ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δίκτυα Καθοριζόμενα από Λογισμικό

Ενότητα 1.4: Controllers

Ξενοφώντας Δημητρόπουλος
Τμήμα Επιστήμης Υπολογιστών

HY436: Controllers

Xenofontas Dimitropoulos
13/10/2014

Credits: Bernhard Ager (ETH Zurich) for some of the slides

Agenda

- From Internet to SDN abstractions
- SDN controllers:
  - NOX
  - POX
  - Ruy
  - Floodlight
  - ONIX
  - OpenDayLight

Abstractions

- How to build a complex system?

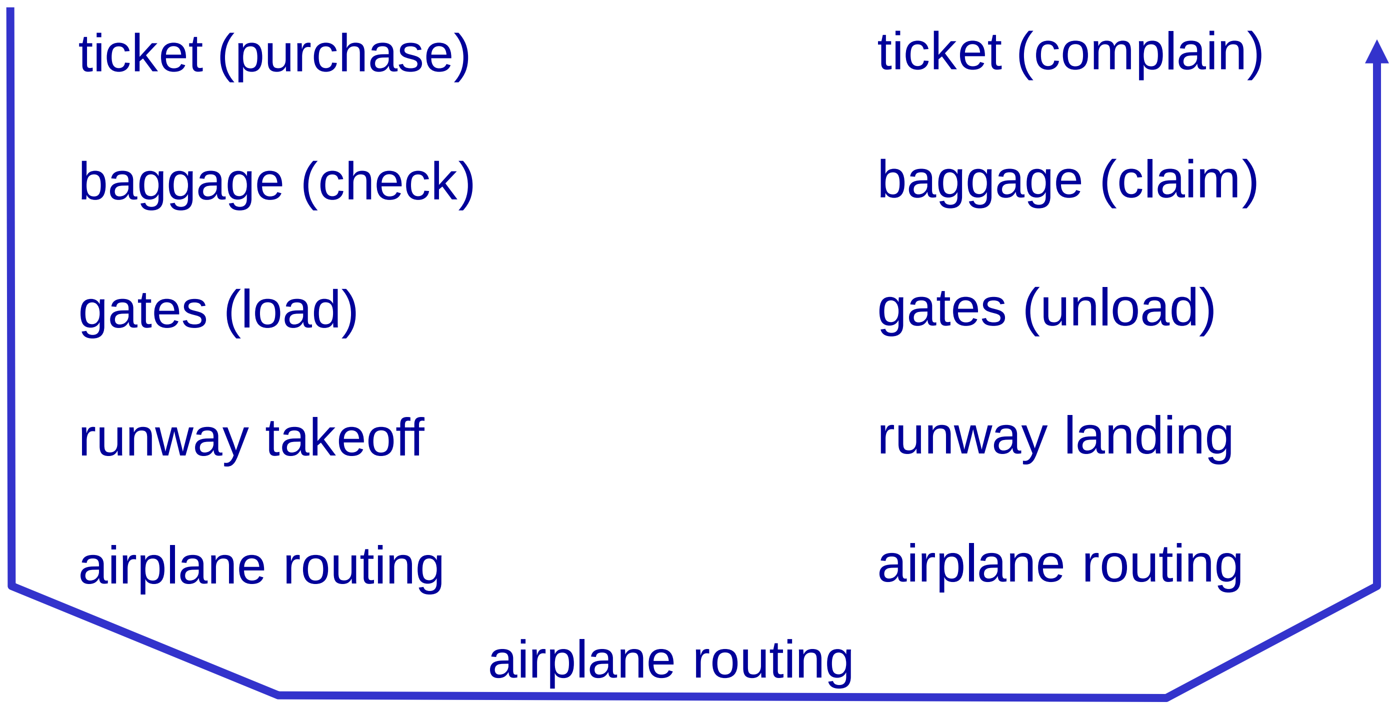
  General principle: break it down into tractable pieces

- Abstractions:
  - Identify re-usable components
  - Hide unnecessary details

  Think of objects in object-oriented programming

- Abstractions are often organized in layers

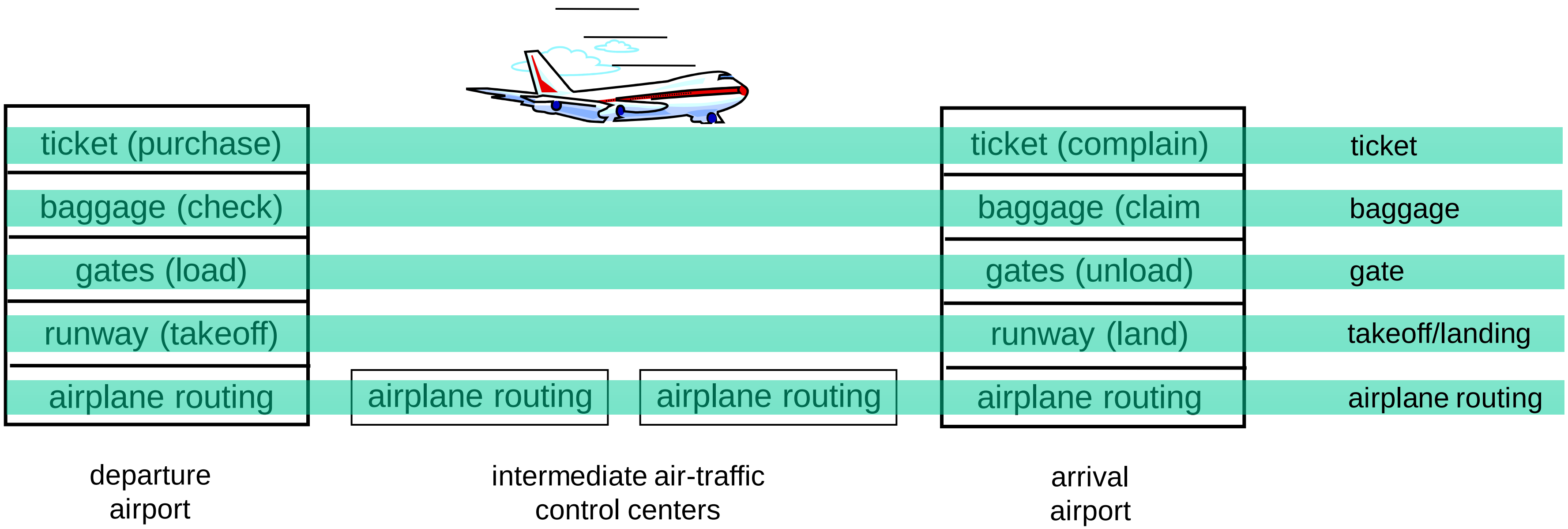
Complex system example: air travel!



