ONOS Distributed Tutorial

for ONOS 1.5.0 (Falcon)

#ONOSProject

Open Network Operating System

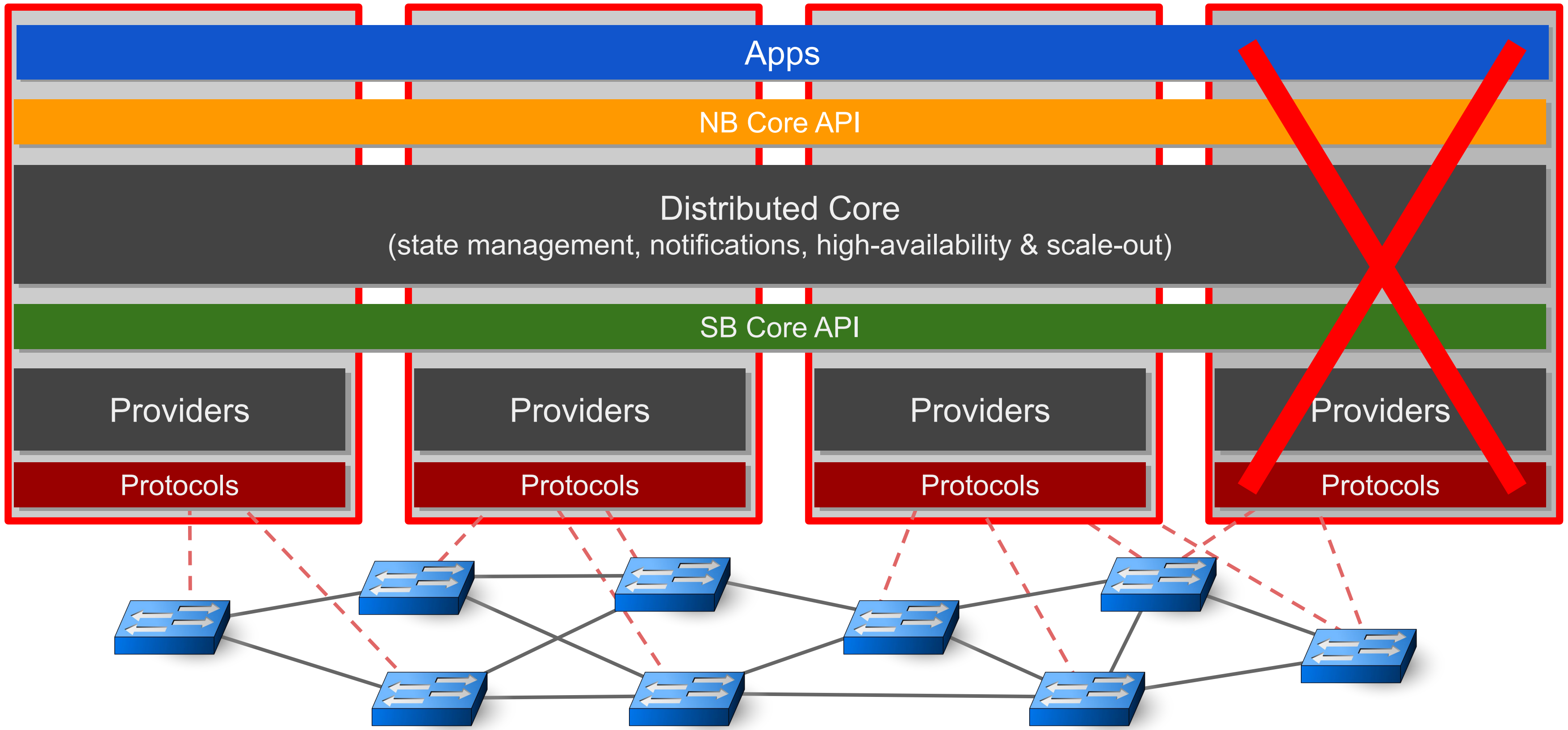
ONOS Tutorial Sessions

- 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
- Events & Monitoring
  - enhance core components for event dispatching
  - implement `NetworkMonitor` component to intercept events

