Software-defined Networking
Session IV

Block Course – Winter 2016/17

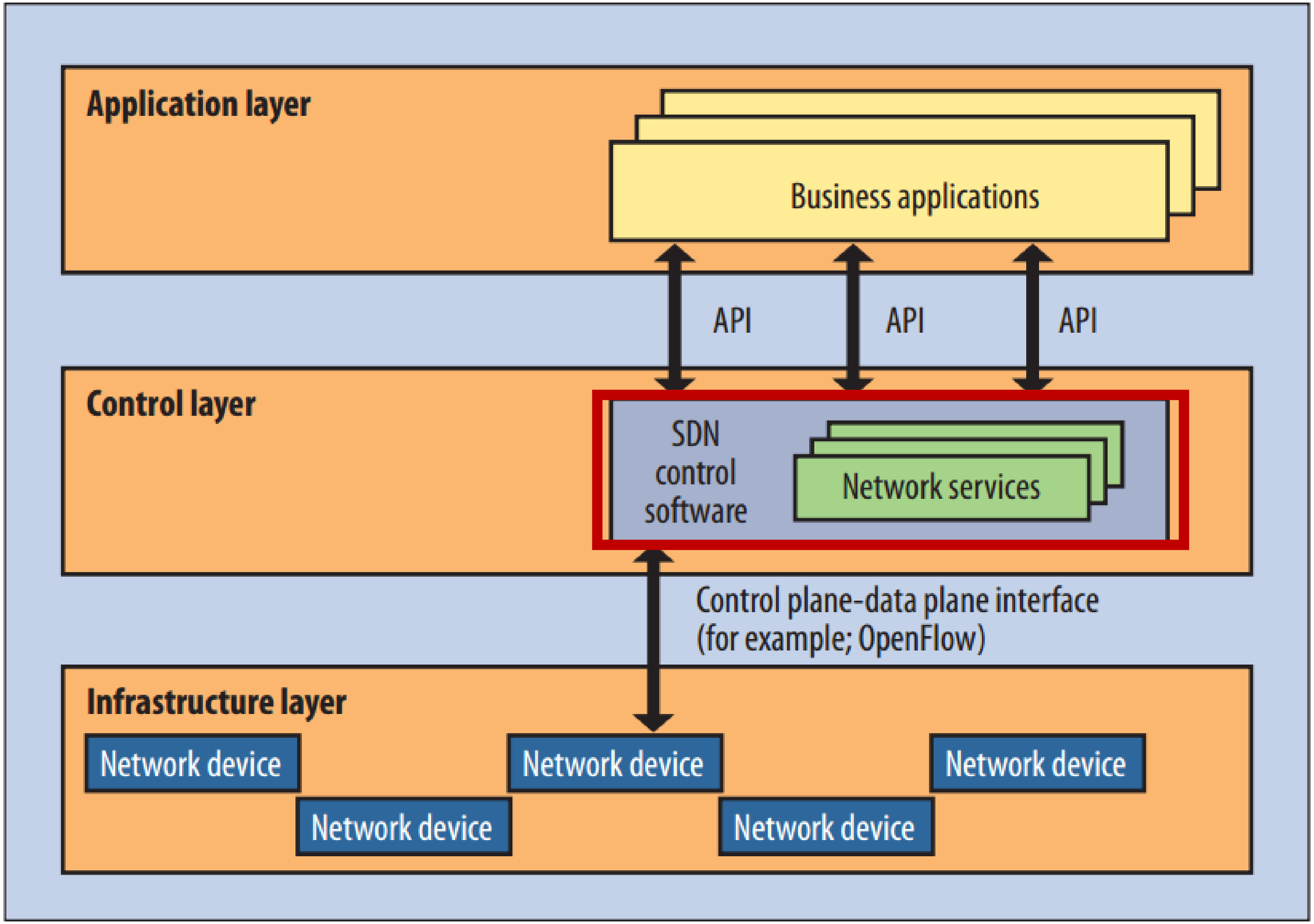
David Koll

Notes

Exercise 5a:

- JAVA Dependency Error (jre-6-headless missing, etc.): java apt-get –f install

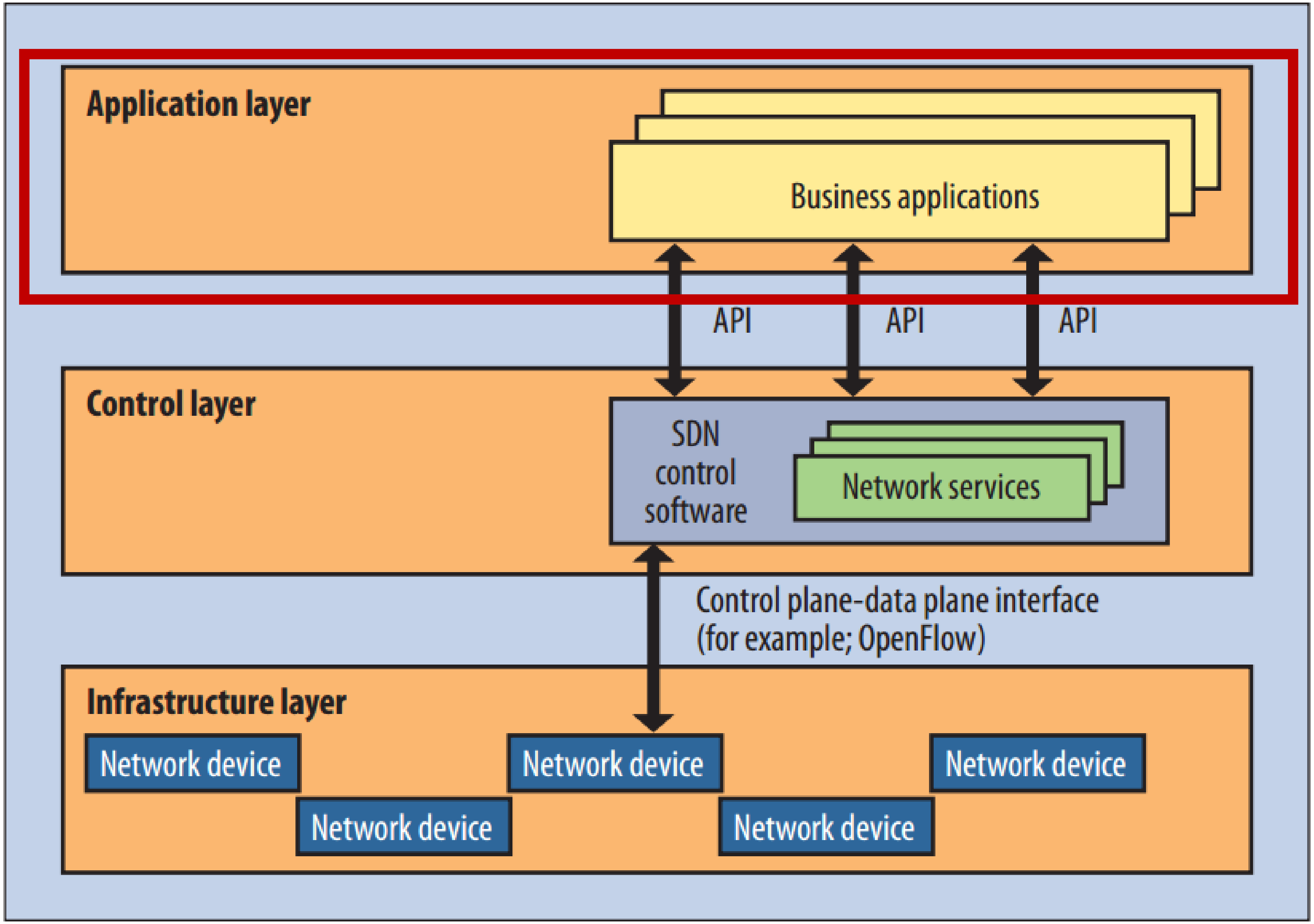
This Lecture



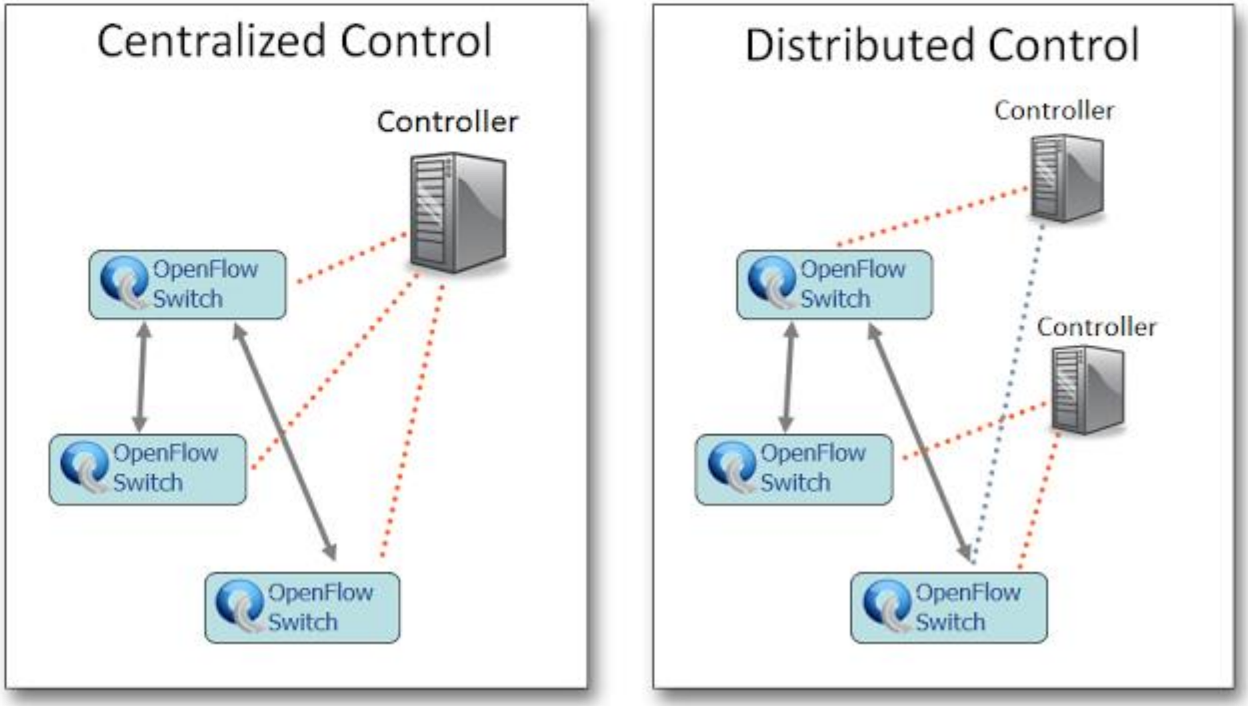
SDN Controllers

- SDN promises to facilitate network management and ease the burden of solving networking problems
- Main means: the logically-centralized control offered by a network controller (or network operating system (NOS))
- Crucial value of a controller is to provide abstractions, essential services, and common application programming interfaces (APIs) to developers.

Controller interact both northbound and southbound!



SDN Controllers



Centralized Controllers

- Single entity that manages all forwarding devices of the network.

- Single point of failure and may have scaling limitations.
  - May not be enough to manage a network with a large number of data plane elements.

Centralized Controllers - Examples

- NOX/POX, Beacon, Floodlight, …
- NOX-MT, Beacon and Floodlight: designed as highly concurrent systems
  - Goal: achieve throughput required by enterprises and data centers.
  - Beacon can deal with more than 12 million flows per second

- Other centralized controllers such as Trema or Ryu target specific environments (e.g., carrier networks)

Distributed Controllers

- A distributed controller can be a centralized cluster of nodes…
  - high throughput for very dense data centers

- …or a physically distributed set of elements
  - more resilient to different kinds of logical and physical failures.

- Multiple data centers interconnected by a wide area network
  - Hybrid approach: clusters of controllers inside each data center and distributed controller nodes in the different sites

