Effects of Service Function Relocation on Application-level Delay in Multi-access Edge Computing

Junichi Kaneda, Shin'ichi Arakawa, and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University,
1–5 Yamadaoka, Suita, Osaka, 565–0871 Japan
Email: {j-kaneda, arakawa, murata}@ist.osaka-u.ac.jp

***Abstract*—Multi-access edge computing (MEC) is expected to mitigate delays and more flexibly provide services by virtualizing service functions and deploying them closer to users. However, live migration of virtual machine (VM) that enables relocation of service functions for flexible service provision may cause temporary delays or packet loss. For future deployment of MEC, it is therefore important to investigate whether responsiveness improves as expected, and to evaluate the effects of service function relocation on application-level delay experienced by users. In this paper, we investigate application-level delay in a MEC environment for services such as video live streaming. Experiments in the MEC environment constructed within our laboratory reveal that application-level delays are reduced by 15–30%, and that application-level delay is improved by relocating remote service functions at an edge close to the user. We also reveal that delays and packet loss due to VM live migration are very temporary, confirming that service function relocation is useful for maintaining application-level delay.**

***Index Terms*—Internet of Things (IoT), Multi-access Edge Computing (MEC), OpenStack, Live Migration, Video Live Streaming, Application-level Delay**

I. Introduction

IoT (Internet of Things) is expected to bring new service applications, such as health monitoring [1] and smart building monitoring [2], to our life. In the current IoT environments, devices equipped with many sensors and cameras gather the information on surroundings, and the information is first transferred to the data center and processed there [3]. Then, the results are returned to devices and/or users as necessary [4]. Processing at the data center may be acceptable as long as the size of data and the number of interconnected devices are sufficiently small. However, the growth of IoT-related markets will lead to penalties in the form of application-level delay (i.e., delay experienced by devices/users) due to geographical factors and load concentration [5]. The penalty is even more expensive when high-bandwidth service applications, e.g., but not limited to, health/building monitoring through low latency real-time video analytics are deployed in accordance with 5G wireless systems.

The concept of multi-access edge computing (MEC), has been introduced to mitigate delays [5]–[7]. MEC virtualizes service functions and deploys them on edge servers. An edge server is a secondary data center located at the network edge, closer to the user. Incorporating network function virtualization (NFV) to edge servers is expected to allow flexible changes in resources and deployment locations for virtual machines (VM) on which the function operates [6]–[8]. Service applications use functions at edge servers rather than at the data center. It is expected that response to the service will be improved by eliminating geographical delays and relaxing load concentrations. One application for MEC is augmented reality (AR) content delivery services [6]. AR adds information to real-world images and displays them to users in real time. Low latency is thus required for content distribution and processing for adding contents. Caching AR content that is frequently used at edge servers for delivery to mobile devices is expected to reduce round-trip times (RTT) and ensure high bandwidth in 5G wireless systems.

However, since the processing capability in edge is lower than that in the data center, there are concerns that processing delays at edge servers may increase due to software operation in a virtualized environment. Furthermore, VM live migration that enables relocation of service functions for flexible service provision may cause temporary delays or packet loss because of a reestablishment of connections during the live migration. For deployment of MEC for future IoT applications, it is therefore important to investigate whether responsiveness improves as expected, and to evaluate the effects of service function relocation on application-level quality, such as application-level delay.

In this paper, we investigate application-level delay experienced by users occurring at nodes in a MEC environment. To that end, we construct a MEC environment using server machines and OpenStack [9]. We also build a service in which users and remote robots cooperate via video live streaming, assuming a robotic monitoring agent works with sensor devices. By manually and dynamically changing service function locations, we clarify the effect on application-level quality. Finally, we investigate the penalty of VM live migration from the aspect of packet loss, and reveal the extent to which the impact of background traffic is relaxed by relocation of service functions.

