An Implementation Model and Solutions for Stepwise Introduction of SDN

Hiroki Nakayama†, Tatsuo Mori‡, Satoshi Ueno‡, Yoshihide Watanabe‡, Tsunemasa Hayashi†
†BOSCO Technologies Inc.
‡NTT Communications Corporation
Background

• Software-Defined Network (SDN) has been interested in the field of network management.
  – It enables flexible and uniform management.
  – It has been expected to overcome the issues of network administrations.
    • Reduction of human error by reducing human intervention.
    • Providing high quality network with small cost by integrating network resource.
• Example of actual use case
  – Data Center
    • Using network resource with aggregated control
  – Wide Area Network
    • Rapid control against service dependent network
  – Security
    • Countermeasure against DDoS
Problem on conventional SDN implementation model

- Adaption against conventional network protocols
  - Necessary to handle traditional network routing protocols.
  - Implementing against closed local network
    - Requires to change almost all of the switches in the network
- Issues on the scalability
  - Load against controller increases depending on the scale of controlled network
Objective

• Propose the stepwise implementation model for SDN.
  – Adaption against traditional network routing protocols.
  – Improvement of scalability by distributing controllers.

• Cooperation with traditional IP networks and implementation of network resource management faculty.
  – Run Quagga on each SDN switches to exploit management function of traditional IP network resources.
  – Run SDN controller on each SDN switches to construct a hierarchical and de-centralized structure.

• Implementation of prototype of proposed model.
  – Demonstrate that our proposal can be useful.
Proposal

• Improvement of scalability
  – Hierarchical structure enables to distribute the load of controller.
  – Processing load derived by adapting against traditional routing protocol can be reduced.

• Possibility of stepwise implementation
  – SDN can be implemented with small implementation costs.
Implementing each module on SDN switch

- OF ports cannot send out packets without OF functions.
  - Enable daemon program to communicate with others via OF ports.
  - Functions to packet out local transferred packets (SDN-NAPT).

- Packet transfers are based on flow information.
  - Translation function between routing information and flow entry.

Quagga on Linux machine

Quagga on SDN switch
Outline of SDN-NAPT

- RAW socket for internal packet translation and OF function for external packet translation are used for communication.
  - RAW socket is adopted to support the equivalence of translation.
  - Not only BGP but also any service can be provided.

<table>
<thead>
<tr>
<th>Global</th>
<th>Local (Loopback)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst</td>
<td>src</td>
</tr>
<tr>
<td>10.10.0.4:179</td>
<td>10.10.0.1:37348</td>
</tr>
<tr>
<td>src</td>
<td>dst</td>
</tr>
<tr>
<td>127.0.0.2:8179</td>
<td>127.0.0.1:44156</td>
</tr>
</tbody>
</table>
Implementation of SDN-NAPT

- Loop back address will be default gateway from the local service.
  - IP address of itself cannot be set as default gateway.
  - IP address of SDN controller have to be set as destination address.

- BGP makes a connection only against BGP peer node.
  - Spoofing function against BGP process is required.

- BGP establishes peer connection.
  - It is necessary to notice the information of destination node to controller for an active connection from BGP process.
Sequence of session initiation of SDN-NAPT

Override the IP address of destination peer after establishment of BGP session.

Notice information of destination node before connection establishment.
Translation of routing information to flow entry

• Correspondence between routing information and flow entry

<table>
<thead>
<tr>
<th>Kernel routing table</th>
<th>Flow entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>match : nw_dst</td>
</tr>
<tr>
<td>Gateway</td>
<td>actions : setfield</td>
</tr>
<tr>
<td>Metric</td>
<td>priority</td>
</tr>
<tr>
<td>Iface</td>
<td>actions : output</td>
</tr>
</tbody>
</table>

– MAC address is required to be handled by flow entry, where kernel routing table does not.

• Example of translation

– Kernel routing table

```
# route -n
Kernel IP routing table
Destination       Gateway       Genmask      Flags Metric Ref    Use Iface
20.10.0.0         10.10.0.2    255.255.0.0  UG   0   0         0  eth1
```

– Flow entry

```
# ovs-ofctl dump-flows br0 --protocol=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
  cookie=0x0, duration=2.975s, table=0, n_packets=0, n_bytes=0, priority=1000,
in,nw_dst=20.10.0.0/16 actions=set_field:08:00:27:e0:db:bf->eth_dst,
set_field:08:00:27:bf:22:e9->eth_src,output:2
```
Evaluation environment

- Conducted an operational verification
  - Correct BGP communication between OFS and each GWs
  - Routing information and flow entry of each switches
Evaluation result

- Dump of packets between GreenGW and OFS

BGP messages are transferred as expected
Evaluation result cont.

- Dump of flows in each node
  - GreenGW
    
    ```
    # route -n
    Kernel IP routing table
    Destination Gateway Genmask Flags Metric Ref Use Iface
    0.0.0.0 10.10.0.1 0.0.0.0 UG 0 0 0 eth0
    10.10.0.0 0.0.0.0 255.255.0.0 U 0 0 0 eth0
    10.20.0.0 10.10.0.1 255.255.0.0 UG 1 0 0 eth0
    20.10.0.0 0.0.0.0 255.255.0.0 U 0 0 0 eth0
    30.10.0.0 10.10.0.1 255.255.0.0 UG 0 0 0 eth0
    40.10.0.0 10.10.0.1 255.255.0.0 UG 0 0 0 eth0
    ```
  - OFS
    
    ```
    Switch# ovs-ofctl dump-flows br0 --protocol=OpenFlow13
    OFPST_FLOW reply (OF1.3) (xid=0x2):
    cookie=0x0, duration=216.25s, table=0, n_packets=0, n_bytes=0,
    priority=1000,ip,nw_dst=40.10.0.0/16 actions=set_field:00:0a:85:07:0c:34-
    >eth dst,set field:00:1e:08:93:38->eth src,output:4
    cookie=0x0, duration=200.05s, table=0, n_packets=0, n_bytes=0,
    priority=1000,ip,nw_dst=20.10.0.0/16 actions=set_field:00:0a:85:07:0c:38-
    >eth dst,set field:00:1e:08:93:36->eth src,output:2
    cookie=0x0, duration=218.471s, table=0, n_packets=31, n_bytes=2692,
    priority=10 actions=CONTROLLER:65535
    ```

Routing information are transferred as expected

Routing information of AS1030 (OFS)

Routing information of AS1020 (Green)
Conclusion and future work

• Conclusion
  – We proposed the implementation model which can overcome the problems of current SDN implementation model.
  – We introduced the specific design and implementation of our prototype and demonstrated that our implementation can handle conventional network protocols.

• Future work
  – Implementation of distributed controller for the wide area network.
  – Evaluation using real monitored data in the service network.
  – Overcome the problem of small flow entry size of current OpenFlow switch.

Our implementation will be deployed as open source