Distributed Controllers

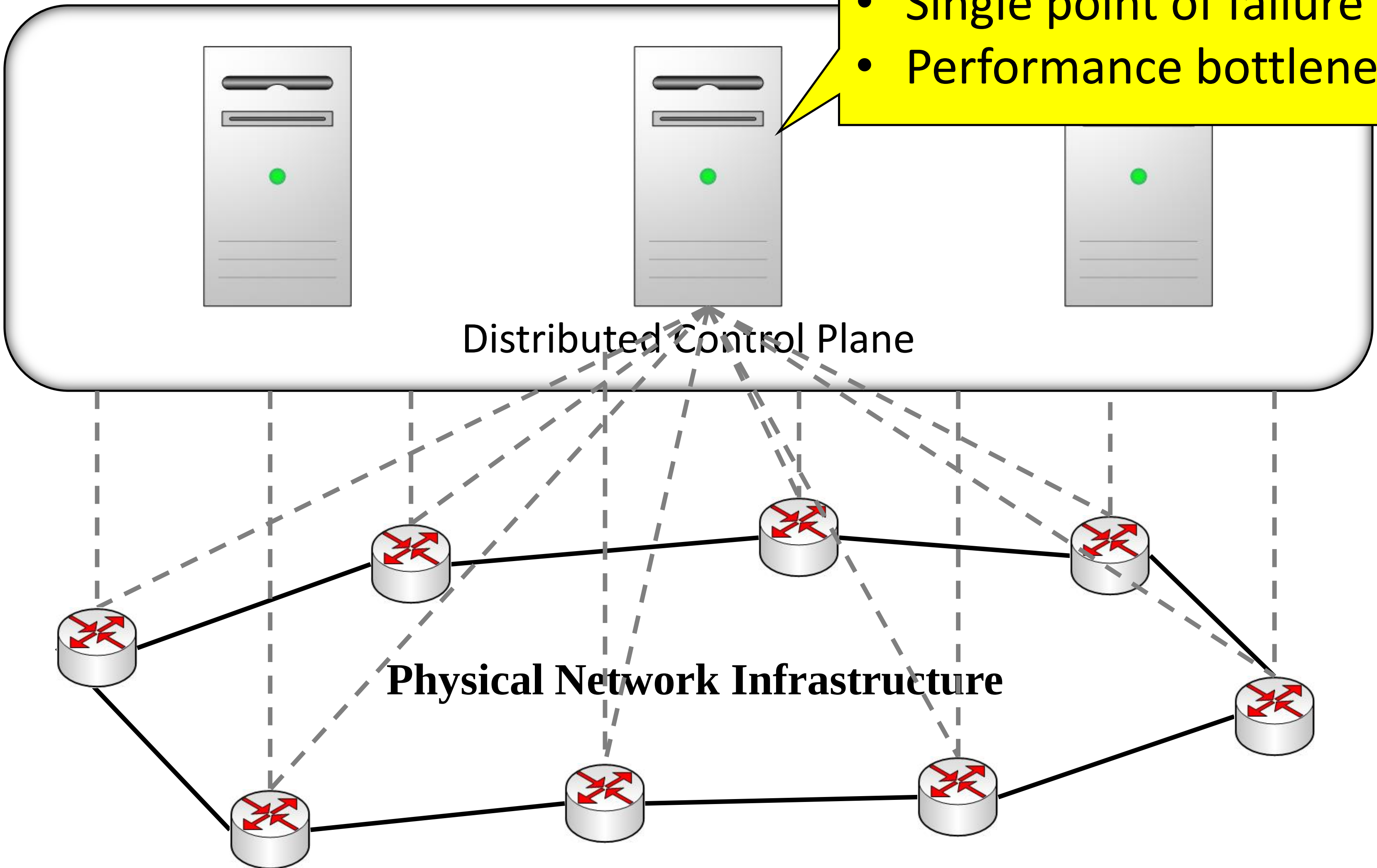
- Consistency semantics: weak or strong
  - Weak: data updates on distinct nodes will eventually be updated on all controller nodes.
    - implies that there is a period of time in which distinct nodes may read different values (old value or new value) for the same property.
  - Strong: all controller nodes will read the most updated property value after a write operation.
    - Impact on system performance
    - Offers a simpler interface to application developers.
- Failure recovery

Distributed Controllers - Examples

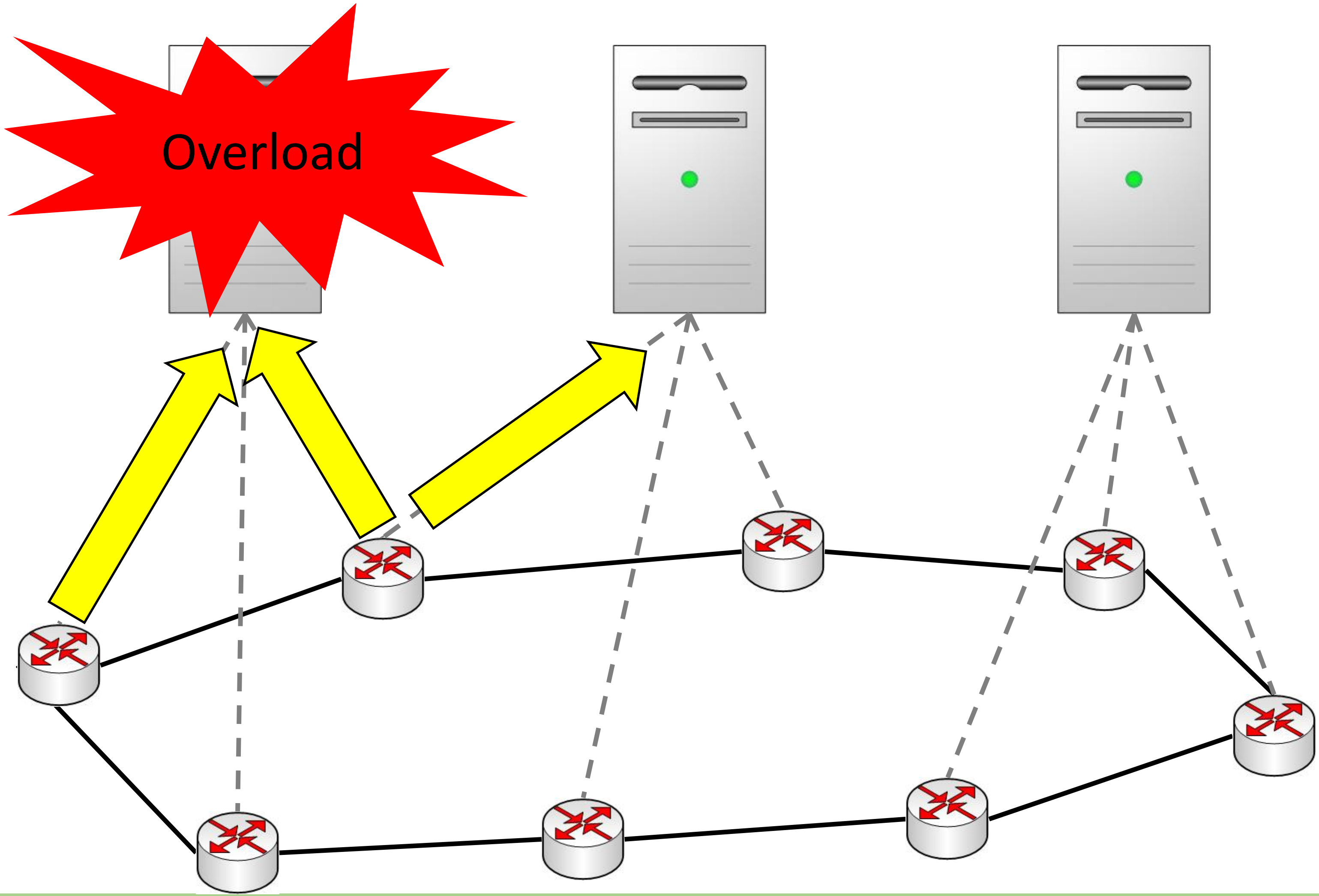
- Onix, HyperFlow, HP VAN SDN, ONOS, DISCO, Fleet…
  - Most offer weak consistency semantics
  - Only Onix and ONOS provide (close to) strong consistency
  - Some controllers tolerate crash failures
  - But: Controllers do not tolerate arbitrary failures
    - Any node with an abnormal behavior will not be replaced by a potentially well behaved one

Distributed Controllers - Operation

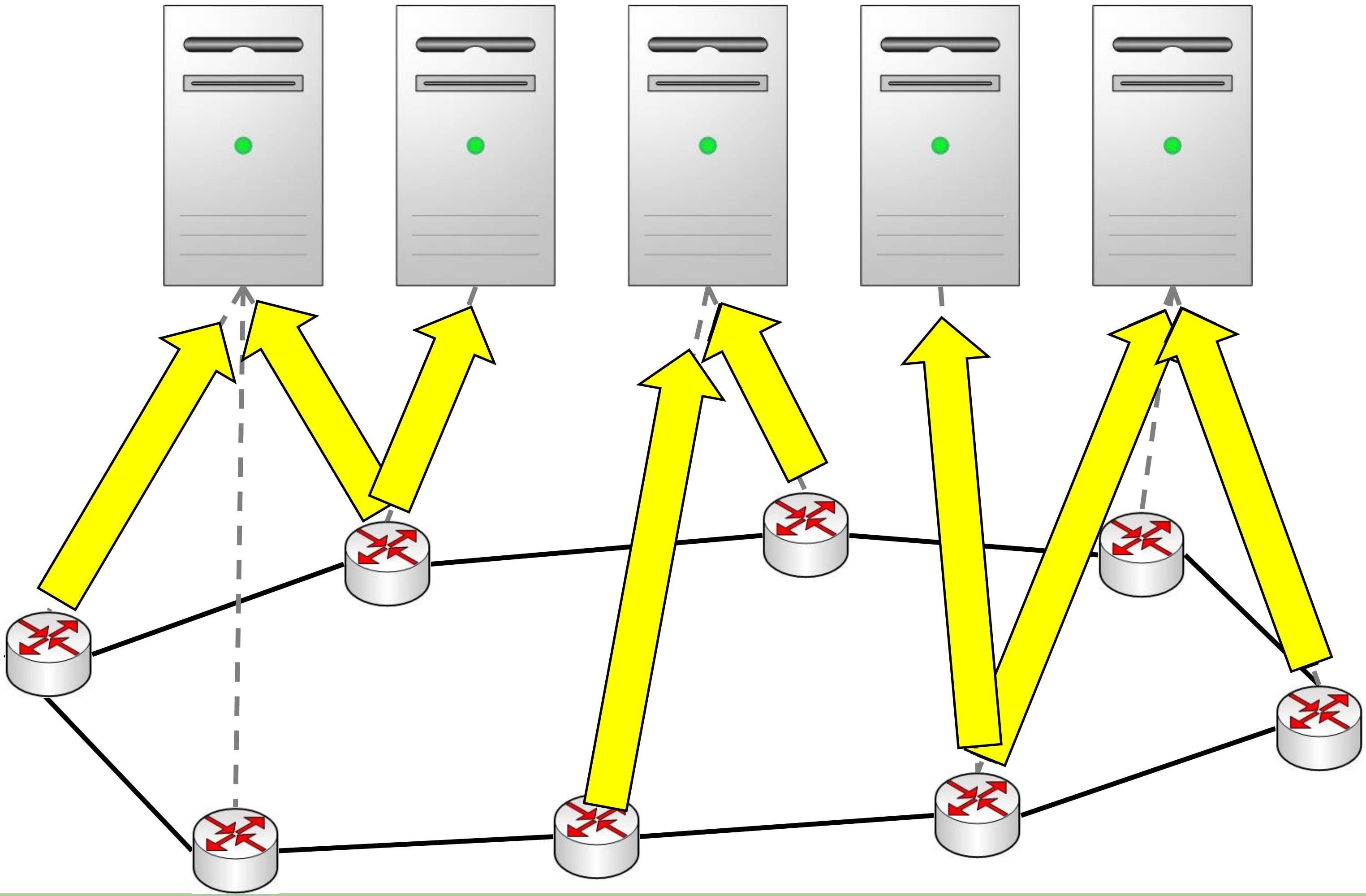
- Single point of failure
- Performance bottleneck



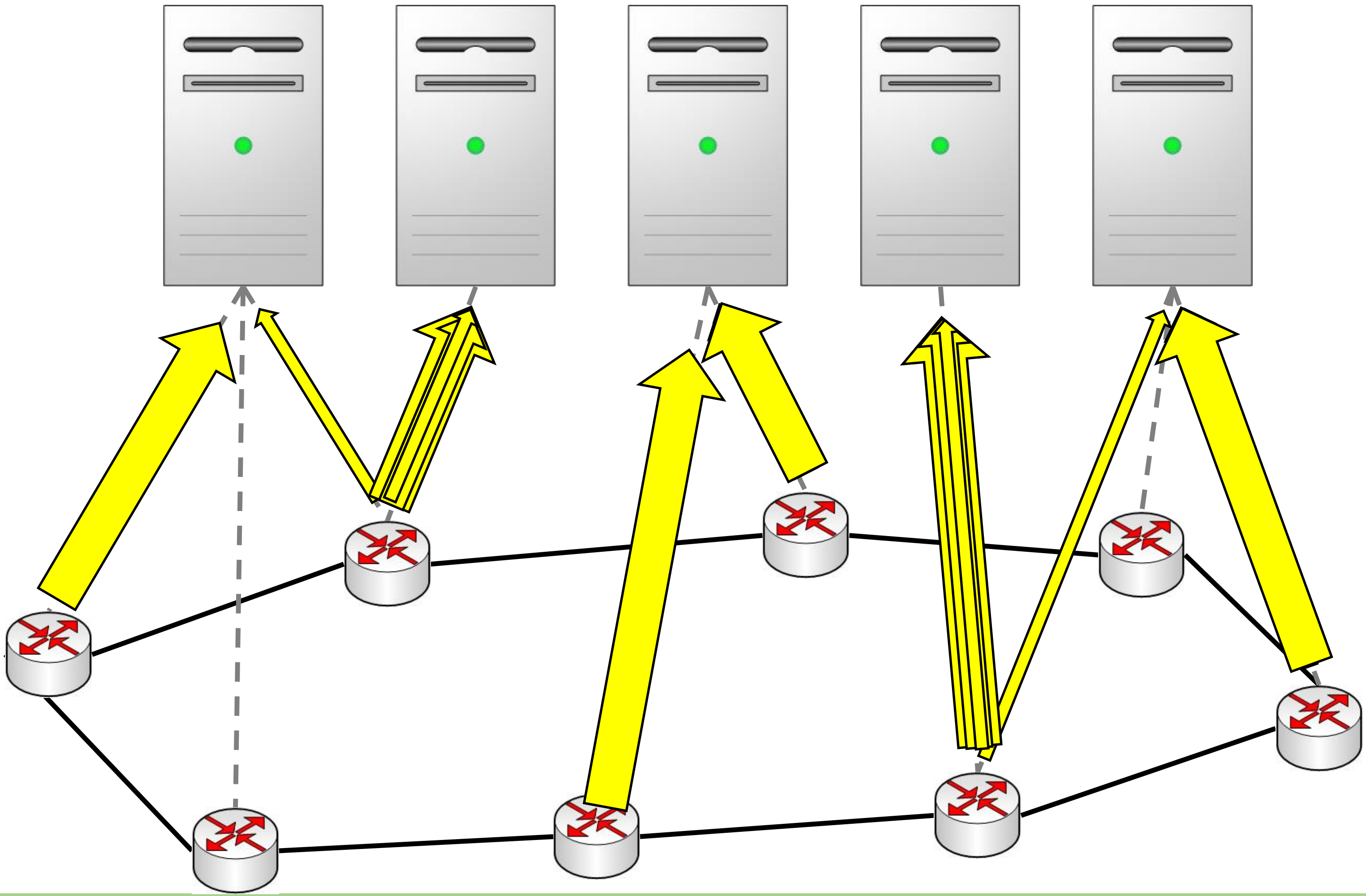
Spatial Partitioning



Growing the Control Plane



Shrinking the Control Plane



Goals

- Build a distributed control plane which
  - Load balances
  - Grows
  - Shrinks
- Requires
  - Load estimation at controllers
  - Switch migration protocol
  - Consistency protocols

