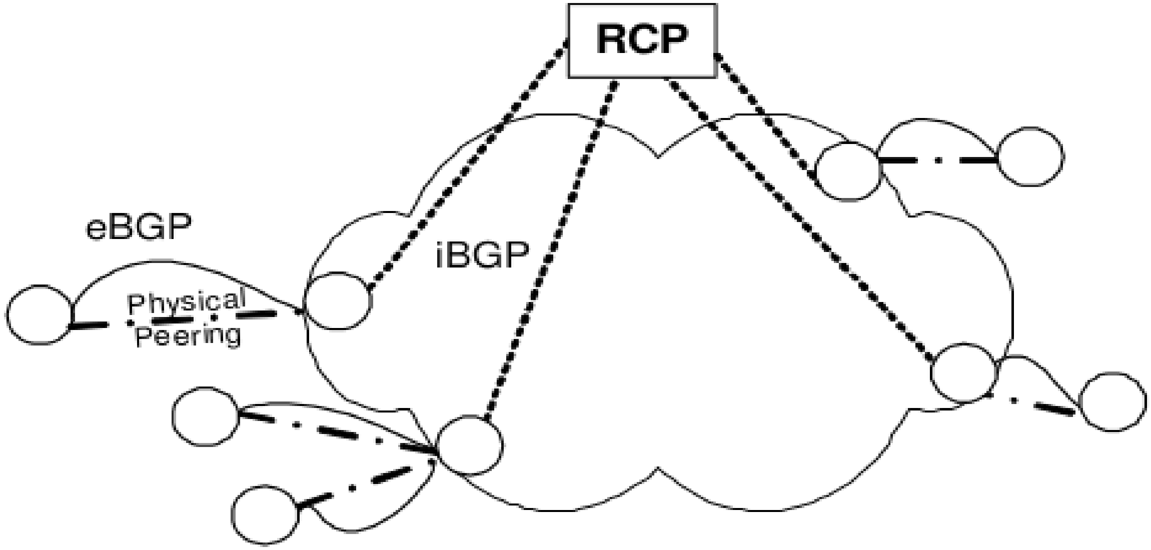

app
app
software abstractions
Logically centralized controller
OpenFlow
Thin, ideally open interface to data plane

Evolution of SDN



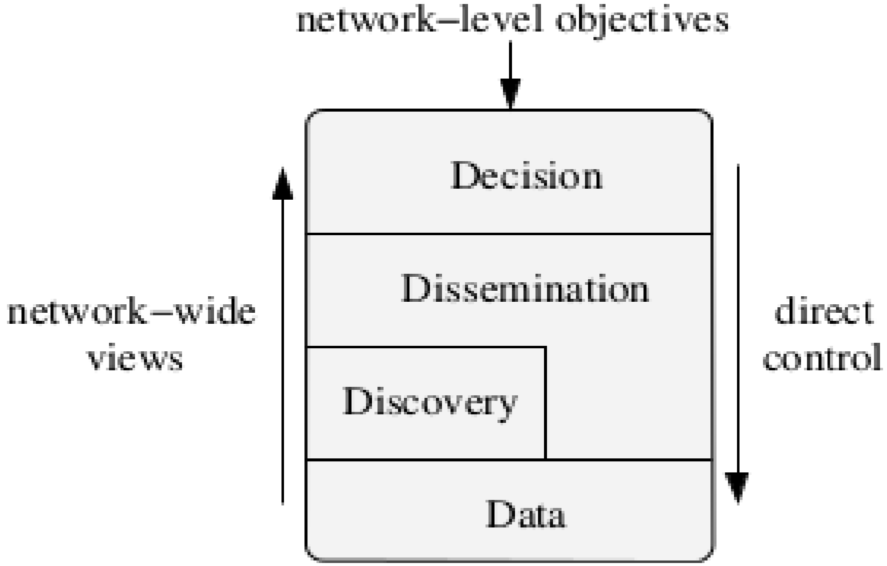
Evolution of SDN

- Routing Control Platform (2005)
  - [Caesar, Caldwell, Feamster, Rexford, Shaikh, van der Merwe, NSDI 2005]
  - Centralized computation of BGP routes, pushed to border routers via iBGP



Evolution of SDN

- 4D architecture (2005)
  - A Clean Slate 4D Approach to Network Control and Management [Greenberg, Hjalmtysson, Maltz, Myers, Rexford, Xie,Yan, Zhan, Zhang, CCR Oct 2005]
  - Logically centralized “decision plane” separated from data plane



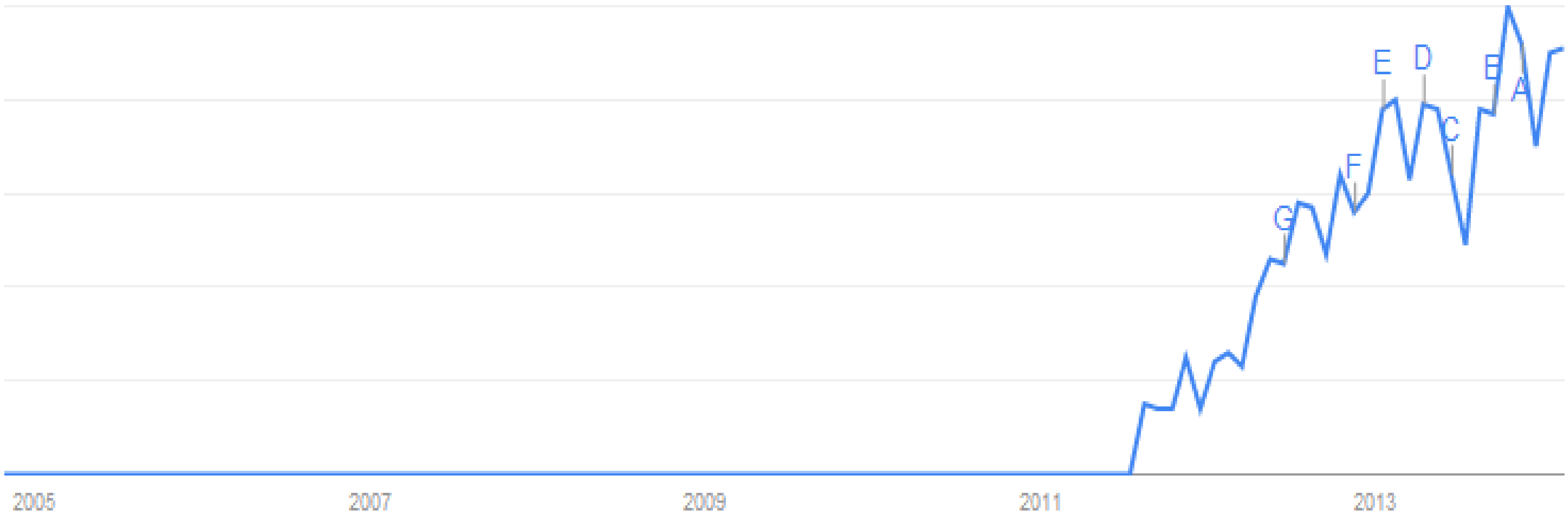
Evolution of SDN

- Ethane (2007)
  - [Casado, Freedman, Pettit, Luo, McKeown, Shenker, SIGCOMM 2007]
  - Centralized controller enforces enterprise network Ethernet forwarding policy using existing hardware

Evolution of SDN

- OpenFlow (2008)
  - [McKeown, Anderson, Balakrishnan, Parulkar, Peterson, Rexford, Shenker, Turner, CCR 2008]
  - Thin, standardized interface to data plane
  - General-purpose programmability at controller
- NOX (2008)
  - First OF controller: centralized network view provided to multiple control apps as a database

SDN trend



- Startup
- Accademic research

IBM

CISCO

HUAWEI

intel

vmware

redhat

Microsoft

Software Defined Networking

- A controller is a software program that sends and receives OpenFlow from network devices.

