ODL OpenFlowPlugin

Release master

Aug 15, 2018

Contents

This documentation provides information needed to help you write ODL Applications/Projects that can co-exist with other ODL Projects.

Contents:

CHAPTER 1

Openflowplugin Design Specifications

Starting from Nitrogen, Openflowplugin uses RST format Design Specification document for all new features. These specifications are perfect way to understand various Openflowplugin features.

Contents:

Table of Contents

1.1 Reconciliation Framework

Reconciliation Framework Reviews

This feature aims to overcome the drawbacks of the current reconciliation implementation. As part of this enhancement, reconciliation framework will be introduced which will coordinate the reconciliation across various applications.

Applications should register themself with reconciliation framework with a specific priority. Application should decide the priority and the reconciliation framework will use it for executing in an priority.

1.1.1 Problem description

When a switch connected to controller, the current ODL reconciliation implementation pushes all the table/meters/groups/flows from the inventory configuration datastore to the switch.

When the switch is connected, all the applications including FRM(Forwarding Rules Manager) will receive the node added DTCN(Data Tree Change Listener) and starts pushing the flows for the openflow switch. FRM reconciliation will read the data from the config and starts pushing the flows one by one. In the meantime, applications can react to the node added DTCN change and will start pushing the flows through the config DS. With this, there is a high chance the application flow can be overwritten by the old flows by FRM via reconciliation.

With framework, the problem will be avoided by doing the reconciliation for all the registered services including FRM and then the openflow switch will be submitted to the DS. With this, applications won't receive the node added DTCN until registered applications are done with reconciliation for the switch.

The current reconciliation mechanism lacks an ordered execution of tasks across multiple applications resulting in the forwarding plane not correctly reflecting the changes in the control plane. The issue becomes more prominent in case of multi-application scenarios, resulting in errors.

Use Cases

Priority based/Ordered Coordination of Reconciliation across multiple applications.

1.1.2 Proposed change

Reconciliation Framework will be introduced, framework will coordinate the reconciliation across applications. The Openflow switch won't be advertised to application until Openflow switch is in KNOWN state.

`KNOWN state` controller and switch state should be in sync(reconciliation), once the switch connects.

Application participating in reconciliation needs to register with framework.

- Application can either be FRM, FRS or any other application(s).
- Application(s) registering with Reconciliation module is encouraged since: Applications would know the right Flows/Groups/Meters which needs to be replayed (Add/Delete/Update). FRM/FRS(Forwarding Rules Sync) would not have application view of flows/group, it would blindly replay the flows/groups. Also flows having idle/hard timeout can be gracefully handled by application rather than FRM/FRS.

As applications register with reconciliation module

- Reconciliation module maintains the numbers of application registered in an order based on the priority.
- Applications will be executed in the priority order of higher to lower, 1 - Highest n - lowest
- Reconciliation will be triggered as per the priority, applications with same priority will be processed in parallel, once the higher priority application completed, next priority of applications will be processed.

Openflow switch establishes connections with openflowplugin.

- Openflow switch sends connection request.
- Openflowplugin accepts connection and than establishes the connection.

Openflowplugin after establishing the connection with openflow switch, elects the mastership and invokes reconciliation framework through ReconciliationFrameworkEvent onDevicePrepared.

- Before invoking the reconciliation API, all the RPCs are registered with MD-SAL by openflowplugin.
- Reconciliation framework will register itself with the MastershipChangeServiceManager.

All registered applications would be indicated to start the reconciliation. * DeviceInfo would be passed for the API/Event and it contains all the information needed by application.

Application(s) would than fetch the flows / groups for that particular Node, which needs to be replayed.

Application(s) would than replay the selected flows / group on to the switch.

Application(s) would also wait for error from switch, for pre-defined time.

Application(s) would inform the reconciliation status to reconciliation module.

Reconciliation framework would co-relate result status from all the applications and decides the final status. If success, framework will report back DO_NOTHING and in case of failure it will be DISCONNECT.

Based on result state, openflowplugin should do the following

- On success case, openflowplugin should continue with the openflow switch –> write the switch to the operational datastore.
- On failure case, openflowplugin should disconnect the openflow switch.
- When the switch reconnects, the same steps will be followed again.

When there is a disconnect/mastership change while the reconciliation is going on, openflowplugin should notify the framework and the framework should halt the current reconciliation.

1.1.3 Implementation Details

Following new interface will be introduced from Reconciliation framework (RF).

