# OpendayLight

## 01 Introduction

Software Defined Network (SDN) is a new way of deploying network infrastructure. The SDN adapts to the requirements of applications deployed on the network. Current generation networks and architectures are statically configured and vertically integrated. New generation applications require networks to be agile and flexibly adapt to application requirements.

As networks get larger, especially in Massively Scalable Data Centers and Cloud, there is a large desire at ease-of-management and orchestration. This is leading to the need for programmatic interface (API) to the network to make it easier to write scripts as CLI and SNMP are not conducive to automation. There has been a change going on in IT and the CIO are getting influenced more by the Application and Server Admins and they who are used to the API and the ease of server management tools are demanding similar things from the network.

As application programmers desire their applications to be moved around in a data center or across clouds, it becomes imperative that the network becomes agile in meeting the requirements (bandwidth, services like load balancing, firewall) of the applications.

* Network Abstraction and Virtualization is desired as it allows the network operators to operate the network at a higher level without worrying about the quirkiness of different products from the same or different vendors.
* Additionally, there has been a desire from network operators for the ability to influence the forwarding and other network behavior based on their own algorithms and business logic. That means there is a need for the network to no longer be vertically integrated with the networking control logic coming only from the networking vendor.
* Finally, there has been a desire see the cost of networking gear come down especially amongst the Web Services and Cloud providers who build out large Data Centers. They thus view vendor neutrality and the rise of merchant silicon as leverage to be used against the networking vendors.

## Hardware and Software Requirements

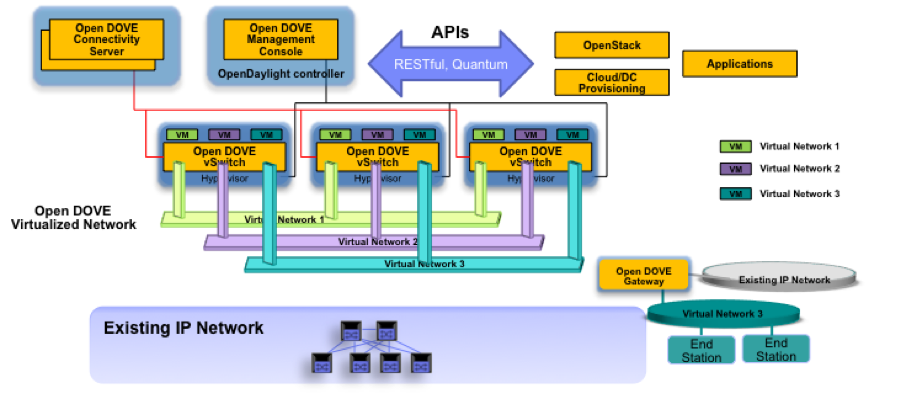
Open Daylight Controller is a JVM so it can run on any metal and OS provided it supports Java JVM 1.7+. We recommend the following:

* Linux (Ubuntu or RHEL or Fedora or Any other popular Linux Distro that supports Java)
* JVM 1.7+ (JAVA\_HOME should be set to environment)
* The Controller has a built in GUI. The GUI is implemented as an application using the same Northbound API as would be available for any other user application.

Architectural Principles

1. **Runtime Modularity and Extensibility**: Allow for a modular, extensible controller that supports installation, removal and updates of service implementations within a running controller, also known as in service software upgrade (ISSU)
2. **Multiprotocol Southbound**: Allow for more than one protocol interface with network elements with diverse capabilities southbound from the controller.
3. **Service Abstraction Layer (SAL)**: Where possible, allow for multiple southbound protocols to present the same northbound service interfaces.
4. **Open Extensible Northbound API**: Allow for an extensible set of application-facing APIs both across runtimes via REST (level 3 API) and within the same runtime, i.e., via function calls (level 2 API). The set of accessible functions should be the same.
5. **Support for Multitenancy/Slicing**: Allow for the network to be logically (and/or physically) split into different slices or tenants with parts of the controller, modules, explicitly dedicated to one or a subset of these slices. This includes allowing the controller to present different views of the controller depending on which slice the caller is from.
6. **Consistent Clustering**: Clustering that gives fine-grained redundancy and scale out while insuring network consistency.

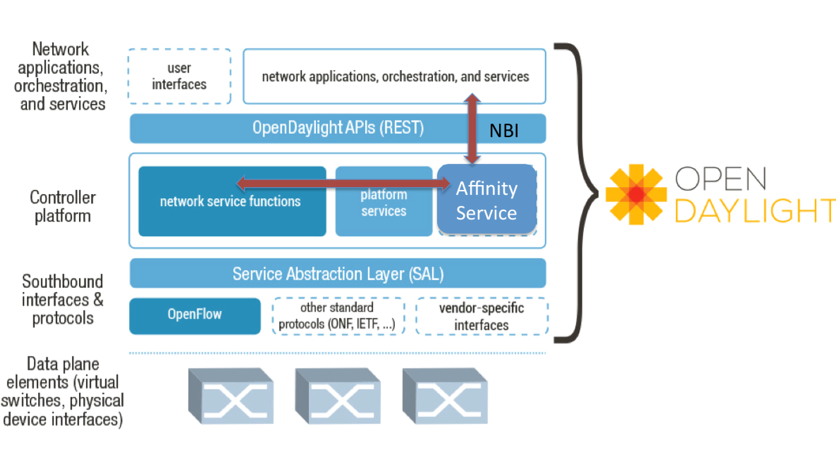
DOVE (distributed overlay virtual Ethernet) is a network virtualization platform that provides isolated multi-tenant networks on any IP network in a virtualized data center. DOVE provides each tenant with a virtual network abstraction providing layer-2 or layer-3 connectivity and the ability to control communication using access control policies. Address dissemination and policy enforcement in DOVE is provided by a clustered directory service. It also includes a gateway function to enable virtual machines on a virtual network to communicate with hosts outside the virtual network domain. The overall DOVE architecture is shown below:

[](https://wiki.opendaylight.org/view/File:Opendove-arch.png)

Users interact with Open DOVE to create and manage virtual networks through the Open DOVE Management Console (DMC) which provides a REST API for programmatic virtual network management and a basic graphical UI. The DMC is also used to configure the Open DOVE Gateway to configure connectivity to external, non-virtualized networks.

