논문 19-44-10-23 The Journal of Korean Institute of Communications and Information Sciences '19-10 Vol.44 No.10
https://doi.org/10.7840/kics.2019.44.10.1956

네트워크 기능 가상화를 위한 VNF 배치 최적화 프레임워크

응웬트리하이◆, 유 명 식°

A VNF Placement Optimization Framework for Network Function Virtualization

Tri-Hai Nguyen◆, Myungsik Yoo°

요 약

네트워크 기능 가상화 (NFV)는 가상화 기술을 통해 네트워크 서비스의 동적 프로비저닝을 용이하게하는 패러다임이다. 네트워크 서비스는 가상화 된 범용 하드웨어, 즉 가상 네트워크 기능 (VNF) 위에 소프트웨어 구성 요소로 구현되는 일련의 기능을 연결함으로써 구현할 수 있다. VNF 배치는 사용 가능한 컴퓨팅 자원 및 네트워크 링크의 현재 특성에서 서비스 요청에 따라 VNF 체인에 대한 최적 위치 집합을 선택하는 문제이다. 기존 VNF 배치 문제를 다룬 연구들은 자체 모델 및 솔루션을 제공하지만, 일반화된 VNF 배치 최적화 시뮬레이터는 아직 다루어지지 않았다. 따라서 본 논문에서는 입력 및 출력을 일관되게 정의하는 VNF 배치 시뮬레이터를 제안한다. 본 시뮬레이터는 개발자와 연구원이 VNF 배치 알고리즘 개발에 효과적으로 활용할 수 있다.

Key Words : NFV, VNF, Latency, Optimization, Simulator

ABSTRACT

Network Function Virtualization (NFV) is a paradigm that facilitates dynamic provisioning of network services through virtualization technologies. In this vision, network services can be implemented by chaining a set of functions, implemented as software components on top of virtualized general-purpose hardware, i,e., Virtual Network Functions (VNFs). VNF placement is the problem of choosing the set of optimal locations for a chain of VNFs according to the service request at the current characteristics of available computing resources and network links. Existing works on VNF placement problem make their own models and solutions, but the standard VNF placement optimization simulator is still not addressed. Therefore, we propose a VNF placement simulator that consistently defines the inputs and outputs. The simulator helps the developer and researcher to focus more on the VNF placement algorithm development.

※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2018-2017-0-01633) supervised by the IITP(Institute for Information & communications Technology Promotion).

◆ First Author : Tri-Hai Nguyen, Department of ICMC Convergence Technology, Soongsil University, Seoul, Republic of Korea, nguyentrihai@soongsil.ac.kr, 학생회원

° Corresponding Author : Myungsik Yoo, School of Electronic Engineering, Soongsil University, Seoul, Republic of Korea, myoo@ssu.ac.kr, 정회원

논문번호 : 201907-138-D-RN, Received July 24, 2019; Revised August 8, 2019; Accepted August 8, 2019

Ⅰ. Introduction

Network functions virtualization (NFV)[1] is a paradigm that presents several advantages in comparison with traditional middle boxes, including the flexibility for service provisioning and the reduction of both operational (OPEX) and capital(CAPEX) expenditures. NFV uses virtualization techniques to provide network services through virtual devices running on generic hardware (eg, x86 architecture). Thus, it is much simpler to make new services available as well as to modify and update existing services. There is no need for a proprietary or specialized hardware equipment, resulting in cost reduction and flexible service provisioning.

In principle, all network functions and other network elements can be considered for virtualization. These virtualized instances are referred to as Virtual Network Functions (VNFs) in the context of NFV, which provide the same functionalities as the corresponding physical instances. Besides, VNFs can be instantiated, executed and deployed by service providers in the NFV Infrastructure (NFVI) environment which provides the required resources such as compute, storage, and network. To manage and orchestrate these VNFs in NFVI, there is an element called NFV Management and Orchestration (MANO). NFV MANO can be further divided into three entities, that is, Virtualized Infrastructure Manager (VIM), VNF Manager (VNFM) and NFV Orchestrator (NFVO). The NFV architecture is depicted in Figure 1.