- a series of steps (will map them to layered abstractions)

Modified from: Kurose, James F., and Keith W. Ross. Computer Networking: A top-down approach featuring the Internet.

Layered abstractions of the system



layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

Modified from: Kurose, James F., and Keith W. Ross. Computer Networking: A top-down approach featuring the Internet.

Why (layered) abstractions?

dealing with complex systems:

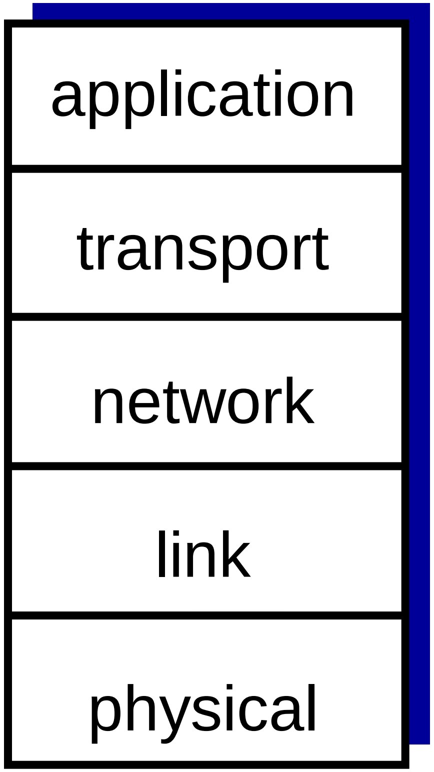
- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for **understanding**

- modularization eases **managing** and **updating the system**
  - change of implementation of layer's service transparent to rest of system

Modified from: Kurose, James F., and Keith W. Ross. Computer Networking: A top-down approach featuring the Internet.

Today's Internet abstractions

- *application:* supporting network applications
- *transport:* process-process data transfer
- *network:* routing of datagrams from source to destination
- *link:* data transfer between neighboring network elements
- *physical:* bits "on the wire"



Modified from: Kurose, James F., and Keith W. Ross. Computer Networking: A top-down approach featuring the Internet.

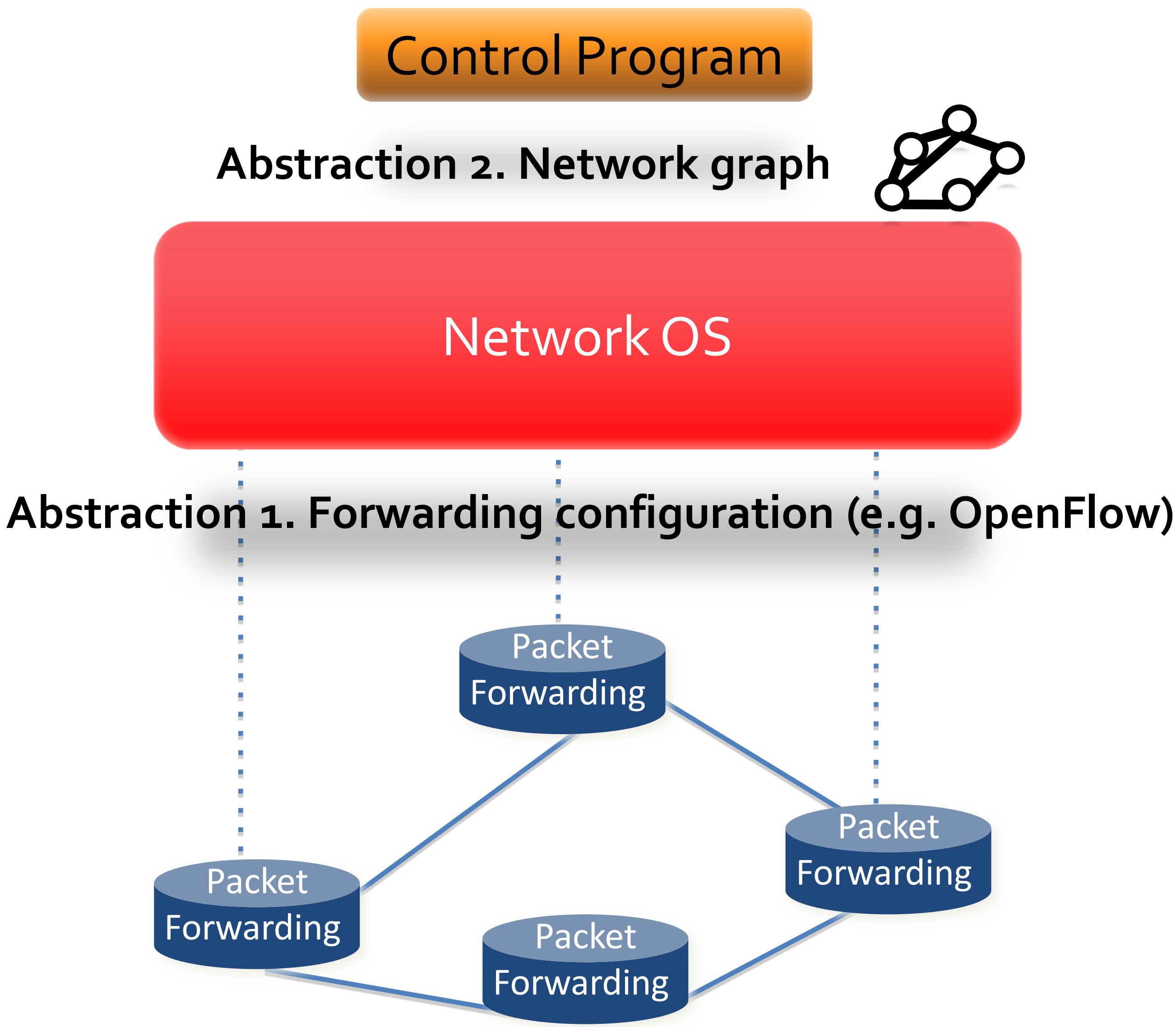
Control plane vs data plane

- Data plane:
  - Forward packets (milisecond granularity)
  - Based on 5-layer reference model
- Control plane:
  - Monitor and configure forwarding elements (seconds to hours if manual)
  - Presently: no abstractions!

