



OpenDayLight Controller project Overview

Giovanni Meo (gmeo@cisco.com)

September 2013

Agenda

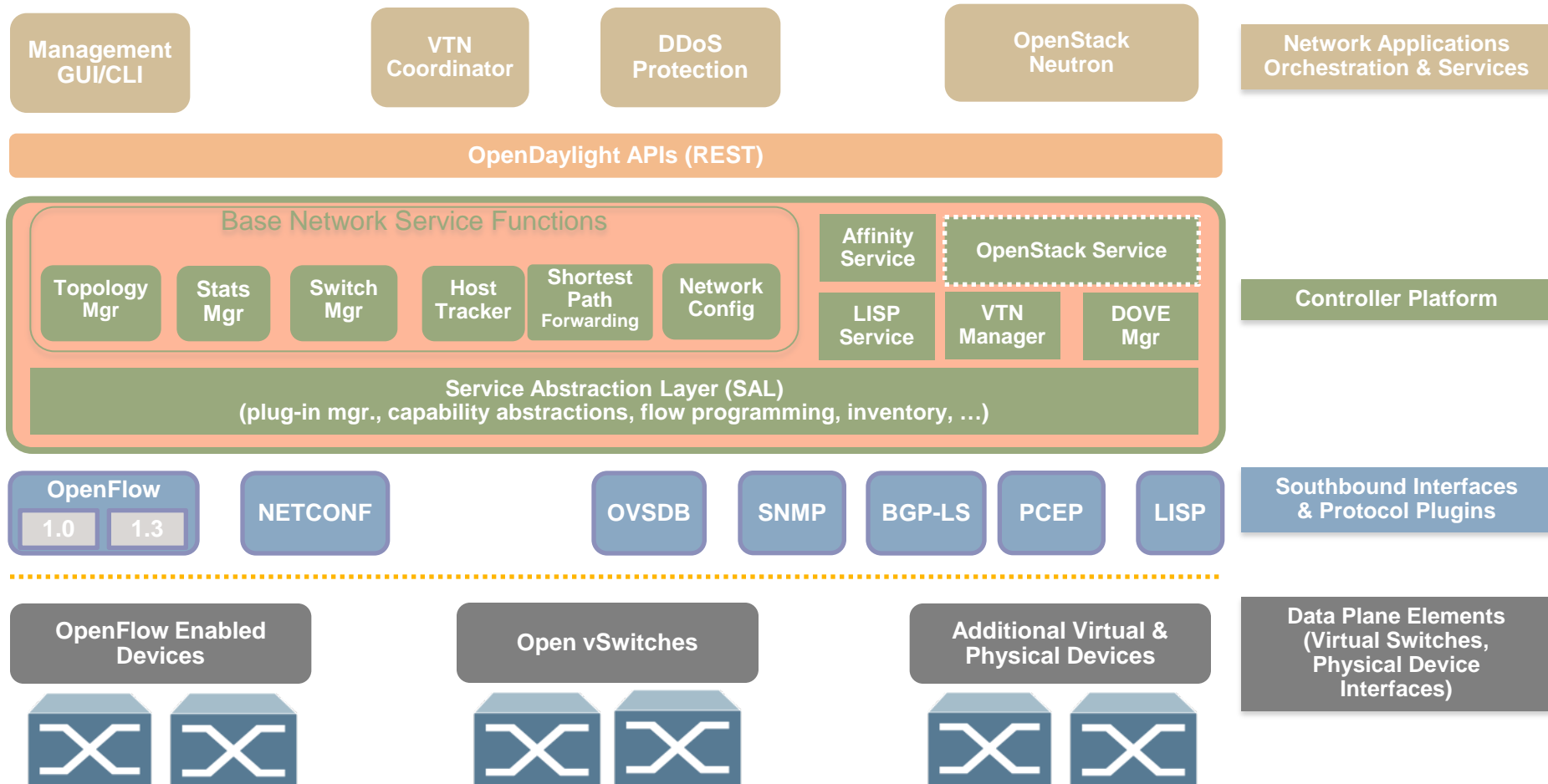
- Controller project in OpenDayLight context
- Controller project goals and how they map in reality
- Service Abstraction Layer
- Q&A