Nevertheless, there are several challenges to the practical use of NFV technology, including VNF placement optimization problem [2][3]. For a set of requested network services or service function chains (SFC), the aim of VNF Placement is placing the VNFs on suitable servers in the network regarding the given objectives while satisfying specific constraints. However, the existing works focus on their proposed model and the source code of the proposed models cannot be shared. Therefore, we need an open-source standard or framework to evaluate the algorithms for VNF placement optimization problem.

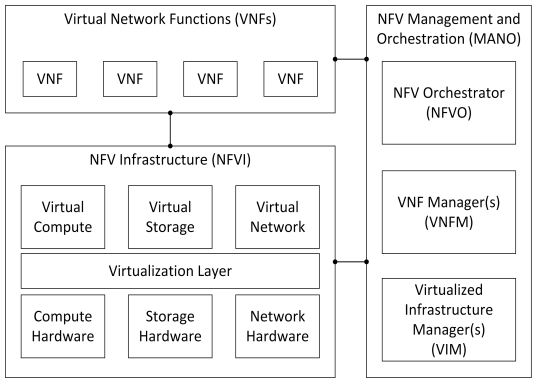

Fig. 1. NFV architecture

In this paper, we propose a VNF placement optimization simulator that places the SFC with the inputs of sources, networks and SFC template. The objective can be resource consumption minimization, data rate required minimization, latency minimization, or energy consumption minimization. However, for simplicity, the objective is to minimize the latency of the SFCs in the network in this paper. The algorithms used for placement process are greedy algorithm and random algorithm. Greedy algorithm places VNFs at the closest node (minimum latency) and connect with the shortest paths. Random algorithm places VNFs at random nodes and connect with shortest paths.

The remainder of this paper is organized as follows. Section II introduces the proposed VNF placement optimization simulation framework. The implementation of the VNF placement optimization simulator is shown in Section III. Finally, the conclusion of this paper is presented in Section IV.

Ⅱ. A VNF Placement Optimization Simulator

In recent years, the problem of VNF placement has been formalized and tackled by many authors, leading to a variety of different optimization problems and algorithms. So far, there is no generic VNF placement optimization simulator that abstracts the internals of these placement algorithms while

consistently defining their inputs and outputs. To overtake this issue, we propose a VNF placement optimization simulator that standardizes the inputs and outputs of the VNF placement optimization problem.

Figure 2 provides an overview of the workflow of the propose simulator. The inputs include the SFC requests, networks, algorithm and objective. In this paper, we choose the latency as the objective when placing VNFs into the network. The objective can be changed based on the need of service providers.

In term of algorithms, we adapt two common algorithms, i.e., greedy algorithm and random algorithm. The greedy algorithm places VNFs at the closest node (minimum latency) and connects with shortest paths. Random algorithm places VNFs at random nodes and connects with shortest paths.

Figure 3 shows the structure of the SFC placement input. Each request must specify a network, at least one service, and at least one source. The underlying network G = (V; E) consists of nodes (V) and edges (E). Nodes represent NFVI PoPs at different geographical locations, interconnected by edges. Additional, optional fields may specify the geographic locations or compute capacities of nodes and the data rates or delay of edges. For implementation, we suggest using the GraphML format, which is used by the popular datasets such as TopologyZoo [4] and SNDlib [5] that include realistic inter-PoP delays and bandwidth limits. Network services consist of VNFs that are interconnected by virtual links (vLinks). VNFs can be annotated with VNF type, resource requirements, etc. vLinks specify which VNFs are interconnected,

