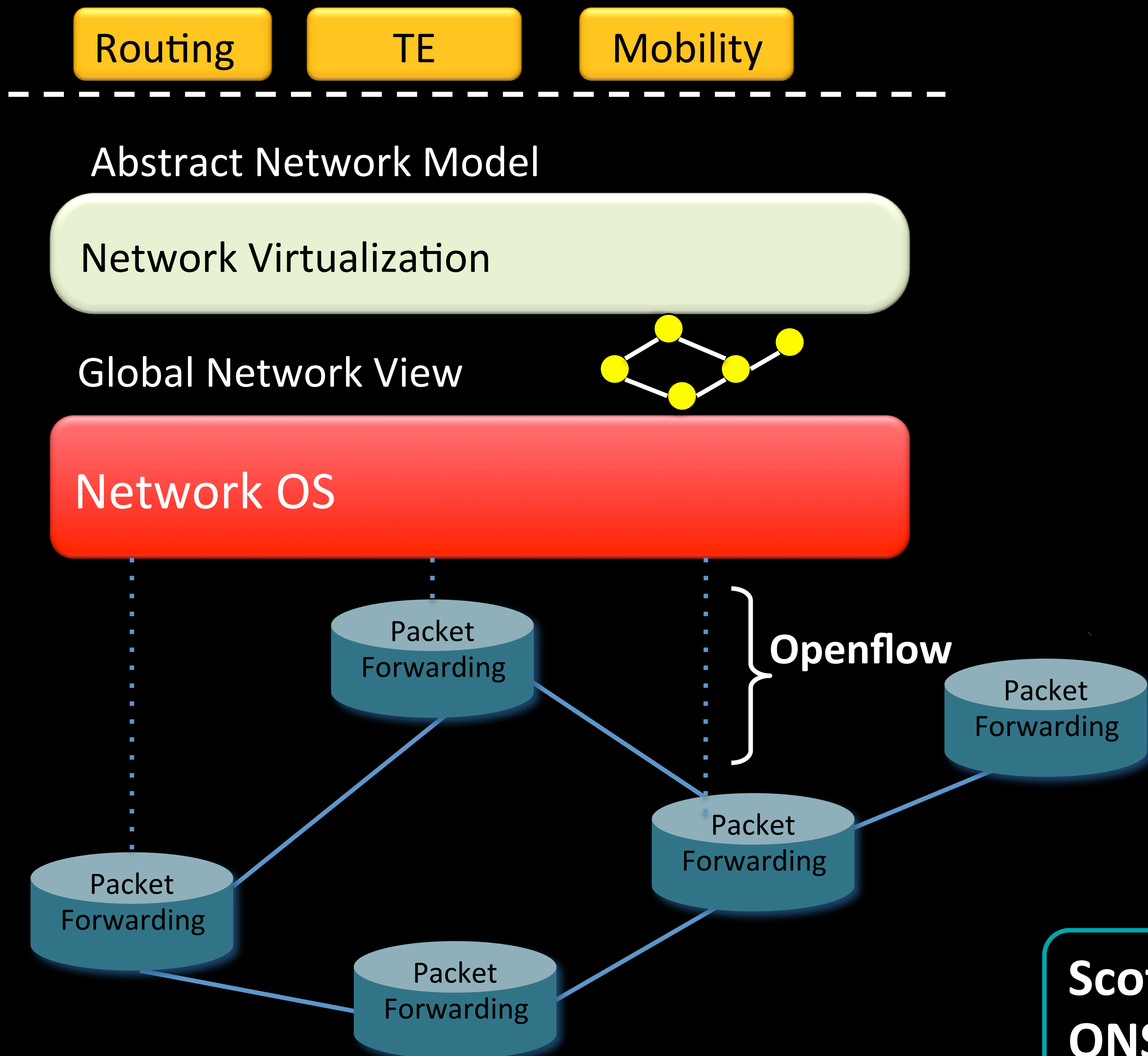
ON.LAB

ONOS
Open Network Operating System

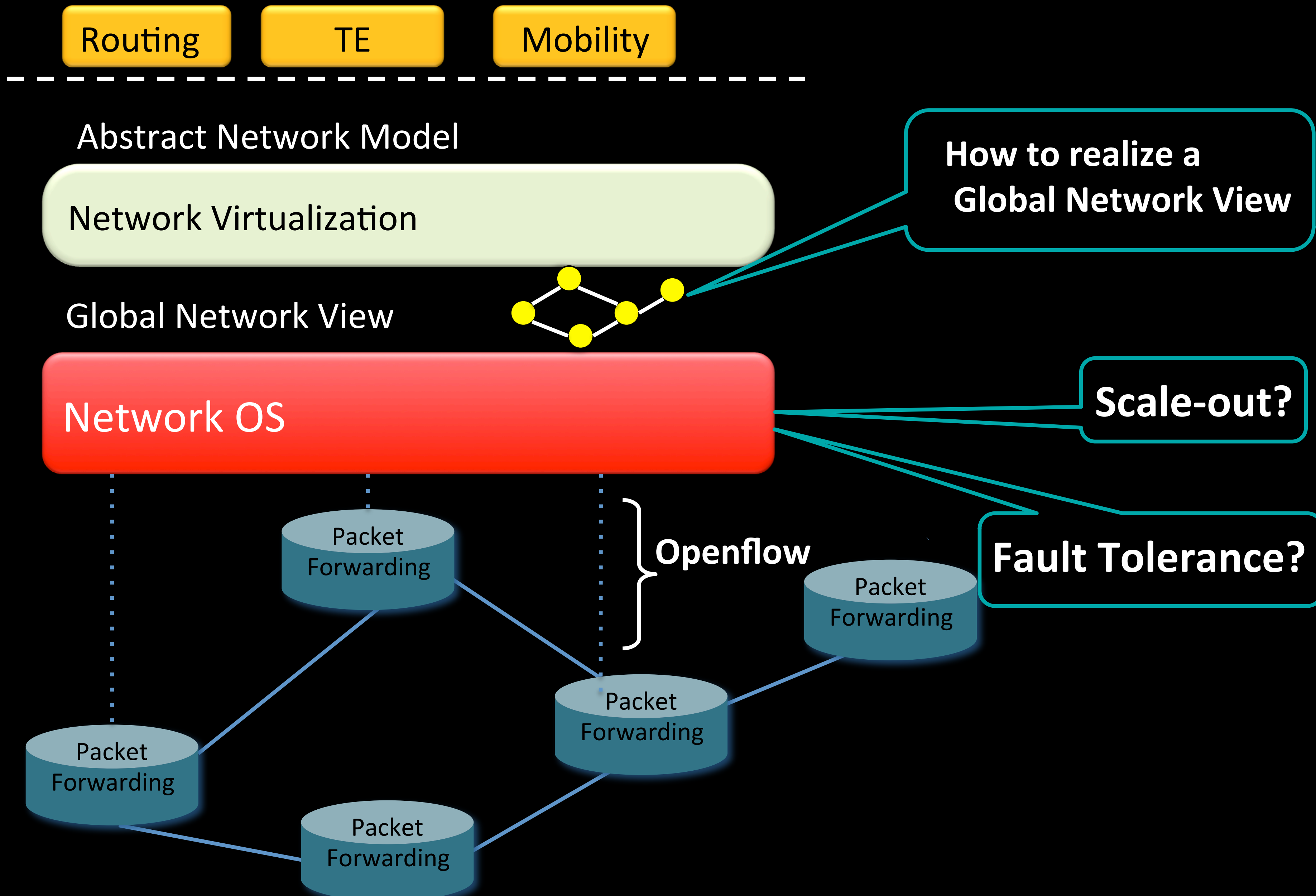
Experimental Open-Source Distributed SDN OS

Software Defined Networking



Scott Shenker, ONS '11

Logically Centralized NOS – Key Questions



Related Work

ONIX

Distributed control platform for large-scale networks

Focus on reliability, scalability, and generality

State distribution primitives, global network view, ONIX API

Other Work

Helios, Hyperflow, Maestro, Kandoo distributed control planes

NOX, POX, Beacon, Floodlight, Trema controllers

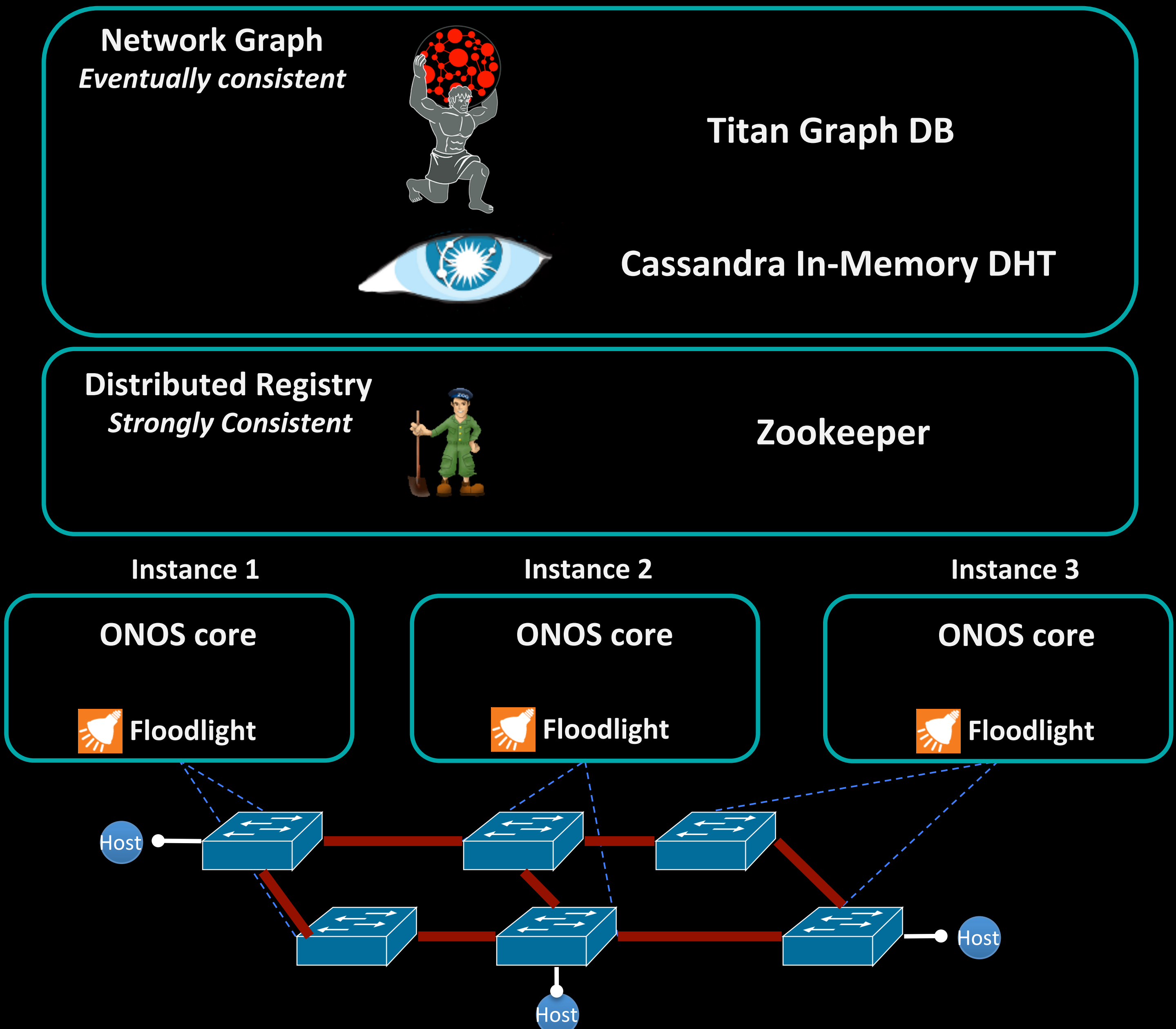
Motivation

- Build an open source distributed NOS
- Learn and share with community
- Target WAN use cases

