Tutorial on Software-Defined Networking

Maurício Tsugawa
University of Florida

PRAGMA Cloud Computing and Software-Defined Networking (SDN) Technology Workshop

March 20, 2013

Outline

- Introduction to Software-Defined Networking
- Traditional Networking vs. SDN
  - Case study with VLAN
- ViNe IP overlays
  - UF research project
  - How it relates to SDN

What is Software-Defined Networking?

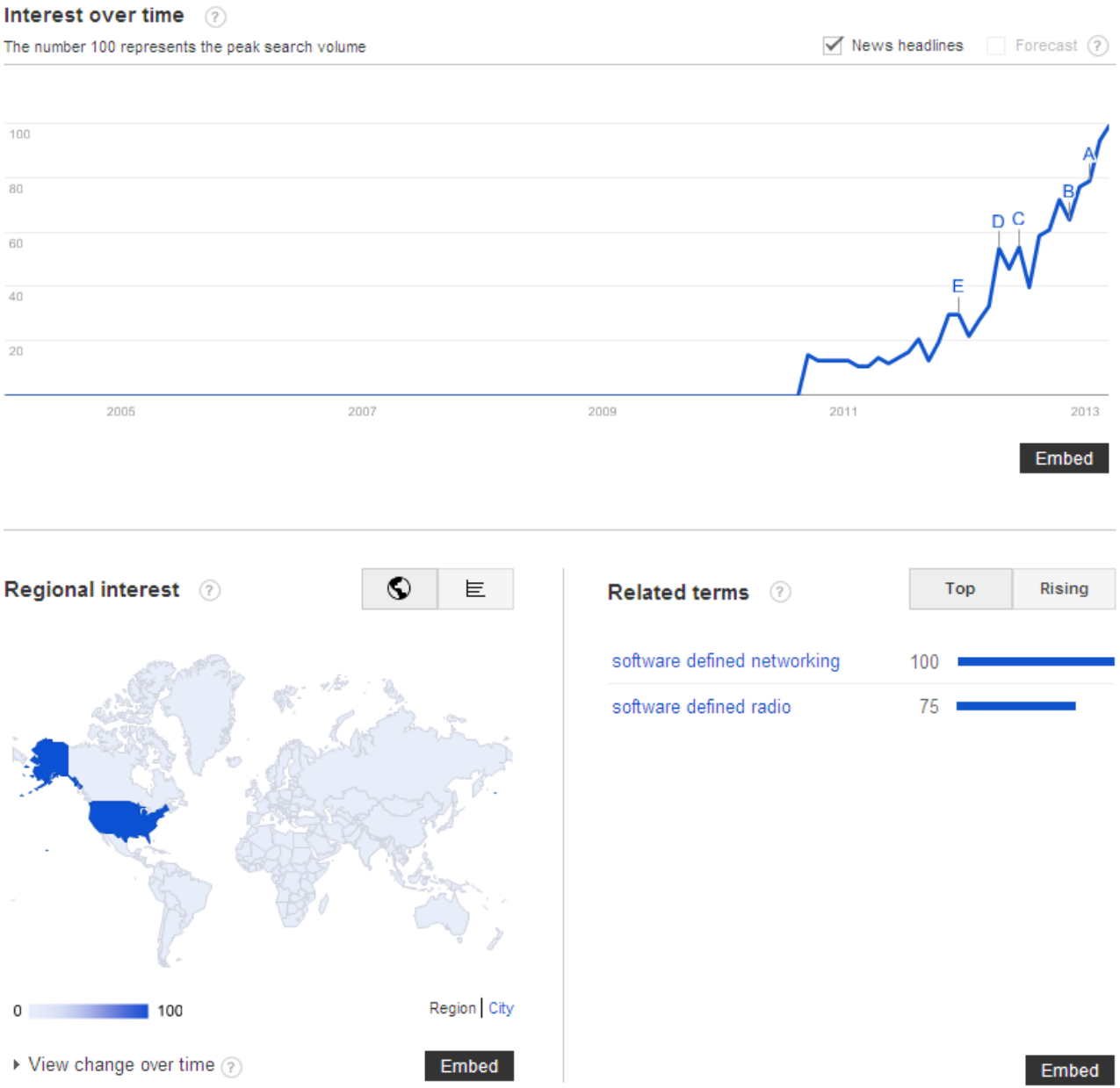
- Broad Definition
  - Open Network Foundation: “an architecture that enables direct programmability of networks”
  - Internet Engineering Task Force: “an approach that enables applications to converse with and manipulate the control software of network devices and resources” – *Internet Draft, Sep. 2011 by T. Nadeau*

What SDN is not

- OpenFlow
  - An approach to SDN with physical separation between control and data planes
  - Provides open interfaces (APIs)
  - Existing mechanisms can be implemented using OpenFlow, but existing mechanisms cannot implement Openflow
- SDN does not solve all networking problems
  - Offers a clean architecture to apply software engineering to networking
  - Enable buying network control system separately from hardware (switches)

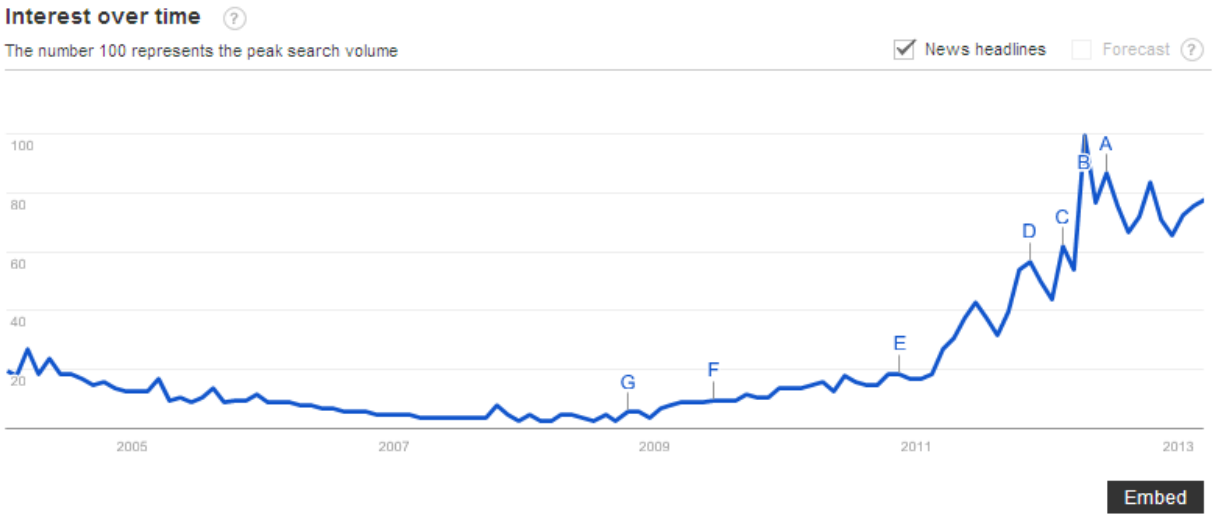
Software-Defined Networking

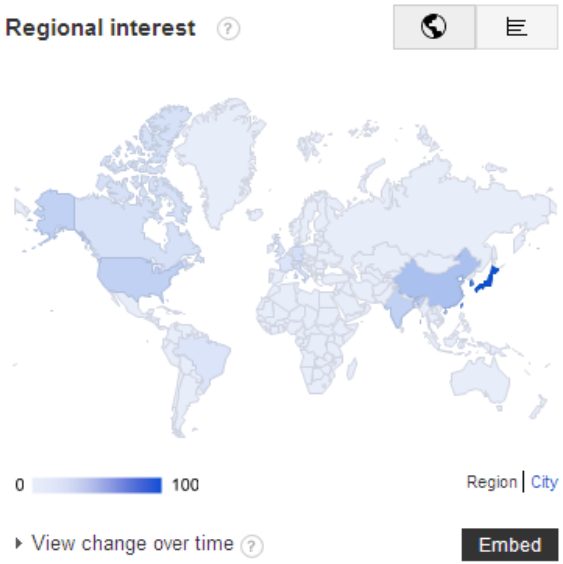
- Google Trends
  - Software-Defined Networking
- 2008: OpenFlow interface and NOX
- 2011: Open Networking Foundation