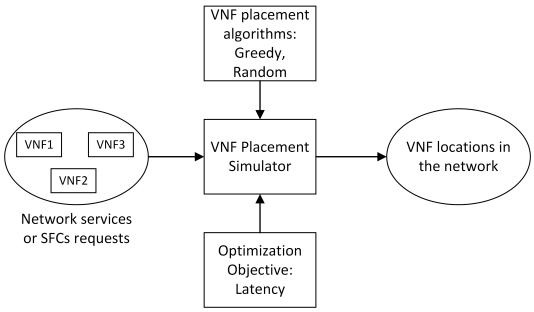

Fig. 2. VNF placement optimization simulator workflow

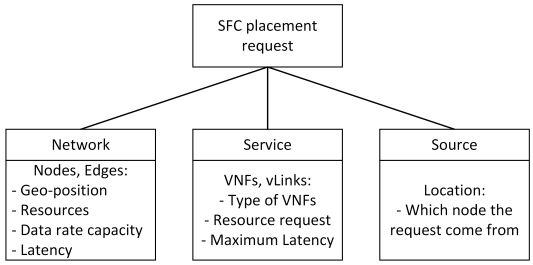

Fig. 3. Placement input structure

thus defining the VNF forwarding graph. Each vLink is unidirectional with a source and a destination. To specify bidirectional connections, two vLinks must be used. Sources, e.g., users or sensors, are located at different nodes in the network, request a service. Thus, the list of sources contains the location and the requested network service per source. Additionally, they can define traffic characteristics, e.g., the data rate.

After receiving a placement request, placement algorithms calculate a placement output. Figure 4 shows the structure of such a placement output, maps VNF instances to network nodes and specifies to which network service they belong to.

Since the number of VNF instances and their interconnections (vLinks) may be decided dynamically by placement algorithms, the placement response also needs to specify the source and destination of these vLinks. This mandatory information about VNF mapping and vLinks may be extended by further annotations. For example, vLinks may specify the calculated routes between interconnected VNFs. Placement responses may contain performance claiming about the calculated placement. For instance, the expected delay between connected VNFs and on the end-to-end service chain

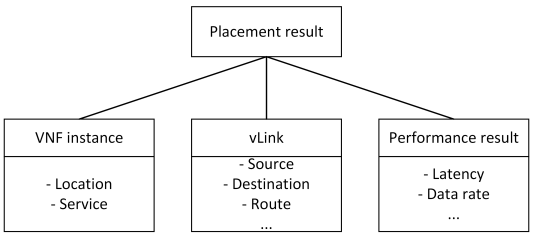

Fig. 4. Placement output structure

is specified.

Ⅲ. Experiment

The source code of the VNF placement optimization simulator is available online[6]. The VNF placement optimization simulator is written in Python, running on Linux (i.e., Ubuntu 16.04) environment. Some additional packages need to be installed such as networkx, geopy, numpy. etc.

For realistic network emulation, the simulator uses annotated geographical node positions from real-world network topologies[4,5] to calculate the distance between interconnected nodes and then derives the corresponding edge delays. The simulator can be run as follows.

First, the input of the simulator needs to be specified. There are some examples in the 'inputs' folder where the networks are the realistic network topologies, the services define SFC requests and are formatted as YAML file, the sources define the location of users in the network.

After specifying the inputs, we can start the placement process as following command:

'python3 placement_simulator.py -- algorithm ALG --network NETWORK --service SERVICE --sources SOURCES'

Where:

'--algorithm': the placement algorithm: 'random', and 'greedy'.

'--network': the network.

'--service': network service or service function chain (SFC) to be placed in the network.

'--sources': is the source of the user. It can be one or many sources in one file.