The Open DOVE Connectivity Server (DCS) supplies address and policy information to individual Open DOVE vswitches which implement virtual networks by encapsulating tenant traffic in overlays that span virtualized hosts in the data center. The DCS also includes support for high-availability and scale-out deployments through a lightweight clustering protocol between replicated DCS instances. The Open DOVE vswitches serve as policy enforcement points for traffic entering virtual networks. Open DOVE uses the VxLAN encapsulation format but implements a scalable control plane that does not require the use of IP multicast in the data center.

The DOVE technology was originally developed by IBM Research and has also been included in commercial products.

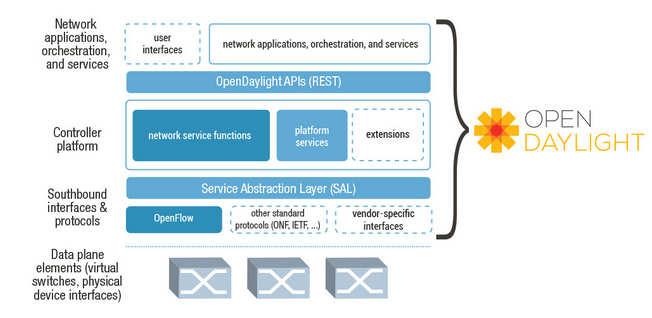
[](https://wiki.opendaylight.org/view/File:ODPAffinity.png)

The OSGi framework allows dynamically linking plugins for the evolving southbound protocols. The SAL provides basic services like Device Discovery which are used by modules like Topology Manager to build the topology and device capabilities. Services are constructed using the features exposed by the plugins (based on the presence of a plugin and capabilities of a network device). Based on the service request the SAL maps to the appropriate plugin and thus uses the most appropriate Southbound protocol to interact with a given network device. Each plugin is independent of each other and are loosely coupled with the SAL. (Please note that the OpenFlow 1.0 plugin is currently provided and other plugin shown in the pictures above are examples of the extensibility of the SAL framework. The SAL framework is included in the Open Daylight controller contribution).

* Topology Service is a set of services that allow to convey topology information like a new node a new link has been discovered and so on.
* Data Packet services, in summary the possibility to deliver to applications the packets coming from the agents, if any.
* Flow Programming service is supposed to provide the necessary logic to program in the different agents a Match/Actions rule.
* Statistics service will export API to be able to collect statistics at least per:
  + Flow
  + Node Connector (port)
  + Queue
* Inventory service will provide APIs for returning inventory information about the node and node connectors for example
* Resource service is a placeholder to query resource status

### 01-1 Overview

From a high level view, software defined networking is commonly described in layers.



**Network Apps & Orchestration:**The top layer consists of applications that utilize the network for normal network communications. Also included in this layer are business and network logic applications that control, and monitor network behavior.

**Controller Platform:**The middle layer is the framework in which the SDN abstractions can manifest; providing a set of common APIs to the application layer (commonly referred to as the northbound interface), while implementing one or more protocols for command and control of the physical hardware within the network (typically referred to as the southbound interface).

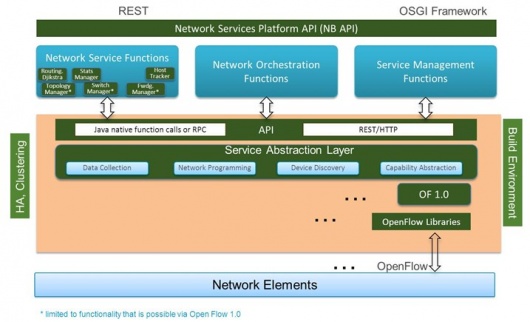
**Physical & Virtual Network Devices:**The bottom layer consists of the physical & virtual devices, switches, routers, etc., that make up the connective fabric between all endpoints within the network.

OpenDaylight is an open source project with a modular, pluggable, and flexible controller platform at its core.

The controller exposes open northbound APIs which are used by applications. OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API. The OSGi framework is used for applications that will run in the same address space as the controller while the REST (web based) API is used for applications that do not run in the same address space (or even necessarily on the same machine) as the controller. The business logic and algorithms reside in the applications. These applications use the controller to gather network intelligence, run algorithms to perform analytics, and then use the controller to orchestrate（精心策划，和谐安排） the new rules, if any, throughout the network.

The controller platform itself contains a collection of dynamically pluggable modules to perform needed network tasks. There are a series of base network services for such tasks as understanding what devices are contained within the network and the capabilities of each, statistics gathering, etc. In addition, platform oriented services and other extensions can also be inserted into the controller platform for enhanced SDN functionality.

### 01-2 Framework

[](https://wiki.opendaylight.org/view/File:Architectural_Framework.jpg)

The southbound interface is capable of supporting multiple protocols (as separate plugins), e.g. OpenFlow 1.0, OpenFlow 1.3, BGP-LS, etc. These modules are dynamically linked into a Service Abstraction Layer (SAL). The SAL exposes device services to which the modules north of it are written. The SAL determines how to fulfill the requested service irrespective of the underlying protocol used between the controller and the network devices.

This provides investment protection to the Applications as the OpenFlow and other protocols evolve over time. For the Controller to control devices in its domain it needs to know about the devices, their capabilities, reachability, etc. This information is stored and managed by the Topology Manager. The other components like ARP handler, Host Tracker, Device Manager and Switch Manager help in generating the topology database for the Topology Manager.