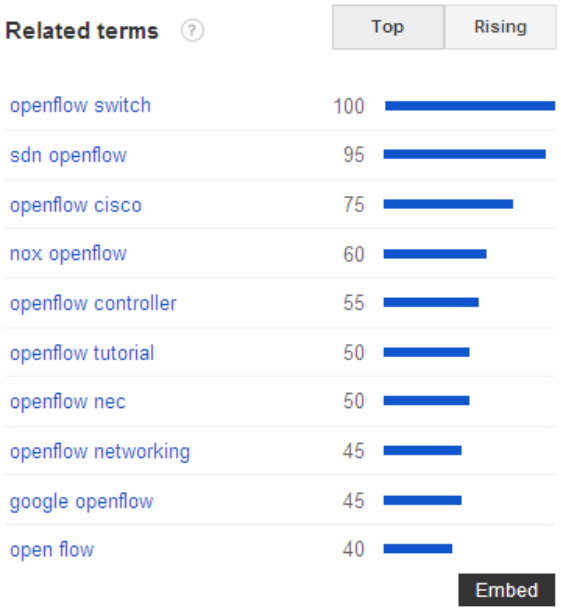


OpenFlow

- Google Trends
  - OpenFlow







Need for SDN

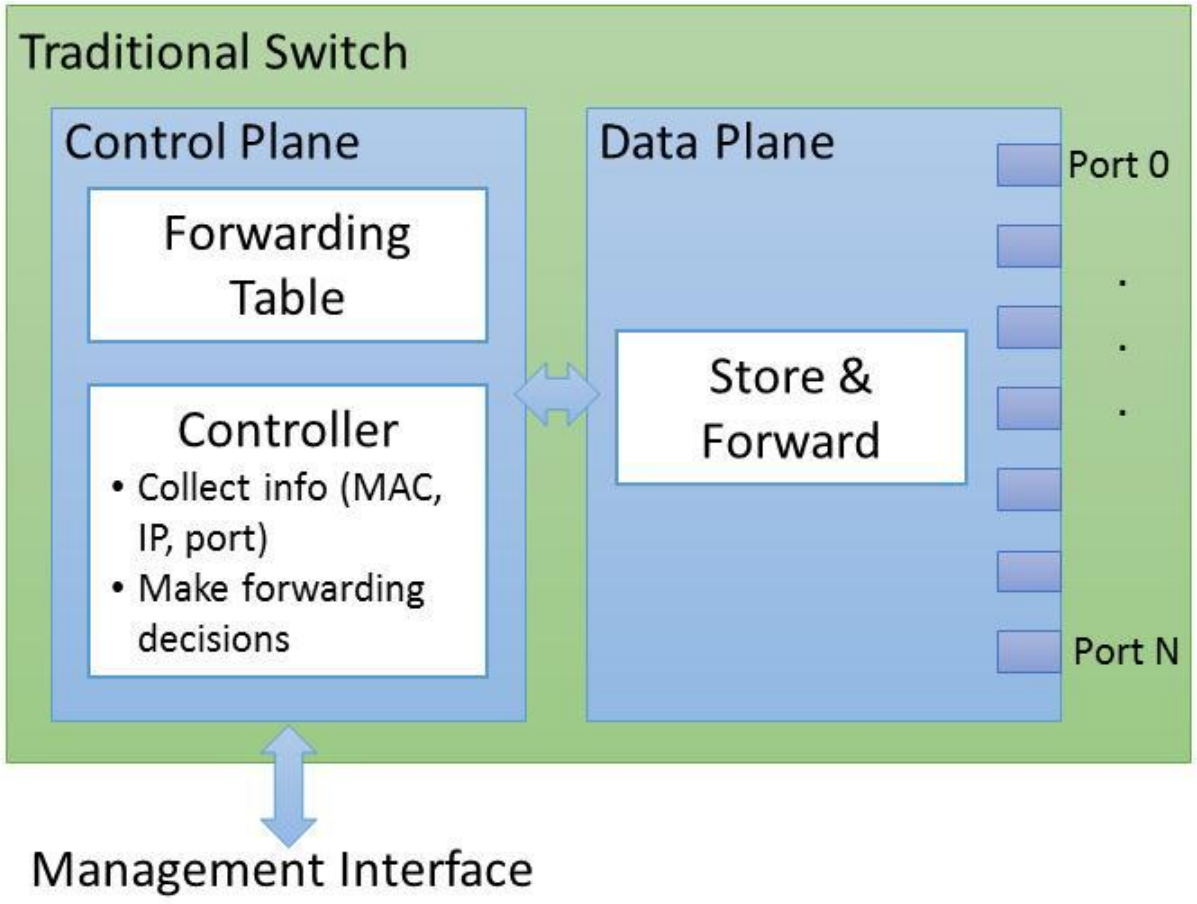
- Network infrastructure “ossification”
  - Large base of equipments and protocols
  - Networking experiments cannot compete with production traffic
  - No practical way to test new network protocols in realistic settings
- Closed systems
  - Vendor lock-in
  - Proprietary management interfaces – lack of standard or open interfaces
  - Hard to establish collaborations

Complexity in Networking

- Every problem requires an end-to-end solution
  - Involve all routers/switches (configure each one of them)
  - Consider the entire network topology
  - Interface with heterogeneous low level hardware/software
- Too many solutions/mechanisms
  - Routing, VLAN, VPN, MPLS, ACL, firewall
  - RFCs/IEEE standards
  - No software re-use