An example can be seen in Figure 5. The running command is as follows:

```
nguyenhai@NGUYENHAI-PC: ~/vnf-placement-simulator/results/greedy
 GNU nano 2.5.3                File: Abilene-sfc1-source-2019-04-02_17-33-59.yaml

input:
  algorithm: greedy
  network: inputs/networks/Abilene.graphml
  num_edges: 14
  num_nodes: 11
  num_sources: 1
  num_vnfs: 3
  service: inputs/services/sfc1.yaml
  sources: inputs/sources/source.0.yaml
metrics:
  delays:
  - delay: 1
    dest: vnf_fw1
    dest_node: pop2
    src: vnf_user
    src_node: pop0
  - delay: 2
    dest: vnf_web
    dest_node: pop9
    src: vnf_fw1
    src_node: pop2
  total_delay: 3
placement:
  vlinks:
  - dest_node: pop2
    dest_vnf: vnf_fw1
    path:
    - pop0
    - pop2
    src_node: pop0
    src_vnf: vnf_user
  - dest_node: pop9
    dest_vnf: vnf_web
    path:
    - pop2
    - pop9
    src_node: pop2
    src_vnf: vnf_fw1
  vnfs:
  - name: vnf_user
    node: pop0
  - name: vnf_fw1
    node: pop2
  - name: vnf_web
    node: pop9
time: 2019-04-02 17-33-59
```

Fig. 6. An example of results of VNF placement process

'python3 placement_simulator.py --algorithm greedy —network inputs/networks/Abilene.graphml --service inputs/services/sfc1.yaml —sources inputs/sources/source.0.yaml'. The output can be seen at 'results' folder. An example of the results is shown in Figure 6.

```
nguyenhai@NGUYENHAI-PC:~/vnf-placement-simulator$ python3 placement_simulator.py --algorithm greedy --network inputs/net
works/Abilene.graphml --service inputs/services/sfc1.yaml --sources inputs/sources/source.0.yaml

Starting greedy placement

Placed vnf_user at pop0
Closest available node from pop0: pop2 (shortest path length: 1)
Placed vnf_fw1 at pop2
Closest available node from pop2: pop9 (shortest path length: 2)
Placed vnf_web at pop9
Writing solution to results/greedy/Abilene-sfc1-source-2019-04-02_17-33-59.yaml
```

Fig. 5. An example of running VNF placement simulator

Ⅳ. Conclusion

In this paper, a VNF placement optimization simulator is proposed and implemented. It based on Python and run on the Ubuntu environment. The simulator provides the standard inputs and outputs based on YAML model. By using this VNF placement optimization simulator, the developers and researchers can focus on developing their own algorithms with different objectives.

References

[1] ETSI NFV, "*Network function virtualisation: An introduction, benefits, enablers, challenges & call for action*," Introductory White Paper, Issue 1, SDN & OpenFlow World Congr., Darmstadt, Germany, Oct. 2012.

[2] F. Bari, et al., "Orchestrating virtualized network functions," *IEEE Trans. Network and Service Management*, vol. 13, no. 4, pp. 725-739, 2016.

[3] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Network and Service Management*, vol. 13, no. 3, pp. 533-546, 2016.

[4] *The Internet Topology Zoo*. [Online] Available: http://www.topology-zoo.org/dataset.html

[5] *SNDlib*. [Online]. Available: http://sndlib.zib.de

[6] *VNF Placement Optimization Simulator*. [Online]. Available: https://anda.ssu.ac.kr/vnf-placement-simulator/

응웬트리하이 (Tri-Hai Nguyen)

Tri-Hai Nguyen received his B.S. degree in computer science from the University of Information Technology, Ho Chi Minh City, Vietnam, in 2015 and M.Eng. degree in information and communication technology from Soongsil University, Republic of Korea, in 2017. He is currently pursuing the Ph.D. degree in information and communication technology at Soongsil University, Seoul, Republic of Korea. His research focuses on network function virtualization and cloud computing.

[ORCID:0000-0002-2132-2290]

유 명 식 (Myungsik Yoo)

Myungsik Yoo received his B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, Republic of Korea, in 1989 and 1991, and his Ph.D. in electrical engineering from State University of New York at Buffalo, New York, USA in 2000. He was a senior research engineer at Nokia Research Center, Burlington, Massachusetts. He is currently a professor in the school of electronic engineering, Soongsil University, Seoul, Republic of Korea. His research interests include visible light communications, sensor networks, Internet protocols, control, and management issues.

[ORCID:0000-0002-5578-6931]