ONOS Code Walkthrough #1

James Won-Ki Hong, Jian Li, Seyeon Jeong

Dept. of Computer Science & Engineering
POSTECH

http://dpnm.postech.ac.kr/~jwkhong
jwkhong@postech.ac.kr

Outline

- ONOS Application Structure
- BYON Application
  - Requirement
  - Development environment
  - Lab sections

ONOS Tutorial Sessions

- **Overview & Setup**
  - ONOS overview, description of BYON app
  - Run-time environment & development setup, initial app deployment
- **Controlling Network via Intents**
  - Enhance `NetworkManager` to use `IntentService` to control connectivity
  - Implement a CLI command
- **Distributed Store Component**
  - Implement `DistributedNetworkStore` component

ONOS Applications

- **Application as a mere Component**
  - offers no API, self-contained, e.g. reactive forwarding, proxy ARP
  - generally interacts only with the network environment
- **Application with Service Interface**
  - offers API; for other Apps, CLI, REST or GUI
  - interacts with network environment, but also other software entities (hence API)
- **Applications may have their own state; use Store pattern**
  - delegates responsibility for tracking state to a separate component

OSGi Bundles & Karaf Features

- **OSGi Bundles are Java JAR Files with an Enhanced Manifest**
  - bundles have name and version
  - bundles explicitly require/import other Java packages
  - bundles explicit provide/export Java packages for others
- **Karaf Features Are Means to Install or Uninstall a Set of Bundles as a Group**
  - features are defined via an XML artifact - a feature repository
  - feature references, but does not deliver the bundle JAR artifacts
- **Karaf uses Maven Repos as OSGi Bundle Repositories for Retrieval of Feature and Bundle Artifacts**

Service Component Runtime (SCR)

❖ SCR

- Components are effectively stateful singletons whose life-cycle is controlled by the framework
  - components defined by `OSGI-INF/*.xml` files at run-time
  - ONOS uses `maven-scr-plugin` to convert Java annotations to `OSGI-INF/*.xml` files at compile-time
- Components can provide `@Services` to others
- Components can `@Reference` services from others
- `@Activate`, `@Modified` and `@Deactivate` methods serve as component life-cycle hooks

Bundle & Feature Shell Commands

❖ Karaf Built-in Commands

- Bundle related commands
  - `onos> bundle:*`
- Feature related commands
  - `onos> feature:*`
- Service component runtime related commands
  - `onos> scr:*`

Developing ONOS Apps

❖ ONOS Applications

- Maven archetypes
  - `onos-api-archetype` - basis for a app Java API bundle
  - `onos-bundle-archetype` - basis for an ONOS bundle or an app
  - `onos-cli-archetype` - overlay for apps with CLI extensions
  - `onos-ui-archetype` - overlay for apps with GUI extensions
  - `onos-uitab-archetype` - overlay for apps with GUI table views
  - `onos-uitopo-archetype` - overlay for apps with GUI topo overlays
- Run `mvn archetype:generate` to create a working minimal project module
- For simpler usage run `onos-create-app` shell tool