Controller Implementations

SDN Controllers

TABLE VI
CONTROLLERS CLASSIFICATION

Name	Architecture	Northbound API	Consistency	Faults	License	Prog. language	Version
Beacon [186]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	Java	v1.0
DISCO [185]	distributed	REST	—	yes	—	Java	v1.1
ElastiCon [229]	distributed	RESTful API	yes	no	—	Java	v1.0
Fleet [200]	distributed	ad-hoc	no	no	—	—	v1.0
Floodlight [189]	centralized multi-threaded	RESTful API	no	no	Apache	Java	v1.1
HP VAN SDN [184]	distributed	RESTful API	weak	yes	—	Java	v1.0
HyperFlow [195]	distributed	—	weak	yes	—	C++	v1.0
Kandoo [230]	hierarchically distributed	—	no	no	—	C, C++, Python	v1.0
Onix [7]	distributed	NVP NBAPI	weak, strong	yes	commercial	Python, C	v1.0
Maestro [188]	centralized multi-threaded	ad-hoc API	no	no	LGPLv2.1	Java	v1.0
Meridian [192]	centralized multi-threaded	extensible API layer	no	no	—	Java	v1.0
MobileFlow [223]	—	SDMN API	—	—	—	—	v1.2
MuL [231]	centralized multi-threaded	multi-level interface	no	no	GPLv2	C	v1.0
NOX [26]	centralized	ad-hoc API	no	no	GPLv3	C++	v1.0
NOX-MT [187]	centralized multi-threaded	ad-hoc API	no	no	GPLv3	C++	v1.0
NVP Controller [112]	distributed	—	—	—	commercial	—	—
OpenContrail [183]	—	REST API	no	no	Apache 2.0	Python, C++, Java	v1.0
OpenDaylight [13]	distributed	REST, RESTCONF	weak	no	EPL v1.0	Java	v1.{0,3}
ONOS [117]	distributed	RESTful API	weak, strong	yes	—	Java	v1.0
PANE [197]	distributed	PANE API	yes	—	—	—	—
POX [232]	centralized	ad-hoc API	no	no	GPLv3	Python	v1.0
ProgrammableFlow [233]	centralized	—	—	—	—	C	v1.3
Pratyaastha [198]	distributed	—	—	—	—	—	—
Rosemary [194]	centralized	ad-hoc	—	—	—	—	v1.0
Ryu NOS [191]	centralized multi-threaded	ad-hoc API	no	no	Apache 2.0	Python	v1.{0,2,3}
SMaRtLight [199]	distributed	RESTful API	yes	yes	—	Java	v1.0
SNAC [234]	centralized	ad-hoc API	no	no	GPL	C++	v1.0
Trema [190]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	C, Ruby	v1.0
Unified Controller [171]	—	REST API	—	—	commercial	—	v1.0
yanc [196]	distributed	file system	—	—	—	—	—

Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE* 103.1 (2015): 14-76.

Controller Architectures

TABLE V
ARCHITECTURE AND DESIGN ELEMENTS OF CONTROL PLATFORMS

Component	OpenDaylight	OpenContrail	HP VAN SDN	Onix	Beacon
Base network services	Topology/Stats/Switch Manager, Host Tracker, Shortest Path Forwarding	Routing, Tenant Isolation	Audit Log, Alerts, Topology, Discovery	Discovery, Multi-consistency Storage, Read State, Register for updates	Topology, device manager, and routing
East/Westbound APIs	—	Control Node (XMPP-like control channel)	Sync API	Distribution I/O module	*Not present*
Integration Plug-ins	OpenStack Neutron	CloudStack, OpenStack	OpenStack	—	—
Management Interfaces	GUI/CLI, REST API	GUI/CLI	REST API Shell / GUI Shell	—	Web
Northbound APIs	REST, REST-CONF [201], Java APIs	REST APIs (configuration, operational, and analytic)	REST API, GUI Shell	Onix API (general purpose)	API (based on OpenFlow events)
Service abstraction layers	Service Abstraction Layer (SAL)	—	Device Abstraction API	Network Information Base (NIB) Graph with Import/Export Functions	—
Southbound APIs or connectors	OpenFlow, OVSDB, SNMP, PCEP, BGP, NETCONF	—	OpenFlow, L3 Agent, L2 Agent	OpenFlow, OVSDB	OpenFlow

Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey."*Proceedings of the IEEE* 103.1 (2015): 14-76.

That‘s a lot of Choices!?

„There are almost as many controllers for SDNs as there are SDNs“ – Nick Feamster

Which controller should I use for what problem?

Which controller?

Concept?

Architecture?

Programming language and model?

Advantages / Disadvantages?

Learning Curve?

Developing Community?

Type of target network?

Centralized Controllers

NOX

- **The first controller**
  - Open source
  - Stable
  - No… …thon

    No longer supported
  - „New“ NOX: C++ only
    - OF version supported: 1.0

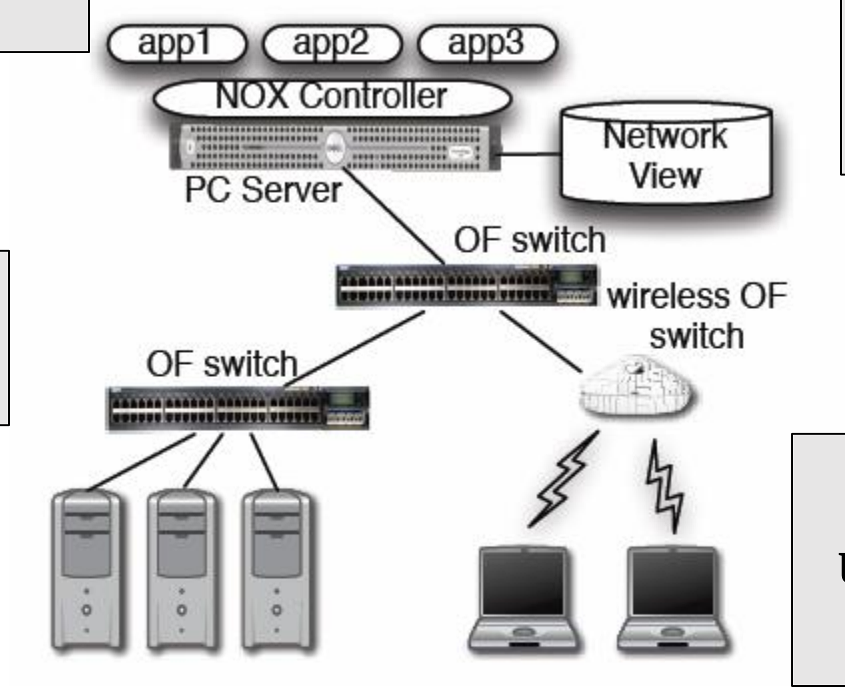
NOX Architecture

Granularity of Control: Per Flow

Controller maintains a network view

switches and attached servers

OpenFlow is used to control switches



NOX Architecture

Programming model: Controller listens for OF events

Programmer writes action handlers for events

When to use NOX

- Need to use low-level semantics of OpenFlow
  - NOX does not come with many abstractions
- Need of good performance (C++)
  - E.g.: production networks

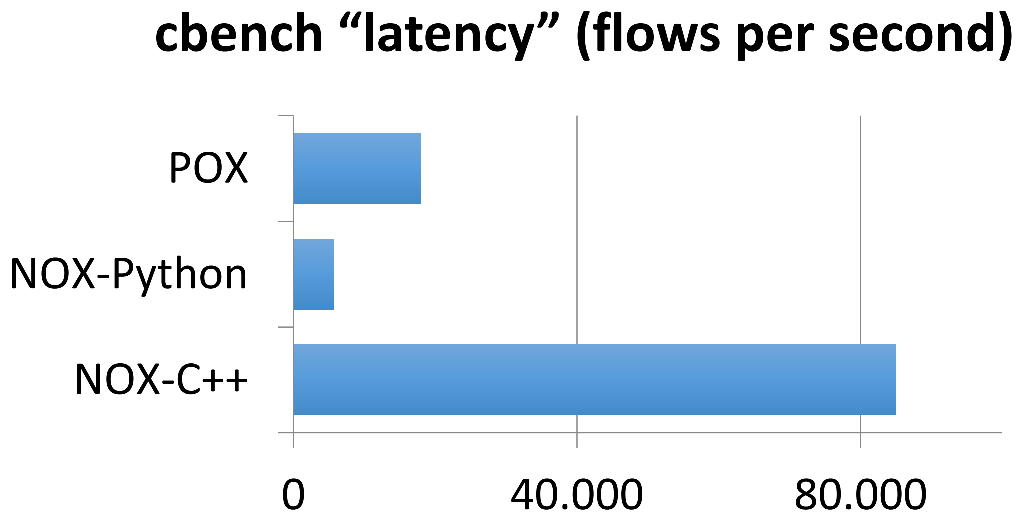
POX

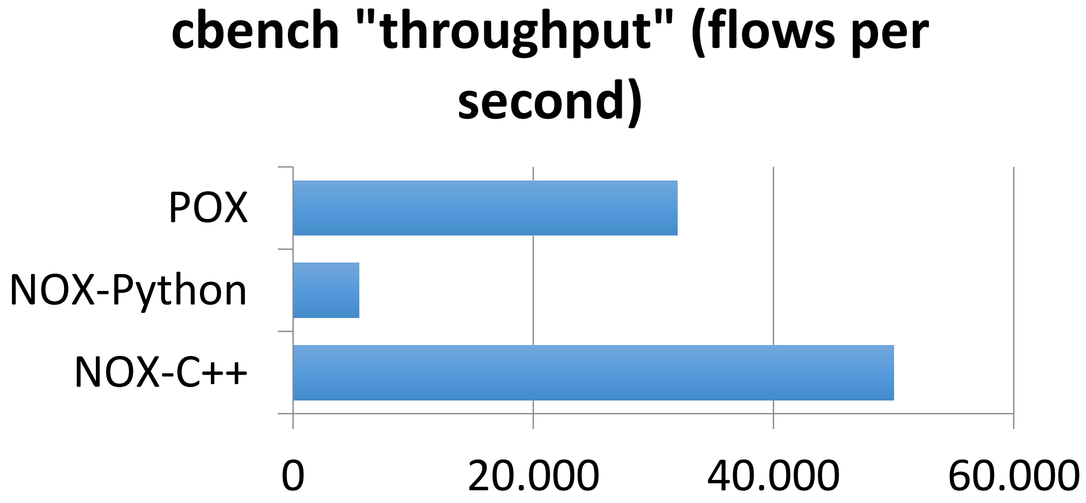
- **POX = NOX in Python**

- Advantages:
  - Widely used, maintained and supported
  - Relatively easy to write code for

- Disadvantage:
  - Performance (Python is slower than C++)
  - But: can feed POX ideas back to NOX for production use

POX


cbench “latency” (flows per second)
POX
NOX-Python
NOX-C++
0
40.000
80.000


cbench "throughput" (flows per second)
POX
NOX-Python
NOX-C++
0
20.000
40.000
60.000

When to use POX

- Learning, testing, debugging, evaluation

In this class :)

- Probably not in large production networks

More advanced: Floodlight

- Java

- Advantages:
  - Documentation,
  - REST API conformity
  - Production-level performance

- Disadvantage:
  - Steep learning curve

Floodlight: Users

Floodlight Adopters:

- University research
- Networking vendors
- Users
- Developers / startups

Floodlight Overview

- FloodlightProvider (IFloodlightProviderService)
- TopologyManager (ITopologyManagerService)
- LinkDiscovery (ILinkDiscoveryService)
- Forwarding
- DeviceManager (IDeviceService)
- StorageSource (IStorageSourceService)
- RestServer (IRestApiService)
- StaticFlowPusher (IStaticFlowPusherService)
- VirtualNetworkFilter (IVirtualNetworkFilterService)

- Floodlight is a collection of modules
- Some modules (not all) export services
- All modules in Java
- Rich, extensible REST API

