Software Defined Networking

Richard T. B. Ma

School of Computing

National University of Singapore

Material from: Scott Shenker (UC Berkeley), Nick McKeown (Stanford), Jennifer Rexford (Princeton)

CS 3103: Compute Networks and Protocols

An Intern Position

- The IT department if A*STAR's Bioinformatics Institute (BII) is seeking an intern for training in network administration and programming. The focus of the internship is on software-defined networks and Open Network Install Environment-related work so it will be interesting for those students who are keen with new technologies. Interested students please submit the followings
  - - detailed CV
  - - copy of NRIC or passport (if foreigner)
  - - copy of educational certificates & transcripts
  - - period available for internship
- To Ms Betty Kee
  - Senior Admin Manager, Bioinformatics Institute (BII), 30 Biopolis Street, #07-01 Matrix, Singapore 138671
  - Main Tel: (65) 6478 8298
  - Email: bettyk@bii.a-star.edu.sg
  - http://www.bii.a-star.edu.sg/

Motivation

- What is Software Defined Networking?
  - Is it OpenFlow? or what is OpenFlow?
  - How about Network Virtualization?
  - How about separation of Control/Data planes?
  - What is the software part of SDN?

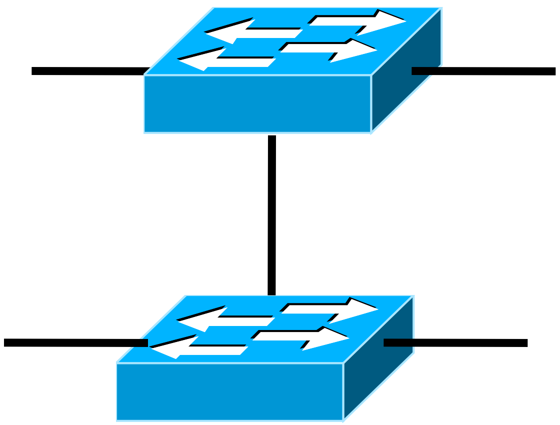
- Before knowing what it is, let us understand why we need it in the first place
  - What is wrong with the current Internet?

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

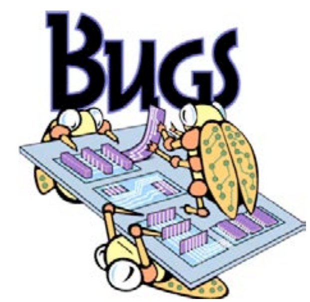
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features



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

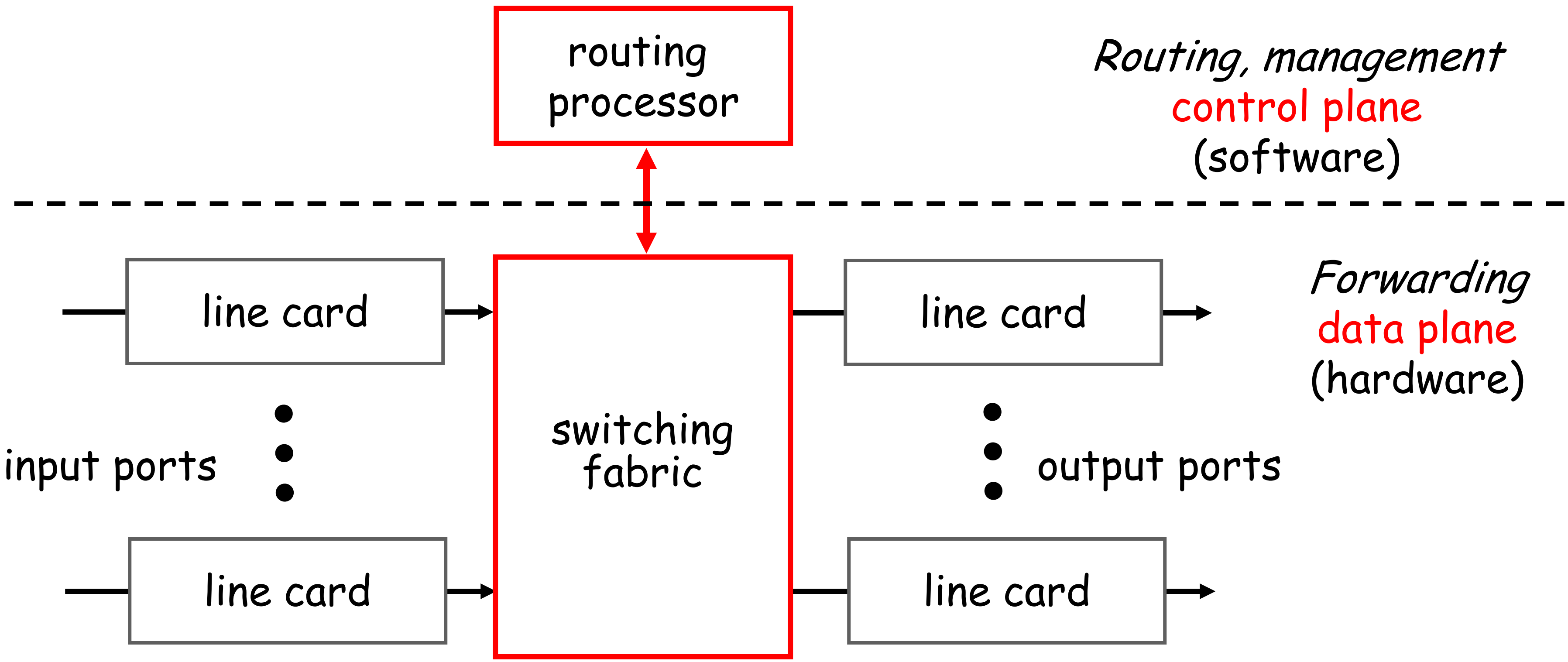
Networks Vs. Other Systems

- Networks are hard to manage
  - Computation and storage have been virtualized
    - Creating a more flexible and manageable infrastructure
  - Networks are still notoriously hard to manage
    - Still heavily rely on network administrators
- Networks are hard to evolve
  - Ongoing innovation in systems software
    - New programming languages, operating systems, etc.
  - Networks are stuck in the past
    - Routing algorithms change very slowly
    - Network management extremely primitive

Complicated Router at the Core

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



Two Key Definitions

- **Data Plane**: processing and delivery of packets
  - Based on state in routers and endpoints
  - E.g., IP, TCP, Ethernet, etc.
  - Fast timescales (per-packet)
- **Control Plane**: establishing the state in routers
  - Determines how and where packets are forwarded
  - Routing, traffic engineering, firewall state, …
  - Slow time-scales (per control event)

Networking as a Discipline

- Other fields in "systems": OS, DB, DS, etc.
  - Teach basic principles
  - Are easily managed
  - Continue to evolve

- Networking:
  - Teach big bag of protocols
  - Notoriously difficult to manage
  - Evolves very slowly

A failure from an academic point of view

Why Does Networking Lag Behind?

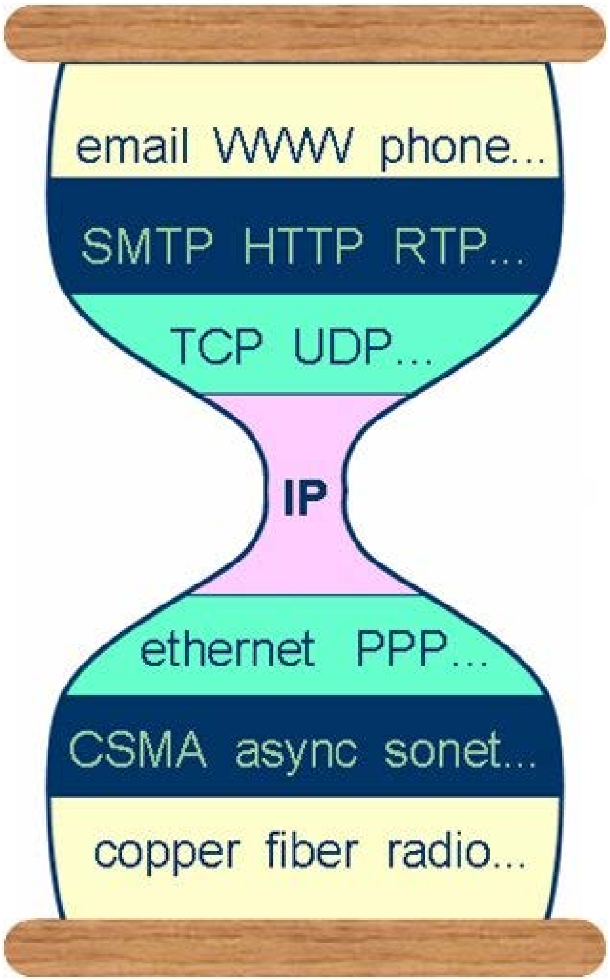
- Networks used to be simple: Ethernet, IP, TCP....
- New **control** requirements led to great complexity
  - Isolation ➔ VLANs, ACLs
  - Traffic engineering ➔ MPLS
  - Packet processing ➔ Firewalls, NATs
  - Payload analysis ➔ Deep packet inspection
- Mechanisms designed and deployed independently
  - Complicated "control plane" design, primitive functionality
  - Stark contrast to the elegantly modular "data plane"
- Ability to master complexity ➔ a curse
  - Extract simplicity is needed to build a discipline

A Good Example: Programming

- Machine languages: no abstractions
  - Mastering complexity was crucial
- Higher-level languages: OS and other abstractions
  - File system, virtual memory, abstract data types, ...
- Modern languages: even more abstractions
  - Object orientation, garbage collection,...

Abstractions key to extracting simplicity

Key to the Internet's Success



- Hourglass IP model
- Layered service abstractions (why is this important?)
  - Decompose delivery into fundamental components
  - Independent, compatible innovation at each layer
- Only for network edges

What have we learned so far?

- Layers are great abstractions
  - Layers only deal with the data plane
  - No powerful control plane abstractions!

"Modularity based on abstraction is the way things get done" – Barbara Liskov

Abstractions → Interfaces → Modularity

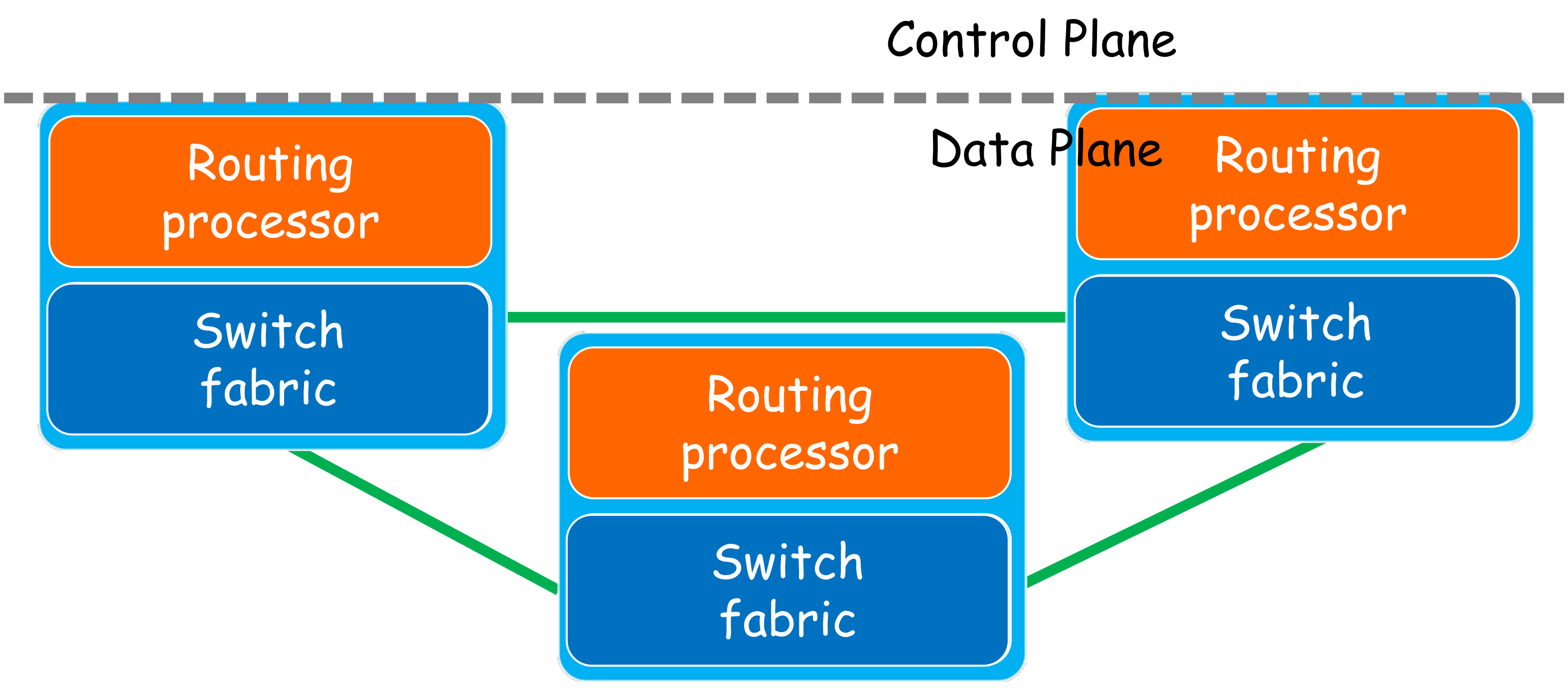
- How do we find control plane abstractions?
  - first define problem
  - and then decompose it

The network control problem

- Compute the configuration of each physical devices, e.g., forwarding tables
- Operate without communication guarantees
- Operate within given network-level protocol, e.g., RIP, OSPF.