### 01-3 Interface

|  |  |  |
| --- | --- | --- |
| OpenDaylight Controller | | |
| **Package/OSGi Bundle** | **Exported Interfaces** | **Description** |
| org.opendaylight.controller.arphandler.internal | * **IHostFinder** * IListenDataPacket | Handles all ARP traffic: sending ARP requests to find hosts as well as responding to ARP requests from hosts. |
| org.opendaylight.controller.clustering.services\_implementation.internal | * **IClusterContainerServices** * **IClusterServices** * **IClusterGlobalServices** |  |
| org.opendaylight.controller.clustering.stub.internal | * **IClusterContainerServices** * **IClusterGlobalServices** |  |
| org.opendaylight.controller.configuration.internal | * **IConfigurationService** * **IConfigurationContainerService** * IConfigurationAware |  |
| org.opendaylight.controller.containermanager.internal | * **IContainerManager** * **IContainer** |  |
| org.opendaylight.controller.forwarding.staticrouting.internal | * **IForwardingStaticRouting** * ICacheUpdateAware * IfNewHostNotify * IConfigurationContainerAware |  |
| org.opendaylight.controller.forwardingrulesmanager.internal | * IContainerListener * ISwitchManagerAware * **IForwardingRulesManager** * IInventoryListener * ICacheUpdateAware * IConfigurationContainerAware * IFlowProgrammerListener |  |
| org.opendaylight.controller.hosttracker.internal | * ISwitchManagerAware * IInventoryListener * **IfIptoHost** * IfHostListener * ITopologyManagerAware |  |
| org.opendaylight.controller.protocol\_plugin.openflow.internal | * IContainerListener * **IController** * IDataPacketListen * **IDataPacketMux** * **IDiscoveryService** * IFlowProgrammerNotifier * IInventoryShimExternalListener * IMessageListener * **IOFStatisticsManager** * **IPluginInDataPacketService** * **IPluginInFlowProgrammerService** * **IPluginInInventoryService** * **IPluginInReadService** * **IPluginInTopologyService** * **IPluginReadServiceFilter** * **IRefreshInternalProvider** * IStatisticsListener * ITopologyServiceShimListener |  |
| org.opendaylight.controller.protocol\_plugins.stub.internal | * **IPluginInReadService** |  |
| org.opendaylight.controller.routing.dijkstra\_implementation.internal | * ITopologyManagerAware * **IRouting** |  |
| org.opendaylight.controller.sal.implementation.internal | * **IReadService** * **IPluginOutTopologyService** * **ITopologyService** * **IInventoryService** * **IPluginOutInventoryService** * **IFlowProgrammerService** * **IPluginOutFlowProgrammerService** * **IPluginOutDataPacketService** * **IDataPacketService** |  |
| org.opendaylight.controller.samples.loadbalancer.internal | * IListenDataPacket * **IConfigManager** |  |
| org.opendaylight.controller.samples.simpleforwarding.internal | * IInventoryListener * IfNewHostNotify * IListenRoutingUpdates |  |
| org.opendaylight.controller.statisticsmanager.internal | * **IStatisticsManager** |  |
| org.opendaylight.controller.switchmanager.internal | * IListenInventoryUpdates * **ISwitchManager** * ICacheUpdateAware * IConfigurationContainerAware |  |
| org.opendaylight.controller.topologymanager.internal | * IListenTopoUpdates * **ITopologyManager** * IConfigurationContainerAware |  |
| org.opendaylight.controller.usermanager.internal | * ICacheUpdateAware * **IUserManager** * IConfigurationAware |  |

## 02 arphandler

1. 简介

该模块用于解析arp包.ARP(Address Resolution Protocol，地址解析协议)是获取[物理地址](http://baike.baidu.com/view/883168.htm" \t "_blank)的一个TCP/IP协议。某节点的IP地址的ARP请求被广播到网络上后，这个节点会收到确认 其[物理地址](http://baike.baidu.com/view/883168.htm)的应答，这样的[数据包](http://baike.baidu.com/view/25880.htm)才能被传送出去。RARP(逆向ARP)经常在[无盘工作站](http://baike.baidu.com/view/51922.htm" \t "_blank)上使用，以获得它的逻辑IP地址。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.hosttracker,

org.opendaylight.controller.hosttracker.hostAware,

org.apache.felix.dm,

org.osgi.service.component,

org.slf4j

<dependencies>

switchmanager

hosttracker

sal

1. 流程图

## 03 clustering

### 03-1 Services

1. 简介

定义了一系列集群服务接口，包含ICacheUpdateAware集群缓存更新时需要监听的接口，IClusterServicesCommon提供了一组集群中常用的接口，ICoordinatorChangeAware，IGetUpdates，IListenRoleChange角色改变时需要监听的接口, IClusterServices.

1. 类图



1. 基本原理
2. 依赖服务

导出包：

<Export-Package>

org.opendaylight.controller.clustering.services

</Export-Package>

<Import-Package>

javax.transaction

</Import-Package>

1. 流程图

### 03-2 Services-implementations

1. 简介

该模块是集群服务的实现模块，主要实现了上述Services包中的接口服务。集群之间通过JGroup来进行通信，同步数据。集群数据采用infinispan来进行存储。

***Inﬁnispan*** 是个开源的数据网格平台。它公开了一个简单的数据结构（一个Cache）来存储对象。虽然可以在本地模式下运行Inﬁnspan，但其真正的价值在于分布 式，在这种模式下，Inﬁnispan可以将集群缓存起来并公开大容量的堆内存。这可比简单的复制强大的多，因为它会为每个结点分配固定数量的副本——服 务器故障的一种恢复手段——同时还提升了可伸缩性，这是由于存储每个结点所需的工作量是与集群大小息息相关的。

***JGroup***是当前被广泛使用的可靠组间通信的工具之一。例如OSCache以及JBossTreeCache都是用的是JGroup。JGroup功能十分强大，通过配置各种参数就可以充分利用它所提供的各项功能。JGroup最大的特点就是支持协议栈的可配置性，它本是实现了基本的Java的协议栈实现，也就是最基本的消息广播的基础，同时支持附加协议栈的配置，消息的传递就是在这些协议栈之间相互传递，封装，检查，丢弃，重发。JGroup可以基于TCP协议来实现消息广播，也可以通过UDP方式来广播消息，利弊不言而喻，TCP可靠，但是代价大，性能没有UDP来的好，UDP速度快，代价小，但是消息的丢失率以及无序性有着很大的限制。但是JGroup在UDP方式的基础上，增加了协议栈的配置，通过配置上层的协议，可以保证消息的重发，大包体的分解（同时保证消息包体顺序），组内机器的状态检测等功能。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.slf4j,

