SFC Opensource Experience

Reinaldo Penno (repenno@cisco.com)

Summary

- Control Plane
  - Controller = Opendaylight
  - Device = REST Server
- Data Plane
  - VXLAN+NSH

Control Plane

- Implemented as a application on top of Opendaylight controller
- Java, ~12K LOC and counting…
- JavaScript (UI), ~30K LOC and counting…
- Many intricate scenarios. Example:
  - Deleting, modifying SFs in use by Service Function Paths
  - Deleting modifying SFFs in use by active Service Function Paths
  - Deleting, modifying classifiers
  - High-availability
- Multiple southbound protocols: Openflow, Ovs, Netconf, etc.

Control Plane - Southbound

- Currently two implementations, Openflow renderer and REST.
- The SB REST implementation allows the southbound to be almost the same as the NBI. One protocol instead of multiple.
- A REST Server maps southbound requests to the specific device protocol
- This approach has considerable advantages…and some disadvantages of course.

Data Plane - I

- VXLAN+NSH (**draft-quinn-sfc-nsh-03)**
- Python
- SFF ~ 400 LOC
- SF (just networking part) ~ 200 LOC
- Very simple.
- Everything User-space. This alone makes approach very attractive and opens many possibilities. Example is SF and SFF implementation in end-hosts and CPEs respectively

Data Plane - II

- VXLAN+NSH
- Open vSwitch as Service Function Forwarder
- OVS patch for NSH
- Openstack scenario (multiple bridges)
- Co-located REST Server receives controller requests and configures switch
- Considerable difficulty in setting data plane in order for context to be passed around. Multiple bridges, multiple hops, multiple locators.
- Still ironing out the kinks

More info and next steps

- https://github.com/opendaylight/sfc
- https://wiki.opendaylight.org/view/Service_Function_Chaining:Main
- Further implementation for next Opendaylight release
- Hardening Openstack scenario
- Services Chaining with Netconf
- Multiple Service Function selection algorithms
- Others…