- The controller sends OpenFlow entries for the forwarding table

- Because the controller must compute the *flow paths* in software this is usually known as **SOFTWARE DEFINED NETWORKING**.
  The controller sends OpenFlow entries for the forwarding table

- Abbreviated to “**SDN**”

Controller Concepts

- Controller drives a level of network convergence that was previously unimaginable.
- Consider changing all the configuration on your network to support new network path every 10 minutes ?
- Todays’ tools cannot do that.
- SNMP can’t do configuration. CLI programming is too diverse between vendors (and no standards will ever solve that).

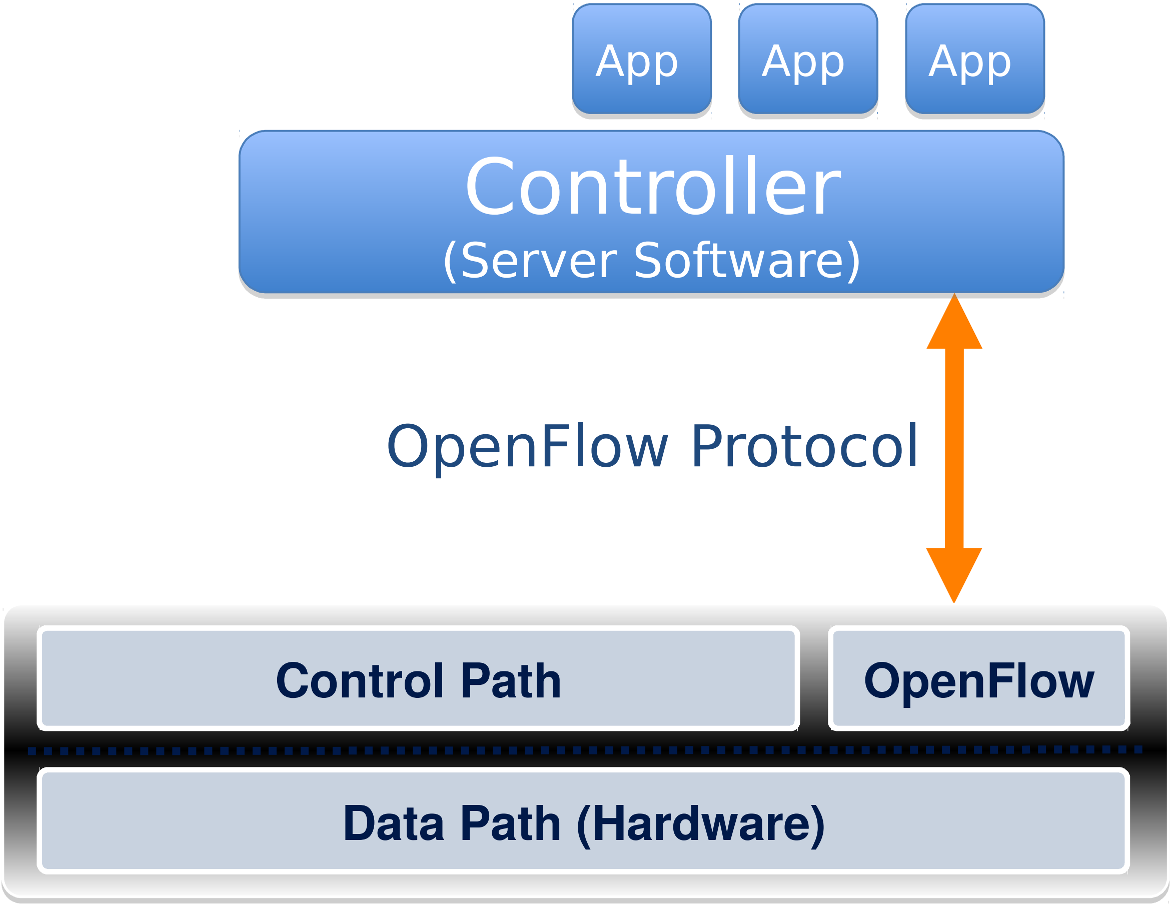
Why Software Defined Networking

- **Separate hardware from software**
  - Choose hardware based on necessary features
  - Choose software based on protocol requirements
- **Logically centralized network control**
  - More deterministic
  - More efficient
  - More fault tolerant
- **Separate monitoring, management, and operation from individual boxes**
- **Flexibility and Innovation**

