Software Defined Networking, openflow protocol and its controllers

Isaku Yamahata <yamahata@private.email.ne.jp>
<yamahata@valinux.co.jp>
VALinux Systems Japan K.K

LinuxCon Japan June 6th, 2012

Agenda

- SDN and openflow protocol
- Openflow controllers
- Related academic researches
- Openflow controller to network operating system

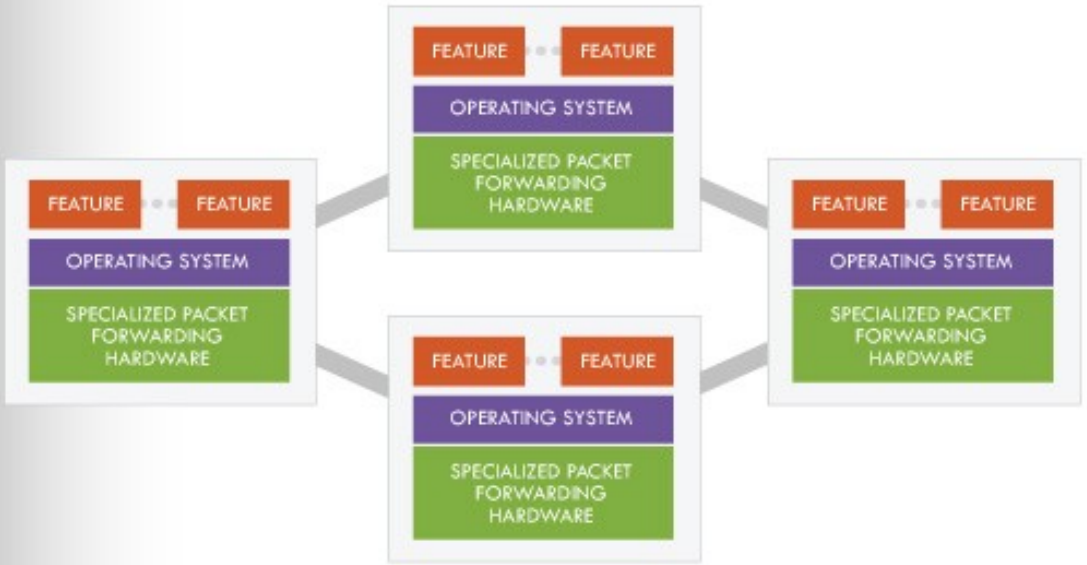
Software Defined Networking and Openflow protocol

SDN: Software Defined Networking

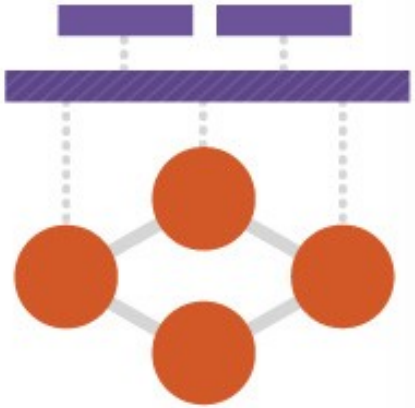
- http://opennetsummit.org/why.html
  - SDN is a new approach to networking and its key attributes include: separation of data and control planes; a uniform vendor-agnostic interface called OpenFlow between control and data planes; a logically centralized control plane; and slicing and virtualization of the underlying network. The logically centralized control plane is realized using a network operating system that constructs and presents a logical map of the entire network to services or control applications implemented on top of it. With SDN, a researcher or network administrator can introduce a new capability by writing a simple software program that manipulates the logical map of a slice of the network. The rest is taken care of by the network operating system.
- [Paraphrased from the HotSDN ‘12 Solicitaion]
  - Software Defined Networking (SDN) is a refactoring of the relationship between network devices and the software that controls them.