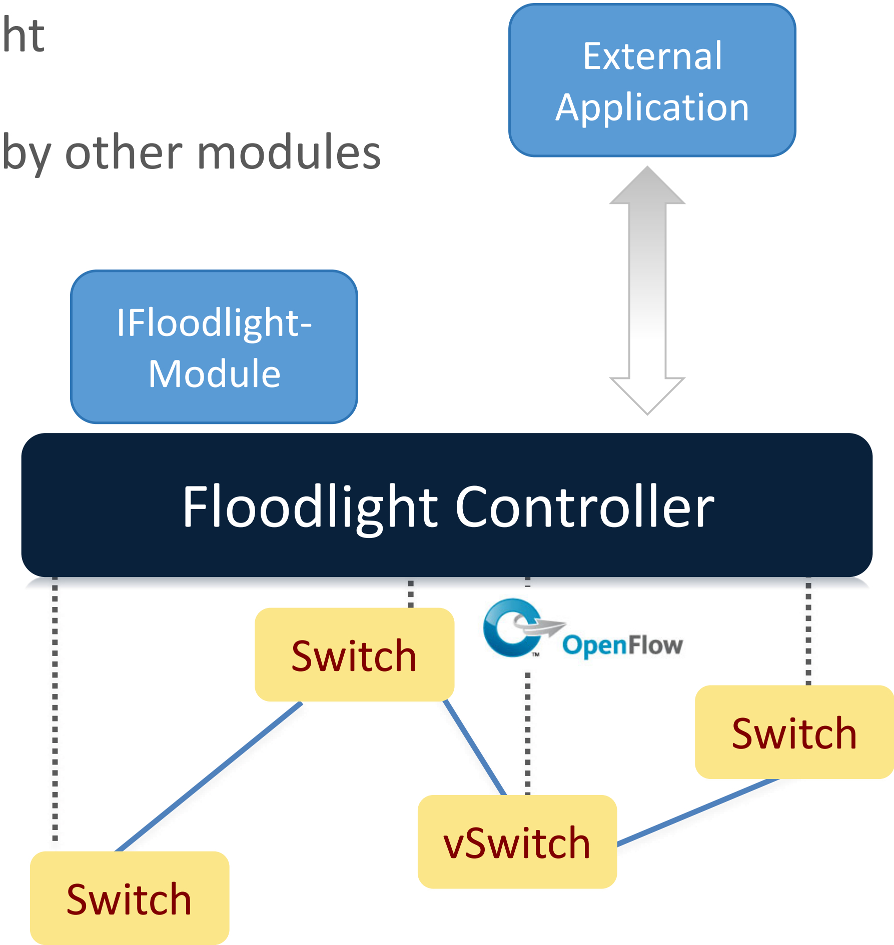
Floodlight Overview

FloodlightProvider (IFloodlightProviderService)	• Translates OF messages to Floodlight events • Managing connections to switches via Netty
TopologyManager (ITopologyManagerService)	• Computes shortest path using Dijsktra • Keeps switch to cluster mappings
LinkDiscovery (ILinkDiscoveryService)	• Maintains state of links in network • Sends out LLDPs
Forwarding	• Installs flow mods for end-to-end routing • Handles island routing
DeviceManager (IDeviceService)	• Tracks hosts on the network • MAC -> switch,port, MAC->IP, IP->MAC
StorageSource (IStorageSourceService)	
RestServer (IRestApiService)	• Implements via Restlets (restlet.org) • Modules export RestletRoutable
StaticFlowPusher (IStaticFlowPusherService)	• Supports the insertion and removal of static flows • REST-based API
VirtualNetworkFilter (IVirtualNetworkFilterService)	• Create layer 2 domain defined by MAC address

Floodlight Programming Model

IFloodlightModule

- Java module that runs as part of Floodlight
- Consumes services and events exported by other modules
  - OpenFlow (ie. Packet-in)
  - Switch add / remove
  - Device add /remove / move
  - Link discovery



External Application

- Communicates with Floodlight via REST

Floodlight Modules

Network State

List Hosts

List Links

List Switches

GetStats (DPID)

GetCounters (OFType…)

Static Flows

Add Flow

Delete Flow

List Flows

RemoveAll Flows

Virtual Network

Create Network

Delete Network

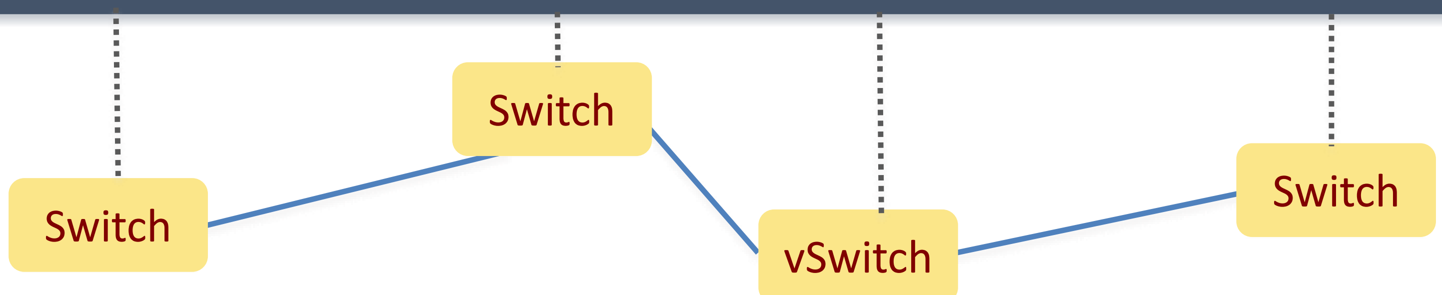
Add Host

Remove Host

User Extensions

…

Floodlight Controller



When to use Floodlight

- If you know JAVA
- If you need production-level performance
- Have/want to use REST API

Other Controllers...

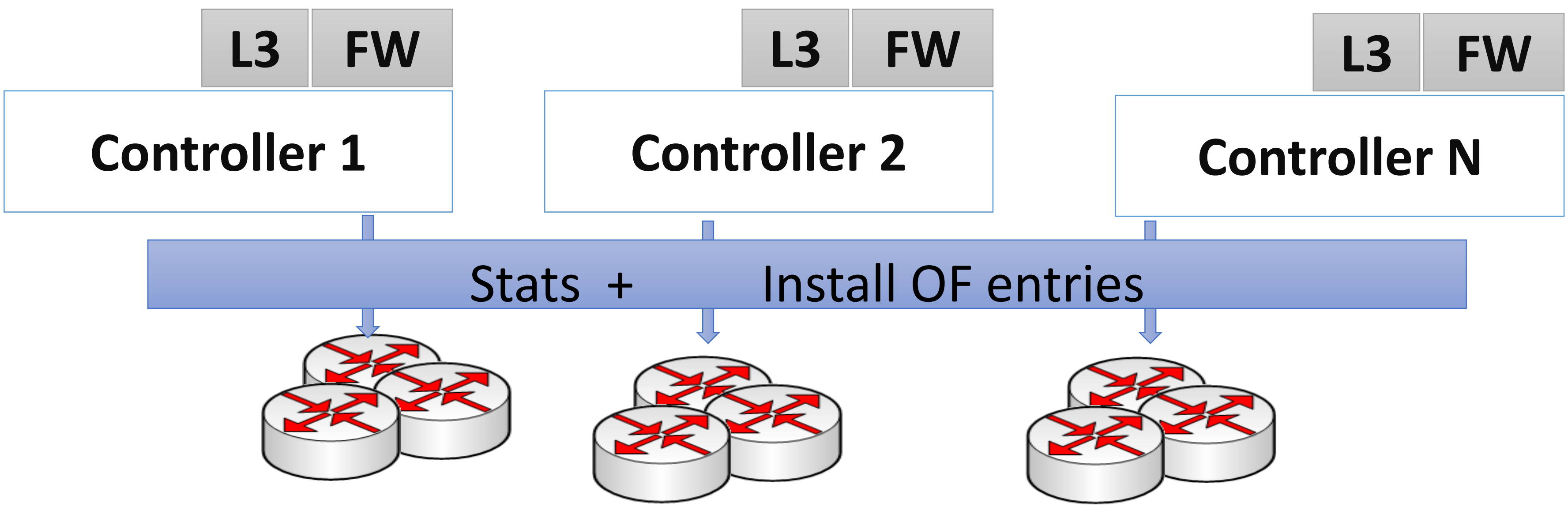
TABLE VI
CONTROLLERS CLASSIFICATION

Name	Architecture	Northbound API	Consistency	Faults	License	Prog. language	Version
Beacon [186]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	Java	v1.0
DISCO [185]	distributed	REST	—	yes	—	Java	v1.1
ElastiCon [229]	distributed	RESTful API	yes	no	—	Java	v1.0
Fleet [200]	distributed	ad-hoc	no	no	—	—	v1.0
Floodlight [189]	centralized multi-threaded	RESTful API	no	no	Apache	Java	v1.1
HP VAN SDN [184]	distributed	RESTful API	weak	yes	—	Java	v1.0
HyperFlow [195]	distributed	—	weak	yes	—	C++	v1.0
Kandoo [230]	hierarchically distributed	—	no	no	—	C, C++, Python	v1.0
Onix [7]	distributed	NVP NBAPI	weak, strong	yes	commercial	Python, C	v1.0
Maestro [188]	centralized multi-threaded	ad-hoc API	no	no	LGPLv2.1	Java	v1.0
Meridian [192]	centralized multi-threaded	extensible API layer	no	no	—	Java	v1.0
MobileFlow [223]	—	SDMN API	—	—	—	—	v1.2
MuL [231]	centralized multi-threaded	multi-level interface	no	no	GPLv2	C	v1.0
NOX [26]	centralized	ad-hoc API	no	no	GPLv3	C++	v1.0
NOX-MT [187]	centralized multi-threaded	ad-hoc API	no	no	GPLv3	C++	v1.0
NVP Controller [112]	distributed	—	—	—	commercial	—	—
OpenContrail [183]	—	REST API	no	no	Apache 2.0	Python, C++, Java	v1.0
OpenDaylight [13]	distributed	REST, RESTCONF	weak	no	EPL v1.0	Java	v1.{0,3}
ONOS [117]	distributed	RESTful API	weak, strong	yes	—	Java	v1.0
PANE [197]	distributed	PANE API	yes	—	—	—	—
POX [232]	centralized	ad-hoc API	no	no	GPLv3	Python	v1.0
ProgrammableFlow [233]	centralized	—	—	—	—	C	v1.3
Pratyaastha [198]	distributed	—	—	—	—	—	—
Rosemary [194]	centralized	ad-hoc	—	—	—	—	v1.0
Ryu NOS [191]	centralized multi-threaded	ad-hoc API	no	no	Apache 2.0	Python	v1.{0,2,3}
SMaRtLight [199]	distributed	RESTful API	yes	yes	—	Java	v1.0
SNAC [234]	centralized	ad-hoc API	no	no	GPL	C++	v1.0
Trema [190]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	C, Ruby	v1.0
Unified Controller [171]	—	REST API	—	—	commercial	—	v1.0
yanc [196]	distributed	file system	—	—	—	—	—

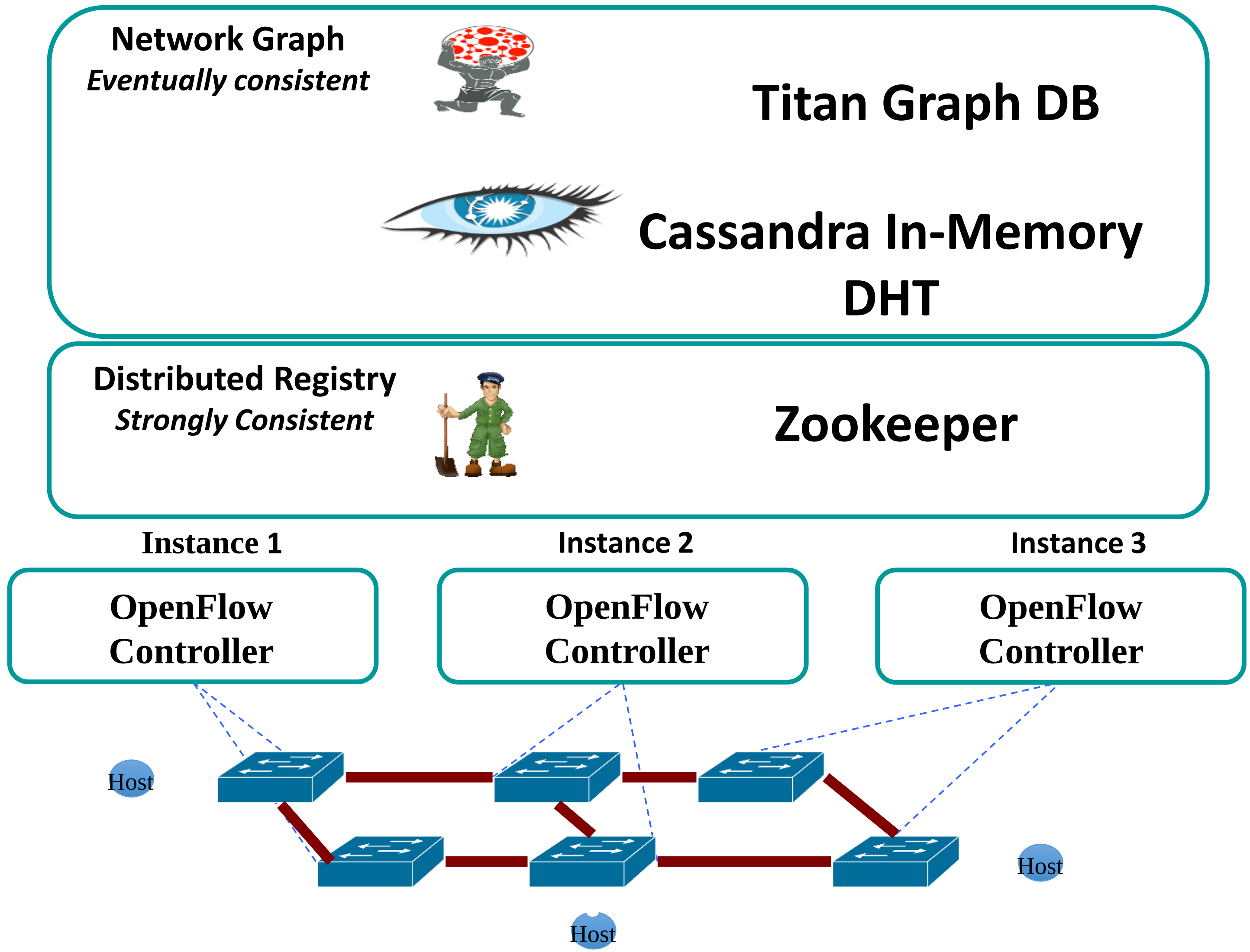
Distributed Controllers

How to scale the Controller?

- Obvious: add more controllers.
- BUT: how about the applications?
  - Synchronization/concurrency problems.
    - Who controls which switch?
    - Who reacts to which events?

