ERICSSON

OPENDAYLIGHT SERVICE FUNCTION CHAINING USE-CASES

14 October 2014

Contact: Abhijit Kumbhare & Vinayak Joshi

OPEN DAYLIGHT

SDN OpenFlow WORLD CONGRESS

AGENDA

- SFC Architecture
- Use Cases

OPEN DAYLIGHT

SDN OpenFlow WORLD CONGRESS

SERVICE FUNCTION CHAINING (SFC)

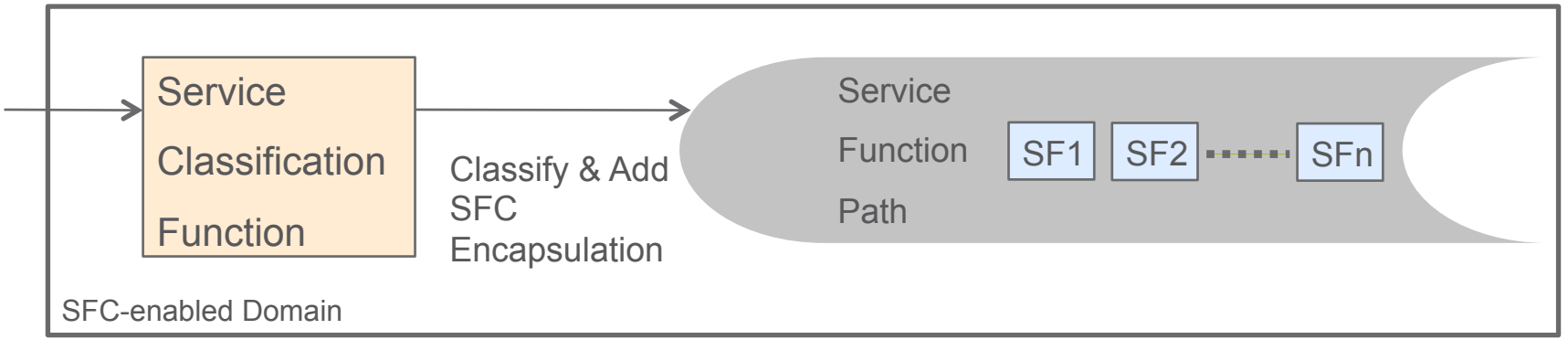
- Service Function Chaining provides the ability to define an ordered list of a network services (e.g. firewalls, load balancers, DPI, etc. – a.k.a. Service Functions).
- Services "stitched" together in the network to create a service chain.