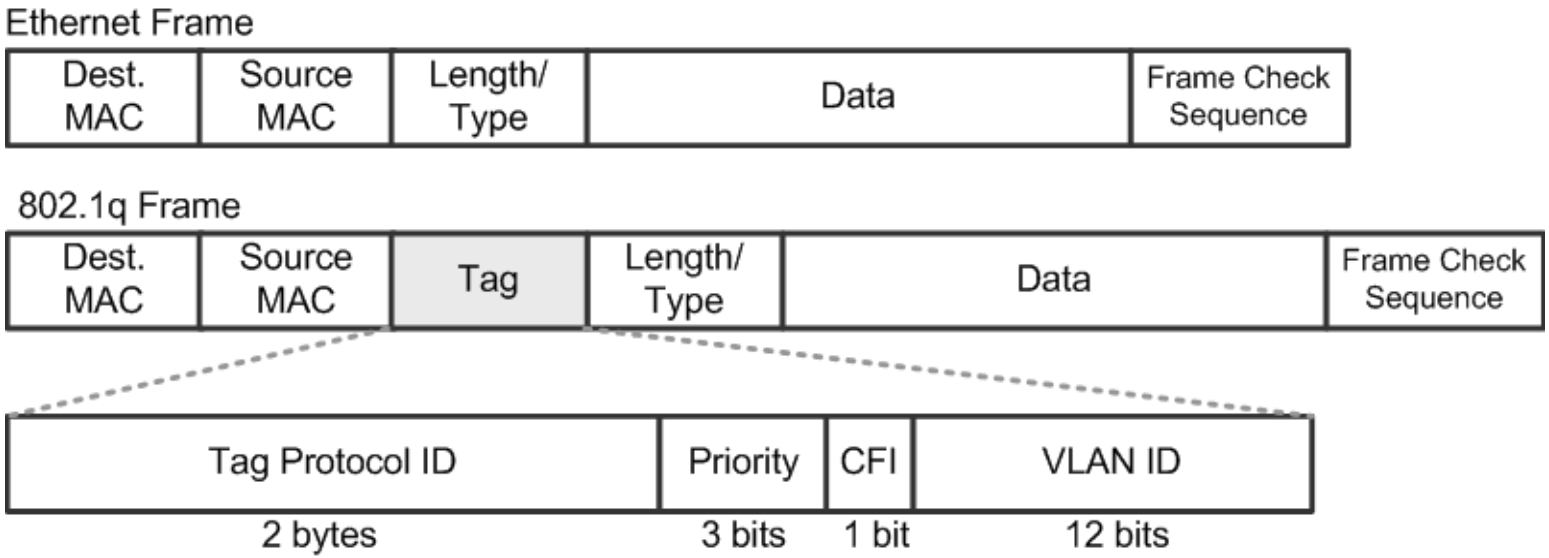
Traditional Switch/Router

- Control Plane and Data Plane on the same box

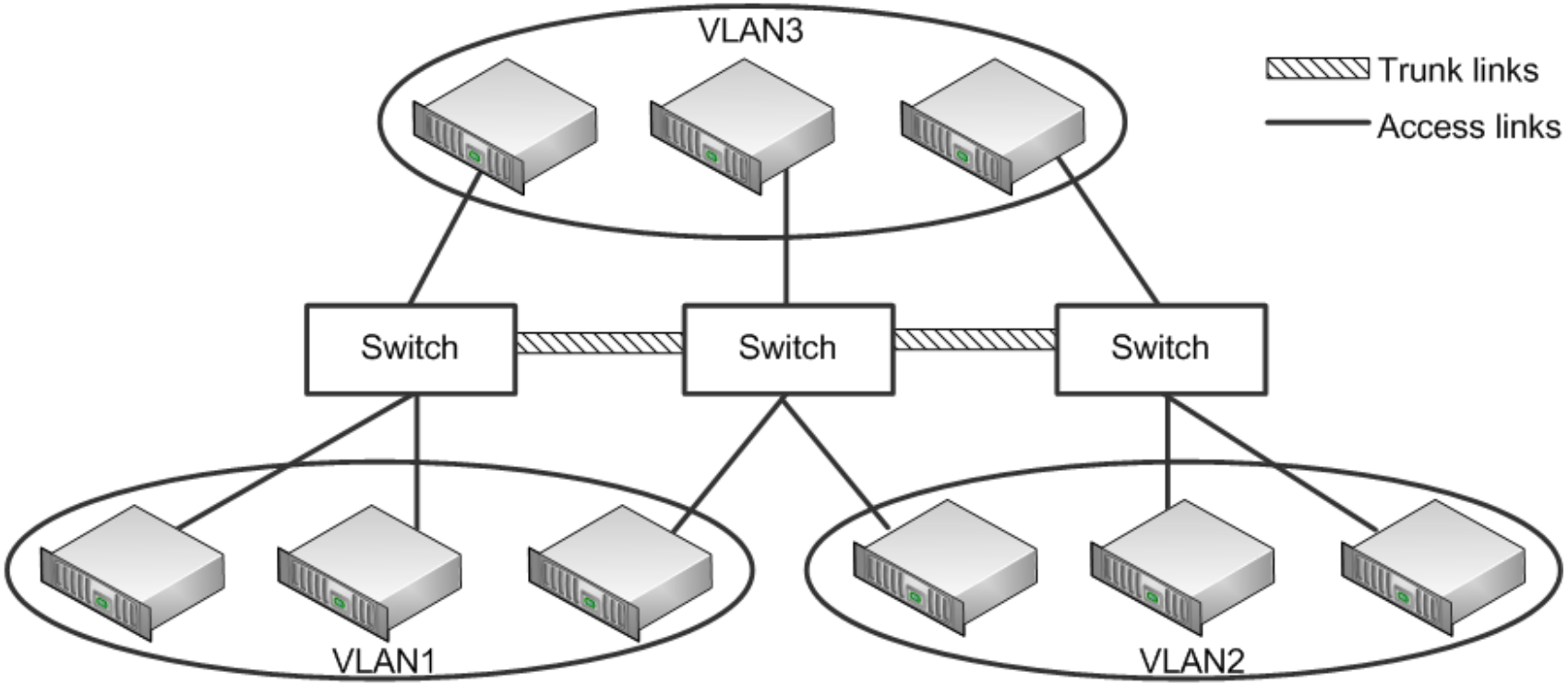


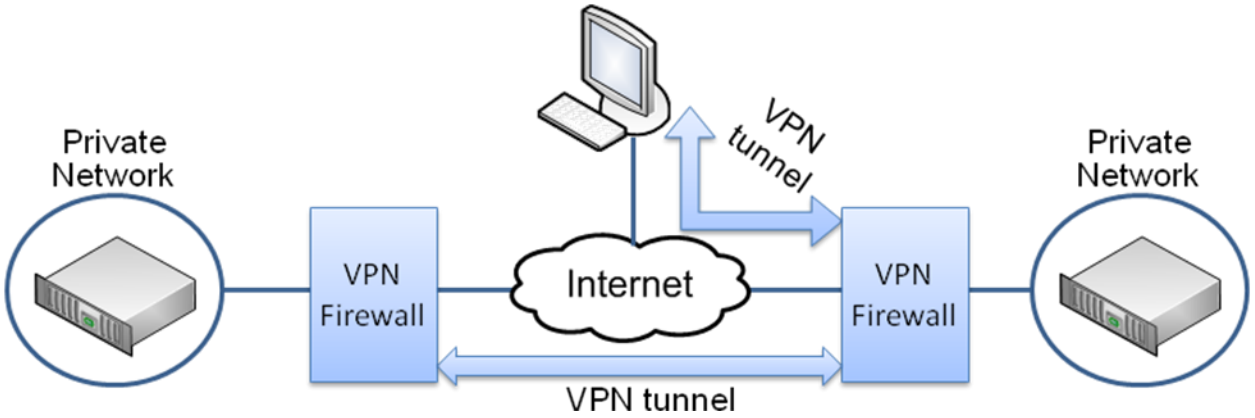
VLAN/802.1q

- Enables the co-existence of multiple broadcast domains (LANs) on the same infrastructure
- Ethernet frame extended to accommodate VLAN information