- ReconciliationManager
- ReconciliationNotificationListener

ReconciliationManager

```
/* Application who are interested in reconciliation should use this API to register␣
↪themself to the RF */
/* NotificationRegistration will be return to the registered application, who needs␣
↪to take of closing the registration */
NotificationRegistration registerService(ReconciliationNotificationListener object);

/* API exposed by RF for get list of registered services */
Map<Integer, List<ReconciliationNotificationListener>> getRegisteredServices();
```

ReconciliationNotificationListener

```
/* This method will be a callback from RF to start the application reconciliation */
ListenableFuture<Boolean> startReconciliation(DeviceInfo deviceInfo);

/* This method will be a callback from RF when openflow switch disconnects during␣
↪reconciliation */
ListenableFuture<Boolean> endReconciliation(DeviceInfo deviceInfo);

/* Priority of the application */
int getPriority();

/* Name of the application */
String getName();

/* Application's intent when the application's reconciliation fails */
ResultState getResultState();
```

Priority

Framework will maintain the list of registered applications in an order based on the priority. Applications having the same priority will be executed in parallel and once those are done. Next priority applications will be called. Consider

2 applications, A and B. A is handling of programming groups and flows and B is handling of programming flows which is dependent of the groups programmed by A. So, B has to register with lower priority than A.

Application don't do any conflict resolution or guarantee any specific order among the application registered at the same priority level.

Result State - Intent Action

When the application fails to reconcile, what is the action that framework should take.

- DO_NOTHING - continue with the next reconciliation
- DISCONNECT - disconnect the switch (reconciliation will start again once the switch connects back)

Name

Name of the application who wants to register for reconciliation

ReconciliationNotificationListener

Applications who wants to register should implement ReconciliationNotificationListener interface.

- ReconciliationNotificationListener having api's like startReconciliation and endReconciliation
- startReconciliation –> applications can take action to trigger reconciliation
- endReconciliation –> application can take action to cancel their current reconcile tasks

1.1.4 Command Line Interface (CLI)

CLI interface will be provided to get all the registered services and their status

- List of registered services
- Status of each application for respective openflow switch

1.1.5 Other Changes

Pipeline changes

None.

Yang changes

None

Configuration impact

None

Clustering considerations

None

Other Infra considerations

N.A.

Security considerations

None.

Scale and Performance Impact

None.

Targeted Release

Nitrogen.

Alternatives

N.A.

1.1.6 Usage

Features to Install

Will be updated

REST API

None

CLI

None

1.1.7 Implementation

Assignee(s)

Primary assignee:

- Prasanna Huddar <prasanna.k.huddar@ericsson.com>
- Arunprakash D <d.arunprakash@ericsson.com>

- Gobinath Suganthan <gobinath@ericsson.com>

Other contributors:

Work Items

N.A.

1.1.8 Dependencies

This doesn't add any new dependencies.

1.1.9 Testing

Capture details of testing that will need to be added.

Unit Tests

None

Integration Tests

None

CSIT

None

1.1.10 Documentation Impact

This feature will not require any change in User Guide.

1.1.11 References

[1] Openflowplugin reconciliation enhancements

Table of Contents

1.2 Group Command OFPGC_ADD_OR_MOD support

Group ADD-MOD Reviews

This spec addresses following enhancement in Openflowplugin module:

Addition of new command OFPGC_ADD_OR_MOD for OFPT_GROUP_MOD message that adds a new group that does not exist (like ADD) or modifies an existing groups (like MODIFY).

OFPGC_ADD_OR_MOD group command will be supported only for OVS2.6 and above.

1.2.1 Problem description

In OpenFlow 1.x the Group Mod commands OFPGC_ADD and OFPGC_MODIFY have strict semantics: ADD fails if the group exists, while MODIFY fails if the group does not exist. This requires a controller to exactly know the state of the switch when programming a group in order not run the risk of getting an OFP Error message in response. This is hard to achieve and maintain at all times in view of possible switch and controller restarts or other connection losses between switch and controller.

Due to the un-acknowledged nature of the Group Mod message programming groups safely and efficiently at the same time is virtually impossible as the controller has to either query the existence of the group prior to each Group Mod message or to insert a Barrier Request/Reply after every group to be sure that no Error can be received at a later stage and require a complicated roll-back of any dependent actions taken between the failed Group Mod and the Error.

Reconciliation

The current implementation of reconciliation is to read the complete set of groups from config inventory and start pushing the groups one by one. This will always end up in GROUP_ALREADY_EXITS error as the reconciliation will always send GROUP ADD.

This can be avoided by reading the groups from switch and compare with the list from inventory config and push only the delta. This is an overhead comparision and can be simply avoided by updating the group command as OFPGC_ADD_OR_MOD.