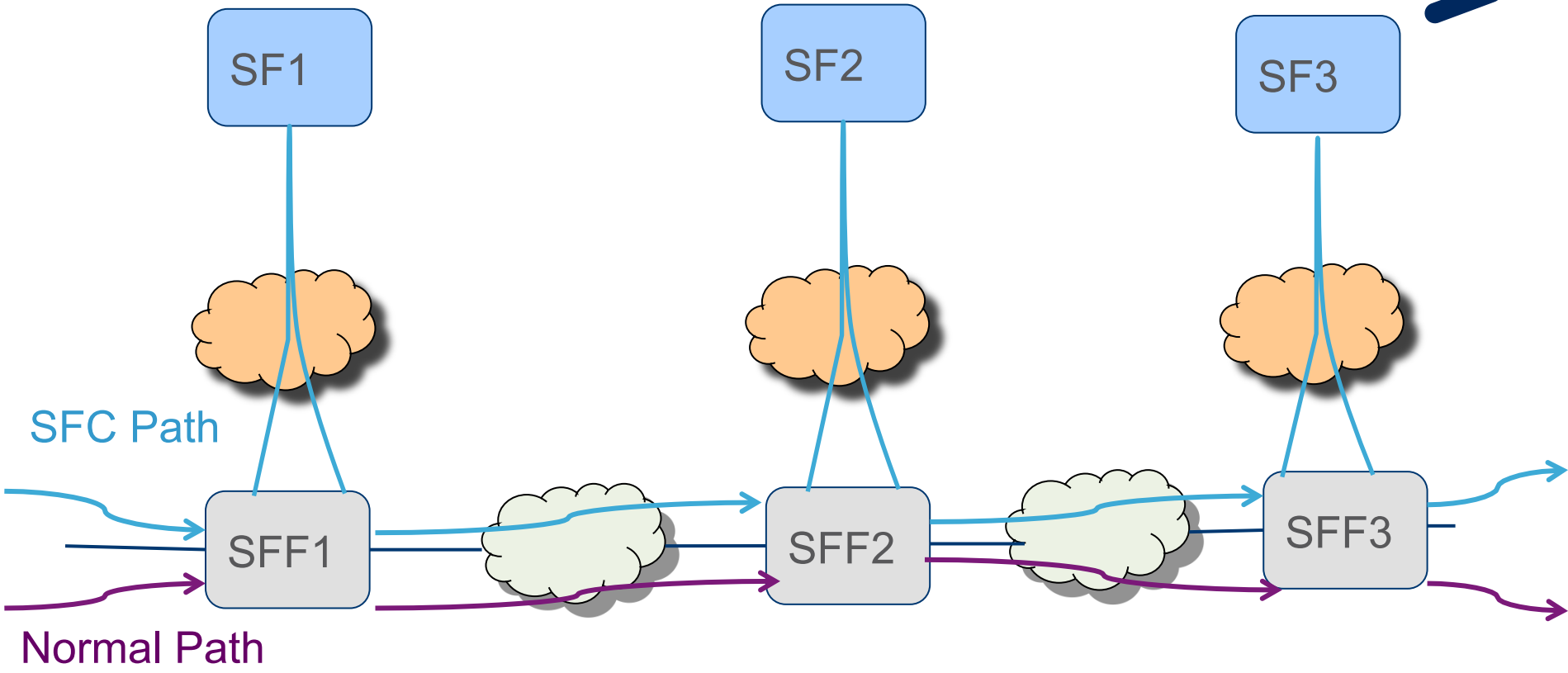
SFC PROBLEM STATEMENT

- Few problems in current Service Function Deployment Models:
  - Topological dependencies
  - Configuration complexity
  - Manual service chain construction
  - Consistent ordering of service functions enforcement
  - Elastic service delivery
- SFC provides a framework to address problems mentioned
  - Service Overlay
    - Service specific overlay creates a path between service nodes
    - Independent of network topology
      - Multiple transport (overlay/underlay)
      - Service functions located in the network as needed
      - Allows arbitrary order of service functions
      - Allows easy adding / removing of service functions

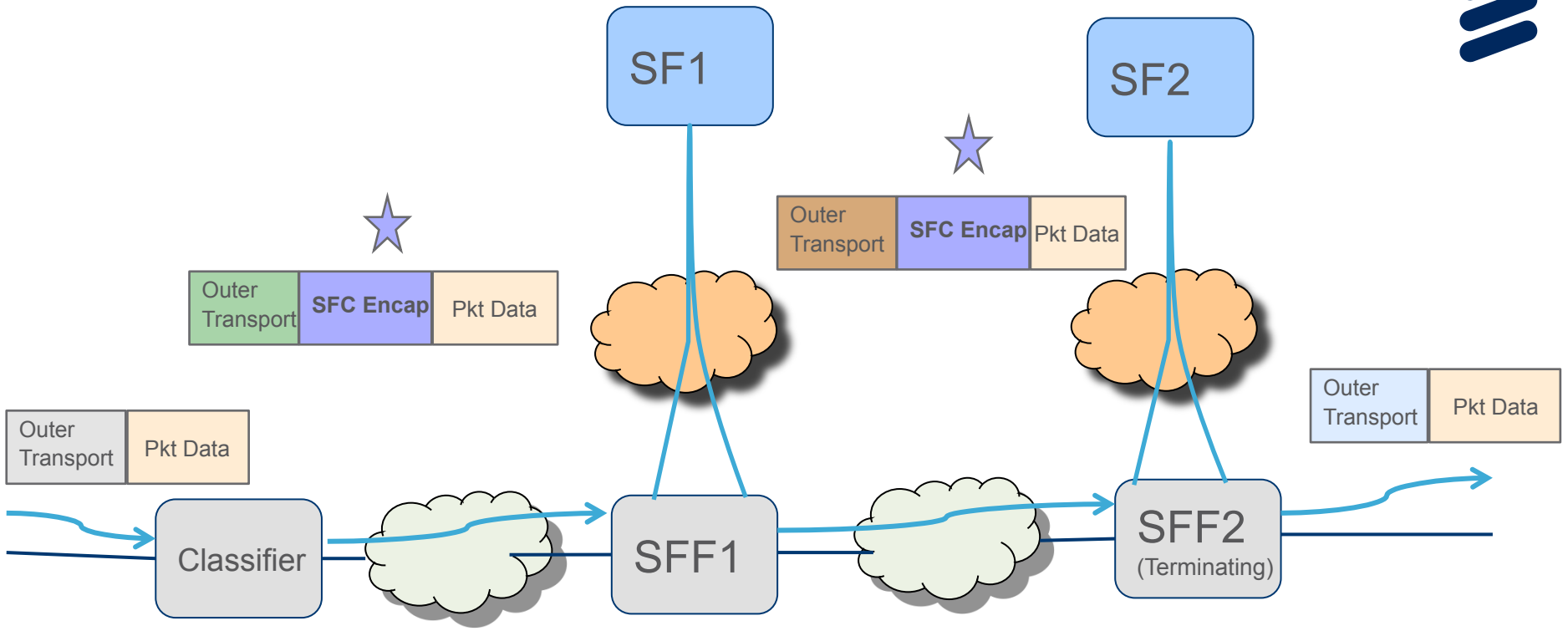
SFC HIGH LEVEL ARCHITECTURE

- Service Function Chain:
  - the intended set of abstract service functions that must be traversed by the classified packets and the order of traversal
  - e.g. DPI followed by parental control
- Service Function Path:
  - the actual instances of services traversed or specific instantiation of a service chain.
- Classifier:
  - determines what traffic needs to be chained based on policy.