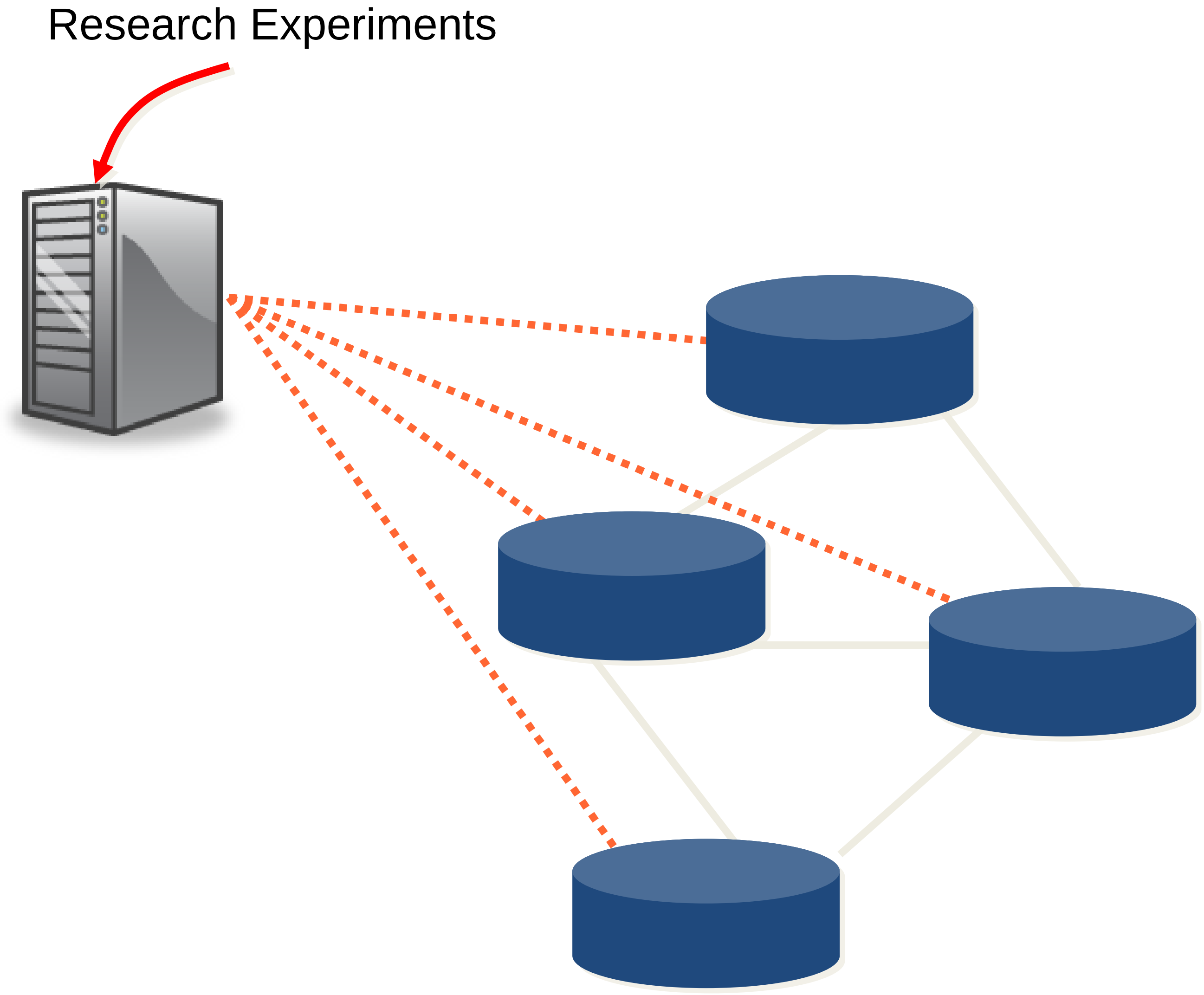
SDN is not OpenFlow

- Often people point to OpenFlow as being synonymous with SDN, but it is only a single element in the overall SDN architecture.
- OpenFlow is an open standard for a communications protocol that enables the control plane to interact with the forwarding plane
- Alternative protocols (see ONOS)

OpenFlow


App
App
App
Controller
(Server Software)
OpenFlow Protocol
Control Path
OpenFlow
Data Path (Hardware)

Separate Control from Datapath

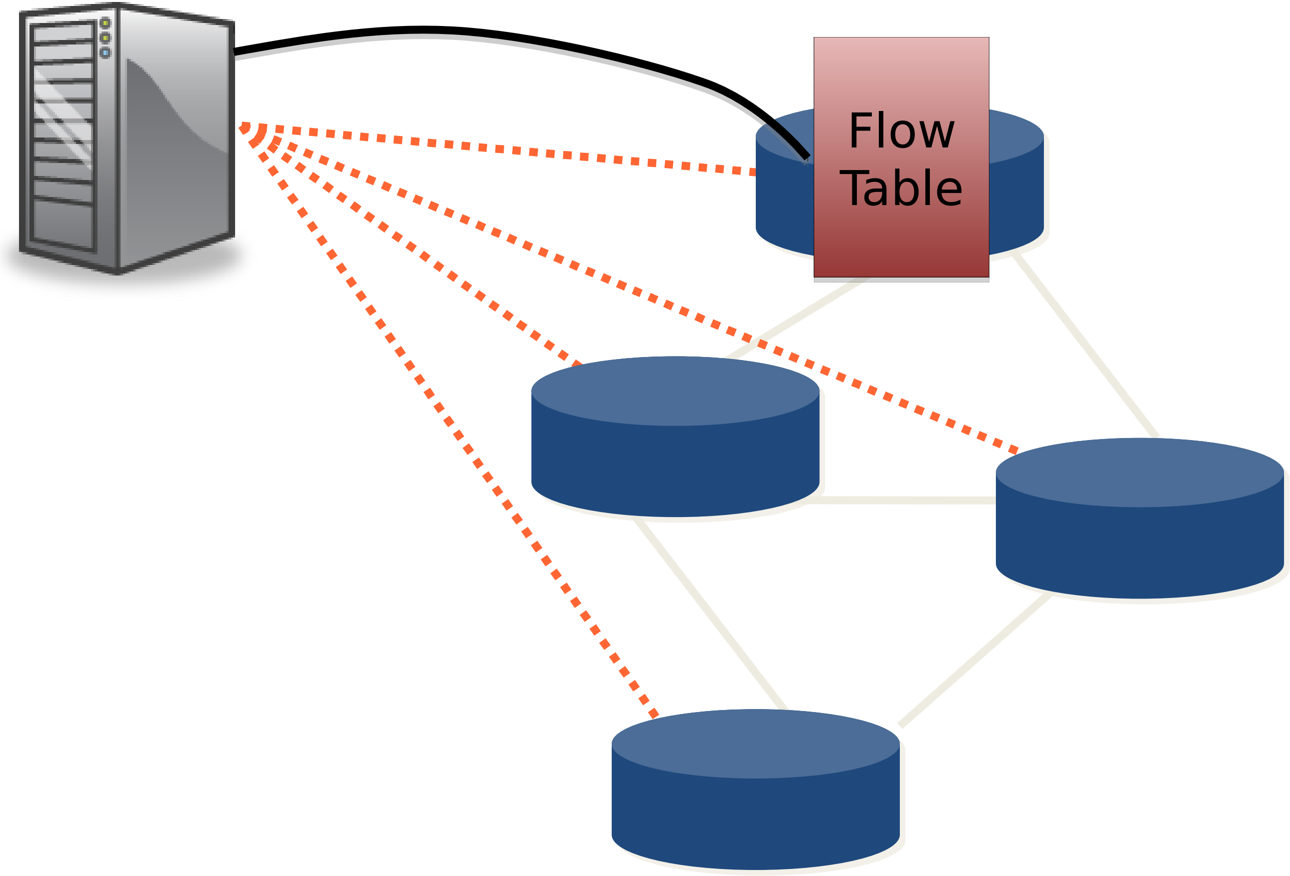


Cache flow decisions in datapath

“If header = *x*, send to port 4”

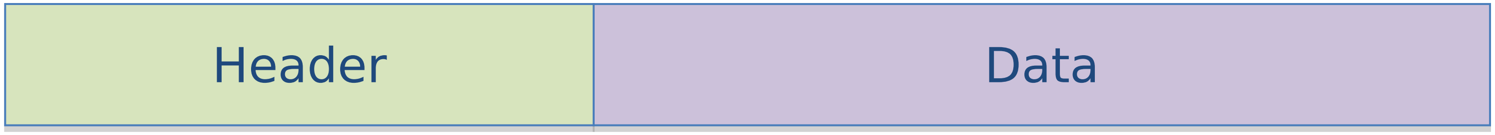
“If header = *y*, overwrite header with *z*, send to ports 5,6”

“If header = ?, send to me”