Use Cases

1. Normal group provisioning via FRM: ADD/UPDATE group should send new command OFPGC_ADD_OR_MOD.
2. Reconciliation of groups should send OFPGC_ADD_OR_MOD. Current implementation of openflowplugin will always send group add OFPGC_ADD irrespective of the state of the switch. This results in failure with GROUP_ALREADY_EXISTS error.

Proposed change

The implementation of OFPGC_ADD_OR_MOD command is specific to OVS2.6 and above and the same can be extended to other openflow switch based on the group command support by them.

New configuration parameter will be introduced in default-openflow-connection-config.xml and legacy-openflow-connection-config.xml, which can be modified by users to enable the GROUP ADD MOD support.

Listing 1: default(legacy)-openflow-connection-config.xml

```
<group-add-mod-enabled>false</group-add-mod-enabled>
```

By default the group-add-mod-enabled flag will be kept as false, which means existing group mod commands OFPGC_ADD/OFPGC_MODIFY will be used.

GroupMessageSerializer will use the above flag to determine which group command should be set for group add/update. The above class is applicable for single layer serialization and the for multi-layer serialization changes will be done in openflowjava GroupModInputMessageFactory java classs.

When flag is enabled, openflowplugin will always send OFPGC_ADD_OR_MOD (32768) for both group add and modify.

Pipeline changes

None

Yang changes

Below yang changes will be done in order to provide configuration support for group-add-mod-enabled field.

Listing 2: openflow-switch-connection-config.yang

```
leaf group-add-mod-enabled {
     description "Group Add Mod Enabled";
     type boolean;
     default false;
}
```

Configuration impact

None

Clustering considerations

None

Other Infra considerations

None

Security considerations

None

Scale and Performance Impact

Unknown

Targeted Release

Oxygen

Alternatives

None

1.2.2 Usage

No external rpc/api will be provided. The implementation is internal to openflowplugin.

User can enable OFPGC_ADD_OR_MOD by changing the value to true in below files,

Listing 3: default(legacy)-openflow-connection-config.xml

```
default-openflow-connection-config.xml  <group-add-mod-enabled>false</group-add-mod-
↪enabled>
legacy-openflow-connection-config.xml   <group-add-mod-enabled>false</group-add-mod-
↪enabled>
```

REST API

No new REST API is being added.

CLI

No new CLI being added.

1.2.3 Implementation

Assignee(s)

Primary assignee: Arunprakash D <d.arunprakash@ericsson.com>

Other contributors: Gobinath Suganthan <gobinath@ericsson.com>

Work Items

- Implementation of GROUP ADD MOD support
- Addition of configuration flag to enable/disable group add mod command

1.2.4 Dependencies

No new dependencies.

1.2.5 Testing

Unit Tests

1. Verify group provisioning via FRM with group-add-mod-supported disabled
2. Verify group provisioning via FRM with group-add-mod-supported enabled
3. Verify reconciliation via FRM with with group-add-mod-supported disabled
4. Verify reconciliation via FRM with with group-add-mod-supported enabled

CSIT

CSIT test cases will be added in future

1.2.6 Documentation Impact

None

1.2.7 References

Openvswitch ADD_OR_MOD

Table of Contents

1.3 Openflow Bundle Reconciliation

Bundle Reconciliation Review

This spec addresses following enhancement in Openflowplugin module:

Addition of new reconciliation mechanism in openflowplugin using openflow bundles.

Bundle reconciliation will be supported from OVS2.6 and above.

1.3.1 Problem description

Current reconciliation mechanism exists in FRM will read the config inventory data and push all the groups and flows via group and flow add messages and this mechanism is having the following limitations,

1. Group add during reconcilation will fail with GROUP_ALREADY_EXISTS error
2. Stale flows won't be removed from openflow switch after reconciliation. This leads to stale flow aggregation after every controller version upgarde.
3. Datapath traffic will get impacted as the flows will get replaced during reconciliation window.

Bundle Reconciliation

Reconciliation using openflow bundles will overcome all the above mentioned limitations. Mainly there will be minimal or no datapath traffic hit.

Bundle Concepts

A bundle is a sequence of OpenFlow requests from the controller that is applied as a single OpenFlow operation. The first goal of bundles is to group related state changes on a switch so that all changes are applied together or that none of them is applied. The second goal is to better synchronise changes across a set of OpenFlow switches, bundles can be prepared and pre-validated on each switch and applied at the same time.

A bundle is specified as all controllers messages encoded with the same bundle_id on a specific controller connection. Messages part of the bundle are encapsulated in a Bundle Add message, the payload of the Bundle Add message is formatted like a regular OpenFlow messages and has the same semantic. The messages part of a bundle are pre-validated as they are stored in the bundle, minimising the risk of errors when the bundle is applied. The applications of the message included in the Bundle Add message is postponed to when the bundle is committed.

