위치인지 능동 네트워크 제공을 위한 프레임워크 구현

뉘엔휴쥐, 뉘엔반퀴엣, 뉘엔양쯔엉, 권태용, 염성웅, 김경백
전남대학교 전자컴퓨터공학부
e-mail: huuduyce@gmail.com, quyetict@utehy.edu.vn, truongnguyengiang.bk@gmail.com, taaii6569@naver.com, yeom4032yeom4032@gmail.com, kyungbaekim@jnu.ac.kr

Implementation of a Framework for Location-aware Dynamic Network Provisioning

Huu-Duy Nguyen, Van-Quyet Nguyen, Giang-Truong Nguyen, Taeyong Kwon, Sungwoong Yeom
Kyungbaek Kim
Department of Electronics and Computer Engineering, Chonnam National University

Abstract

In these days, providing flexible and personalized network services subject to customers' requirements becomes an interesting issue for network service providers. Moreover, because each network service provider own finite network resources and infrastructure, dynamic network provisioning is essential to leverage the limited network resources efficiently and effectively for supporting personalized network services. Recently, as the population of mobile devices increases, the location-awareness becomes as important as the QoS-awareness to provision a network service dynamically. In this paper, we propose a framework for providing location-aware dynamic network services. This framework includes the web user interface for obtaining customers' requirements such as locations and QoS, the network generator for mapping the requested locations and network infrastructure, the network path calculator for selecting routes to meet the requested QoS and the network controller for deploying a prepared network services into SDN(Software-Defined Networking) enabled network infrastructure.

1. Introduction

So far, there are a few studies considering locations where users connect to the network and use services for providing better network paths under wireless networks [1][2]. In these works, locations of users are collected implicitly by using current GPS coordinates. In these days, more personalized network services are required by vigorous users and location information may not tightly coupled to the current location of users or devices. A user may want to select a number of specific locations in order to deploy his applications and network services. For example, a project leader in a company, which has many branches located at different location in a city or a country, wants to initiate a conference call application or operate some applications related to a human resource management system. To support these needs, a location-aware network provisioning can be a solution. It enables to generate a network slice based on requested locations for users to receive proper network services [3].

Moreover, network provisioning needs to consider various types of application needs, that is, different QoS levels requested by users. Users expect a specific QoS level to be guaranteed on the network services over the requested locations. Naturally some network connections between requested locations can be provided in somehow, but it does not means that the QoS level is guaranteed by the provided network connections. Consequently, to provide proper personalized network services, it is needed to calculate network connections among the requested locations dynamically based on the given QoS requirements.

Recently, the concept of software defined networking (SDN) becomes popular and core network infrastructures may provide network slices to establish sub-networks with various properties dynamically. It is possible because the data forwarding layer is decoupled to network control layer and a SDN controller has a global view of an entire network as well as full control functionality for the network [5].

In this paper, we implement an SDN-based framework which obtains user requests including locations and QoS level and deploy a proper network service dynamically. Through a map-based web user interface, a user can provide requested locations and a QoS level which represents the required bandwidth and delay. After obtaining the user request, the framework sorts out the corresponding network switches and generates network paths between the corresponding network switches in order to guarantee the given QoS level. Then, a SDN controller deploys the flow rules to the corresponding network switches to initiate the generated network paths.

2. Design of the proposed framework

In this section, we introduce the design of the proposed framework for a location-aware dynamic network provisioning with QoS constraints. Fig. 1 depicts the overview of the proposed framework. This framework aims at handling user requests including selected locations and pre-specified QoS. We describes five main components in our proposed framework more detail as below.

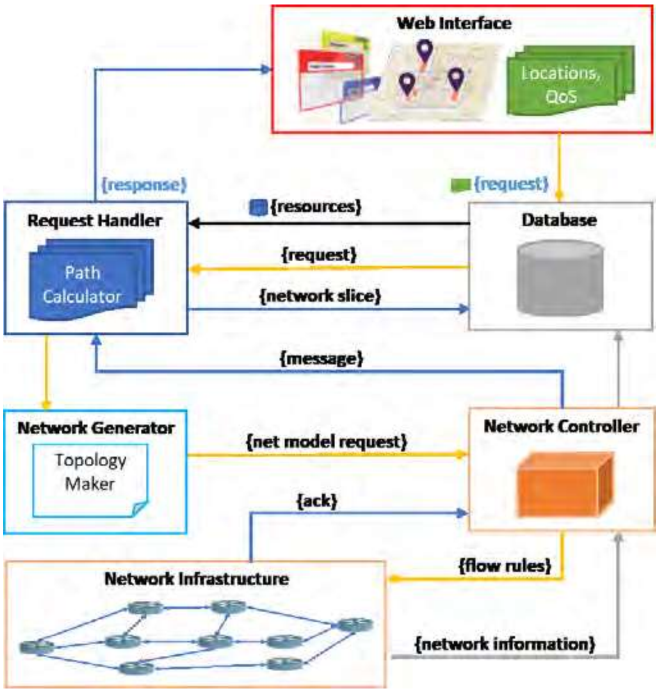

Figure 1. Overview of the proposed framework

Web Interface is a map-based region-selector web user interface (UI) for obtaining user's requests. In this UI, a user can select their desired locations and to specify required QoS level through network parameters such as bandwidth and delay. When a user submits these requirements, a request for a network service with the given requirements is created and it is sent to the database. After obtaining the request, the database finds the corresponding switches in the network mapping to the requested locations, and stores the results of the corresponding switches along with the request.