ONOS Architecture Tenets

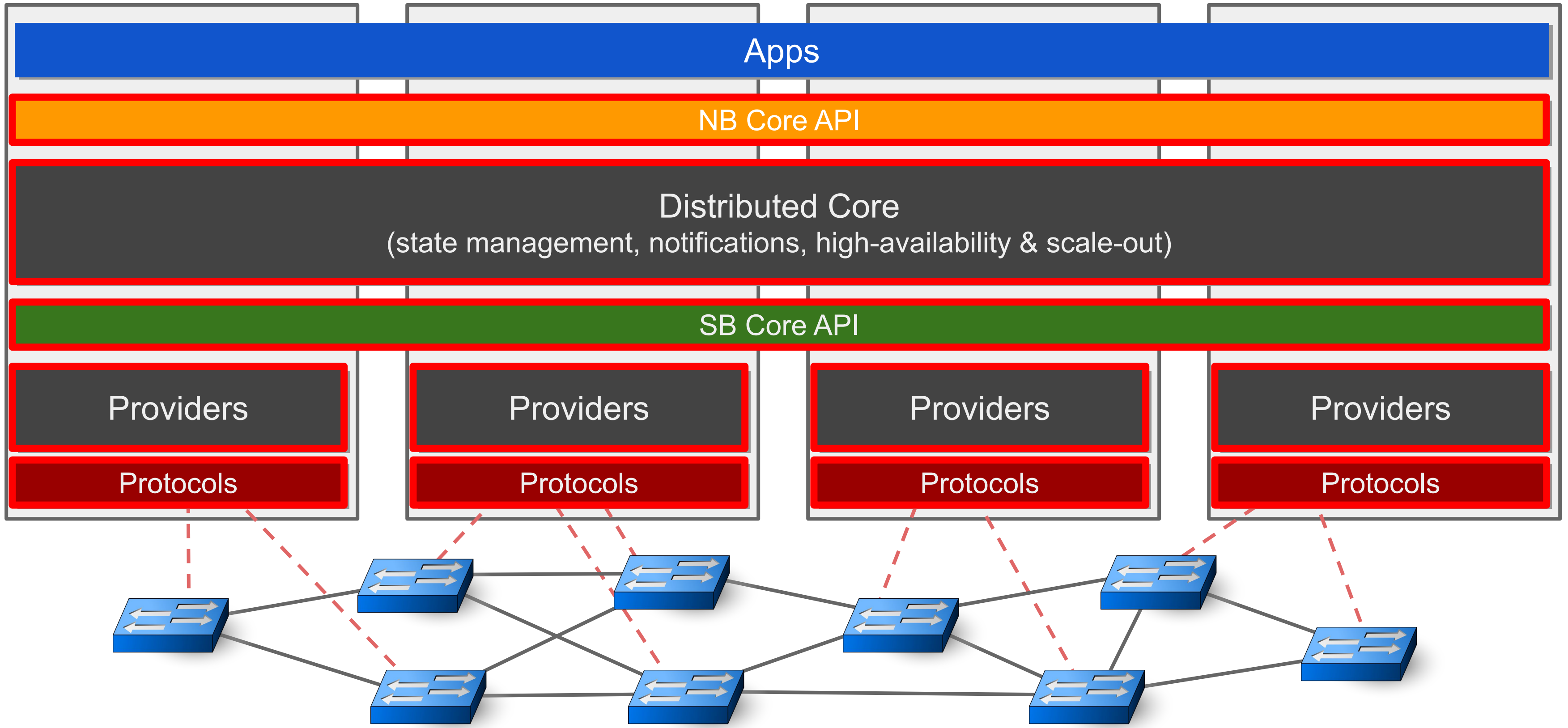
- High-availability, scalability and performance
    - required to sustain demands of service provider & enterprise networks
- Strong abstractions and simplicity
    - required for development of apps and solutions
- Protocol and device behaviour independence
    - avoid contouring and deformation due to protocol specifics
- Separation of concerns and modularity
    - allow tailoring and customization without speciating the code-base

