OpenDaylight OpenFlow Plugin

- Abhijit Kumbhare, Principal Architect, Ericsson; Project Lead
- Anil Vishnoi, Sr. Staff Software Engineer, Brocade
- Jamo Luhrslen, Sr. Software Engineer, Red Hat

#ODSummit
Agenda

• Project Overview
• High level architecture
• OpenFlow plugin example use case
• Lithium accomplishments
• Plan for Beryllium
• Potential areas for contribution
• References
• Q & A
Agenda

• Project Overview
• High level architecture
• OpenFlow plugin example use case
• Lithium accomplishments
• Plan for Beryllium
• Potential areas for contribution
• References
• Q & A
Project Overview

• Inception in Hydrogen Release
• One of the first community projects
  • Past & Present Participants from Brocade, Cisco, Ericsson, HP, IBM, Red Hat, TCS, etc.
• Meetings: Mondays 9 am Pacific
• Number of commits: ~950
• Source code: 160 KLoCs
• Number of contributors (w/ at least one commit): 60
• Bugs fixes to-date (resolved/verified and fixed): 313
Where does it fit in OpenDayLight?

OpenFlow Plugin is a key offset 1 project
Consumers include OVSDB, GBP, SFC, VTN, VPN, L2 switch, etc.
Agenda

• Project Overview
• High level architecture
• OpenFlow plugin example use case
• Lithium accomplishments
• Plan for Beryllium
• Potential areas for contribution
• References
• Q & A
..well, this is how yang rpc/notifications really works
Agenda

- Project Overview
- High level architecture
- OpenFlow plugin example use case
- Lithium accomplishments
- Plan for Beryllium
- Potential areas for contribution
- References
- Q & A
OpenFlow plugin example use case: OVSDB Project

OpenFlow Plugin Services consumed by OVSDB:
- OpenFlow node connectivity
- Flow Installation, modification & removal
- Nicira extensions
- Packet-in
Agenda

• Project Overview
• High level architecture
• OpenFlow plugin example usecase
• Lithium accomplishments
• Plan for Beryllium
• Potential areas for contribution
• References
• Q & A
Lithium accomplishments

- Migration of OpenFlow Yang models
- Migration of OpenFlow applications
- Alternate design for performance improvement
- Addition of new features
- Integration / CI testing improvements
Migration of OpenFlow Yang models

Migrated following OpenFlow specific models from controller project to OpenFlow plugin project:

- `model-flow-base`
- `model-flow-service`
- `model-flow-statistics`

Why it’s done:

- To have all the OpenFlow specific models at one place to avoid any confusion for the developers.
- Avoid maintenance overhead of managing the relevant pieces in two different project

What’s the impact on consumer:

- No major impact

Backward compatibility

- No impact

Stability impact:

- Improved project maintenance
Migration of OpenFlow applications

Migrated following OpenFlow specific applications (NSF) from controller project to OpenFlow plugin project:

- forwarding rule manager
- statistics manager
- inventory manager
- topology manager

Why it’s done:

- To have all the OpenFlow specific NSF at one place to avoid any confusion for the developers.
- Avoid maintenance overhead of managing the relevant pieces in two different project
- Avoid gerrit patch dependencies

What’s the impact on consumer:

- No major impact

Backward compatibility

- No impact

Stability impact:

- Improved project maintenance
Alternate design for performance improvement

New performance improvement design proposal [4] was implemented.

Why it’s done:
  • To improve the performance, stability and user experience

What’s the impact on consumer:
  • Should be transparent in most cases

Current Status
  • Both existing design (a.k.a. Helium design) and alternate design (a.k.a. Lithium design) are available as options
    • Existing design: features-openflowplugin
      • OpenFlow Plugin consumers currently use this
    • Alternate design: features-openflowplugin-li
## Existing Design / Alternate Design Quick Comparison (Partial)

<table>
<thead>
<tr>
<th></th>
<th>Existing</th>
<th>Alternate</th>
<th>Details of change</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>No significant changes</td>
<td>not supported notifications (except packetIn), statistics rpc new barrier, table-update</td>
<td>Stats &amp; inventory-manager now internal to OFPlugin. Hence no reason for them to communicate via MD-SAL. <strong>Advantages</strong> stats not flooding MD-SAL, a bit faster and reliable, better control over statistics polling. <strong>Consequences</strong> applications outside OFPlugin can not query stats directly from device. They need to listen Operational Data Store changes.</td>
</tr>
<tr>
<td><strong>RPC completion (flow/meter/group management)</strong></td>
<td>upon message sent to device</td>
<td>upon change confirmed by device</td>
<td><strong>Advantages</strong> Provides more information in RPC result <strong>Consequences</strong> RPC processing takes more time</td>
</tr>
<tr>
<td><strong>Exposing device</strong></td>
<td>right after handshake</td>
<td>after device explored</td>
<td><strong>Advantage</strong> when new device in DS/operational all informations are consistent and all RPCs ready. <strong>Consequence</strong> by devices with large stats reply it might take longer time till they get exposed in DS/operational.</td>
</tr>
</tbody>
</table>