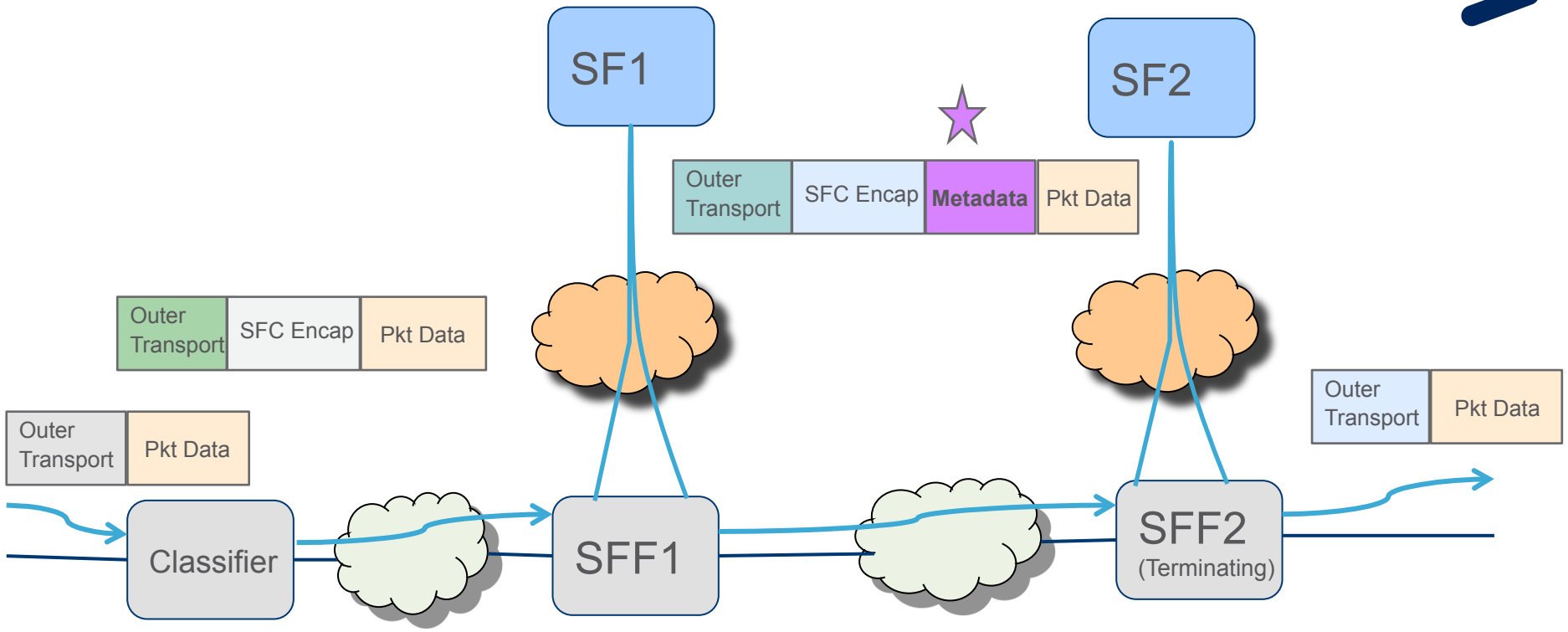


- Service Function Forwarder
  - responsible for delivering traffic to a Service Function using SFC encapsulation
  - e.g. an overlay switch like OVS



› SFC Encapsulation

- Provides SFC identification
- Used by SFC-aware functions (SFFs & SFC-aware SFs)
- Different from outer transport encapsulation
- Transit routers/switches forward based on outer encapsulation
- SFC encap transport independent – any network transport may be used



