

Genius Overview

Contents

- 1 [Contents](#)
- 2 [Background](#)
 - 2.1 [Application Co-existence and Integration Challenges](#)
 - 2.2 [Case-by-Case Approach](#)
 - 2.3 [Ingress De-multiplexing](#)
- 3 [Genius Proposal](#)
 - 3.1 [Generic Functions for Multi-Network Service Support](#)
 - 3.2 [VPN Service Modules and Inter-relationships](#)
 - 3.3 [Interface Manager](#)
 - 3.3.1 [Defining Granular interfaces \(ODL-interfaces data-model\)](#)
 - 3.3.2 [Binding Services on an Interface](#)
 - 3.3.3 [Service binding dataplane semantics](#)
 - 3.3.4 [Making SBI protocol agnostic \(South bound renderers\)](#)
 - 3.4 [Shared Overlay Tunnel Service](#)
 - 3.5 [Aliveness Monitor](#)
 - 3.6 [ID Manager](#)
 - 3.7 [MD-SAL UTIL](#)
 - 3.8

Background

Application Co-existence and Integration Challenges

- **Partitioning of OpenFlow Resources**
 1. Every application must have their private flow state space (on every switch)
 - a. Flow tables, group table, meter table, cookies
- **Ingress demultiplexing (aka “Table 0 Problem”)**
 1. Packets entering the switch have to be directed to the correct application pipeline
 2. Applications cannot simply write ingress flow entries into table 0 without coordination
 3. Need smaller granularity than OF port (e.g. VLAN, VNI, etc)? generalized interface concept
 4. Co-existence of multiple applications on the same interface
 5. Multi-tenancy: Several isolated service instances of the same application
- **Integration/Co-operation of different applications**
 1. Control plane: Service APIs between applications
 2. Data plane: Transfer packets between the pipelines of different applications
 - a. Take the use of various packet metadata into account!
 3. Each application comes with its own overlay solution

Case-by-Case Approach

- **Chosen approach in ODL Beryllium between SFC, GPB and Netvirt**
 1. Partitioning of OpenFlow Resources
 - a. Design time coordination between different applications
 2. Ingress demultiplexing (aka “Table 0 Problem”)
 - a. All application write into table 0, but the flow entries and priorities have been agreed at design time to avoid unwanted interference
 3. Integration/Co-operation of different applications
 - a. Control plane: MD-SAL service APIs between applications?
 - b. Data plane: Direct GOTO Table to transfer the packet from one application pipeline to another
- **Analysis**
 1. Can lead to an optimal solution for a group of specific applications
 2. Design time coordination needed for every detail
 3. Hard-coded dependencies between applications
 4. Does not scale to many applications

Ingress De-multiplexing

- Multiple applications writing into table 0 (directly or through an Ingress Manager function)
- Flow conflict detection mechanisms do not allow for any overlap between flows
- Overlap (or rather refinement) should be allowed using priorities to disambiguate:

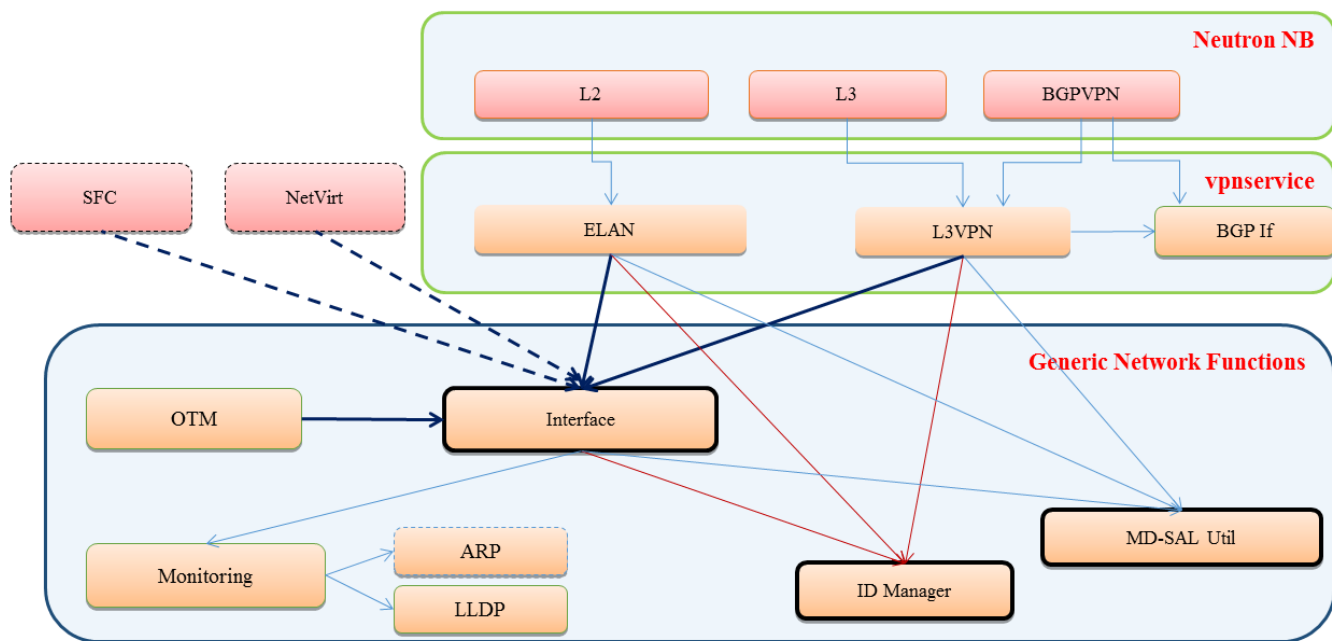
1. e.g. packets on in_port with certain DMAC to application A, all the rest to application B
- How can a generic Ingress Manager ensure that there is no semantic conflict between the flows if the simple non-overlap criterion is not sufficient?

Genius Proposal

Generic Functions for Multi-Network Service Support

- Any ODL application can use these to achieve at minimum interference-free co-existence with other applications using the services
- Provide support for co-operation between applications with the minimal amount of design-time coordination and hard-coded dependencies
- Use APIs to move design-time coordination to run-time
 1. Generic infrastructure APIs to avoid direct coupling where possible
 2. Direct inter-application (client-server) APIs where necessary for stronger coupling
- Factor out commonly used functions into shared services to avoid duplication & waste of resources, e.g.
 1. Overlay Tunnel Manager
 2. ID manager
 3. MD-SAL Util

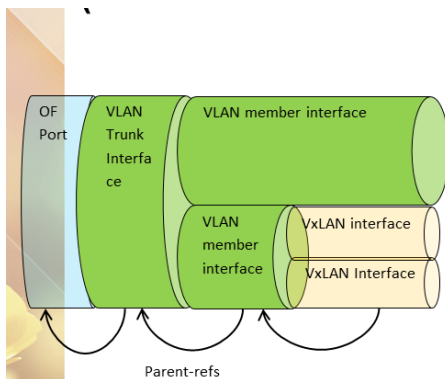
VPN Service Modules and Inter-relationships



Interface Manager

- **Models generic interfaces as attachment point for applications**
 1. Supports Hierarchy of: Port, VLAN, VXLAN Trunk, VXLAN VNI, GRE Tunnel, ...
 2. Extendible to arbitrary other types of interfaces (virtual link, VPN interface, ...)
 3. Interface ID/tag system wide unique identifier in control/data plane.
 4. Ingress interface tag stored in metadata
- **Handles ingress de-capsulation and de-multiplexing**
 1. Owns table 0 (and possibly additional tables needed for demultiplexing of interfaces)
 2. Application bind to interfaces through API and register application-specific instructions/actions to be added to the interfaces ingress flow entry (e.g. write metadata, goto table)
 3. Each bound service is assigned to a separate interface handle, no risk of interference on ingress traffic
- **South bound protocol agnostic**
 1. Ability to plug in different south bound renderers
 2. Provides tunnel monitoring services
 3. Handles egress encapsulation and output, service processing priority

Defining Granular interfaces (ODL-interfaces data-model)