!org.jboss.\*,

!bsh\*,

!net.jcip.\*,

javax.transaction,

\*,

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.sal.core

</Import-Package>

**<dependencies>**

<dependency>

<groupId>org.infinispan</groupId>

<artifactId>infinispan-core</artifactId>

<version>5.2.3.Final</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>clustering.services</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>sal</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

**</dependencies>**

1. 流程图

### 03-3 Stub

1. 简介

实现了Services中的IClusterContainerServices，IClusterGlobalServices，IClusterServicesCommon接口。

1. 类图



1. 基本原理
2. 依赖服务

**<Import-Package>**

javax.transaction,

org.apache.felix.dm,

org.slf4j,

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.sal.core

**</Import-Package>**

**<dependencies>**

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>clustering.services</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>sal</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

**</dependencies>**

1. 流程图

## 04 commons

简介

该模块提供了3个配置文件，定义了一些公共属性，供其它模块使用。

## 05 configuration

**configuration**

1. 简介

该模块提供了配置管理的接口。

1. 类图



1. **基本原理**
2. **依赖服务**

**<Import-Package>**

**org.opendaylight.controller.sal.utils,**

**org.apache.commons.lang3.builder**

**</Import-Package>**

**<Export-Package>**

**org.opendaylight.controller.configuration**

**</Export-Package>**

**<dependencies>**

**<dependency>**

**<groupId>org.opendaylight.controller</groupId>**

**<artifactId>sal</artifactId>**

**<version>0.4.0-SNAPSHOT</version>**

**</dependency>**

**</dependencies>**

**configuration.implementation**

1. 简介

该模块实现了configuration的口，完成一些和配置相关的任务。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.configuration,

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.core,

org.osgi.framework,

org.slf4j,

org.apache.felix.dm

</Import-Package>

<dependencies>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>clustering.services</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>configuration</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>sal</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

</dependencies>

## 06 containermanger

**containermanger**

1. 简介

提供容器管理的基本接口。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.sal.action,

org.opendaylight.controller.sal.authorization,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.flowprogrammer,

org.opendaylight.controller.sal.match,

org.opendaylight.controller.sal.reader,

org.opendaylight.controller.sal.utils,

org.apache.commons.lang3.builder

</Import-Package>

<Export-Package>

org.opendaylight.controller.containermanager

</Export-Package>

**containermanager.implementation**

1. 简介

容器管理实现类，

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.containermanager,

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.action,

org.opendaylight.controller.sal.flowprogrammer,

org.opendaylight.controller.sal.match,

org.opendaylight.controller.sal.reader,

org.eclipse.osgi.framework.console,

org.osgi.framework,

org.slf4j,

org.apache.felix.dm

</Import-Package>

<dependencies>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>clustering.services</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>containermanager</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>sal</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

</dependencies>

## 07 distribuction

1. 简介

是opendaylight的主入口，提供了启动程序的脚本，run.bat为windows下运行脚本，run.sh为linux下运行脚本。

1. 类图
2. 基本原理
3. 系统模块

<modules>

<module>../../forwarding/staticrouting</module>

<module>../../clustering/services</module>

<module>../../clustering/services\_implementation</module>

<module>../../clustering/stub</module>

<module>../../clustering/test</module>

<module>../../configuration/api</module>

<module>../../configuration/implementation</module>

<module>../../routing/dijkstra\_implementation</module>

<module>../../arphandler</module>

<module>../../forwardingrulesmanager</module>

<module>../../hosttracker</module>

<module>../../containermanager/api</module>

<module>../../containermanager/implementation</module>

<module>../../switchmanager</module>

<module>../../statisticsmanager</module>

<module>../../topologymanager</module>

<module>../../usermanager</module>

<module>../../security</module>

<module>../../../third-party/openflowj</module>

<module>../../../third-party/net.sf.jung2</module>

<module>../../../third-party/jersey-servlet</module>

<!-- SAL bundles -->

<module>../../sal/api</module>

<module>../../sal/implementation</module>

<!-- Web bundles -->

<module>../../web/root</module>

<module>../../web/flows</module>

<module>../../web/devices</module>

<module>../../web/troubleshoot</module>

<module>../../web/topology</module>

<!-- Northbound bundles -->

<module>../../northbound/commons</module>

<module>../../northbound/topology</module>

<module>../../northbound/staticrouting</module>

<module>../../northbound/statistics</module>

<module>../../northbound/flowprogrammer</module>

<module>../../northbound/hosttracker</module>

<module>../../northbound/subnets</module>

<module>../../northbound/switchmanager</module>

<!-- Debug and logging -->

<module>../../logging/bridge</module>

<!-- Southbound bundles -->

<module>../../protocol\_plugins/openflow</module>

<!-- Samples -->

<module>../../samples/simpleforwarding</module>

<module>../../samples/loadbalancer</module>

<module>../../samples/northbound/loadbalancer</module>

</modules>

## 08 forwarding

**forwarding.staticrouting**

1. 简介

提供静态路由配置与静态路由的创建功能，用于连接SDN网络与非SDN网络。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.configuration,

org.opendaylight.controller.forwardingrulesmanager,

org.opendaylight.controller.hosttracker,

org.opendaylight.controller.hosttracker.hostAware,

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.routing,

org.opendaylight.controller.topologymanager,

org.eclipse.osgi.framework.console,

org.osgi.framework,

org.slf4j,

org.apache.felix.dm,

org.apache.commons.lang3.builder

</Import-Package>

<Export-Package>

org.opendaylight.controller.forwarding.staticrouting

</Export-Package>

<dependencies>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>topologymanager</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>forwardingrulesmanager</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>hosttracker</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>configuration</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>sal</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