OpenFlow/SDN

OpenFlow/SDN Difference



Network of vertically integrated, closed, proprietary switches

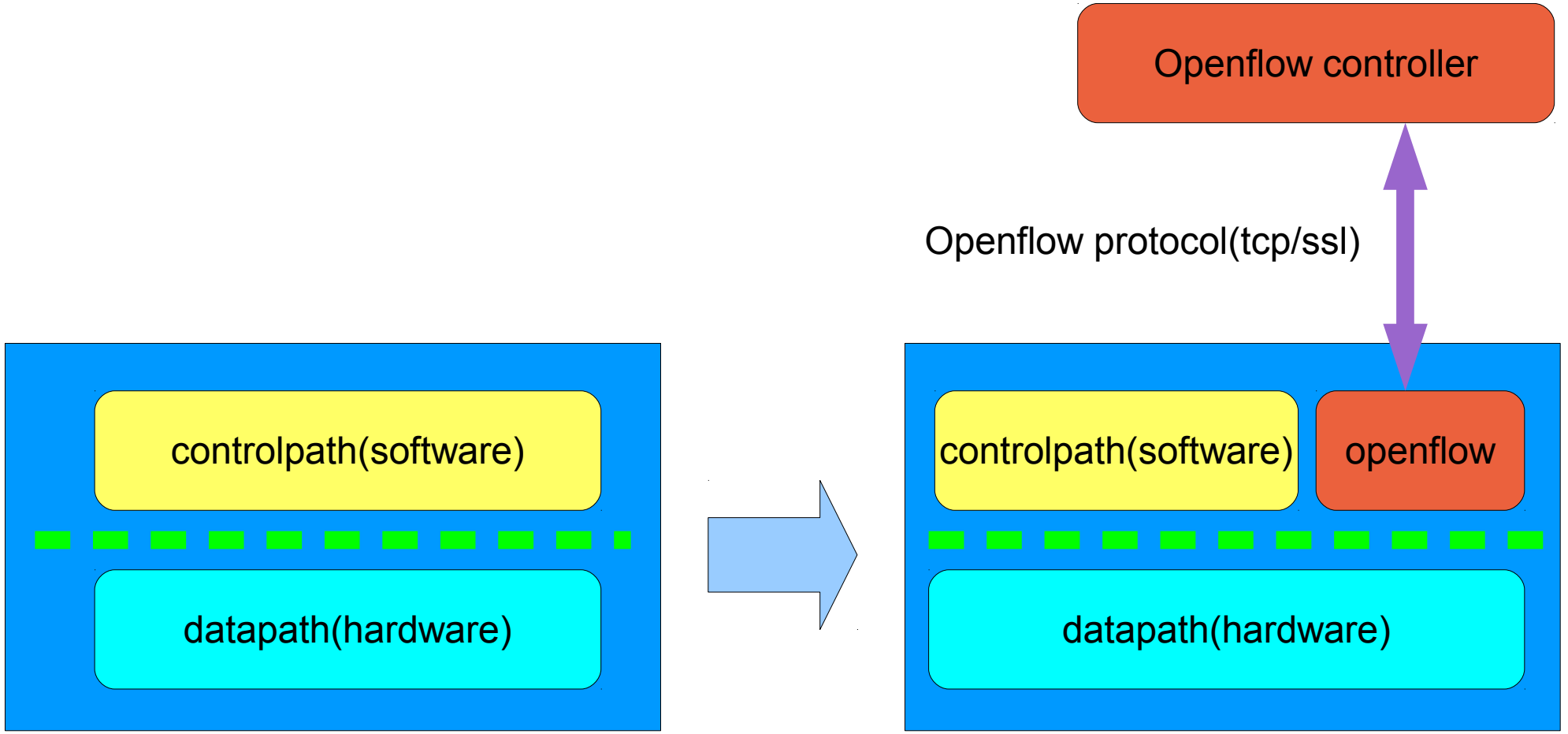


OpenFlow/SDN:

Separation of control and data plane
Open interface between control and data plane
Open interface to the control plane
Network control and management features in software