Phase 1: Goals
December 2012 – April 2013

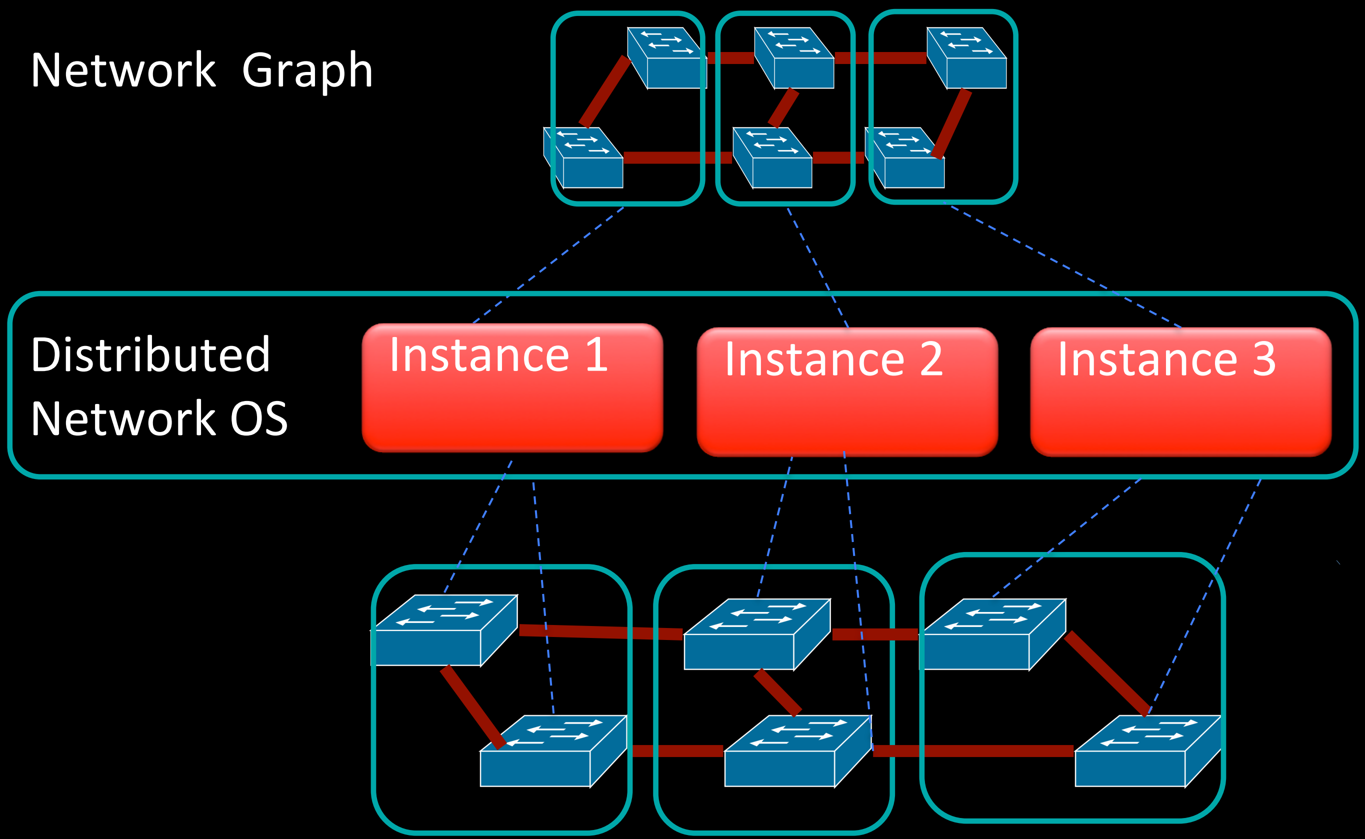
- Demo Key Functionality
  - Fault-Tolerance: Highly Available Control plane
  - Scale-out: Using distributed Architecture
  - Global Network View: Network Graph abstraction
- Non Goals
  - Performance optimization
  - Support for reactive flows
  - Stress testing

ONOS High Level Architecture



How does ONOS scale-out?

ONOS: Scale-out using control isolation

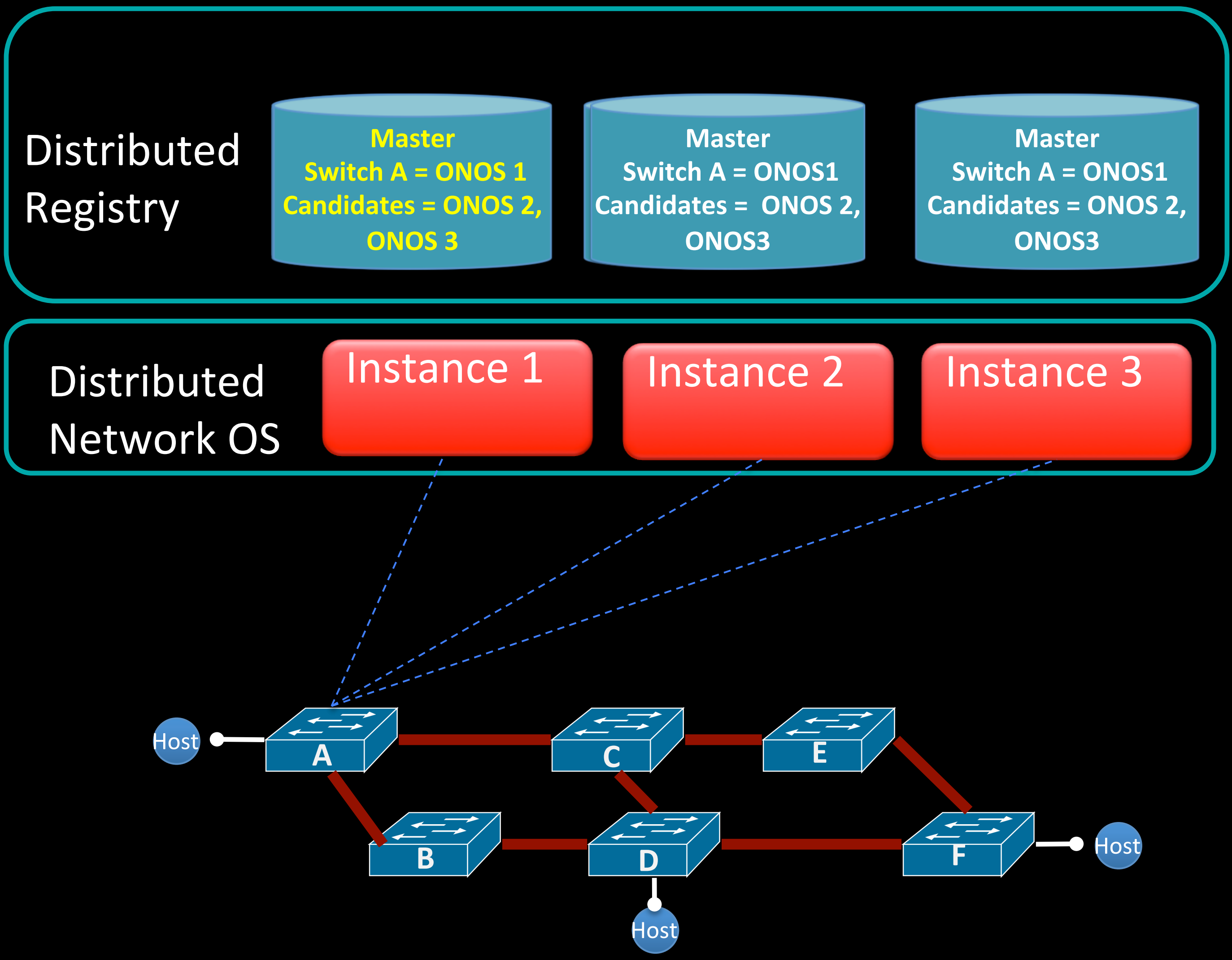


Simple Scale-out Design

- An instance is responsible for building & maintaining a part of network graph
- Control capacity can grow with network size

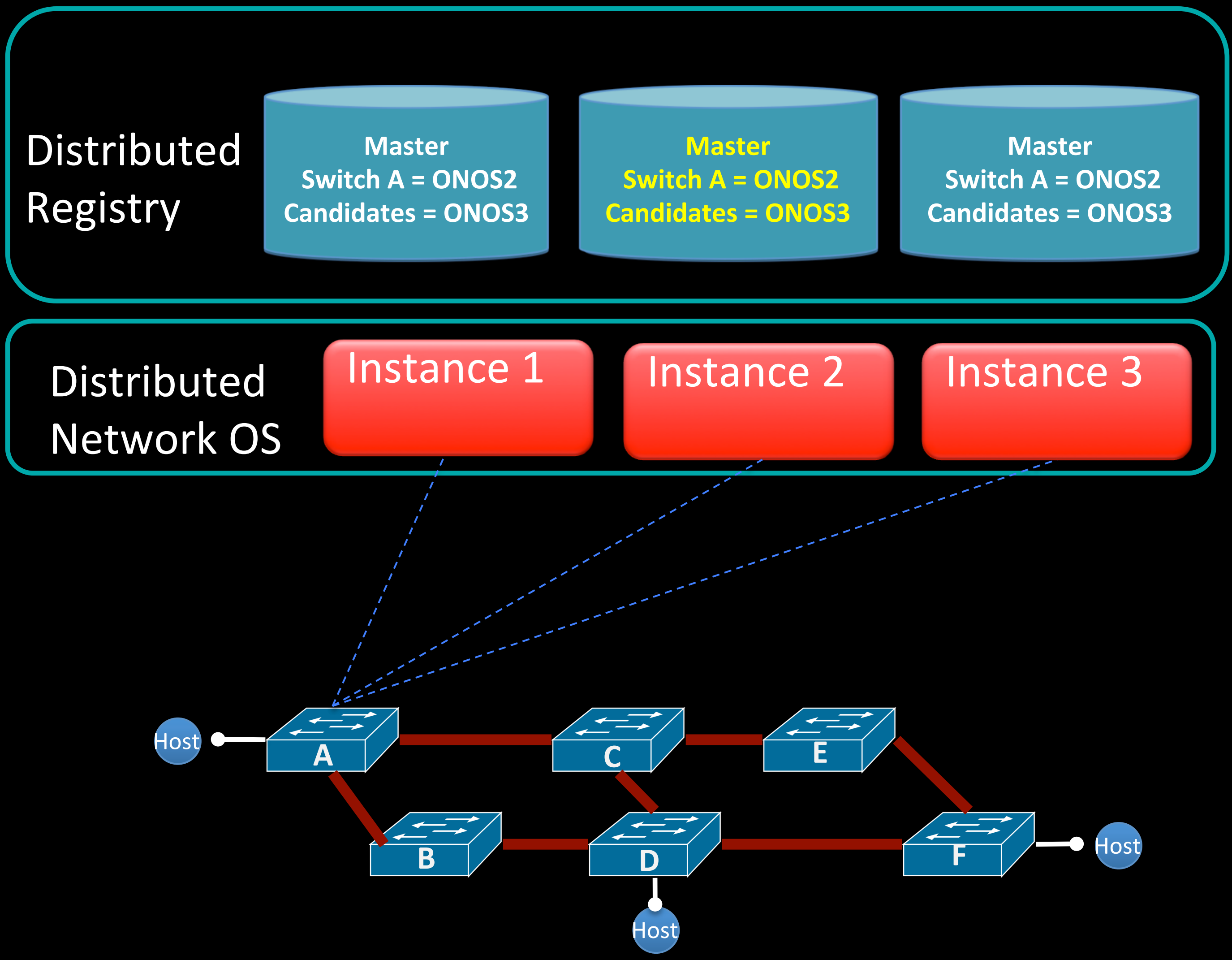
Distributed Registry for Control Isolation

