Service Function Chaining - Helium, Lithium and the way forward...

Shuva Jyoti Kar
Senior Software Engineer, Ericsson & OpenDaylight SFC Contributor

https://wiki.opendaylight.org/view/Service_Function_Chaining:Main
Agenda

• Service Function Chaining Problem Description
• Service Function Chaining Architecture in OpenDaylight (SFC ODL)
• SFC ODL Use cases in Helium Release
• ODL SFC Lithium & Beyond
Service Function Chaining provides the ability to define an ordered list of network services, or service functions (like firewalls, load balancers, DPI, TIC) for a set of packets.
Service Function Chaining (SFC)

• Services across the network “stitched” together to create a service chain.
SFC- Problem Statement

• Current Service Function Deployment models have few problems:
  • Topological dependencies
  • Complexity in configuration
  • Packet Classification
  • Agile/elastic service delivery
  • Enforcement of consistent ordering of service functions
How does SFC address these
Service Overlay through SFC

- SFC provides a framework to address these problems through
  - Creation of service specific overlays between service nodes
  - Creation of service overlay that is independent of network topology
SFC implementation in ODL

• Service Chaining is a classical application in an SDN domain
  • Provisioning of service chain, configuring and reconfiguring them is easy
  • Controller has the consistent view of the entire network
SFC ODL Components
Data Model (Yang) for SFC (in Helium)

- SFC data model is defined in Yang files
- Yang files are fed into the MD-SAL at compile time
- RESTCONF APIs and southbound hooks created from Yang

Diagram courtesy: OpenSDNIndia-ONF-ODL-SFC by Vinayak Joshi
SFC in Helium

• Data model definitions
• Terminologies – SF, SFF, SFP, SFC, SFC proxy, data-plane locators
• Receive the configuration of service paths/chains, switches, service-functions through northbound REST apis
• Network Service Header and non Network Service Header approaches
• Integration with OVS and LISP
• Contributors from Cisco, Ericsson, Red Hat, Contextream, Brocade, IBM, Citrix, etc.
Terminologies...terminologies.... and more
SFC Terminologies

• **Service Function Forwarder (SFF):** Switch/dpn

• **Service Function (SF):** any application such as DPI/FW/LB/TIC

• **Service Function Chain (SFC):** the intended list of SFs that the packets have to traverse in a definite order

• **Service Function Path (SFP):** actual instance of the services that are traversed, or a specific instance of the sfc
SFC Terminologies (contd.)

• **Service Classifier**: Function that helps in packet classification

• **Metadata**: Information that is carried across nodes

• **Network Service Header**: SFC encapsulation used by SFC-aware nodes, in case of SFC-unaware nodes, SFC-proxy has to be used

• **Nodes could be either SFs or SFFs**
A traditional SFC

Diagram courtesy: OpenSDNIndia-ONF-ODL-SFC by Vinayak Joshi
The NSH-aware way..

• NSH provides
  • Service Path Information,
  • Progress/location within Service Path,
  • opaque application metadata
• “Classify once” model
• Expandable header that is inserted after initial classification at service plane entry
• Lifetime confined within SFC domain

<table>
<thead>
<tr>
<th>Mandatory: Base Header (flags, next protocol) -4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory: Service Path Header (service plane forwarding info i.e. SFP ID, service index) – 4 bytes</td>
</tr>
<tr>
<td>Mandatory: Context Headers (four headers, 4 bytes each)</td>
</tr>
<tr>
<td>Optional: Variable length Opaque context headers</td>
</tr>
</tbody>
</table>
SFC- NSH with OVS

Diagram courtesy: OpenSDNIndia-ONF-ODL-SFC by Vinayak Joshi
The non-NSH way…

• “Re-classify at re-entry” model
• metadata to carry the service-path information
• No headers, no proxies, off the shelf use case
• But there are challenges…
  • Service path identification
  • Service hop identification
Ericsson’s Contribution to SFC in Helium

- No SFC encapsulation, i.e. no NSH
- OpenFlow 1.3.1 compliant
- There is no particular transport affinity, precludes VxLAN, GRE
- L2 connectivity of SFs and SFFs provided in Yang Files
- Packet re-classification at every entry to the SFP
- Current hop determined by MAC of previous hop
NO SFC Encap with L2 Reachability: Example Service Function Path

