A Study on Automotive Management and Orchestration System for Video Streaming over the Internet

[1]Linh Van Ma, [2]YoungYun Lee, [3]Sung-Hoon Hong,
[4]Baeho Lee, [5]Sung-June Baek, [6]Jinsul Kim
School of Electronics and Computer Engineering
Chonnam National University, Gwangju, Korea
[1]linh.mavan@gmail.com, [2]clonelee@chonnam.ac.kr,
[3]hsh@jnu.ac.kr, [4]bhlee@jnu.ac.kr,
[5]tozero@jnu.ac.kr,[6]jsworld@chonnam.ac.kr

***Abstract*—Network functions virtualization offers a new way to design, deploy and manage networking services. Besides, the network virtualization management and orchestration provides an ability to spin up network components with short time constraints. The management allows rapid onboarding and prevents system chaos. In this paper, we present a study on the management system to control adaptive streaming servers. We use the management system to start and stop the servers based on current system load to optimize resources. We propose a system using several performance metrics to monitor resources. We implement the system using python, the adaptive streaming servers using Node.js. Besides, we run all of the system components in Docker to virtualize a cloud streaming system. Experiment result indicated that the system reduce resource consumption up to 10%.**

Keywords—MANO; adaptive streaming; NFV; node.js; video streaming

I. Introduction

The network today works on heterogeneous devices, such as mobiles, computers, routers, switches. Cooperating these devices sometimes causes vulnerable functions, or system networks work inefficiently. The rise of significant competition from major Internet technology companies, such as Netflix, Microsoft, Skype, and Google has proposed a new way of managing network resources. Network function virtualization (NFV) [1, 2] describes how to virtualize and manage network resources, such as storage, and computing in a network. The virtual network functions manager (VNFM) [3] is a fundamental component of the NFV management and organization (MANO) architectural framework. The VNFM works in concert with other NFV-MANO [3] functional blocks, such as NFV orchestrator (NFVO) and the virtualized infrastructure manager (VIM), to increase the interoperability of software-defined networking elements and help standardize the functions of virtual networking. These standard components can increase new feature deployment and velocity and help lower costs by providing a standard framework for building NFV applications.

VNFs is an important essential to carry out the business benefits outlined by NFV architectural. They provide real network functions as real network components do but it requires VNFM to manage virtualized components. VNFMs are very necessary for the expansion, operational changes, add new resources, and inform other VNFs function blocks in the architecture NFV-MANO. During the life cycle of a VNF, the management functions of VNF can track key performance indicator (KPIs) of a VNF. KPI represents the efficiency of a VNF throughout its performance. It is a measurable value and is a key to achieve resource performance objective.

In this paper, we exploit the MANO structure to build a video streaming system. The MANO manages streaming servers. It can start/stop a server depending on the current load of the system. In this way, we can manage network resource efficiently. We also propose several metrics to monitor resources. We use these metrics to find the best severing streaming server according to a client's request.

We arrange the order of articles as follows. In section 2, we present an overview of Docker, Dynamic Adaptive Streaming over HTTP (DASH) in cloud computing and transcoding multimedia in the cloud. We present related works in Section 3. In Section 4, we describe the system in details including how we can manage streaming server based on proposed metrics. In section 5, we present experiment and discuss the results. Section 6 presents the findings with future research directions.

II. Background Technology

A. Docker

Docker [4, 5] is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers [6] allow developers to wrap up an application with all the necessary components, such as libraries and other dependencies, and deliver them all out as a package. Thanks to the container, the developers can rest and assure that the application will run on any other Linux machines regardless of the computer may have some customized settings that differ from the computer used for testing and writing the code. Containers use shared operating systems. They are much more efficient than hypervisors in terms of system resources. The containers reside on the same Linux instance instead of virtualizing the hardware. A comparison between virtual machines (VMs) and Docker.

B. Adaptive Streaming

Adaptive bitrate streaming [7, 8, 10] is a technique used in streaming multimedia over computer networks. It works by detecting a user's bandwidth and CPU capacity in real time

and adjusting the quality of a video stream accordingly. It requires the use of an encoder which can encode a single source video at multiple bit rates. The player client switches between streaming the different encodings depending on available resources. The result: very little buffering, fast start time and a good experience for both high-end and low-end connections. Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH, is an adaptive bitrate streaming technique that enables high-quality streaming of media content over the Internet delivered from conventional HTTP web servers. MPEG-DASH works by breaking the content into a sequence of small HTTP-based file segments, each segment containing a short interval of playback time of content that is potentially many hours in duration, such as a movie or the live broadcast of a sports event. Video Streaming is provided at a variety of bit rates. The alternated segments are encoded at different bit rates it is then divided into small pieces of segments with short play back time. The content is played at an adaptive streaming client. It has algorithms which automatically chooses the next optimal segment to download and play back based on current network conditions. The clients choose segments with the highest possible bit rate that can be downloaded on time for playback without causing stalls or buffering events in the playback. Thus, a MPEG-DASH client can smoothly adapt to changing network conditions, it also provides high-quality playback with fewer stalls or buffering events.