More details at: [5]
Addition of new features

Table features

• Update to the inventory based on Table Features response. Tested manually only against the CPqD switch

  • OpenFlow Spec 1.3 (A.3.5.5 Table Features)

Role Request Message

• Implementation of Role Request Messages for Multi-controller operation (done on existing implementation only, not done on alternate design)

  • OpenFlow Spec 1.3 (A.3.9 Role Request Message)
Integration / CI testing improvements

• Varying levels of contributions from at least 6 individuals
• More than 300 new test cases introduced
• Scale Monitoring Suites:
  • switch discovery
  • link discovery
  • host discovery (depends on L2-Switch project)
  • flow programming
• Performance Monitoring Suites:
  • Northbound flow programming
  • Southbound packet-in response
• Job replication for both code bases
• A Openflow longevity suite close to being in CI
• Bug regression cases
Integration / CI testing improvements

- Varying levels of contributions from at least 6 individuals
- More than 300 new test cases introduced
- Scale Monitoring Suites:
  - switch discovery
  - link discovery
  - host discovery (depends on L2-Switch project)
  - flow programming
- Performance Monitoring Suites:
  - Northbound flow programming
  - Southbound packet-in response
- Job replication for both code bases
- A Openflow longevity suite close to being in CI
- Bug regression cases

Big Thanks to Peter Gubka
### 30 stable/lithium system test jobs

### over 2100 test cases gives good confidence of stability
Switch Scalability Monitoring
Two tests, same goal, different implementations and verifications

**GOAL:** iteratively increase the number of switches in the topology until the max (500) is achieved or record/plot the value where failure occurred

**FAILURE TRIGGERS:**
- OutOfMemory Exception in log file
- Switch count wrong in operational store
- topology links presence

**FAILURE TRIGGER:**
- Switches discovered in operational within 35s

starts and stops X switches where X starts at 100 and increases by 100.

adds 10 switches at a time and never removes them.
Link Scalability Monitoring

**GOAL:** iteratively increase the number of switches (up to 200) using a full mesh topology. The maximum links tested would be $200 \times (200 - 1) = 39800$ (NOTE: 1 connection would be 2 unidirectional “links”)

**FAILURE TRIGGERS:**
- OutOfMemory Exception
- NullPointer Exception
- Switch count wrong in operational store
- Link count wrong in operational store

![Graph showing max number of switch links](image)

bugzilla/3706
**Host Discovery Monitoring**

**GOAL:** iteratively increase the number of hosts (up to 2000) connected to a single switch, starting from 100 and increasing by 100.

**FAILURE TRIGGERS:**
- OutOfMemory Exception
- Host count wrong in operational
- Switch count (1) wrong in operational

![Graph showing max number of hosts](image_url)

- bugzilla/3326
- bugzilla/???
Northbound Flow Programming Performance Monitoring

- Configures 100k flows
  - 63 switches in linear topology
  - 25 flows per request
  - Rate seen is approx. 1600 flows/sec

- Configures 10k flows
  - 25 switches in linear topology
  - 1 flow per request
  - 2k flows handled by each of 5 parallel threads
  - Rate seen in default plugin is approx. 160 flows/sec
  - Rate seen in alternate plugin is approx. 200 flows/sec (was > 400 flows/sec)
Southbound Packet-In Response Monitoring
(using cbench tool)

**GOAL:** to monitor and recognize when significant changes occur.

**throughput mode average**
~ 100k flow_mods/sec

**latency mode average**
~ 16k flow_mods/sec

Throughput Mode

Latency Mode

existing plugin
Southbound Packet-In Response Monitoring (using cbench tool)

**GOAL:** to monitor and recognize when significant changes occur.

- **throughput mode average**
  - ~ 110k flow_mods/sec

- **latency mode average**
  - ~ 16k flow_mods/sec

![Graphs showing throughput and latency modes](image)
Performance Monitoring In Action

After communication and hard work a final merge (gerrit patch 20810) triggered the test that saw performance come back to what we expect.
Automating Reported Issues

Initial Issue Reported (Nov, 2014) and Fixed (Dec, 2014):

- **Bug 2429** - Need to close the ODL Denial of Service interface

It was reported again 7 months later (Jun, 2015):

- **Bug 3794** - OFHandshake thread leak leads to OOM

OF Handshake threads were leaking when a raw TCP connection was open and closed to the openflow port (6633). Anything with malicious intent could disable the controller in short order if this issue returns.