ONOS: Master Election



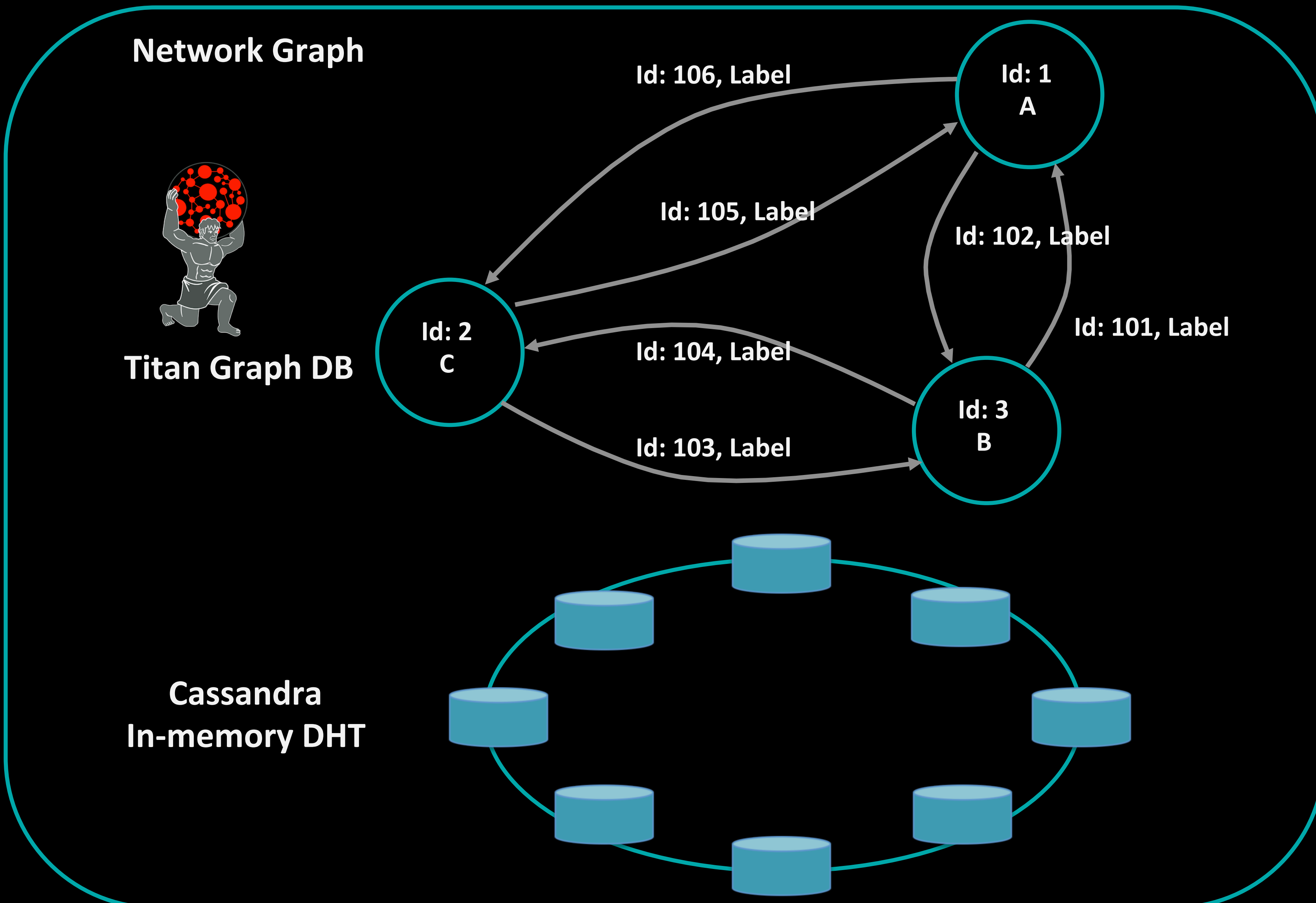
Fault-tolerance using Distributed Registry