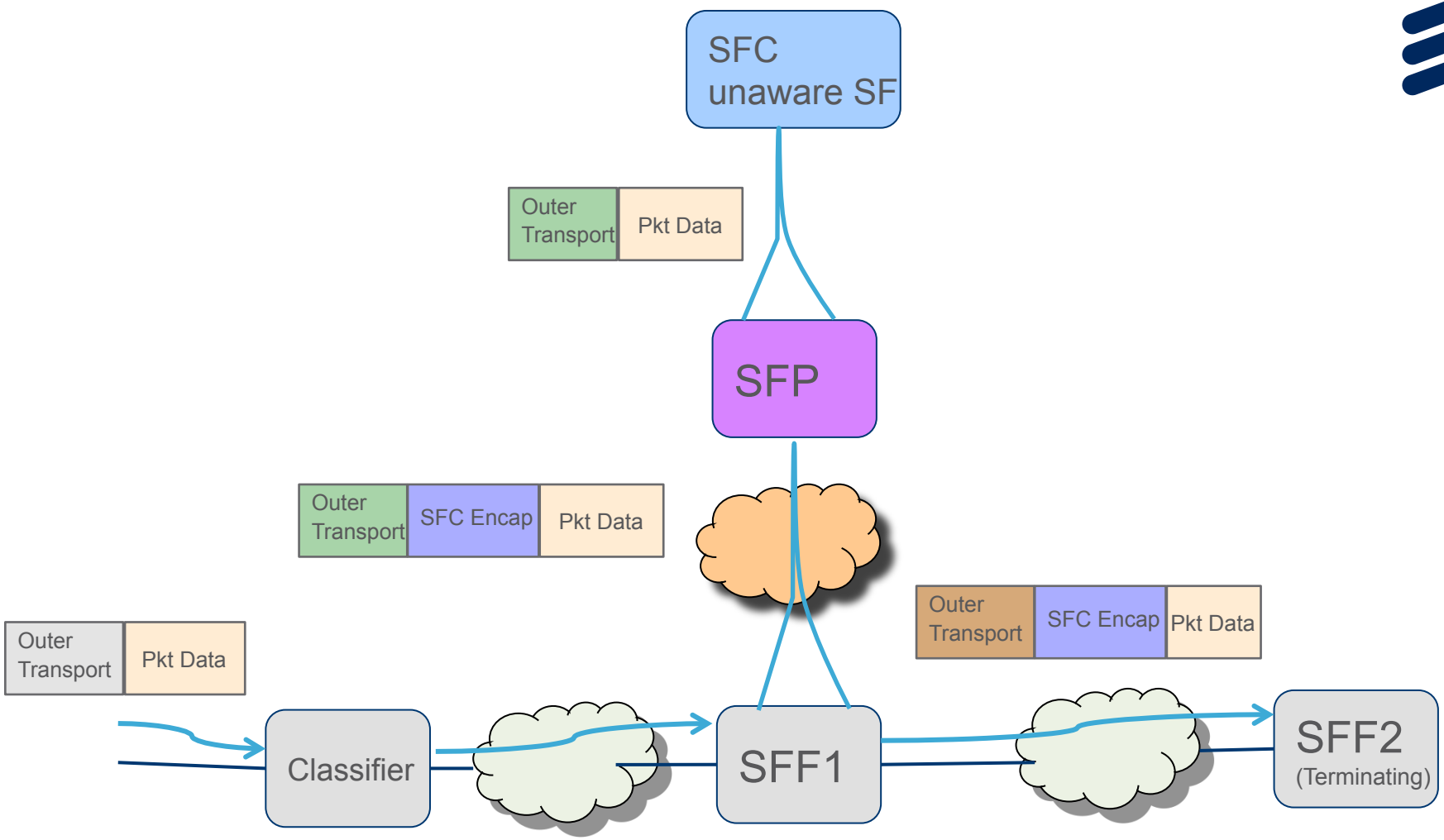
› Metadata ★

- Information passed between nodes
- Provides ability to exchange context information between classifiers and SFs as well as between SFs

SFC ENCAP/METADATA: NETWORK SERVICE HEADER (NSH)

Mandatory: Base Header (flags, next protocol) -4 bytes
Mandatory: Service Path Header (service plane forwarding info i.e. SFP ID, service index) – 4 bytes
Mandatory: Context Headers (four headers, 4 bytes each)
Optional: Variable length Opaque context headers

- Provides
  - Service Path Information (used for forwarding along SFP).
  - Location within Service Path.
  - Opaque application metadata
- Inserted after initial classification at service plane entry.
  - May be inserted after L2 Header (new EthType requested)
  - Expandable header
- Lifetime only within SFC domain.



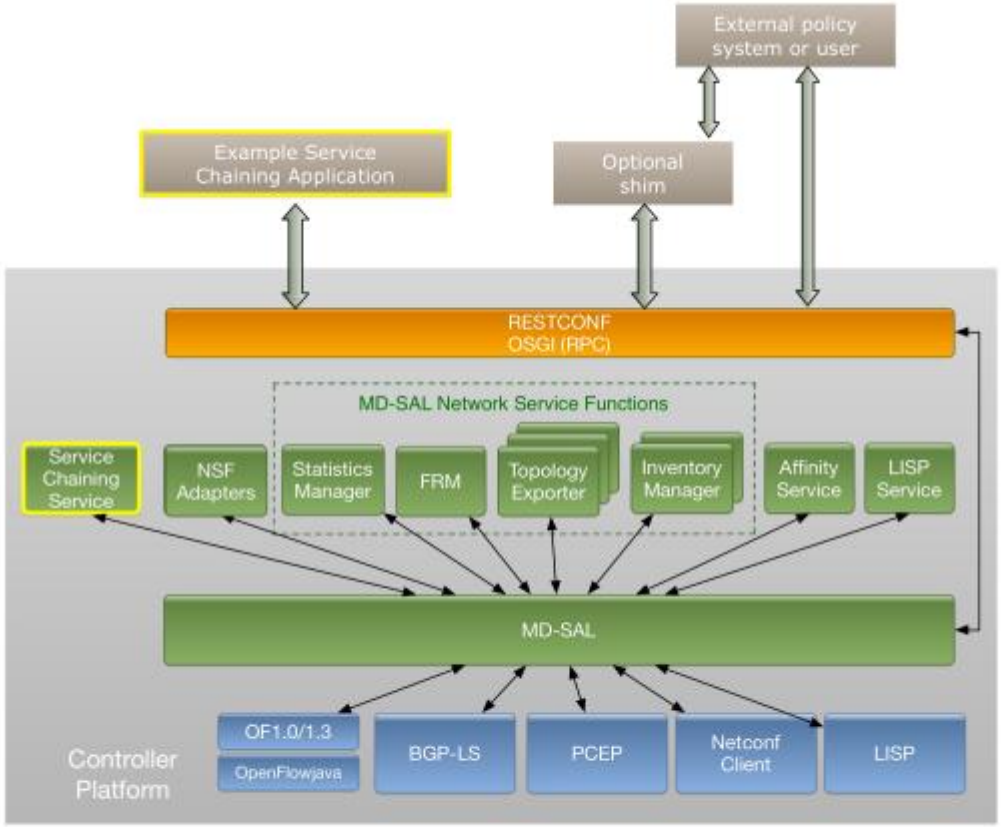
- SFC Proxy
  - Removes and inserts SFC encapsulation on behalf of an SFC-unaware service function

SFC IN ODL

- Rationale
  - SDN simplifies service chain provisioning & management
  - Entire service chains can be provisioned & easily reconfigured
    - Controller has overall view of the network – reduces chance of inconsistent device configurations

- Service Chaining in Helium
  - Receives information about chain and/or path to be constructed via northbound API
  - Constructs service paths
  - Contributors from Cisco, Ericsson, Red Hat, Contextream, Brocade, IBM, Citrix, etc.