Only people who love complexity would find this a reasonable request!

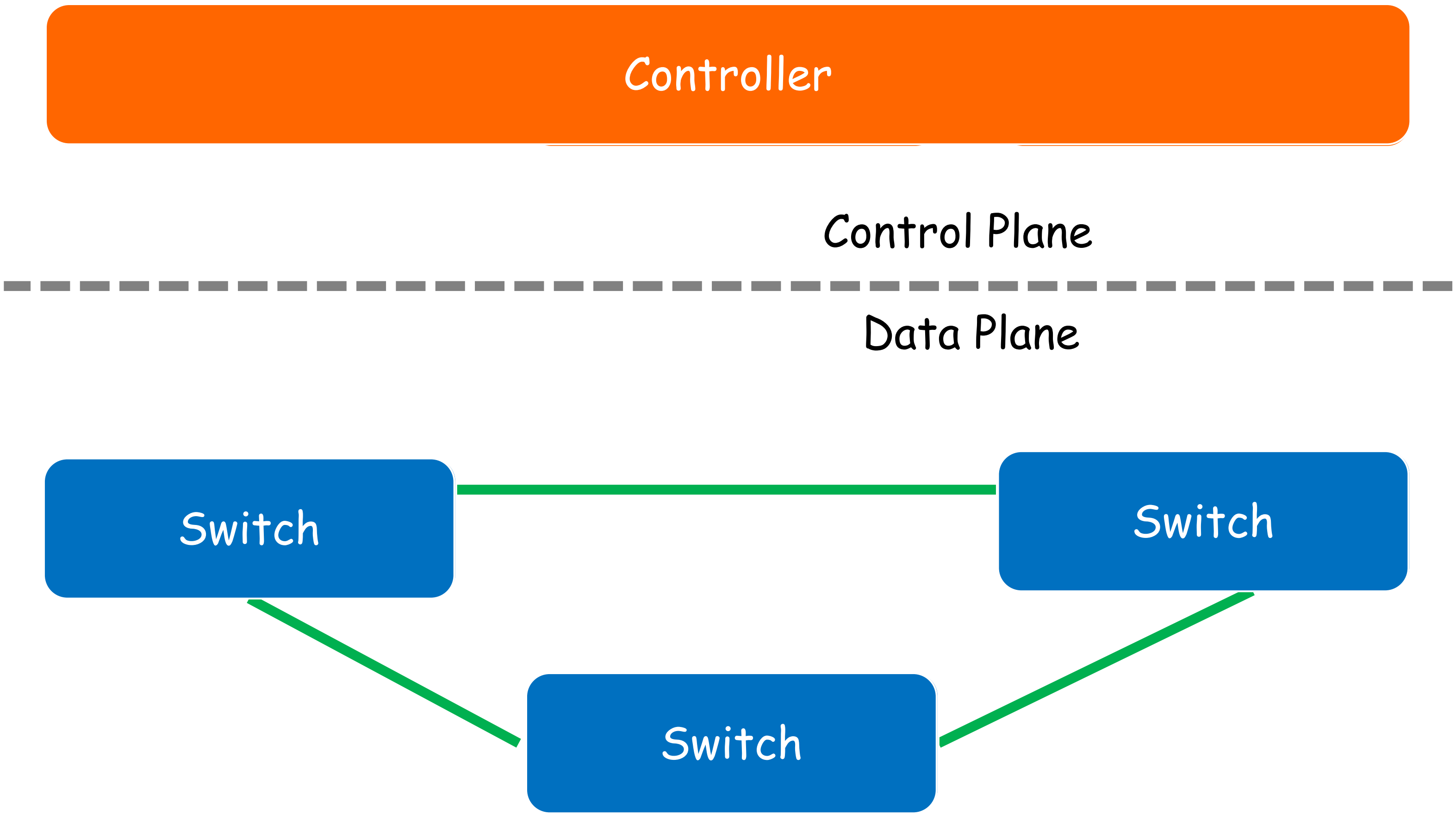
Separation of Control/Data Plane



Benefits of Separation

- Independent evolution and development
  - The software control of the network can evolve independently of the hardware.
- Control from high-level software program
  - Control behavior using higher-order programs
  - Debug/check behavior more easily

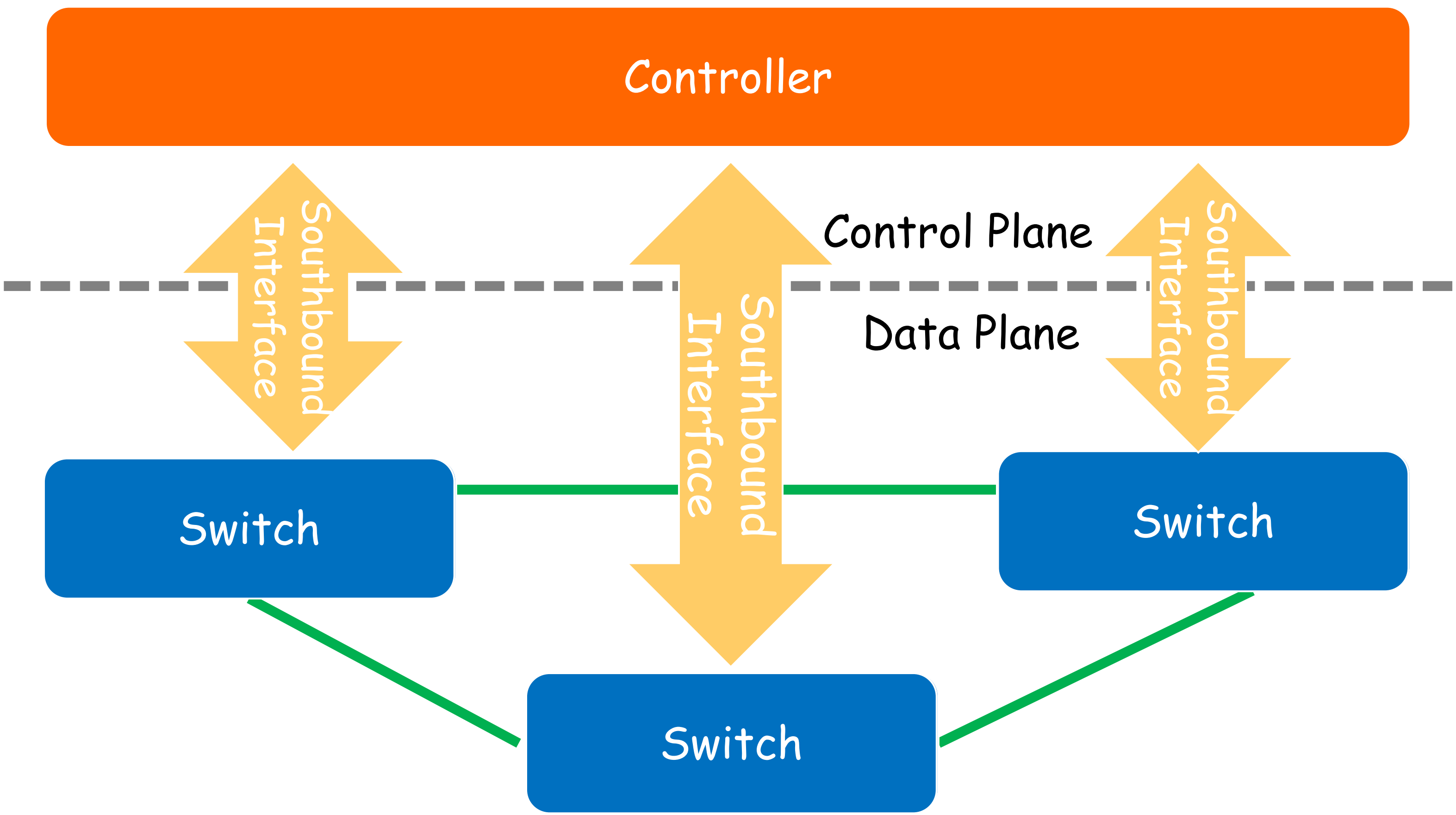
Logically Centralized Control



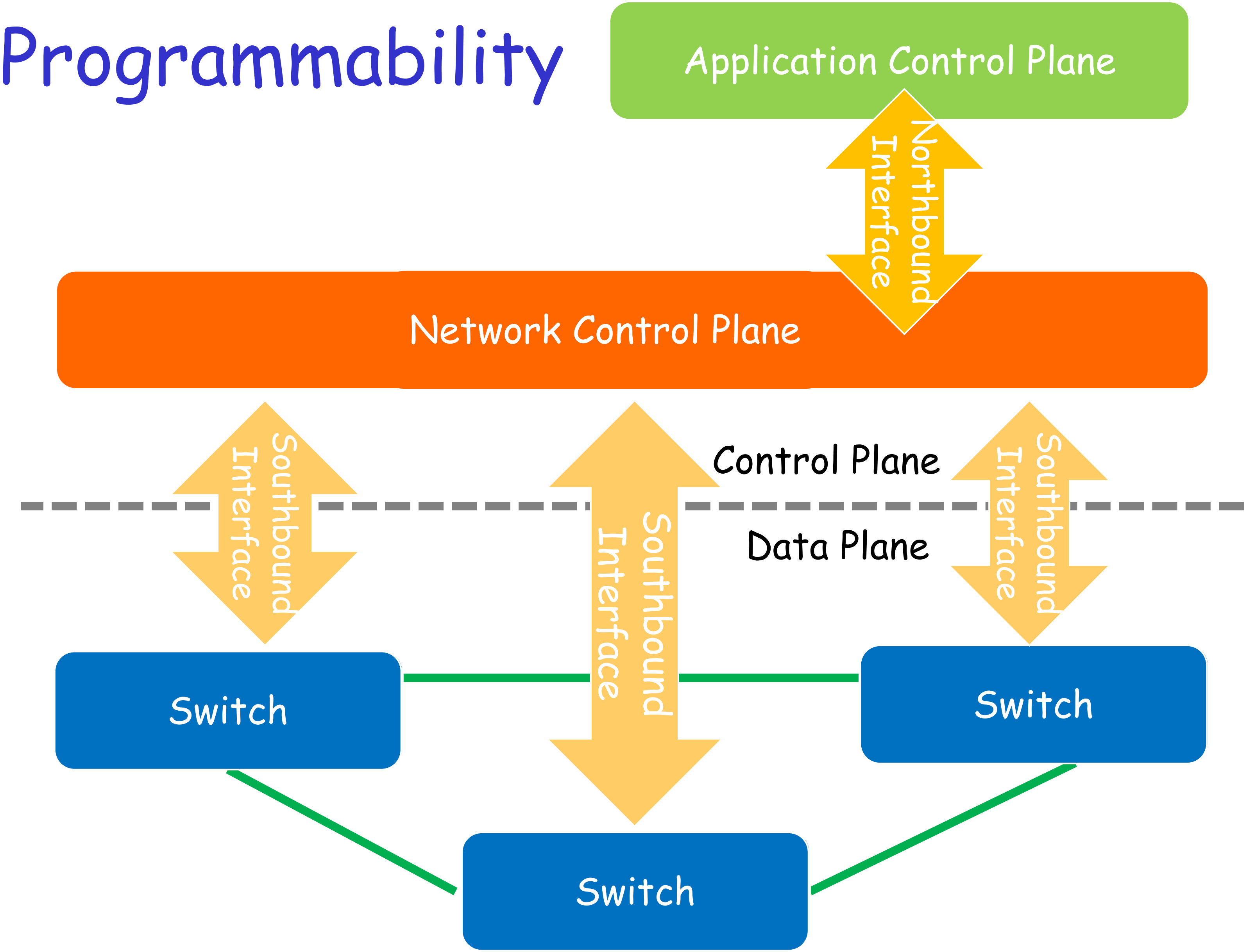
Benefits of Centralization

- Centralized decisions are easier to make
  - E.g., OSPF (RFC 2328) 244 pages
  - Distributed system part (builds consistent network 100 pages)
  - Routing algorithm (Dijkstra's algorithm 4 pages)
- Logically vs. physically centralized
  - Issues with of a physically centralized controller?
  - How to implement a logically centralized one?