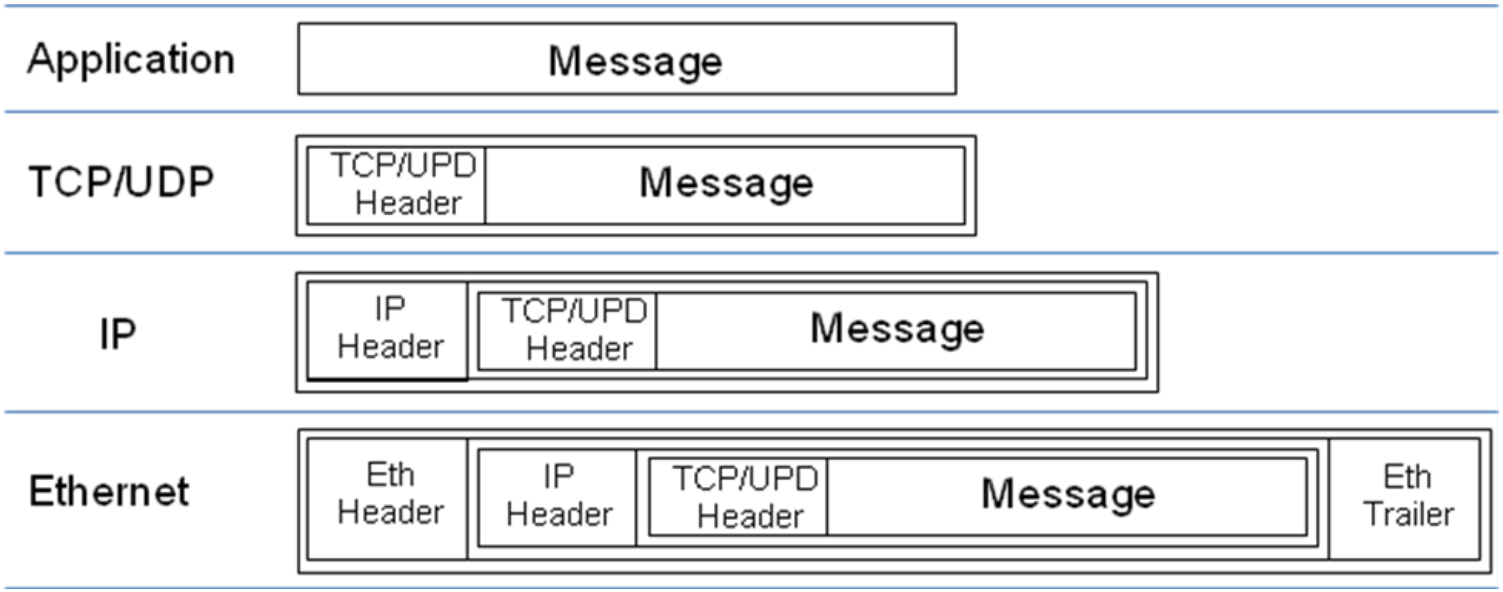
VLAN/802.1q


VLAN3
Trunk links
Access links
Switch
Switch
Switch
VLAN1
VLAN2

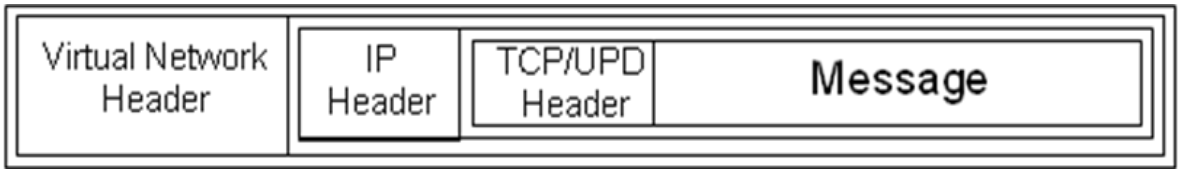

Private Network
VPN Firewall
Internet
VPN tunnel
VPN Firewall
Private Network
VPN tunnel

Tunneling/Encapsulation

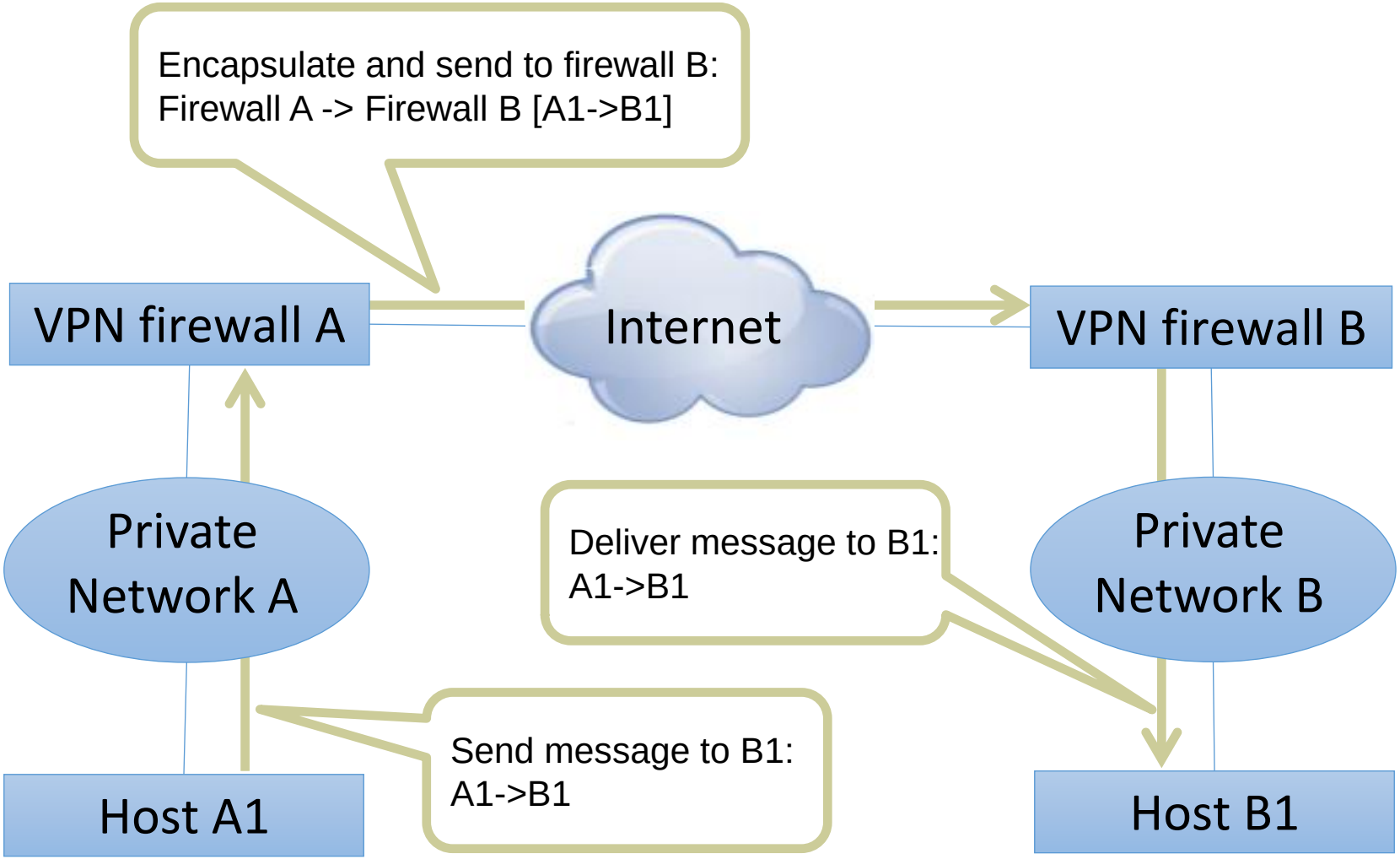
- Headers are added as messages are processed in each layer of network stack



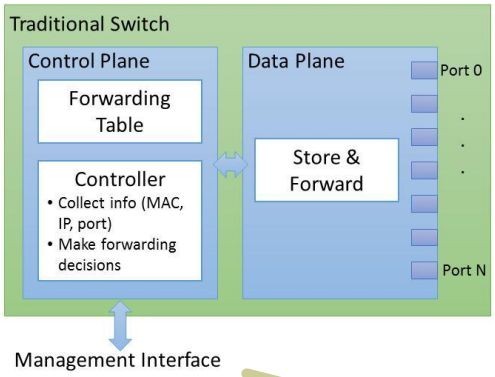
- IP packet encapsulation: additional header is added



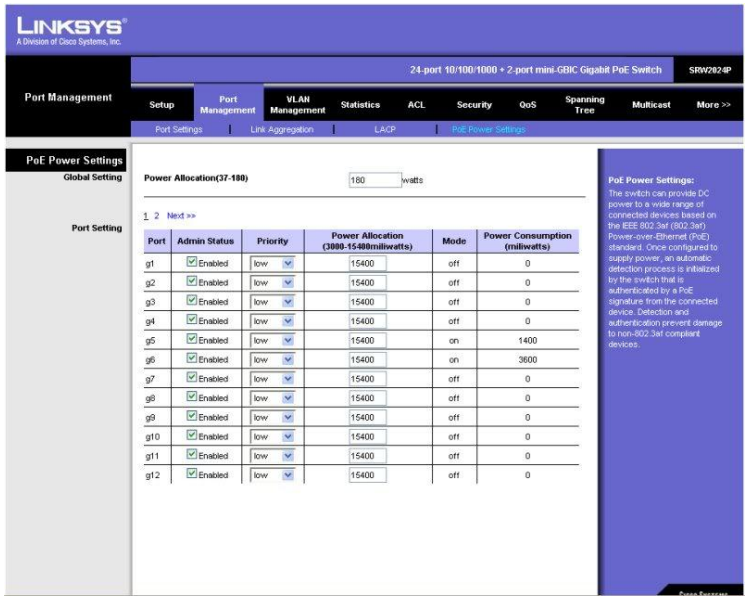
Tunneling


Encapsulate and send to firewall B:
Firewall A -> Firewall B [A1->B1]
VPN firewall A
Internet
VPN firewall B
Private Network A
Deliver message to B1:
A1->B1
Private Network B
Send message to B1:
A1->B1
Host A1
Host B1