From http://opennetsummit.org/why.html

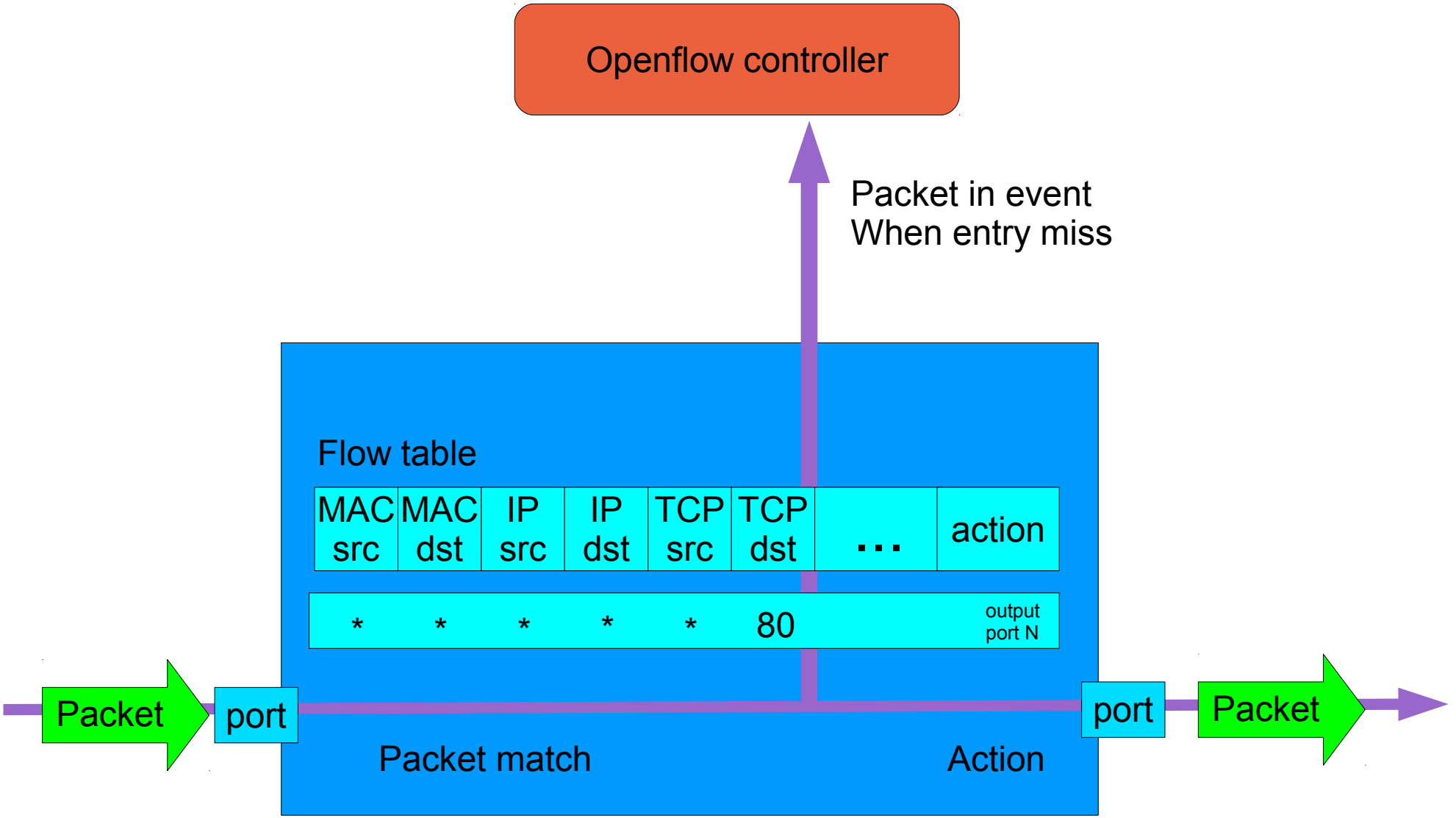
Openflow



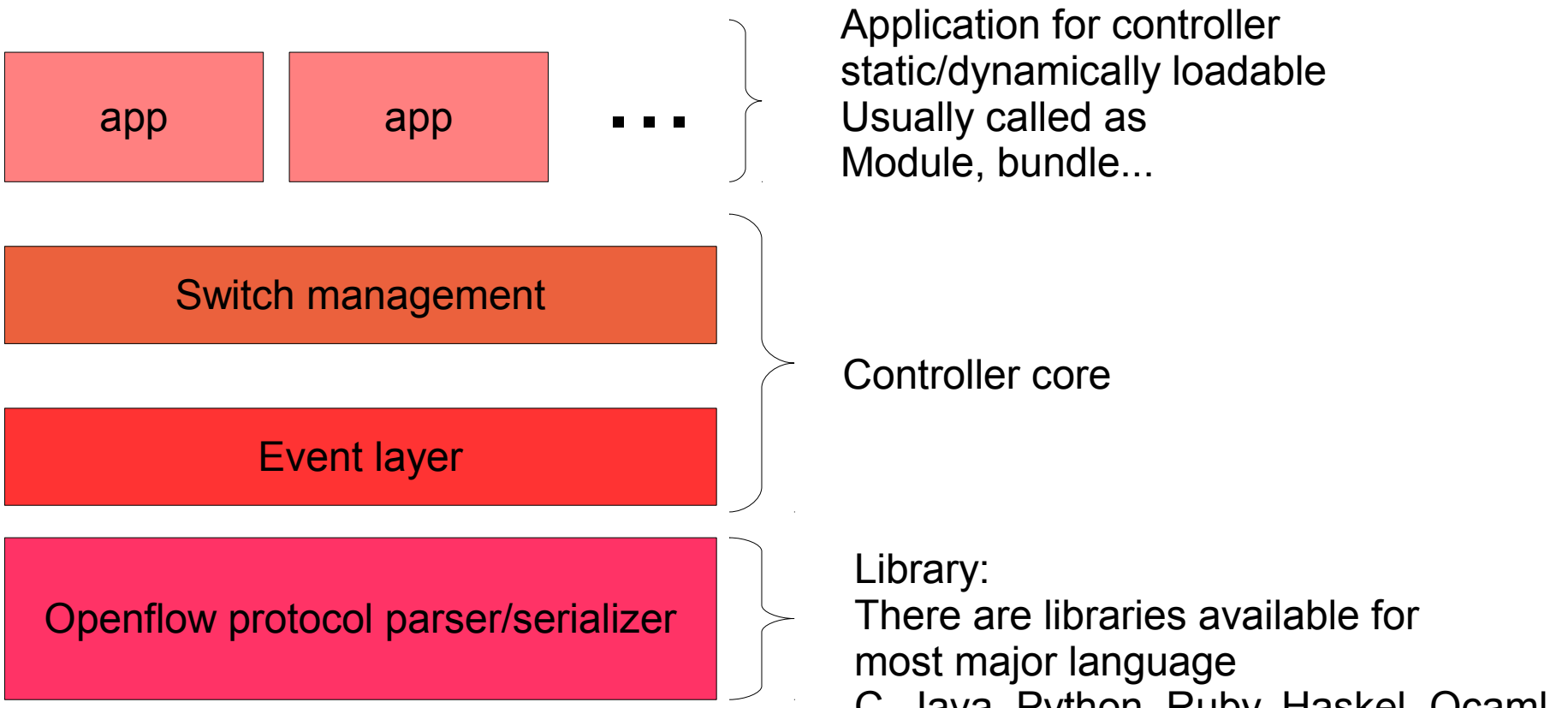
Ethernet switch

Openflow ethernet switch

Flow table and match/action