ONOS Distributed Architecture

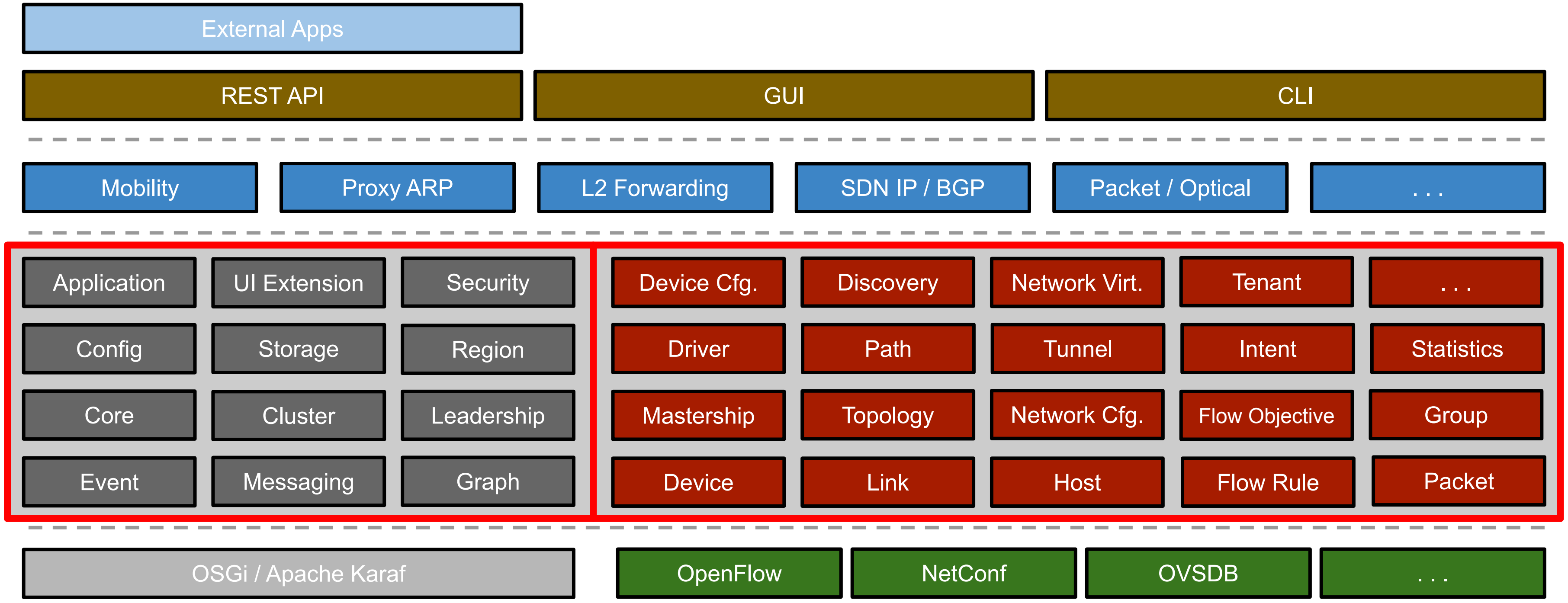


#ONOSProject

ONOS Distributed Architecture

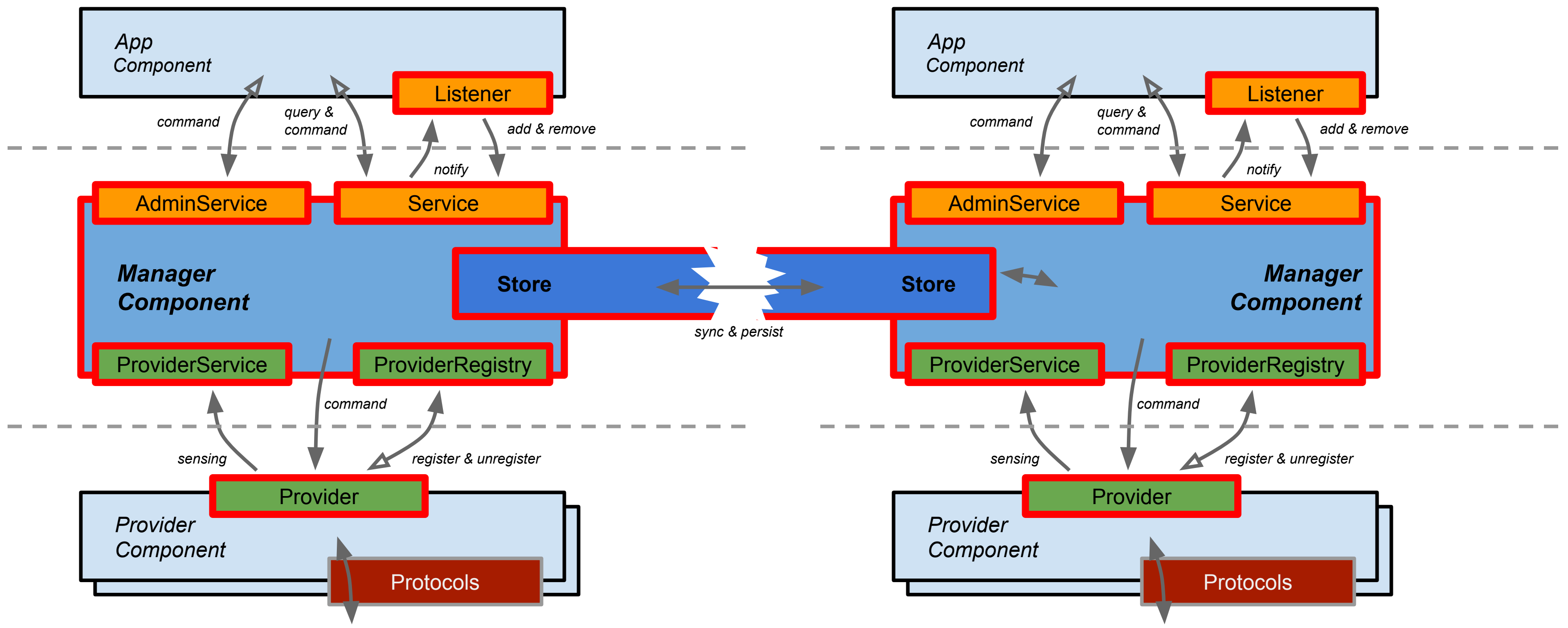


ONOS Core Subsystems



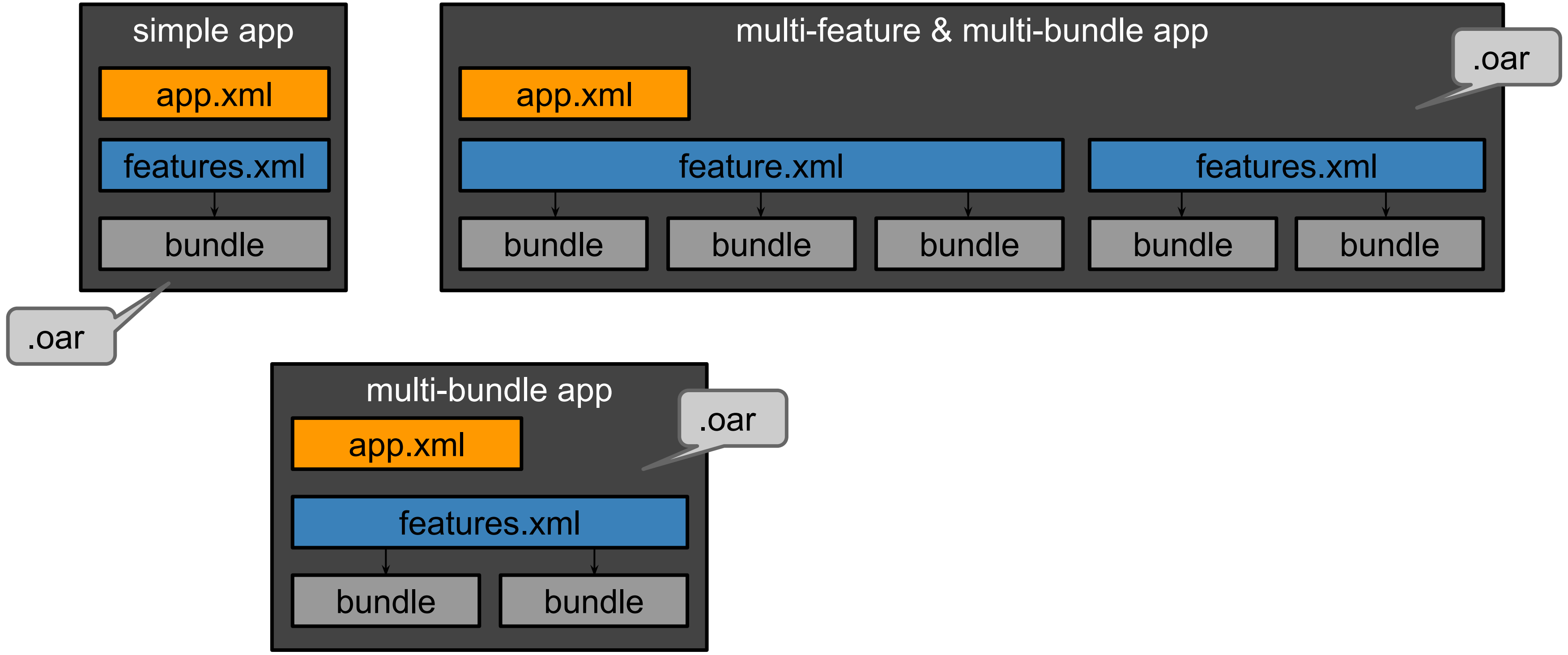
#ONOSProject

ONOS Core Subsystem Structure



#ONOSProject

ONOS Applications & OSGi



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)
- Application ignited as “service component”
  - “singleton”, with activate/deactivate/modify methods
  - ignited by OSGi service component run-time (SCR)
  - dependencies on other services auto-wired by OSGi SCR
- 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

- 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 `@Service`s to others
- Components can `@Reference` services from others
- `@Activate`, `@Modified` and `@Deactivate` methods serve as component life-cycle hooks