Deriving control plane abstractions

- What does it take to control a network?
  - Learn network state (topology, etc.)
  - Decide how to configure it (routing, isolation, traffic engineering, etc.)
  - Push configuration to network elements
- Which processes are re-usable?
  - Build network topology
  - Push configuration to network elements

SDN: 2+1 abstractions



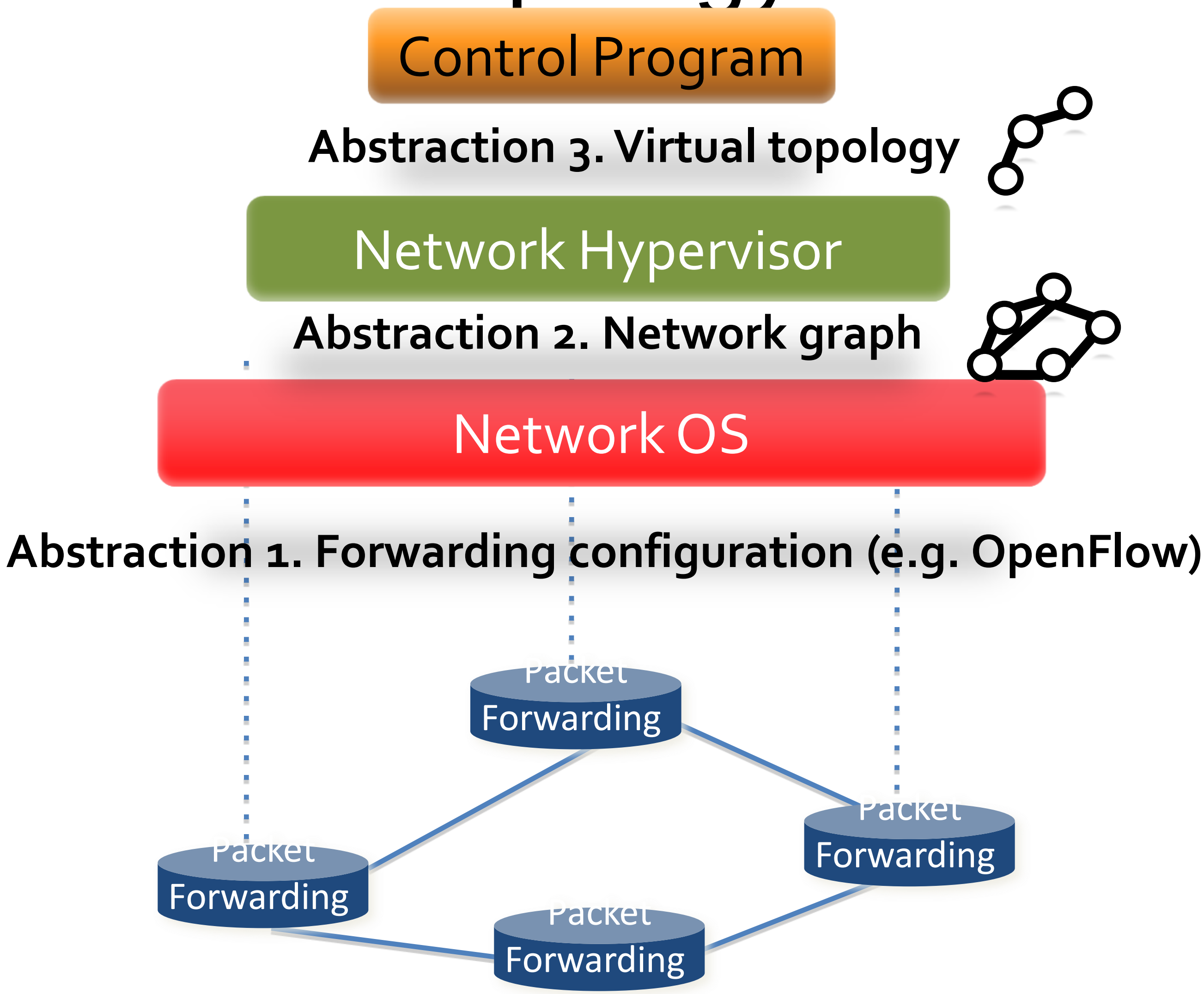
Clean Separation of Concerns

- **Control program**: expresses operator goals
  - Implemented on global network view abstraction
  - Computes forwarding state for each router/switch
- **NOS**: links global view and physical switches
  - Gathers information for global network view
  - Conveys configurations from control program to swtiches
- **Routers/switches**: merely follow orders from NOS

Enables independent innovation in "layers"

Based on S. Shenker's talk "SDN at the Crossroads" Stanford Seminar, 2013 See: http://youtu.be/WabdXYzCAOU

SDN's 3rd abstraction: Virtual topology



To be discussed in a future lecture

Today: Network OS

- Maintain an up-to-date view of the network state (topology, etc.)
- Configure network elements (OpenFlow, etc.)

  ➔Also known as “south-bound” interface
- Provide a graph abstraction to the applications on top

  ➔Also known as “north-bound” interface

Challenges in building a NOS

- Scalability to large networks
- Reliability to failures
- Good performance (fast, etc.)
- Generality and simplicity of "north-bound" API

➔ Largely still an open question

Agenda

- From Internet to SDN abstractions
- SDN controllers:
  - NOX
  - POX
  - Ruy
  - Floodlight
  - ONIX
  - OpenDayLight

Controllers Overview

Many controllers!