Diagram courtesy: OpenSDNIndia-ONF-ODL-SFC by Vinayak Joshi
SFC example use case in Helium
SFC use case example Deep-Dive
SFC use case example DeepDive

- Json for Service Function Path

```json
{
    "service-function-paths": {
        "service-function-path": [
            {
                "starting-index": 4,
                "service-chain-name": "request",
                "name": "request",
                "path-id": 23,
                "service-path-hop": [
                    {
                        "service_index": 1,
                        "hop-number": 1,
                        "service-function-name": "sf1",
                        "service-function-forwarder": "sff.1"
                    },
                    {
                        "service_index": 1,
                        "hop-number": 2,
                        "service-function-name": "pe",
                        "service-function-forwarder": "sff.1"
                    }
                ]
            },
            {
                "starting-index": 4,
                "service-chain-name": "response",
                "name": "response",
                "path-id": 39,
                "service-path-hop": [
                    {
                        "service_index": 1,
                        "hop-number": 1,
                        "service-function-name": "ggsn",
                        "service-function-forwarder": "sff.1"
                    }
                ]
            }
        ]
    }
}
```
SFC use case example DeepDive

Transport Configuration

```bash
sudo ofdatapath -i sff1_ggsn,sff1_sf1,sff1_pe -d 000000000001 ptcp:6681
sudo ofprotocol tcp:127.0.0.1:6681 tcp:<controller IP>:6653
```

Forward Path Transport

```bash
sudo dpctl tcp:127.0.0.1:6681 flow-mod cmd=add,table=10
eth_dst=00:00:08:01:02:01,vlan_vid=500 apply:output=2

sudo dpctl tcp:127.0.0.1:6681 flow-mod cmd=add,table=10
eth_dst=00:00:08:01:09:01,vlan_vid=100 apply:output=3
```

Reverse Path Transport

```bash
sudo dpctl tcp:127.0.0.1:6681 flow-mod cmd=add,table=10
eth_dst=00:00:08:01:01:01,vlan_vid=100 apply:output=1
```
SFC use case example DeepDive

Table 2: Packet classification

<table>
<thead>
<tr>
<th>Table 2: Packet classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>{table=&quot;2&quot;, match=&quot;oxm[in_port=&quot;1&quot;, eth_type=&quot;0x800&quot;]&quot;, dur_s=&quot;155&quot;, dur_ns=&quot;272000&quot;, prio=&quot;32768&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x0&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[meta[meta=&quot;0x17&quot;, mask=&quot;0xffffffffffffff&quot;]], goto{table=&quot;3&quot;}].}</td>
</tr>
<tr>
<td>{table=&quot;2&quot;, match=&quot;oxm[in_port=&quot;2&quot;, eth_type=&quot;0x800&quot;]&quot;, dur_s=&quot;155&quot;, dur_ns=&quot;218000&quot;, prio=&quot;32768&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x0&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[meta[meta=&quot;0x17&quot;, mask=&quot;0xffffffffffffff&quot;]], goto{table=&quot;3&quot;}].}</td>
</tr>
<tr>
<td>{table=&quot;2&quot;, match=&quot;oxm[in_port=&quot;3&quot;, eth_type=&quot;0x800&quot;]&quot;, dur_s=&quot;155&quot;, dur_ns=&quot;150000&quot;, prio=&quot;32768&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x0&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[meta[meta=&quot;0x27&quot;, mask=&quot;0xffffffffffffff&quot;]], goto{table=&quot;3&quot;}].}</td>
</tr>
</tbody>
</table>