Bundle & Feature Shell Commands

- Bundle related commands

```
onos> bundle:*
```

- Feature related commands

```
onos> feature:*
```

- Service Component Runtime related commands

```
onos> scr:*
```

Developing ONOS apps

- 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

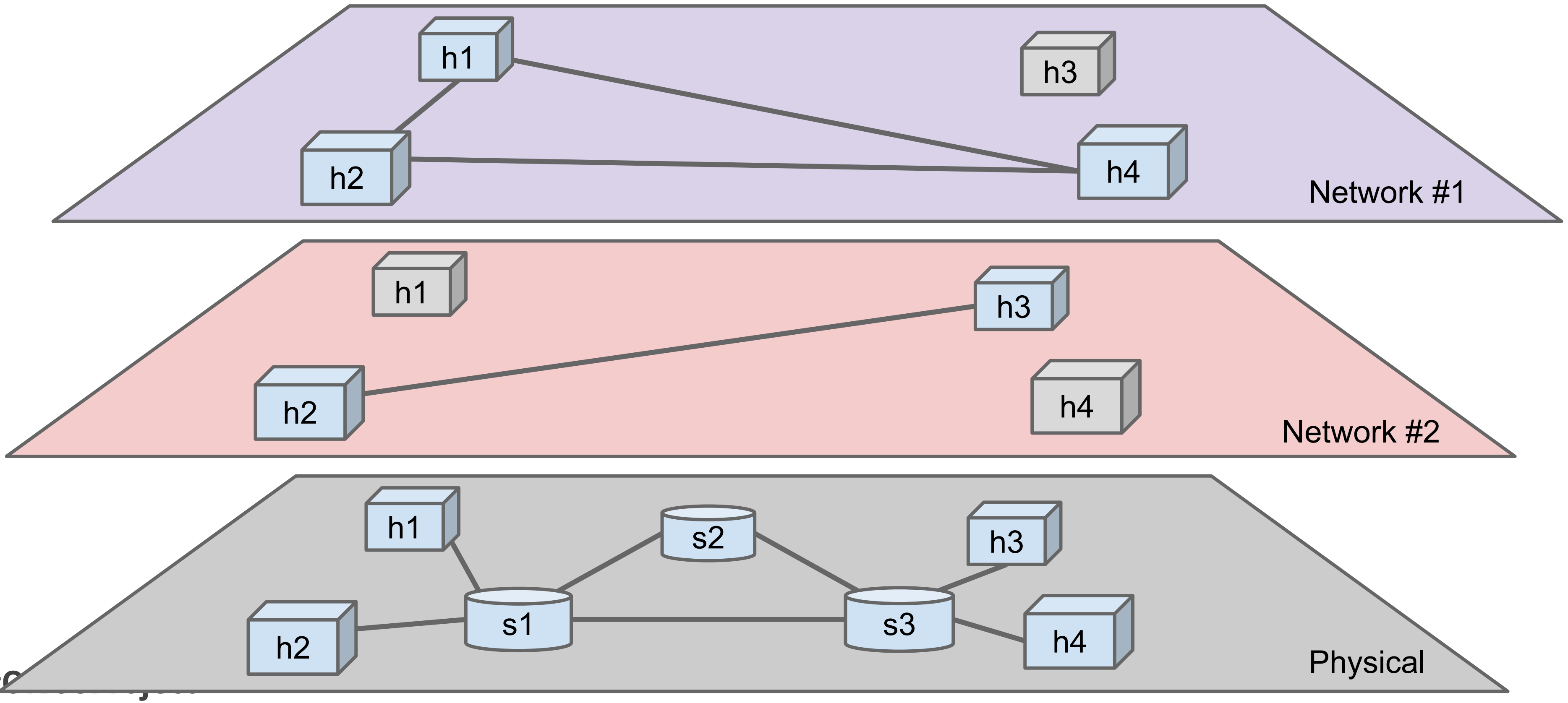
Bundles, Features & ONOS Apps