VLAN configuration

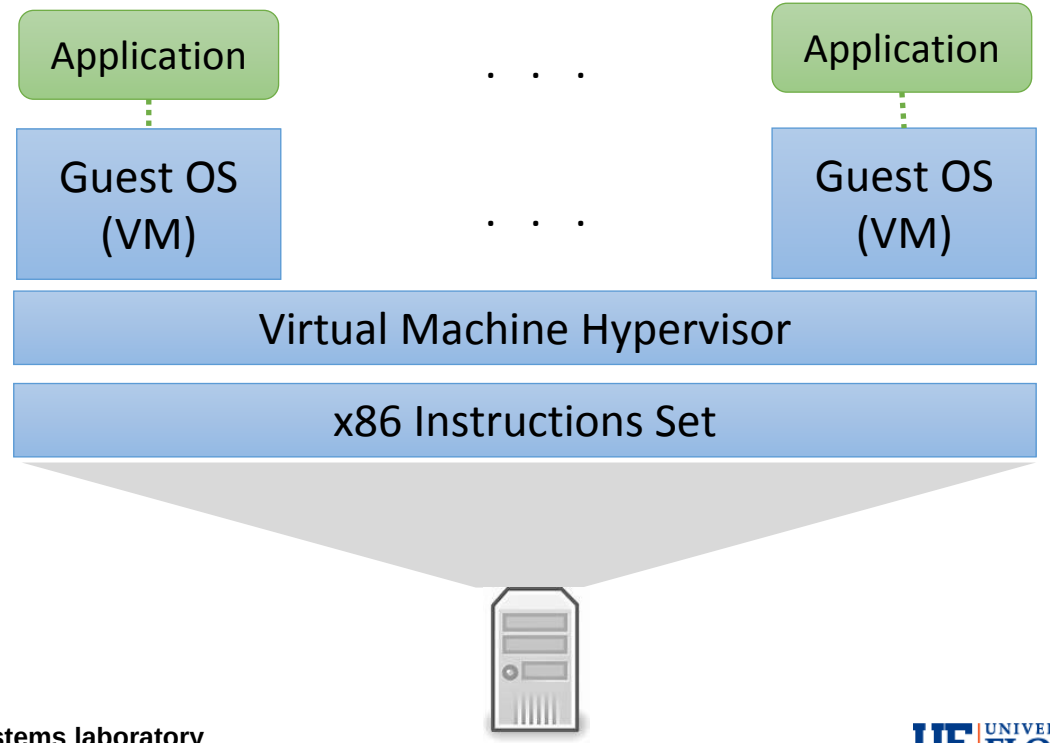



```
# enable
# cofigure terminal
# vlan 300 Test_VLAN
# interface gigabit 0/1
# switch port mode trunk
# switch port trunk encapsulation dot1q
# switch port trunk native vlan 1
# switch port trunk allowd vlan 1,300
# exit
# interface gigabit 0/2
# switch port access access vlan 5
```



Software Engineering

- Abstractions and interfaces
  - Decompose large problems into smaller (manageable pieces)
  - Separation of concerns
  - Code reuse