Bundles, Features & ONOS Apps

❖ Details on ONOS Apps

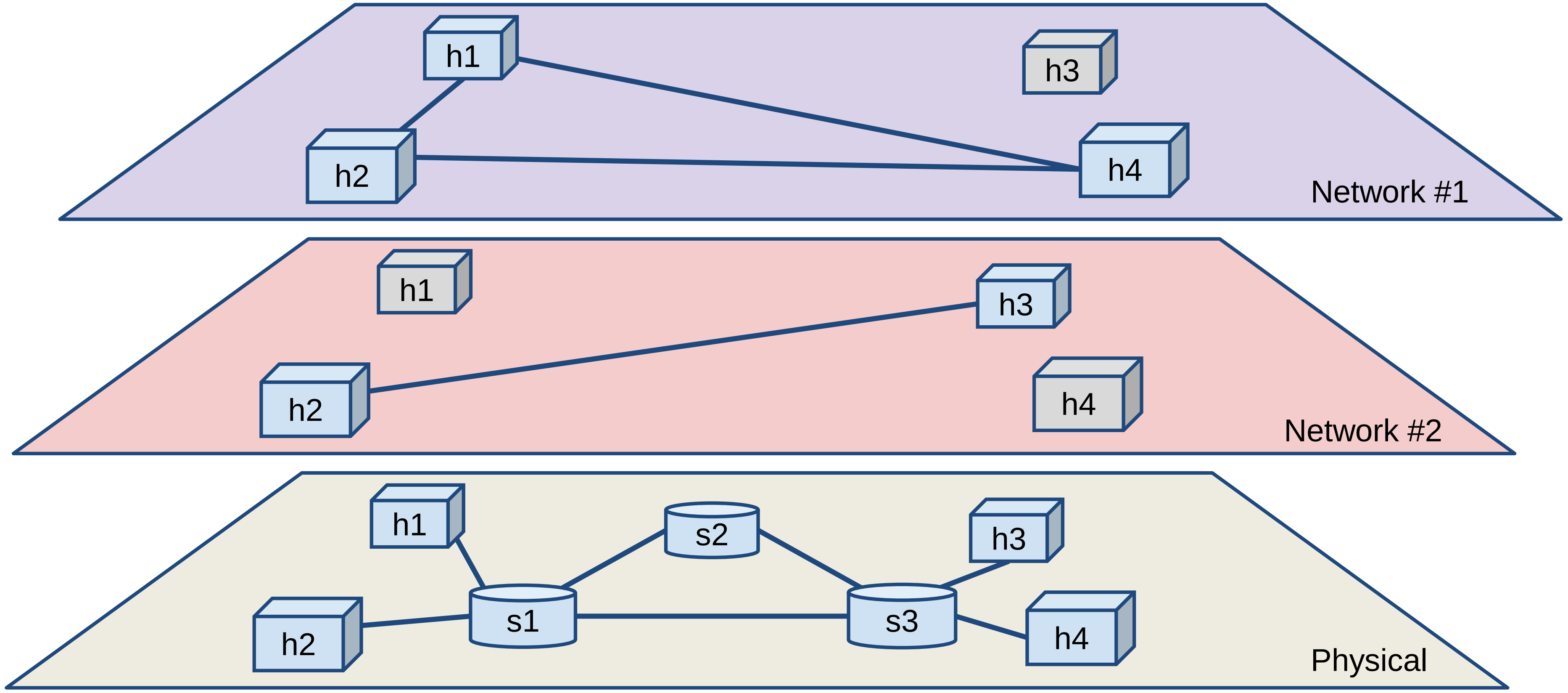
- Apps are delivered via ONOS App aRchive (`.oar`) files
  - OAR is a JAR with `app.xml`, `features.xml` and bundle artifacts
  - `onos-maven-plugin` generates an `*.oar` file as part of Maven build
- Apps are managed on the entire ONOS cluster
  - via REST API: `GET|POST|DELETE /onos/v1/applications`
  - via shell tool: `onos-app {install|activate|deactivate|uninstall}`
  - via CLI: `onos:app {install|activate|deactivate|uninstall}`
  - via GUI
- Back-end installation and activation is done via normal feature & bundle services

BYON Application

❖ Bring Your Own Network (BYON)

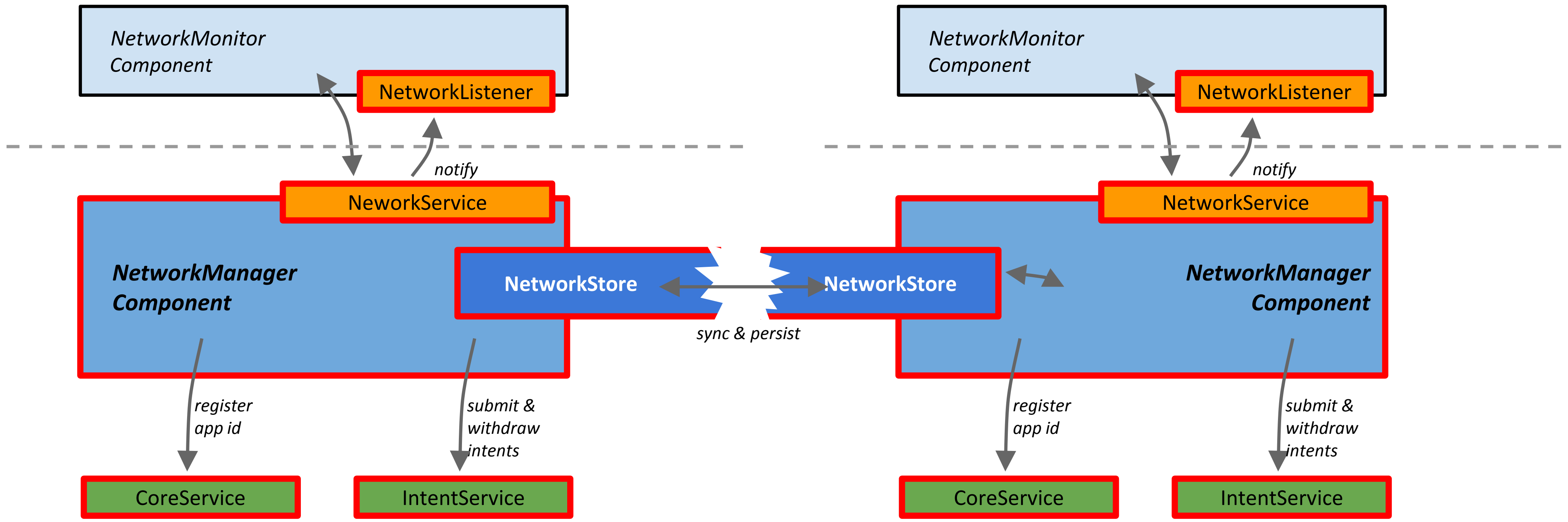
- BYON is a service which allows you to spawn virtual networks
  - All hosts in the virtual networks are interconnected through a full mesh
- Each virtual network contains a full mesh of the hosts within it
- BYON allows users to interact with it through CLI commands
  - In particular, `list-networks` is a CLI command that you will use in this part
  - Other available CLI commands are:
    - `create-network` - provided
    - `add-host` - provided
    - `remove-host` - to be implemented
    - `remove-network` - to be implemented