- 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

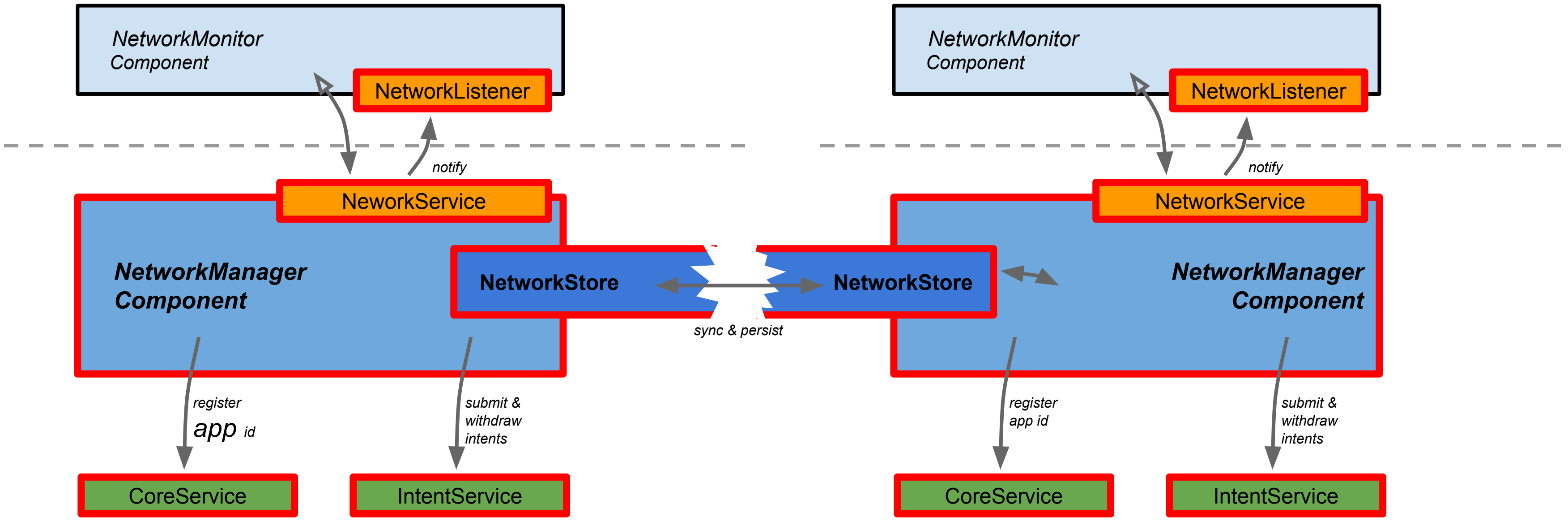
BYON Application

- 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

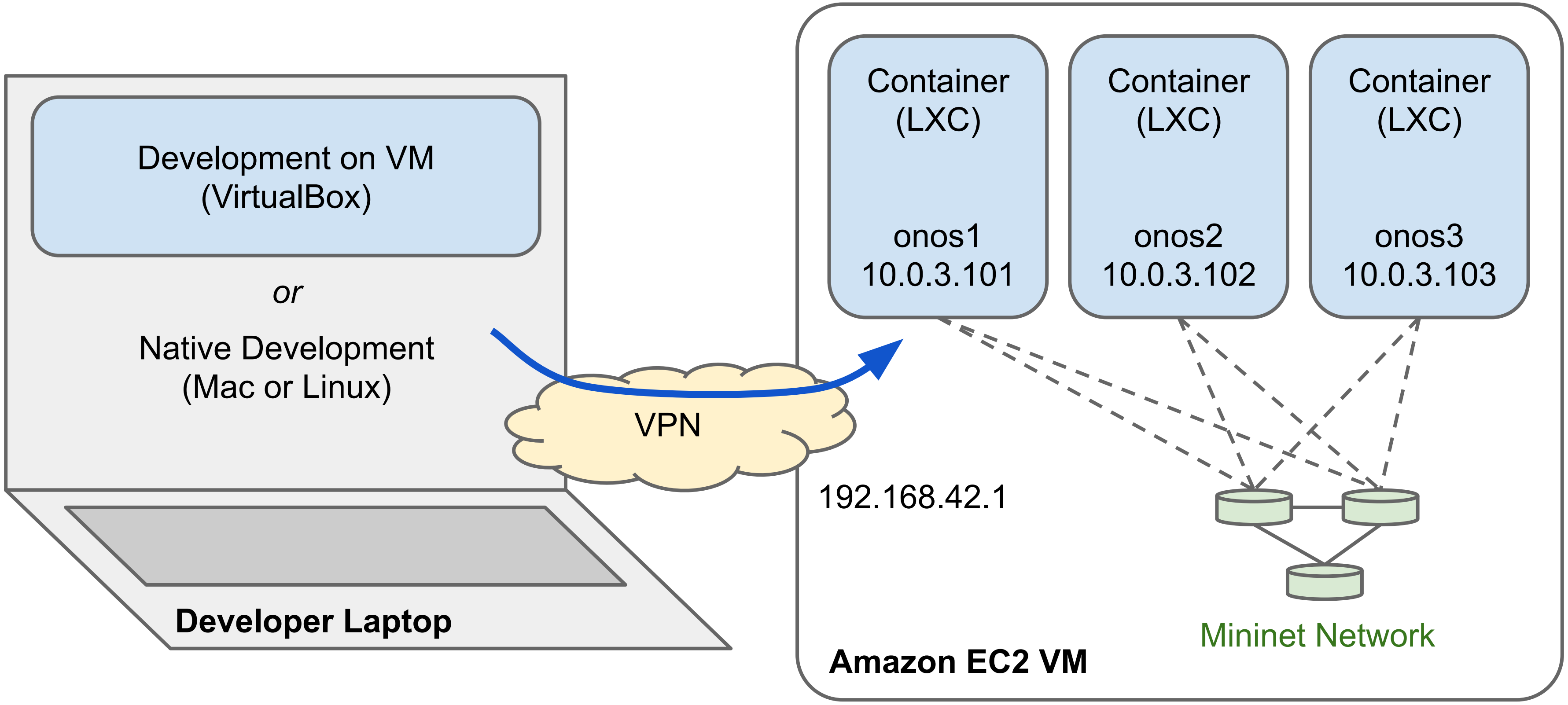
BYON Application Example



BYON App Structure



Environment Overview



#ONOSProject

Environment Setup (Laptop)