---

TEST CASE: Bug 2429 (non-critical)

Full Name: openflowplugin-flow-service.txt:Bug Validation.2429 Bug_2429

Documentation: Using the "nc" tool, a number of connections to the 6633 will be opened and closed to simulate a failed larger number of failed connections will be used and a small margin of error will determine if the test is passed.

Start/End/Elapsed: 20150724 14:38:55.070 / 20150724 14:37:11.681 / 00:00:13.611

Status: PASS (non-critical)

KEYWORD Log Environment Details


KEYWORD $controller_pid = Util.GetProcessID Based On Regex On Remote System $controller\_pid.0-9+.PID was not discovered


KEYWORD Buildn.RepeatKeyword $number_of_connections_to_fail, StartProcess, nc, -w, 1, $controller\_pid. $openflow_port

KEYWORD Log Environment Details


KEYWORD Log Environment Details

KEYWORD Buildn.Log starting count $starting_thread_count$ending_thread_count $ending_thread_count

KEYWORD $acceptable_thread_count = Buildn.Evaluate $starting_thread_count + $number_of_connections_to_fail * $margin_of_error

KEYWORD Buildn.ShouldBe True $ending_thread_count.$acceptable_thread_count $ending_thread_count.$acceptable_thread_count exceeds acceptable count $acceptable_thread_count

---

11 lines of Robot code should prevent this from surprising us again.
Agenda

• Project Overview
• High level architecture
• OpenFlow plugin example usecase
• Lithium accomplishments
• Plan for Beryllium
• Potential areas for contribution
• References
• Q & A
Plan for Beryllium:

• Enable Clustering support
• New Features
  • Flow entry eviction
  • Flow vacancy events
• Integration testing and CI improvements
  • Longevity tests
  • Clustering tests
  • Performance/Stability Tests
• Sonar code coverage
• Documentation improvement
Enable Clustering Support:

Clustering will provide

- High Availability for the plugin
  - More than one instance running the plugin
- Scalability
  - Set of switch connects to set of controller
- Persistence
  - Clustering takes care of user config data
Enable Clustering Support (contd..):
Enable Clustering Support (contd..):
Flow entry eviction

• Extension for OpenFlow 1.3 & part of OpenFlow 1.4
• Mechanism enabling the switch to automatically eliminate entries of lower importance to make space for newer entries
• Configure flow entry eviction
  • New messages: set, get request, get reply
  • Per-table configuration, on/off boolean
• New field: Flow importance
• Encoded as experimenter instruction, per flow
  • Optional hint for eviction algorithm
• Eviction process
  • Entirely switch defined
  • Report flows with reason OFPRR_DELETE
  • Flags in table desc to describe eviction criteria
Vacancy Events

• Extension for OpenFlow 1.3 & part of OpenFlow 1.4
• In OpenFlow 1.3 – abrupt behavior once switch flow table gets full
  • New flow entries not inserted – error returned
  • Likely disruption of service
• Provides a mechanism enabling the controller to get an early warning based on a capacity threshold chosen by the controller
  • Allows controller to react in advance and avoid getting the table full
  • New table status event with reasons VACANCY_DOWN & VACANCY_UP
• Table-mod vacancy property to set vacancy thresholds
Agenda

- Project Overview
- High level architecture
- OpenFlow plugin example usecase
- Lithium accomplishments
- Plan for Beryllium
- Potential areas for contribution
- References
- Q & A
Potential areas for contribution:

- Fixing Open and new bugs
- Contribution to CI/Integration testing
- Documentation (User & Developer Guides)
- Clustering
- Full OpenFlow 1.4 support
- Stats collection optimizations
  - Stats collection only to verify successful programming of flows
  - Enable / disable stats collection on a per flow basis
- Extensions to support for conntrack (stateful firewall) feature [6] in the latest OVS
- Filter packet-ins based on protocol
  - Allow applications to subscribe to packet-ins based on packet types
  - User defined filters for packet-ins
Agenda

• Project Overview
• High level architecture
• OpenFlow plugin example usecase
• Lithium accomplishments
• Plan for Beryllium
• Potential area for contribution
• References
• Q & A
References

[1] OpenFlow Plugin Wiki Main Page
[2] Potential Beryllium Items
[3] End to End Flow Programming
[4] Alternate Design for Performance Improvement Implemented in Lithium
[5] Comparison between existing design and the alternate design implemented in Lithium
[6] OVS Connection Tracking
Q & A