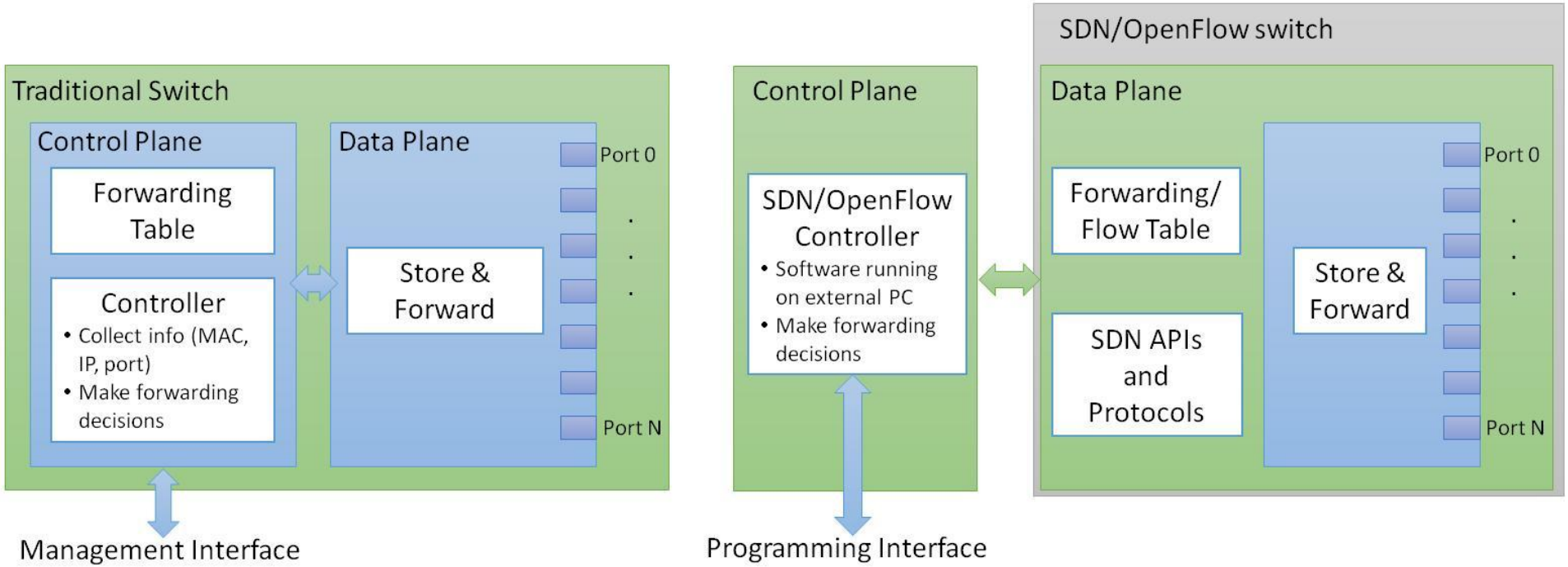


Networking Planes

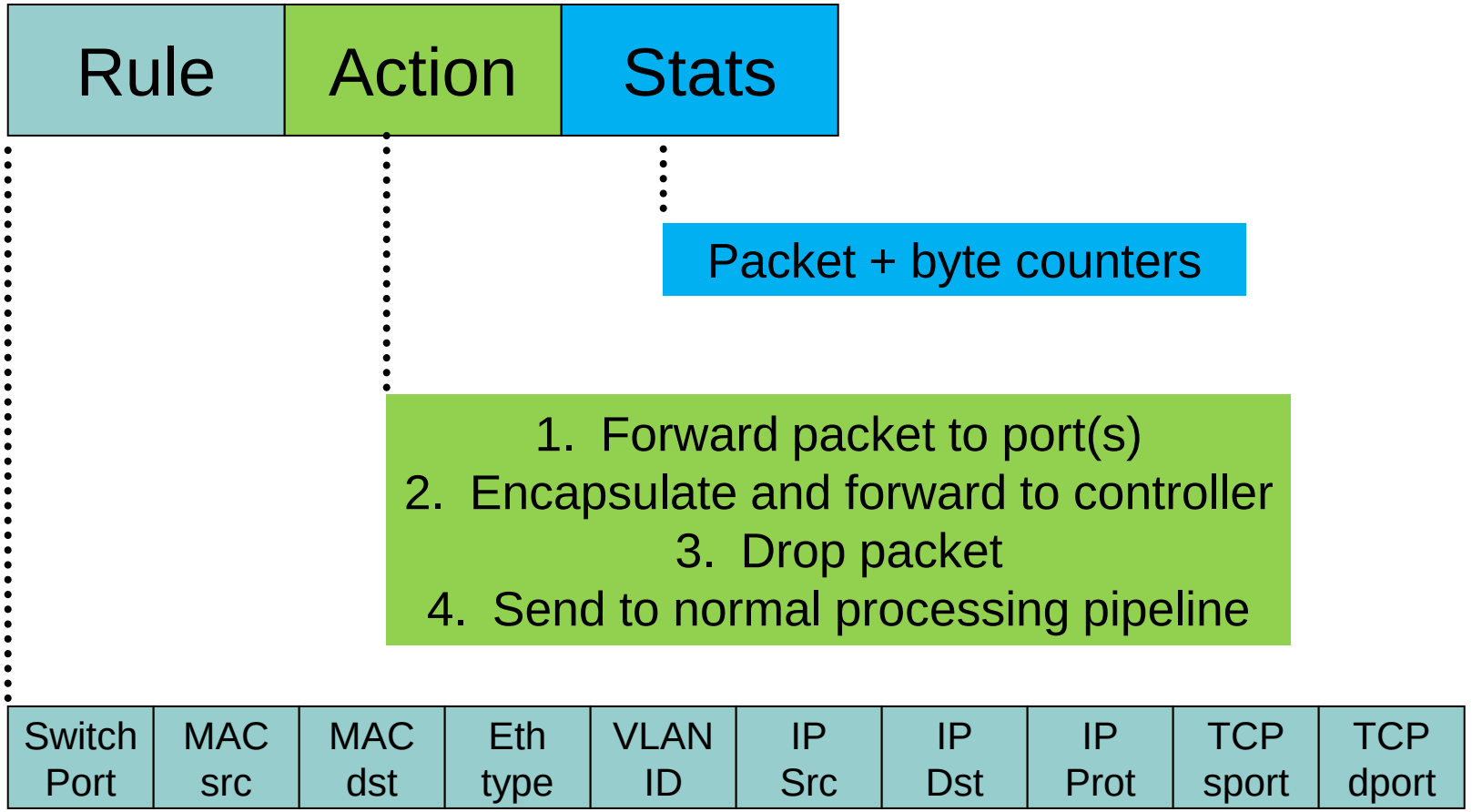
- Data Plane
  - Process messages/packets/frames according to local forwarding state
  - Implemented/optimized in hardware
- Control Plane
  - Adjust forwarding state
  - Distributed protocols/algorithms
  - Manual configuration and scripting
- SDN advocates full separation of control and data planes

OpenFlow Architecture

- Separate control plane and data plane
  - Run control plane software on general purpose hardware
  - Programmable data plane

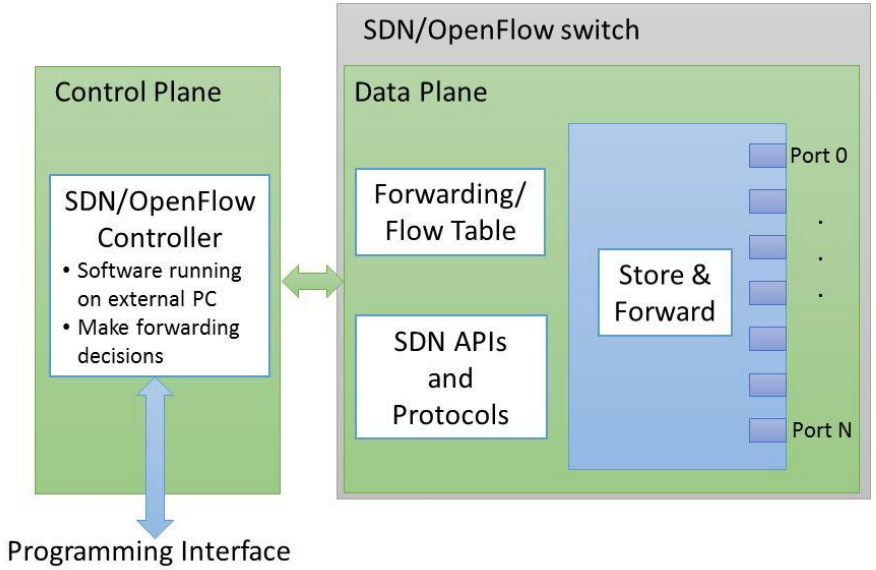


OpenFlow Flow Table Entry



Source: Nick McKeown, “Why Can't I Innovate in My Wiring Closet?”, MIT CSAIL Colloquium, April 2008