SFC ODL COMPONENTS



SOME HELIUM SFC GAPS

- Possible future work items

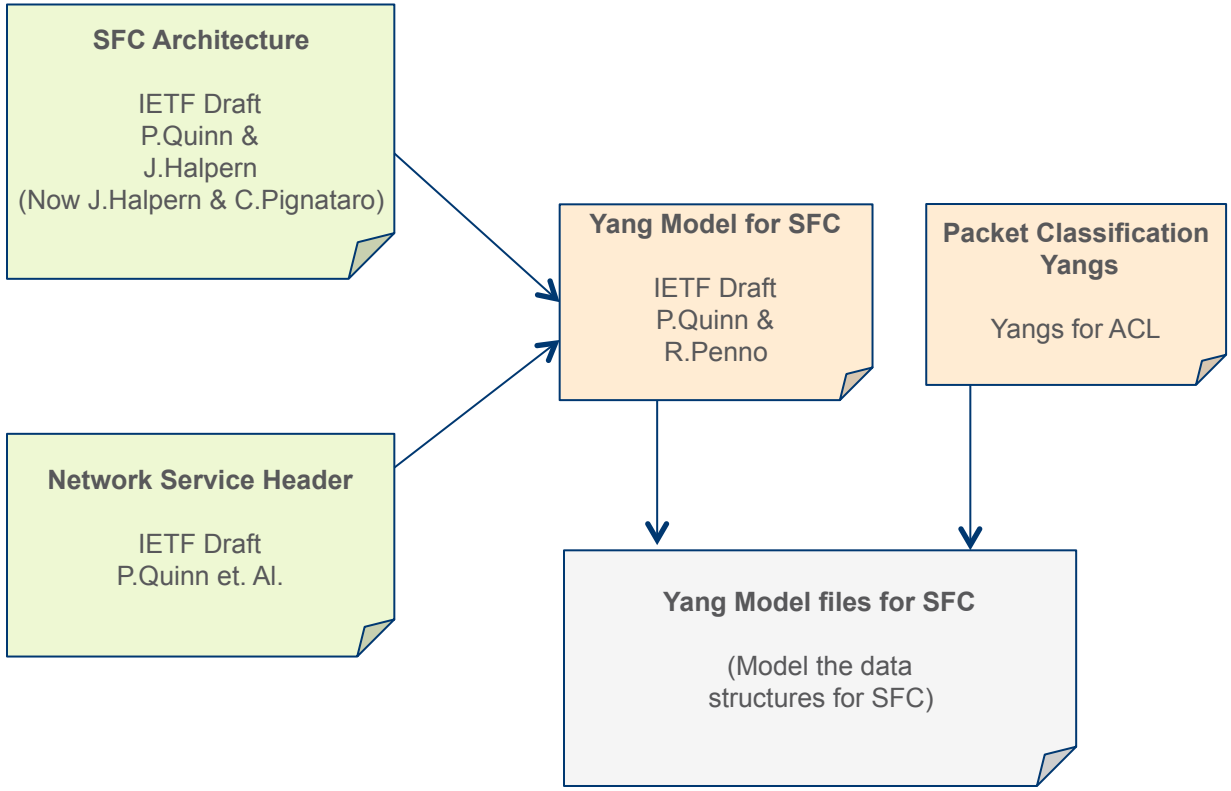
- Load Balancing & Fast Failover across SFs
  - Keep alives for monitoring of SFs (input to fast failover & load balancing).

- Enforcement of policy
  - In Helium – simple ACL based classification
  - Possible integration with Group based policy