</dependencies>

1. 流程图

**forwardingrulesmanager**

1. 简介

管理网络容器中的转发规则，包括安装与删除；同时也负责维护安装在网络节点上的转发规则的中心存储仓库。

1. 类图



1. 基本原理
2. 依赖服务

<Export-Package>

org.opendaylight.controller.forwardingrulesmanager

</Export-Package>

<Import-Package>

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.configuration,

org.opendaylight.controller.hosttracker,

org.opendaylight.controller.hosttracker.hostAware,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.sal.action,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.flowprogrammer,

org.opendaylight.controller.sal.match,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.packet,

javax.xml.bind.annotation,

javax.xml.bind,

org.apache.felix.dm,

org.apache.commons.lang3.builder,

org.osgi.service.component,

org.slf4j,

org.eclipse.osgi.framework.console,

org.osgi.framework

</Import-Package>

<dependencies>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>hosttracker</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>configuration</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>clustering.services</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>switchmanager</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.opendaylight.controller</groupId>

<artifactId>sal</artifactId>

<version>0.4.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

## 10 hosttracker

1. 简介

主机跟踪模块，hosttracker的作用类似于floodlight中的devicemanager，用于发现终端主机。

1. 类图



1. 基本原理

hosttracker能够学习终端主机信息，并把host信息保存于本地数据库中。记录host的信息包括IP/MAC/VLAN/SW/PORT.学习的过程包括动态、静态两种，动态就是通过ARP消息学习主机信息，静态是通过特定的NorthBounds API来添加、删除本地数据库中host信息。

1. 流程图

## 11 northbound

该包负责提供控制器的北向接口。采用了Jersey框架发布REST服务。

### 11-1 commons

1. 简介

该模块提供了资源层常见的异常类型以及rest消息的类型，常见的rest消息类型有：

*SUCCESS*("Success"), *NOCONTAINER*("Container does not exist"), *NOFLOWSPEC*(

"Flow Spec does not exist"), *NOSUBNET*("Subnet does not exist"), *NOSTATICROUTE*(

"Static Route does not exist"), *NOHOST*("Host does not exist"), *NOFLOW*(

"Flow does not exist"), *NONODE*("Node does not exist"), *NOPOLICY*(

"Policy does not exist"), *NORESOURCE*("Resource does not exist"), *RESOURCECONFLICT*(

"Operation failed due to Resource Conflict"), *NODEFAULT*(

"Container default is not a custom container"), *DEFAULTDISABLED*(

"Container(s) are configured. Container default is not operational"), *NOTALLOWEDONDEFAULT*(

"Container default is a static resource, no modification allowed on it"), *UNKNOWNACTION*(

"Unknown action"), *INVALIDJSON*("JSON message is invalid"), *INVALIDADDRESS*(

"invalid InetAddress"), *AVAILABLESOON*(

"Resource is not implemented yet"), *INTERNALERROR*("Internal Error"), *SERVICEUNAVAILABLE*(

"Service is not available. Could be down for maintanence"), *INVALIDDATA*(

"Data is invalid or conflicts with URI");

1. 类图



1. 基本原理
2. 依赖服务

<Export-Package>

org.opendaylight.controller.northbound.commons.exception,

org.opendaylight.controller.northbound.commons

</Export-Package>

<Import-Package>

javax.ws.rs,

javax.ws.rs.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.usermanager,

javax.servlet.http,

org.slf4j,

</Import-Package>

### 11-2 flowprogrammer

1. 简介

提供了流表编程的rest接口，包含增加，删除，查询流表内容。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.forwardingrulesmanager,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.containermanager,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

com.sun.jersey.spi.container.servlet,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind.annotation,

javax.xml.bind,

org.slf4j,

!org.codehaus.enunciate.jaxrs

</Import-Package>

<Web-ContextPath>/controller/nb/v2/flow</Web-ContextPath>

### 11-3 hosttracker

1. 简介

提供了主机相关的api。包含添加主机，删除主机，查询主机信息等。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.hosttracker,

org.opendaylight.controller.hosttracker.hostAware,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.containermanager,

org.opendaylight.controller.switchmanager,

org.apache.commons.logging,

com.sun.jersey.spi.container.servlet,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind.annotation,

javax.xml.bind,

org.slf4j,

!org.codehaus.enunciate.jaxrs

</Import-Package>

<Web-ContextPath>/controller/nb/v2/host</Web-ContextPath>

### 11-4 staticrouting

1. 简介

该模块定义了静态路由的rest api。接口包含添加，删除，查询静态流表。

一个静态路由对象包含：

InetAddress networkAddress;

InetAddress mask;

NextHopType type;

InetAddress nextHopAddress;

Node node;

NodeConnector port;

HostNodeConnector host;

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.forwarding.staticrouting,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.containermanager,

com.sun.jersey.spi.container.servlet,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

org.slf4j,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind.annotation,

javax.xml.bind,

!org.codehaus.enunciate.jaxrs

</Import-Package>

<Web-ContextPath>/controller/nb/v2/staticroute</Web-ContextPath>

### 11-5 statistics

1. 简介

该模块提供了端口统计，流统计的rest api。

端口统计的字段有：

**struct** NodeConnectorStatistics {

@XmlElement

**private** NodeConnector nodeConnector;

@XmlElement

**private** **long** receivePackets;

@XmlElement

**private** **long** transmitPackets;

@XmlElement

**private** **long** receiveBytes;

@XmlElement

**private** **long** transmitBytes;

@XmlElement

**private** **long** receiveDrops;

@XmlElement

**private** **long** transmitDrops;

@XmlElement

**private** **long** receiveErrors;

@XmlElement

**private** **long** transmitErrors;

@XmlElement

**private** **long** receiveFrameError;

@XmlElement

**private** **long** receiveOverRunError;

@XmlElement

**private** **long** receiveCrcError;

@XmlElement

**private** **long** collisionCount;

}

**流统计的字段有**：

**struct** FlowOnNode {

@XmlElement

**private** Flow flow; // Flow descriptor

@XmlElement

**private** **byte** tableId;

@XmlElement

**private** **int** durationSeconds;

@XmlElement

**private** **int** durationNanoseconds;

@XmlElement

**private** **long** packetCount;

@XmlElement

**private** **long** byteCount;

}

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.forwardingrulesmanager,

org.opendaylight.controller.sal.action,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.flowprogrammer,

org.opendaylight.controller.sal.match,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.reader,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.containermanager,

org.opendaylight.controller.statisticsmanager,

org.opendaylight.controller.switchmanager,

org.apache.commons.logging,

com.sun.jersey.spi.container.servlet,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind.annotation,