<Match, Action>

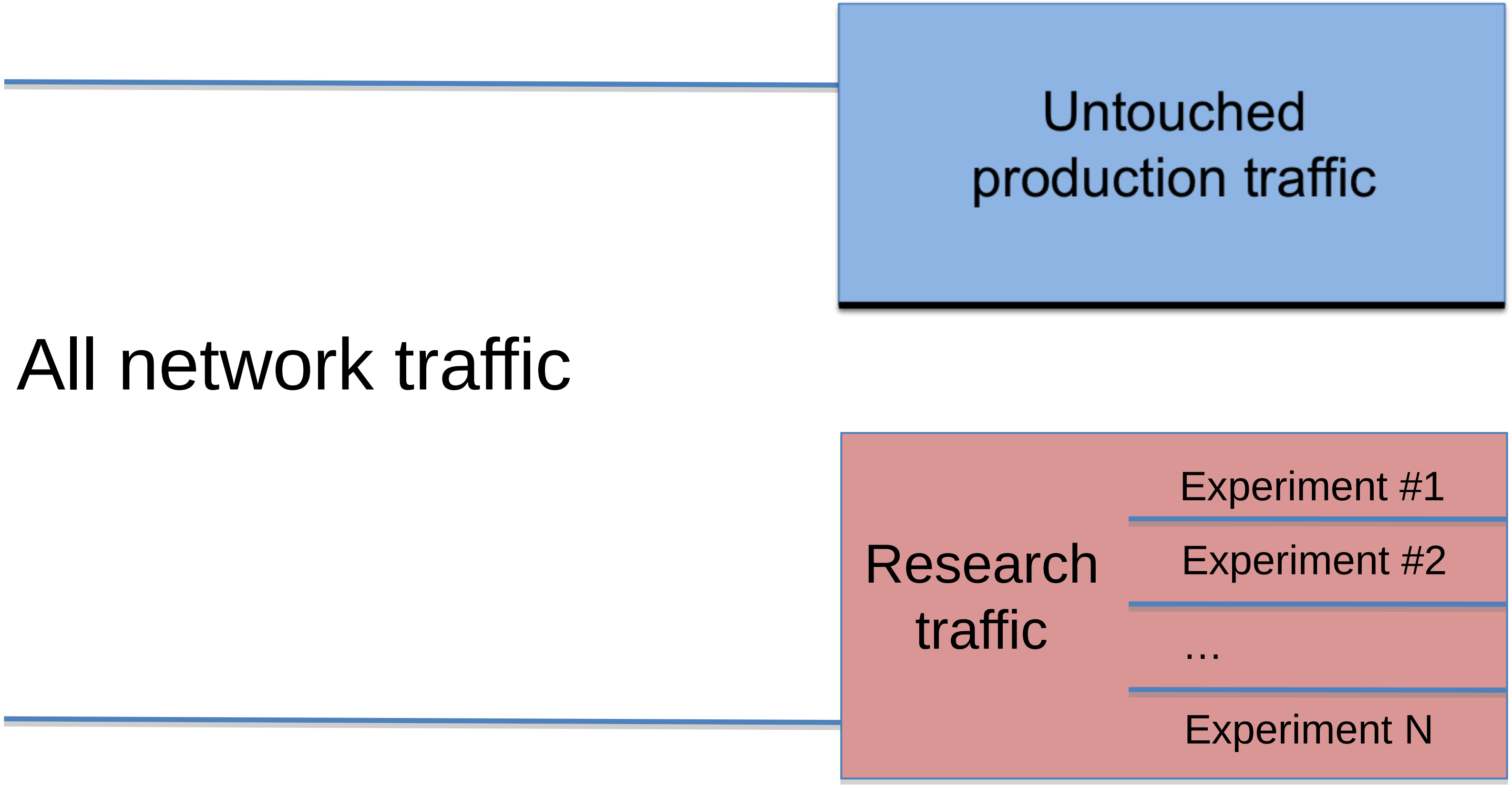
- Match arbitrary bits in headers:



Match: 1000x01xx0101001x

  - Match on any header, or new header
  - Allows any flow granularity
- Action
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, push or pop
  - Forward at specific bit-rate

Slicing Traffic



All network traffic

Untouched production traffic

Research traffic

Experiment #1

Experiment #2

…

Experiment N

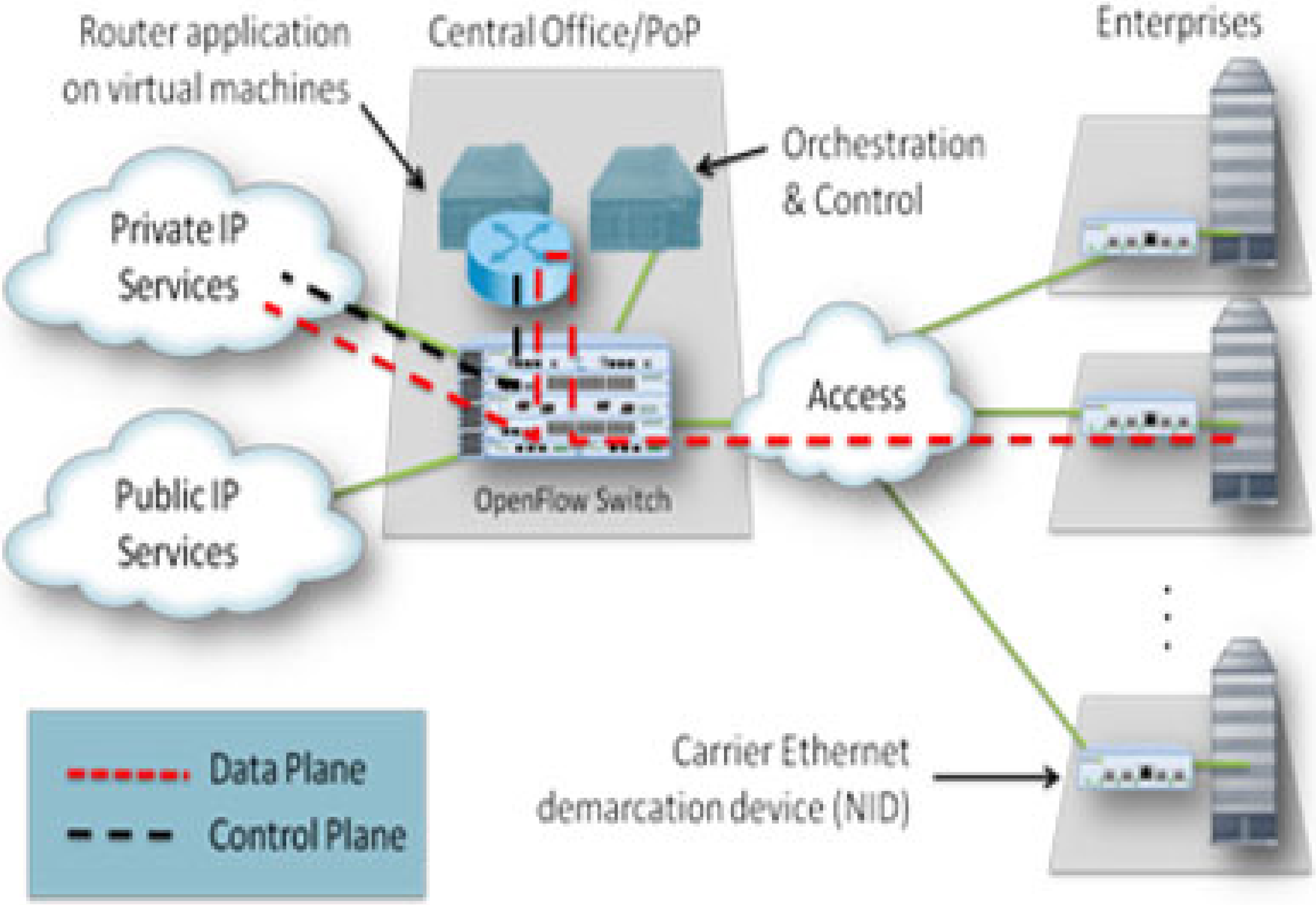
SDN vs NFV

- Software-Defined Networking (SDN), Network functions Virtualization (NFV) and Network Virtualization (NV), are all complementary approaches.
- They each offer a new way to design deploy and manage the network and its services:
  - SDN separates the network's control (brains) and forwarding (muscle) planes and provides a centralized view of the distributed network for more efficient orchestration and automation of network services.
  - NFV focuses on optimizing the network services themselves. NFV decouples the network functions, such as DNS, Caching, etc., from proprietary hardware appliances, so they can run in software to accelerate service innovation and provisioning, particularly within service provider environments.
  - NV ensures the network can integrate with and support the demands of virtualized architectures, particularly those with multi-tenancy requirements.