Open Interfaces



Programmability



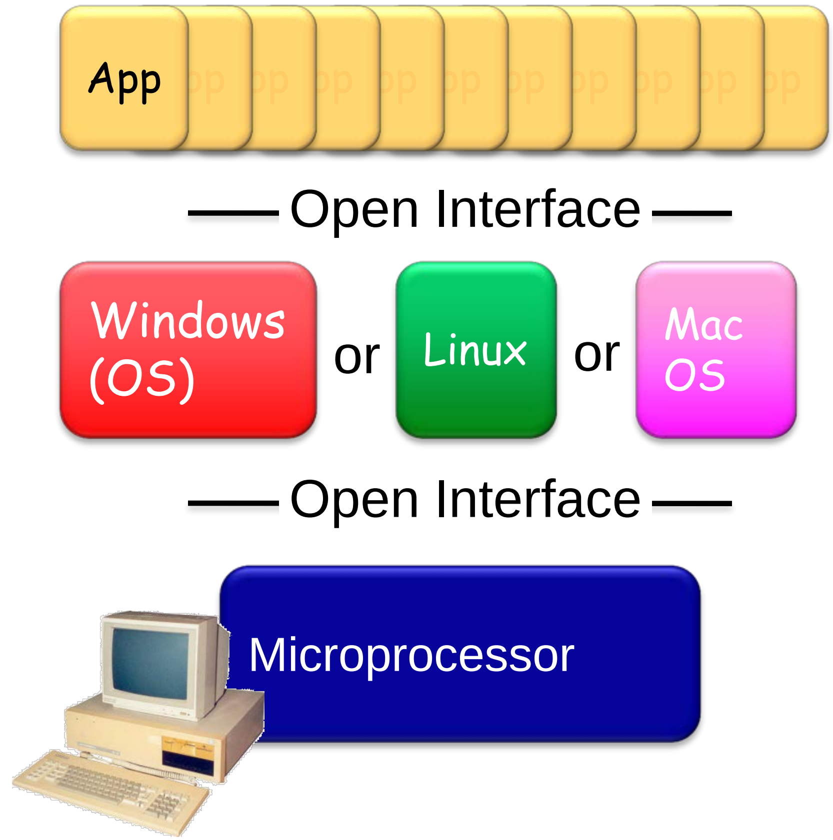
Benefits of Open Interfaces and Programmability

- Enable competitive technologies
  - Independent developments
  - Rapid innovation and fast evolution
  - Cheap and better networks
- Make network management much easier
  - Management goals are expressed as policies
  - New control/services for network providers
  - Detailed configuration are done by controller

A Quick Summary

- Principles of Software Defined Networking
  - Separation of Control Plane and Data Plane
  - Logically Centralized Control
  - Open Interfaces
  - Programmability
- All these principles use abstractions to modularize the network control problem.
- A nice analogy from Prof. Nick McKeown

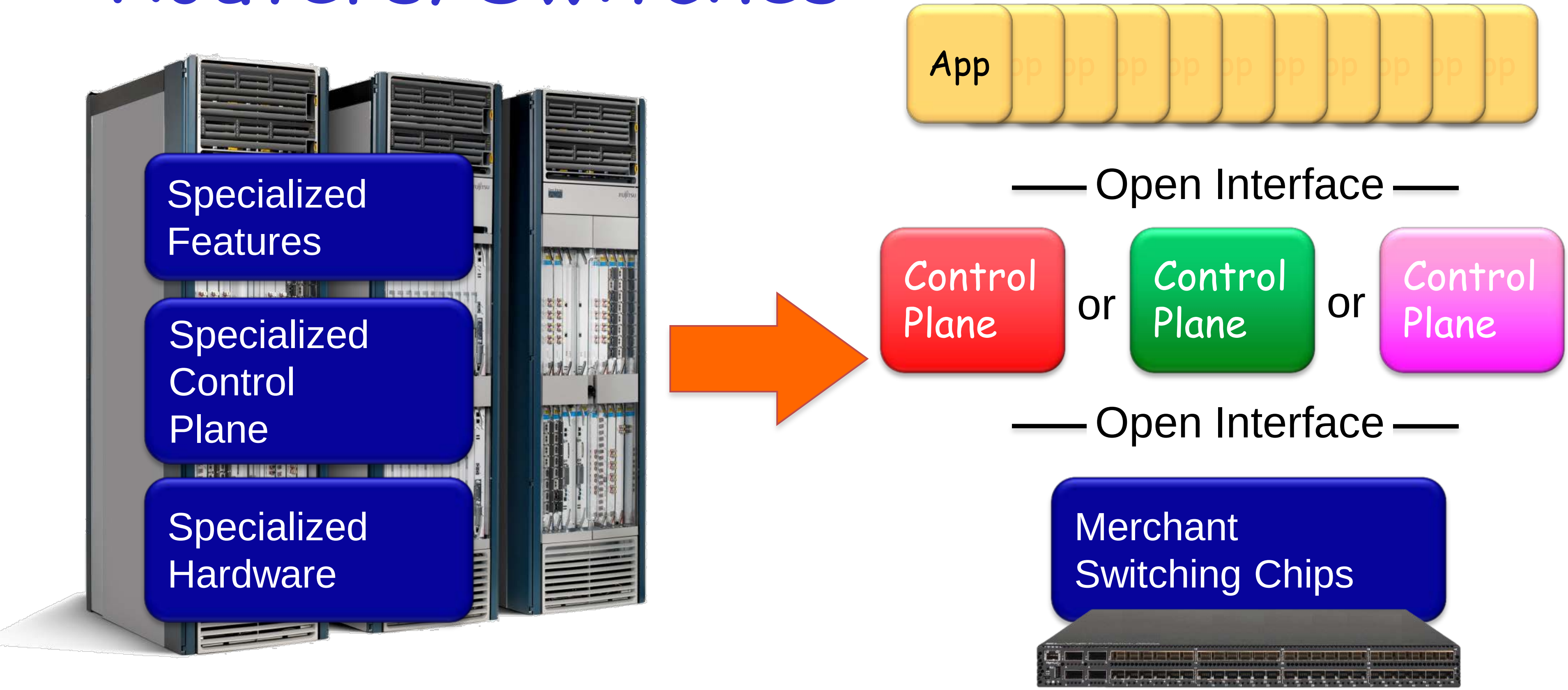
Mainframes



Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

Routers/Switches

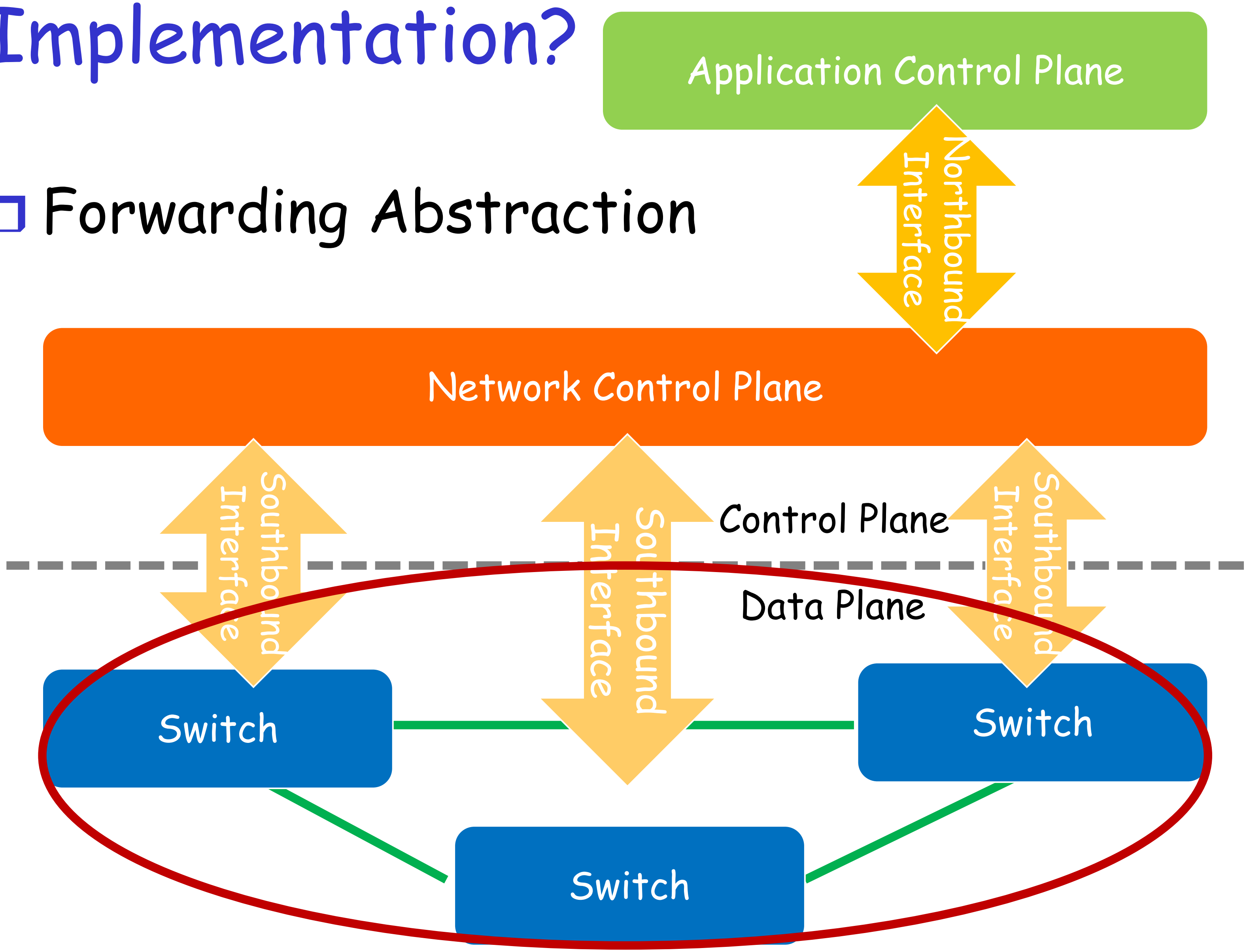


Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

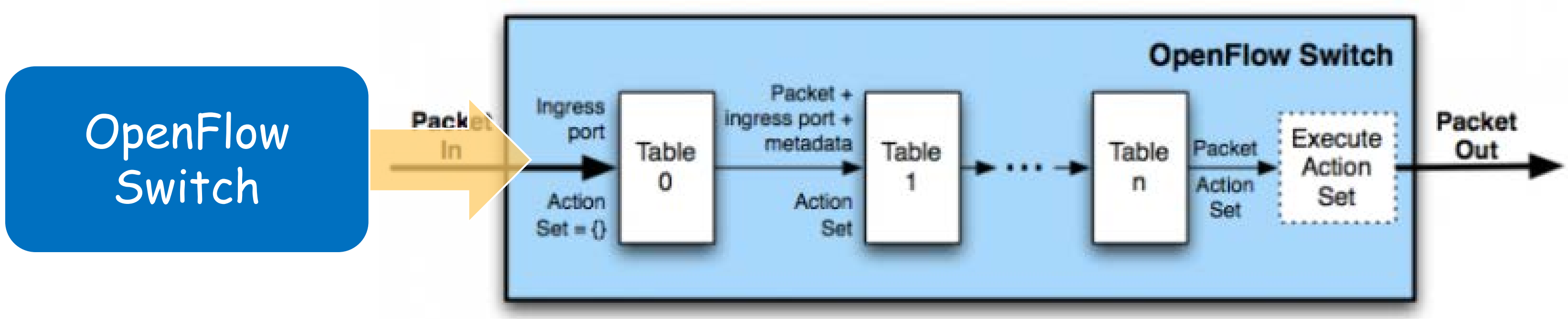
Implementation?