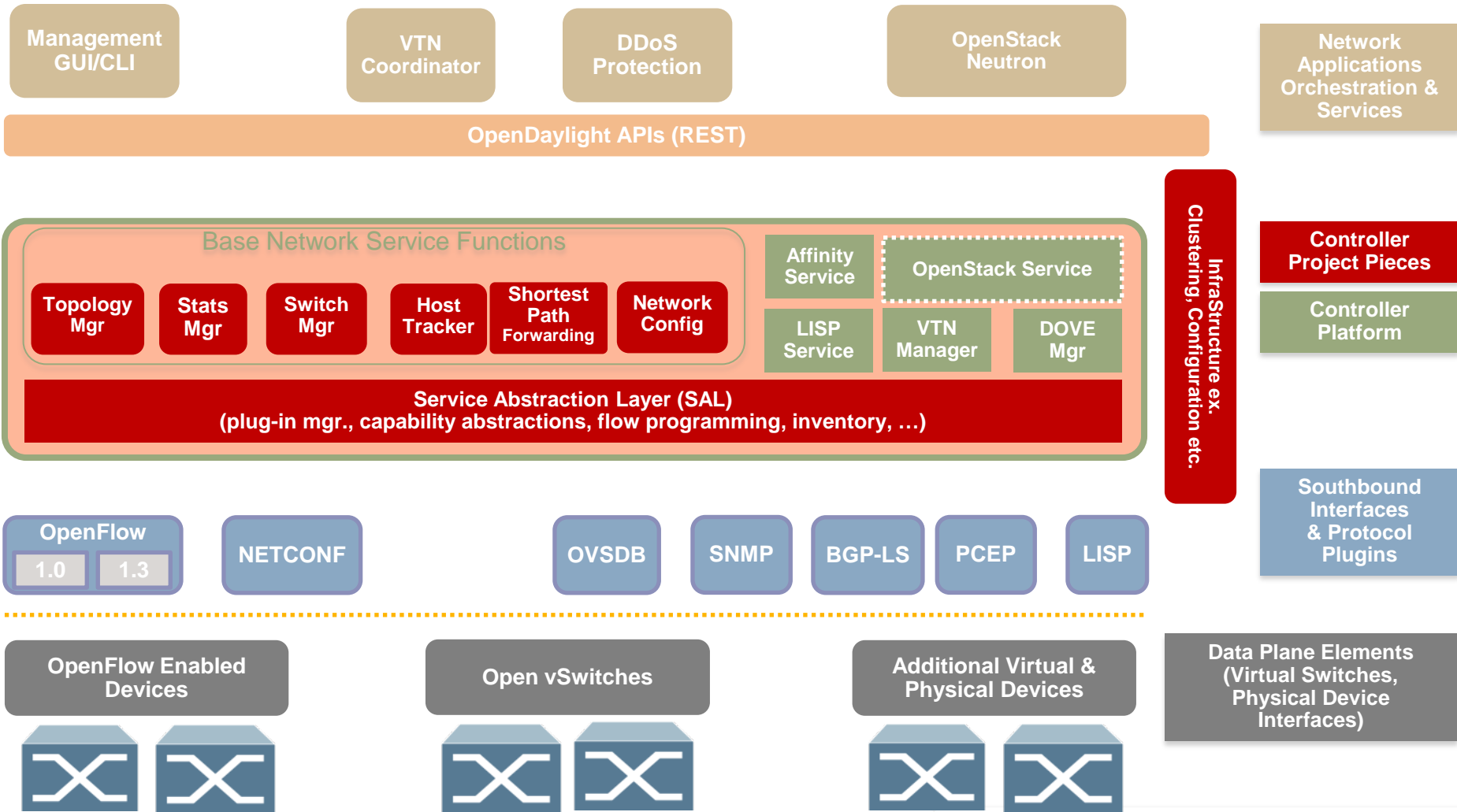
Agenda

- Controller project in OpenDayLight context
- Controller project goals and how they map in reality
- Service Abstraction Layer
- Q&A





VTN: Virtual Tenant Network
 DOVE: Distributed Overlay Virtual Ethernet
 DDoS: Distributed Denial Of Service
 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



VTN: Virtual Tenant Network
 DOVE: Distributed Overlay Virtual Ethernet
 DDoS: Distributed Denial Of Service
 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

Controller – basic facts

- Written in Java to leverage the huge amount of existing 3rd party to implement for example clustering or REST framework etc.
- NOT only JAVA: Any JVM (Java Virtual Machine) language can be used to create components for the controller
- Non-JVM components also present in OpenDayLight, which interact toward the rest of the system via northbound APIs or JNI/JNA
- Built to be a middleware where integrators or users can add their added value