javax.xml.bind,

org.slf4j,

!org.codehaus.enunciate.jaxrs

</Import-Package>

### 11-6 switchmanager

1. 简介

该模块提供了交换机管理模块的rest api。包含添加节点，删除节点，获取节点，获取链路信息以及它们的属性信息。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.containermanager,

org.opendaylight.controller.switchmanager,

org.apache.commons.lang3.tuple,

org.apache.commons.logging,

com.sun.jersey.spi.container.servlet,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind.annotation,

javax.xml.bind,

org.slf4j,

!org.codehaus.enunciate.jaxrs

</Import-Package>

<Web-ContextPath>/controller/nb/v2/switch</Web-ContextPath>

### 11-7 subnets

1. 简介

该模块用于提供容器中子网操作的rest api。包含添加，查询，删除子网接口。业务逻辑在switchmanage中实现。

一个子网包含如下信息：

**private** String name;

**private** String subnet; // A.B.C.D/MM Where A.B.C.D is the Default

// Gateway IP (L3) or ARP Querier IP (L2

**private** List<String> nodePorts; // datapath ID/port list:

// xx:xx:xx:xx:xx:xx:xx:xx/a,b,c-m,r-t,y

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.containermanager,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

com.sun.jersey.spi.container.servlet,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind.annotation,

org.slf4j,

!org.codehaus.enunciate.jaxrs

</Import-Package>

### 11-8 topology

1. 简介

该模块提供了拓扑逻辑相关api。包含获取拓扑信息，获取用户链接，删除用户链接，增加用户链接接口。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.containermanager,

org.opendaylight.controller.northbound.commons,

org.opendaylight.controller.northbound.commons.exception,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.packet.address,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.topologymanager,

com.sun.jersey.spi.container.servlet,

javax.ws.rs,

javax.ws.rs.core,

javax.xml.bind,

javax.xml.bind.annotation,

org.slf4j,

!org.codehaus.enunciate.jaxrs

</Import-Package>

<Web-ContextPath>/controller/nb/v2/topology</Web-ContextPath>

## 12 protocol\_plugins

1. 简介

协议插件模块，该模块负责与交换机建立连接，并监听消息。采用NIO方式与交换机进行通信,处理交换机的消息事件。主要在Controller类中实现。Openflow协议的实现在第三方包中实现。

1. 类图







1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.action,

org.opendaylight.controller.sal.discovery,

org.opendaylight.controller.sal.topology,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.flowprogrammer,

org.opendaylight.controller.sal.reader,

org.opendaylight.controller.sal.inventory,

org.opendaylight.controller.sal.match,

org.opendaylight.controller.sal.utils,

org.apache.commons.lang3.builder,

org.apache.commons.lang3.tuple,

org.apache.felix.dm,

org.slf4j,

org.eclipse.osgi.framework.console,

org.osgi.framework,

javax.net.ssl

</Import-Package>

<Export-Package>

org.opendaylight.controller.protocol\_plugin.openflow.internal

</Export-Package>

<Embed-Dependency>

org.openflow.openflowj

</Embed-Dependency>

## 13 routing

1. 简介

该模块实现了dijkstra最短路径路由算法。监听了IListenTopoUpdates接口，当拓扑更新时会自动计算相应路径。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.slf4j,

org.opendaylight.controller.sal.routing,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.reader,

org.apache.commons.collections15,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.topologymanager,

edu.uci.ics.jung.graph,

edu.uci.ics.jung.algorithms.shortestpath,

edu.uci.ics.jung.graph.util,

org.apache.felix.dm,

org.junit;resolution:=optional

</Import-Package>

## 14 sal

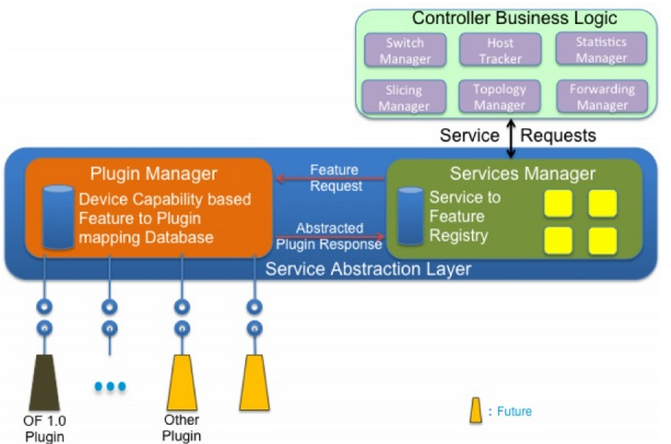
1. 简介

服务抽象层用来连接各个模块，它提供了设备服务用于北向模块向里面写。SAL决定了如何响应请求，而不需要控制器和网络设备之间用了何种协议。

[These modules are dynamically linked into a Service Abstraction Layer (SAL). The SAL exposes device services to which the modules north of it are written. The SAL determines how to fulfill the requested service irrespective of the underlying protocol used between the controller and the network devices.]

1. 类图
2. 基本原理

服务抽象层，在该控制器的模块化设计的核心，南向它能够支持多种协议，并且为模块和应用程序提供一致的服务。



* Topology Service is a set of services that allow to convey topology information like a new node a new link has been discovered and so on.
* Data Packet services, in summary the possibility to deliver to applications the packets coming from the agents, if any.
* Flow Programming service is supposed to provide the necessary logic to program in the different agents a Match/Actions rule.
* Statistics service will export API to be able to collect statistics at least per:
* Flow
* Node Connector (port)
* Queue
* Inventory service will provide APIs for returning inventory information about the node and node connectors for example
* Resource service is a placeholder to query resource status

1. 依赖服务

<Import-Package>

org.slf4j,

org.osgi.framework,

org.apache.commons.lang3.builder,

org.apache.felix.dm,

org.apache.commons.lang3.tuple,

javax.xml.bind.annotation,

javax.xml.bind.annotation.adapters

</Import-Package>

<Export-Package>

org.opendaylight.controller.sal.authorization,

org.opendaylight.controller.sal.action,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.discovery,

org.opendaylight.controller.sal.topology,

org.opendaylight.controller.sal.routing,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.packet.address,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.match,

org.opendaylight.controller.sal.inventory,

org.opendaylight.controller.sal.flowprogrammer,

org.opendaylight.controller.sal.reader

</Export-Package>

### 14-1 action

1. 简介

该模块定义了一系列控制器支持的行为，用于匹配的帧 /包/消息中。包含：drop,floodALL,controller, software path, harware path, enqueue, setDlDst, setVlanPcp, stripVlan, pushVlan, setDlType, setNwSrc等.