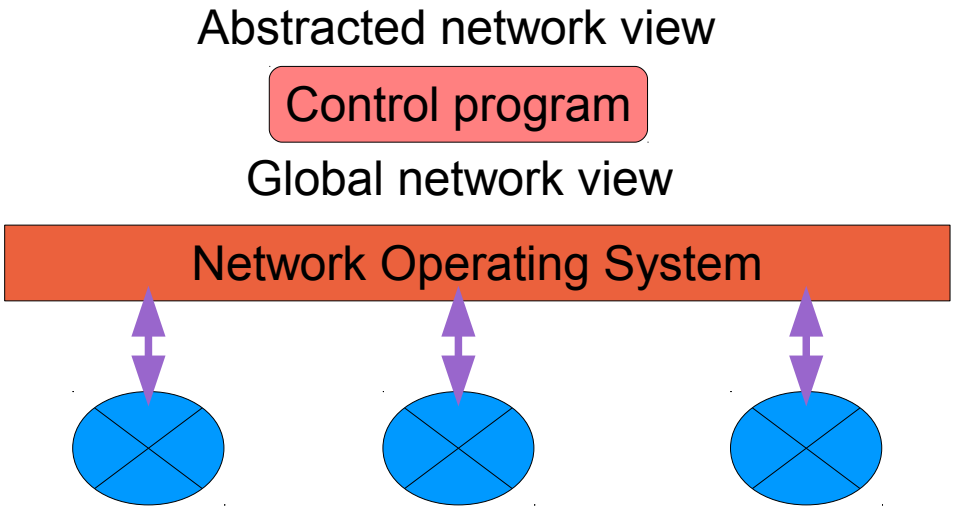
OpenFlow controller structure



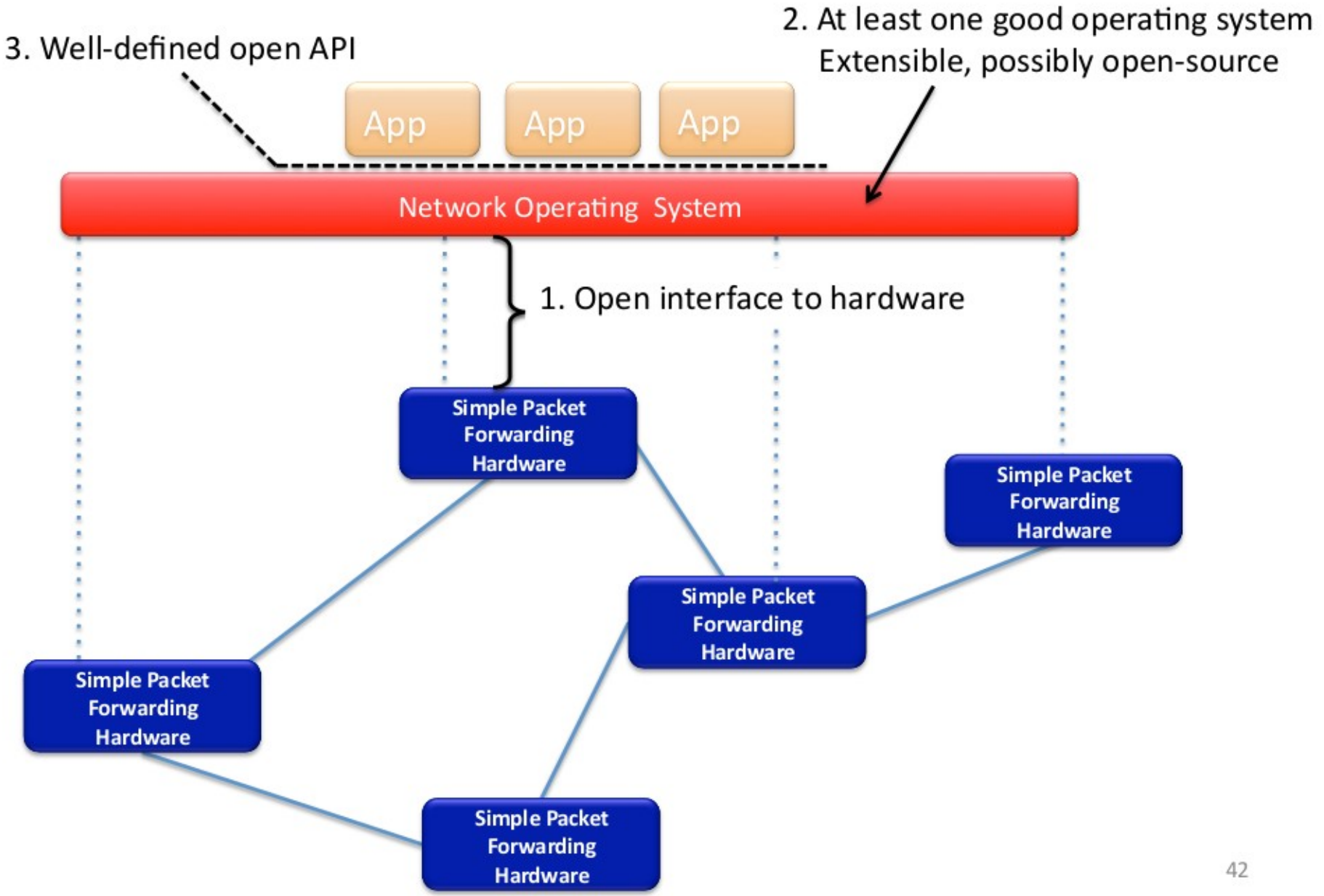
Network Operating System(NOS)

- Distributed system
- Communicate with forwarding planes
- Provides control programs
  - Abstract network view
    - State distribution abstraction
    - Specification abstraction
  - abstract interfaces to network application
    - NOS takes care of distributed details

- Controle program
  - Configuration = f(network view)



