Location-aware Dynamic Network Provisioning

Van-Quyet Nguyen, Sinh-Ngoc Nguyen, Deokjai Choi, Kyungbaek Kim
School of Electronics and Computer Engineering
Chonnam National University
Gwangju, South Korea
quyetict@utehy.edu.vn, sinhngoc.nguyen@gmail.com, dchoi@jnu.ac.kr, kyungbaekkim@jnu.ac.kr

***Abstract*—With the wide development of network services and the increasing network quality of service (QoS) requirements, dynamic network provisioning, which offers a smart way to segment the network to support particular services, becomes an important problem. Past studies of dynamic network provisioning mostly consider dynamic bandwidth and delay assignment in order to maximize the system throughput and minimize communication overhead. These studies leveraged Software Defined Network (SDN) and Network Function Virtualization (NFV) to create dedicated end-to-end virtual networks which are called network slices. But, these works are lack of consideration of provisioning a dynamic network based on request locations where users would use the network to deploy their services. In this paper, a method for generating a network slice dynamically based on requested locations of users is proposed to support the location-aware dynamic network provisioning. The requested locations are obtained by users through a web interface, then the requested locations are encapsulated and sent to SDN controller which knows the geographical location of SDN switches. Then SDN controller sets up a location-aware network slice. To provide a viability of the proposed approach, we implemented the web interface and the protocol to share the requested locations.**

Keywords—Dynamic Network Provisioning, Software-defined networks; Network Function Virtualization; Location-aware Network Provisioning

I. INTRODUCTION

In recent years, the trend of service-oriented demands of network services calls for more flexible and efficient network provisioning. The dynamic network provisioning concept is a solution to provide flexibility and service customization in the existing networks via sharing the network resources, which includes physical resources (e.g., computation, storage, etc.) and logical resources (e.g., network functions, etc.) among various service providers. Generally, this concept relates to the optimization problems in resources allocation, to segment the network into network slices which support specific QoS requirements offered services.

There have been a few studies on dynamic network provisioning. For instance, in [1], the network provisioning algorithms for time-varying traffic patterns are proposed. The authors presented an adaptive tunnel bandwidth assignment policy that re-allocates bandwidths between the fixed-route tunnels based on the dynamic traffic variations. In [2], the authors proposed an algorithm for path finding in time-dependent transport networks with bandwidth guarantees. In their approach, the users specified the total needed volume, a maximum bandwidth, and the desired time window, then the proposed algorithm could find alternate allocation possibilities including earliest time for completion, or shortest transfer duration leaving the choice to the user. In [3], the authors investigated the dynamic resource provisioning for wireless virtualized networks (WVN) based on both physical parameters (i.e., wireless channel state information) and the medium access layer parameter (i.e., queue state information). The main aim of their proposal is to maximize the total rate of WVN subject to average queue stability condition of each user and average minimum required rate of each network slice.

Recently, there are several studies focusing on dynamic network slicing techniques to support network provisioning [4]-[6]. The dynamic network slicing enables the design, deployment, customization, and optimization of different network slices running on a common network infrastructure. It leverages SDN and NFV to create many dedicated end-to-end virtual networks which could be referred as network slices. A network slice is a group of physical or virtual (network, compute, storage) resources which can act as a sub-network and it can accommodate service components and network functions. Based on these characteristics, Internet service providers (ISPs) can exploit dynamic network slicing for reducing significantly operations expenditures.

However, the most of network provision approaches above needs the specified network parameters (e.g., bandwidth, delay, total volume), but they are lack of consideration of the request locations where users would use the network to deploy their services. For example, a university has three campuses in different cities in a country. They want to organize the online classes at these campuses together via video conference. So, they would need a network with a high quality for lecturing in this way. In this case, without network parameters, the network services provider need to consider to the requested locations to set up a network slice with performance guarantees.

In this paper, we propose a method for generating a network slice dynamically based on requested locations of users to support the location-aware dynamic network provisioning. The requested locations are obtained by users through a web interface, then they are packed and sent to SDN controller which knows the geographical location of SDN switches. Then SDN controller setups a location-aware network slice. To provide a viability of the proposed approach, we implemented the web interface and the protocol to share the requested locations.

II. Design of a Location-Aware Dynamic Network Provisioning System

We present the architecture of our proposed system for location-aware dynamic network provisioning as illustrated in Fig. 1. The architecture consists of three main modules: *Application Service Provider*, *Thin Abstraction Layer* (TAL), and *Intelligent Orchestrator*. We will take into account each module in more details as follows.