ONOS: Control Plane Failover

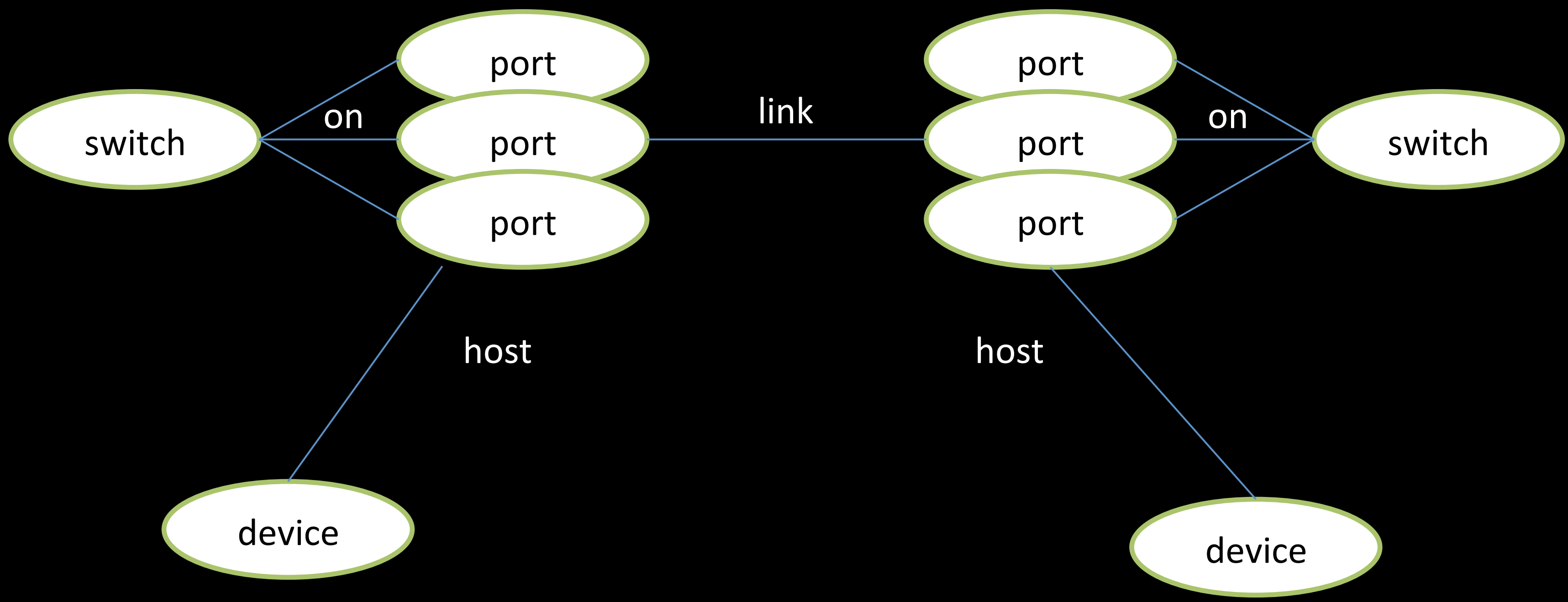


Network Graph to realize global network view

ONOS Network Graph Abstraction

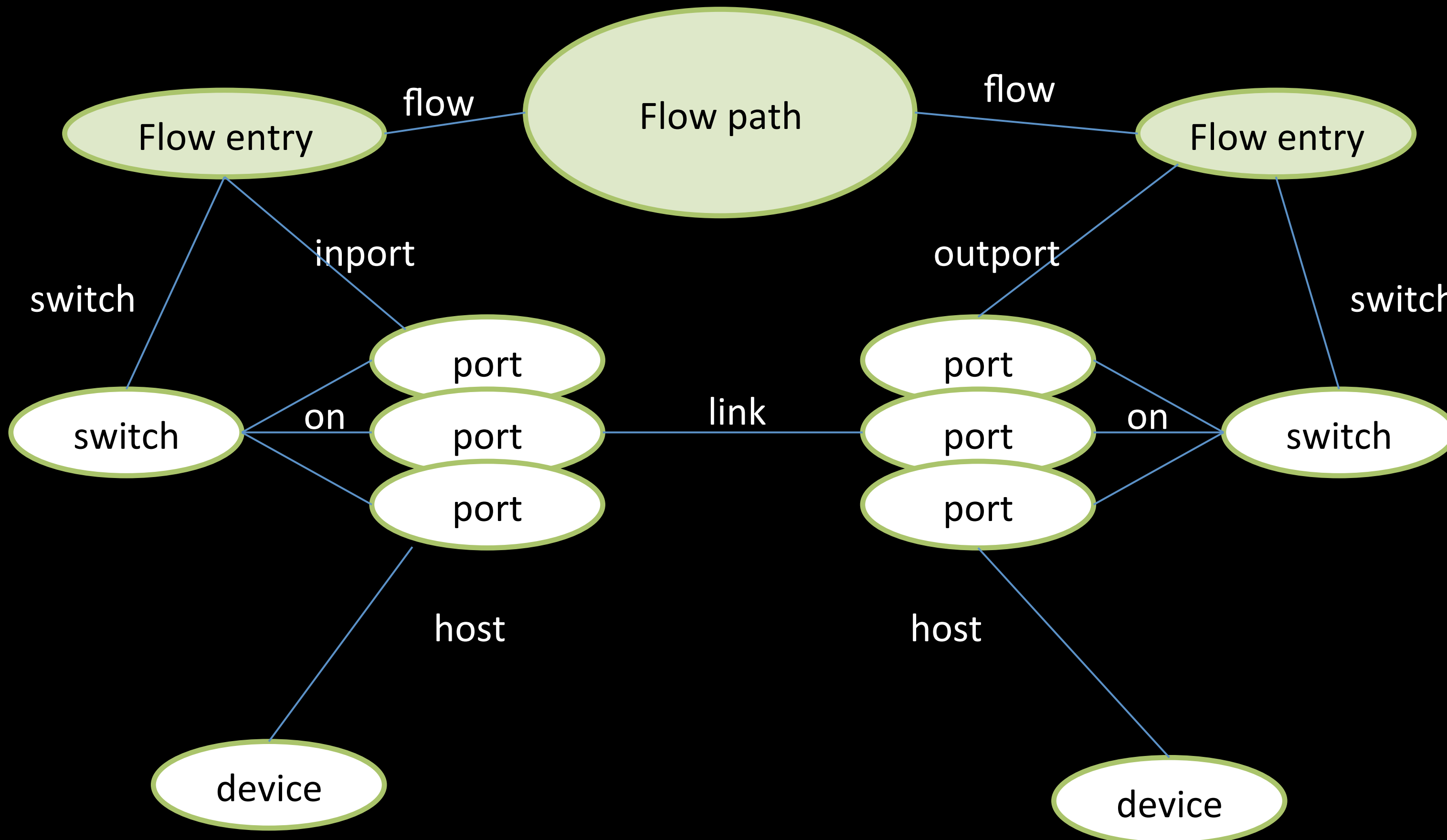


Network Graph



- 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

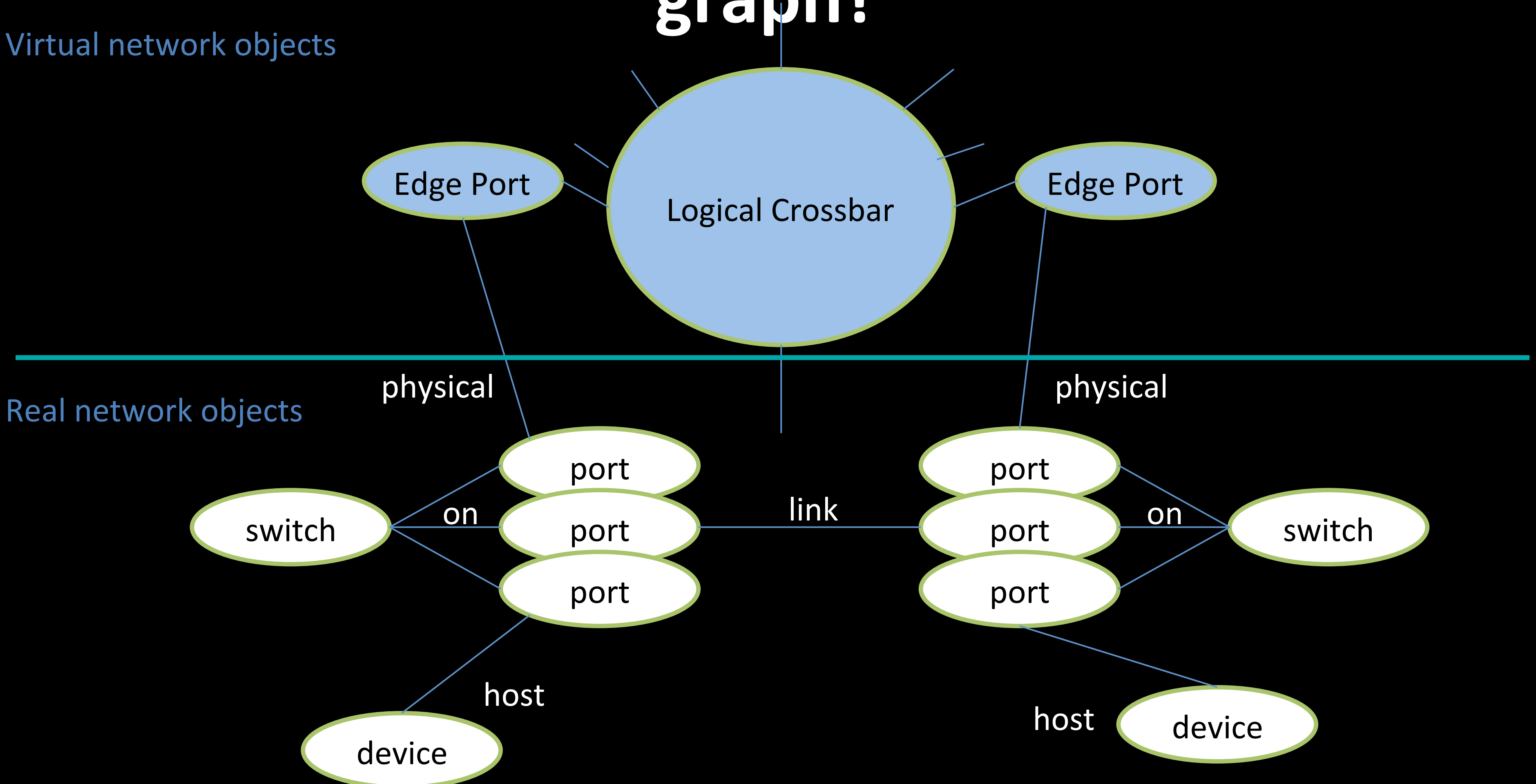
Example: Path Computation App on Network Graph



- **Application computes path by traversing the links from source to destination**
- **Application writes each flow entry for the path**

Thus path computation app does not need to worry about topology maintenance

Example: A simpler abstraction on network graph?



- **App or service on top of ONOS**
- **Maintains mapping from simpler to complex**

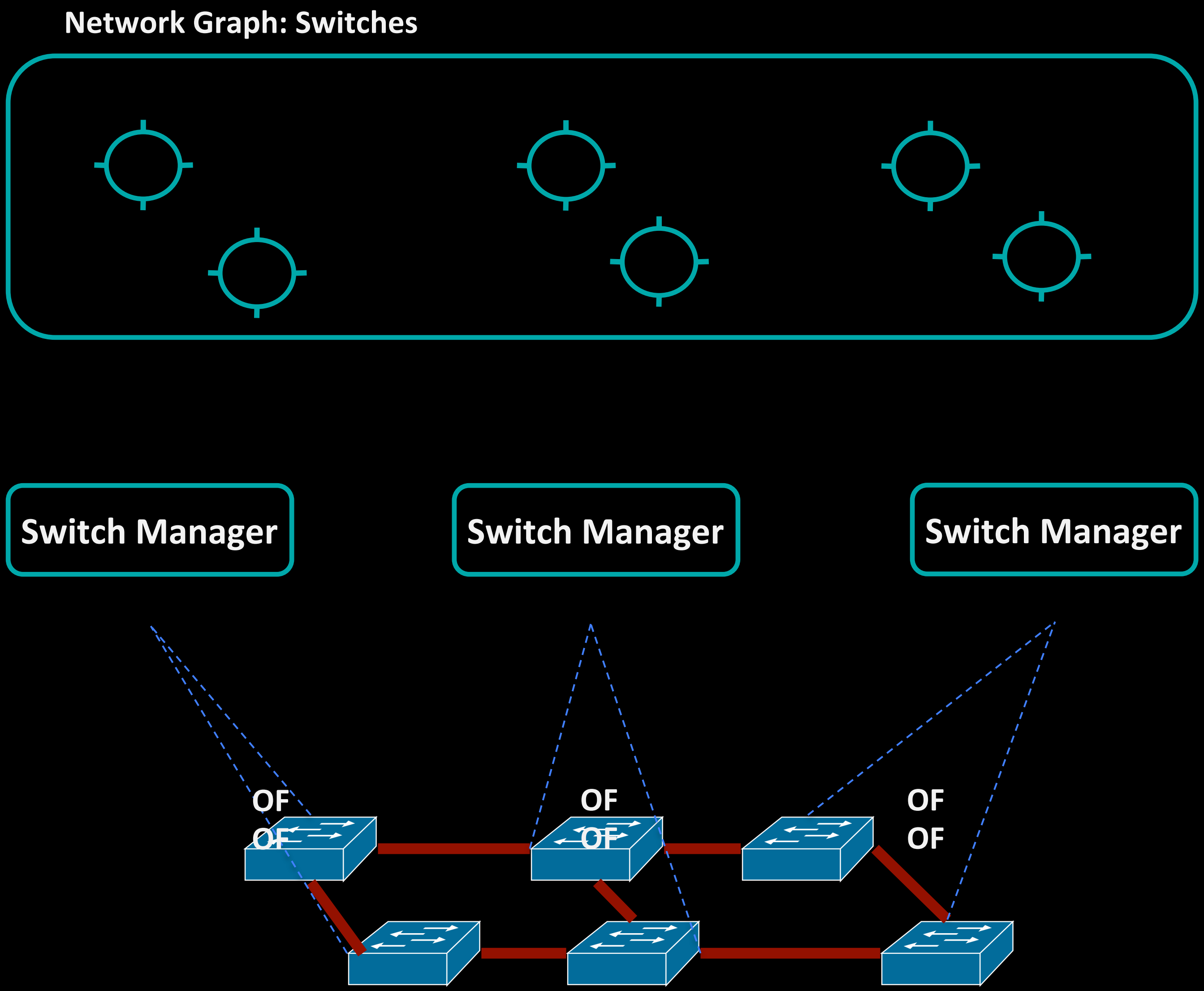
Thus makes applications even simpler and enables new abstractions

Phase 1: Goals
December 2012 – April 2013

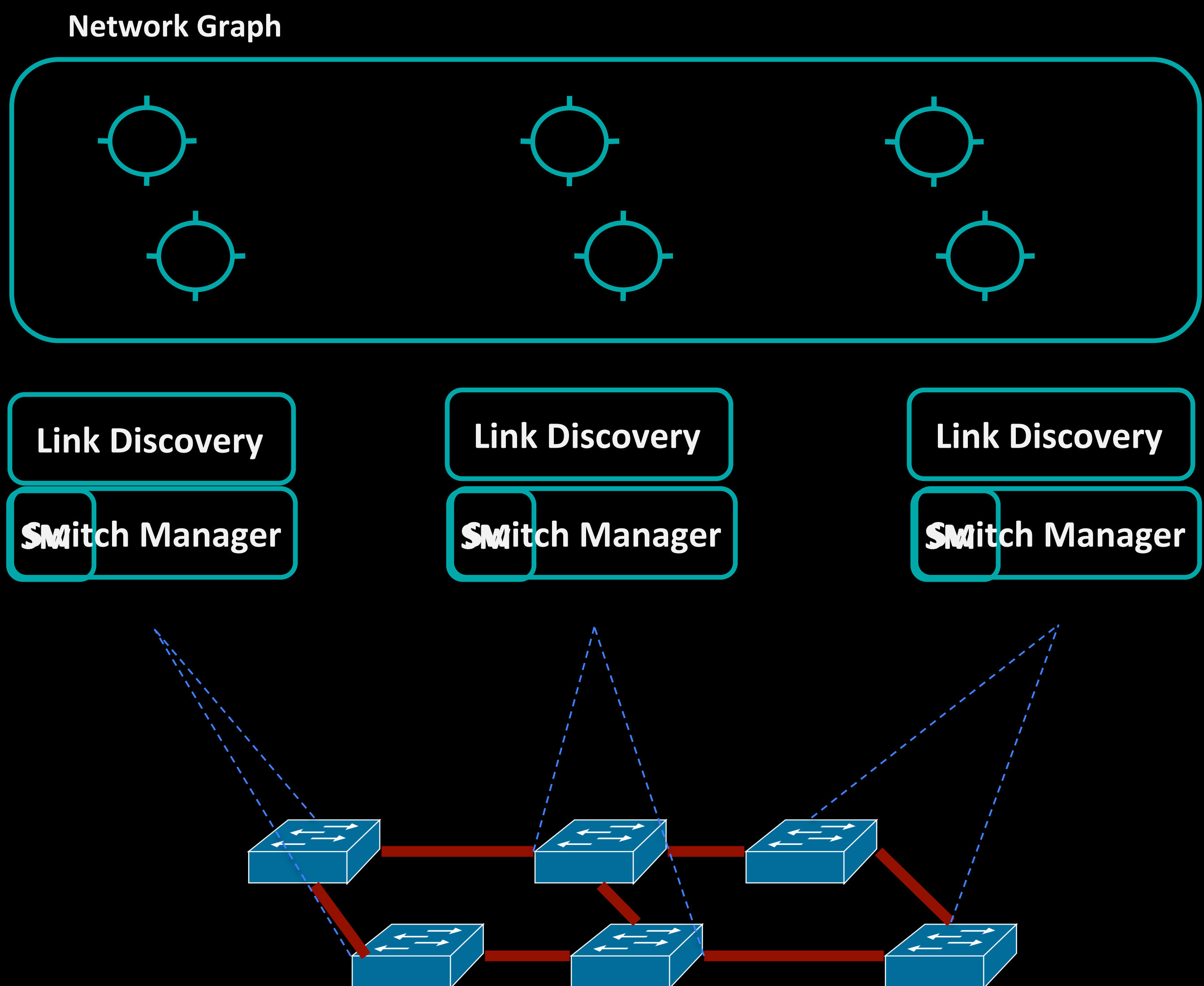
- Demo Key Functionality
  - ✓ Fault-Tolerance: Highly Available Control plane
  - ✓ Scale-out: Using distributed Architecture
  - ✓ Global Network View: Network Graph abstraction

How is Network Graph built and maintained by ONOS?