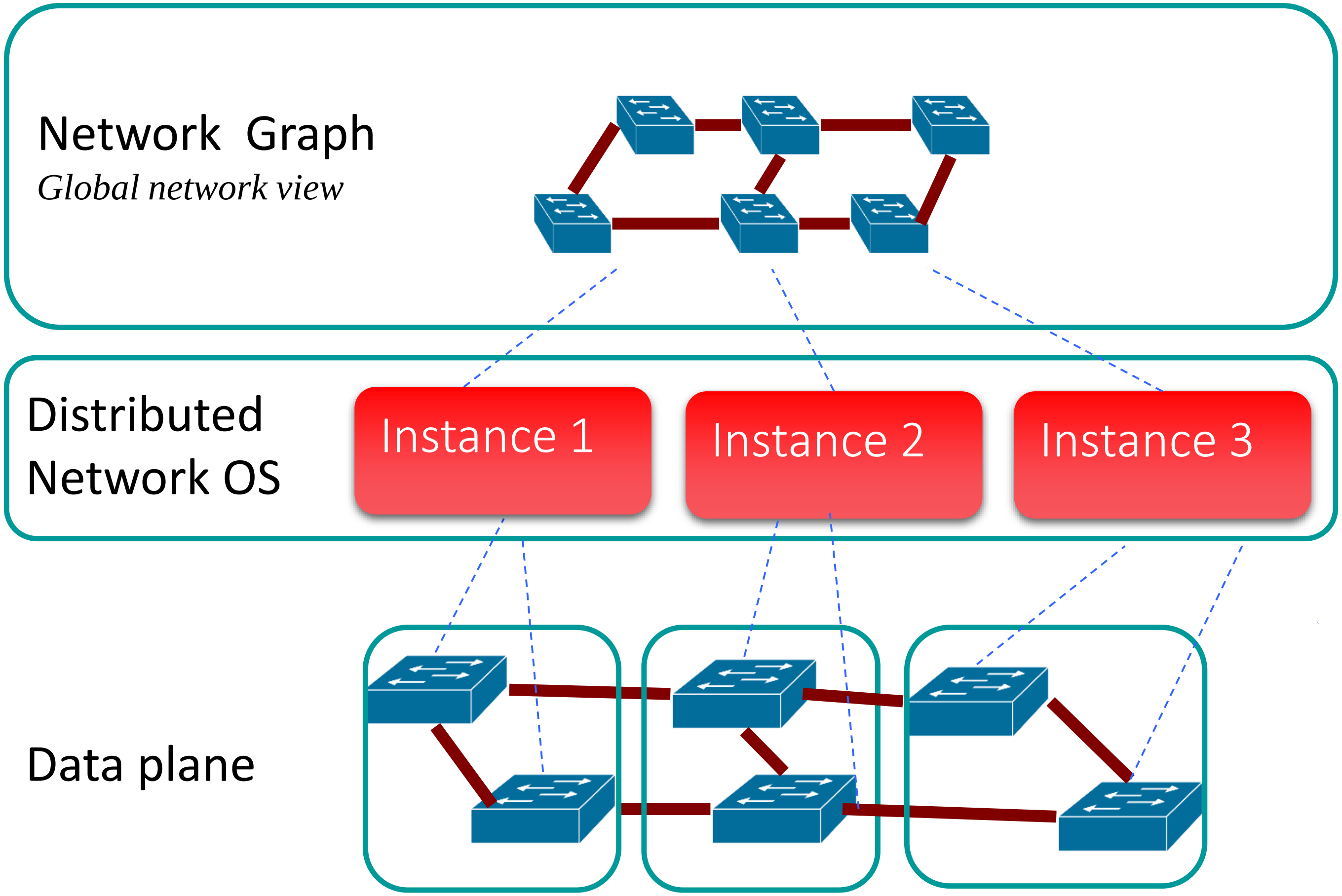


The ONOS Controller



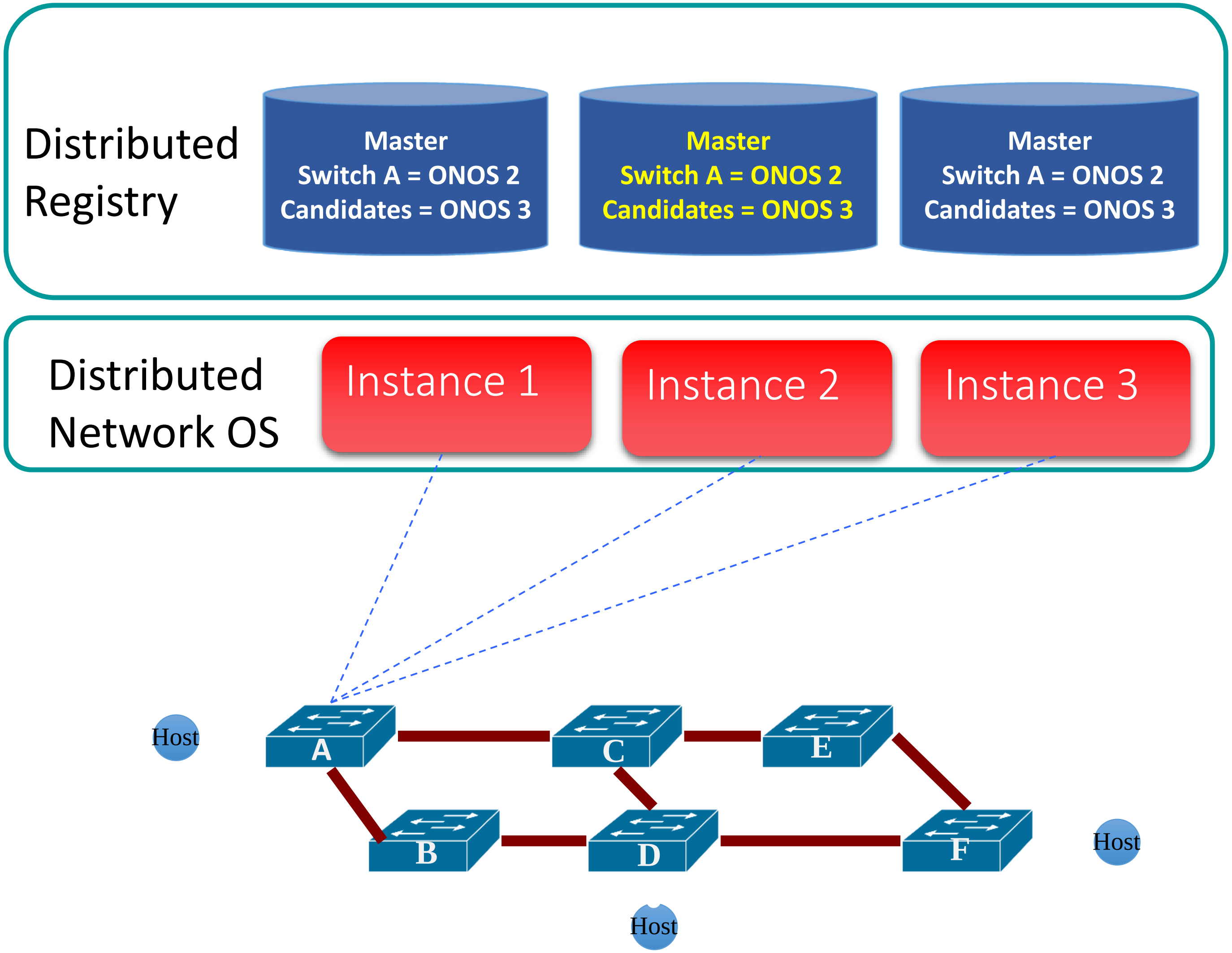
Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS." *HotSDN 2014*

ONOS Scaling

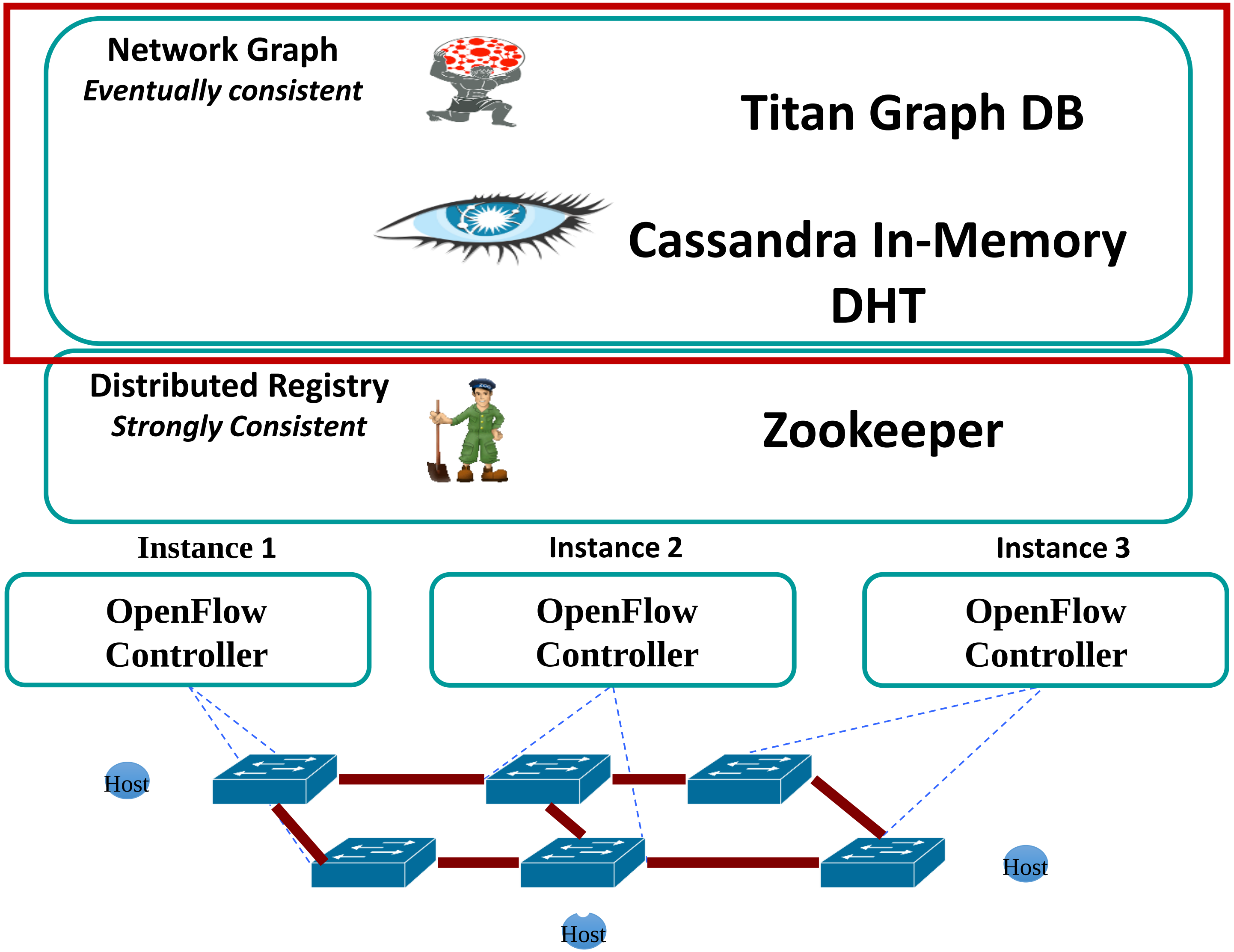


- An instance is responsible for maintaining a part of network graph
- Control capacity can grow with network size or application need