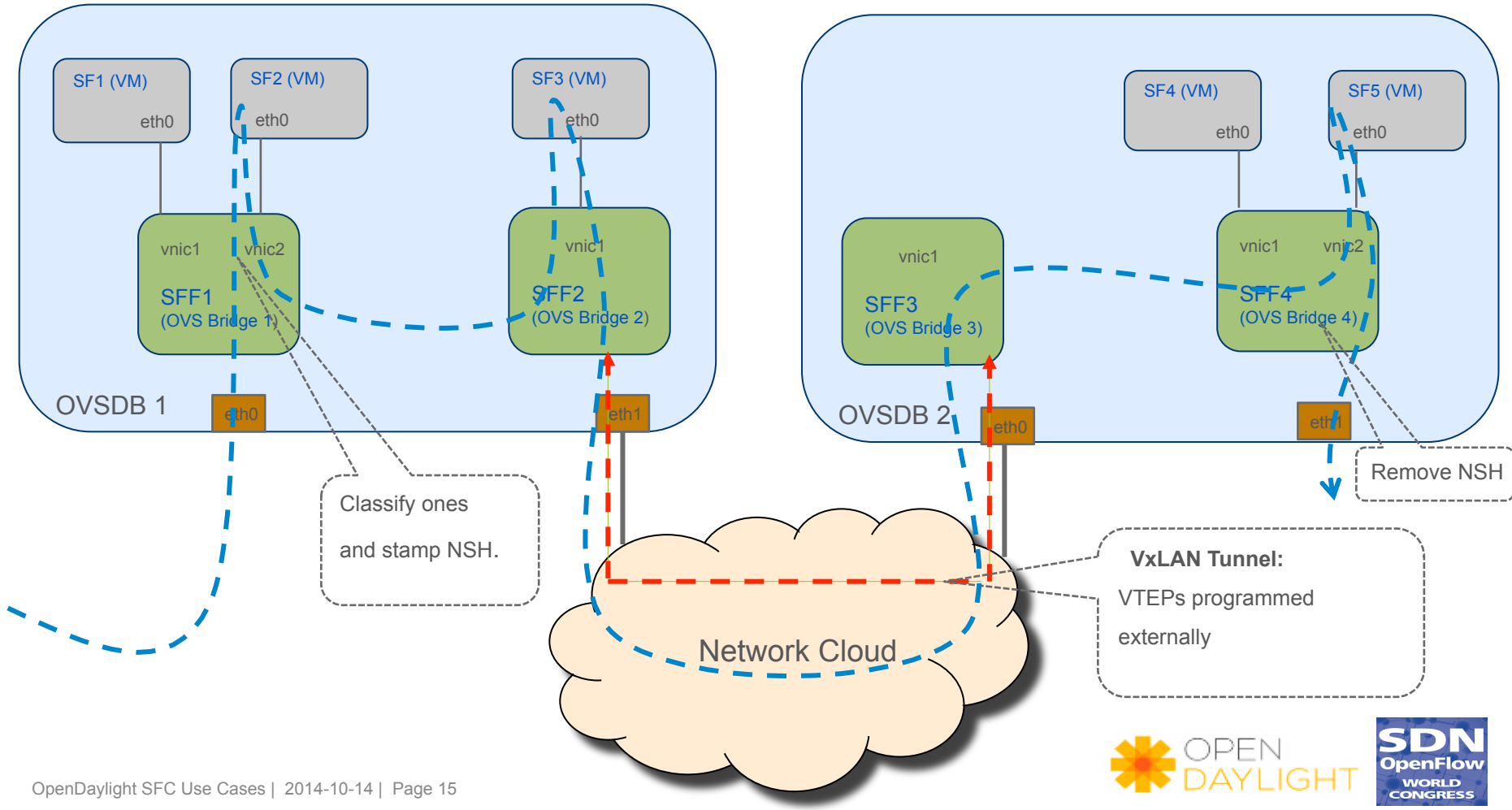
- Transport & Discovery to be programmed outside SFC
  - Outside SDN, separate OF flows etc.
  - Possible integration of other ODL discovery projects such as Source Tracker in future.

USE CASES: YANG MODEL FOR SFC

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



USE CASE 1: NSH WITH OVS (BASED ON SFC ODL MAILING LIST EXAMPLE)



USE CASE 2: OPENFLOW-NO SFC ENCAP

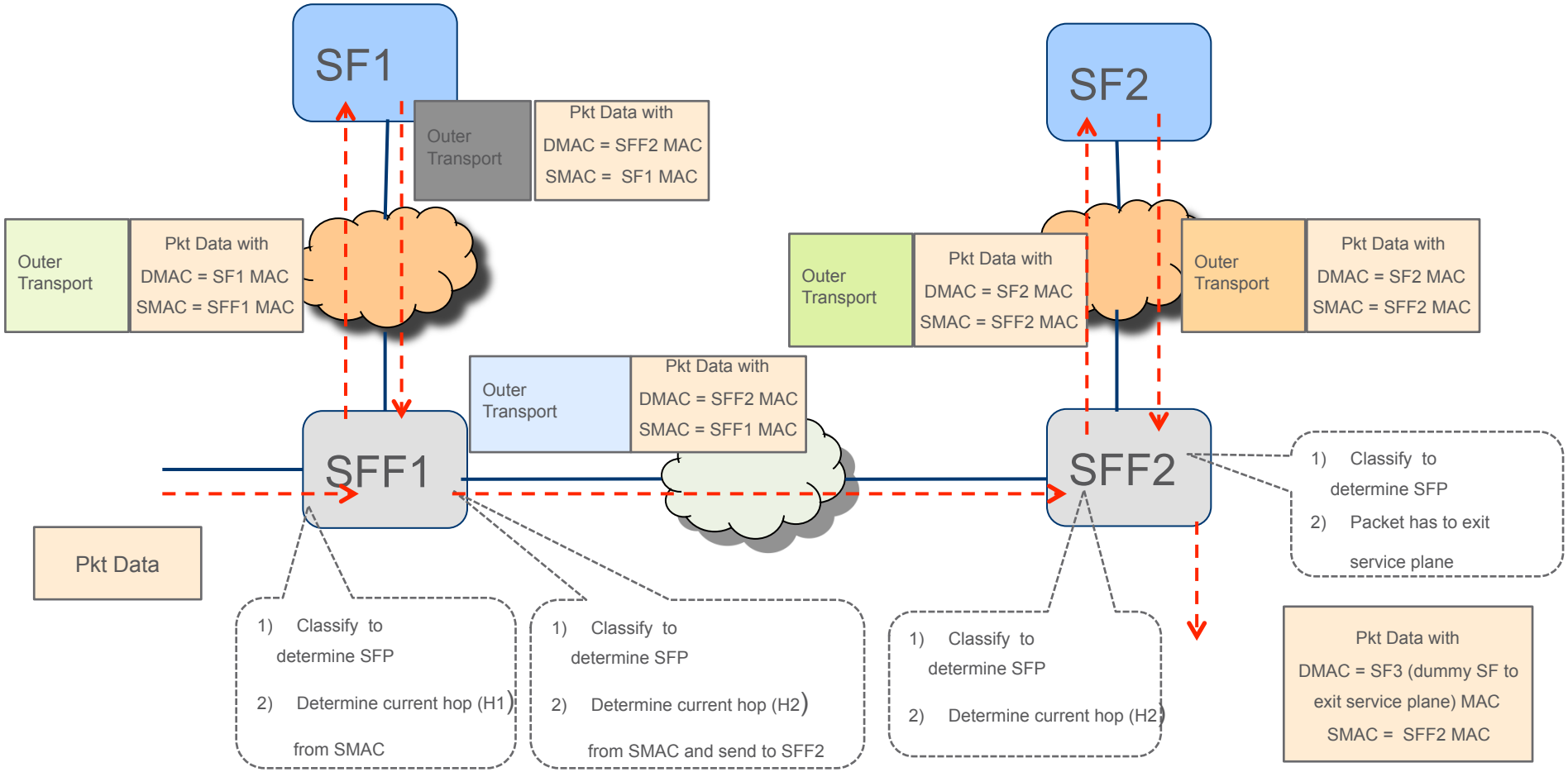
- Need for non-NSH service plane
  - Not all SFFs are NSH compliant
  - NSH support in OpenFlow is not standardized yet
  - NSH implementation in h/w SFFs is even more challenging
  - NSH EthType may not be recognized by some network elements