The "Software-defined Network"



http://opennetsummit.org/talks/ONS2012/heller-mon-intro.pdf

Openflow Controllers

NOX

- New NOX
  - Stanford Univ. UC Berkly,
  - GPL v3
  - C++
  - Native thread model
- NOX classic
  - Stanford Univ. Nicira
  - GPL v3
  - Python based on C++ and swig(http://www.swig.org)
  - Its own thread model
  - Esepcially for python support, threading is limited.
- http://www.noxrepo.org/nox/about-nox/
- https://github.com/noxrepo/nox
- http://groups.google.com/group/nox_dev

POX

- Stanford Univ.
- GPL v3
- python
- Pure Python version of Nox
- http://www.noxrepo.org/pox/about-pox/
- https://github.com/noxrepo/pox
- http://groups.google.com/group/pox_dev

Trema

- NEC
- GPL v2
- C and Ruby
- TremaShark: integrated network simulator/controller debugger
- Many apps(TremaApps) and tutorial
  - https://github.com/trema/apps
- http://trema.github.com/trema/
- https://github.com/trema
- https://groups.google.com/group/trema-dev

Beacon

- David Erickson of Stanford Univ.
- GPL v2 license and the Stanford University FOSS License Exception v1.0
- Java with OSGI, OpenflowJ
- Multithreaded
  - They claim that Beacon scales well
    - http://www.openflow.org/wk/index.php/Controller_Performance_Comparisons
- https://openflow.stanford.edu/display/Beacon/Home
- git://gitosis.stanford.edu/beacon.git

Floodlight

- BigSwitch
- Apache 2.0
- Java
  - Python support via Jython
- Forked from Beacon
  - Redesigned to removed OSGI dependency
    - Its own module support
- Actively defining North bound API(REST API)
  - e.g. Static flow pusher
- http://floodlight.openflowhub.org/
- https://github.com/floodlight/floodlight
- http://groups.google.com/a/openflowhub.org/group/floodlight-dev/topics

Maestro

- Rice Univ.
- LGPL v2
- Java
- Multi threaded
  - Using DAG(Directed Acyclic Graph) to exploit parallelism
- http://code.google.com/p/maestro-platform/
- http://maestro-platform.googlecode.com/svn/trunk/
  - subversion
- http://groups.google.com/group/maestro-platform

Ryu

- NTT + VALinux Systems Japan K.K.
- Apatche 2.0
- Python
- OpenStack support
- Tunneling/VLan
- For details: session: June 8th 14:00-

Full disclosure: I am a core developer of Ryu

Node Flow

- Cisco: Gary Berger(personal project?)
- MIT lincense
- Java script (with Node.js + oflib Node)
- http://garyberger.net/?p=537
- https://github.com/gaberger/NodeFLow

FlowER

- Travelping
  - Closly working with Telcom company?
- BSD-like lisence (refer the code for details)
- Erlang
- Used as a port of their products?
- https://github.com/travelping/flower

Nettle

- Yale Univ.
- BSD3
- Haskel
- http://haskell.cs.yale.edu/?page_id=376
- http://www.cs.yale.edu/publications/techreports/tr1431.pdf

Mirage

- BSD
- OCaml
- http://openmirage.org/
- https://github.com/avsm/mirage
- http://anil.recoil.org/papers/2010-hotcloud-lamp.pdf

Open vSwitch: ovs-controller

- Nicira
- Apatch 2.0(ovs-controller.c itself)
- C
- Included in Open vSwitch
- simple OpenFlow controller reference implementation

Proprietary Products (Just for completeness)

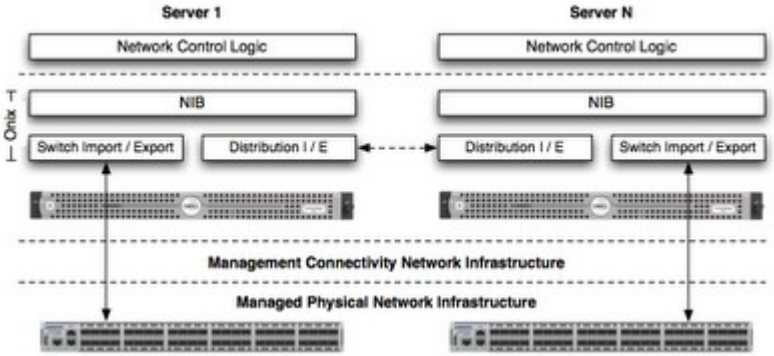
- Nicira: NVP Network Virtualization Platform
- BigSwitch: Floodlight based?
- NEC: ProgrammableFlow
- Midokura: Midonet
- NTT Data:
- Travelping: FlowER based?