VLAN Configuration with SDN



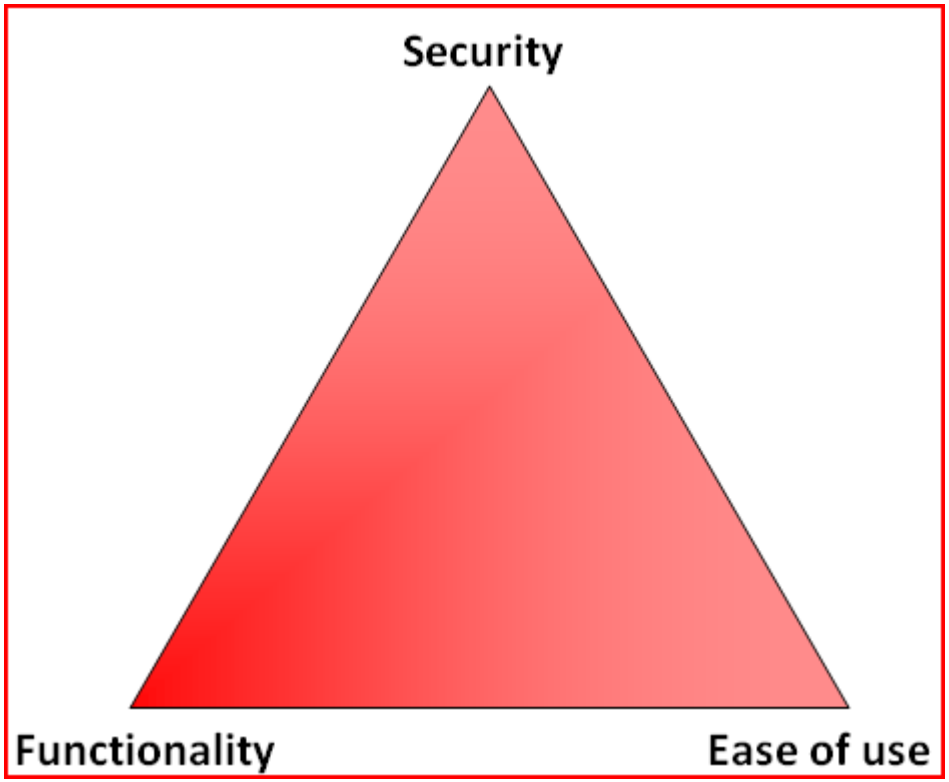
MAC Addresses of VLAN A

MAC Addresses of VLAN B

If src and dst MAC addresses belong to the same VLAN then deliver packet/frame

Security-Functionality-Usability

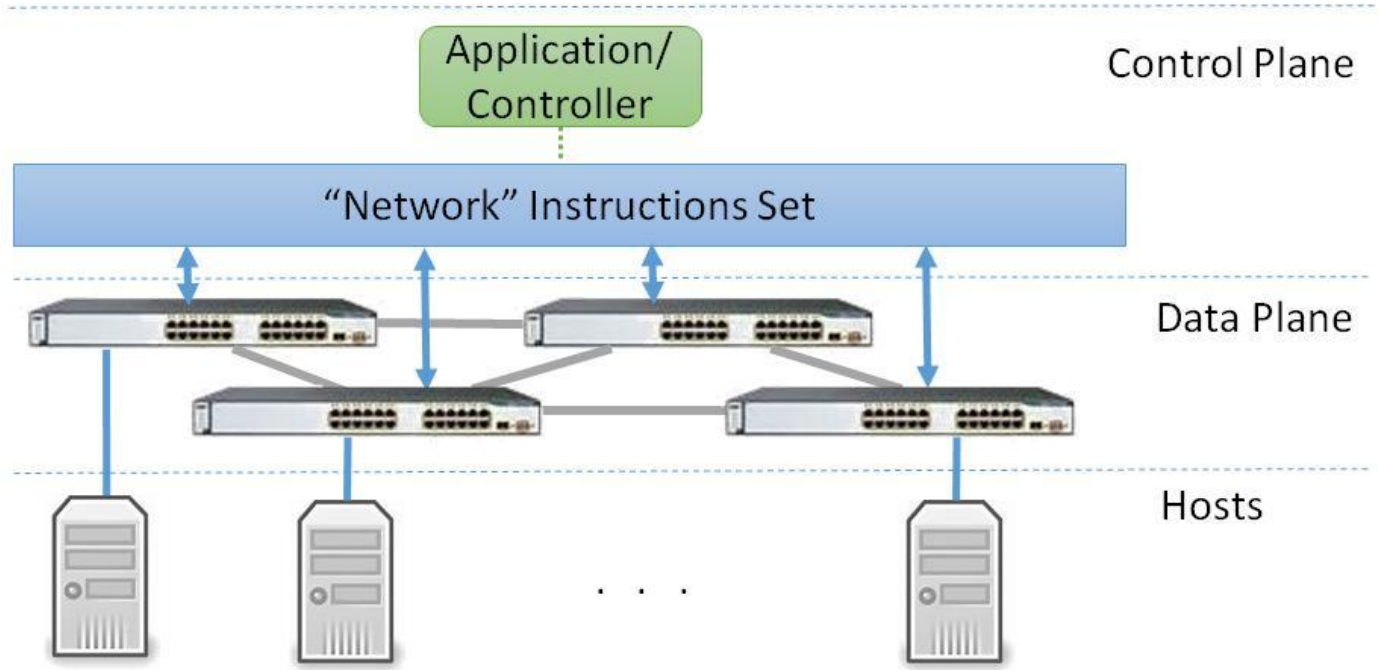
- OpenFlow and current SDN movement started as an academic project
  - Originally intended for campus networks
  - Now, many wish to apply it everywhere (including WAN)
  - Main focus on functionality (programmability of networks)



Source: Andrew Waite. InfoSec Triads: Security/Functionality/Ease-of-Use. June 12, 2010.

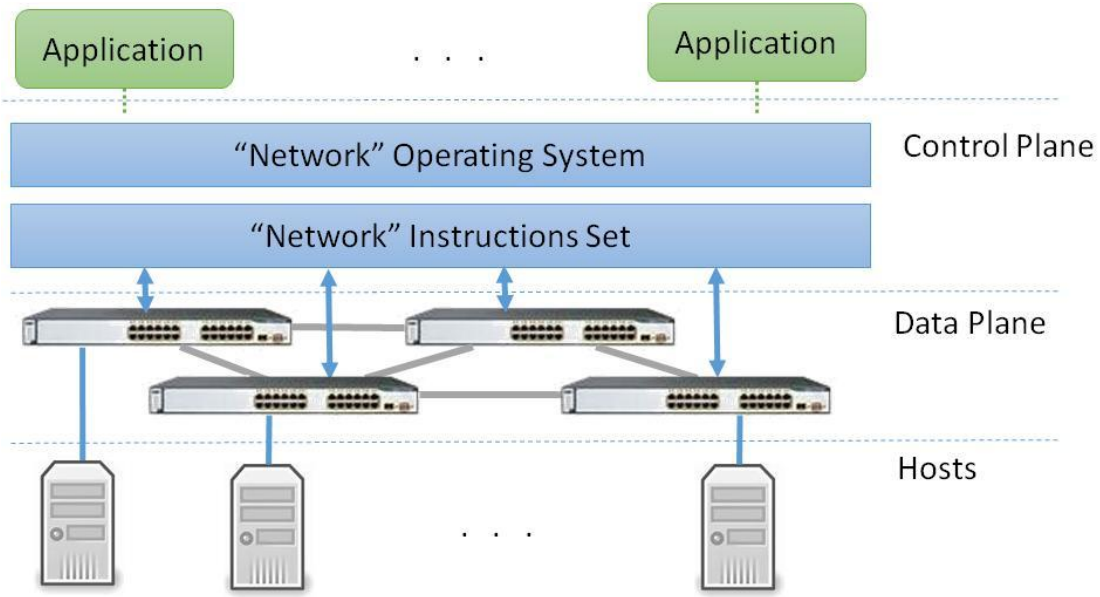
Stand-alone deployment

- Single controller
  - A user/administrator has full control of the network environment



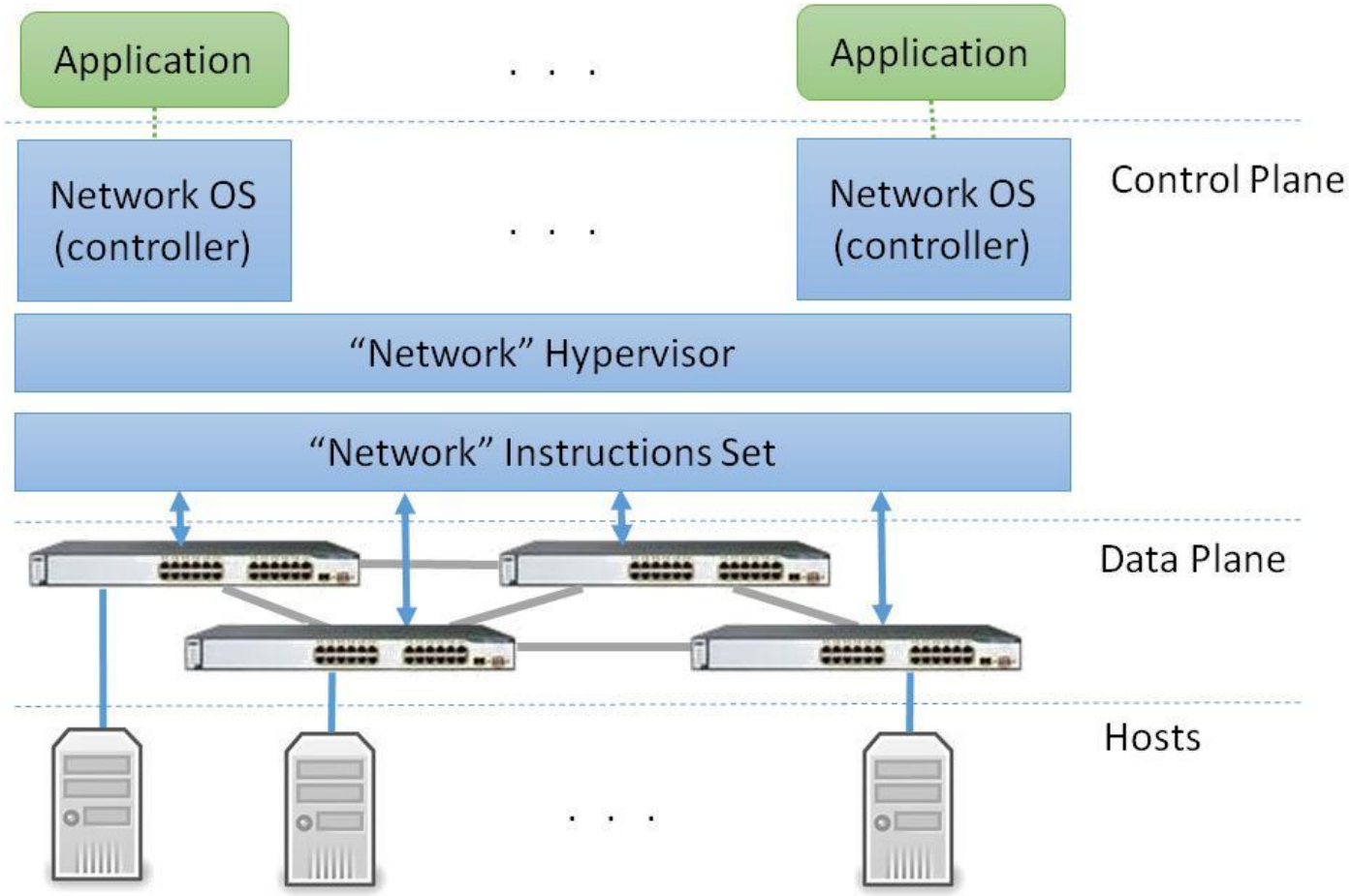
Multiple applications

- Network Operating System
  - Coordinate data plane resources
    - Each application needs to be identified
    - Needs AAA
  - “Network system call”-like interface needed
    - Accommodate conflicting requests
    - Potential vulnerabilities