- Challenges due to absence of NSH
  - Service plane re-identification at every service hop
    - ACL classification at every SFF.
    - No support for dynamic header changes.
  - Identification of the service hop within service path
    - Service index is not available.
    - Determine the hop within an SFP using packet contents.

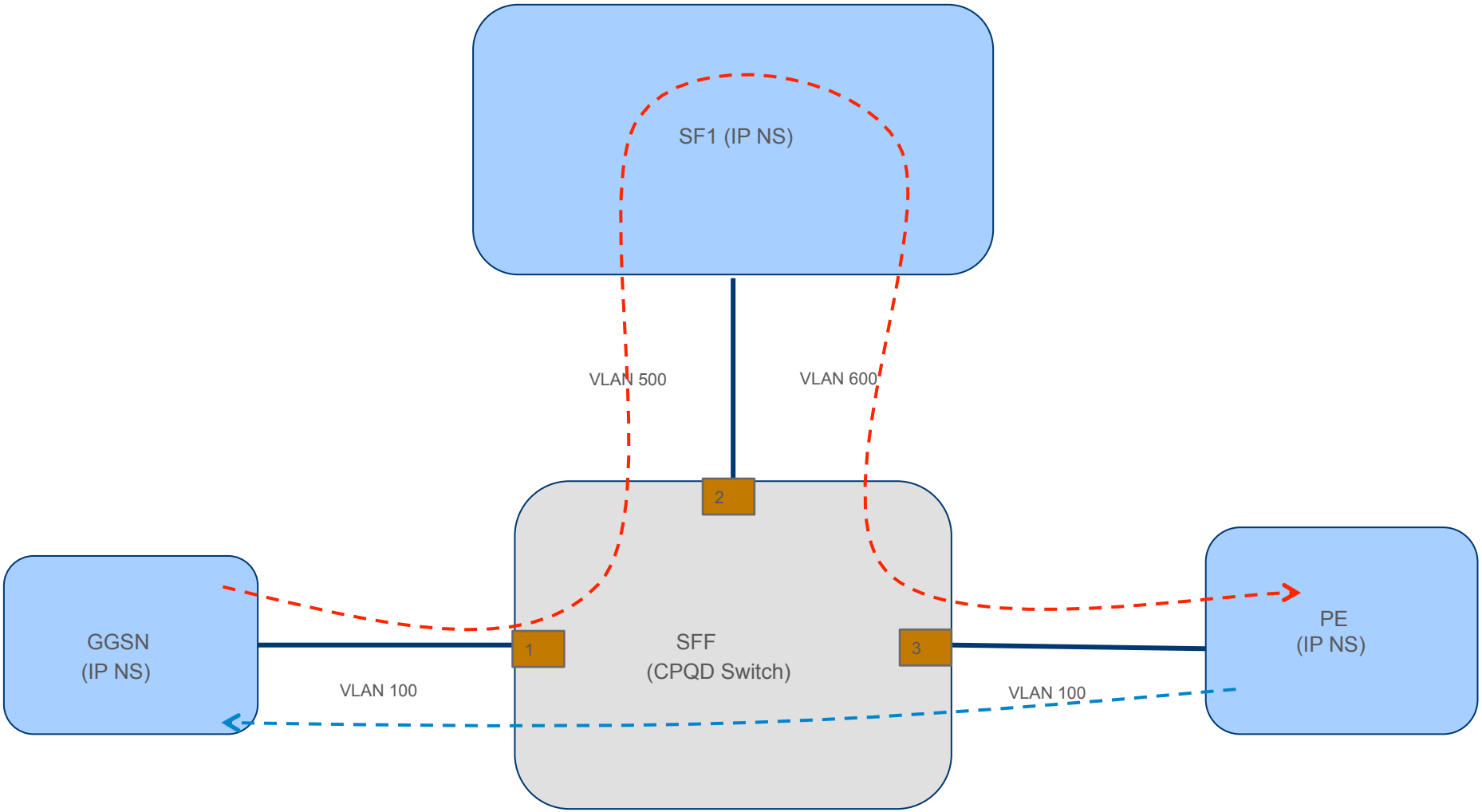
OPENFLOW: NO SFC ENCAP
(ERICSSON'S CONTRIBUTION)

