Micro distribution and agents

ODL Virtual Developer & Testing Forum - Jun 2020

Tejas Nevrekar & Ravi Sankar, Lumina Networks

Agenda

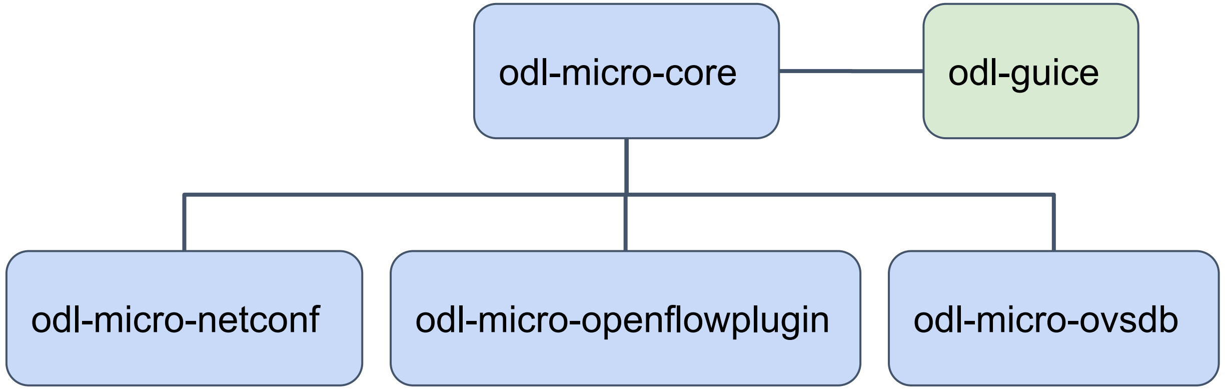
- Progress
- Modules
- Next Steps
- Goals
- Work
- Use Case Discussion

Progress

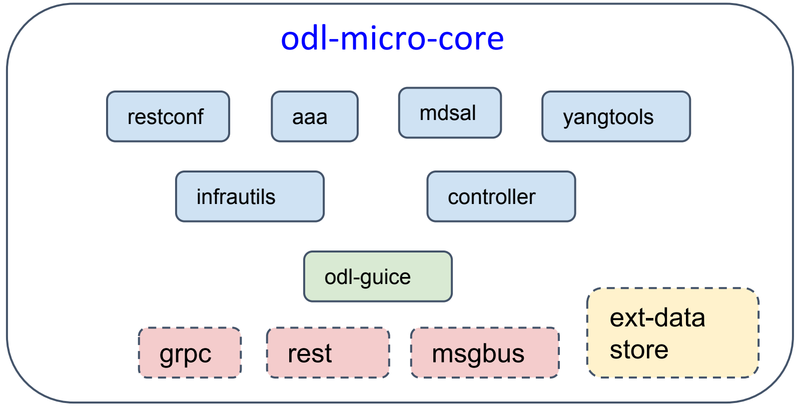
- Project
  - odl-guice - https://wiki.opendaylight.org/display/ODL/ODL+Guice+Project
  - odl-micro - https://wiki.opendaylight.org/display/ODL/ODL+Micro+Project

- Inflight:
  - odl-guice - change to add mycilla-guice to odl-guice, add guice projects from infrautils
  - infrautils - move guice sub-projects out of infrautils
  - odl-micro - add seed code from @voburger's github repo (https://github.com/vorburger/opendaylight-simple).
    - added support for netconf, openflowplugin and ovsdb micro packages
    - *Import of code pending on getting signed commits from michael, WIP*

Modules


odl-micro-core
odl-guice
odl-micro-netconf
odl-micro-openflowplugin
odl-micro-ovsdb