BYON Application Example

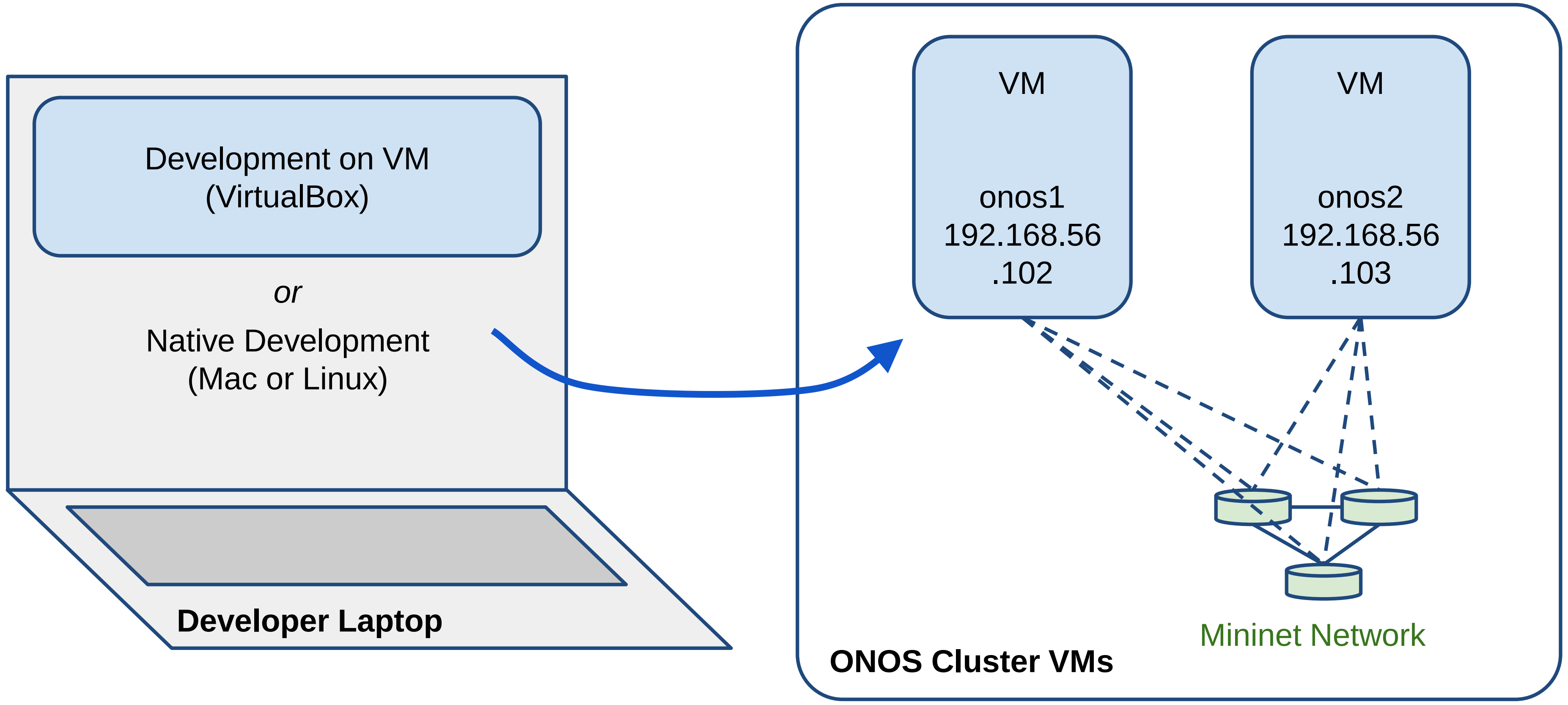


BYON App Structure

- ❖ **Follow the ONOS Architecture**



Environment Overview


Development on VM
(VirtualBox)
or
Native Development
(Mac or Linux)
Developer Laptop
VM
onos1
192.168.56
.102
VM
onos2
192.168.56
.103
Mininet Network
ONOS Cluster VMs