Implementation of NRF in the Docker-based NFV platform

Shasha Liu1,2, Zhan Xu1,*, Zhigang Tian2

1School of Information and Communication Engineering, Beijing Information Science & Technology University, Beijing, People’s Republic of China
2Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, People’s Republic of China

E-mail: xuzhan@bistu.edu.cn

Abstract: Network function virtualisation (NFV) represents one of the key enablers of the next generation mobile network systems (5G). NFV allows running virtual network functions (NFs) as software components on top of a virtualisation system (i.e. virtual machines or containers) hosted in a cloud, allowing high flexibility and elasticity to deploy network services and functions. Therefore, the NFs in 5G core network can be deployed on the common hardware platform in the form of software. In this study, network repository function (NRF) is implemented in Docker-based NFV platform in the form of JSON + HTTP/2.0 that supports service discovery function. The performance of NRF is also tested on real hardware platforms, including NF Register, NF Update, and NF Deregister. From the experimental data, it can be seen that the NRF in the NFV-based 5G architecture has good performance, and Docker-based NFV platform is more flexible than traditional communication networks.

1 Introduction

The communication industry requires networks with high reliability and high performance. Therefore, traditional communication networks usually use dedicated hardware that integrates hardware and software. However, with the development of mobile broadband, revenue from traditional voice services has been gradually reduced while traffic has increased exponentially. The construction methods for special hardware and special switches, and storage hardware, allowing administrators to replace new NFs for 5G system have been newly discussed and specified, servers. Using NFV can reduce or even remove middleware infrastructure.

In the field of the internet, the application of virtualisation and cloud computing technology has realised the generalisation of equipment, rapid deployment of services, sharing of resources, and flexible management. As a result, world-class telecommunication operators have jointly proposed the concept of NFV. NFV integrates hardware and software. However, with the development of mobile broadband, revenue from traditional voice services has been gradually reduced while traffic has increased exponentially. The construction methods for special hardware and special switches, and storage hardware, allowing administrators to replace new NFs for 5G system have been newly discussed and specified, servers. Using NFV can reduce or even remove middleware infrastructure.

2 Architecture

2.1 5G system architecture

From Fig. 1 we can see that the 5GC network infrastructure defined by 3GPP is a service-based architecture [1], which is based on cloud-native architecture design and draws on the concept of micro-services in the information technology (IT) field. The traditional network element is a tightly coupled black box design. The NFV decouples the NF software from the black box device and realises integration through an open application programming interface (API) interface to improve the overall agility and flexibility of application development. Microservices interact with each other through APIs, and each microservice is deployed, upgraded, and expanded independently of other services. It can frequently update the applications that are in use without affecting the use of the customer. In the 5GC, the NRF supports service discovery function. NRF receives NF discovery request from NF instance and provides the information of the discovered NF instances (be discovered) to the NF instance. NRF also maintains the NF profile of the available NF instances and their supported services. In this study, our main implementation on NRF service is Nnrf_NFManagement in the form of JSON + HTTP/2.0 that provides support for discovery of NF, NF services and its performance is tested on the hardware platform.

2.2 Network function virtualisation (NFV)

In the field of the internet, the application of virtualisation and cloud computing technology has realised the generalisation of equipment, rapid deployment of services, sharing of resources, and flexible management. As a result, world-class telecommunication operators have jointly proposed the concept of NFV. NFV integrates hardware and software. However, with the development of mobile broadband, revenue from traditional voice services has been gradually reduced while traffic has increased exponentially. The construction methods for special hardware and special switches, and storage hardware, allowing administrators to replace new NFs for 5G system have been newly discussed and specified, servers. Using NFV can reduce or even remove middleware infrastructure.
3 Implementation