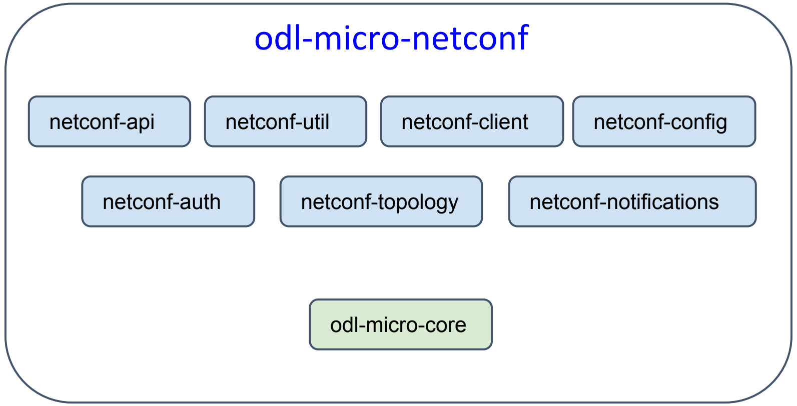
odl-micro-core



- Contains base modules needed by all the odl-micro distributions
- Builds on top of odl-guice and uses google guice as service dependency mechanism
- Uses existing odl components as-is
- Adds annotation based wiring of certain blueprint configuration dependencies

- Remove dependencies on mdsal-eos
- Add interfaces for grpc, rest and message bus for inter-service communications
- Add mechanism to let services use an external data store

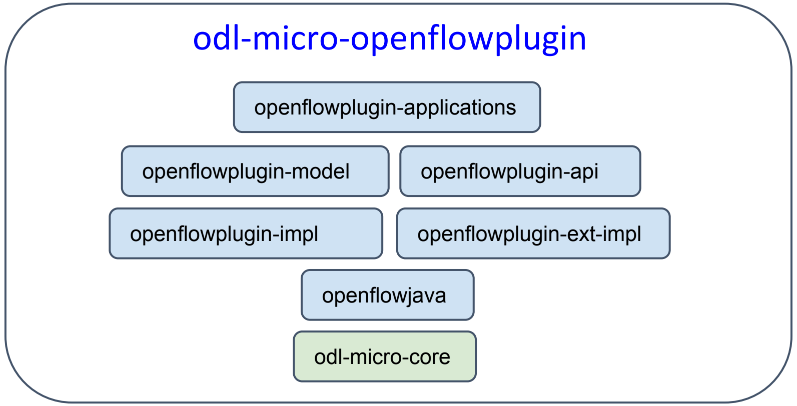
odl-micro-netconf



- Modules needed for netconf connector
- Depends on odl-micro-core exposed annotations for wiring dependencies
- Uses odl-micro-core

- Regression testing
- Pre-mounted devices via XML

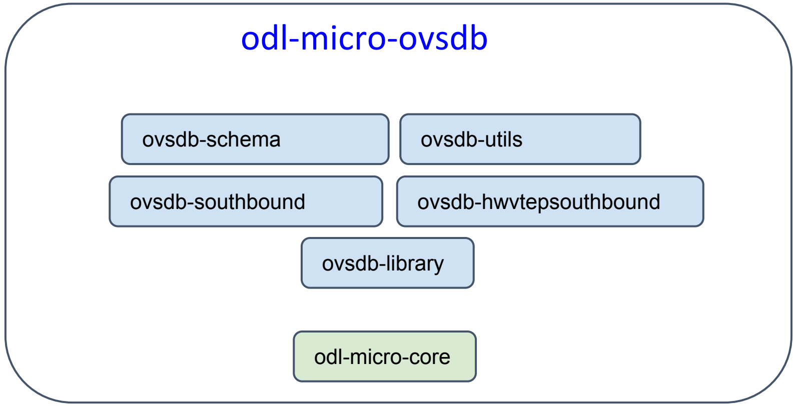
odl-micro-openflowplugin



- Modules needed for openflowplugin
- Includes support for reading the switch connector configuration
- Uses odl-micro-core

- Regression testing

odl-micro-ovsdb



- Modules needed for ovsdb
- OVSDB project has already moved to non-blueprint initialization, hence odl-micro ovsdb has very little code
- Uses odl-micro-core

- Regression testing

Next Steps

- Tools to generate micro images for ODL plugins
  - Allow a plugin like NETCONF, OPENFLOW, OVSDB to have a micro image bundling all required dependencies at compile time

- Tools to generate micro images for ODL applications
  - Allow an application deployed on ODL to generate its own micro image bundling all required dependencies at compile time

- Manage transactions with multiple plugin and application odl-micro services to co-operate