Request Handler is a component which handles requests from the database periodically and responds to user requests the results of network provisioning. When it picks a request from a database, it pass the requirements such as locations and QoS levels to *Path Calculator*. Path Calculator employs some algorithms to find some candidate routing paths between the corresponding switches for requested locations. These algorithms relies upon the information of the current network state stored in the database. The candidate routing paths are handed to *Network Generator* for further processing of a request. This component also has responsibility of responding the results of handling a request to a user. If this component receives a success message from *Network Controller*, it informs that the requested network is successfully provisioned and ready to use. Otherwise, a failure message is released whether there is no routing path which meets the request or no network slice is available.

Network Generator is a component to generate a logical network topology only contains switches along the candidate routing paths calculated by *Request Handler*. The topology then is encapsulated in a form of network model request and sent to *Network Controller*.

Network Controller is a component which holds an SDN application which is responsible for receiving and handling network model requests which come from *Network Generator*. When a network model request arrives, it generates flow rules and deploy them into the corresponding switches in order to make a network slice for the user request. If a network slice is deployed successfully, it notices *Request Handler* a success message, otherwise a failed message is sent. Also, *Network Controller* communicates with network infrastructure and updates network information to database periodically.

Database is a warehouse for storing all useful network information retrieved by *Network Controller* dynamically and periodically. Besides, every request from users through *Web Interface* component as well as result of request handling are maintained in the database.

Figure 2. A view of map-based Web UI with several selected locations by a user

3. Implementation of the framework

In order to show the viability of the framework, we implemented the framework with Ajax, Python, MySQL, Mininet and OpenDaylight. *Web Interface* is implemented by using Ajax and Fig. 2 shows a sample view of region selection over the area near to Gwangju, South Korea. *Request Handler* is implemented by using Python and MySQL is used for implementing *Database*. *Network Generator* is implemented by using Python with Mininet and *Network Controller* is implemented by using Python with OpenDaylight APIs.

All requirements from a user request are pushed in *database* and they contains locations, bandwidth, delay and others like service type, service content, and so on.

The user requested area are described by using geographical grid and their coordinates. To determine the corresponding switches to user requested locations, *database* holds the geographical coordinates of the covered area of a network switch. The covered area is described as a polygon, and currently it is assumed that covered area of a network switch is given by the network providers. *Web Interface* component finds the corresponding switches by checking whether the requested locations are overlapped or covered by the covered area of each switch.

Currently, the QoS requirements are bandwidth and delays, and they are obtained through *Web Interface*. This QoS information is used to determine the network paths between the switches, and it is considered as a Constrained Shortest Path (CSP) problem. We adopt an algorithm for finding least cost paths with delay constraint, named as Lagrange Relaxation based Aggregated Cost (LARAC), and implement the algorithm in *Path Calculator*. This algorithm is able to give a lower bound of minimizing a path cost function of a routing path subject to delay constraint [4].

When candidate routing paths between switches are calculated, we generate a logical network topology based on the current network status of Mininet. Then, based on the topology, we generate flow rules to deploy the logical network topology into the Mininet by using OpenDaylight APIs. Flow rules are generated as shown in Fig. 4 and imposed on switches along routing paths of logical topology for each request.

```
"in_port=6, dl_type=0x0800, nw_dst=10.0.0.3, action=output:4"
"in_port=2, dl_type=0x0800, nw_dst=10.0.0.3, action=output:4"
"in_port=5, dl_type=0x0800, nw_dst=10.0.0.3, action=output:6"
"in_port=5, dl_type=0x0800, nw_dst=10.0.0.3, action=output:2"
"in_port=7, dl_type=0x0800, nw_dst=10.0.0.3, action=output:8"
```

Figure 4. An example of flow rules

4. Conclusion

This paper implements a framework aiming at support for location-aware dynamic network provisioning with QoS constraints using SDN network. Each request includes three main requirements location, bandwidth and delay which is obtained through a map-based web UI. This UI then maps between all requested locations and underlying network switches. In the core of our framework, we implement a network path calculator with LARAC algorithm, which rely upon user's given QoS to find appropriate routes. Eventually, one logical topology representing a network slice is generated for each request and impose flow rules on network switches through an SDN-based controller module. Besides, topology generation is also important, so we are currently working on enhancing performance of topology generation.

Acknowledgment

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). This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2016-0-00314) supervised by the IITP(Institute for Information & communications Technology Promotion). This research was one of KOREN projects supported by National Information Society Agency (18-951-00-001).

References

[1] Anagha Raich, Amarsinh Vidhae. "Best Path Finding using Location aware AODV for MANET." International Journal of Advanced Computer Research Volume 3, Num3, pp.336-340, Sep. 2013.

[2] Ting Bi, Gabriel-Miro Muntean. "Location-aware Network Selection Mechanism in Heterogeneous Wireless Networks." The IEEE INFOCOM International Workshop on Mobility Management in the Network of the Future World, pp.583-588, 2017.

[3] Van-Quyet Nguyen et al., "Location-aware Dynamic Network Provisioning." In Proceedings of APNOMS Conference, pp. 239-242, 2017.

[4] A. Juttner et al., "Lagrange relaxation based method for the QoS routing problem." In proceeding of IEEE INFOCOM, pp.859-868, 2001.

[5] "Open Networking Foundation." [Online]. Available: https://www.opennetworking.org