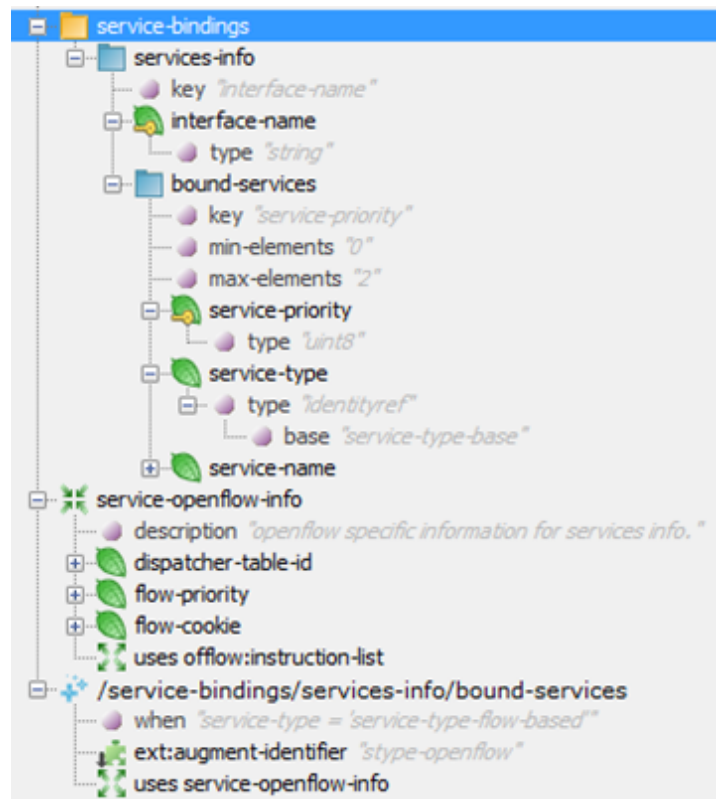


Black – From ietf-interfaces data-model
 Blue – Augmentations added from odl-interfaces
 Bold – mandatory