- Forwarding Abstraction

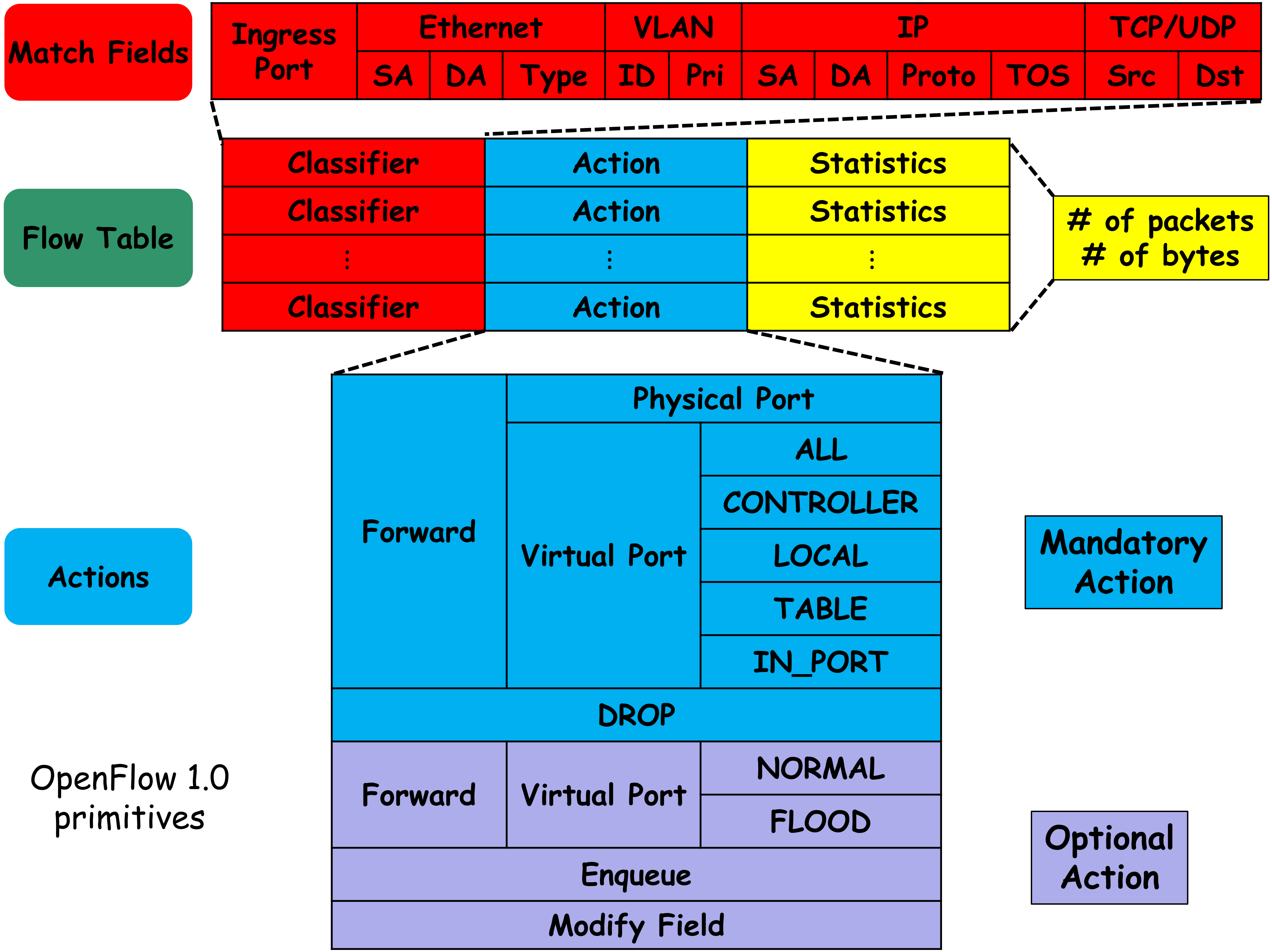


OpenFlow Protocol and Switch

- OpenFlow switches have flow tables
  - Each entry is a (Match, Actions) primitive
    - Match: packets are compared with flow entries
    - Actions: depending on whether a match is found
  - If no match, traffic is sent to controller



- communicate with controller via secure channel


Match Fields
Ingress Port
Ethernet
SA
DA
Type
VLAN
ID
Pri
IP
SA
DA
Proto
TOS
TCP/UDP
Src
Dst
Flow Table
Classifier
Action
Statistics
# of packets
# of bytes
Actions
Forward
Physical Port
Virtual Port
ALL
CONTROLLER
LOCAL
TABLE
IN_PORT
DROP
Forward
Virtual Port
NORMAL
FLOOD
Enqueue
Modify Field
Mandatory Action
Optional Action
OpenFlow 1.0 primitives

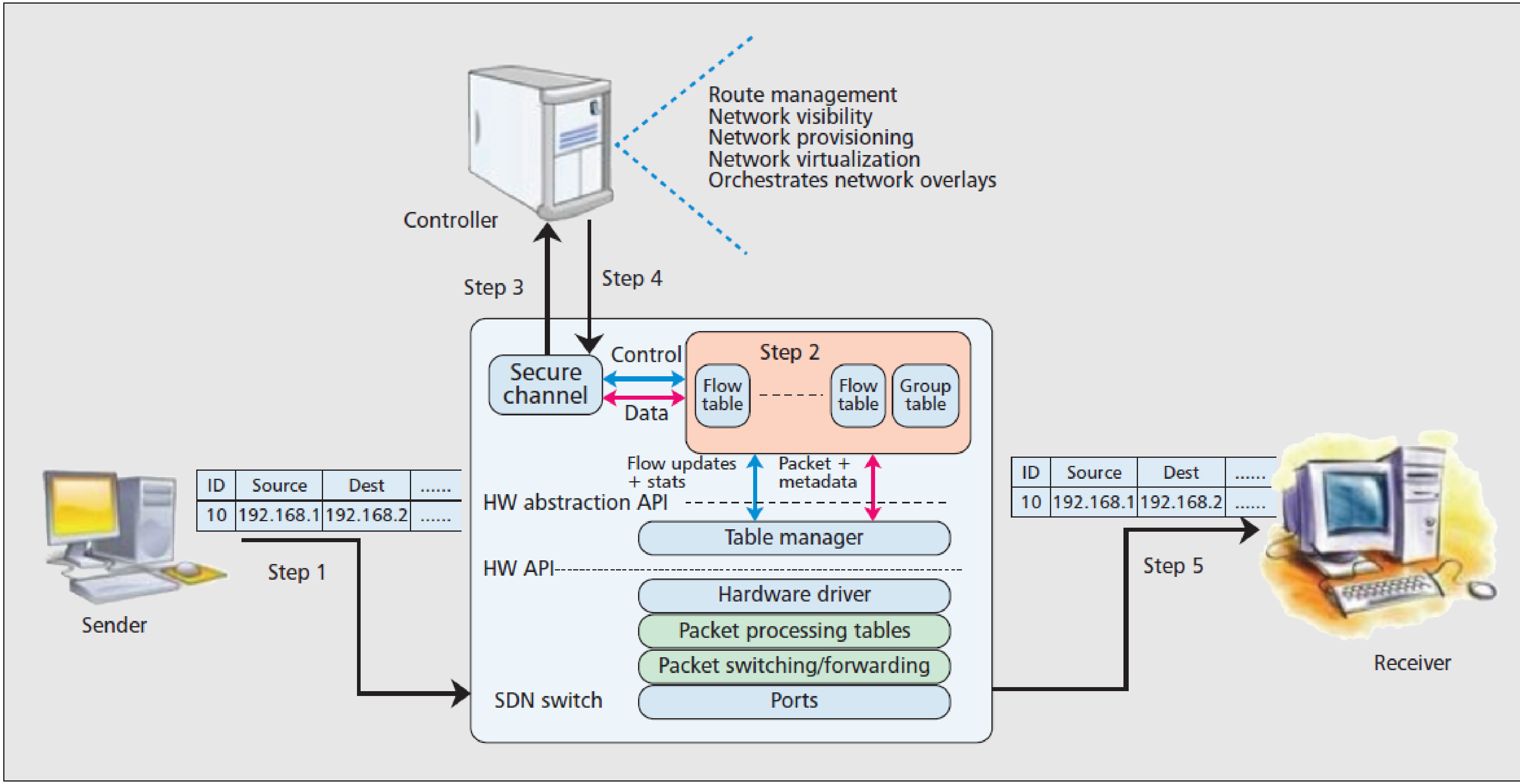
Flow Actions

- Forward
  - ALL: all but not including the incoming interface
  - CONTROLLER: send the packet to the controller
  - IN_PORT: send packet back to its incoming port
- Drop
  - a flow-entry with no specified action indicates that all matching packets should be dropped

Flow Actions

- Modify: option to modify packet header values in the packet
  - set VLAN ID, priority, etc.
  - set destination IP address
- Enqueue: send the packet through a queue attached to a port
- More details from OpenFlow switch specification:
  https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

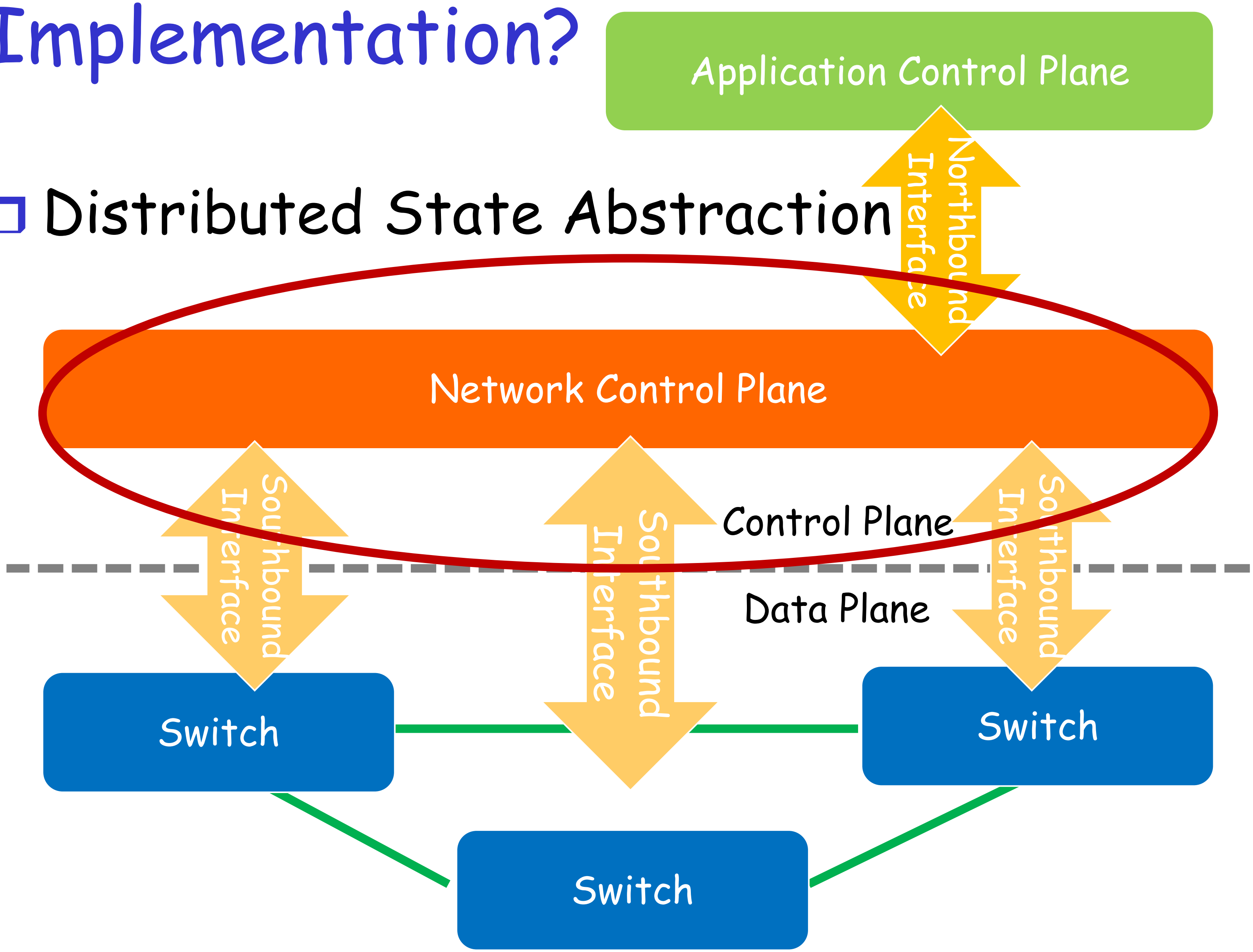
Operation of SDN (controller-switch)



S. Sezer et al. “Are we ready for SDN? Implementation Challenges for Software-Defined Networks”, IEEE Communications Magazine, July 2013.