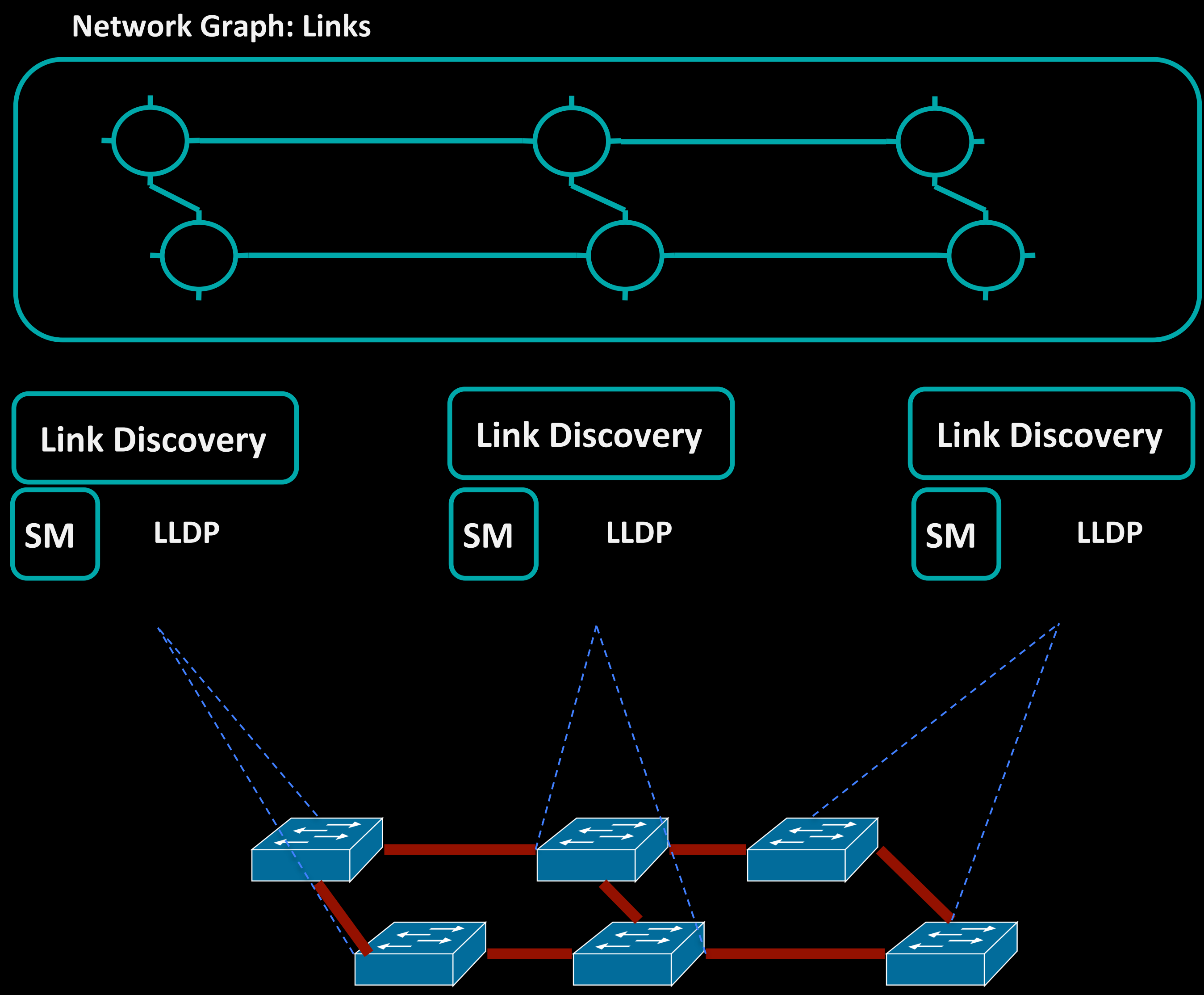
Network Graph and Switches



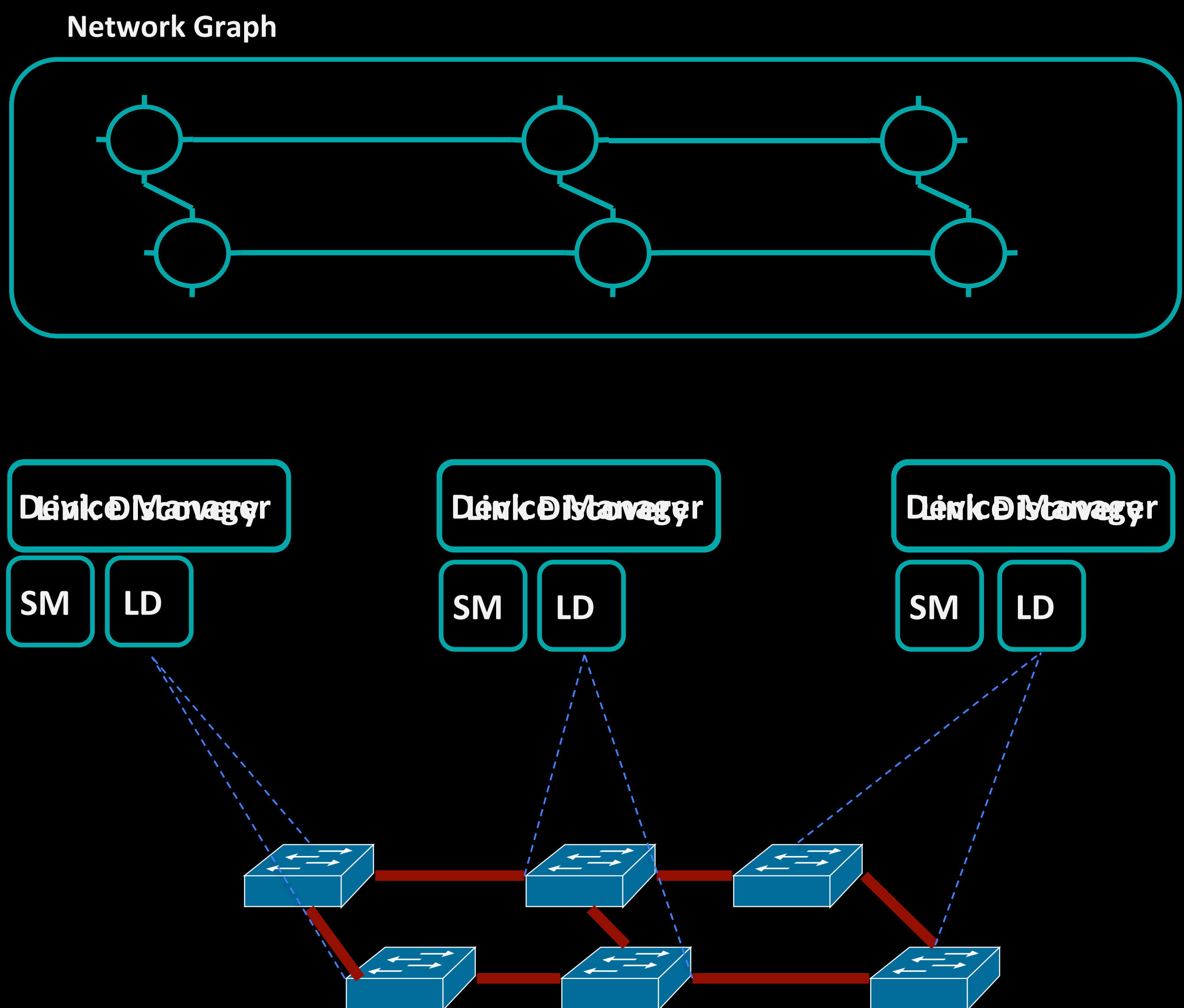
Network Graph and Link Discovery



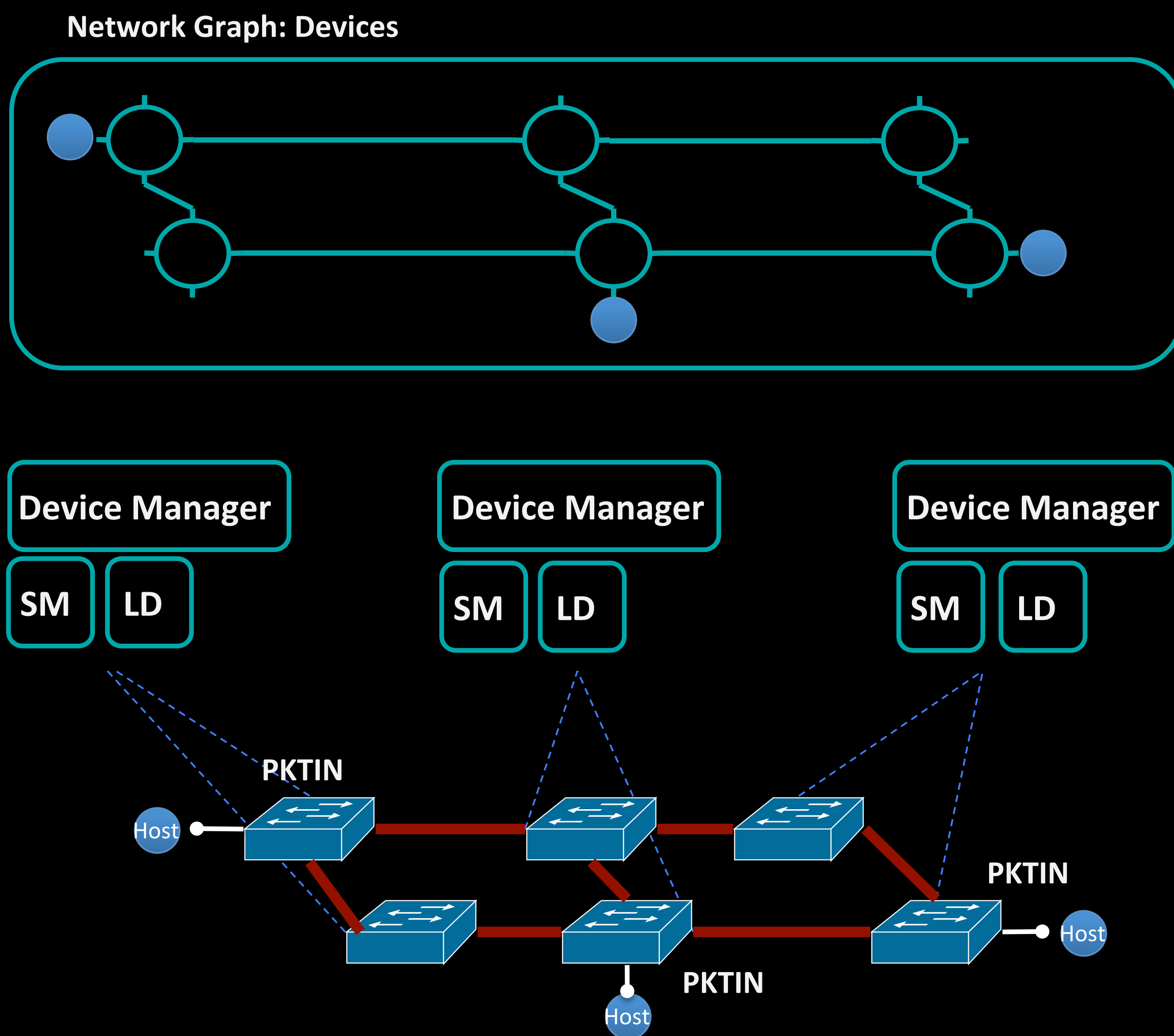