- NOX
- POX
- Ryu
- Floodlight
- OpenDayLight
- Trema
- OpenFaucet
- Beacon

- Focused on north-bound API:
  - Pyretic
  - Frenetic
  - Procera
  - …
- Focused on inter-domain routing
  - RouteFlow
  - ONOS
  - …

Controllers' diversification

- Programming language (can affect performance)
- Focus:
  - Southbound API
  - Northbound API
  - Support for OpenStack:
    - Widely-used cloud operating system
    - Manage storage, computation, and network in the cloud
    - Allows to write cloud applications
  - Education/Research, Production?
- Learning curve
- User base and community support

Based on N. Feasmter's lecture "Examples of SDN Controllers"2014 See:http://youtu.be/dpcw2XqLp-E

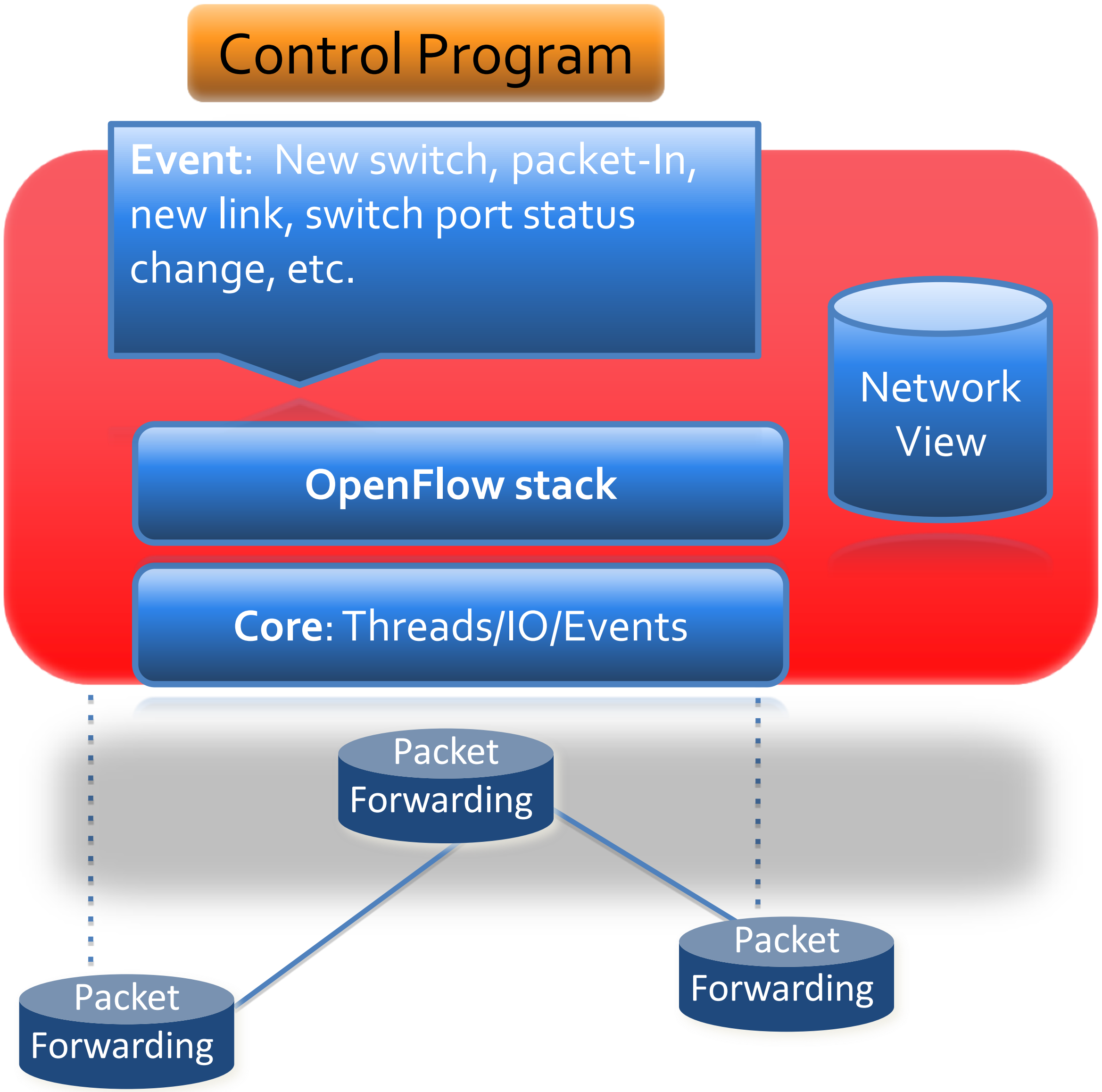
NOX

http://www.noxrepo.org/nox/about-nox/

- First-generation OpenFlow controller
  - Open source and widely used
  - Fast IO, well maintained
  - C++, OpenFlow 1.0
- Programming model
  - Controller registers for events
  - Programmer writes event handlers

Natasha Gude, Teemu Koponen et al, SIGCOMM CCR, 2008.
Talk by Martin Casado, „A Network Operating System for OpenFlow", SDN Workshop 2009

NOX: Event-based model



Network view = Graph Abstraction

- In network view:
  - Switch-level topology
  - Locations of hosts, middleboxes, other network elements
  - Locations of users
  - Namespace: bindings between names and addresses
- Not in network view: state of network traffic
- Example use case: policy-based access control

NOX Characteristics

- Performance ☑
- Generality ☑
- Robustness ☐?
- Simplicity ☒

Based on Martin Casado's retrospective in the talk:
"A Network Operating System for OpenFlow", 2009
http://archive.openflow.org/downloads/Workshop2009/OpenFlowWorkshop-MartinCasado.pdf

POX

http://www.noxrepo.org/pox/about-pox/

- NOX in Python
  - Supports OpenFlow 1.0 only
- Advantages:
  - Widely used, maintained, supported
  - Relatively easy to read and write code
- Disadvantages: Performance