Table 3: Hop identification

<table>
<thead>
<tr>
<th>Table 3: Hop identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>{table=&quot;3&quot;, match=&quot;oxm[metadata=&quot;0x17&quot;, metadata_mask=&quot;0xffffffff&quot;, eth_src=&quot;00:00:08:01:02:01&quot;, eth_type=&quot;0x800&quot;]&quot;, dur_s=&quot;36&quot;, dur_ns=&quot;228000&quot;, prio=&quot;256&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x14&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[apply[acts=[set_field(field:eth_dst=&quot;00:00:08:01:02:01&quot;), vlan_psh(eth=&quot;0x8100&quot;), set_field(field:vlan_vid=&quot;100&quot;)], goto{table=&quot;10&quot;}].}</td>
</tr>
<tr>
<td>{table=&quot;3&quot;, match=&quot;oxm[metadata=&quot;0x27&quot;, metadata_mask=&quot;0xffffffff&quot;, eth_src=&quot;00:00:08:01:02:01&quot;, eth_type=&quot;0x800&quot;]&quot;, dur_s=&quot;36&quot;, dur_ns=&quot;191000&quot;, prio=&quot;256&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x14&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[apply[acts=[set_field(field:eth_dst=&quot;00:00:08:01:02:01&quot;), vlan_psh(eth=&quot;0x8100&quot;), set_field(field:vlan_vid=&quot;100&quot;)], goto{table=&quot;10&quot;}].}</td>
</tr>
<tr>
<td>{table=&quot;3&quot;, match=&quot;oxm[all match]&quot;, dur_s=&quot;36&quot;, dur_ns=&quot;196000&quot;, prio=&quot;100&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x14&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[goto{table=&quot;4&quot;}].}</td>
</tr>
</tbody>
</table>

Table 4: Sending Packet to the First Hop

<table>
<thead>
<tr>
<th>Table 4: Sending Packet to the First Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>{table=&quot;4&quot;, match=&quot;oxm[metadata=&quot;0x17&quot;, metadata_mask=&quot;0xffffffff&quot;, eth_type=&quot;0x800&quot;]&quot;, dur_s=&quot;36&quot;, dur_ns=&quot;278000&quot;, prio=&quot;256&quot;, idle_to=&quot;0&quot;, hard_to=&quot;0&quot;, cookie=&quot;0x14&quot;, pkt_cnt=&quot;0&quot;, byte_cnt=&quot;0&quot;, insts=[apply[acts=[set_field(field:eth_dst=&quot;00:00:08:01:02:01&quot;), vlan_psh(eth=&quot;0x8100&quot;), set_field(field:vlan_vid=&quot;500&quot;)], goto{table=&quot;10&quot;}].}</td>
</tr>
</tbody>
</table>
SFC in Lithium and future releases

• Lithium:
  • Integration of Group Based Policy and Service Function Chaining
  • SFF Load Balance
  • Service Function Selection Algorithms

• Future:
  • Netconf integration
  • OpenStack integration
SFC: Lithium Overview

- SFF load-balance to change 1-1 mapping between SFs and SFFs
- Group of SFs -> SFG
- Mapping table being introduced
- Triggers mainly OF now, non OF in future
- Service chains are part of a contract
- Policy metadata passed along service chain
OPNFV gateway

• OpenFlow programmed service chains for:
  • L2 VLAN encapsulation
  • MPLS encapsulation

• VxLAN overlay based service chains for:
  • VxLAN-GPE encapsulation with NSH headers

• Basic load balancing at SFC (planned for ODL Lithium)

• Programmatic service function selection algorithms
  • Round robin
  • Load balanced (choose the least loaded service function)
  • Random allocation

https://wiki.opnfv.org/network_function_chaining
References

• Detailed ODL SFC Presentation (where this presentation is derived from):  https://wiki.opendaylight.org/images/8/89/Ericsson-Kumbhare_Joshi-
• OpenSDNIndia-ONF-ODL-SFC by Vinayak Joshi
• OpenDaylight_Service_Function_Chaining.pdf
• ODL SFC Wiki Page-
  https://wiki.opendaylight.org/view/Service_Function_Chaining:Main
• Service Function Chaining Architecture IETF Draft:
  https://datatracker.ietf.org/doc/draft-ietf-sfc-architecture/
• Yang Model for Service Function Chaining IETF Draft
  https://datatracker.ietf.org/doc/draft-penno-sfc-yang/?include_text=1
• SFC IETF Working Group
  https://datatracker.ietf.org/wg/sfc/charter/
Thank You