A. Application Service Provider

We design a web interface supporting users to send *Service Request* to the network service provider. The users declare their requirements with the information related to service content, context, and locations. For example, in case of video conference service, the users can provide the related information including resolution, encoder (content), time-widows (context), and three regions where they want to deploy the service (locations).

To support location-aware network provisioning, we design a region-selector web map as a gird of regions, in which the users can select the regions for their service registration. Each region in the map is drawn by a cell with size 5x5 kilometer square, and assigned by a unique identity region. We store the requested locations of the user in JSON format as shown in Fig. 2. Subsequently, it is sent along with other service requirements to the SDN controller via *TAL Service API*, and wait for service response.

B. Thin Abstraction Layer

This module is designed to receive the service requests from *Application Service Provider*, then handle the requests by creating network service abstraction models. Next, the *Network Request Models* are sent to orchestrator for finding an appropriate network slice that can meet the requirements of the users. We will take a look at each component of this module in more detail.

The first component is *TAL Service API* which provides the functions such as service creating or service updating. It is invoked by the web server in Application Service Provider, in which the users fill out and send their requests, then wait for the response result. As we presented before, the users send their request in JSON format; therefore, this component support to read JSON data, and send them to the next component for handling the request.

An important component is *Service Request Handler* which receives the service requests in JSON format, extracts them into some categories including content, context, and locations, then analyzes them to make *Network Request Models*. This module consists of two main sub-components: *Network Topology Generator* and *Network Parameters Generator*.

For the first one, to generate network topology based on requested locations of the user, the Network Location-awareness module is proposed. This module performs finding SDN switches which their geographical locations belong to one of the requested locations. As shown in Fig. 3, two tables are needed for implementing network location-awareness.

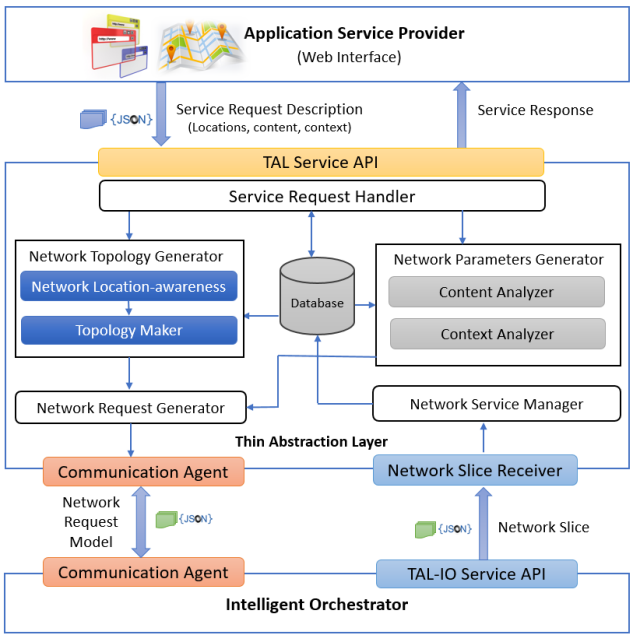

Fig. 1. An overview of the architecture

```
{
  "service_request": {
    "service_request_id": 1,
    "service_type": "video conference",
    "locatons": [
      {
        "region_id": 1,
        "north_east": "35.2625,127.0400",
        "south_west": "35.0375,126.6400"
      },
      {
        "region_id": 2,
        "north_east": "35.1375,126.9150",
        "south_west": "35.1125,126.8900"
      },
      {
        "region_id": 3,
        "north_east": "35.2125,126.8400",
        "south_west": "35.1875,126.8150"
      }
    ]
    ...
  }
}
```

Fig. 2. An example of service request in JSON format

The first table contains the request locations of the user which are obtained by using region-selector web map. The second table contains the region coordination of network devices (e.g., switches), in which each switch can cover the network of a large region. The region which covered by a switch is larger than a region (location) on the map. The geographical location of the covered region of the switch is identified by network administrator and stored in the database. Each region coordination in both two tables is represented by two corners named "north-east" and "south-west" with the format: "*latitude1, longitude1; latitude2, longitude2*".

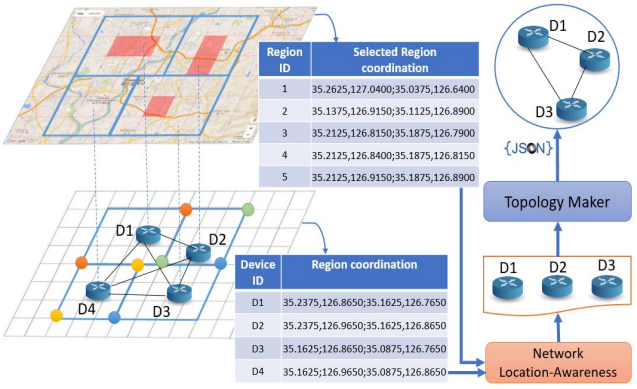

Fig. 3. Modeling network location-awareness

After getting the list of switches which can be used to provide the network for the user, the *Topology Maker* module create a network topology using mesh network type by default, then send the topology information to the *Network Request Generator* module.