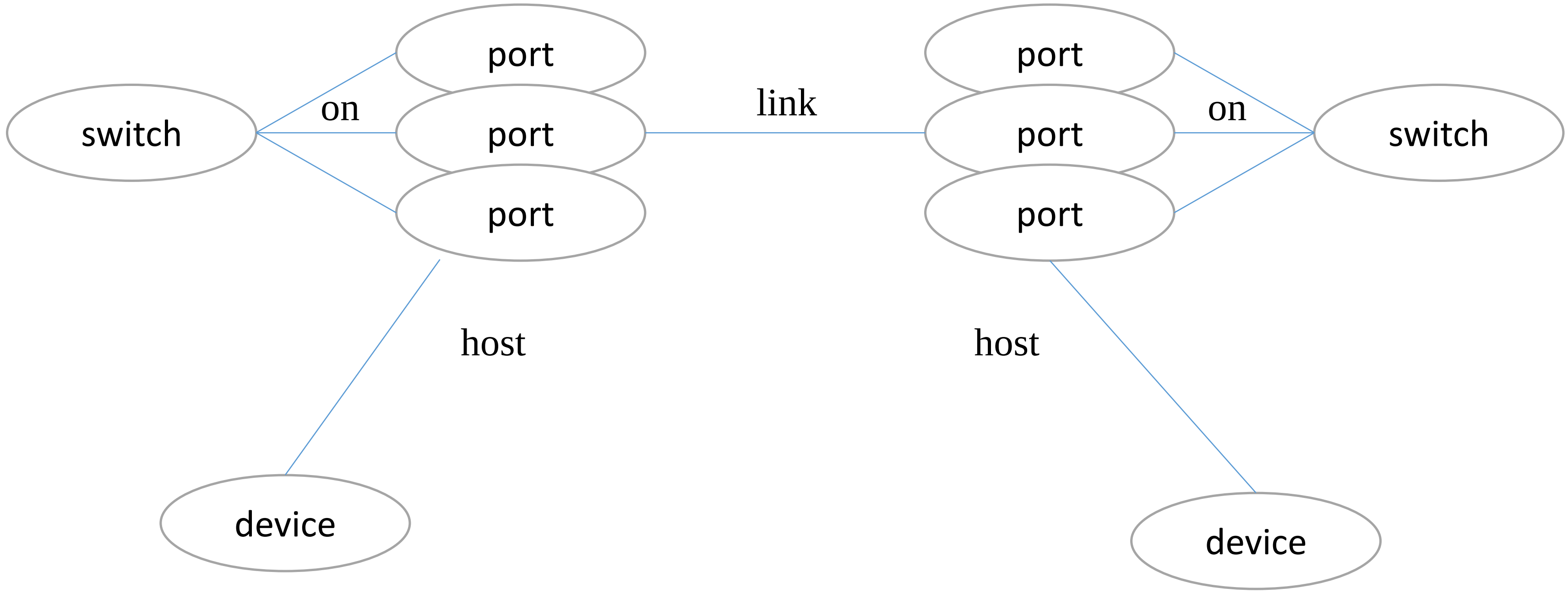
ONOS Control Plane Failover



ONOS Network Graph

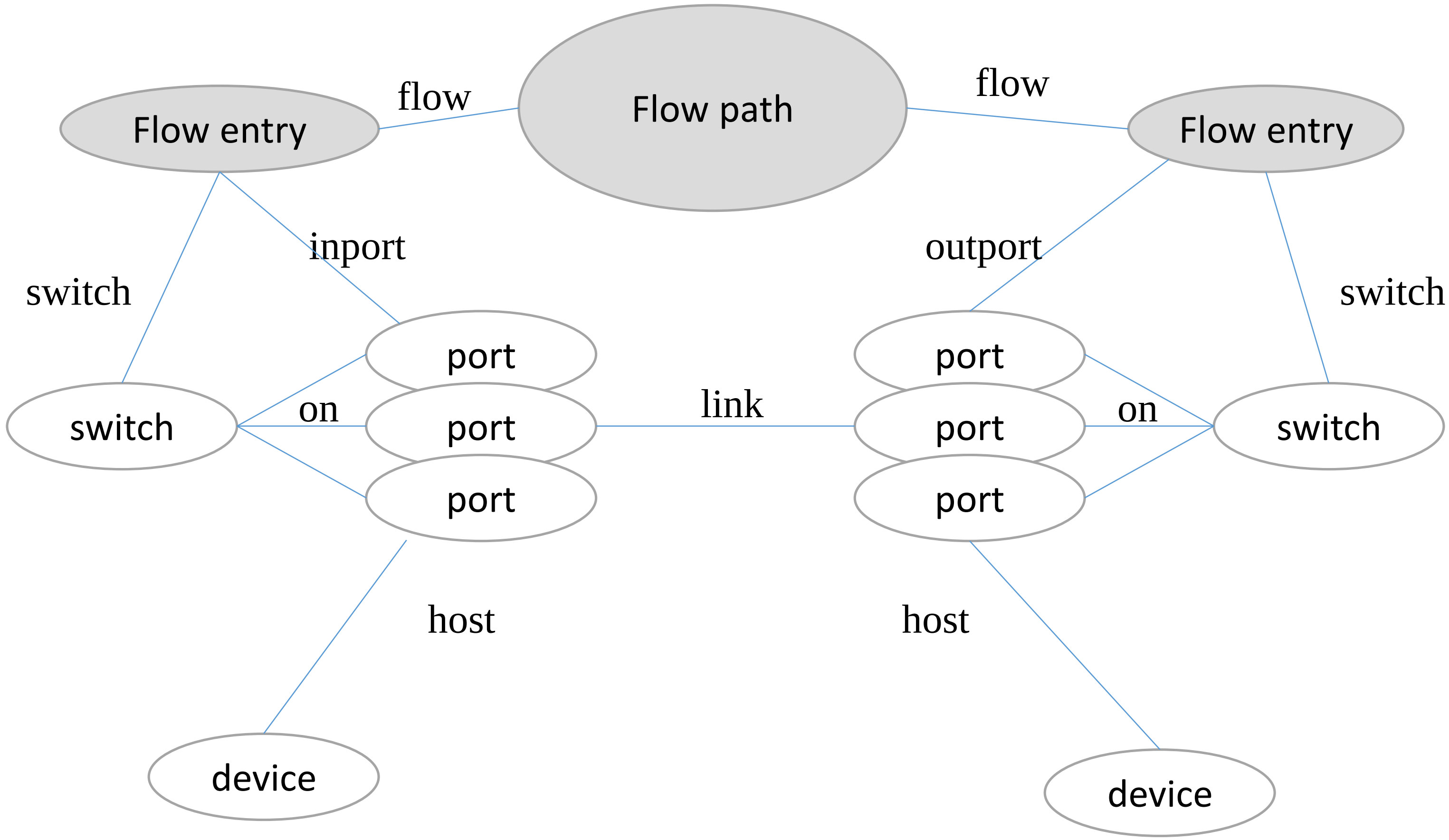


ONOS Network Graph



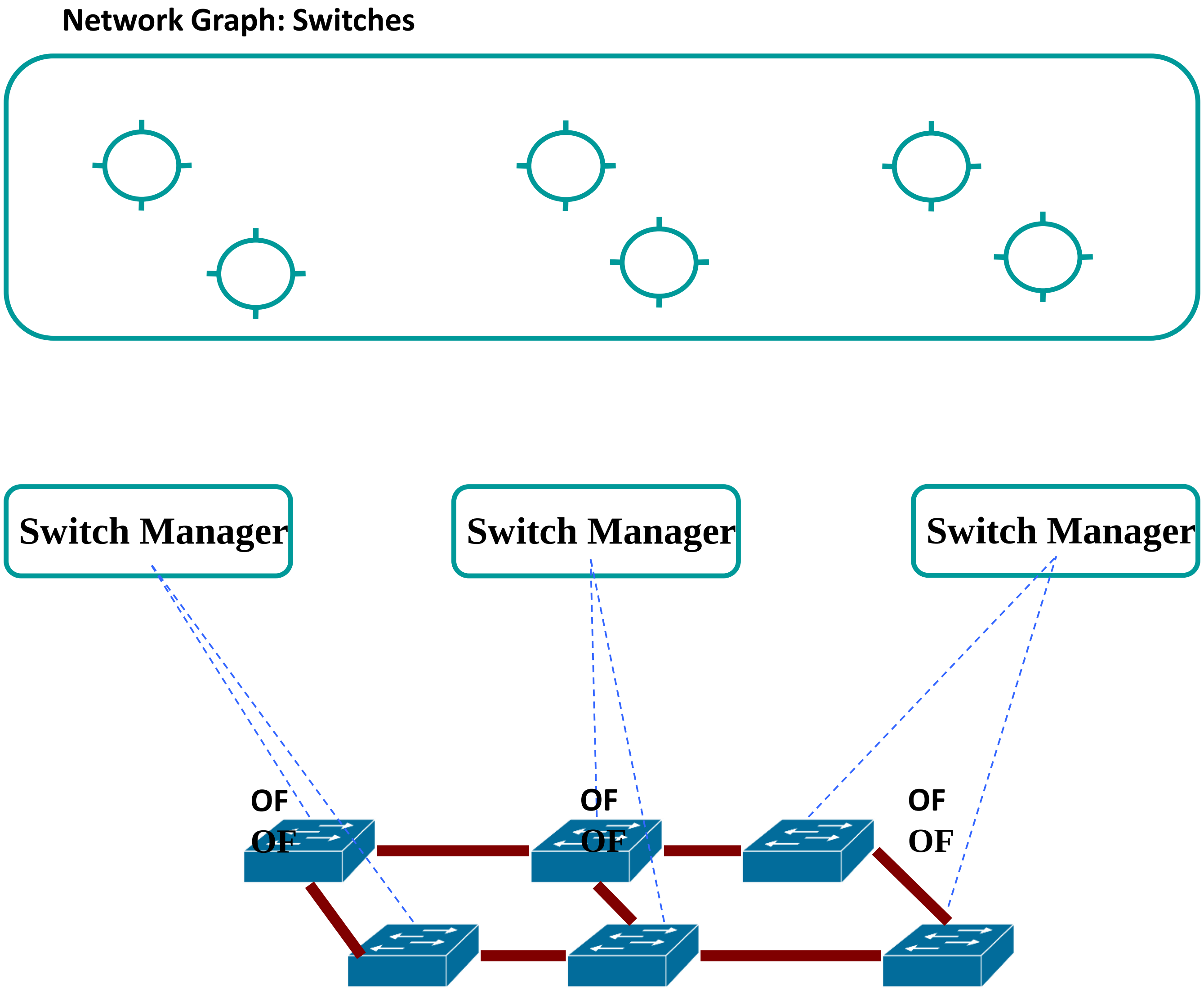
- Network state is naturally represented as a graph
- Graph has basic network objects like switch, port, device and links
- Application writes to this graph & programs the data plane

Example: Path Computation App on Network Graph



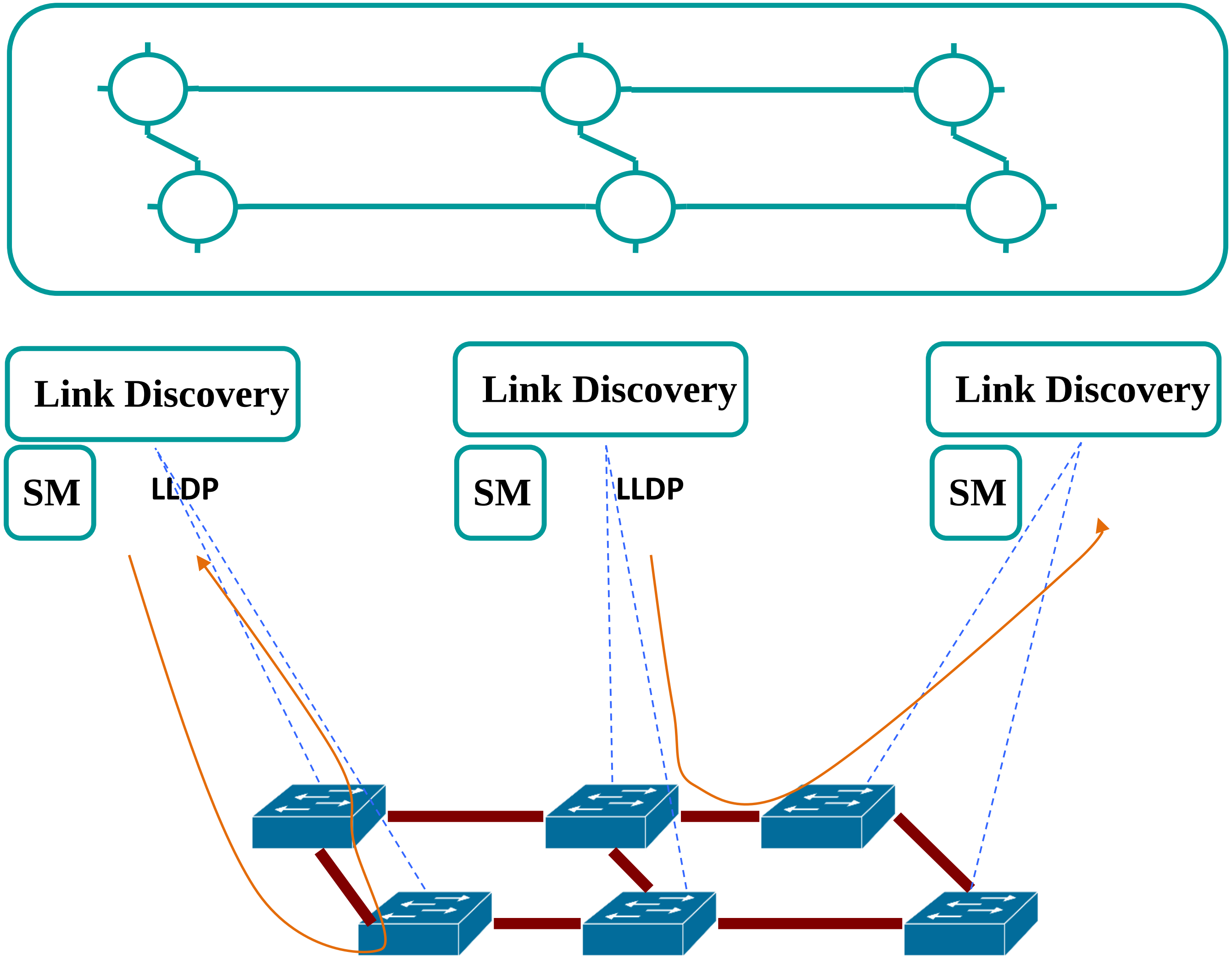
- Application computes path by traversing source->destination