A switch is not required to accept arbitrary messages in a bundle, a switch may not accept some message types in bundles, and a switch may not allow all combinations of message types to be bundled together. For example, a switch should not allow to embed a bundle message within a Bundle Add message. At a minimum, a switch must be able to support a bundle of multiple flow-mods and port-mods in any order.

When a bundle is opened, modifications are saved into a temporary staging area without taking effect. When the bundle is committed, the changes in the staging area are applied to the state (e.g. tables) used by the switch. If an error occurs in one modification, no change is applied to the state.

Use Cases

1. Reconciliation using openflow bundles when controller restarts
2. Reconciliation using openflow bundles when openflow switch restarts

Proposed change

Bundle reconciliation will be supported by ovs2.6 and above version or any openflow switch with bundles support.

Bundle reconciliation will be disabled by default and user has to manually enable it when needed by making a configuration change. New configuration parameter will be introduced in openflowplugin.cfg to support the same.

Listing 4: openflowplugin.cfg

```
#
# Bundle reconciliation can be enabled by making this flag to true.
# By default bundle reconciliation is disabled and reconciliation happens
# via normal flow/group mods.
# NOTE: This option will be effective with disable_reconciliation=false.
#
# bundle-based-reconciliation-enabled=false
```

By default bundle-based-reconciliation-enabled flag will be kept as false, which means reconciliation will happen via flow/group mod commands.

Following steps will be executed in order to achieve bundle reconciliation,

1. Send open bundle message to the openflow switch
2. Send delete all flows bundle message
3. Send delete all groups bundle message
4. Read flows and groups from config inventory
5. Push groups via bundle message
6. Push flows via bundle message
7. Send commit bundle message to the openflow switch

Pipeline changes

None

Yang changes

Below yang changes will be done in order to provide configuration support for bundle-based-reconciliation-enabled field.

Listing 5: forwardingrules-manager-config.yang

```
leaf bundle-based-reconciliation-enabled {
     type boolean;
     default false;
}
```

Configuration impact

None

Clustering considerations

None

Other Infra considerations

None

Security considerations

None

Scale and Performance Impact

Unknown

Targeted Release

Oxygen

Alternatives

None

1.3.2 Usage

No external rpc/api will be provided. The implementation is internal to openflowplugin.

User can enable bundles reconciliation by changing the value to true in openflowplugin.cfg

Listing 6: openflowplugin.cfg

```
#
# Bundle reconciliation can be enabled by making this flag to true.
# By default bundle reconciliation is disabled and reconciliation happens
# via normal flow/group mods.
# NOTE: This option will be effective with disable_reconciliation=false.
#
bundle-based-reconciliation-enabled=true
```

REST API

No new REST API is being added.

CLI

No new CLI being added.

1.3.3 Implementation

Assignee(s)

Primary assignee: Arunprakash D <d.arunprakash@ericsson.com>

Other contributors: Sunil Kumar G <sunil.g.kumar@ericsson.com>

Suja T <suja.t@ericsson.com>

Work Items

- Implementation of bundle reconciliation
- Addition of configuration flag to enable/disable bundle reconciliation

1.3.4 Dependencies

No new dependencies.

1.3.5 Testing

Unit Tests

1. Verify bundle reconciliation for controller restart
2. Verify bundle reconciliation for openflow switch restart

CSIT

CSIT test cases will be added in future

1.3.6 Documentation Impact

None

1.3.7 References

[1] https://wiki.opendaylight.org/view/OpenDaylight_OpenFlow_Plugin:Bundles_extension_support

[2] https://wiki.opendaylight.org/view/OpenDaylight_OpenFlow_Plugin:Bundles_usage

Table of Contents

1.4 Southbound CLI

Southbound CLI Reviews

This spec addresses following enhancement in Openflowplugin module:

Addition of new Karaf feature *odl-openflowplugin-app-southbound-cli* under openflowplugin module that provides useful CLIs for users. This feature won't be part of any existing openflowplugin feature and user needs to explicitly install it in addition to the existing features.

1.4.1 Problem description

Currently there is no way of getting the formatted list of openflow nodes connected to the OpenDaylight controller. User has to fetch operational inventory using Restconf and search for all the connected nodes. Even to get the list of ports available under a OpenFlow node, user need to search the entire inventory dump. From user experience perspective it's not really very helpful, and at scale fetching the entire inventor from data store can cause CPU spike for the controller because of the huge data present under inventory tree.

1.4.2 Southbound CLI

New Karaf feature is developed that will provide command line interface to the user using which user can retrieving the list of connected OpenFlow nodes and the ports available under each OpenFlow node.