Network Functions Virtualization (NFV)

- NFV decouples the network functions (NAT, firewalling, IDS, DNS, caching, etc.), from proprietary hardware appliances, so they can run in software.
- It utilizes standard IT virtualization technologies that run on high-volume service, switch and storage hardware to virtualize network functions.
- It is applicable to any data plane processing or control plane function in both wired and wireless network infrastructures.

How a Managed Router Service Would be Deployed with NFV



SDN and NFV Are Better Together

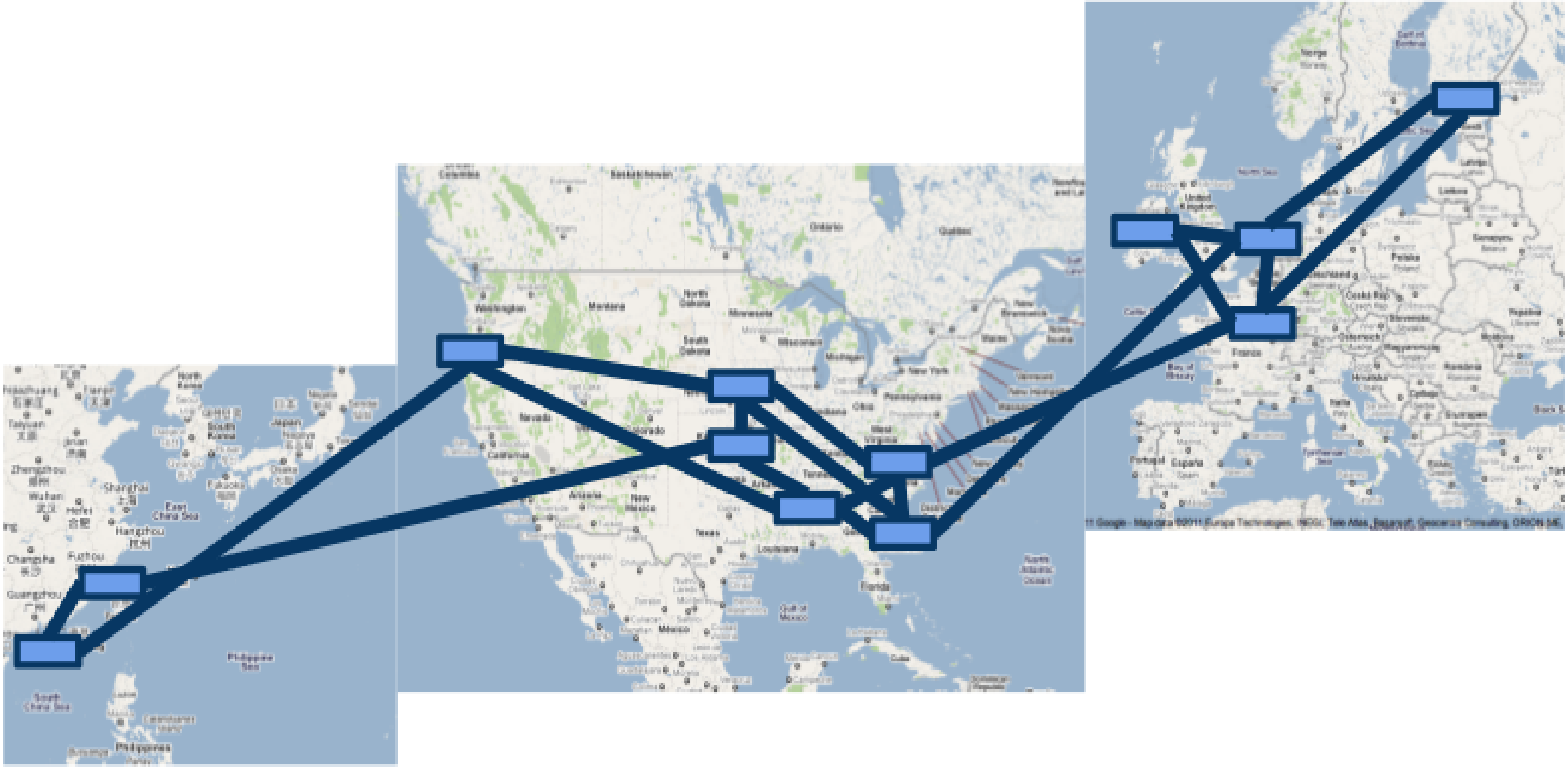
- These approaches are mutually beneficial, but are not dependent on one another
- SDN contributes network automation that enables policy-based decisions to orchestrate which network traffic goes where
- NFV focuses on the services, and NV ensures the network’s capabilities align with the virtualized environments they are supporting

- Move functionality to software
- Use commodity servers and switches over proprietary appliances
- Leverage programmatic application interfaces (APIs)
- Support more efficient orchestration, virtualization and automation of network services

Google WAN

- Two backbones
  - Internet facing (user traffic)
  - Datacenter traffic (internal)
- Widely varying requirements: loss sensitivity, availability, topology, etc.
- Widely varying traffic characteristics: smooth/diurnal vs. bursty/bulk
- Therefore: built two separate logical networks
  - I-Scale (bulletproof)
  - G-Scale (possible to experiment)

Google's OpenFlow WAN



Google SDN Experiences

- Much faster iteration time: deployed production-grade centralized traffic engineering in two months
  - fewer devices to update
  - much better testing ahead of rollout
- Simplified, high fidelity test environment
  - Can emulate entire backbone in software
- Hitless SW upgrades and new features
  - No packet loss and no capacity degradation
  - Most feature releases do not touch the switch

Google's conclusion

- OpenFlow is ready for real-world use
- SDN is ready for real-world use
  - Enables rapid rich feature deployment
  - Simplifies network management
- Google's datacenter WAN successfully runs on OpenFlow
  - Largest production network at Google
  - Improved manageability
  - Improved cost (too early to have exact numbers)

Opportunities

- Open data plane interface
  - Hardware: easier for operators to change hardware, and for vendors to enter market
  - Software: can finally directly access device behavior
- Centralized controller
  - Direct programmatic control of network
- Software abstractions on the controller
  - Solve distributed systems problems only once, then just write algorithms
  - Libraries/languages to help programmers write net apps

Challenges

- Scalability (controller is bottleneck)
- Single point of failure (or small number)
- Latency to controller
- Needs new hardware or software
- Distributed system challenges still present
  - Imperfect knowledge of network state
  - Consistency issues between controllers
- Security

Ideas?

- Cloud virtualization
  - Create separate virtual networks for tenants
  - Allow flexible placement and movement of VMs
- WAN traffic engineering
  - Drive utilization to near 100% when possible
  - Protect critical traffic from congestion
- Key characteristics of the above
  - Special-purpose deployments with less diverse hardware
  - Existing solutions aren’t just annoying, they don’t work!