Controller – list of components

- Base OSGI Framework
- Clustering infrastructure
- Service Abstraction Layer (SAL)
- OF 1.0 SB Plugin
- Switch Manager (inventory repository)
- Hosttracker
- Forwarding Rules Manager
- Includes(cont) :
 - Stats Manager
 - GUI
 - Topology Manager
 - and more...
- [Controller Release Plan](#)



Agenda

- Controller project in OpenDayLight context
- **Controller project principles and how they map in reality**
- Service Abstraction Layer
- Q&A



Controller goals

- Runtime Modularity and Extensibility
 - Extensible Northbound layer
 - Highly available and fault tolerant, with horizontal scalability properties
 - Support for Multi-Tenancy
- Multi-Protocol
 - Evolutionary
 - Transactional from application point of view



Controller goals in practice

[Runtime modularity and extensibility]

- Modularity and Extensibility achieved via the usage of OSGi framework as a base for developing all the controller components
- Every components of the controller is an OSGi bundle i.e a self-contained unit that can be installed/uninstalled at runtime
- No core components, only components with an high reference count

Controller goals in practice

[Extensible northbound layer]

- REST services implemented via a Servlet container
- Allows to insert at runtime HTTP resources to be manipulated via REST
- Why? Because not every one has same needs, and differentiations is created by the little devilish details
- Northbound for co-located apps implemented via OSGi services, which can be added and remove at runtime

Controller goals in practice

[HA and horizontally scalable]

- High availability achieved by allowing to create cluster of controllers
- Network elements can connect to any controller in the cluster to spread the load
- Network elements can be multi-homed to multiple controllers node. Only 1 controller will control the network element. (1+N redundancy on southbound)
- Applications can connect to any controller node and get the job done (N redundancy on northbound)
- States can be replicated in every node of the cluster or can be distributed using a DHT approach.

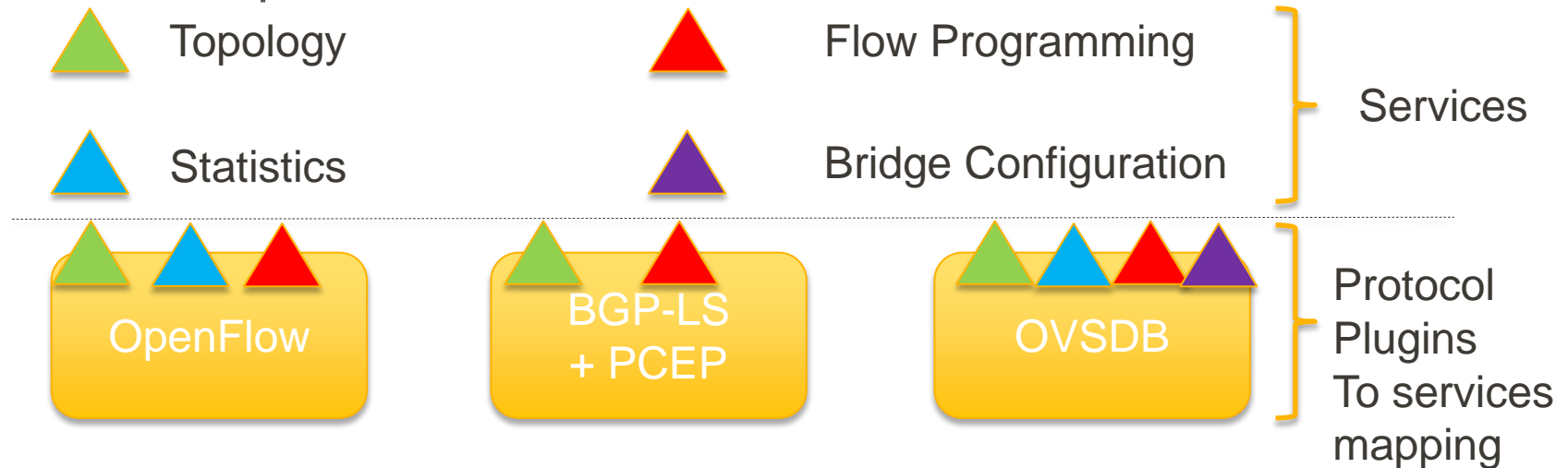
Controller goals in practice

[Support for multi-tenancy]

- A controller cluster can be partitioned in multiple containers
- Each container has an instance of all the components running independently from each other
- Allows separating the same cluster in multiple management domains (Network slicing)

Controller goals in practice [Multi-Protocol]

- Controller project is built to allow multiple SDN protocols, existing or yet to come.
- Achieved by defining common aspects of the different SDN protocols and mapping a protocol to one or more of these aspects



Controller goals in practice

[Evolutionary]

- Every bundle constituting the controller base is either an API bundle or an implementation bundle
- Each bundle has a version attached (mandated by OSGi framework)
- Multiple versions of the same API can coexist at the same time in the system so if a new implementation is provided still an adapter can be supplied to support older applications. It's by design that not all the bundles can migrate at the same time to newer versions.
- On the REST layer every API is versioned and multiple versions for the same API can be installed at the same time, to support legacy apps along with new ones.