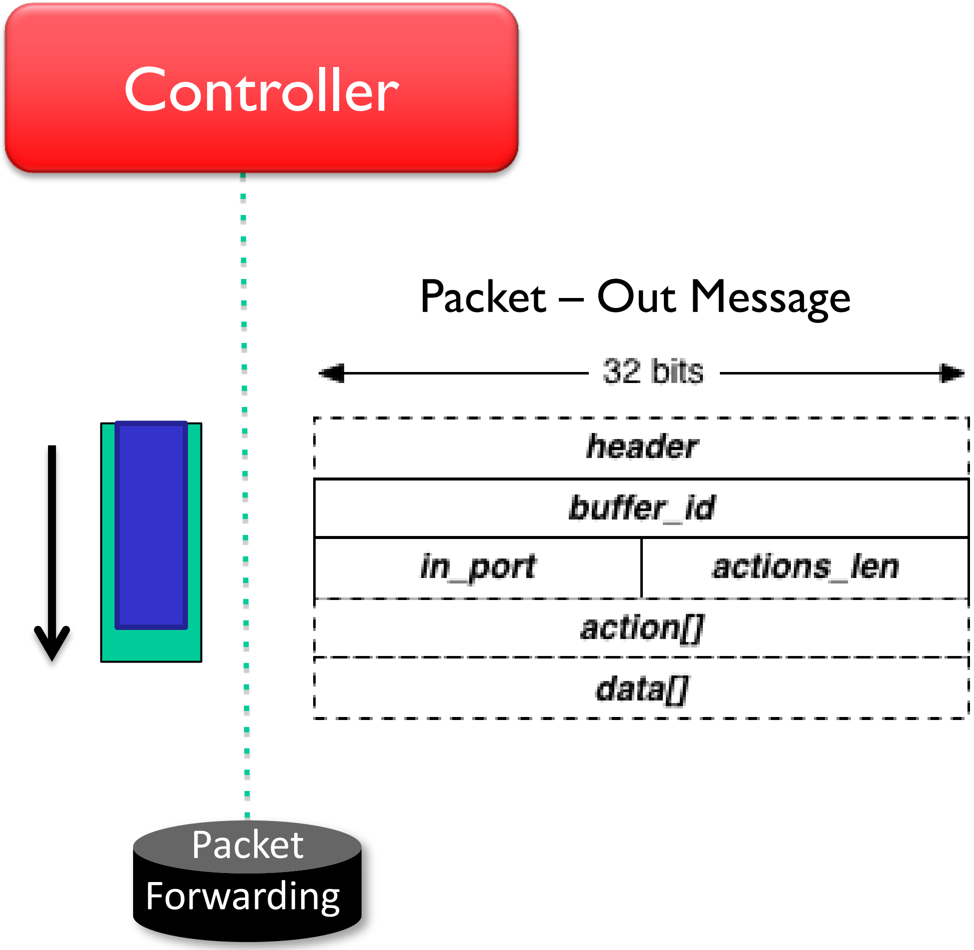
https://openflow.stanford.edu/display/ONL/POX+Wiki

POX programming example

```
def _handle_PacketIn (self, event):

      packet = event.parsed # This is the parsed pkt data
      packet_in = event.ofp  # The ofp_packet_in message

      msg = of.ofp_packet_out()
      msg.buffer_id = packet_in.buffer_id
      msg.in_port = packet_in.in_port
      msg.match = of.ofp_match.from_packet(packet)
      action = of.ofp_action_output(port = of.OFPP_FLOOD)
      msg.actions.append(action)

      self.connection.send(msg)
```

Reminder: Packet-Out

- Instruct switch to send a packet out
  - Response to Packet-In
  - New packet
- Must contain a full packet or reference a buffered packet ID
- May include a list of actions to be applied



See OpenFlow message structures in http://flowgrammable.org/sdn/openflow/message-layer/packetout/

Ryu

- Open source Python controller
  - Supports OpenFlow 1.0-1.4, Netconf, etc.
  - Works with OpenStack
- Advantages:
  - OpenStrack integration, OpenFlow 1.2-1.4
- Disadvantages: Performance

Floodlight

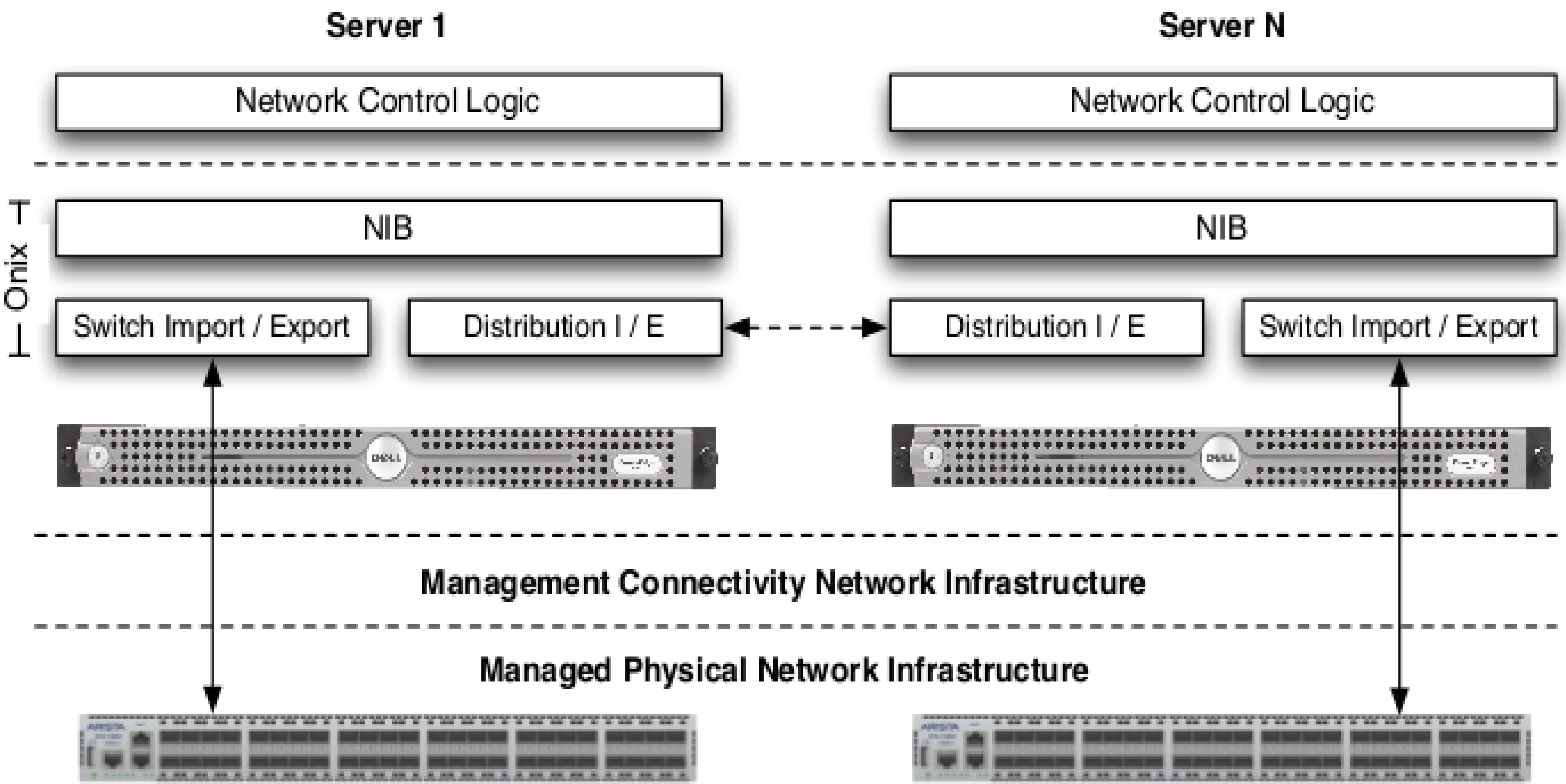
- Open-source Java controller
  - Supports OpenFlow v1.0
  - Fork from Beacon Java OpenFlow controller
  - Maintained by Big Switch Networks
- Advantages:
  - Good documentation
  - Integration with REST API
  - Production-level performance, OpenStack
- Disadvantages: Steep learning curve