Use Cases

- List of all OpenFlow node(s) connected to the OpenDaylight controller in either standalone or cluster environment.
- List ports information available under a connected OpenFlow node

Proposed change

New karaf feature *odl-openflowplugin-app-southbound-cli* will be added and it will not be part of any existing openflowplugin feature. User will have to explicitly install the feature to get the available CLIs.

Following 2 CLIs will be added:

- *openflow:getallnodes*
- *openflow:shownode*

openflow:getallnodes will display information like NodeId and NodeName(datapath description) for all the connected nodes.

openflow:shownode will display information like NodeId, NodeName(datapath description) and Ports for a given openflow node.

Yang changes

None

Targeted Release

Oxygen

Alternatives

Use RestConf to fetch entire operational inventory and parse through it.

1.4.3 Usage

Install `odl-openflowplugin-app-southbound-cli` feature as it is not part of any existing openflowplugin features.

List the connected openflow nodes under odl controller either in standalone or cluster environment. In clustered environment user need to install this feature on all the three nodes if it wants to use any node to run these CLI commands, but user also can choose to install it on a dedicated node only if that's the master node to run CLI commands. This feature can be install at any point of time during or after controller start.

Listing 7: openflow:getallnodes

```
opendaylight-user@root>openflow:getallnodes
Number of nodes: 1
NodeId              NodeName

--------------------------------------------------------------------------
137313212546623     None
```

List the available ports under openflow node.

Listing 8: openflow:shownode

```
opendaylight-user@root>openflow:shownode -d 137313212546623
OFNode                   Name                    Ports
----------------------------------------------------------------------------------
137313212546623          None                    br-int
```

1.4.4 Implementation

Assignee(s)

Primary assignee: * Arunprakash D <d.arunprakash@ericsson.com>

Contributors: * Gobinath Suganthan <gobinath@ericsson.com>

Work Items

- Implementation of cli to list the connected openflow nodes across standalone or clustered environment.
- Implementation of cli to list the ports available under openflow node.

1.4.5 Dependencies

No new dependencies.

1.4.6 Testing

Unit Tests

1. Verify CLI to list all the connected openflow nodes
2. Verify CLI to list all the ports under openflow node

CSIT

None

1.4.7 Documentation Impact

None

1.4.8 References

None

Table of Contents

1.5 Reconciliation CLI and Alarm

This spec addresses following enhancement in Openflowplugin module:

Addition of user triggered reconciliation via karaf cli command or rpc in Openflowplugin.

1.5.1 Problem description

Whenever there is a state (flow/group) mismatch between config inventory and Openflow switch, user has to either restart the Openflow switch or odl controller. This will sync the state again between odl controller and Openflow switch.

1.5.2 Reconciliation

User can trigger reconciliation to sync the state between controller and Openflow switch. It can be done either via karaf cli command or rest rpc.

1.5.3 Reconciliation Alarm

Reconciliation alarm will be generated whenever user trigger the reconciliation via cli command or rest rpc and the same will be cleared once reconciliation is completed.

Use Cases

1. Trigger reconciliation for a single Openflow switch
2. Trigger reconciliation for a list of Openflow switch
3. Trigger reconciliation for all the connected Openflow switches
4. Raise alarm whenever user triggers reconciliation for a Openflow switch
5. Clear the alarm when the reconciliation completed for a Openflow swtich

Proposed change

Karaf CLI command will be added to trigger reconciliation for the given Openflow nodes. Rest rpc will be exposed to trigger reconciliation for the given Openflow nodes.

Feature *odl-openflowplugin-app-southbound-cli* should be installed in order to get these karaf cli and rest rpc. This feature is not part of any existing openflowplugin features and has to be installed explicitly by user.

Ref: Southbound CLI

Below two CLIs will be added,

- openflow:reconcile
- openflow:getreconciliationcount

Pipeline changes

None

Yang changes

Listing 9: reconciliation.yang

```
container reconciliation-counter {
    description "Number of reconciliation triggered for openflow nodes";
    config false;
    list reconcile-counter {
        key node-id;
        uses counter;
    }
}

grouping counter {
    leaf node-id {
        type uint64;
    }
    leaf success-count {
        type uint32;
```

(continues on next page)