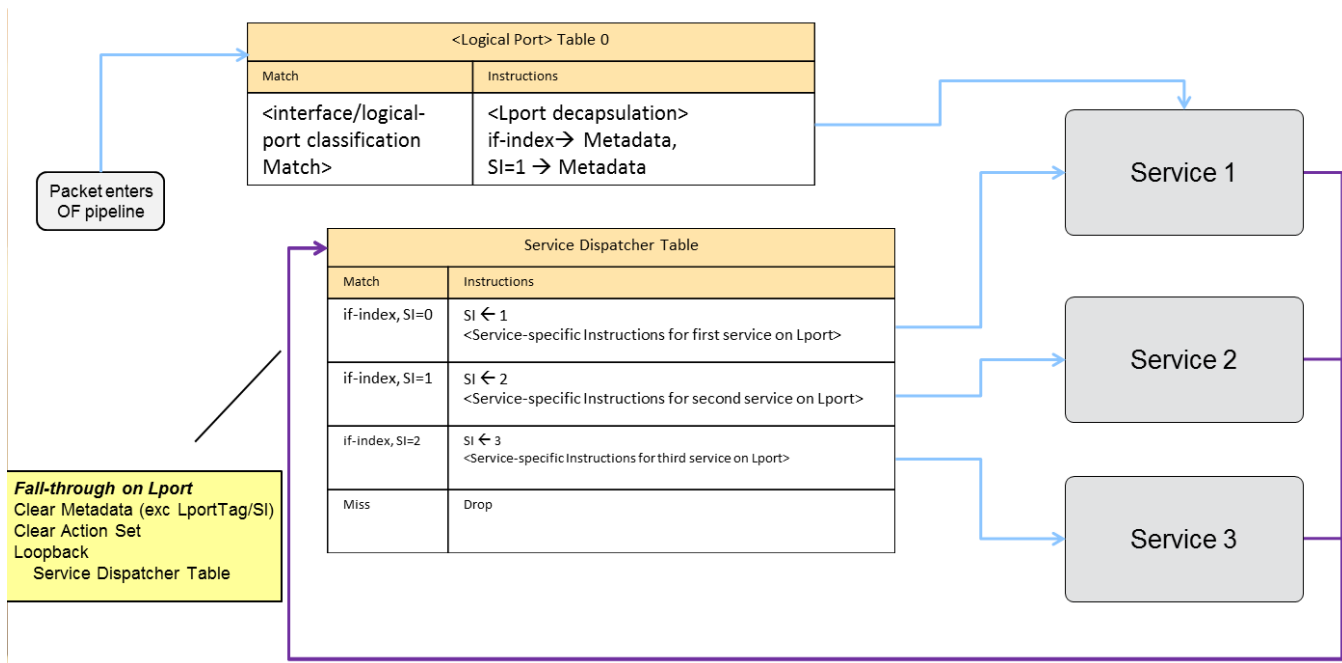
Base params	Vlan-trunk	Vlan-trunk-member	vxlan	gre
Configuration Data				
Name	<unique-name/uuid>	<unique-name/uuid>	<unique-name/uuid>	<unique-name/uuid>
description				
Type	l2vlan	l2vlan	tunnel	tunnel
enabled	enabled	enabled	enabled	enabled
Parent-refs	<Datapath-node-id>	<64 bit dpnid>	<64 bit dpnid>	<64 bit dpnid>
	<port_name on SBI>	<Parent vlan-trunkIf>	<Parent l2vlanIf>	<Parent l2vlanIf>
	l2vlan-mode = trunk	l2vlan-mode = trunk-member	tunnel-type = vxlan	tunnel-type = gre
	Vlan-id (N.A.)	Vlan-id = trunk-vlanId	dpn-id	dpn-id
			Vlan-id	Vlan-id
			source-ip	source-ip
			destination-ip	destination-ip
			gateway-ip	gateway-ip
			Monitor-enabled	Monitor-enabled
			Monitor-interval	Monitor-interval
Operational Data				
name	name	name	name	name
type	type	type	type	type
admin-status	admin-status	admin-status	admin-status	admin-status
oper-status	oper-status	oper-status	oper-status	oper-status
List higher-layer-if	List higher-layer-if	List higher-layer-if	List higher-layer-if	List higher-layer-if
List lower-layer-if	List lowerlayer-if	List lowerlayer-if	List lowerlayer-if	List lowerlayer-if
Phys-address	Phys-address	Phys-address	Phys-address	Phys-address
Ifindex	Ifindex	Ifindex	Ifindex	Ifindex
Node-connector-id	Node-connector-id	Node-connector-id	Node-connector-id	Node-connector-id
Statistics	Statistics	Statistics	Statistics	Statistics

Binding Services on an Interface

- Service binding data model used to bind a service on a particular interface
- Service module configures following parameters
 - Service-Priority
 - Service-Name
 - Service-Type
 - Service-Info
 - (for service-type openflow-based)
 - Flow-priority
 - Instruction-list
- Interface Manager maintains a Service dispatcher table to regulate pipeline dynamically between services
- Listens to service-binding changes and accordingly programs the dataplane (Table 0 & Service Dispatcher)