Fully virtualized networking

- SDN offers functionality to implement network virtualization services



Many Projects/Implementations

- Software Switch
  - Open vSwitch
- Network Operating Systems
  - NOX, Trema, FloodLight, Maestro
- Hypervisor
  - FlowVisor
- Routing
  - RouteFlow
- Many others

Hands-on tutorials

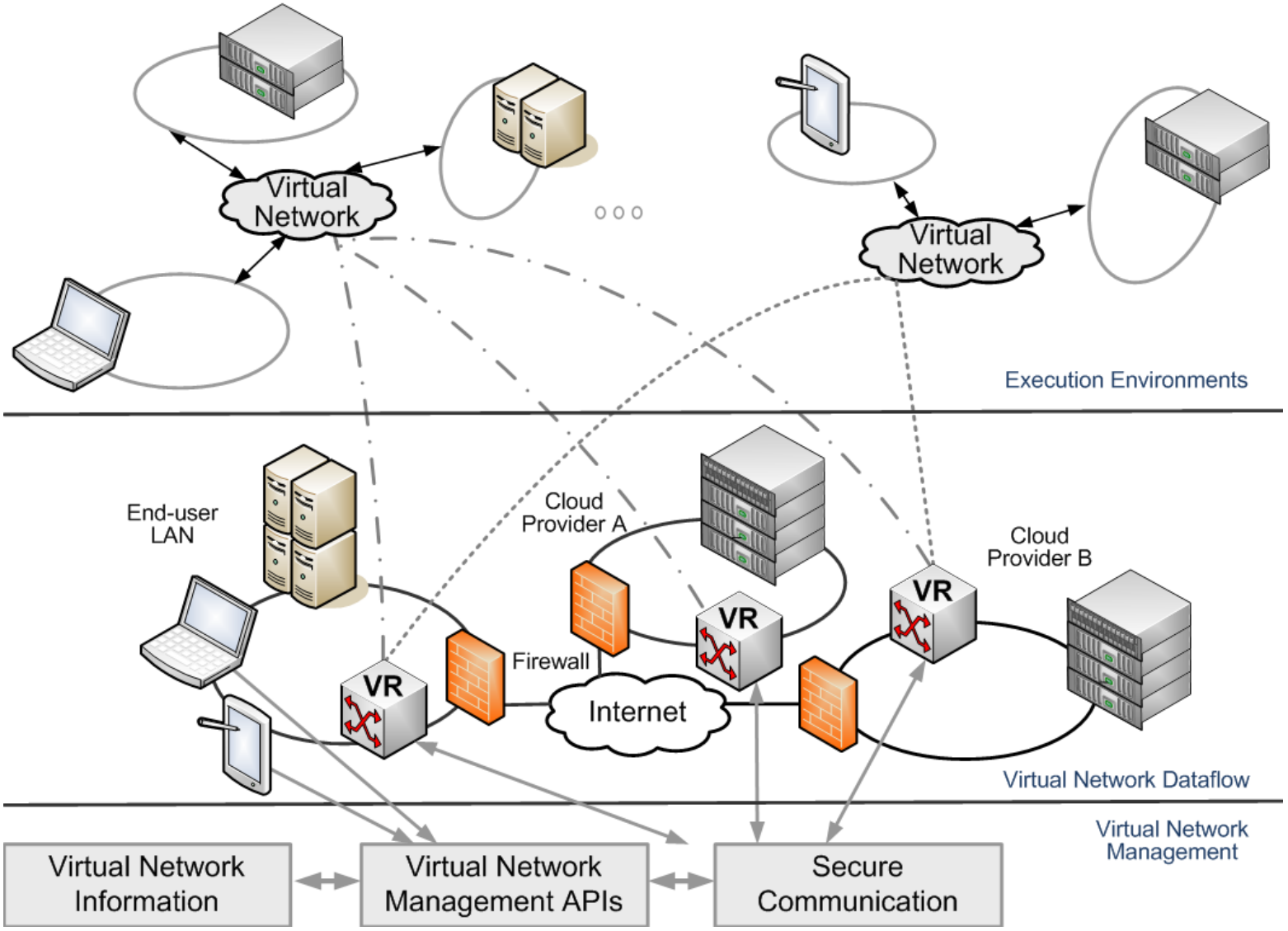
- http://www.openflow.org/wk/index.php/OpenFlow_Tutorial
  - Use tools pre-packaged in a VM
  - Modify a OpenFlow hub to a learning switch
  - Many controller/platform options

- http://trema-tutorial.heroku.com/
  - OpenFlow controller development using trema

Quick ViNe Overview

- ViNe implements routing and other communication mechanisms needed to deploy user-level virtual networks across WAN
- ViNe offers:
  - Full connectivity among machines (physical and virtual) on public and private networks – built-in firewall traversal
  - Multiple isolated overlays
  - Management APIs

ViNe Architecture



ViNe and SDN/OpenFlow

- Investigating mechanisms to integrate ViNe technology with SDN/OpenFlow
  - Help incremental deployment
  - Connect edges (servers running OVS) or OpenFlow islands
  - No immediate need for SDN in core networking
- Extend ViNe to talk OpenFlow
  - Implement OpenFlow controller features
  - Accept OpenFlow commands

Command-Line Interface design

- Completely anew
  - Define set of commands
  - Define what each command will do when executed
  - Use compiler techniques to implement CLI
    - Lexical analysis
    - Parsing
    - Regular expression
  - Implement the commands behavior

CLI design for ViNe

- ~~Completely anew~~
  - ~~Define set of commands~~ - want extensible set of commands
  - ~~Define what each command will do when executed~~ – link to existing APIs
  - ~~Use compiler techniques to implement CLI~~
    - ~~Lexical analysis~~
    - ~~Parsing~~
    - ~~Regular expression~~

Difficulties due to:
•Undefined set of commands
•Requirement of no recompilation

  - ~~Implement the commands behavior~~ – most are already implemented

Reconfigurable CLI

- Map commands to API
  - show lndt → call API that lists ViNe router's local routing table
- Reconfigurable commands set approach
  - Commands definition file describes how commands are mapped to APIs
    - Commands can be changed without recompilation
    - Commands behavior can be changed without recompilation

Commands definition file

```
; Each line is separated by a #
; The left part denotes what a user can enter as a command, and the
; right part shows what class + method is associated with that command.
; Words prefixed by an asterisk (*) are variables passed to the method.
; All variables are assumed to be Strings.
; Words prefixed by an dollar ($) are constant strings passed to the method.
; Any lines above a command prefixed by a '#' are descriptions of that command.

testCat          # vine.mgt.cli.CmdTest method1
testCat *ip      # vine.mgt.cli.CmdTest method2
test             # vine.mgt.cli.CmdAnotherTest.c printMe
```

- At startup commands definition table are loaded (can be reloaded at run-time)
- Commands are interpreted through table lookup

ViNe CLI demo