(continued from previous page)

```
        default 0;
    }
    leaf failure-count {
        type uint32;
        default 0;
    }
    leaf last-request-time {
        description "Timestamp when reconciliation was last requested";
        type string;
    }
}

container reconciliation-state {
    description "Reconciliation state for the given openflow nodes";
    config false;
    list reconciliation-state-list {
        key node-id;
        uses node-reconcile-state;
    }
}

grouping node-reconcile-state {
    leaf node-id {
        type uint64;
    }
    leaf state {
        description "Expresses the current state of the reconcile on a specific NODE";
        type enumeration {
            enum IN_PROGRESS;
            enum COMPLETED;
            enum FAILED;
        }
    }
}

rpc reconcile {
    description "Request the reconciliation for given device or set of devices to the␣
↪controller."
    input {
        leaf-list nodes {
            description "List of nodes to be reconciled";
            type uint64;
        }
        leaf reconcile-all-nodes {
            description "Flag to indicate that all nodes to be reconciled";
            type boolean;
            mandatory false;
            default false;
        }
    }
    output {
        leaf result {
            type boolean;
        }
        leaf-list inprogress-nodes {
            description "List of nodes that are already in reconciling mode";
            type uint64;
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

Targeted Release

Flourine

Alternatives

Disconnect the device from controller and reconnect or restart the controller.

REST API

- POST: http://localhost:8181/restconf/operations/reconciliation:reconcile
- GET: http://localhost:8181/restconf/operational/reconciliation:reconciliation-counter

1.5.4 Usage

Install `odl-openflowplugin-app-southbound-cli` feature.

CLI:

Trigger reconciliation for a connected openflow node via cli command `openflow:reconcile`.

Listing 10: openflow:reconcile

```
opendaylight-user@root>openflow:reconcile 244711506862915
reconcile successfully completed for the nodes
```

Trigger reconciliation for all the connected openflow nodes via cli command `openflow:reconcile -all`.

Listing 11: openflow:reconcile -all

```
      opendaylight-user@root>openflow:reconcile -all
      reconcile successfully completed for the nodes
```

Get details about number of times user triggered reconciliation for openflow nodes via `openflow:getreconciliationcount`.

Listing 12: openflow:getreconciliationcount

```
opendaylight-user@root>openflow:getreconcilecount
NodeId              ReconcileSuccessCount     ReconcileFailureCount    ␣
↪LastReconcileTime
---------------------------------------------------------------------------------------
↪----------
244711506862915     2                         0                         2018-06-
↪06T11:51:51.989
```

REST:

Trigger reconciliation for a single datapath node.

Listing 13: http://localhost:8181/restconf/operations/reconciliation:reconcile

```
POST /restconf/operations/reconciliation:reconcile
{
  "input" :  {
    "nodes":["244711506862915"]
  }
}
```

Get reconciliation counter details

Listing 14: http://localhost:8181/restconf/operational/reconciliation:reconciliation-counter

```
GET /restconf/operational/reconciliation:reconciliation-counter

OUTPUT:
=======
Request URL
http://localhost:8181/restconf/operational/reconciliation:reconciliation-counter

Response Body
{
  "reconciliation-counter": {
    "reconcile-counter": [
      {
        "node-id": 244711506862915,
        "success-count": 4,
        "last-request-time": "2018-06-06T12:09:53.325"
      }
    ]
  }
}
```

Trigger reconciliation for a openflow switch using routed rpc. This rpc will be exposed without installing southbound-cli feature and user can trigger reconciliation for the given Openflow node. This will not affect the counter and alarm.

Listing 15: http://localhost:8181/restconf/operations/reconciliation:reconcile-node

```
POST /restconf/operations/reconciliation:reconcile-node
{
  "input": {
    "nodeId": "244711506862915",
    "node": "/opendaylight-inventory:nodes/opendaylight-inventory:node[opendaylight-
↪inventory:id='openflow:244711506862915']"
  }
}

Request URL
http://localhost:8181/restconf/operations/reconciliation:reconcile-node
```

(continues on next page)

(continued from previous page)