III. Related Work

The growing demand for online distribution of high quality and high throughput content is dominating today's Internet infrastructure. This domination leads to quality of experience (QoE) fluctuations on delivered content, and unfairness between end users, while new network protocols, technologies, and architectures, such as software defined networking (SDN), are being developed for the future Internet. The programmability, flexibility, and openness of these emerging developments can significantly assist the distribution of video over the Internet. The authors [10] introduced a novel user-level fairness model UFair and its hierarchical variant UFairHA, which orchestrated HAS media streams using emerging network architectures and incorporate three fairness metrics (video quality, switching impact, and cost efficiency) to achieve user-level fairness in video distribution. Their experimental results demonstrated the performance and feasibility of their design for video distribution over future networks.

In the research [11], the authors implemented and evaluated a novel architecture that leveraged the benefits of SDN to provide network assisted Adaptive Bitrate Streaming. With clients retaining full control of their streaming algorithms they showed that by this network assistance, both the clients and the content providers benefited significantly regarding QoE and content origin offloading. They also illustrated the difficulty of the problem and the impact of SDN-assisted streaming on QoE metrics using various well-established player algorithms. Their results showed the substantial improvement in cache bitrates indicating a rich design space for jointly optimized SDN-assisted caching architectures for video streaming applications.

IV. System Overview

In the streaming system, we have three main components. First, a DASH management server is responsible for listening requests from clients and directing those requests to content servers based on selecting algorithms. Secondly, an uploading and transcoding server makes uploaded files from clients to DASH content. Finally, DASH (content delivery network) CDN servers respond requests coming from clients and provide video streaming to clients. Fig.1 illustrates the system overview.

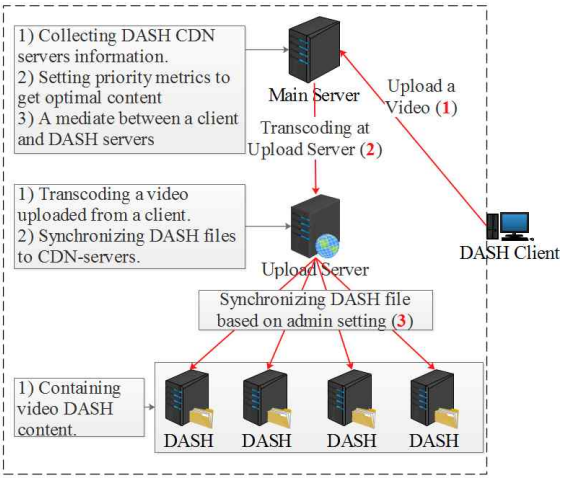

Fig.1. CDN-DASH streaming server system

The main server collects measurement information of CDN DASH servers to redirect a request from a client to the best serving server at the requesting time. We use seven parameters to evaluate the performance of a DASH streaming server. Regarding local measurement, we have; 1) Working CPU, 2) Free memory, 3) Total memory, 4) Load average. Regarding network measurement, we have; 5) Ping, 6) Throughput upload, 7) Throughput download. The management server also shares information about the order of the servers for all servers through a synchronization mechanism using a file-sharing protocol such as File Transfer Protocol (FTP). DASH servers use this information for the purpose of retrieving DASH files from other servers in case it does not serve a transcoding request from a client. This technique is a solution for sharing DASH files of a server when it contains a multimedia video while other servers do not have the video content. A server can retrieve DASH files from other servers based on metric information from the available parameters. For example, Server A and Server B has the same DASH video, and Server A is faster than B 25%. If Server C does have the video content, it can get 75% video DASH files from A and 25% DASH files from B. Fig.2 describes data flow structure between the managed server and DASH servers. Suppose that we have n servers, eight measurement metrics m_i have difference weights $w_i, i \in \{1,2,\dots,8\}$. The main server orders DASH servers based on the total metric in (1) where m_j is the j^{th} metric of the server i^{th}.