The remainder of this paper is organized as follows. We review related work in Section II. Section III describes implementation of the MEC service and the MEC environment. In Section IV, we measure and evaluate application-level delay

occurring at nodes in the MEC environment. In Section V, we examine and evaluate application-level quality due to high network load at edge servers and service function relocations. In Section VI, we present our conclusions and future work.

II. Related Work

Network measurement has intensively investigated the content of quality-of-service (QoS). Network measurement researchers have intensively investigate the relationship between QoS metrics and application-level quality on the Internet [10], [11]. Even under low-level jitter and packet loss (8%), quality opinion scores are halved and users' perceived quality drops as compared with perfect conditions [10]. A 20% packet loss caused errors in about 90% of MPEG frames [11].

There are few studies of application-level delay measurement in MEC environments. End-to-end latency of 5G MEC has been investigated, but only performs network-level RTT measurement [12]. A study that examines relation between differences in live migration latency for three container storage types; local storage, shared sync storage and shared async storage, shows that the latency in shared async storage condition is smallest and it takes about 10 s in a MEC environment [13]. There are several studies that investigate the application-level downtime caused by live migrations (see Ref. [14] and references therein). It is concluded that the application-level downtime caused by live migration with 200 MB memories in a metropolitan area network (MAN) or wide area network (WAN) is 0.8–1.6 s [15]. A live migration approach that minimizes the network-level downtime is presented [16]. However, it remains unclear how application-level downtime affects application-level quality, such as video quality or delays in MEC environment.

In this paper, we directly measure application-level quality and delay in a MEC environment. We also examine the improvement of user's experienced delay by relocation of service functions under heavy background traffic.

Recently, quality of experience (QoE) has been used to evaluate users' perceived quality of services [17]. Unlike QoS metrics, which are measured on the network side, QoE is based on user experience, perception, and expectations regarding application and network performance. The work in our paper does not focus on QoE, but the application-level quality measured in our experiment will contribute to understanding QoE in MEC environments, because QoE metrics include application-level quality [17].

III. Implementation of the MEC Service and the MEC Environment

In this section, we explain the construction of a MEC environment using OpenStack and a service application operating on the environment.

A. MEC Service

As a potential new service, we consider realization of a monitoring agent service using robots. In this service, robots go to a physical place, and users can monitor from home as if they were actually there. Using AR technology, object information is superimposed on video acquired by a camera mounted on the robot and presented to the user. This process is performed on an external server, and object information is acquired from the cloud. Sensing technology can also used to present tactile sensations of objects and to control the robot.

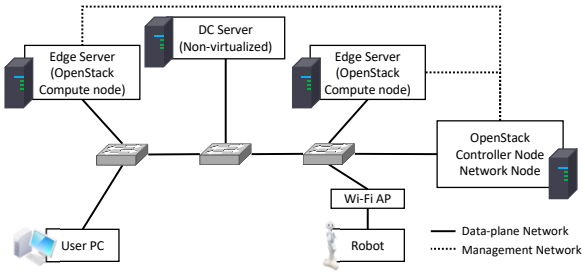


Fig. 1: Configuration of the MEC environment

In this paper, we assume only live streaming of video from the "Pepper" robot [18], which is equipped with a camera that compresses video into MPEG2 format using FFmpeg [19], which is part of its operating system. Video is then transferred to an edge server and text information is added using FFmpeg installed on a VM. Another VM relays the video stream to a VM hosting FFserver, a streaming server application, to stream it to a PC for real-time user viewing using FFplay. To simplify the implementation, we do not insert the superimposition of product information, but insert a simple text. The text is manually inserted using drawtext filter of FFmpeg. Note that FFserver uses the UDP and TCP transport protocols for reception and transmission, because of its specification.

Pepper is a humanoid robot. Its software development kit (SDK) and application programming interface (API) are publicly available, so internal and external applications using its equipped cameras, sensors, and motion modules can be developed easily.