```
Response Body
{
  "output": {
    "result": true
  }
}
```

1.5.5 Implementation

Assignee(s)

Primary assignee:

- Arunprakash D <d.arunprakash@ericsson.com>

Contributors:

- Suja T <suja.t@ericsson.com>
- Somashekhar Javalagi <somashekhar.manohara.javalagi@ericsson.com>

Work Items

- Implementation of cli to trigger reconciliation for openflow node(s).
- Implementation of reconciliation alarm for user triggered reconciliation.

1.5.6 Dependencies

No new dependencies.

1.5.7 Testing

Unit Tests

1. Verify reconciliation for single openflow node
2. Verify reconciliation for list of openflow nodes
3. Verify reconciliation for all the openflow nodes
4. Verify reconciliation alarm generated for user triggered reconciliation node
5. Verify reconciliation alarm cleared once the reconciliation completed

CSIT

None

1.5.8 Documentation Impact

None

1.5.9 References

None

Table of Contents

1.6 Arbitrator Reconciliation using OpenFlow bundle

Arbitrator Reconciliation Reviews

This spec addresses following enhancement in openflowplugin module:

Addition of new reconciliation mode in openflowplugin which will allow applications to program flow/group within reconciliation window instead of frm reads and pushes the configuration down to the openflow switch.

AUTONOMOUS mode implies that Openflowplugin shall perform reconciliation autonomously as it does now without any change in the workflow - ie. Switch-connection-handling followed by flow-based/bundle-based reconciliation execution followed by publishing of switch to the Inventory Operational datastore. This will be the default mode until arbitrated mode is enabled.

ARBITRATED mode implies that the default openflowplugin reconciliation will be disabled and consumer application will have to initiate and complete the reconciliation including any error-handling. In the current implementation ARBITRATED mode will only supported bundle based reconciliation.

Openflowplugin will switch to arbitrator reconciliation mode based on the upgradeState provided by ServiceUtils.

1.6.1 Problem description

During replay based upgrade, the inventory configuration DS will be empty and applications has to program flows/groups based on the configuration pushed by user or external orchestrator. These new configurations has to applied on the switch without datapath disruption.

This can be achieved using OpenFlow bundles. Bundle is a sequence of OpenFlow requests from odl controller that switch will apply in atomic transaction.

Use Cases

Application controlled reconciliation of OpenFlow devices after controller re/start.

Proposed change

Arbitrator Reconciliation using bundles support will be provided. Openflowplugin will switch to arbitrator reconciliation based on the upgradeState provided by ServiceUtils. Orchestrator can enable or disable this mode as per their deployment requirements.

upgradeInProgress presents in ServiceUtils project and can be changed to true to enable arbitrator reconciliation.

Listing 16: serviceutils-upgrade-config.xml

```
<upgrade-config xmlns="urn:opendaylight:serviceutils:upgrade">
    <upgradeInProgress>false</upgradeInProgress>
</upgrade-config>
```

ArbitratorReconciliation module registers itself with reconciliation framework with priority 1.

When OpenFlow switch connect event received by Openflowplugin, it notifies Reconciliation Framework(RF).

FlowNode Reconciliation will be notified first by RF as it registered with higher priority. FlowNode reconciliation module is the one responsible for reconciliation of OpenFlow node. It can be done either via flow/group based or OpenFlow bundle based.

When upgradeInProgress is set to true, FlowNode reconciliation will be skipped as the config datastore will be empty and return success to the RF.

RF callbacks Arbitrator Reconcilition to executes its task.

Arbitrator Reconcilition will do the following steps in arbitrator-reconcilition(upgradeInProgress) mode

- Open OpenFlow bundle on the connected OpenFlow switch and stores the bundle id in the local cache
- Send delete-all groups and delete-all flows message to the opened bundle in the OpenFlow switch

NOTE: Above clean up step is needed during upgrade to clean the previous version controller states, but the real switch clean-up will only happen when controller will commit the bundle.

Arbitrator Reconciliation module sends success to RF if the previous steps are successful or it sends failure.

RF notifies Openflowplugin with the completion state.

- Success: Openflowplugin writes the OpenFlow node information into operational inventory datastore.
- Failure: OpenFlow node will be disconnected and all the above steps will be repeated on the next reconnect till the mode is in arbitrator reconciliation

Consumer application listening to inventory data store will receive Node added, Port status Data Tree Change Notification(DTCN) from data store.

Applications programs flows and groups into config inventory datastore and Forwarding Rules Manager(FRM) application in in Openflowplugin receives DTCN from config inventory for the flows and groups.

Arbitrator Reconciliation exposes rpc to get Active bundle id for the OpenFlow node.

FRM Flow/Group Forwarder invokes get-active-bundle rpc and gets the bundle id.

GetActiveBundle will executes the following steps.

- Check if bundle commit is in progress for the requested node, if yes wait on commit bundle future
- Returns Active bundle id and the same will be used by FRM forwarder to push the configuration via bundle add messages.
- This call will return null in case of arbitrator-reconciliation disabled and FRM will push the configuration via normal Flow/Group messages.

Listing 17: arbitrator-reconcile.yang

```
rpc get-active-bundle {
    description "Fetches the active available bundle in openflowplugin";
    input {
        uses "inv:node-context-ref";
        leaf node-id {
            description "Node for which the bundle active has to be fetched";
            type uint64;
        }
    }
    output {
        leaf result {
            description "The retrieved active bundle for the node";
            type "onf-ext:bundle-id";
        }
    }
}
```

Routed RPC will be exposed for committing the bundle on a specified Openflow node. It's orchestrator responsibility to commit the bundle across connected OpenFlow node. Configurations will be pushed only via OpenFlow bundles till the commit bundle rpc is invoked.

Listing 18: arbitrator-reconcile.yang