Ask your vendors for details

Related Academic research

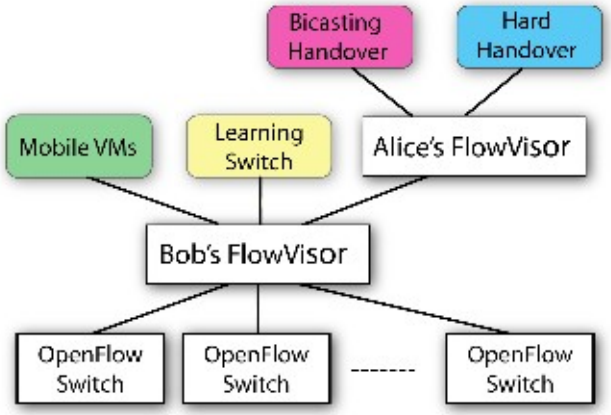
Onix

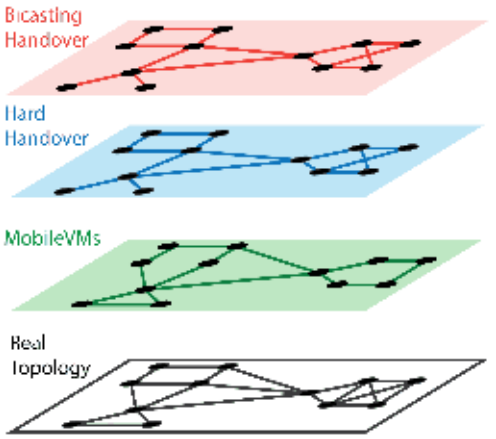
- Teemu Koponen, Martin Casado, Natasha Gude, and Jeremy Stribling, Nicira Networks; Leon Poutievski, Min Zhu, and Rajiv Ramanathan, Google; Yuichiro Iwata, Hiroaki Inoue, and Takayuki Hama, NEC; Scott Shenker, International Computer Science Institute (ICSI) and UC Berkeley
- No codes publicly available
- http://static.usenix.org/event/osdi10/tech/#wed
- http://static.usenix.org/events/osdi10/tech/full_papers/Kopone
- Network Operating System
- Network Information Base(NIB)

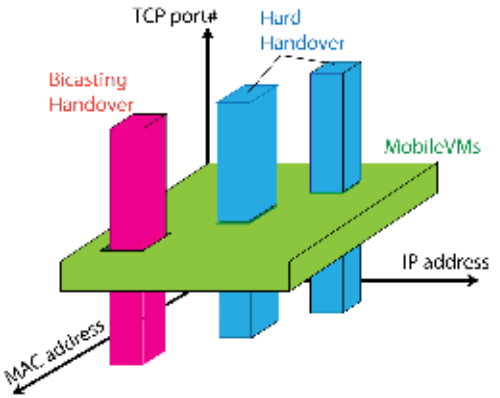


flowvisor

- Its own license (refer the repo for details)
- Java
- OF virtualization/network slicing
- http://www.openflow.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf
- https://bitbucket.org/onlab/flowvisor