- No SFC encapsulation (i.e. no NSH)
- OpenFlow 1.3.1 based implementation.
- L2 reachability of SFs and SFFs provided in Yang files.
  - MAC & VLAN of the SFs/SFFs for L2 connected SFs/SFFs
  - MAC & VLAN of the gateway for SFs/SFFs across L3 hops.
- Does not preclude various transports (VxLAN, GRE)
  - Programmed outside SFC with/without OpenFlow
- Packet re-reclassification at every SFF.
- Determines service hop based on MAC of previous hop.

NO SFC ENCAP WITH L2 REACHABLITY: EXAMPLE SFP



SFC USE CASE EXAMPLE DEEPDIVE



SFC USE CASE EXAMPLE DEEPDIVE…

JSON for Service Function Path (for RESTCONF generated from Yang)

```
{
  "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

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
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
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…

Flows created by ODL

Table 2: Packet classification

{table="2", match="oxm{in_port="1", eth_type="0x800"}", dur_s="155", dur_ns="272000", prio="32768", idle_to="0", hard_to="0", cookie="0x0", pkt_cnt="0", byte_cnt="0", insts=[meta{**meta="0x17"**, mask="0xffffffffffffffff"}, goto{table="3"}]},

{table="2", match="oxm{in_port="2", eth_type="0x800"}", dur_s="155", dur_ns="218000", prio="32768", idle_to="0", hard_to="0", cookie="0x0", pkt_cnt="0", byte_cnt="0", insts=[meta{**meta="0x17"**, mask="0xffffffffffffffff"}, goto{table="3"}]},

{table="2", match="oxm{in_port="3", eth_type="0x800"}", dur_s="155", dur_ns="150000", prio="32768", idle_to="0", hard_to="0", cookie="0x0", pkt_cnt="0", byte_cnt="0", insts=[meta{**meta="0x27"**, mask="0xffffffffffffffff"}, goto{table="3"}]},

Table 3: Hop identification

{table="3", match="oxm{metadata="0x17", metadata_mask="0xffff", eth_src="00:00:08:01:02:01", eth_type="0x800"}", dur_s="36", dur_ns="228000", prio="256", idle_to="0", hard_to="0", cookie="0x14", pkt_cnt="0", byte_cnt="0", insts=[apply{acts=[set_field{field:eth_dst="00:00:08:01:09:01"}, vlan_psh{eth="0x8100"}, set_field{field:vlan_vid="100"}]}, goto{table="10"}]},

{table="3", match="oxm{metadata="0x27", metadata_mask="0xffff", eth_src="00:00:08:01:09:01", eth_type="0x800"}", dur_s="36", dur_ns="191000", prio="256", idle_to="0", hard_to="0", cookie="0x14", pkt_cnt="0", byte_cnt="0", insts=[apply{acts=[set_field{field:eth_dst="00:00:08:01:01:01"}, vlan_psh{eth="0x8100"}, set_field{field:vlan_vid="100"}]}, goto{table="10"}]},

{table="3", match="oxm{all match}", dur_s="36", dur_ns="196000", prio="100", idle_to="0", hard_to="0", cookie="0x14", pkt_cnt="0", byte_cnt="0", insts=[goto{table="4"}]},

Table 4: Sending Packet to the First Hop

{table="4", match="oxm{metadata="0x17", metadata_mask="0xffff", eth_type="0x800"}", dur_s="36", dur_ns="278000", prio="256", idle_to="0", hard_to="0", cookie="0x14", pkt_cnt="0", byte_cnt="0", insts=[apply{acts=[set_field{field:eth_dst="00:00:08:01:02:01"}, vlan_psh{eth="0x8100"}, set_field{field:vlan_vid="500"}]}, goto{table="10"}]},

USE CASE 3: NSH WITH OPENFLOW

- Mainly meant for non-OVS, OpenFlow compliant switches
- Service plane is similar to Use Case 1
- Instead of OVSDB commands OpenFlow is used as southbound to program SFFs

REFERENCES

- ODL SFC Wiki Page
https://wiki.opendaylight.org/view/Service_Function_Chaining:Main

- Service Function Chaining Problem Statement IETF Draft
https://datatracker.ietf.org/doc/draft-ietf-sfc-problem-statement/

- 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 !

OPEN DAYLIGHT

SDN OpenFlow WORLD CONGRESS