3.1 Docker

As a lightweight virtualisation technology [7–9], container [10, 11] is not a modern concept. It runs on a single machine and shares the same operation system (OS) kernel so as to start instantly and make more efficient use of the key resources such as central processing unit (CPU) and memory [12]. Compared with a traditional virtual machine, it enhances efficiency significantly. Docker [13] is an LXC-based advanced container engine, i.e. open sourced by PaaS provider dotCloud, the source code is hosted on GitHub, based on the GO language and complies with the Apache2.0 protocol. Docker provides a simple way to package an application in a container. Due to its LXC-based lightweight virtualisation features, the most obvious feature of Docker compared with kernel-based virtual machine (KVM) is the fast startup and low resource consumption. One application of containers is building VNF. With NFV, traditional core network components are replaced by software implementations running on general-purpose hardware. In this study, we use Docker to virtualise underlying physical resources such as compute, memory, and network to implement the physical bearer for each element required by VNF. NRF shares physical hardware in the form of Docker, making most of the original physical components replaced by Docker.

3.2 Docker-based NFV platform

Based on the above design concept, the main part of the NFV architecture in this study is shown in Fig. 3. Docker virtualises the underlying physical resources such as compute, memory, and network to implement physical bearers for each element required by the VNF. The NRF is instantiated as VNF.

4 Performance

4.1 Testing cases

This study mainly implements the Nnrf_NFManagement service in SGC service framework in the Docker-based NFV architecture described above. This test covers basic service operation call procedures of Nnrf_NFManagement Service. Service management is for the NRF to properly maintain the information of available NF instances and the services it supports. Each NF instance informs the NRF of the list of NF services it supports. The following services and corresponding service operations are included in this test:

- **NF Register**: It allows an NF instance to register its NF profile in the NRF; it includes the registration of the general parameters of the NF instance, together with the list of services exposed by the NF instance.
- **NF update**: It allows an NF instance to replace or update partially, the parameters of its NF profile (including the parameters of the associated services) in the NRF it also allows to add or delete individual services offered by the NF instance.
- **NF Deregister**: It allows an NF instance to deregister its NF profile in the NRF, including the services offered by the NF instance.

4.2 Testing environment

The experimental system uses a general-purpose multi-core server. The configuration of the server is shown in Table 1. The Docker container is used to provide management of the host physical resources. The topology is shown in Fig. 4. The server and client are running in a separate container. The server contains a Hiredis database [14], i.e. a minimalistic C client library for the Redis database [15]. Hiredis supports for the whole command set, pipelining, event-driven programming.

4.3 Experimental results

First, the client and server establish a transmission control protocol (TCP) connection in our implementation of the NFV platform.
Then the client encapsulates the HTTP body and header, the HTTP body is a form of JSON encoded NFProfile. After that the client sends a request (NF Register, NF Update, or NF Deregister), the server decodes the data, it goes to the Hiredis database to perform the corresponding operation and then returns a response. Finally, the client and the server disconnect the TCP connection. Each operation is repeated 10,000 times. When the client sends an empty request, i.e. there is no NRF request, its cumulative distribution function (CDF) is shown in Fig. 5 and its average latency is shown in Table 2. When the client sends a NRF request, its CDF is shown in Fig. 6, its average latency is shown in Table 2.

From Fig. 5, we can see that the time of the whole process is mainly composed of the following parts. The first part is the process of TCP link establishment, including complete connection and parameter setting, 90.07% of the latency is <620 μs, and the average latency is 562.0821 μs. The second part is encapsulating HTTP header and body, 90.01% of the latency is <699 μs. The average latency of JSON encoding is 52.1795 μs. The third part is the request–response roundtrip, 90.07% of the latency is <365 μs, the average latency is 281.9191 μs. Since this is an empty request, this does not include the server-side application-layer processing latency. When a long TCP connection is established, it is no longer necessary to establish a link every time, so the entire latency will be relatively long only when the first connection is made.