In the system, we have a VNFM managing other network components, such as Element Management (EM), Element Management Agent (EMA). As shown in Fig. 3, EMA acts as a router to mediate between VNFM and EM. Each EM manages a particular network component, such as Management Server, Data Streaming Server.

$$M_i = \sum_{j=1}^{8} w_j \frac{m_j}{\max_{i \in [1,\ldots,n]} m_{ji}} \qquad (1)$$

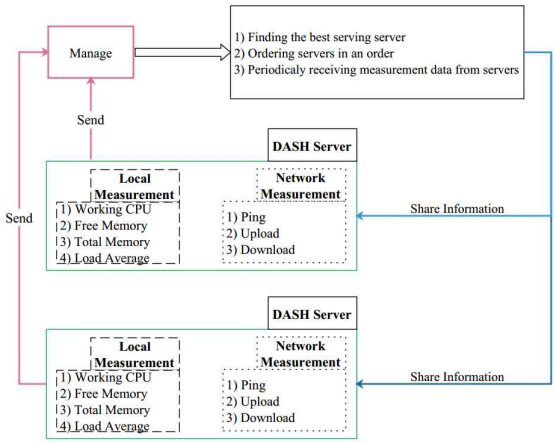

Fig.2. Main server manages parameter from other servers

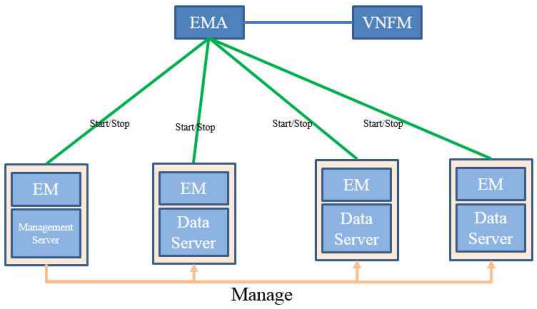

Fig.3. VNF DASH streaming system overview

V. System Overview

We implemented the system in Ubuntu 14.04. Server applications are Node.js-based application. It runs on containerized technology Docker version 1.12. Fig.4 shows the structure of the streaming servers. In the experiment, we use a Docker-based web interface application to test the streaming system. VNFM manages server applications by Docker start/stop command, Element Management Agent (EMA) receives these command and start/stop application correspondingly.

We arrange three physical upload servers with mediocre performance, two physical CDN DASH servers to provide video streaming. Each server can start several Node.js applications. The main server and the client web browser running on a same physical computer but executed on Docker to ensure that it is separate virtualize machines. The client requests to upload video files to the upload servers. These files are transcoded to DASH format. This job requires extensive computation resource. Therefore, the VNFM will start new upload servers if it recognizes that current system cannot respond to client requests. The VNFM can also start new CDN DASH servers if current DASH servers overload upon current requests from clients. In this case, the new starting DASH servers synchronize DASH files to its local disk based on selected metric such as these server sync video files with the most viewed.

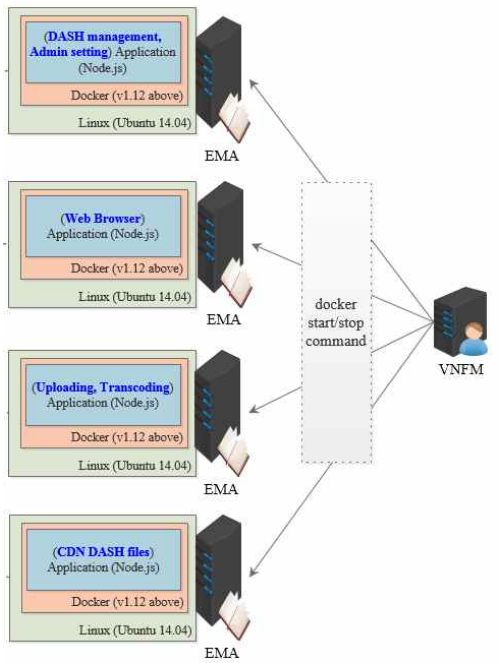

Fig.4. CDN-DASH Streaming Server System Implementation

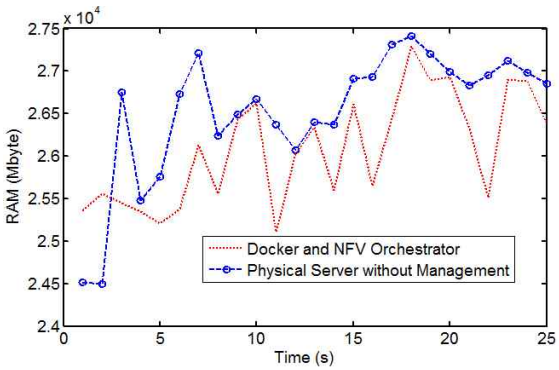

Fig.5. RAM consumption comparison of the system with Docker and NFV orchestrator.