ONIX

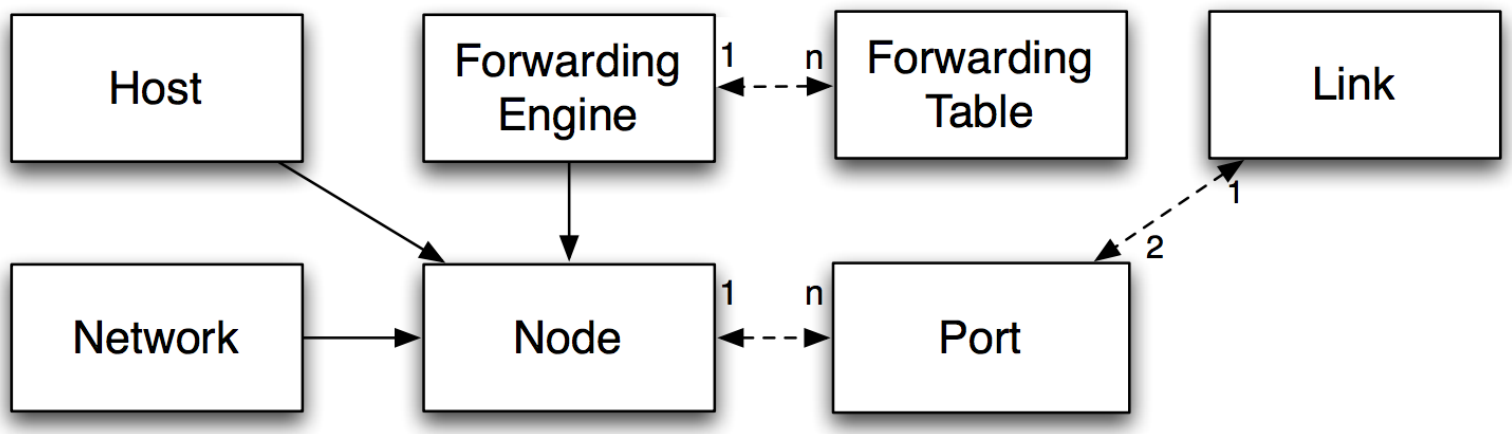
Onix: A Distributed Control Platform for Large-scale Production Networks. T. Koponen , et al. OSDI 2010

- Closed source, not publicly available
- Production quality (likely used in Google backbone)
- Distributed by design
  - Scalability
  - Robustness

ONIX components



Network Information Base



NIB: Network Information Base = Graph++ Abstraction

Queries to the NIB

Category	Purpose
Query	Find entities
Create, destroy	Create and remove entities
Access attributes	Inspect and modify entities
Notifications	Receive updates about changes
Synchronize	Wait for updates being exported to network elements and controllers
Configuration	Configure how state is imported to and exported from the NIB
Pull	Ask for entities to be imported on-demand

Scalability

- Partition
  - An instance keeps only a subset of the NIB
- Aggregation
  - The details of NIB subsets in other controller instances are hidden
- Consistency between instances