Native Development (Mac or Linux)

1. Install *IntelliJ* (or *Eclipse*)
2. Install *Oracle JDK 8*
3. Install *apache-maven*
4. Install *apache-karaf*
5. Install *curl*
6. *git clone https://gerrit.onosproject.org/onos*
7. Set up the ONOS *bash_profile*
8. Build onos
9. *git clone https://github.com/bocon13/onos-byon*

Development on VM (easiest)

1. Install *VirtualBox*
2. Import VM to VirtualBox
3. User: onos / Password: onos

Download Links

- These slides: http://tinyurl.com/onos-tutorial
- IntelliJ IDEA:
  https://www.jetbrains.com/idea/download/
- Eclipse:
  https://eclipse.org/downloads/
- Oracle JDK8
  http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Apache Maven (available via brew):
  https://maven.apache.org/download.cgi?Preferred=ftp://mirror.reverse.net/pub/apache/
- Apache Karaf (3.0.5):
  https://karaf.apache.org/index/community/download.html
- cURL (available via brew):
  http://curl.haxx.se/download.html

Environment Setup (VPN to EC2)

Server Address: <provided on request>   Test by ping 10.0.3.101

VPN Type: **PPTP**   User: **onos**   Password: **onos**   Encryption: **128 bit**

Mac

1. Choose Apple menu (top left corner) > **System Preferences**, then click **Network**
2. Click **Add (+)** at the bottom of the network connection services list, then choose **VPN** from the Interface pop-up menu. Enter password in **Authentication Settings...**
3. Click **Connect**

Windows

1. Right-click the network icon in the system tray and select **Open Network and Sharing Center**
2. Click **Set up a new connection or network**
3. On the wizard, select **Connect to a workplace**, and click **Next**
4. Select **Use my internet connection (VPN)**
5. Enter user and password, then **Connect**

(Windows only) To disable default gateway: *(Note: this must be done before connecting)*

1. Open the **Network Connections** window
2. Right-click the VPN connection > **Properties**, then click the **Networking** tab, then **TCP/IPv4**
3. Click the "**Advanced...**" button, and uncheck **Use default gateway on remote network**
4. Click **OK** three times

Tutorial Cell Setup

- Import the cell

```
$ cell tutorial
```

Lab Sections

- Lab #1: Basics
  - build skeletal app with a few provided files; build deploy app and test via CLI
- Lab #2: Core manager component
  - implement `NetworkService` methods; build, deploy, test
- Lab #3: Add intents
  - implement `addHost` to submit intents; build, deploy, test
- Lab #4: Remove intents
  - implement `removeHost` to withdraw intents
  - implement `remove-host` CLI command; build, deploy, test
- Lab #5: Distributed store component
  - implement `NetworkStore` using `ConsistentMap` primitive; build, deploy, test
- Lab #6: Events & Monitor
  - enhance `NetworkStore` and `NetworkManager` to propagate `NetworkEvents`
  - implement `NetworkMonitor` component to log events

Lab #1: Import & Build BYON App

- Follow Lab #1 of the Distributed Tutorial on the ONOS Wiki

https://goo.gl/5ezwoI

Lab #1: Recap

- Imported the BYON project into IDE
- Built app via `mvn`
- Deployed app via `onos-app` command
- Verified functionality via `list-networks` CLI command

Lab #2: Network Manager & Store

- Follow Lab #2 of the Distributed Tutorial on the ONOS Wiki

https://goo.gl/0SQkPO

```
$ ssh onos@$OCN
$ ./start-net.sh
```

Lab #2: Recap

- Referenced `NetworkStore` component via `@Reference`
- Implemented `NetworkManager` methods to use `NetworkStore` functionality
- Built & re-deployed the app
- Verified that network data is correctly tracked