1. 类图



1. 基本原理
2. 依赖服务

### 14-2 authorization

1. 简介

权限认证模块，包含资源权限，用户角色级别，应用用户角色级别三种。

资源权限：

*NONE*(""), // no privilege

*READ*("r"), // read only

*USE*("u"), // use

*WRITE*("w"); // modify

用户角色级别:

*SYSTEMADMIN*(0, "System-Admin", "System Administrator"), // can do everything

*NETWORKADMIN*(1, "Network-Admin", "Network Administrator"), // can do everything but setting a system admin user profile

*NETWORKOPERATOR*(2, "Network-Operator", "Network Operator"), // can only see what is configured anywhere

*CONTAINERUSER*(4, "Container-User", "Container User"), // container context user

*APPUSER*(5, "App-User", "Application User"), // application context user

*NOUSER*(255, "Not Authorized", "Not Authorized");

应用用户角色级别:

*APPADMIN*(0, "App-Admin", "Application Administrator"),

*APPUSER*(1, "App-User", "Application User"),

*APPOPERATOR*(2, "App-Operator","Application Operator"),

*NOUSER*(255, "Unknown User", "Unknown User");

1. 类图



1. 基本原理
2. 依赖服务

### 14-3 core

1. 简介

该模块是核心模块，定义了一些公用的类, 是连接opendaylight底层业务代码与OSGI框架的桥梁，也是入口。ComponentActivatorAbstractBase实现了BundleActivator接口，并完成了模块服务的注册，是其它模块的基类。继承了该类的服务可以让osgi发现这些模块。

1. 类图

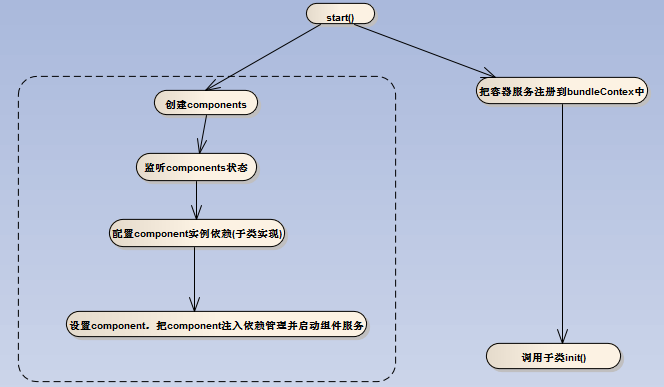


1. 基本原理

其关键类ComponentActivatorAbstractBase实现了BundleActivator接口，其本质是一个大的bundle，通过BundleContext获得bundle上下文管理其他bundle生命周期.当OSGI框架启动时，调用ComponentActivatorAbstractBase类中的start()方法，根据其派生类提供的getGlobalImplementations()，创建组件并监听组件的状态变化，然后执行各个派生类重写的configureGlobalInstance(Component c, Object imp)方法来配置它们的依赖。通过c.setImplementation(imps[i])设置各个组件使其能管理它的生命周期。把组件加入依赖管理并启动组件服务。最后绑定该服务生产者注册容器服务，使继承自ComponentActivatorAbstractBase的子类持有IContainerAware服务。

1. 流程图

ComponentActivatorAbstractBase抽象类逻辑图：



### 14-4 match

1. 简介

该类定义了网络帧/包/消息中的匹配域，属于流表中的一部分。字段有如下：

*IN\_PORT*("inPort", 1 << 0, NodeConnector.**class**, 1, 0),

*DL\_SRC*("dlSrc", 1 << 1, Byte[].**class**, 0, 0xffffffffffffL),

*DL\_DST*("dlDst", 1 << 2, Byte[].**class**, 0, 0xffffffffffffL),

*DL\_VLAN*("dlVlan", 1 << 3, Short.**class**, 1, 0xfff), // 2 bytes

*DL\_VLAN\_PR*("dlVlanPriority", 1 << 4, Byte.**class**, 0, 0x7), // 3 bits

*DL\_OUTER\_VLAN*("dlOuterVlan", 1 << 5, Short.**class**, 1, 0xfff),

*DL\_OUTER\_VLAN\_PR*("dlOuterVlanPriority", 1 << 6, Short.**class**, 0, 0x7),

*DL\_TYPE*("dlType", 1 << 7, Short.**class**, 0, 0xffff), // 2 bytes

*NW\_TOS*("nwTOS", 1 << 8, Byte.**class**, 0, 0x3f), // 6 bits (DSCP field)

*NW\_PROTO*("nwProto", 1 << 9, Byte.**class**, 0, 0xff), // 1 byte

*NW\_SRC*("nwSrc", 1 << 10, InetAddress.**class**, 0, 0),

*NW\_DST*("nwDst", 1 << 11, InetAddress.**class**, 0, 0),

*TP\_SRC*("tpSrc", 1 << 12, Short.**class**, 1, 0xffff), // 2 bytes

*TP\_DST*("tpDst", 1 << 13, Short.**class**, 1, 0xffff); // 2 bytes

1. 类图



1. 基本原理
2. 流程图

### 14-5 flowprogrammer

1. 简介

该类提供了流表的编程接口。包括增加流表，删除流表，修改流表。

1. 类图



1. 基本原理
2. 流程图

## 15 samples

1. 简介

该模块为应用模块，包含loadbalance,northbound,simpleforward 三个app。

## 16 security

1. 简介

控制器用户鉴权模块，判断用户名密码是否正确。在系统配置中tomcat-server.xml中进行了配置。

<Engine name=*"Catalina"* defaultHost=*"localhost"*>

<Host name=*"localhost"* appBase=*""*

unpackWARs=*"false"* autoDeploy=*"false"*

deployOnStartup=*"false"* createDirs=*"false"*>

<Realm className=*"org.opendaylight.controller.security.ControllerCustomRealm"* />