Network Graph and Link Discovery



Devices and Network Graph

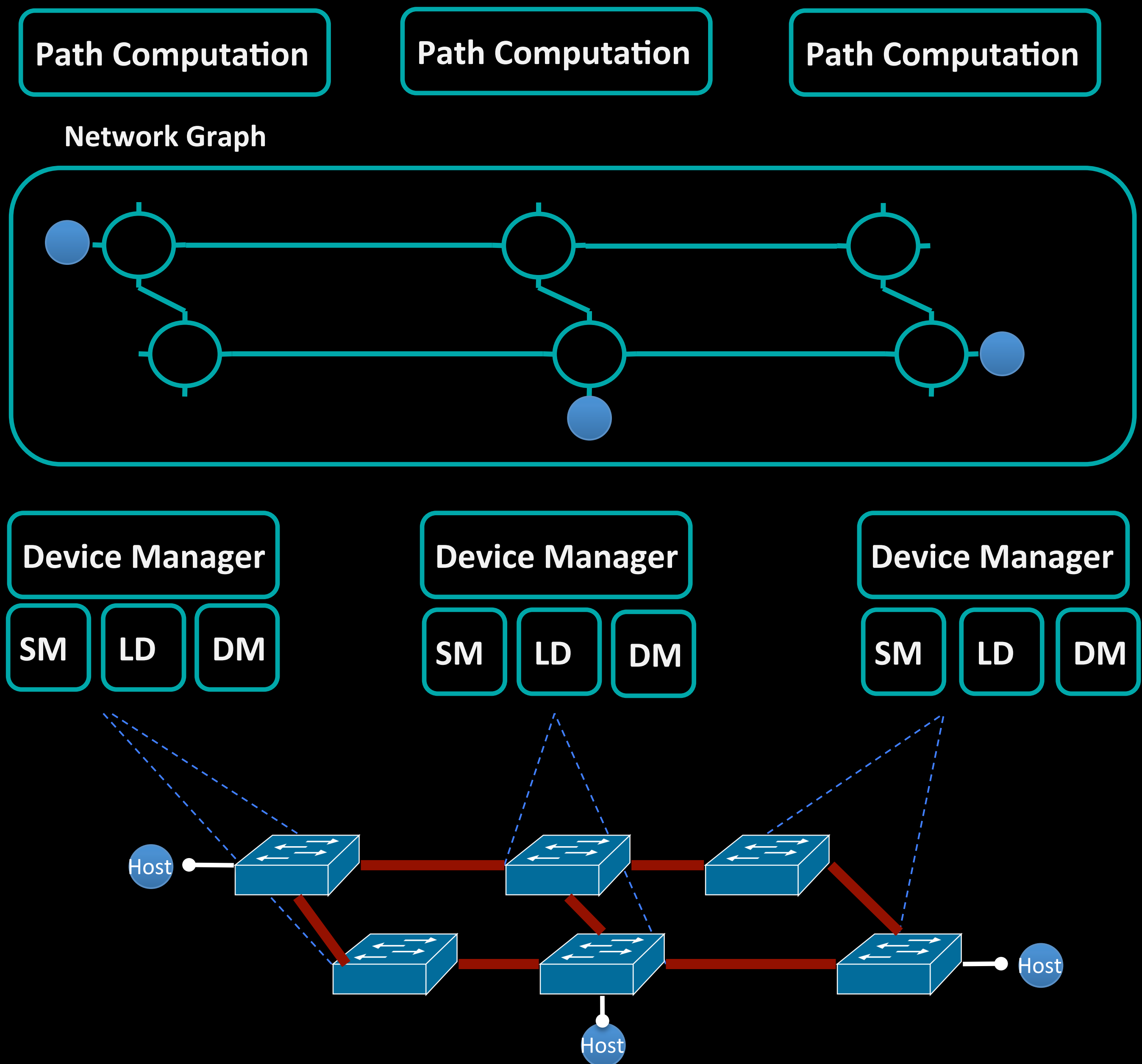


Devices and Network Graph

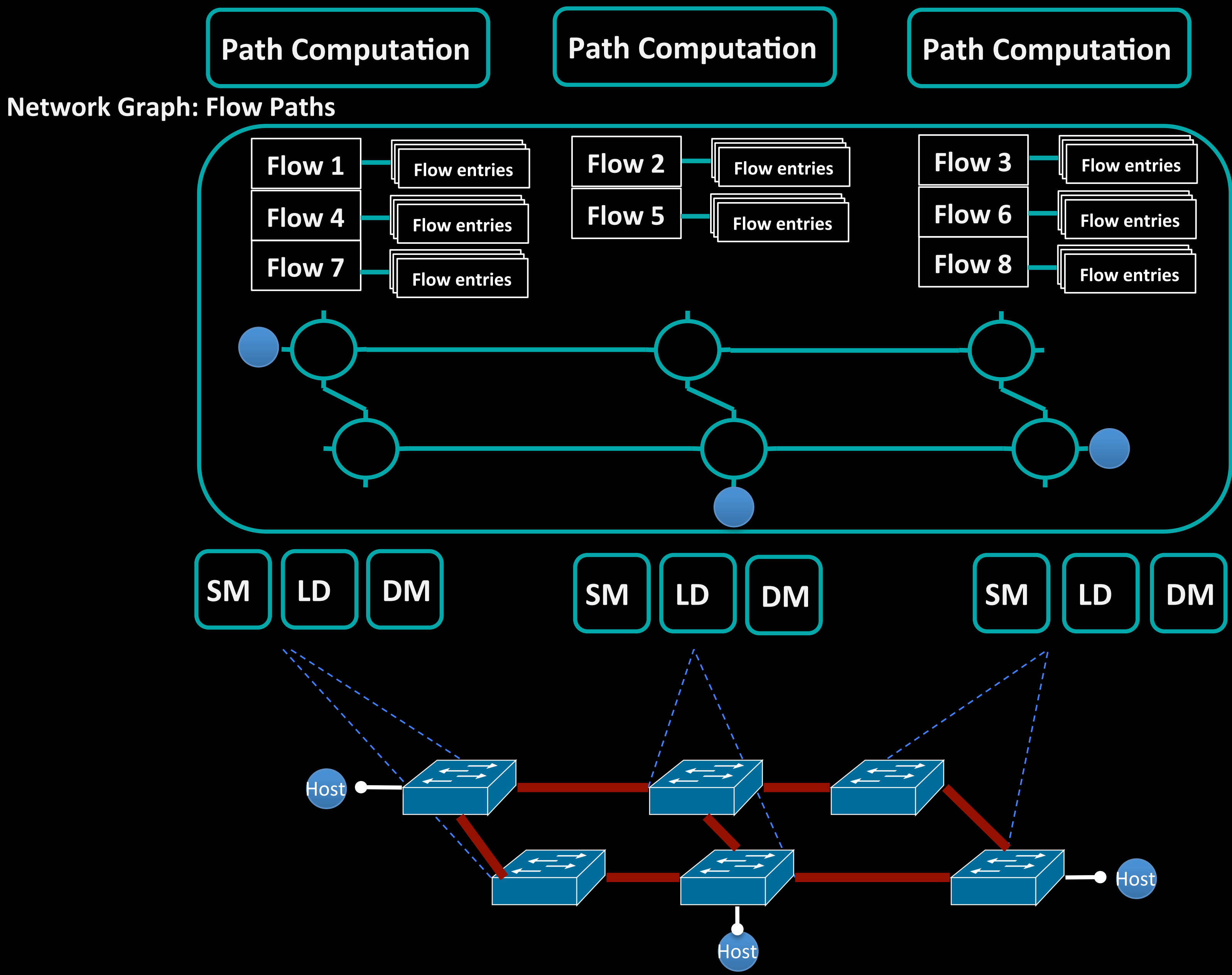


How Applications use Network Graph?