Lab #3: Adding Intents

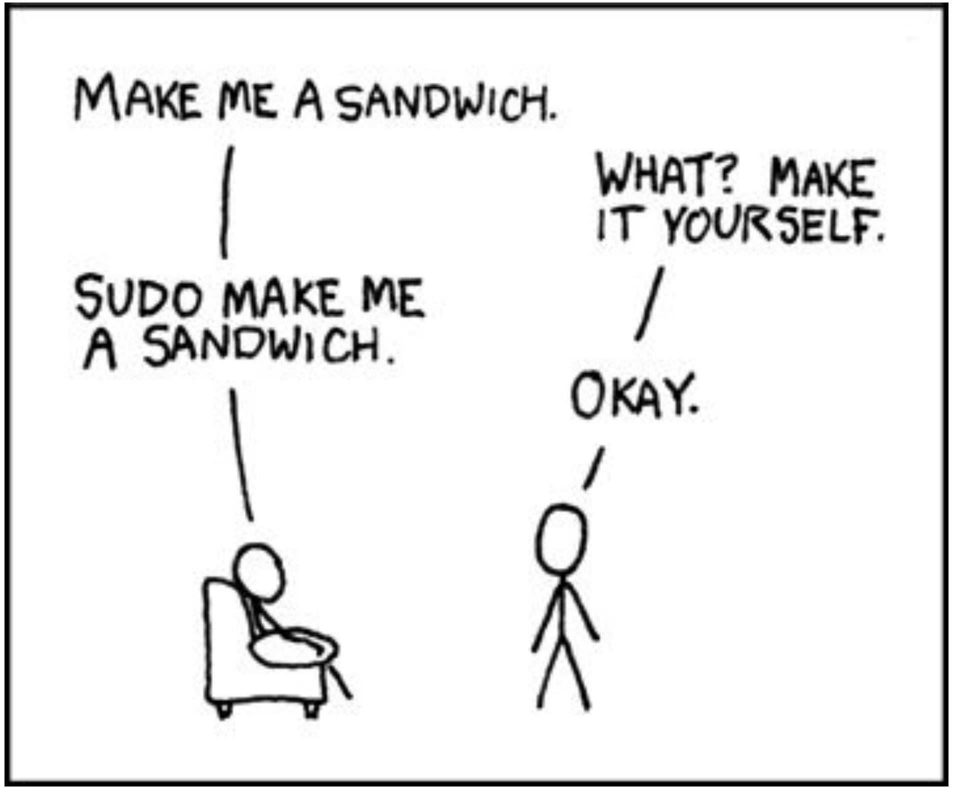
- Follow Lab #3 of the Distributed Tutorial on the ONOS Wiki

https://goo.gl/Xhe5SE

Lab #3: Recap

- Referenced `IntentService` component via `@Reference`
- Enhanced `NetworkManager` `addHost` method to create and submit required `HostToHostIntent`s
- Built & re-deployed the app
- Verified that intent is properly installed
- Verified that connectivity is established between hosts

LUNCH!


MAKE ME A SANDWICH.
WHAT? MAKE IT YOURSELF.
SUDO MAKE ME A SANDWICH.
OKAY.

Lab #4: Removing Intents

- Follow Lab #4 of the Distributed Tutorial on the ONOS Wiki

https://goo.gl/ZIjQlU

Lab #4: Recap

- Enhanced `NetworkManager removeHost` method to withdraw all required `HostToHostIntent`s
- Enhanced `NetworkManager removeNetwork` method to withdraw all required `HostToHostIntent`s
- Implemented and registered CLI commands
- Built & re-deployed the app
- Verified that intent is properly withdrawn
- Verified that connectivity is severed between hosts

Lab #5: Distributed Store

- Follow Lab #5 of the Distributed Tutorial on the ONOS Wiki

https://goo.gl/wx10vS

Lab #5: Recap

- Enhanced `DistributedNetworkStore` to use ONOS `ConsistentMap` distributed primitive
- Built & re-deployed the app
- Verified that intent is properly distributed to other ONOS instances in the cluster

Lab #6: Event Notifications

- Follow Lab #6 of the Distributed Tutorial on the ONOS Wiki

https://goo.gl/omi8tz

Lab #6: Recap

- Defined `NetworkEvent` and `NetworkStoreDelegate`
- Enhanced `DistributedNetworkStore` to delegate events to `NetworkManager` component
- Enhanced `NetworkManager` to notify event listeners
- Created `NetworkMonitor` component as event listener
- Built & re-deployed the app
- Verified that listeners are notified about network events

ONOS Tutorial Recap

- Imported project into IDE
- Used `mvn` to build and `onos-app` to deploy app
- Used `@Reference` to get reference to other components
- Controlled connectivity between hosts via `IntentService`
- Created a distributed store using `ConsistentMap` primitive
- Implemented CLI commands
- Created asynchronous event notification mechanism
- Implemented a component as a `NetworkEventListener`

Wrap-Up

- Browse ONOS Wiki
  https://wiki.onosproject.org/
- Watch ONOS how-to screencasts on YouTube
  https://goo.gl/8Druv0
- Browse ONOS Java API
  http://api.onosproject.org/
- Join ONOS onos-dev@onosproject.org mailing list
  https://wiki.onosproject.org/display/ONOS/ONOS+Mailing+Lists
- Engage with ONOS developers & community