OpenFlow SouthBound Plugin Proxy

ODL Magnesium DDF, Antwerp - September 2019

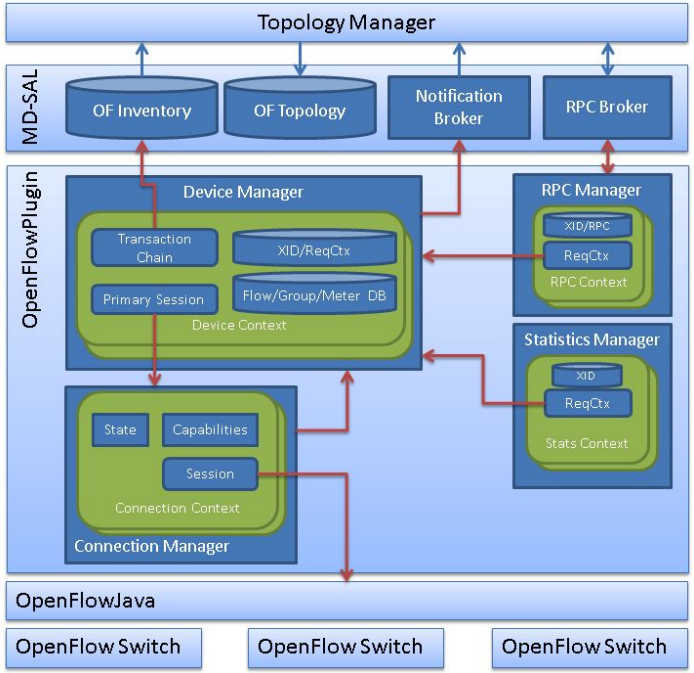
Venkata Satya Jonnadula, Lumina Networks

Agenda

- Why is it required
- What does it provide
- Usage examples

Why is it required

- OpenFlow Plugin is an OpenDaylight south-bound component for managing OpenFlow switches
- Uses OpenFlowJava library to communicate with switches
- Provides switch config management APIs using config DS (inventory model and RPCs)
- Makes operational state available through Operational DS (inventory and topology models)



Why is it required

- App manages OF switches using OF Plugin features
- App can directly consume OF Plugin APIs but it adds complexity and code duplication
- OF Plugin Proxy sits in between App and OF Plugin and provides easy-to-consume APIs that simplifies App development experience
- App usage scenarios
  - Configure switch and waits for the result
  - Register to get notified when interesting OF config on switch changes
  - Handle interdependency between OF config elements
  - Reconcile with switch state during startup

What does it provide

- Device Configuration Service
  - APIs for configuring OF switch
  - Insight on which app/component configured particular resource
  - Opportunity to optimize the device configuration/reconciliation
  - Notification to app/component if resource configuration fails
  - Device flow/group synchronization.
    - Play out missing config flow/group on the device.
    - Cleanup flows/groups from the device that are not present in device config.

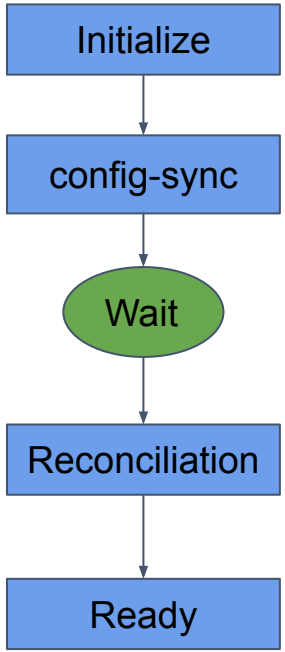
- Resource Change Notification Service
  - Sends change notification only to app/component that configured the resource
  - Optimizes change handling by providing notification about resource only to app/component that cares about the resource
  - Populates change type (config change or external action) in notification
  - App/component can register to receive all (wildcard) notifications

What does it provide

- Device Operational State Management Service
  - Clean device if it reconnects after app becomes ready
  - Clean unprogrammed resources after the reconciliation of all consumer services
  - Clean connected devices if controller recovered from disaster event
- Bootup Initialization Sequencing Service
  - Define and enforce the order of initialization between consumer services during startup
  - Move controller to ready state once all the consumer services are up.
  - In a three node cluster setup
    - Node where the application using south bound proxy is running as master, the device configuration service is enabled. So all the device configuration changes are pushed to device from this node.
    - On the other nodes the device configuration service is disabled. So the writes to device config are not processed by the device configuration service running on these nodes.

What does it provide

- Bootup Initialization Sequencing Service



Controller started.

Controller fetches the statistics from the device and populates the device operational.

Controller is waiting for the user to invoke RPC to move it to next state that is Reconciliation.

The sbproxy first recondilies, making sure the device is in synch with the device config. After that all the other application services reconciles making sure the required service config are pushed to the device. Until the reconciliation is done controller wont accept any new configurations from the user.

Controller is done with reconciliation. Now it is ready to accept new user configurations.

What does it provide

- App Status Service
  - Is controller recovered from disaster?
  - Is application ready for business?
-

Usage examples

-

Thanks