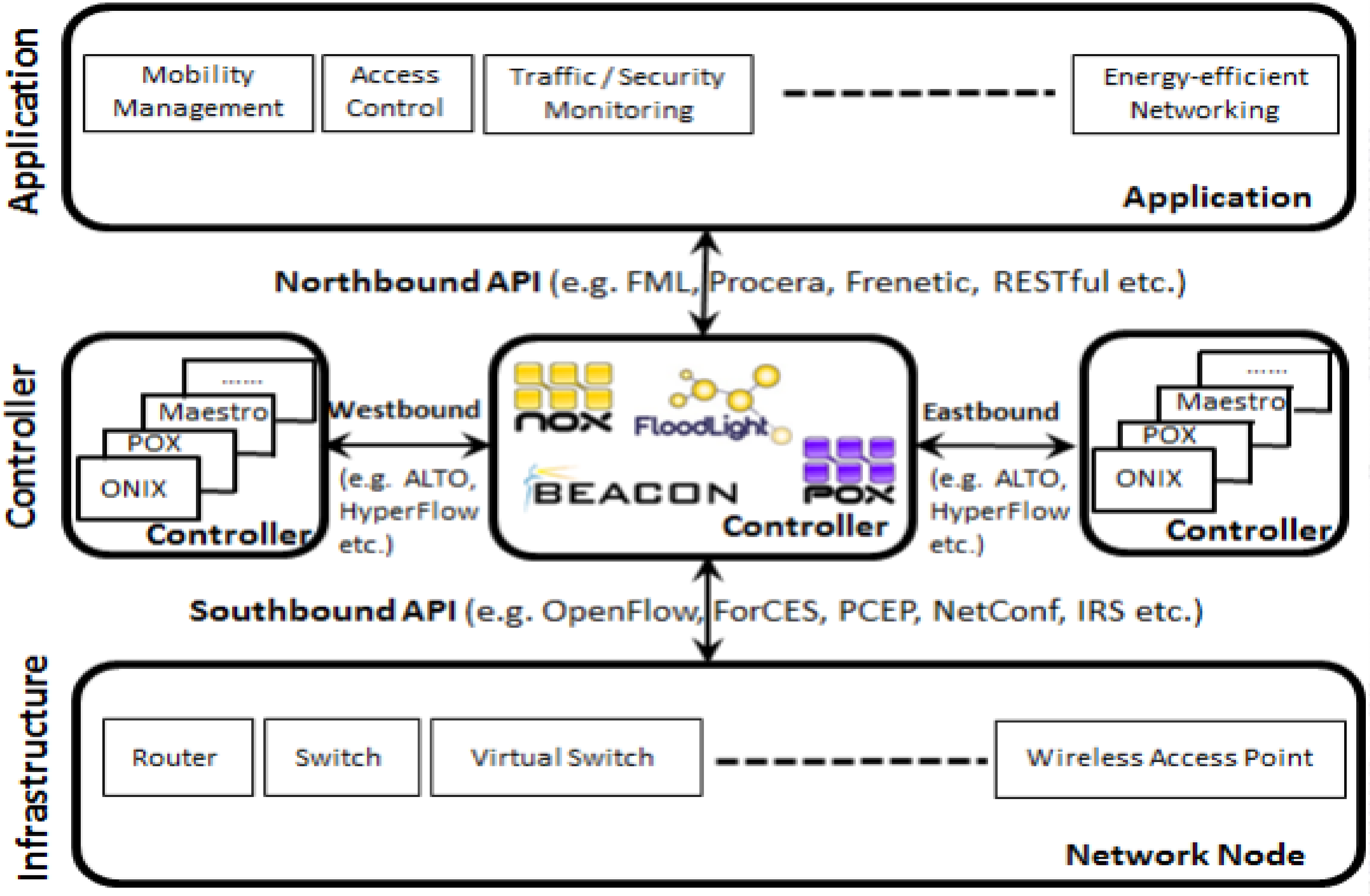
Range of new apps in research

- Environments
  - Mobility
  - Control of wireless infrastructure
  - Performance optimization
  - …
- Infrastructure / “northbound APIs”
  - Programming languages
  - APIs to assist policy-compliant updates
  - Verification
  - …

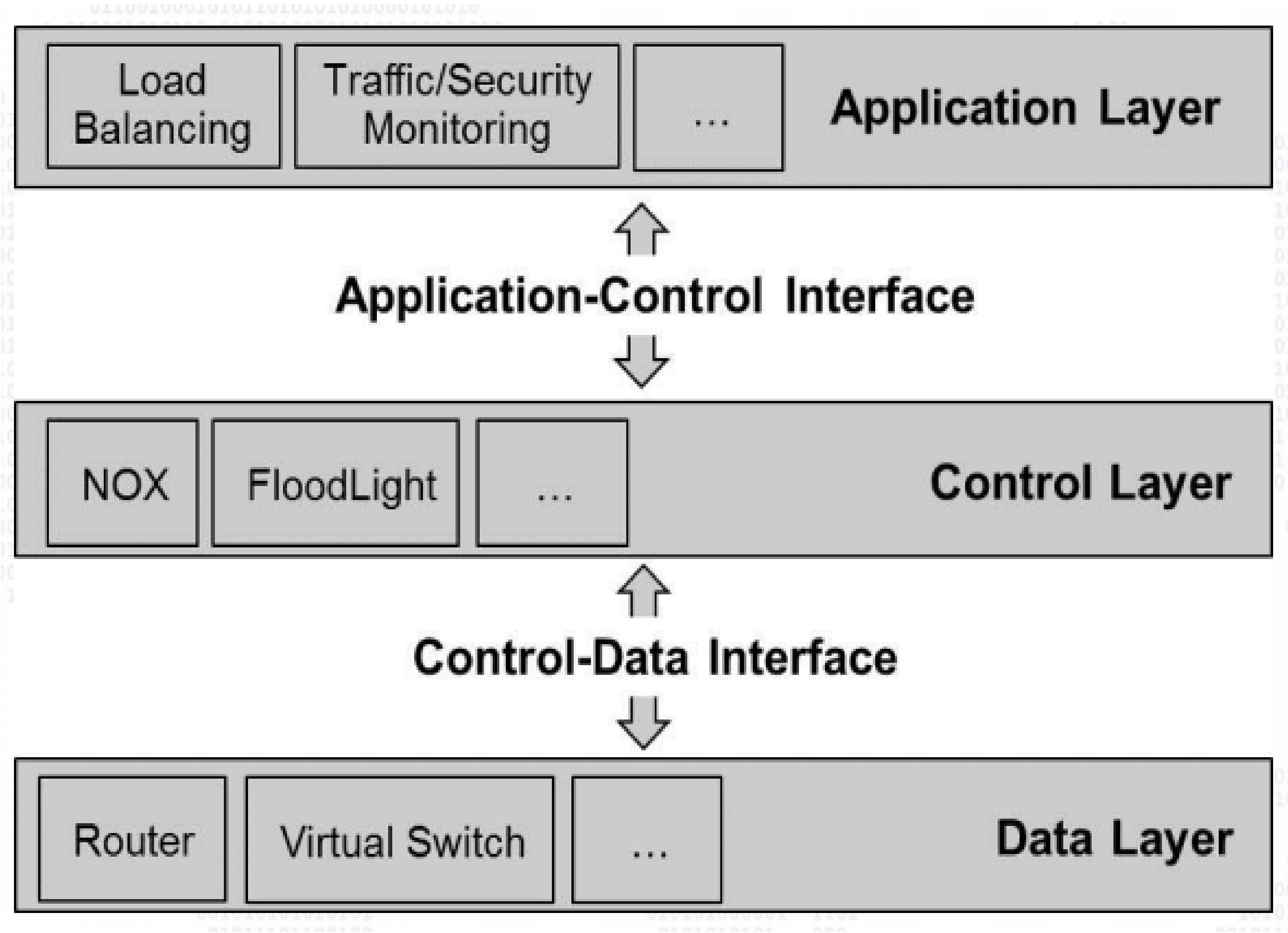
Questions

- When does the SDN controller send instructions to switches?
  - ...in the OpenFlow paper? reactive
  - ...other options? proactive
- How does SDN affect reliability?
  - More bugs in the network, or fewer?