- Support for Spring

Goals

- Firstly build smaller micro-distributions that contain smaller sets of modules suitable for micro-service deployments:
    - odl-micro-openflowplugin
    - odl-micro-netconf
    - odl-micro-ovsdb

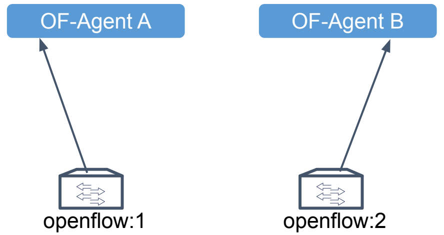
- Eventually expand the scope of odl-micro to cover “Managed” and stable “Self-Managed” projects, for example:
    - NETVIRT, BGPCEP, LISP, etc
    - JSON-RPC, TransportPCE

Work - Development, Validation

- Add code for generating the micro-distributions
  - This code can be in the project repos or centralized
  - We can use exiting System Test (CSIT) to validate the new micro-distribution, a weekly distribution test would be enough
- Perform Benchmarking tests to compare with existing Karaf/OSGI distribution
  - Startup time
  - CPU
  - Memory footprint
- Micro Backlog - https://jira.opendaylight.org/browse/ODLMICRO-1
- Guice Backlog - https://jira.opendaylight.org/browse/ODLGUICE-1

Use Case Discussion

A. Device Initiated connections



Agent requirements

- One agent manages only 1 device
- Simplifies assigning device to agent as it is 1:1
- No need for distributed leader election
- Need a agent that starts in seconds, memory <10M
- New device connection to auto-spawn a new agent - infra specific solution (e.g. k8s or custom)
- Handle device migration between agents
- Load balancer at southbound

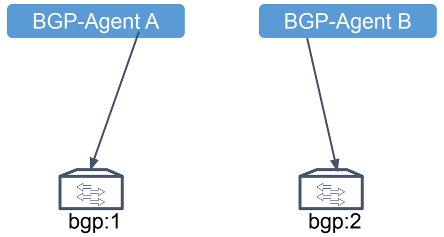
Impacting ODL services

- OpenFlowPlugin Agent
- Passive BGP Agent
- NETCONF call home Agent

Solution Component

- Custom
- k8s

B. Controller initiated connections

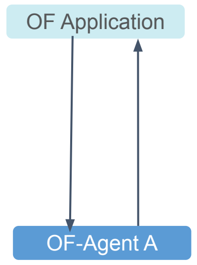


- Custom

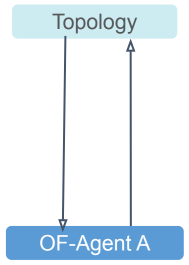
- One agent manages only 1 device
- Simplifies assigning device to agent as it is 1:1
- Need a agent that starts in milliseconds, memory <10M
- On moving a device from one agent to another, mount the device on the new agent.
- NETCONF, schema context gets duplicated across agents for the same device type
- Alternatively have multiple agents per device

- Active BGP Agent
- NETCONF Agent

C. Messaging



Request-Reply



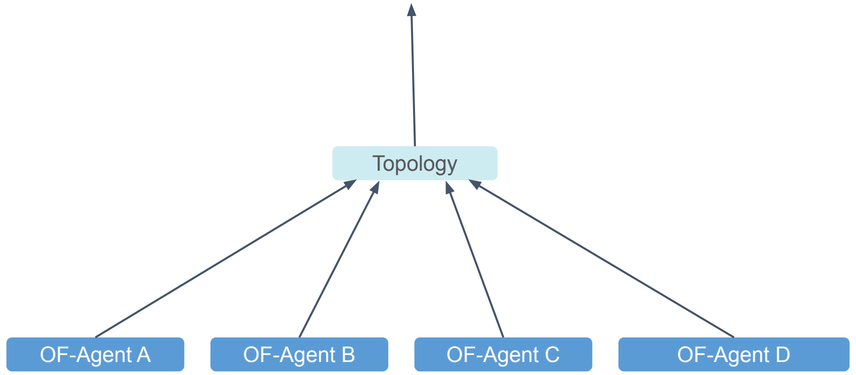
Publish-Subscribe

- Request-Reply - synchronous requests
- Publish-Subscribe - asynchronous requests