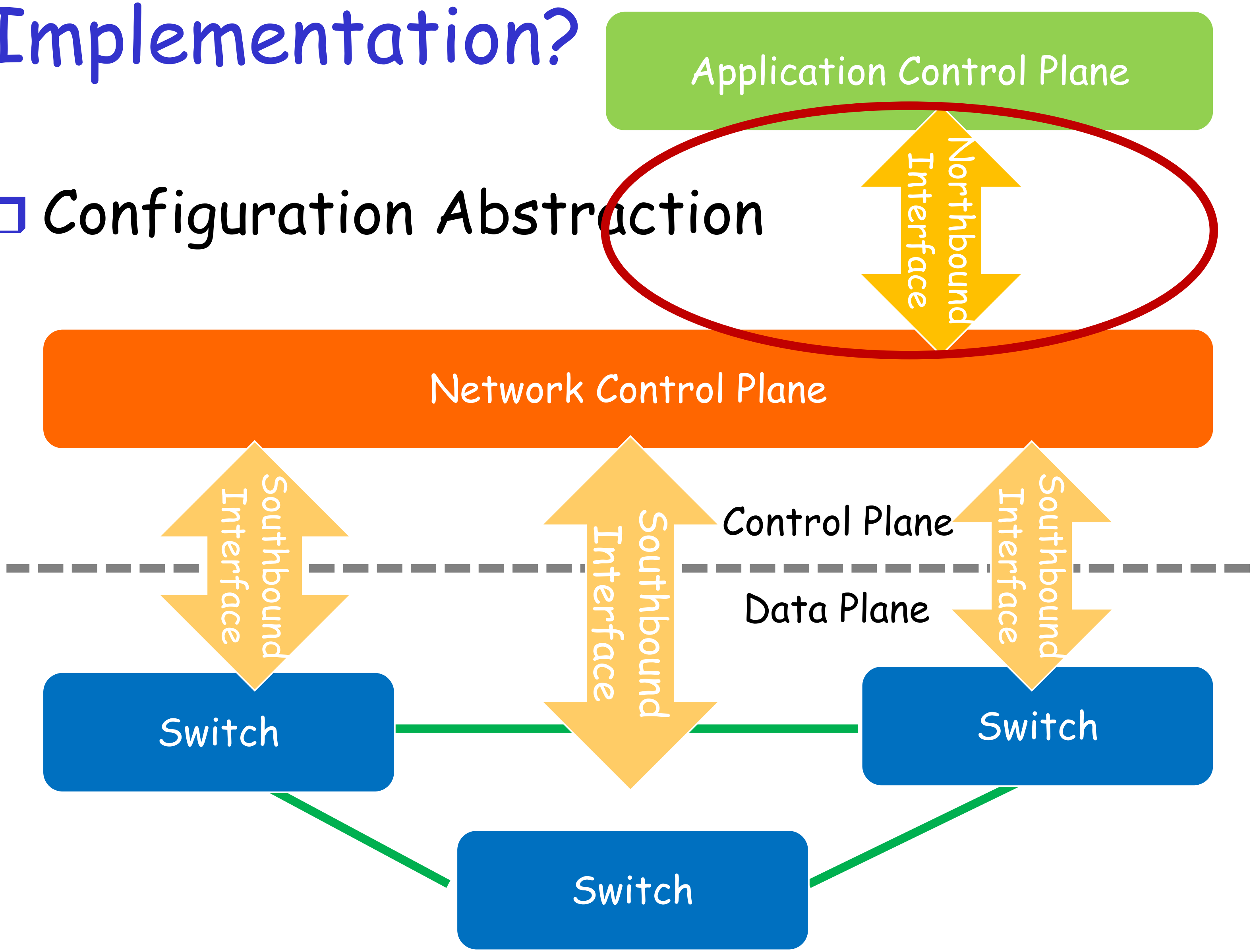
Implementation?

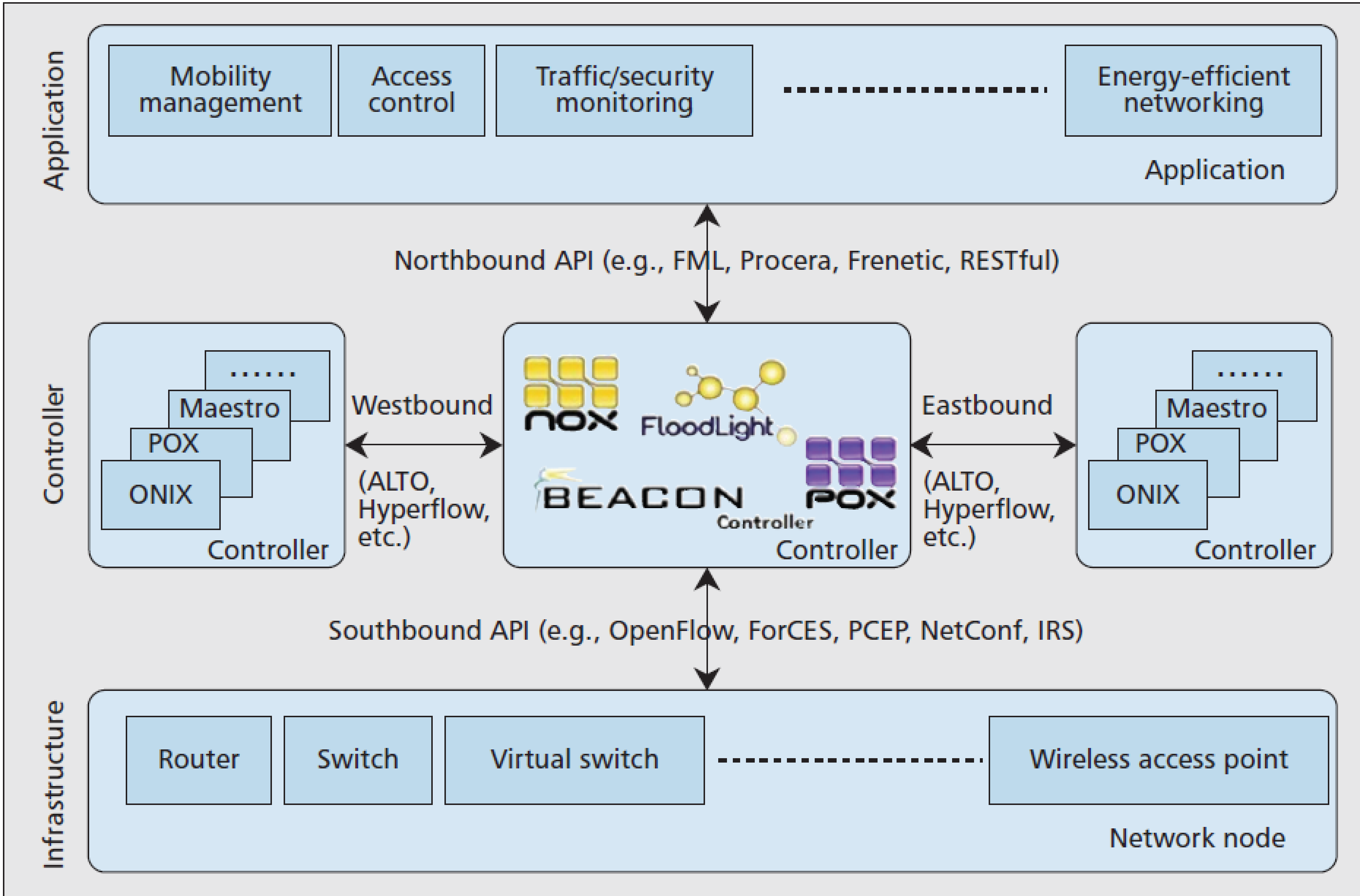
- Distributed State Abstraction



Implementation?

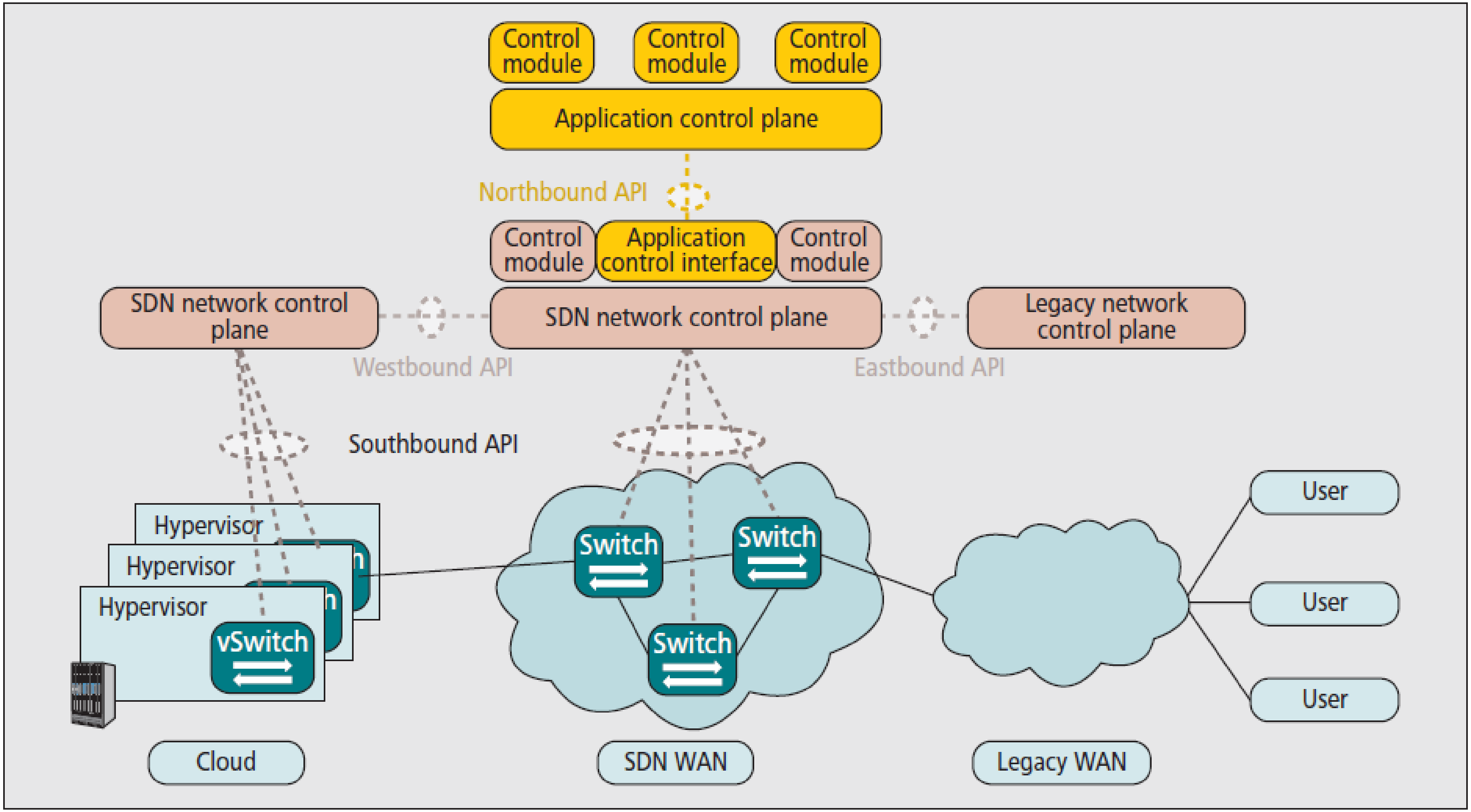
- Configuration Abstraction





S. Sezer et al. "Are we ready for SDN? Implementation Challenges for Software-Defined Networks", IEEE Communications Magazine, July 2013.

Interfaces of a SDN



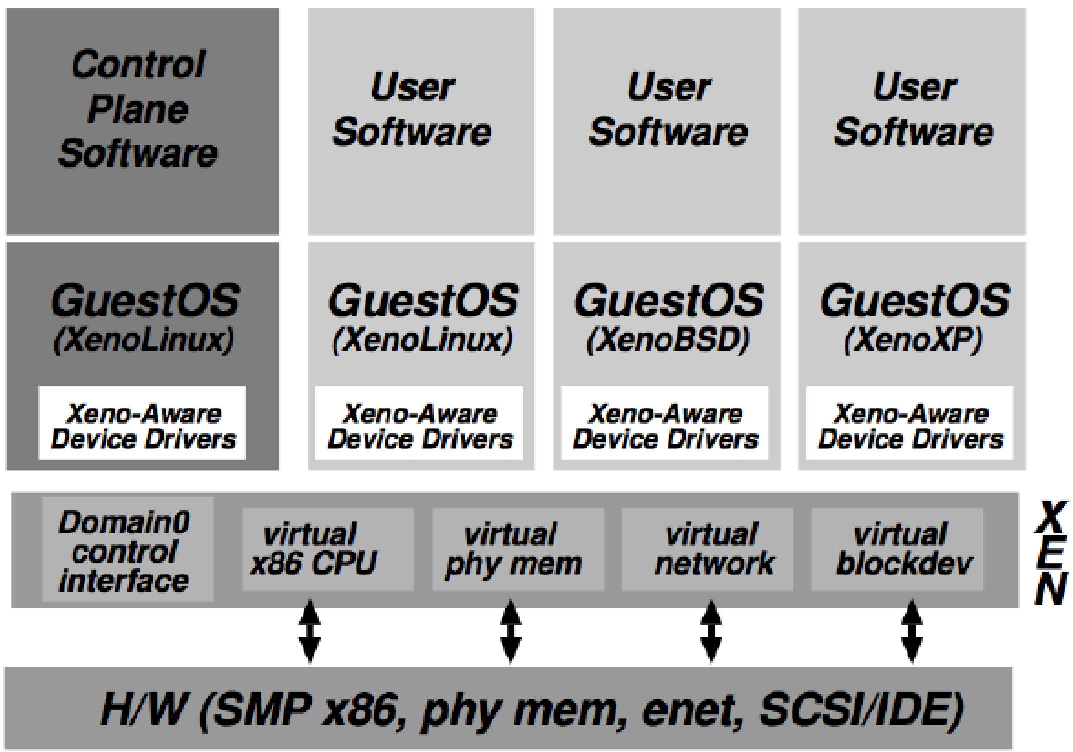
M. Jarschel et al. "Interfaces, attributes, and use cases: a compass for SDN", IEEE Communications Magazine, June 2014.