As can be seen from Fig. 6a, when the client sends a NF Register request, the average TCP connection latency is 598.3567 μs, 90.01% of the latency is <699 μs. The average latency of JSON encoding is 52.1795 μs. Compared with the empty request, the HTTP body of NF Register contains a configuration file nFProfile that contains registered NF and NF service information, so the latency increases. The average latency of request–response roundtrip is 661.7922 μs, which increases the processing latency of the application layer compared with the empty request, including the reading and writing of the database. NF Update is divided into the complete update and partial update depending on the content of the update. Complete update implementation of the complete replacement of nFProfile, the same nature as NF Register, as shown in Fig. 6b, the latency of each part is similar to NF Register. Partial update performs partial replacement of nFProfile, i.e. only properties that need updating in nFProfile. Therefore, the JSON encoding latency is reduced compared with the complete update, but the latency of the request–response roundtrip increased due to the need to compare and then modify the data in the database, as shown in Fig. 6c. Since the HTTP of NF Deregister does not include nFProfile, the JSON encoding latency is almost the same as the empty request, and the request–response roundtrip latency is also reduced compared with the NF Register and the NF Update, as shown in Fig. 6d.
The results of the above tests can prove that the NRF implemented on Docker-based NFV platform has high performance.

5 Conclusion

In this study, we used Docker for infrastructure virtualisation to implement an NFV platform and deployed NRF services in the form of JSON + HTTP/2.0, mainly including NF Register, NF Update, and NF Deregister. Experimental data shows that our idea of deploying NRF on Docker-based NFV platform is completely feasible and has good performance. Compared with traditional communication networks, our architecture is more flexible, so it has lower upgrade complexity and shorter service innovation cycles.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (no. 61620106001), the Beijing Excellent Talent Support Program (no. 20160000268332ZK08), the Beijing Nova Program (no. Z161100004916086), the Construction PLAN of High Level Teachers in Beijing Municipal Universities (no. CIT&TD201704065), and the Discipline construction project of Beijing Information Science and Technology University (no. 5121911006).

7 References

[1] 3GPP TS 23.501: ‘System architecture for the 5G system; stage 2 (release 15)’, 2017
[2] ETSI GS NFV002: ‘Network functions virtualization (NFV): architectural framework’, 2013
[3] Taleb, T., Ksentini, A., Jantiti, R.: ‘Anything as service for 5G mobile systems’ (IEEE Press, Piscataway, NJ, USA, 2016)
[4] 3GPP TR 28.801V15.1.0: ‘Study on management and orchestration of network slicing for next generation network (release 15)’, January 2018
[5] ETSI GS MAN001: ‘Management and orchestration (MANO)’, 2014
[6] Somani, G., Khandelwal, P., Phutani, K.: ‘VUPIC virtual machine usage based placement in IaaS cloud’, Proc. Social Behav. Sci., 2012, 52, pp. 25–34
[7] Armbrust, M., Fox, A., Lee, G., et al.: ‘A view of cloud computing’, Commun. ACM, 2010, 53, (4), pp. 50–58
[8] Sahoo, J., Mohapatra, S., Lath, R.: ‘Virtualization: a survey on concepts taxonomy and associated security issues’. IEEE Second Int. Conf. on Computer and Network Technology (ICCNCT), Bangkok, Thailand, 2010, pp. 222–226
[9] Vaughan-Nichols, S.J.: ‘New approach to virtualization is a lightweight’, Computer, 2006, 39, (11), pp. 12–14
[10] Soltesz, S., Pötzl, H., Finciynski, M.E., et al.: ‘Container-based operating system virtualization: a scalable high performance alternative to hypervisors’. ACM SIGOPS Oper. Syst. Rev., 2007, 41, (3), pp. 275–287
[11] Hogg, S.: ‘Software containers: used more frequently than most realise’, 2014
[12] Rabani, M. G., Pereira Esteves, R., Poulsen, M., et al.: ‘On tackling virtual data center embedding problem’. IFIP/IEEE Int. Symp. on Integrated Network Management, Ghent, Belgium, 2013, pp. 177–184
[13] ‘Docker’. Available at https://docs.docker.com/, accessed 1 April 2018
[14] ‘Hiredis’. Available at https://github.com/redis/hiredis, accessed 13 April 2018
[15] Carlson, I.L.: ‘Redis in action’ (Manning Publications Co., Shelter Island, NY, USA, 2013)

Fig. 6 CDF of NF Request
(a) NF Register, (b) NF complete update, (c) NF partial update, (d) NF Deregister