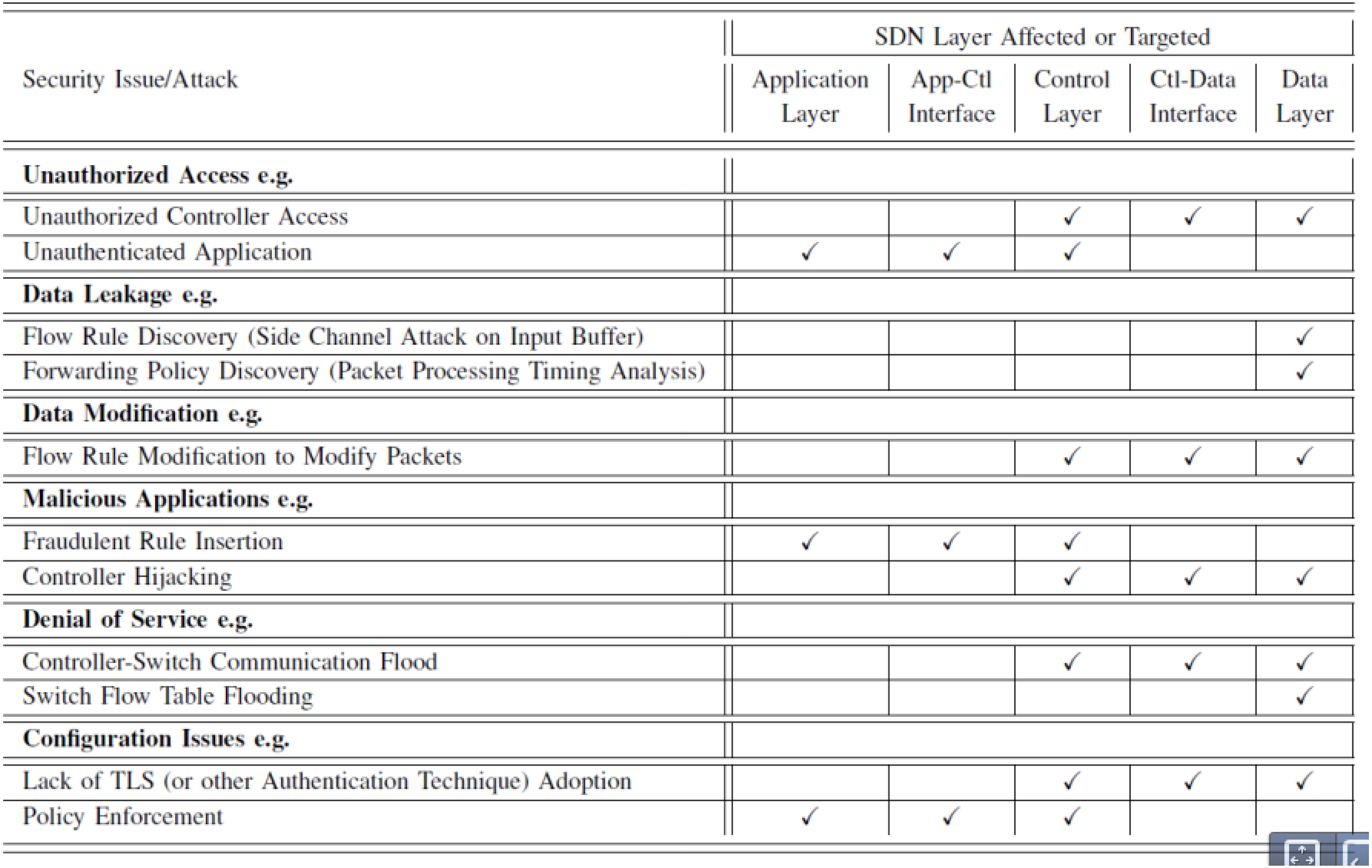
Security: SDN Architecture



Security: SDN Architecture

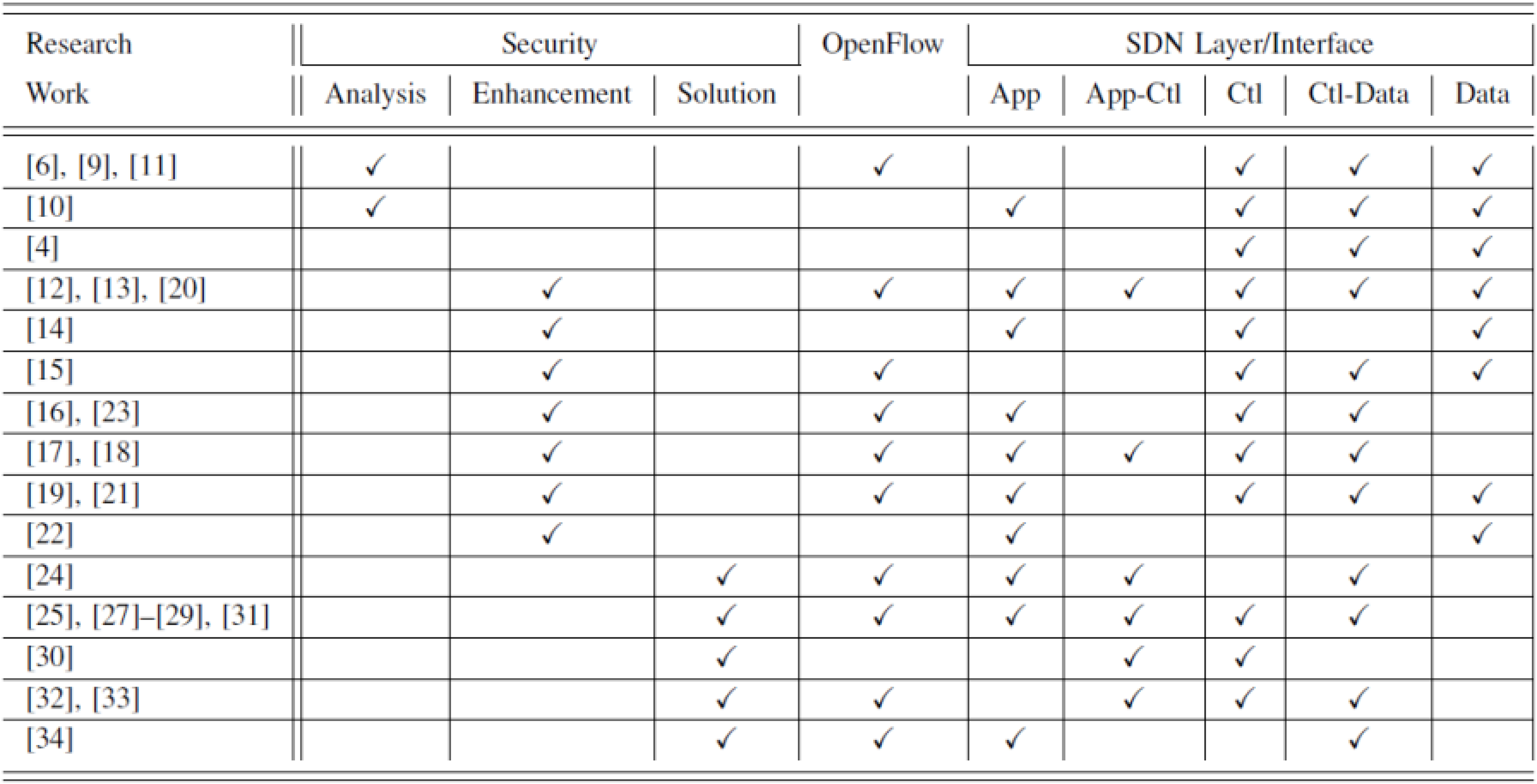


Categorization of SDN Security Research



Security Issue/Attack	SDN Layer Affected or Targeted				
	Application Layer	App-Ctl Interface	Control Layer	Ctl-Data Interface	Data Layer
Unauthorized Access e.g.					
Unauthorized Controller Access			✓	✓	✓
Unauthenticated Application	✓	✓	✓		
Data Leakage e.g.					
Flow Rule Discovery (Side Channel Attack on Input Buffer)					✓
Forwarding Policy Discovery (Packet Processing Timing Analysis)					✓
Data Modification e.g.					
Flow Rule Modification to Modify Packets			✓	✓	✓
Malicious Applications e.g.					
Fraudulent Rule Insertion	✓	✓	✓		
Controller Hijacking			✓	✓	✓
Denial of Service e.g.					
Controller-Switch Communication Flood			✓	✓	✓
Switch Flow Table Flooding					✓
Configuration Issues e.g.					
Lack of TLS (or other Authentication Technique) Adoption			✓	✓	✓
Policy Enforcement	✓	✓	✓		

Categorization of SDN Security Research



Research	Security			OpenFlow	SDN Layer/Interface				
Work	Analysis	Enhancement	Solution		App	App-Ctl	Ctl	Ctl-Data	Data
[6], [9], [11]	✓			✓			✓	✓	✓
[10]	✓				✓		✓	✓	✓
[4]							✓	✓	✓
[12], [13], [20]		✓		✓	✓	✓	✓	✓	✓
[14]		✓			✓		✓		✓
[15]		✓		✓			✓	✓	✓
[16], [23]		✓		✓	✓		✓	✓	
[17], [18]		✓		✓	✓	✓	✓	✓	
[19], [21]		✓		✓	✓		✓	✓	✓
[22]		✓			✓				✓
[24]			✓	✓	✓	✓		✓	