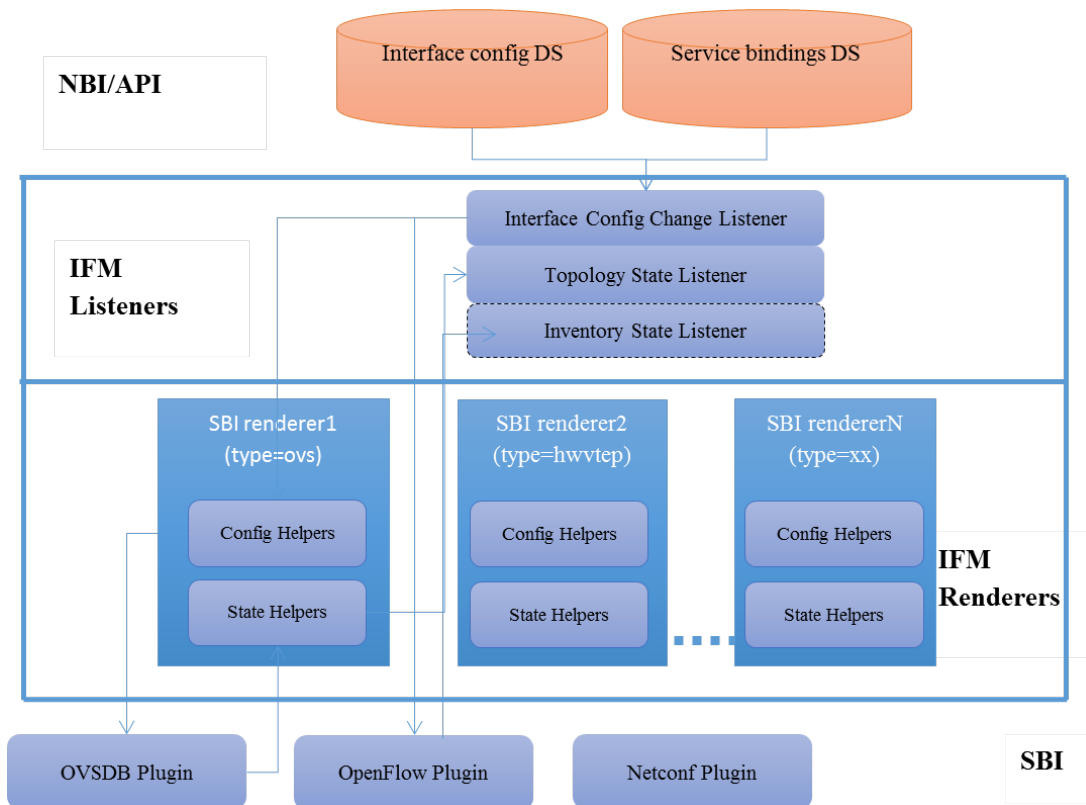


Service binding dataplane semantics



Making SBI protocol agnostic (South bound renderers)

- Provide a flexible way to support multiple south bound protocols
- North-bound interface/data-model is decoupled from south bound plugins
- NBI Data change listeners select and interact with appropriate SBI renderers
- New renderers can be added to support new Southbound interfaces/protocols/plugins
- Needs to be re-conciled with Netvirt re-design proposal



Shared Overlay Tunnel Service

- Provides tunnel creation/management services
 1. Can be configured to automatically create a homogenous bi-directional tunnel mesh (VxLAN/GRE/others) between a given group of DPNs
 2. API to add new nodes into an existing tunnel mesh
 3. API to create uni-directional tunnels from a DPN to an external IP node (CE router) which may not be under SDN control
 4. Support for tunnel level redundancy by creating a logical-group-interface, combining more than one tunnel interfaces, and allow for load-balancing or failovers in the group
- API support to control monitoring of tunnel interfaces
- API to get egress-actions and ingress-rules/bindService for specified uni-directional tunnel
- NB Tunnel Up/Down events for services/ user-facing / analytics apps

Aliveness Monitor

- Provides Controller driven monitoring services for
 1. Point-to-point interfaces (VxLAN/GRE)
 2. From an interface to destination IP Node
- Consumes services from ARP, LLDP, Ping Modules
- Generate Aliveness Events
- Interface manager listens to Aliveness events and updates operational-state of interfaces
- Consumers register for monitoring services specifying monitored interfaces and monitoring parameters
- Uses –
 1. Physical topology monitoring
 2. monitoring of non-BFD transport tunnels
 3. Service Function Monitoring

ID Manager

- Generates and provides unique integer IDs from a pre-configured ID-Pool with configured range to requesting services
 1. APIs for C/D of ID-Pool, assignment and lookups of IDs to services
- Dual modes of operation
 1. Consistent ID generation – consistently provide the same ID for a particular unique key String, Id-value is retained across cluster restarts, and associated with unique key (implemented)
 2. Generic ID assignment, no guarantees of consistency or persistence (not implemented)
- Can be used to assign IDs for and manage resources such as Openflow Tables, Groups, Meters, Service IDs
- Used by Interface Manager for mapping service instances to logical tags in the data plane

MD-SAL UTIL

- Provides Java interfaces to interact with MD-SAL DS and southbound OF-plugin.
 1. APIs for programming Flows & Groups
 2. APIs for Data-Store Read/write
 3. Generic CDN listener
- Others?