```
rpc commit-active-bundle {
    description "Commits the active available bundle for the given node in␣
↪openflowplugin";
    input {
        uses "inv:node-context-ref";
```

(continues on next page)

(continued from previous page)

```
        leaf node-id {
            description "Node for which the commit bundle to be executed";
            type uint64;
        }
    }
    output {
        leaf result {
            description "Success/Failure of the commit bundle for the node";
            type boolean;
        }
    }
}
```

Consumer application calls commit-active-bundle rpc with OpenFlow node id

- It commits the current active bundle on the OpenFlow node and stores the future till it gets completed.
- When bundle commit is in progress, configuration pushed via config datastore will be blocked on the commit future. This will make sure the new configuration is not lost during the transient state. The logic during arbitrator reconciliation will clear all the existing flows and groups and programs the new configuration and if we allow the flow programming during commit bundle phase, we might loose the new configuration.
- When commit bundle is done, it will return the rpc result to the orchestrator and removes the future from the cache.
- Subsequent flow/group provisioning will be done via flow-mod/group-mod messages.
- Orchestrator can decide further actions based on the rpc result.

Once commit bundle executes on all the connected OpenFlow switch, orchestrator can disable the arbitrator reconciliation by invoking rest rpc call on ServiceUtils *http://localhost:8383/restconf/config/odl-serviceutils-upgrade:upgrade-config/*.

Subsequent OpenFlow switch connect/re-connect will go through FlowNode reconciliation.

Note: There is no bundle timeout logic available as of now and the same will be added in future and will be kept as configurable parameter by user.

Pipeline changes

None

Yang changes

Below yang changes will done to enable arbitrator reconciliation.

RPC will be exposed to get current active bundle id for the given openflow node.

Listing 19: arbitrator-reconcile.yang

```
rpc get-active-bundle {
    description "Fetches the active available bundle in openflowplugin";
    input {
        uses "inv:node-context-ref";
        leaf node-id {
            description "Node for which the bundle active has to be fetched";
            type uint64;
```

(continues on next page)

(continued from previous page)

```
        }
    }
    output {
        leaf result {
            description "The retrieved active bundle for the node";
            type "onf-ext:bundle-id";
        }
    }
}
```

RPC will be exposed for external application/user/consumer applications to commit the active bundle for OpenFlow switch.

Listing 20: arbitrator-reconcile.yang

```
rpc commit-active-bundle {
    description "Commits the active available bundle for the given node in␣
↪openflowplugin";
    input {
        uses "inv:node-context-ref";
        leaf node-id {
            description "Node for which the commit bundle to be executed";
            type uint64;
        }
    }
    output {
        leaf result {
            description "Success/Failure of the commit bundle for the node";
            type boolean;
        }
    }
}
```

Configuration impact

None

Clustering considerations

User can fire the commit-bundle rpc call to any controller node in the cluster. This rpc will only be executed by the node that currently be owning the device.

Other Infra considerations

None

Security considerations

None

Scale and Performance Impact

Unknown

Targeted Release

Flourine

Alternatives

Default reconciliation will be used or application can just reconfigure all the configuration using the normal flow/group add/remove process.

1.6.2 Usage

None

REST API

Listing 21: http://localhost:8181/restconf/operations/arbitrator-reconcile:get-active-bundle

```
Output:
======
{
 "output": {1}
}
```

Listing 22: http://localhost:8181/restconf/operations/arbitrator-reconcile:commit-bundle-node

```
{
   "input": {
      "node": "/opendaylight-inventory:nodes/opendaylight-inventory:node[opendaylight-
↪inventory:id='openflow:<OpenFlow datapath id>']",
      "node-id": "<OpenFlow datapath id>"
   }
}
```

CLI

None.

1.6.3 Implementation

Assignee(s)

Primary assignee:

Arunprakash D <d.arunprakash@ericsson.com>

Gobinath Suganthan <gobinath@ericsson.com>

Muthukumaran K <muthukumaran.k@ericsson.com>

Work Items

- Implementation of arbritrator reconcile module
- Changes in FRM for flow/group programming via openflow bundle
- Read reconciliation mode(upgradeInProgress) from service utils
- Expose RPC to commit bundle for a given OpenFlow node

1.6.4 Dependencies

No new dependencies.

1.6.5 Testing

Unit Tests

CSIT

1.6.6 Documentation Impact

None

1.6.7 References

Bundle Extension Support