[25], [27]–[29], [31]			✓	✓	✓	✓	✓	✓	
[30]			✓			✓	✓		
[32], [33]			✓	✓		✓	✓	✓	
[34]			✓	✓	✓			✓	

Open Research Areas

- Moving Target Defense
  - Exploiting the dynamic and adaptive capabilities of SDN
- Trust (Application-Enabled SDN)
  - Application-Control Interface and Control-Data Interface
- Securing the Network Map

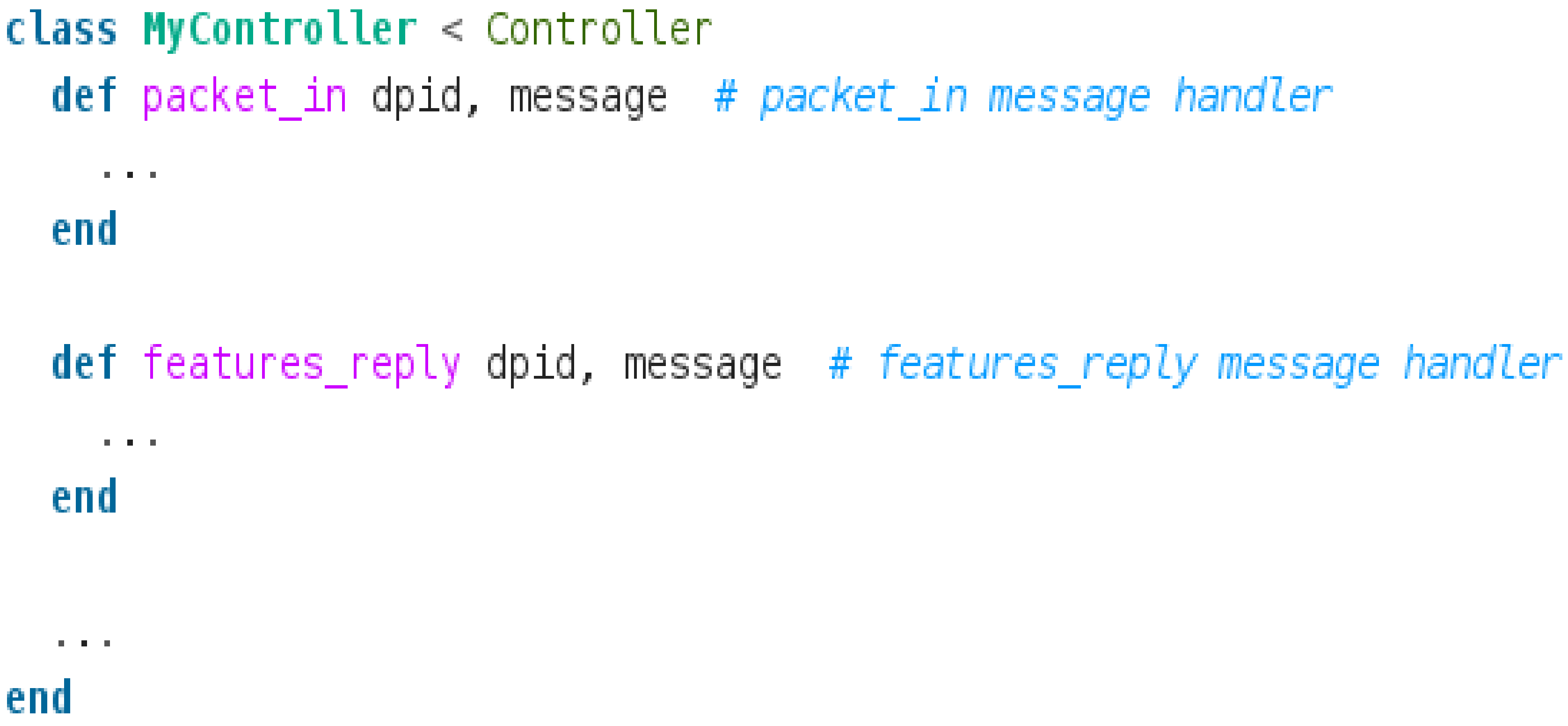
Controller: we are looking for

- clean code base (not a slick GUI or ground-breaking performance),
- versatile core (not ad-hoc solutions like STP to common problems),
- decent set of built-in components (e.g., device/switch query interface, access to topology, links and routing),
- quick deployment cycle (i.e., it should take a few minutes to compile your changes and fire up the controller),
- active development (i.e., presence of developers constantly contributing to the code base),
- some documentation (not just a API reference).

Controller

Feauters	NOX	POX	Beacon	Floodlight	OpenDaylight	Trema
Language supported	C, C++, Python	Python	Java	Java, Python	Java	C, Ruby
Actively developed	N	Y	Mainteined	Y	Y	Y
Active community	N	Y	Y	Y	Y	N
Is documented?	N	N	Y	Y	some	Y

Writing Your Own Controller



```
class MyController < Controller
  def packet_in dpid, message  # packet_in message handler
    ...
  end

  def features_reply dpid, message  # features_reply message handler
    ...
  end


  ...
end
```