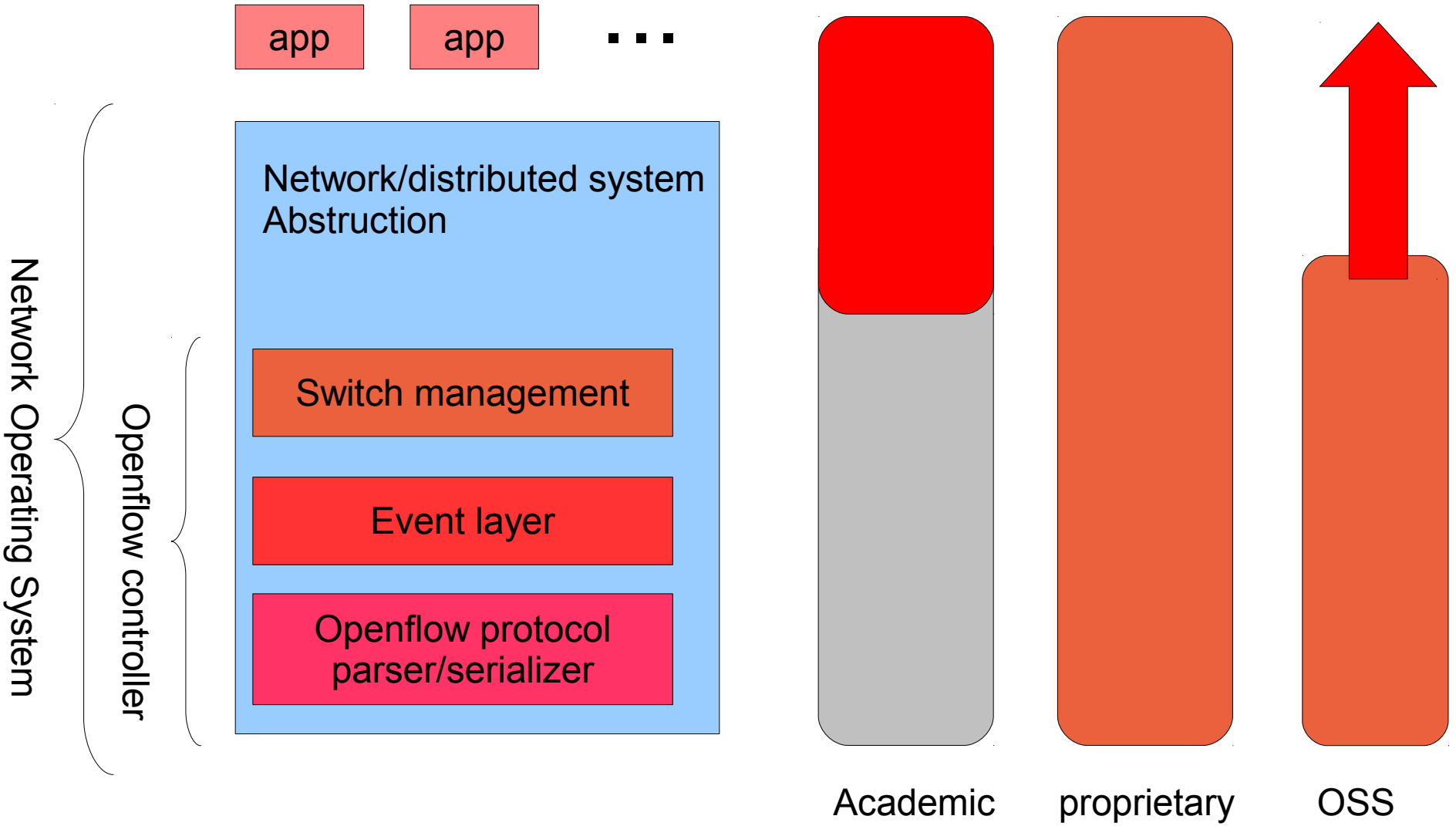




Other researches

- RouteFlow
  - https://sites.google.com/site/routeflow/
- Flowscale
  - Load balancer
  - http://www.openflowhub.org/display/FlowScale/FlowScale+Home
- Frenetic: model checker
  - http://frenetic-lang.org/
- NICE-OF
  - Symbolic Execution with Model checker
  - https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final105.pdf
  - http://code.google.com/p/nice-of/

Openflow controller to network operating system


app
app
. . .
Network/distributed system
Abstruction
Switch management
Event layer
Openflow protocol
parser/serializer
Network Operating System
Openflow controller
Academic
proprietary
OSS

Openflow controller to Network OS

- Distributed programming is hard
  - State distribution
- Event changing the state is hard
  - react chage on network configuration and chage the switch configuration
  - Calculating the switch diff based on network diff is hard
- Configuring network right is hard
  - Verification?
  - Model checker?
- Provide some layer for distributed programming
  - Higher level network view
  - Debugging environment?
    - View network status by single command
    - Network health check: Take network states snapshot, and run verification on it
      - Or runtime check?
  - Distributede database?
  - Switch model
    - tracking switch flows somehow
  - HA, multi controllers
    - Taking over switch
  - Simulator?

Summary

- OSS Openflow controllers are very common
- The next area to investigate is to evolve from openflow controller to network operating system

Thank you

- Questions?