Network Graph and Switches



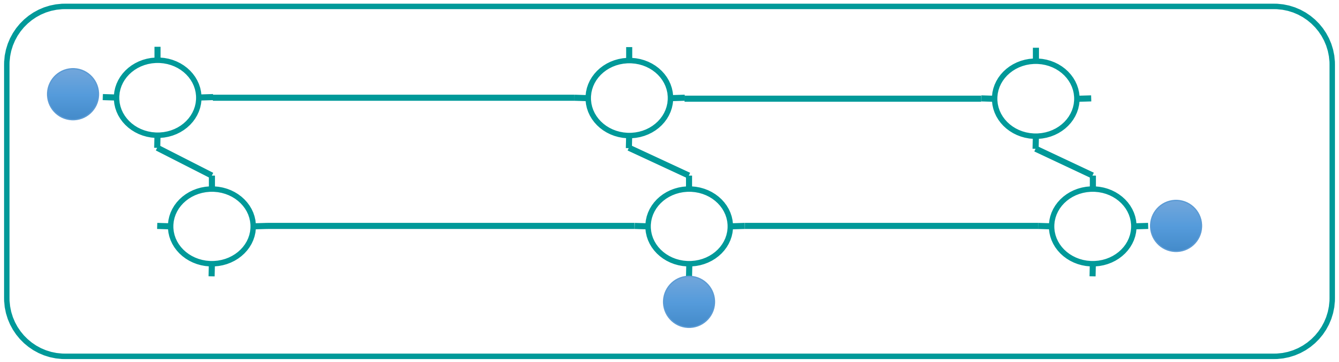
Network Graph and Link Discovery

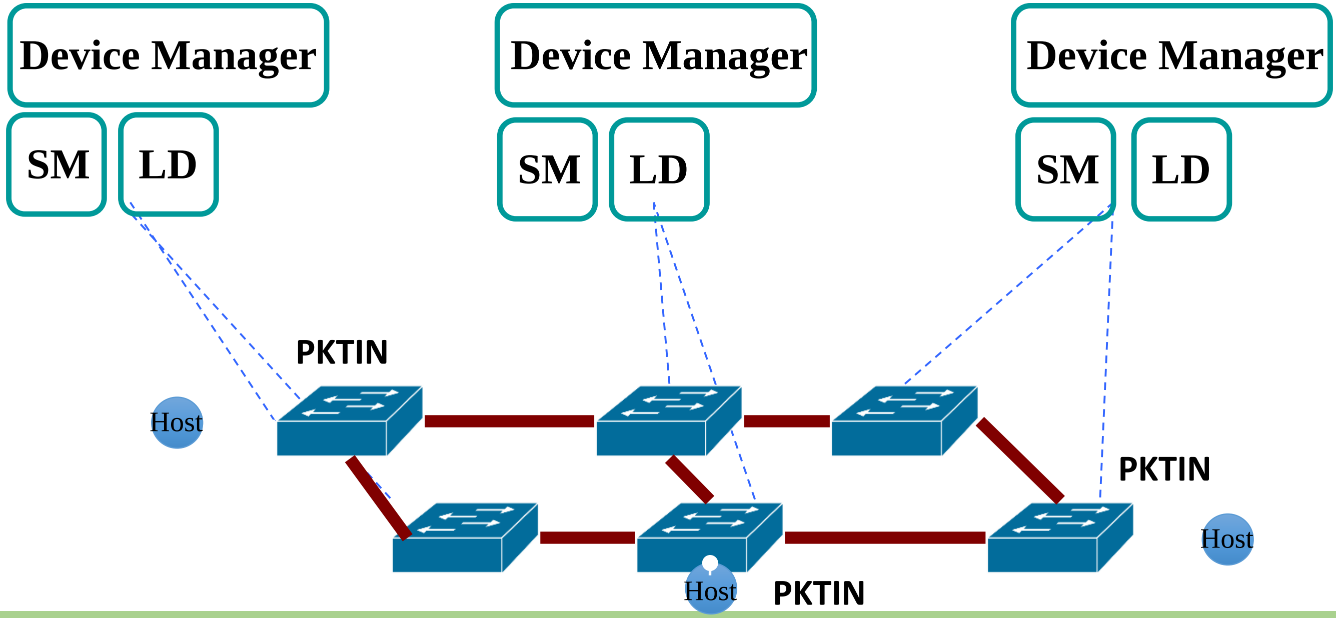
Network Graph: Links



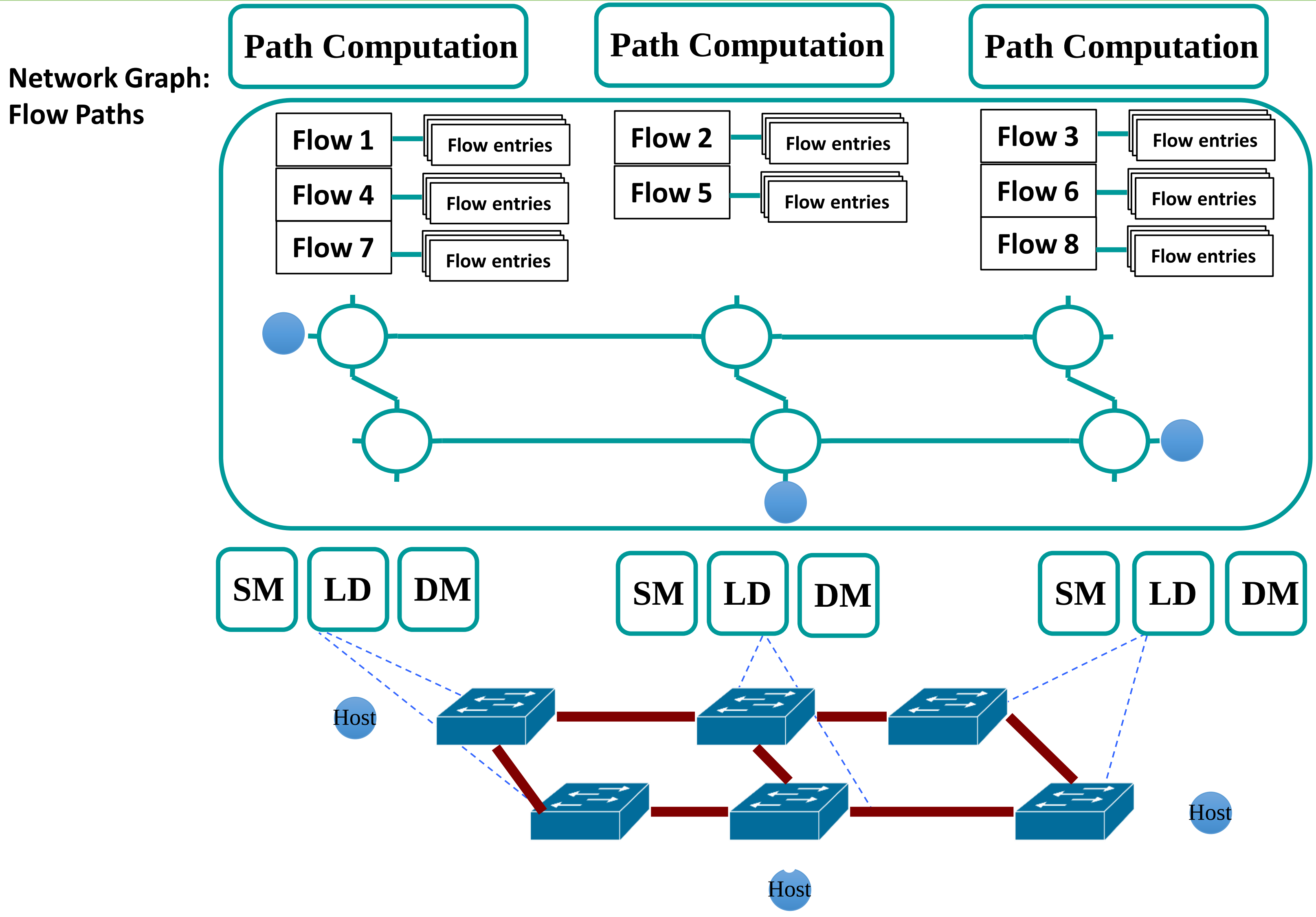
Devices and Network Graph

Network Graph: Devices



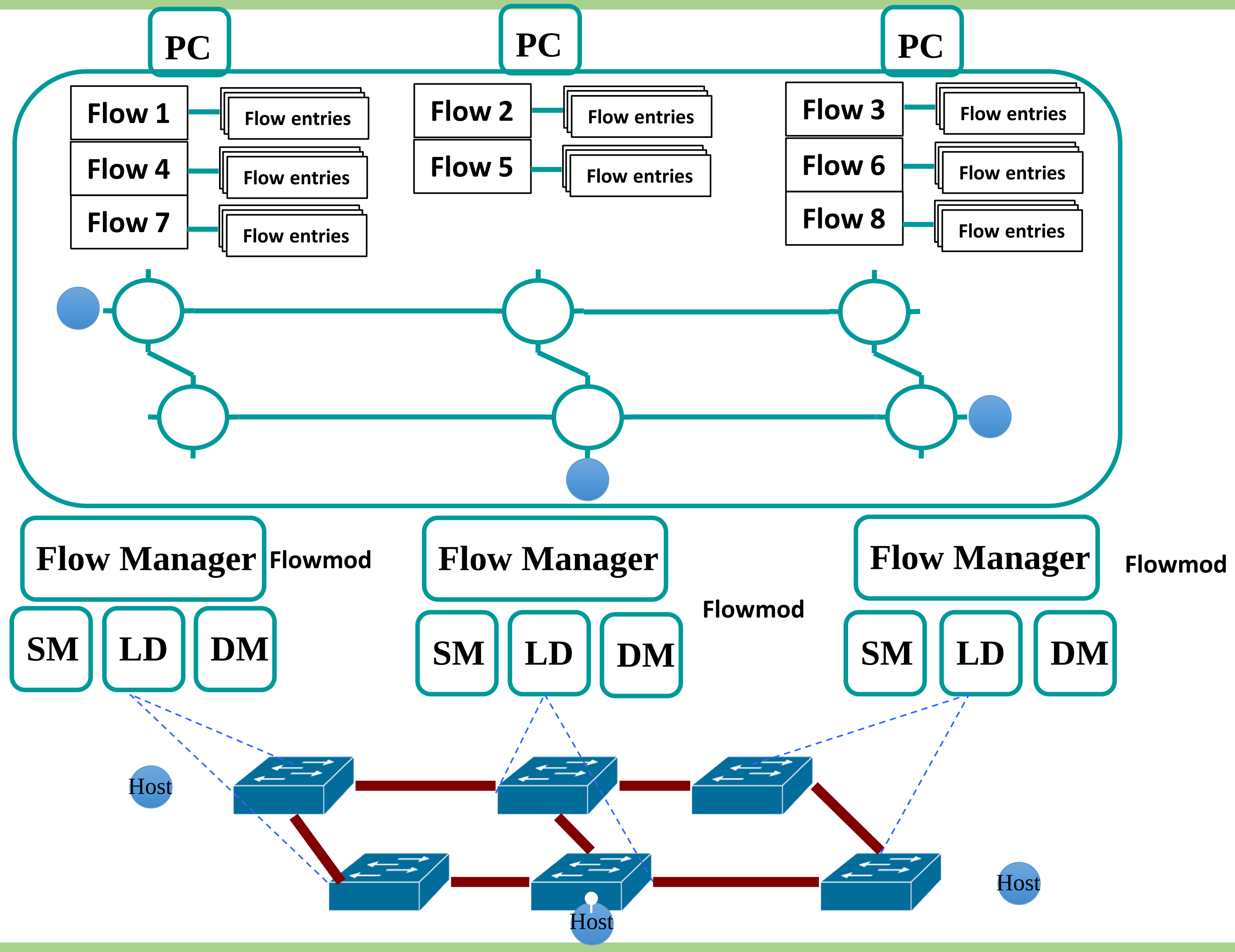


Path Computation with Network Graph



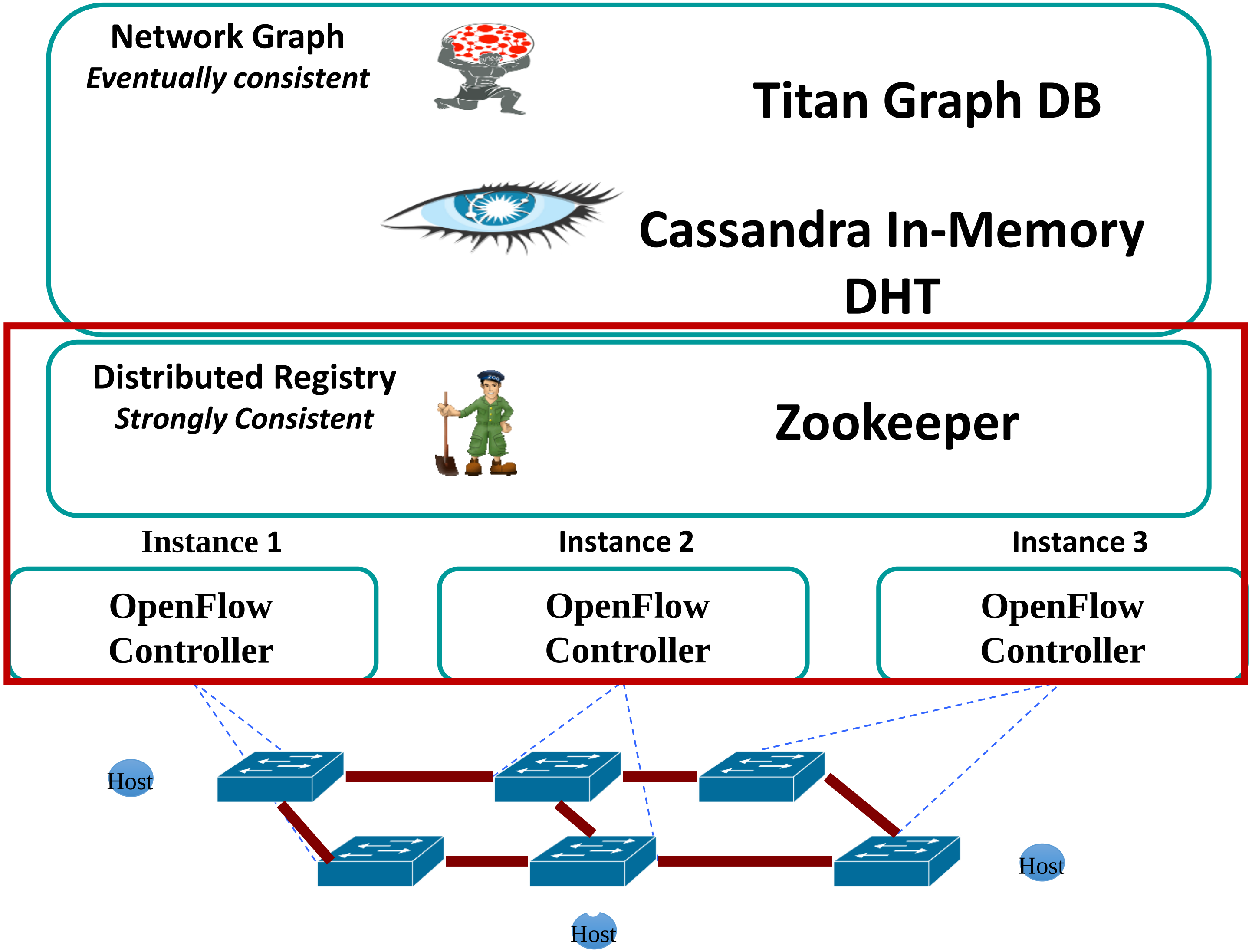
Network Graph and Flow Manager

Network Graph: Flows



CONSISTENCY

ONOS and Consistency



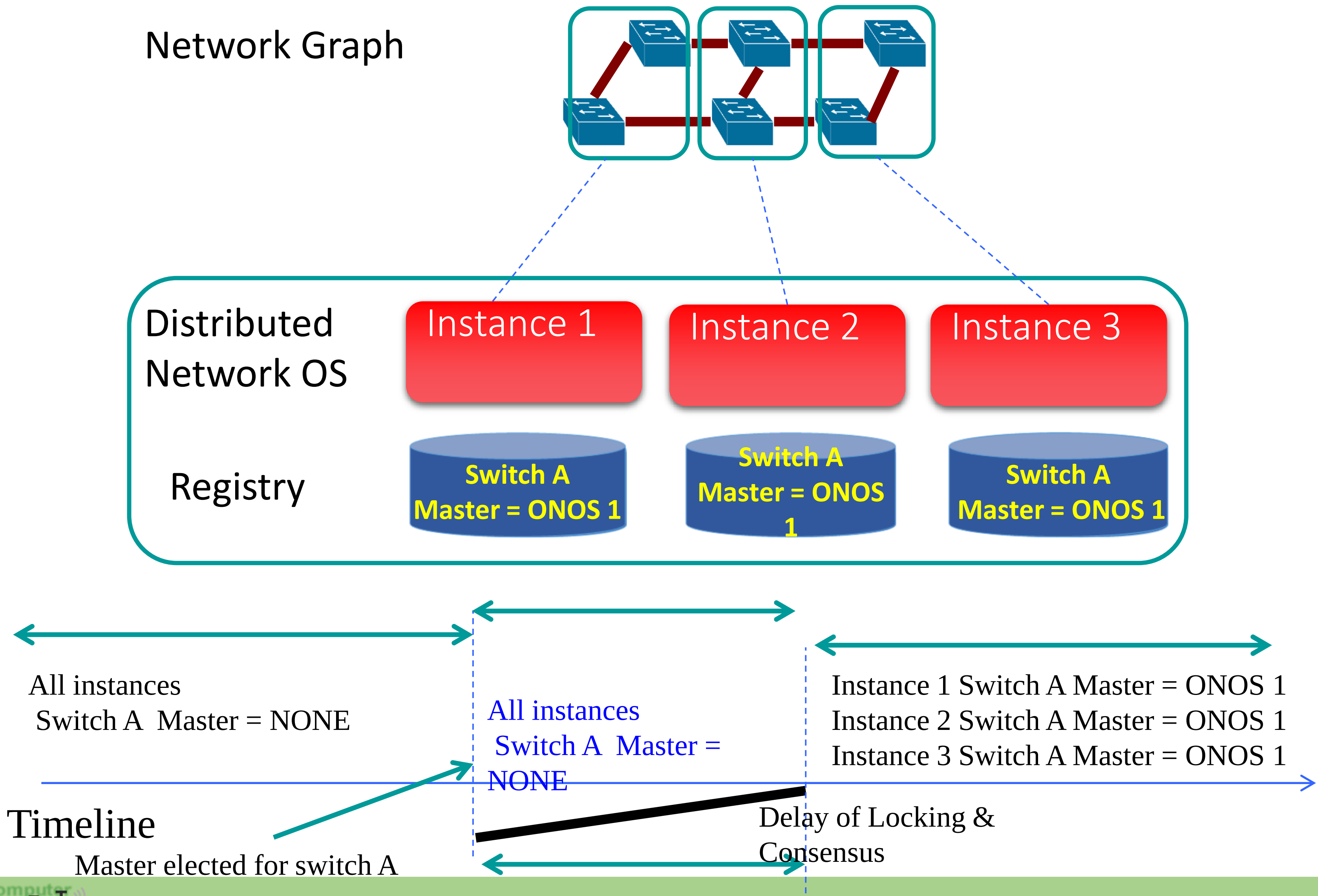
Consistency Definition

- Strong Consistency: Upon an update to the network state by an instance, all subsequent reads by any instance returns the last updated value.

- Strong consistency adds complexity and latency to distributed data management.

- Eventual consistency is slight relaxation – allowing readers to be behind for a short period of time.

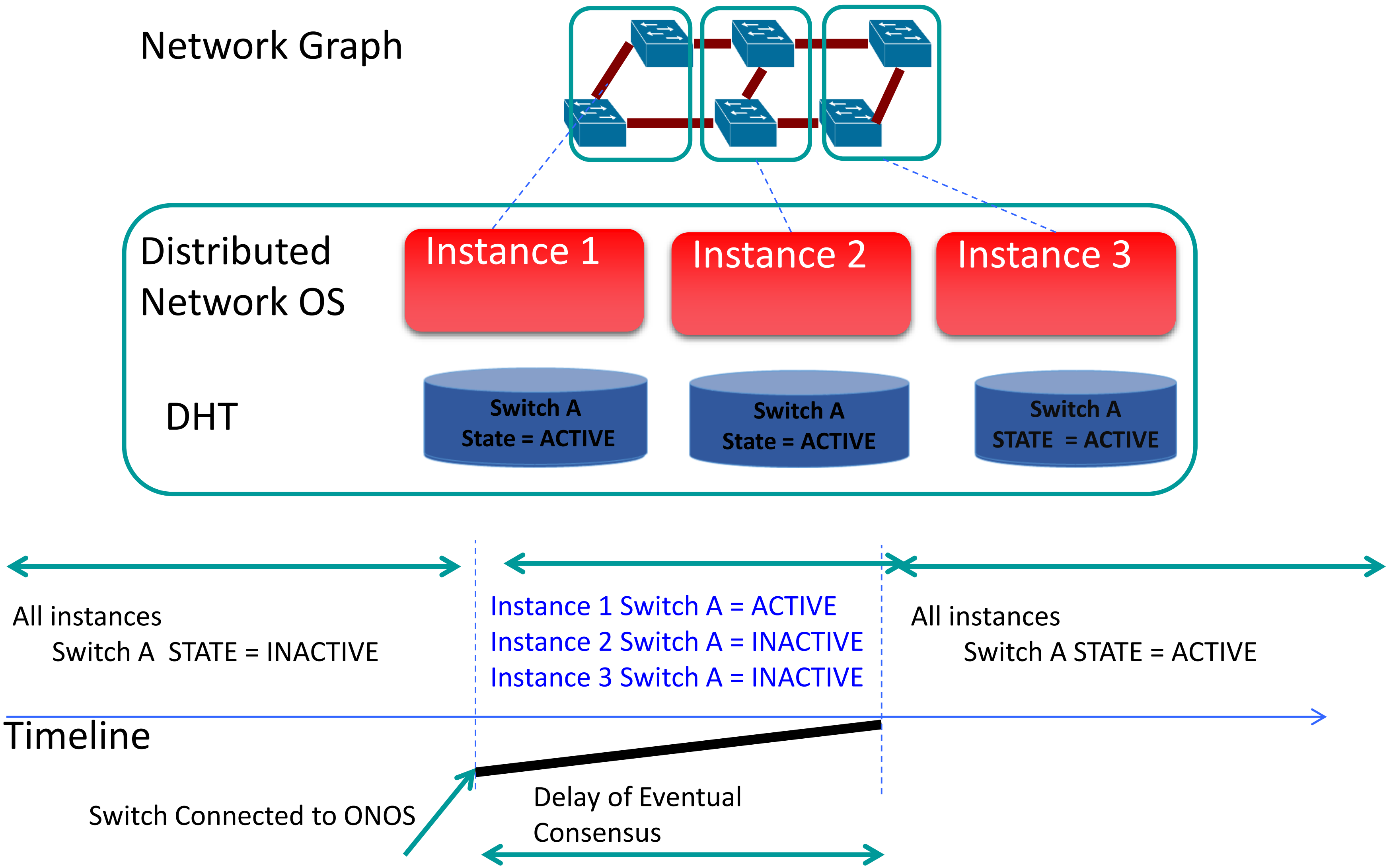
Strong Consistency using Registry



Why Strong Consistency is needed for Master Election

- Weaker consistency might mean Master election on instance 1 will not be available on other instances.
- Can lead to having multiple masters for a switch.

Eventual Consistency in Network Graph



Cost of Eventual Consistency

- Short delay will mean the switch A state is not ACTIVE on some ONOS instances in previous example.

- Applications on one instance will compute flow through the switch A while other instances will not use the switch A for path computation.

Is Eventual Consistency good enough?

- Physical network state changes asynchronously
  - Strong consistency across data and control plane is too hard
  - Control apps know how to deal with eventual consistency

- In the current distributed control plane, each router makes its own decision based on old info from other parts of the network and it works fine

Other Controllers...

TABLE VI
CONTROLLERS CLASSIFICATION

Name	Architecture	Northbound API	Consistency	Faults	License	Prog. language	Version
Beacon [186]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	Java	v1.0
DISCO [185]	distributed	REST	—	yes	—	Java	v1.1
ElastiCon [229]	distributed	RESTful API	yes	no	—	Java	v1.0
Fleet [200]	distributed	ad-hoc	no	no	—	—	v1.0
Floodlight [189]	centralized multi-threaded	RESTful API	no	no	Apache	Java	v1.1
HP VAN SDN [184]	distributed	RESTful API	weak	yes	—	Java	v1.0
HyperFlow [195]	distributed	—	weak	yes	—	C++	v1.0
Kandoo [230]	hierarchically distributed	—	no	no	—	C, C++, Python	v1.0
Onix [7]	distributed	NVP NBAPI	weak, strong	yes	commercial	Python, C	v1.0
Maestro [188]	centralized multi-threaded	ad-hoc API	no	no	LGPLv2.1	Java	v1.0
Meridian [192]	centralized multi-threaded	extensible API layer	no	no	—	Java	v1.0
MobileFlow [223]	—	SDMN API	—	—	—	—	v1.2
MuL [231]	centralized multi-threaded	multi-level interface	no	no	GPLv2	C	v1.0
NOX [26]	centralized	ad-hoc API	no	no	GPLv3	C++	v1.0
NOX-MT [187]	centralized multi-threaded	ad-hoc API	no	no	GPLv3	C++	v1.0