Network Virtualization (NV)

- What is virtualization?
  - Functionality of a single hardware is isolated for and used by multiple virtual instances
- Why do we want virtualization?
  - support multiple variations, easier for evolution
- What have been virtualized?
  - computing platform (virtual machines)
  - storage (virtual memory)
- How about networks?
  - not new, but very limited, e.g., VLAN and VPN

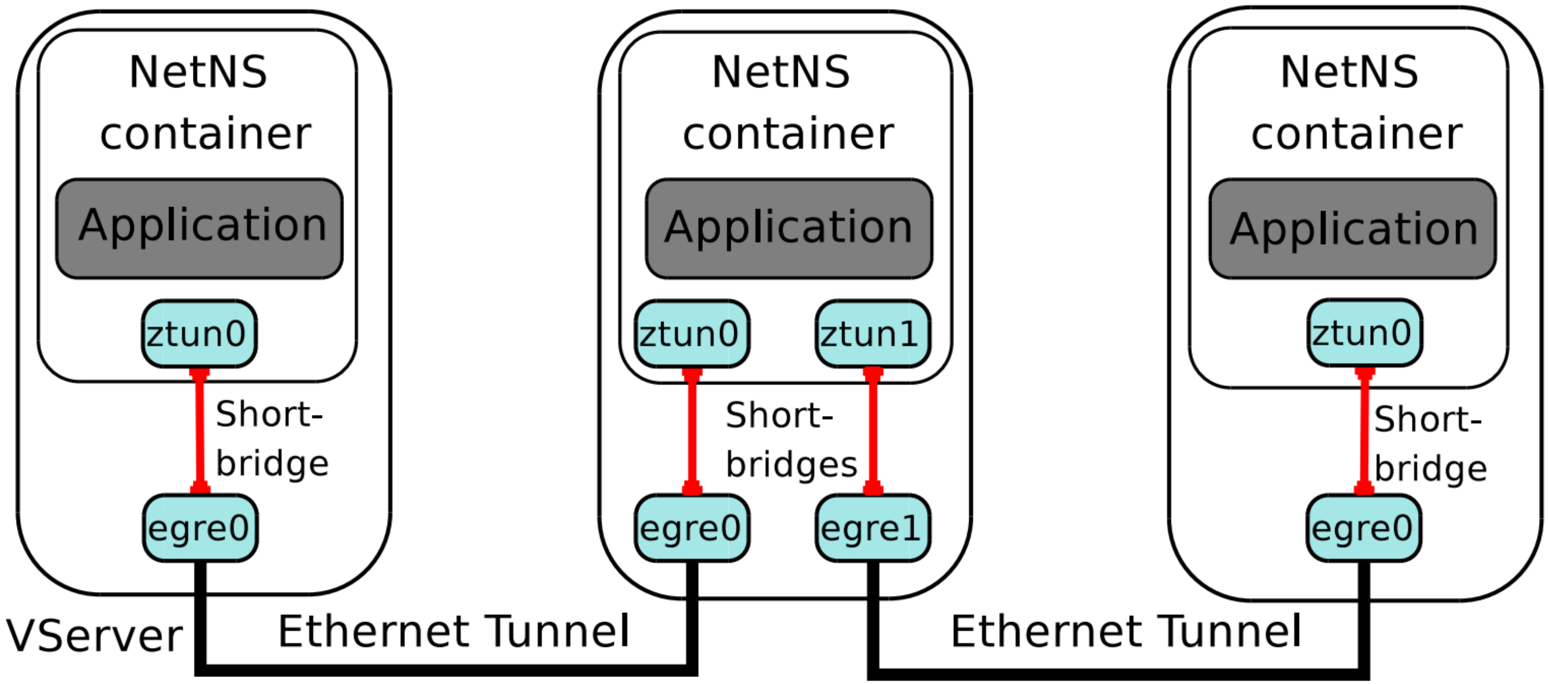
Example VM Environment: Xen

- Xen hosts multiple guest Oses
- Domain0 runs control software



Example Virtual Links: EGRE

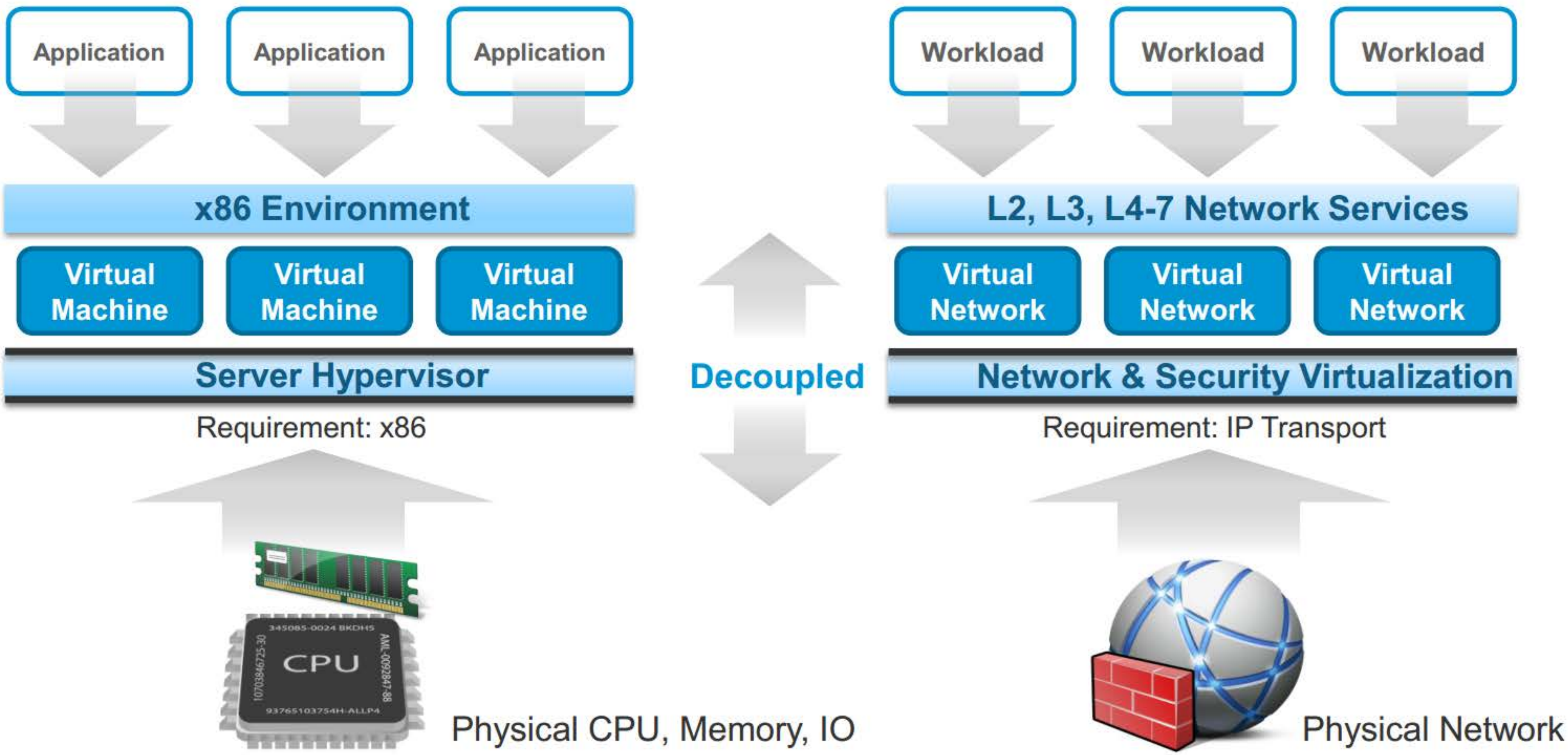
- Ethernet Generic Routing Encapsulation (EGRE): "Tunneling", Ethernet frames from virtual hosts are encapsulated in IP



Virtualizing Networks

- network elements that can be virtualized
  - nodes: virtual machines
  - links: tunnels
  - storage
  - what else?
- abstraction of the whole physical network
  - support for multiple logical networks running on a common shared physical substrate
  - a container of network services
  - virtualization at different layers

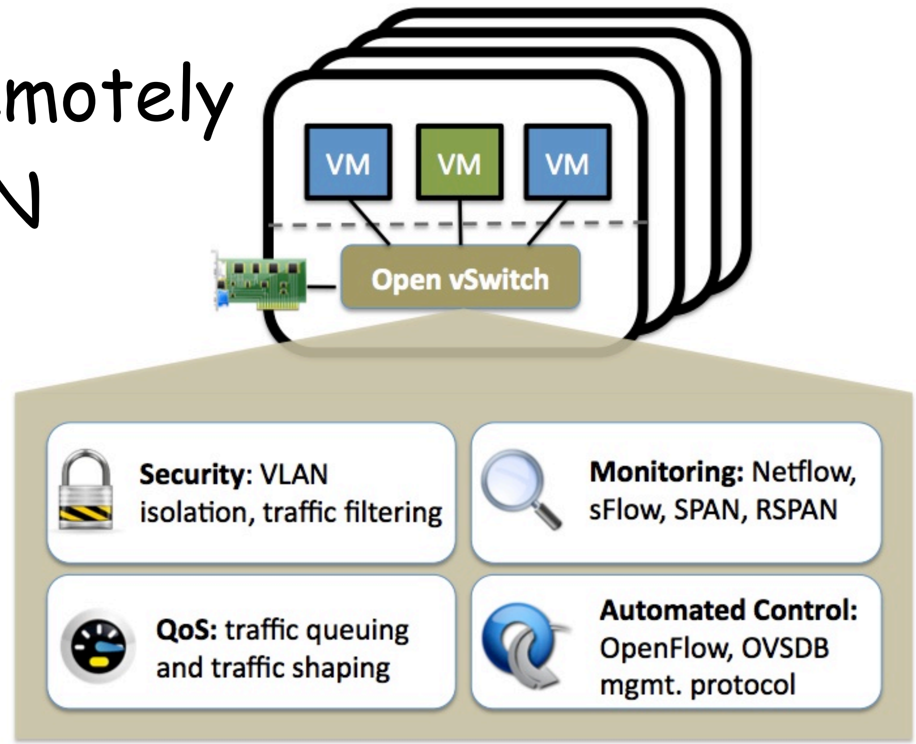
Virtualizing Networks



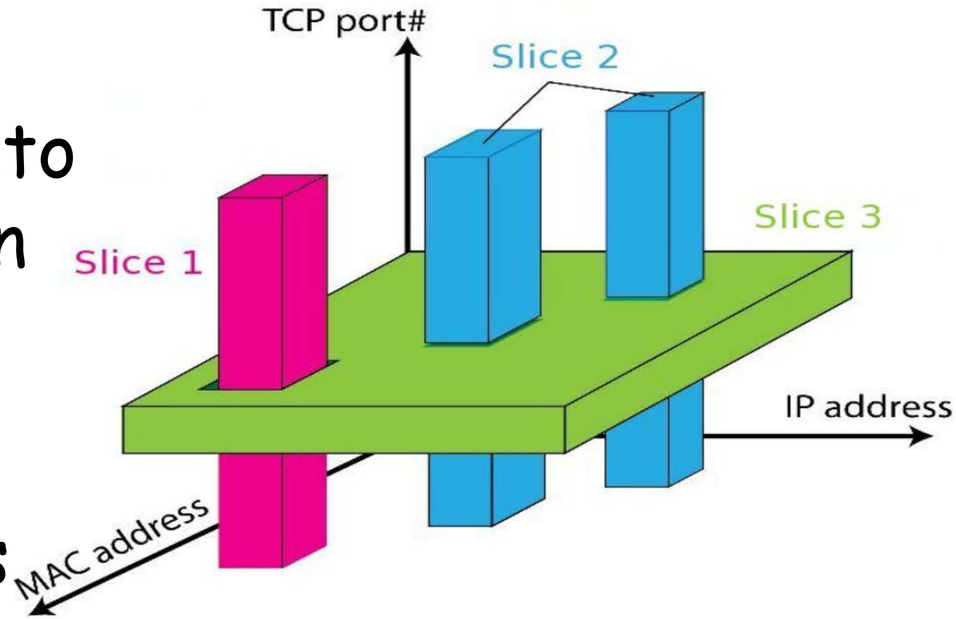
Example Virtual Switch: OVS

- Open vSwitch is a virtual switch designed to enable massive network automation

- Can be configured remotely with OpenFlow, JSON



Example Flow Space Slicing



- divide the network into logical slices based on flow characteristics
- enforce traffic isolation among slices
- application: divide the production network into logical slices for experiments
  - each slice controls its own packet forwarding
- e.g., OpenFlow-based controller: FlowVisor