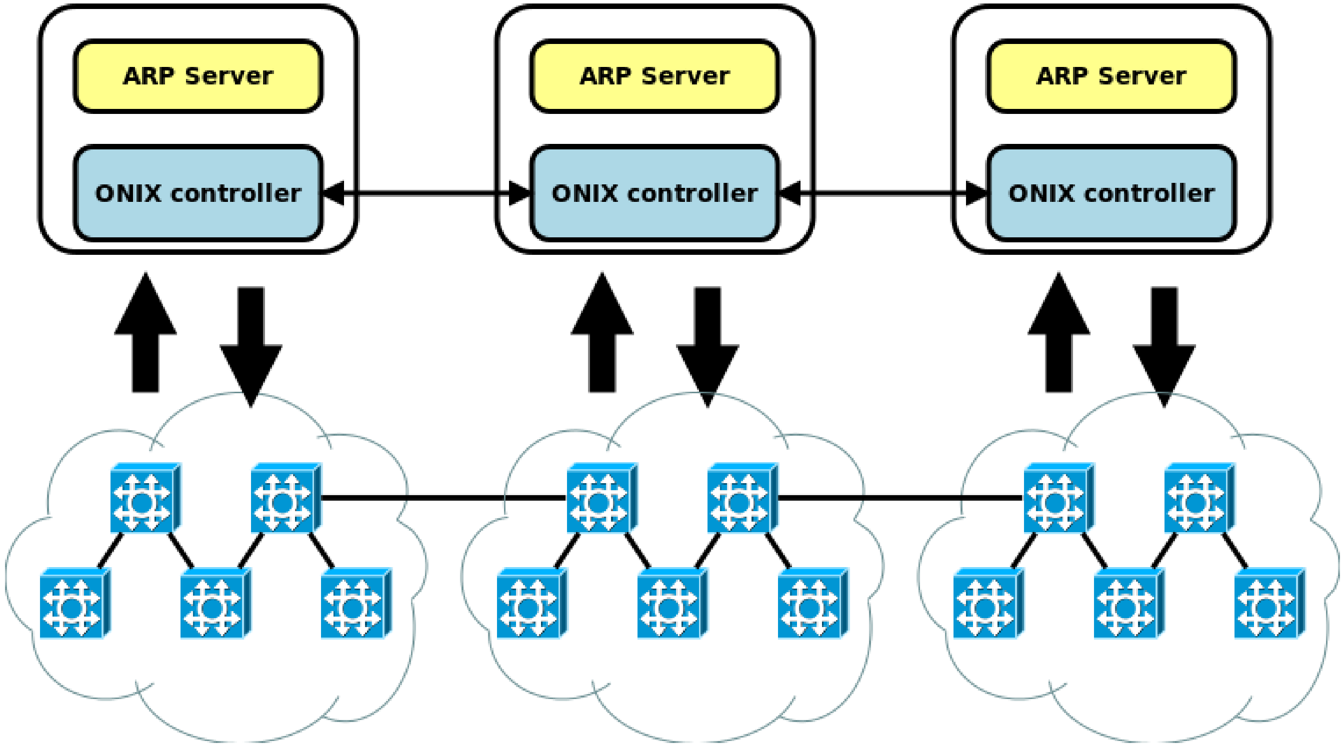
Reliability to Failures

- Network element and link failures
  - Application’s responsibility
- ONIX failures
  - Running instances detect and take over
  - More than one can manage simultaneously
- Connectivity to controller failures

Distributing the NIB

- Different storage options
  - Transactional data storage (SQL)
  - Distributed hash table (DHT)
- Applications determine consistency requirements
  - Strong consistency for critical, stable state
  - Eventual consistency for dynamic, inconsistency-tolerant state

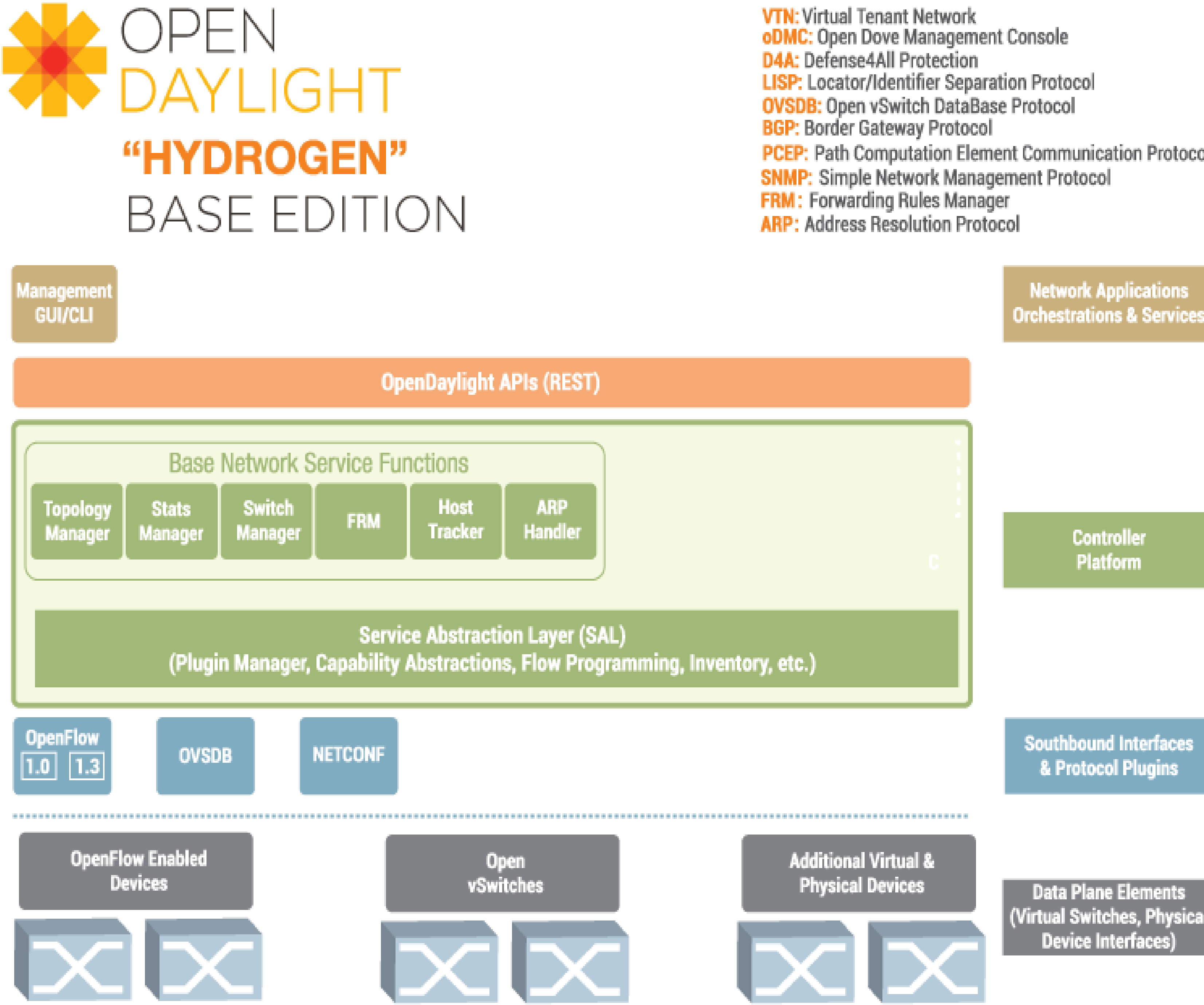
Example: ARP server



- Switch topology: candidate for hard state
- IP-MAC mappings: candidate for soft state
- Free choice: number of controllers for every switch
- Free choice: local ARP tables or single global ARP table