NVP Controller [112]	distributed	—	—	—	commercial	—	—
OpenContrail [183]	—	REST API	no	no	Apache 2.0	Python, C++, Java	v1.0
OpenDaylight [13]	distributed	REST, RESTCONF	weak	no	EPL v1.0	Java	v1.{0,3}
ONOS [117]	distributed	RESTful API	weak, strong	yes	—	Java	v1.0
PANE [197]	distributed	PANE API	yes	—	—	—	—
POX [232]	centralized	ad-hoc API	no	no	GPLv3	Python	v1.0
ProgrammableFlow [233]	centralized	—	—	—	—	C	v1.3
Pratyaastha [198]	distributed	—	—	—	—	—	—
Rosemary [194]	centralized	ad-hoc	—	—	—	—	v1.0
Ryu NOS [191]	centralized multi-threaded	ad-hoc API	no	no	Apache 2.0	Python	v1.{0,2,3}
SMaRtLight [199]	distributed	RESTful API	yes	yes	—	Java	v1.0
SNAC [234]	centralized	ad-hoc API	no	no	GPL	C++	v1.0
Trema [190]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	C, Ruby	v1.0
Unified Controller [171]	—	REST API	—	—	commercial	—	v1.0
yanc [196]	distributed	file system	—	—	—	—	—

Controller Popularity

TABLE VI
CONTROLLERS CLASSIFICATION

Name	Architecture	Northbound API	Consistency	Faults	License	Prog. language	Version
Beacon [186]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	Java	v1.0
DISCO [185]	distributed	REST	—	yes	—	Java	v1.1
ElastiCon [229]	distributed	RESTful API	yes	no	—	Java	v1.0
Fleet [200]	distributed	ad-hoc	no	no	—	—	v1.0
Floodlight [189]	centralized multi-threaded	RESTful API	no	no	Apache	Java	v1.1
HP VAN SDN [184]	distributed	RESTful API	weak	yes	—	Java	v1.0
HyperFlow [195]	distributed	—	weak	yes	—	C++	v1.0
Kandoo [230]	hierarchically distributed	—	no	no	—	C, C++, Python	v1.0
Onix [7]	distributed	NVP NBAPI	weak, strong	yes	commercial	Python, C	v1.0
Maestro [188]	centralized multi-threaded	ad-hoc API	no	no	LGPLv2.1	Java	v1.0
Meridian [192]	centralized multi-threaded	extensible API layer	no	no	—	Java	v1.0
MobileFlow [223]	—	SDMN API	—	—	—	—	v1.2
MuL [231]	centralized multi-threaded	multi-level interface	no	no	GPLv2	C	v1.0
NOX [26]	centralized	ad-hoc API	no	no	GPLv3	C++	v1.0
NOX-MT [187]	centralized multi-threaded	ad-hoc API	no	no	GPLv3	C++	v1.0
NVP Controller [112]	distributed	—	—	—	commercial	—	—
OpenContrail [183]	—	REST API	no	no	Apache 2.0	Python, C++, Java	v1.0
OpenDaylight [13]	distributed	REST, RESTCONF	weak	no	EPL v1.0	Java	v1.{0,3}
ONOS [117]	distributed	RESTful API	weak, strong	yes	—	Java	v1.0
PANE [197]	distributed	PANE API	yes	—	—	—	—
POX [232]	centralized	ad-hoc API	no	no	GPLv3	Python	v1.0
ProgrammableFlow [233]	centralized	—	—	—	—	C	v1.3
Pratyaastha [198]	distributed	—	—	—	—	—	—
Rosemary [194]	centralized	ad-hoc	—	—	—	—	v1.0
Ryu NOS [191]	centralized multi-threaded	ad-hoc API	no	no	Apache 2.0	Python	v1.{0,2,3}
SMaRtLight [199]	distributed	RESTful API	yes	yes	—	Java	v1.0
SNAC [234]	centralized	ad-hoc API	no	no	GPL	C++	v1.0
Trema [190]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	C, Ruby	v1.0
Unified Controller [171]	—	REST API	—	—	commercial	—	v1.0
yanc [196]	distributed	file system	—	—	—	—	—

Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE* 103.1 (2015): 14-76.