SDN != NV

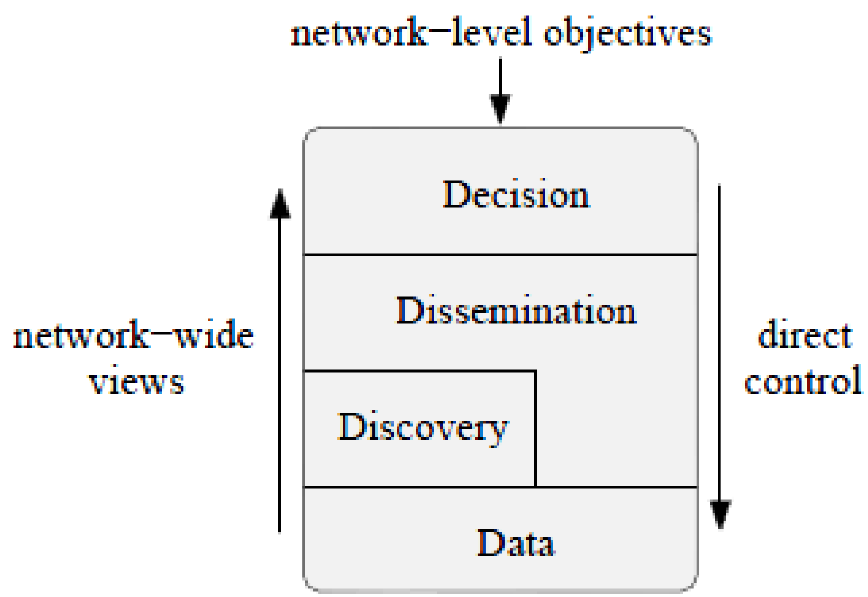
- SDN alone does not abstract away details of physical network
- SDN is not required for NV, but SDN makes NV much easier
- NV is also a way to enable abstractions
- NV and SDN are orthogonal …
  - SDN abstracts layers vertically
  - NV abstracts the same layer horizontally

Applications of SDN

- Killer app
  - Network virtualization
- Other apps
  - Datacenter and cloud computing
  - WAN (Google B4)
  - Software IXP
  - Home API
  - Wireless

Related Work

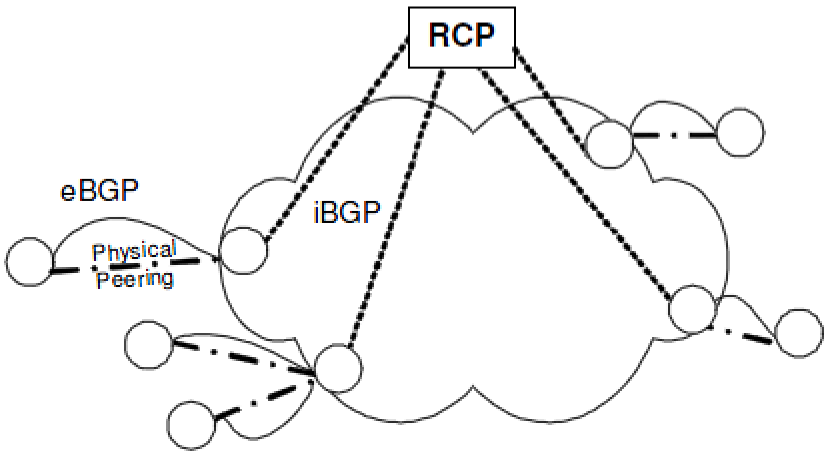
- Principles of the 4D network architecture
  - Network-level objectives
  - Network-wide views
  - Direct control
- Architecture
  - Decision plane
  - Dissemination plane
  - Discovery plane
  - Data plane

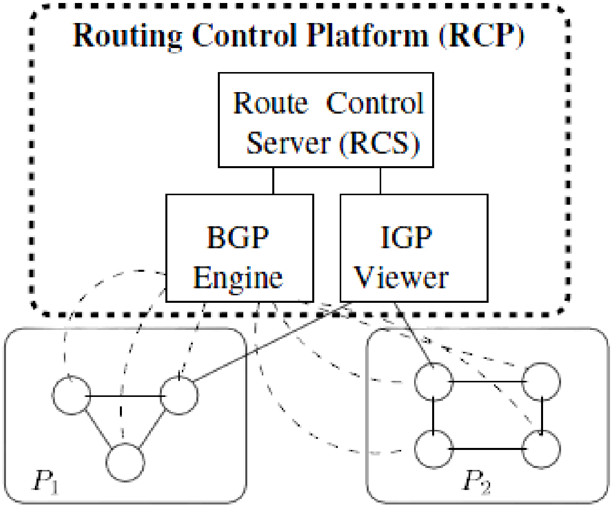


A. Greenberg et al. "A Clean Slate 4D Approach to Network Control and Management", ACM SIGCOMM Computer Communication Review, 35(5), 2005.

Related Work

- Routing Control Platform (RCP)
  - Logically centralized platform
  - Separate from the IP forwarding plane
  - Perform route selection using iBGP

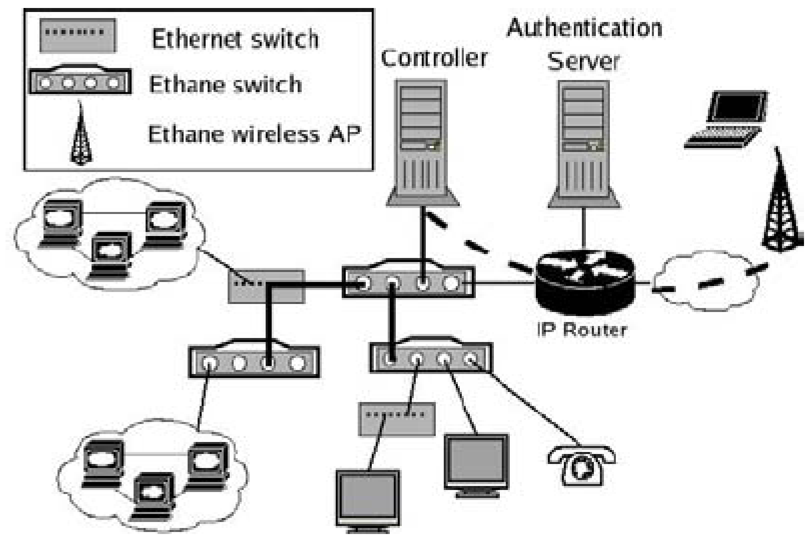


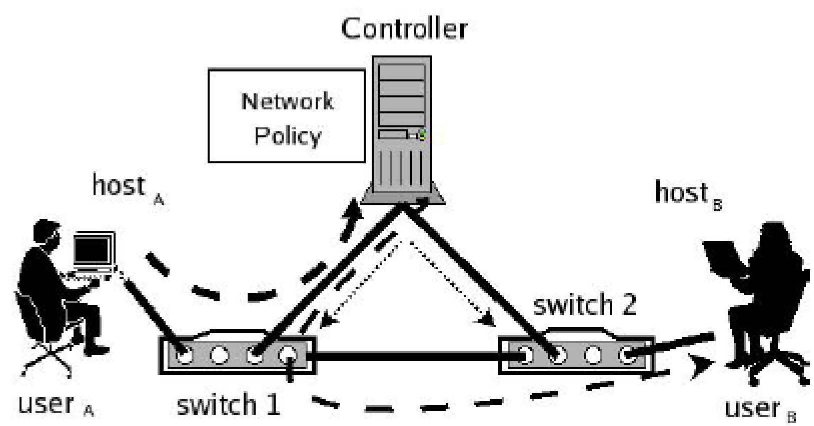


M. Caesar et al. “Design and Implementation of a Routing Control Platform”, Proceedings of USENIX NSDI, 2005.

Related Work

- Ethane architecture
  - Flow-based Ethernet switches with centralized controller
- Principles
  - Network governed by high-level policies
  - Policy determine routing
  - Binding between packets and its origin

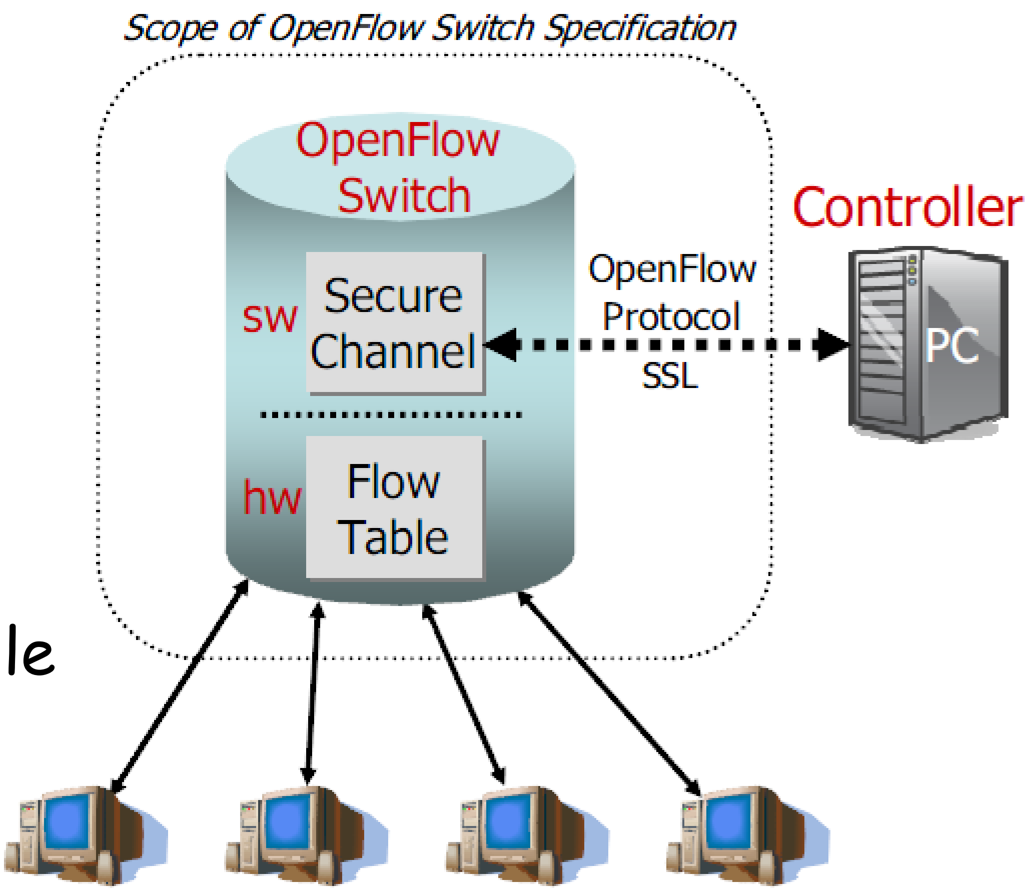




M. Casado et al. "Ethane: Taking Control of the Enterprise", Proceedings of ACM SIGCOMM, 2007.

Related Work

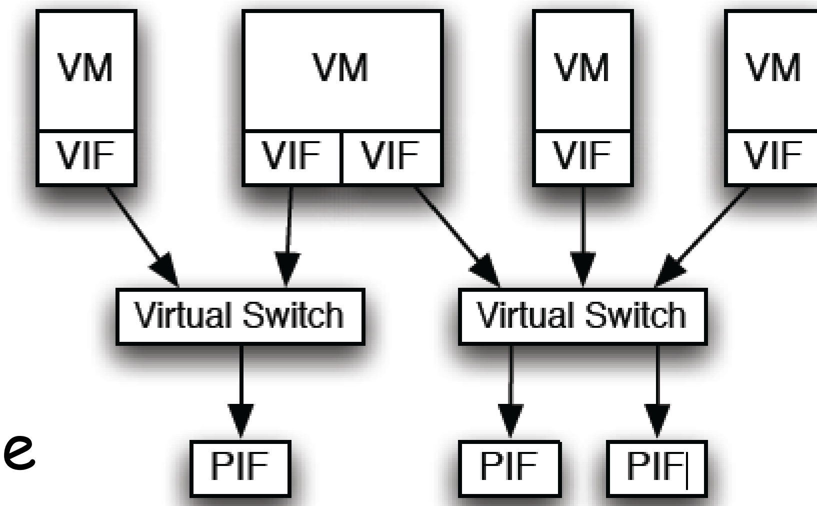
- OpenFlow Switch
  - Ethernet switches with flow tables
  - <header, action>
- OpenFlow Protocol
  - Enable programmable networks
  - Enable testing for new designs in productions networks



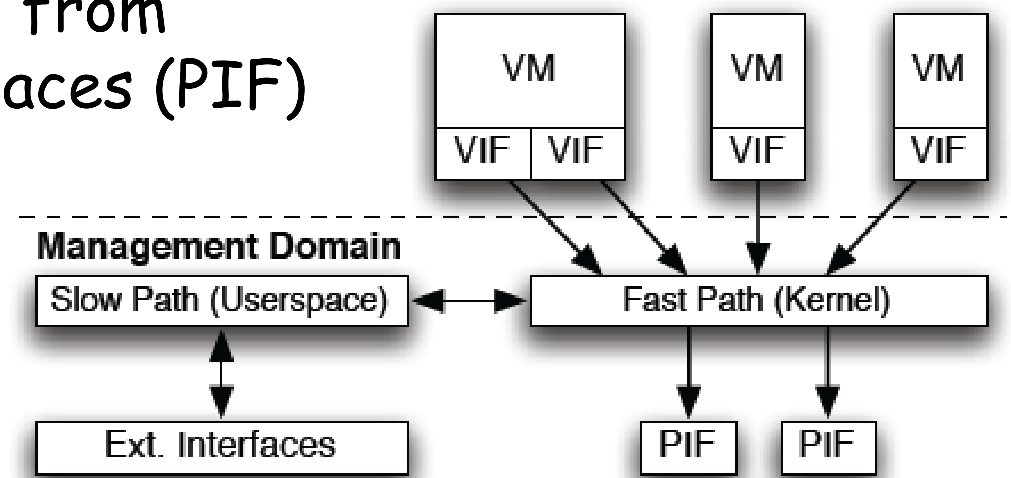
N. McKeown et al. "OpenFlow: Enabling Innovation in Campus Networks", ACM SIGCOMM Computer Communication Review, 38(2), April, 2008.