OpenDayLight Controller

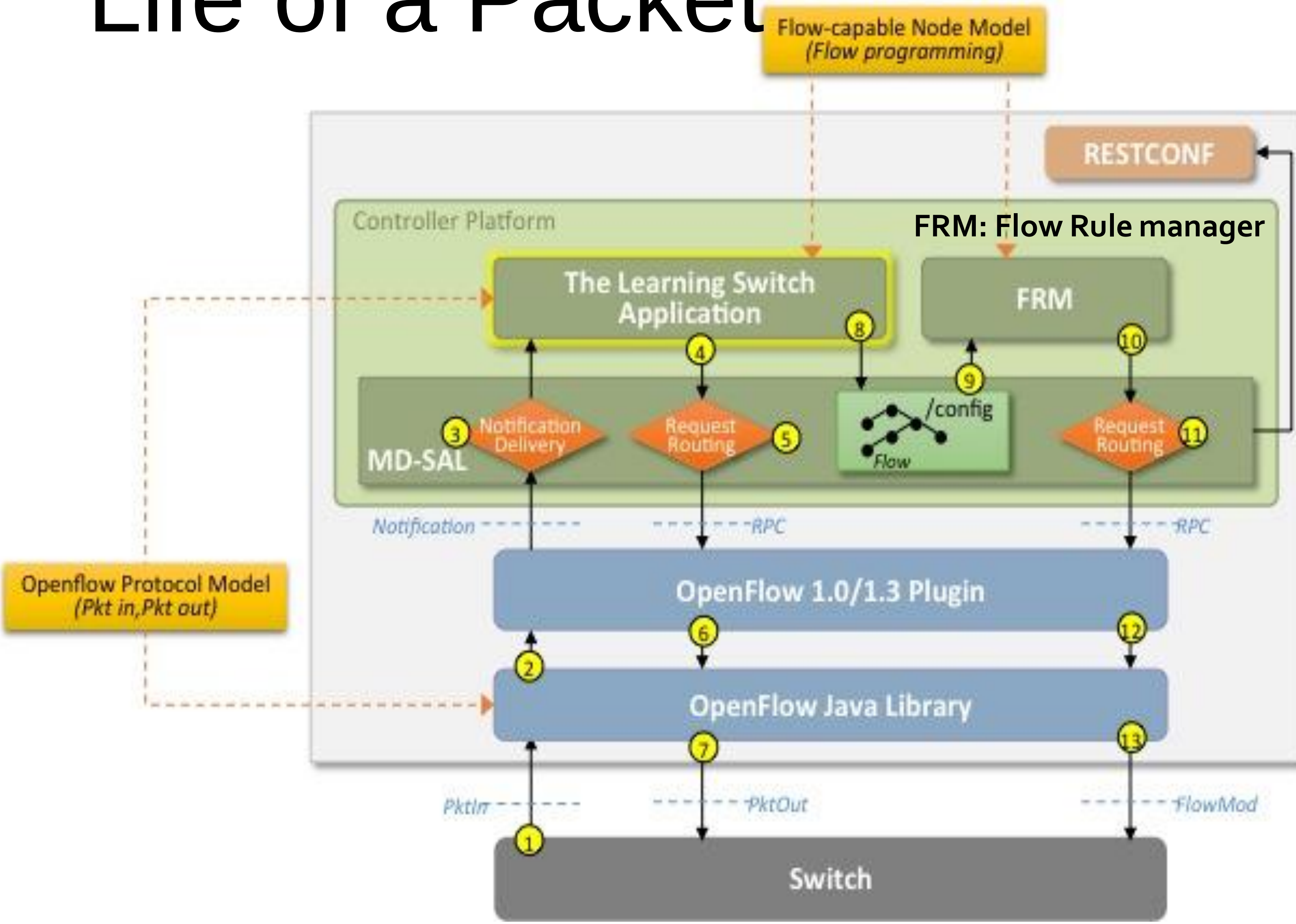
- Heavy industry involvement and backing
- A platform for SDN and Network Function Virtualization (NFV) innovation
  - Not limited to OpenFlow innovations


OPEN
DAYLIGHT
"HYDROGEN"
BASE EDITION
VTN: Virtual Tenant Network
oDMC: Open Dove Management Console
D4A: Defense4All Protection
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase Protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Communication Protocol
SNMP: Simple Network Management Protocol
FRM: Forwarding Rules Manager
ARP: Address Resolution Protocol
Management GUI/CLI
Network Applications Orchestrations & Services
OpenDaylight APIs (REST)
Base Network Service Functions
Topology Manager
Stats Manager
Switch Manager
FRM
Host Tracker
ARP Handler
Controller Platform
Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)
OpenFlow
1.0
1.3
OVSDB
NETCONF
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Design Choices

- Controller:
  - Java chosen as an enterprise-grade cross platform compatible language
  - Java interfaces are used for event listening, specifications and forming patterns
- Installation and dynamic bundle loading
  - Maven build system for Java
  - OSGi :
    - Allows dynamically loading bundles
    - Allows registering dependencies and services exported
    - For exchanging information across bundles

Life of a Packet



Source : OpenDayLight Wiki

OpenDayLight network abstraction

- MD-SAL
  - Model-Driven Service Abstraction Layer
  - Performs event routing
- State: Hierarchical database in MD-SAL
  - Contains all kinds of information
    - Forwarding rules
    - Topology, etc.
  - Plugins can register for change notification events,

    e.g., the FRM registers to flow forwarding rules

OpenDayLight Summary

- OpenDayLight is an industry-backed effort to develop a broader set of SDN solutions
- SDN is no longer just OpenFlow
  - Possible to integrate a broader set of cloud based applications
  - Set of functions is similar to other controllers
- Steep learning curve

See Nick Feamster's Lecture on OpenDayLight
http://youtu.be/ySocIFY_aKk?list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY

Agenda

- From Internet to SDN abstractions
- SDN controllers:
  - NOX
  - POX
  - Ruy
  - Floodlight
  - ONIX
  - OpenDayLight

Further Reading

- Martin Casado's list of OpenFlow Software Projects
  http://yuba.stanford.edu/~casado/of-sw.html

- Nick Feamster's SDN Controllers' lectures
  http://youtu.be/dpcw2XqLp-E

  http://youtu.be/yS0cIFY_aKk?list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Ξενοφώντας Δημητρόπουλος. «**Δίκτυα Καθοριζόμενα από Λογισμικό. 1.4 Controllers**». Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: http://www.csd.uoc.gr/~hy436/