For the *Network Parameters Generator* module, it analyzies the content and context of the service to generate the network parameters (e.g., bandwidth, delay, etc.). To end this, in our previous work [7], we proposed a machine learning based smart service abstraction layer for future network provisioning. However, in this paper, we mainly focus on location-aware network provisioning, therefore, how to generate network parameters based on content and context of the service is out of scope of this work.

After finishing making network topology and network parameters for the service request, the *Network Request Generator* module is used to pack them together into a *Network Request Model.* Then, this model is sent to orchestrator which we named Intelligent Orchestrator (IO), via *Communication Agent*. Next, the *Network Slice Receiver* module waits for the network slice response which meet the requirements of the user. The *Network Slice Receiver* send the network slice to the *Network Service Manager* module which is used to match the request of the users with their the network slice. The selected network slices will be stored in the database and response to the user web interface corresponding to the user request.

C. Intelligent Orchestrator

This module is located inside SDN control panel. It receives *Network Request Model* via *Communication Agent,* then extracts the request including a method which corresponds to a network function (e.g., create tenant, modify tenant, etc.) and the parameters (e.g., network topology, bandwidth, etc.). It uses the standard protocols (e.g., OpenFlow, SNMP, etc.) to communicate with network devices in SDN data panel to enable a network slice which meets the service requirement. Here, the network slice is a group of physical or virtual (e.g., network, compute, storage) resources which can act as a sub network and/or cloud. In this paper, we do not focus on network slides creation, it is presented in our different work [8], which proposed an approach for establishment and management end-to-end slices of various types of resources from distributed and diverse infrastructure.

Besides, this module provides the *TAL-IO Service API* to TAL for retrieving all networks slices which met the network request models.

III. Implementation

A. Implementation of the Network Location-Awareness module

The *Network Location-Awareness* module utilizes an algorithm to find out the switches which have the cover region, which contains at least one requested location of the user. After that, the *Topology Maker* module uses the list of switches which are found by the *Network Location-Awareness* module to make a network topology, and stores it in JSON format.

Algorithm 1 Procedure *FindSwitchesWithNLA* is used to find out the switches which have the cover region containing at least one requested location of the user.

Require: *fileRequest* is the JSON file that contains requested locations of a user; *connStr* is the connection string to the database
Ensure: *lstO*: A set of switches

```
1:  List<Region> lstR ← ExtractLocations(fileRequest);
2:  List<Switch> lstS ← GetListSwitches(connStr);
3:  List<Switch> lstO ← new List<Switch>();  /* output */
4:  for each s ∈ lstS do
5:      for each r ∈ lstR do
6:          if s.region.ne.lat ≥ r.ne.lat &&
               s.region.ne.lng ≥ r.ne.lng &&
               s.region.sw.lat ≤ r.sw.lat &&
               s.region.sw.lng ≤ r.sw.lng   then
7:              lstO.add(s);
8:          end if
9:          break;
10:     end for
11: end for
12: return lstO;
```

The procedure "*FindSwitchesWithNLA*" in *Algorithm 1* illustrates our implementation for the *Network Location-Awareness* module. In this algorithm, two procedures: *ExtractLocations* which returns a list of *Region* objects (line 1) and *GetListSwitches* which returns a list of *Switch* objects (line 2) are not shown in details in this paper. Then, for each *Switch,* the algorithm checks whether or not at least one requested location (*Region*), which belong to the cover region of that *Switch* (lines 4-6), exists. Finally, the algorithm returns a list of *Switch* objects which could be used to make a network topology to provide the network service slices for the user.

B. Implementation of the TAL-IO Communication Agents

In this section, we describe the implementation of TAL-IO communication agents by using JSON-RPC. To do this, we define several data structures and the rules for communicating between a client agent in TAL and a server agent in IO. The data structures of the request and the response are represented in JSON format as shown in Fig. 4.

The client agent will send a request message which has four members (properties) including *method*, *params*, *jsonrpc*, and *id*. Firstly, the *method* property contains the name of the method to be invoked. Secondly, the *params* property contains the *Network Request Model* which generated by our method.