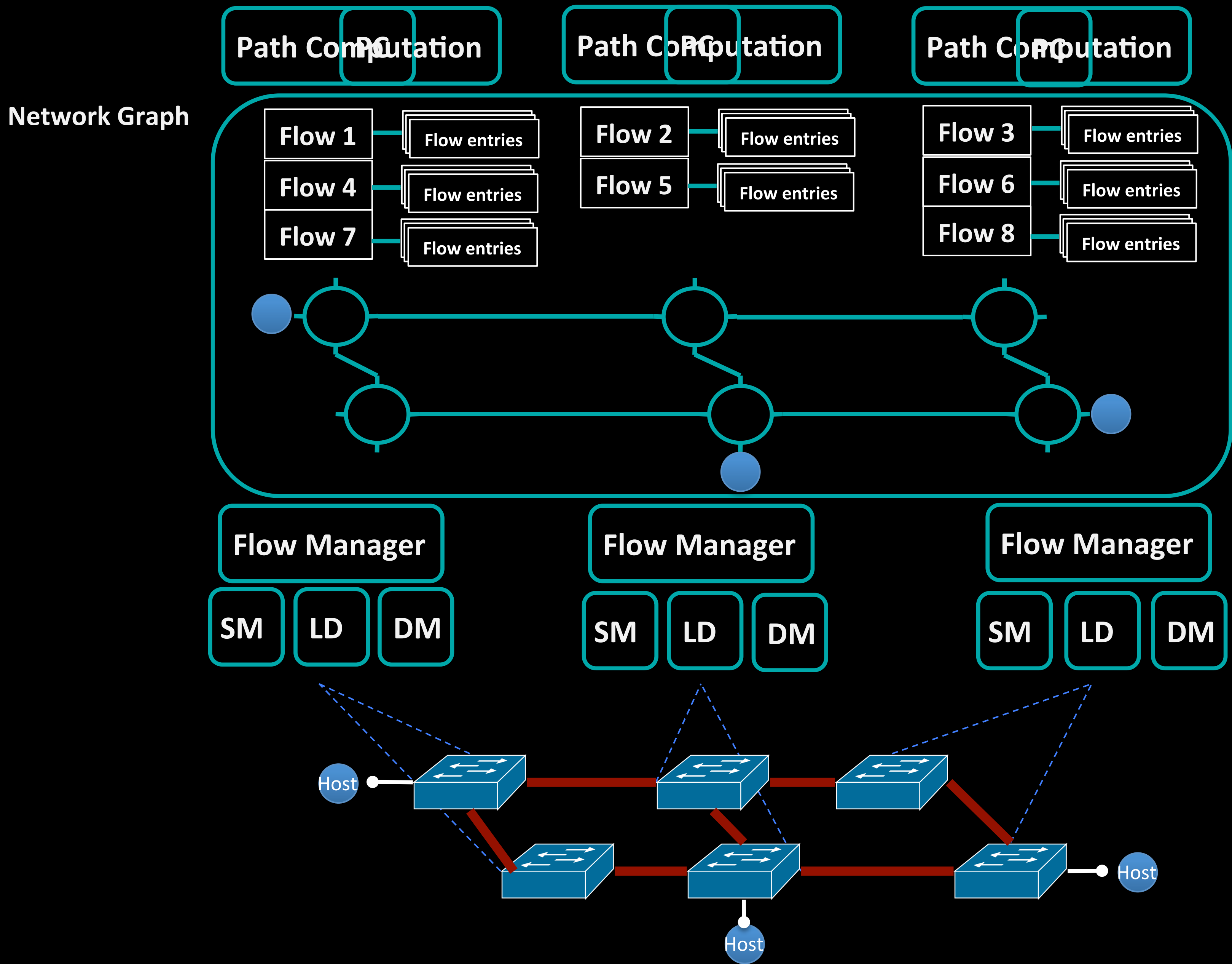
Path Computation with Network Graph



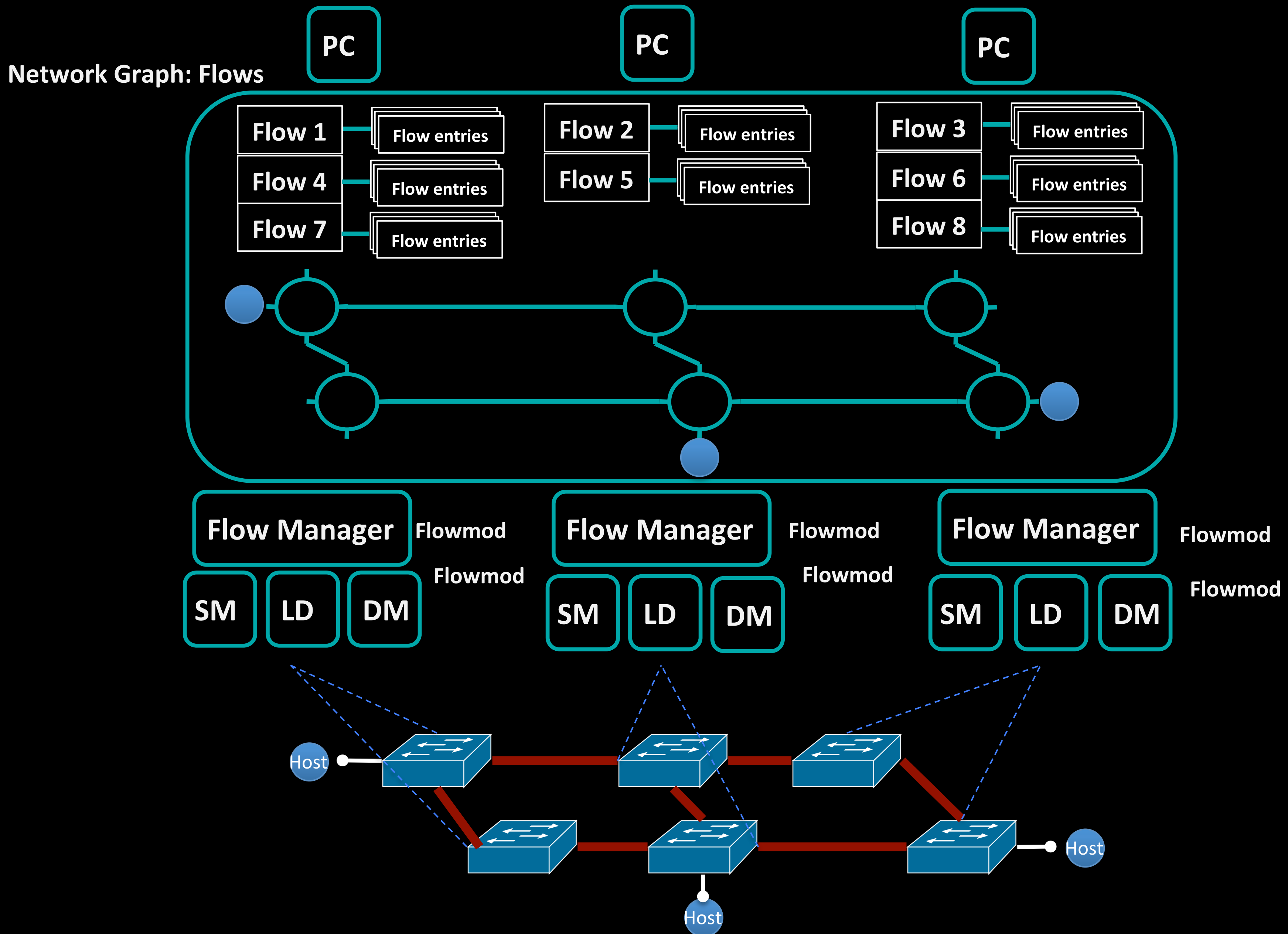
Path Computation with Network Graph



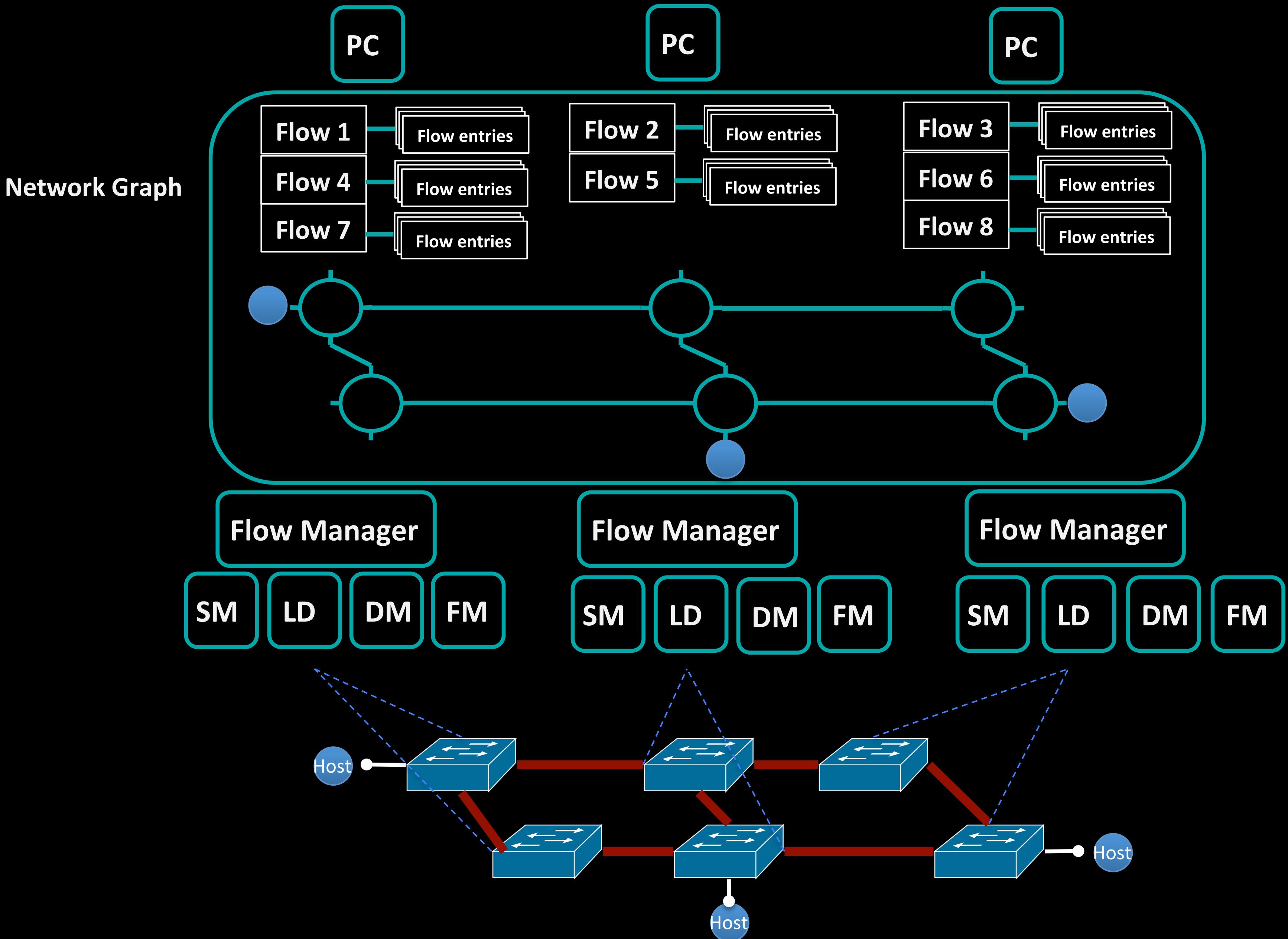
Network Graph and Flow Manager



Network Graph and Flow Manager

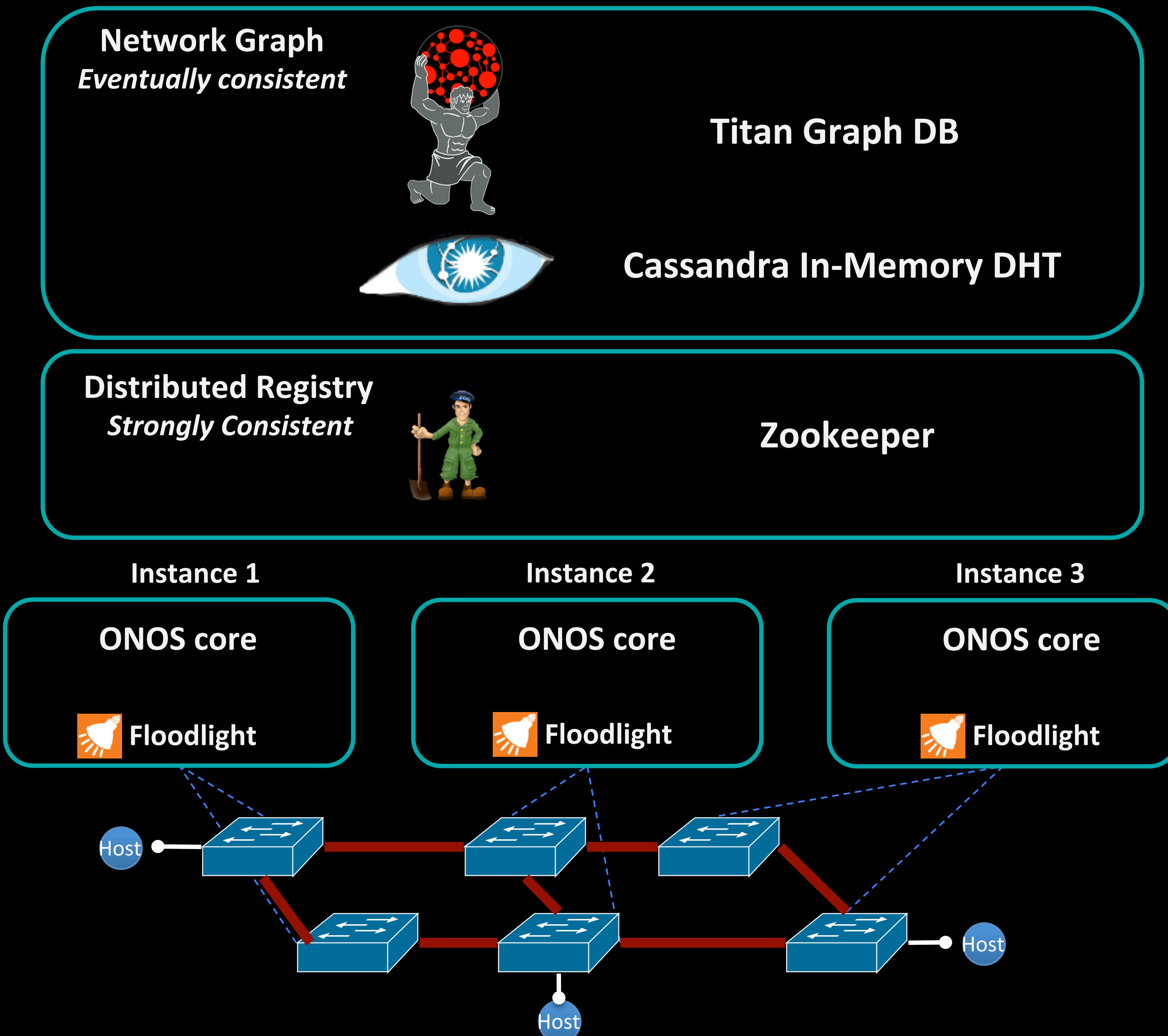


Network Graph and Flow Manager



ONOS uses two different consistency models. Why?