<Valve className="org.apache.catalina.authenticator.SingleSignOn" />

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="web\_access\_log\_" suffix=".txt" resolveHosts="false"

rotatable="true" fileDateFormat="yyyy-MM"

pattern="%{yyyy-MM-dd HH:mm:ss.SSS z}t - [%a] - %r"/>

</Host>

</Engine>

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.slf4j,

javax.servlet,

org.apache.catalina,

org.apache.catalina.connector,

org.apache.catalina.valves,

org.apache.catalina.realm,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.sal.authorization,

org.opendaylight.controller.usermanager

</Import-Package>

<Export-Package>

org.opendaylight.controller.security

</Export-Package>

## 17 staticsmanager

1. 简介

网络节点数据统计的管理模块。提供了获取流统计数据，节点描述信息，节点连接统计信息。

1. 类图



1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.forwardingrulesmanager,

org.opendaylight.controller.containermanager,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.flowprogrammer,

org.slf4j,

org.opendaylight.controller.sal.reader,

org.apache.felix.dm

</Import-Package>

<Bundle-Activator>

org.opendaylight.controller.statisticsmanager.internal.Activator

</Bundle-Activator>

<Export-Package>

org.opendaylight.controller.statisticsmanager

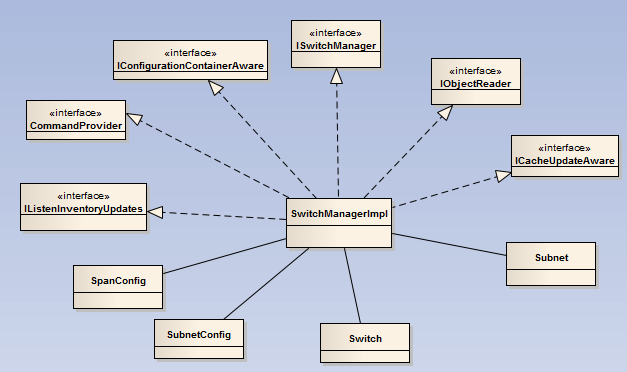
</Export-Package>

## 18 switchmanager

1. 简介

该模块主要维护网路中节点，节点连接器，接入点属性，三层配置，Span配置，节点配置，网络设备标识。提供创建删除查询子网，增加删除查询span端口，节点连接器，配置交换机和span端口功能。北向接口提供获得节点，增加节点属性，删除节点属性，获得节点连接器，增加节点连接器，删除节点连接器，保存交换机配置的功能。

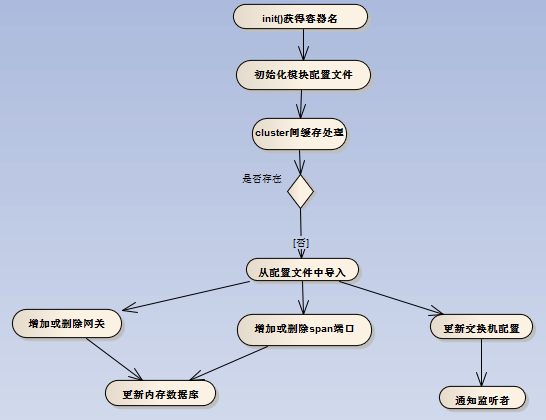
1. 类图



1. 基本原理

当opendaylight控制器收到交换机发送的hello消息时，被org.opendaylight.controller.protocol\_plugin.openflow.core.internal.Controller类里面的switchEventThread线程捕获到，从而建立交换机与控制器的连接，并通知该事件监听者。

1. 流程图

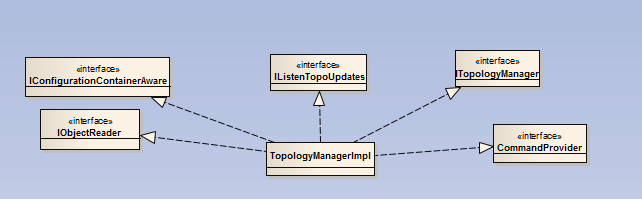


## 19 topologymanager

1. 简介

该模块维护网络拓扑信息，并在拓扑变化时通知其监听者

1. 类图



1. 基本原理

在ITopologyManager接口中提供获得“node”(相当于floodlight中的links)，”NodeConnectorHost“(相当于floodlihgt中的主机设备的attachpoint)，以及它们间的连接关系。由ISwitchManager提供网络中所有的switch.在org.opendaylight.controller.topology.web.Topology类中获得上述数据后，通过CircleLayout类完成拓扑节点的绘制，形成拓扑图。（VNC控制器的拓扑绘制在前台实现）。北向接口提供：

拓扑节点（交换机）和主机的信息。

RESTAPI:

http://localhost:8082/controller/web/topology/visual.json

1. 流程图

## 20 usermanager

1. 简介

用户管理模块。

1. 类图







1. 基本原理
2. 依赖服务

<Import-Package>

org.opendaylight.controller.clustering.services,

org.opendaylight.controller.configuration,

org.opendaylight.controller.sal.authorization,

org.opendaylight.controller.sal.core,

org.opendaylight.controller.sal.packet,

org.opendaylight.controller.sal.utils,

org.opendaylight.controller.switchmanager,

org.opendaylight.controller.containermanager,

org.slf4j,

org.eclipse.osgi.framework.console,

org.osgi.framework,

org.apache.felix.dm,

org.apache.commons.lang3.builder,

org.apache.commons.logging,

javax.servlet,

javax.servlet.http,

org.springframework.security.web.context,

org.springframework.security.core,

org.springframework.security.core.context,

org.apache.commons.lang3,

org.springframework.security.authentication,

org.springframework.security.core.authority,

org.springframework.security.core.userdetails

</Import-Package>

<Export-Package>

org.opendaylight.controller.usermanager,

org.opendaylight.controller.usermanager.internal

</Export-Package>