Algorithm 2 Procedure *SendRequest* is used to send message from the client agent to the server agent.

Require: *fileNRM* is the path file of the network request model; *funcName* is the name of the invoke method; *url* is the address of the server agent including host and port; *headers* contains the HTTP POST variables

Ensure: *resp* is a response message

```
1: fileRequest ← open(fileNRM);
2: nrmParams ← json.loads(fileRequest.read());
3: requestId ← genRequestId();
4: message ← {"method": funcName, "params": nrmParams,
   "jsonrpc":"2.0", "id": requestId};
5: resp = requests.post(url, json.dumps(message), headers);
6: return resp;
```

An example of the content of a *Network Request Model* is as shown in Fig. 5. Thirdly, The *jsonrpc* property is a string specifying the version of the JSON-RPC protocol. Finally, the *id* property is an identifier established by the client agent for the purpose of correlating the context between the two objects: Request and Response.

```
[ Request ]
create_tenant_msg = {
      "method": "create_tenant",
      "params": CREATE_TENANT_PARMS,
      "jsonrpc": "2.0",
      "id": "1",
}

[ Response ]
resp_create_tenant_msg = {
        "method": "create_tenant",
        "result": "5d460e88a8ea450c9c992e7a07d34889",
        "jsonrpc": "2.0",
        "id": "1",                          TenantID
}
```

Fig. 4. Example of request and response message

```
{
  "topology": {
    "nodes": ["1","2","3","4"],
    "links": [
      { "src": "1","dst": "2"}
      ...
    ],
    "type": "dedicated"
  },
  "service_qos": {
    "bandwidth": {
      "scale": "5",
      "unit": "mbps",
    },
    "delay": {
      "scale": "100",
      "unit": "msec",
    },
    "throughput": {
      "scale": "99",
      "unit": "percent",
    }
  },
  "resource": {
    "net_type": [
      "WLAN",
      "Cellular"
    ],
    "processor": {
      "core": "2",
      "frequency":{
          "scale": "1.6",
          "unit": "ghz"
      }
    },
    "memory": {
      "scale": "64",
      "unit": "mb"
    },
    "storage": {
      "scale": "240",
      "unit": "mb"
    }
  }
}
```

Fig. 5. Example of a Network Request Model

For the response message, it includes four properties: *method*, *jsonrpc*, *id, result* or *error*. Three first properties are the same as the request message. The *result* property is a string containing the network *tenant id* if exists a network slice meet the *Network Request Model,* otherwise, the server agent send an *error* back to the client. However, in this paper, we do not describe the algorithms for network slicing in SDN. Therefore, the client agent will be received the *result* with a default value if there was no error triggered during invocation.

Algorithm 2 illustrates the procedure *SendRequest* which is used to send the request from the client agent in TAL to the server agent in the SDN controller. Here, we have implemented two functions: *"create_tenant"* and *"modify_tenant",* in which the server agent can receive the network request model and response to the client agent.

IV. Conclusion

This paper proposed a method for generating a network slice dynamically based on requested locations of users to support the location-aware dynamic network provisioning. The requested locations are obtained by users from a web interface, then they are encapsulated and sent to SDN controller which knows the geographical location of SDN switches. Then SDN controller sets up a location-aware network slice. To provide a viability of the proposed approach, we implemented the web interface and the protocol to share the requested locations.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00314) supervised by the IITP (Institute for Information & communications Technology Promotion). This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B4012559).

References

[1] Sharma, Vicky, et al. "Dynamic network provisioning for time-varying traffic." Journal of Communications and Networks 9.4 (2007): 408-418.

[2] Balman, Mehmet, et al. "A flexible reservation algorithm for advance network provisioning." Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society, 2010.

[3] Jumba, Vikas, et al. "Dynamic resource provisioning with stable queue control for wireless virtualized networks." Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on. IEEE, 2015.

[4] Li, Yong, and Min Chen. "Software-defined network function virtualization: A survey." IEEE Access 3 (2015): 2542-2553.

[5] Gde, Dharma N., et al. "Design of service abstraction model for enhancing network provision in future network." Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific. IEEE, 2016.

[6] Chatras, Bruno, U. Steve Tsang Kwong, and Nicolas Bihannic. "NFV enabling network slicing for 5G." Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on. IEEE, 2017.

[7] Duc-Tiep, Vu, et al. "Design of Machine Learning based Smart Service Abstraction Layer for Future Network Provisioning." In Procceding of the KIPS Fall Conference, 2016.

[8] Seung-Joon, Seok, et al. "multiFIA – Multi-dimensional Future Internet Architecture." In Procceding of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, 2017.