Controller goals in practice

[Transactional]

- Every request made via northbound API to the controller will be transactional, i.e all completely done or none done.
- Important so the applications interacting with the controller can assume when they can start injecting traffic for example. Or to know when a security policy has been installed and traffic will be expected to be blocked.

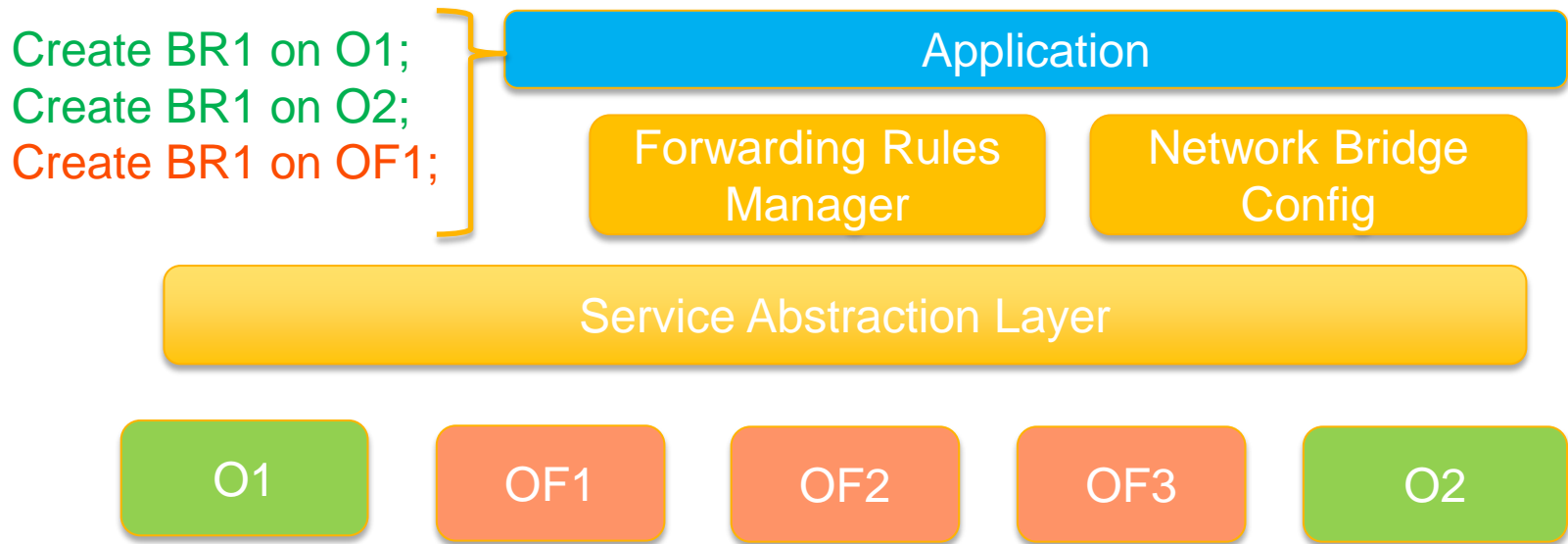
Agenda

- Controller project in OpenDayLight context
- Controller project principles and how they map in reality
- **Service Abstraction Layer**
- Q&A



Service Abstraction Layer

- Tiny layer of controller project which enables the multi-protocol support
- Enable a modules co-located on the controller to program a network element based on its capabilities



Service Abstraction Layer - evolution


- Currently services of a devices are defined via a Java interface
- Moving forward the capability of the device will be modeled using an appropriate modeling language like Yang (MD-SAL)
- Modeling language will then auto-generate the interfaces and from that other information like REST representation for data exchanged and so on.

Agenda

- Controller project in OpenDayLight context
- Controller project principles and how they map in reality
- Service Abstraction Layer
- Q&A



Resources

- More information and to join:
 - wiki.opendaylight.org
- Keep informed and join the conversation
 - IRC: #opendaylight on Freenode
 - Open mailing lists: lists.opendaylight.org
 - [@openDaylightSDN](https://twitter.com/openDaylightSDN) 
 - #OpenDaylight