Related Work

- Open vSwitch
  - Virtualize switch functionality in software
  - Virtual interfaces (VIFs) are virtualized from physical interfaces (PIF)

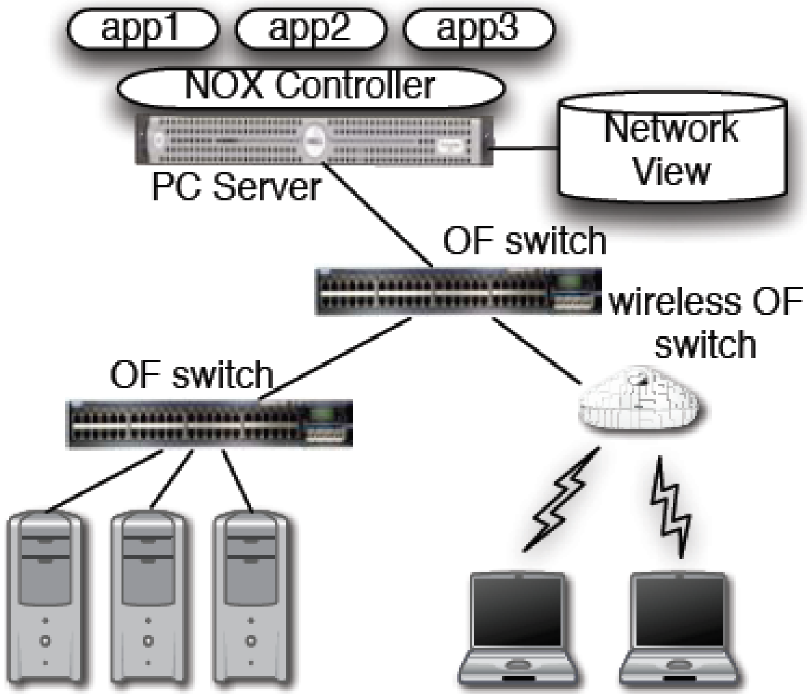


Architecture of the Open vSwitch



B. Pfaff et al. "Extending Networking into the Virtualization Layer", ACM Workshop on Hot Topics in Networks (HotNets), 2009.

Related Work



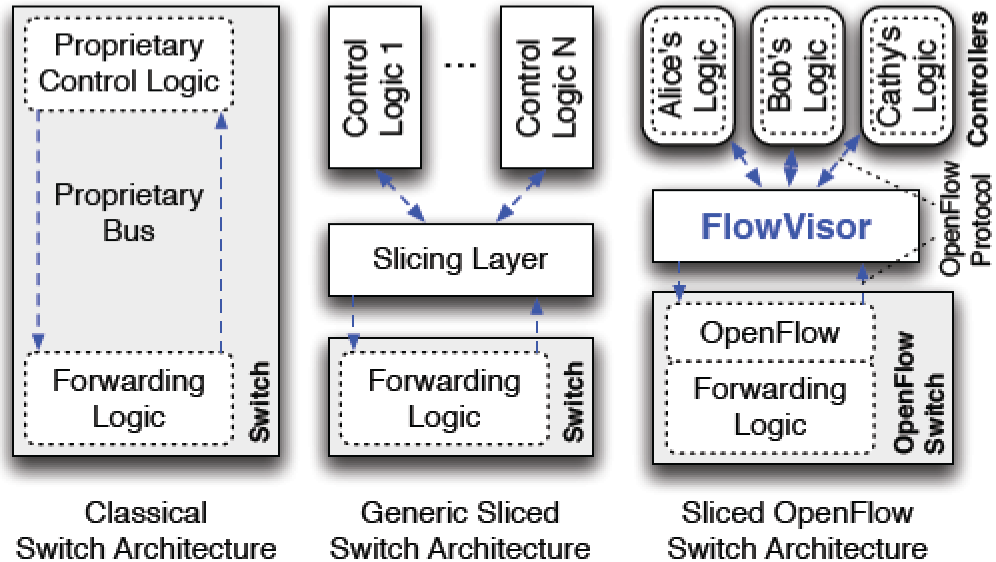
- Network OS
  - Distributed systems
  - NOX (~32K lines)
    - Developed side-by-side with OpenFlow
  - Onix (~150K lines)
    - production-quality control platform
- Program Interface
  - Event driven
  - Network view
  - High-level API

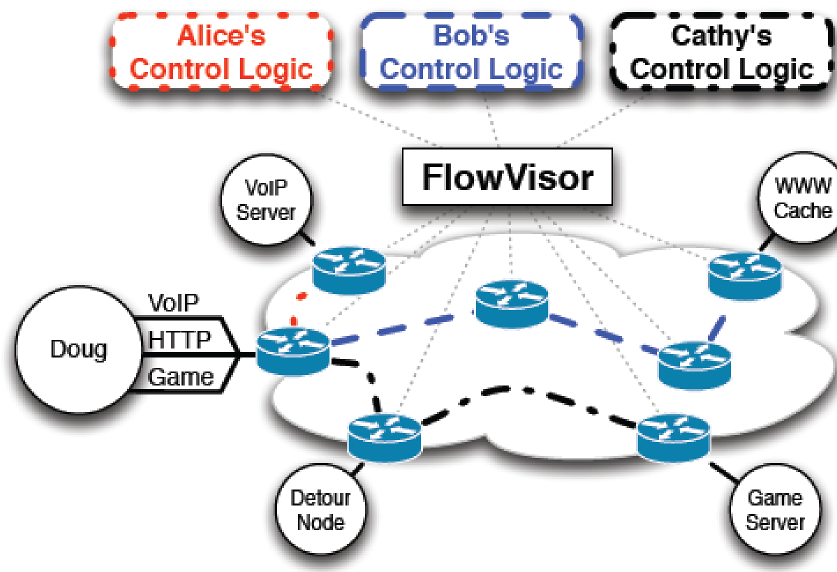
N. Gude et al. "NOX: Towards an Operating System for Networks", ACM SIGCOMM Computer Communication Review, 38(3), July, 2008.

T. Koponen et al. "Onix: A Distributed Control Platform for Large-scale Production Networks", USENIX OSDI, 2010.

Related Work

- Virtualize network
  - Flow space slicing
  - A new slicing layer between forwarding and control planes
- FlowVisor
  - Proxy for multiple OpenFlow controllers
  - Used to experiment on production nets

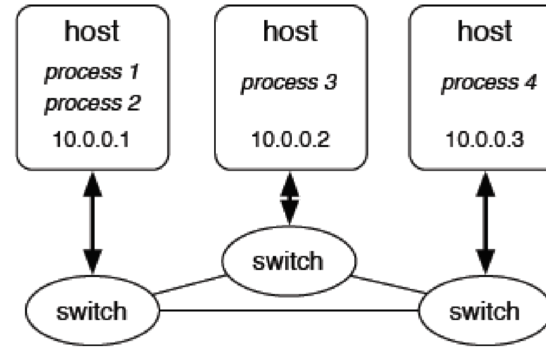


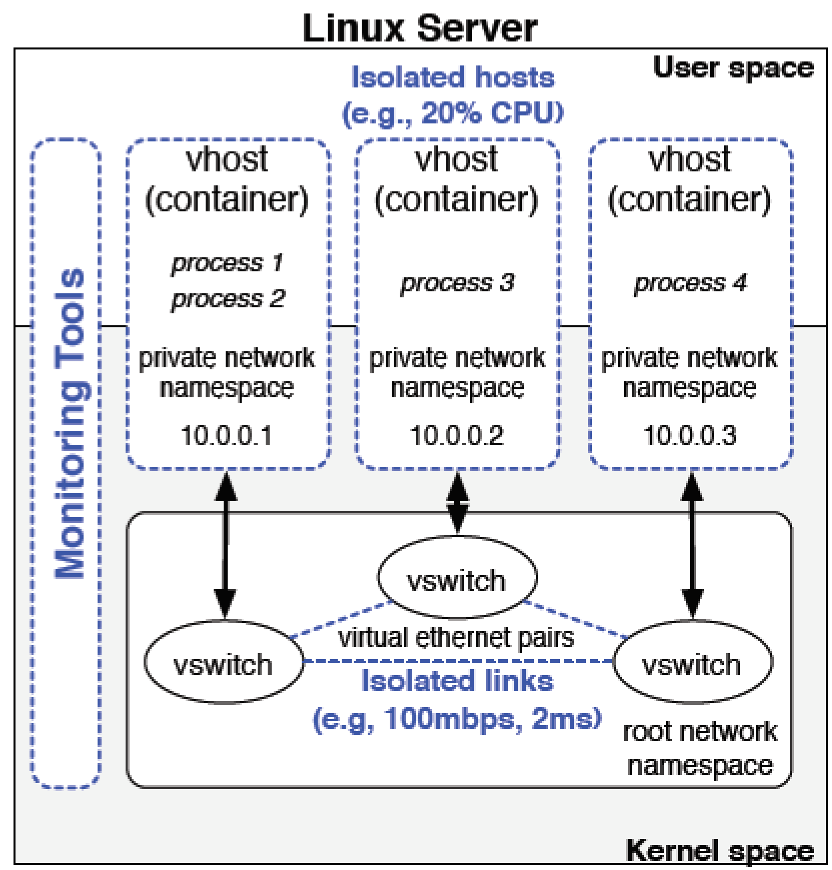


R. Sherwood et al. “Can the Production Network Be the Testbed?”, USENIX OSDI, 2010.

Related Work

- Mininet
  - Emulation platform of network systems
- Design Features
  - Using Linux Container mechanism
  - Different namespaces for virtual hosts
  - Virtual software switch implemented in Kernel

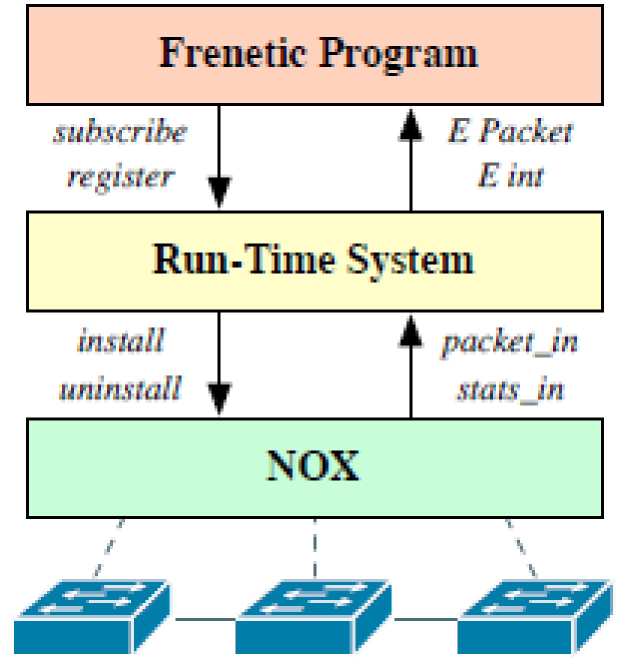




N. Handigol et al. "Reproducible Network Experiments Using Container-Based Emulation", ACM CoNEXT, 2012.

Related Work

- Frenetic and Pyretic
  - Control switches via policies
  - North-bound APIs
  - Frenetic: runtime system, functional reactive language
  - Pyretic: Python & POX based, support multiple concurrent tasks via operation composition



N. Foster et al. "Frenetic: A Network Programming Language", ACM SIGPLAN International Conference on Functional Programming (ICFP), 2011.

C. Monsanto et al. "Composing Software-Defined Networks", Proceedings of USENIX NSDI, 2013.