- Typical Topology Service
- Typical Inventory Service
- BGP RIB

- Custom
- grpc

D. Data Aggregation Service

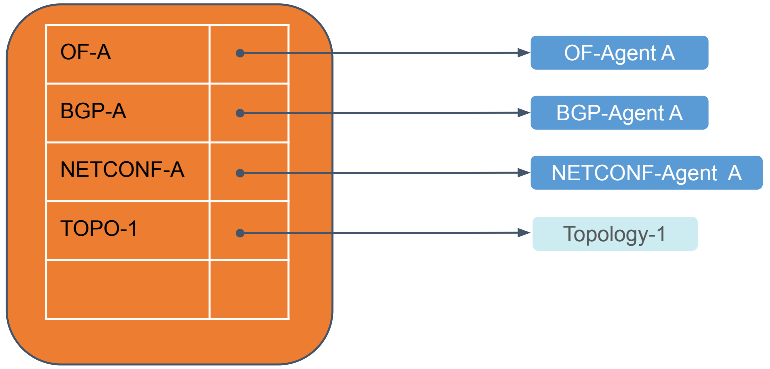


- Consumes updates from multiple downstream agents
- Pub-Sub registration/listener pattern
- Aggregator registers with agents
- Agents publish data
- Same pattern maybe visible in business logic
- Notify changes to upstream components

- Typical Topology Service
- Typical Inventory Service

- Custom

E. Service Locator

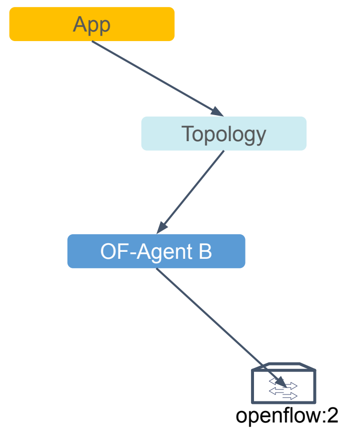


- Resolve service to container address

- Framework component

- Istio
- k8s

F. Multi-Service Control Flow

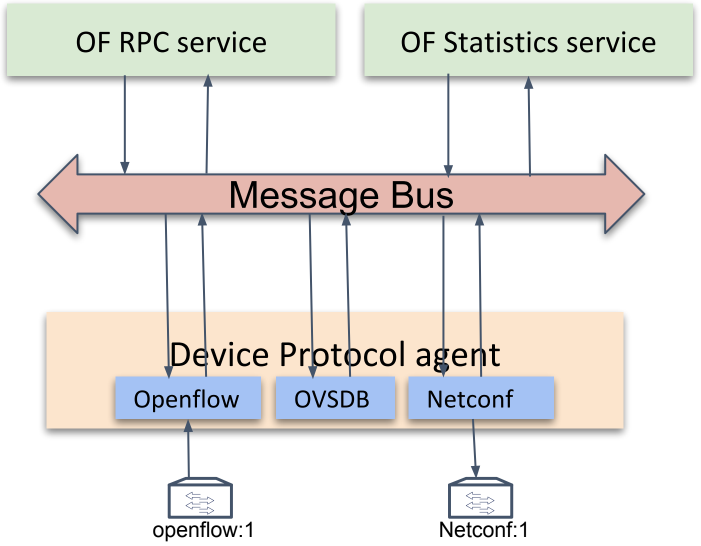


- App requests Topology which requests agent which sends to device
- Need end-to-end visibility
- Need retries at each leg coupled with timeouts
- Need circuit breaking at each leg

- Typical Topology Service
- Typical Inventory Service
- Higher level ODL Application

- Istio

Implementation thoughts



- Device protocol agent will handle all the device communication for different protocols that the device supports. This will not have any ODL MDSAL and yang tools code.
- Spawning a new agent when device connects will be very fast, as it will have minimal protocol realization code.
- An ODL application that wants to program a flow/group will be using the OF RPC service.
- When a OF Device connects, corresponding Device Protocol Agent will be started. This will create a device session and then sends a message.
- This will result in setting up of periodic statistics collection by the OF Statistics Service.

Thanks