ONOS High Level Architecture



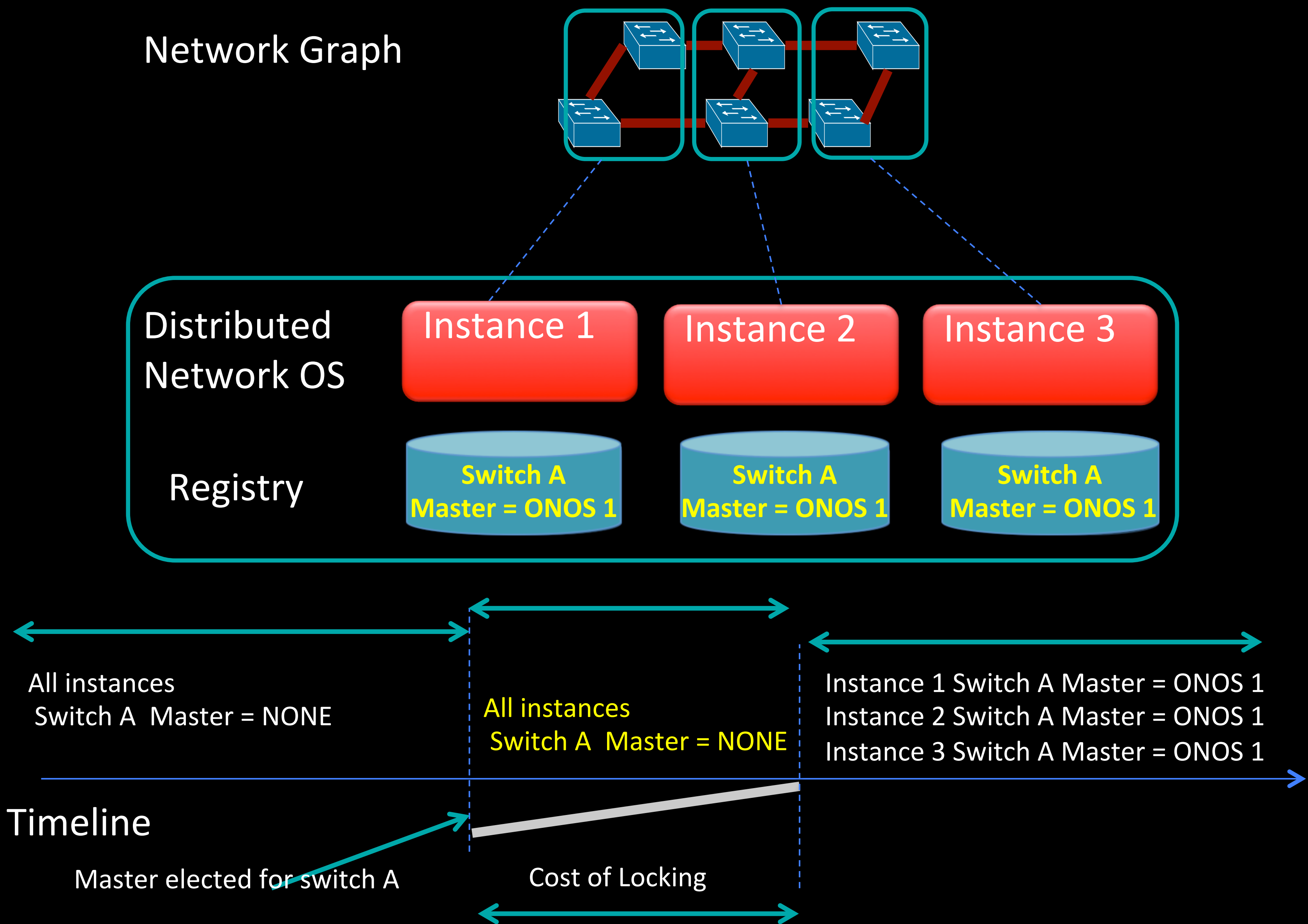
Consistency Definition

- 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

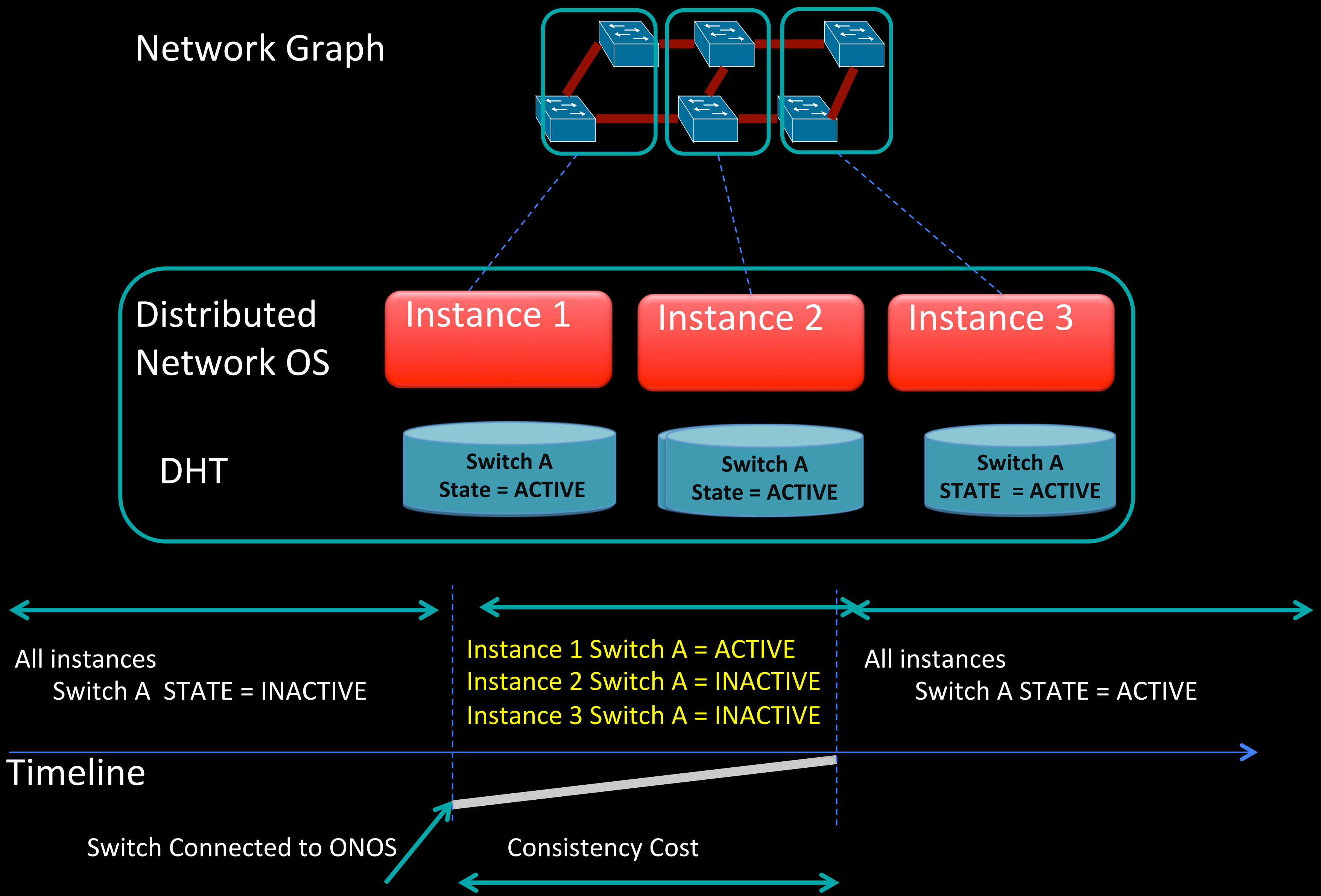
- Weaker consistency might mean Master election on instance 1 will not be available on other instances.

- That can lead to having multiple masters for a switch.

- Multiple Masters will break our semantic of control isolation.

- Strong locking semantic is needed for Master Election

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.

- Eventual consistency becomes more visible during control plane network congestion.

Why is Eventual Consistency good enough for Network State?

- 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

- Strong Consistency is more likely to lead to inaccuracy of network state as network congestions are real.

ONOS High Level Architecture

Network Graph
Eventually consistent

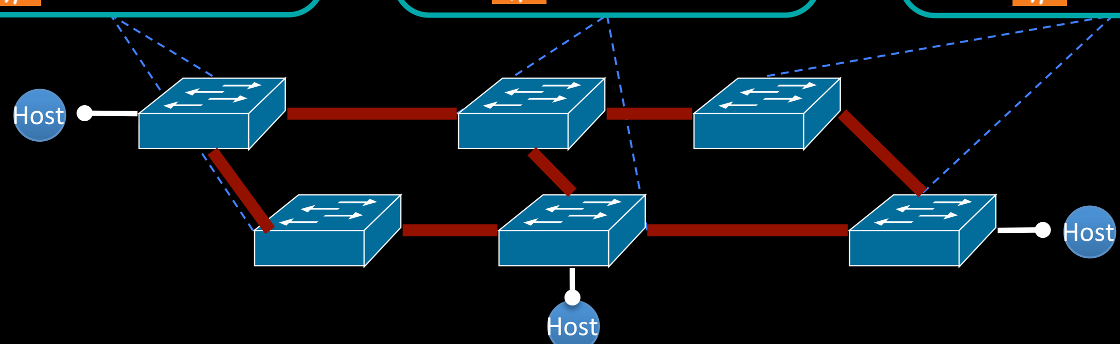
Titan Graph DB

Cassandra In-Memory DHT

Distributed Registry
Strongly Consistent

Zookeeper

Instance 1	Instance 2	Instance 3
ONOS core	ONOS core	ONOS core
Floodlight	Floodlight	Floodlight



ONOS - Questions

- Is graph the right abstraction?
  - Can it scale for reactive flows?
  - What is the Concurrency requirement on graph?
  - Is it large enough to use a NoSQL backend?
  - What about Notifications or publish/subscribe on Graph?

- Are we using the right technologies for Graph?
  - Is DHT a right choice?
  - Cassandra has latencies in order of millisecond – is it ok?
  - What throughput do we need?
  - Titan is good for rapid prototype – is it good enough for production?

- Have we got our Consistency and Partition Tolerance right?
  - What is the latency impact?
  - Should we pick Availability over Consistency?

What is Next for ONOS

ONOS Core

- Performance benchmarks and improvements
- Reactive flows and low-latency forwarding
- Events, callbacks and publish/subscribe API
- Expand graph abstraction for more types of network state

ONOS Apps

- ONOS Northbound API
- Service chaining
- Network monitoring, analytics and debugging framework

Community

- Release as open source and/or contribute to Open DayLight.
- Build and assist developer community outside ON.LAB
- Support deployments in R&E networks

ON.LAB

www.onlab.us