We calculated resource consumption by summarizing the total CPU performance and RAM usage in a period of all physical computers. We set up two scenarios to compare the system performance between the system having Docker, NFV orchestrator and the system having applications which only run the physical servers.

In a client, we upload several files with different resolutions. The resolution is up to 4K (2160p). We then measure the system performance around 30 seconds in the

both scenarios. Fig.5 shows the RAM consume comparison between the system with Docker and NFV Orchestrator and the physical server without any management system. The blue line indicates the RAM usage more than the red line because the system managed resource efficiently.

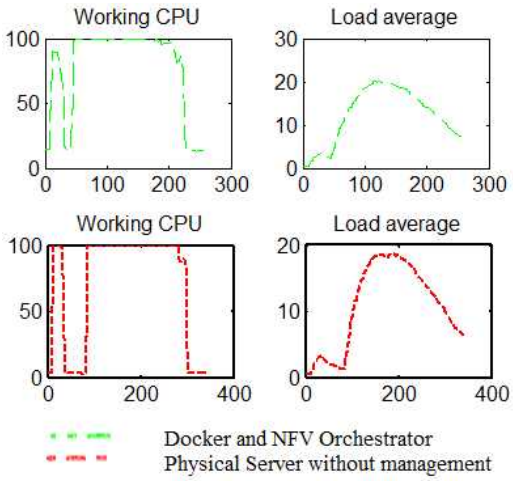

Fig.6. Sever performance comparison between the system with orchestration and without orchestration.

Moreover, Fig.6 shows the server performance comparison between the system with orchestration and without orchestration. As a result, the system with the orchestration reduced transcoding time as well as server performance when transcoding a multimedia file to DASH content.

VI. Conclusion

After In this paper, we discussed the current issue emerged in today virtual network technologies. When the number of network devices increases in a network continuously, we hardly manage them efficiently, and it usually causes latency in the service response. Our study contributed two main points as the following:

1. Discussion of network virtualization technologies such as NFV management and organization and Docker container, a containerized technology.

2. On the basis of our proposed metrics to manage DASH servers, we implemented DASH streaming system using MANO architecture. The result showed that the system reduced resource consumption up to 10%.

In the future research, we tend to reduce space for DASH content in the CDN.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology [Grant No. NRF-2017R1D1A1B03034429], and also was supported by the MSIT (Ministry of Science and ICT), 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). Furthermore, this research was supported by the IT R&D program of the MSIT (Ministry of Science and ICT), Korea/NIPA (National IT Industry Promotion Agency) 12221-14-1001, Next Generation Network Computing Platform Testbed.

References

[1] Mijumbi, Rashid, Joan Serrat, Juan-luis Gorricho, Steven Latre, Marinos Charalambides, and Diego Lopez. "Management and orchestration challenges in network functions virtualization." IEEE Communications Magazine 54, no. 1 (2016): 98-105.

[2] Han, Bo, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. "Network function virtualization: Challenges and opportunities for innovations." IEEE Communications Magazine 53, no. 2 (2015): 90-97.

[3] Giotis, Kostas, Yiannos Kryftis, and Vasilis Maglaris. "Policy-based orchestration of NFV services in Software-Defined Networks." In Network Softwarization (NetSoft), 2015 1st IEEE Conference on, pp. 1-5. IEEE, 2015.

[4] Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." Linux Journal 2014, no. 239 (2014): 2.

[5] Boettiger, Carl. "An introduction to Docker for reproducible research." ACM SIGOPS Operating Systems Review 49, no. 1 (2015): 71-79.

[6] Felter, Wes, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. "An updated performance comparison of virtual machines and linux containers." In Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On, pp. 171-172. IEEE, 2015.

[7] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)." In Packet Video Workshop (PV), 2013 20th International, pp. 1-8. IEEE, 2013.

[8] Seufert, Michael, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hobfeld, and Phuoc Tran-Gia. "A survey on quality of experience of HTTP adaptive streaming." IEEE Communications Surveys & Tutorials 17, no. 1 (2015): 469-492.

[9] Georgopoulos, Panagiotis, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming." In Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, pp. 15-20. ACM, 2013.

[10] Mu, Mu, Matthew Broadbent, Arsham Farshad, Nicholas Hart, David Hutchison, Qiang Ni, and Nicholas Race. "A scalable user fairness model for adaptive video streaming over SDN-assisted future networks." IEEE Journal on Selected Areas in Communications 34, no. 8 (2016): 2168-2184.

[11] Bhat, Divyashri, Amr Rizk, Michael Zink, and Ralf Steinmetz. "Network Assisted Content Distribution for Adaptive Bitrate Video Streaming." In Proceedings of the 8th ACM on Multimedia Systems Conference, pp. 62-75. ACM, 2017.