Gateway Function for Network Slicing
I-D.homma-rtgw-g-slice-gateway-01

Shunsuke Homma –NTT
Xavier De Foy –InterDigital Inc.
Alex Galis –University College London
Luis M. Contreras-Telefonica
Reza Rokui-Nokia
Background

• Network slicing would be needed for applying network to various types of services and applications (e.g., broadband, high-reliability and low latency, etc.)
• Network slicing is supported by 5G architecture, and recently, several SDOs, including 3GPP, are discussing it. (IETF TEAS WG also started to discuss transport slice)
E2E Network Slice Realization

- Network slice is composed of network slice subnets.
  * The Network Slice Subnet represents a group of network functions (and connectivity among them) that form part or complete constituents of a Network Slice.

- For providing an end to end communication services, stitching of several network slice subnets would be required.

http://www.3gpp.org/ftp/Specs/archive/22_series/22.891/22891-e20.zip
Slice Gateway (SLG)

• In addition to stitching subnets, several functionalities for handling slices and traffic would be required at boundaries of domains.

• SLG provides data-plane functionalities (e.g., slice selection, QoS control, encap/decap, etc) for handling slices with interaction with the management-plane.

* NSI: Network Slice Instance
* NSSI: Network Slice Subnet Instance
## Requirements for SLG

- SLG is realized by one device or a group of modules
- Functionalities required for SLG are below:

<table>
<thead>
<tr>
<th>Handling underlay infrastructure</th>
<th>On Data-Plane</th>
<th>On Control/Management-Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Classification</td>
<td>• Forwarding</td>
<td>• IF to controller/orchestrator</td>
</tr>
<tr>
<td>• Isolation</td>
<td>• Isolation</td>
<td>• Address resolution/Routing</td>
</tr>
<tr>
<td>✓ QoS control per NS</td>
<td>✓ Traffic Engineering</td>
<td>• AAA</td>
</tr>
<tr>
<td>✓ Traffic Engineering</td>
<td>• Service Chaining</td>
<td>• OAM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control services on NS</th>
<th>• Classification</th>
<th>• IF to service management system</th>
</tr>
</thead>
<tbody>
<tr>
<td>• QoS control per flow</td>
<td>• Steering/Service Chaining</td>
<td>• Telemetry collection</td>
</tr>
<tr>
<td>• Steering/Service Chaining</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview of SLG Architecture

- SLG is composed of data plane entity and controller.
- SLG controller may accommodate multiple SLG data plane entities.
- SLG controller has two types of APIs:
  - For managing slices (connected with operation systems)
  - For controlling user traffic on slices (connected with customers/tenants)
Work on SLG

- PoC in MEF: Feasibility test of E2E network slicing across multiple administrative domains

- PoC at NTT R&D Forum: Cooperation between network slices and game applications
PoC Scenarios in MEF

Inter-service provider network slice order and control is handled in LSO architecture with extensions:
- Slice customization
- Inventory management and order reception based on real-time service quality

Scenario 1: Semi-custom made slice
Combine of pre-defined slices of each service provider according to customers’ requirements

Scenario 2: Full-custom made slice
Customer specific configuration, i.e. routes and intermediate buildings of external service providers’ network based on E2E latency requirement

Copyright(c)2019 NTT,corp. All Rights Reserved.
PoC Scenarios at NTT R&D Forum

VR applications are developed and licensed by ZIKU Technologies, Inc.

Copyright(c)2019 NTT,corp.All Rights Reserved.
Example of API to Allocate Traffic to Slice

HTTP Body:
```
{  
  "connect": [  
    {  
      "src_ip": "192.168.1.10",  
      "dst_ip": "192.168.110.100",  
      "src_port": 3000,  
      "dst_port": 3001,  
      "ip_proto": "udp",  
      "slice_id": 10100,  
    },  
  ]  
}
```

Flow Identifier

HTTP Body:
```
{  
  "dip": 1,  
  "table_id": 0,  
  "priority": 2,  
  "match": {  
    "ipv4_src": "192.168.1.10",  
    "ipv4_dst": "192.168.110.100",  
    "udp_src": 3000,  
    "udp_dst": 3001,  
    "ip_proto": 17,  
    "eth_type": 2048"),  
  }  
  "actions": [  
    {"type": "METER", "METER_ID": 1},  
    {"type": "OUTPUT", "port": 2}]  
}
```
Next Steps

- Defining SLG northbound interface with referring definitions and specifications of NS-DT in TEAS WG.
- Providing use cases where SLG (i.e., subnet concept) would be beneficial.
- Documenting the results of PoCs if they are useful.

Your feedback would be appreciated.
Thank you