B. Construction of an MEC Environment using OpenStack

To build a MEC environment, we use OpenStack (Ocata), which is open-source software for creating virtualization environments. The white paper of MEC framework [8] states that MEC incorporates NFV for virtualization. Taking advantage of virtualization allows relocating service functions based on load at the edge server and on user movement. Since OpenStack has already been adopted in many implementation projects of NFV [20], [21], it is considered appropriate to use OpenStack to build virtualization environment for MEC.

We built a MEC environment in the laboratory by connecting four similarly configured servers, the user PC and the robot with switches, as shown in Fig. 1. Three of the four machines are OpenStack nodes, one operating as the OpenStack controller and network node and the other two as OpenStack compute nodes operating as edge servers. In the monitoring agent service, since the robot and the user are geographically separated, the two edge servers are prepared as

processing bases close to each other. CentOS 7 and a kernel-based virtual machine (KVM) are installed on the compute nodes as the host OS and hypervisor, respectively. Applications are executed in VMs with 4 GB of memory and 32 GB of storage on the compute nodes. Storage files of all VMs are shared by network file system (NFS) among three OpenStack nodes. The fourth server is not virtualized for comparison. We also assume that this non-virtualized server is a data center at the center of the network. Hereafter, we refer to the non-virtualized server as the DC server.

The robot, the user PC, and VMs on edge servers and the DC server communicate using a data-plane network. The robot is connected to the data-plane network via a Wi-Fi access point. On the two OpenStack compute nodes, virtual routers and virtual switches allow VMs to connect to the data-plane network. We use Open vSwitch for the switching function. In addition to basic Layer 2 and 3 functions, distributed virtual router (DVR) is enabled for the data-plane network. The DVR allows distributing virtual routers to all compute nodes, while only a single virtual router is deployed on the network nodes by default. OpenStack controller nodes and network nodes use the out-of-band network to communicate with compute nodes; they are not at all involved in data-plane communication. Hereafter, we call this out-of-band network the management network. Note that the MEC environment is constructed in a LAN. Therefore, end devices (the user PC and the robot) and the DC server are not geographically separated in our configuration. When evaluating the experimental results, it is thus necessary to consider delays caused by geographical factors.

IV. Application-level Delay Measurement

To evaluate the effect of service function relocation on application-level quality in a MEC environment, we focus on and measure application-level delay between end devices for video live streaming. The processing time of a edge server and a DC server are also measured.

A. Scenario

Four scenarios are used to investigate the application-level delay due to differences in the locations where service functions are deployed. For each scenario, we changed forms of service provision, such as the applications to operate, the existence of a virtualization environment, distance of TCP communication, and the location of service functions. In scenarios with names beginning with "edge," the service function is deployed on an edge server. The "Data-Center" and "Direct" scenarios are used for comparison. Figure 2 shows the locations of service functions and communication paths in each scenario. Note that TCP path lengths represent the number of links on the TCP path. Our preliminary experiments, which are not shown in the current paper, to measure the communication delay of UDP and TCP showed that the delay of TCP is about twice the delay of UDP even in our LAN network. Therefore, we focus on the difference of TCP path length.

- **Edge-User-Side:** In this scenario, applications that perform text insertion, relay, and streaming are deployed on a user-side edge server and executed on VMs. Placing service functions on the server reduces TCP communication distances. The TCP path length is 2.
- **Edge-Robot-Side:** In this scenario, applications that perform text insertion, relay, and streaming are deployed on a robot-side edge server and executed on VMs. Placing service functions on the server increases TCP path lengths. The TCP path length is 4.
- **Data-Center:** In this scenario, applications that perform text insertion, relay, and streaming are deployed on the server serving as the data center and executed directly on that server. The TCP communication distance is larger than in the Edge-User-Side scenario and smaller than in the Edge-Robot-Side scenario. The TCP path length is 3.
- **Direct:** In this scenario, no applications that perform text insertion, relay, or streaming are deployed. Video is directly sent from FFmpeg on the robot to FFplay on the user PC. We aim to measure the processing time at the end device. All communications use UDP.

B. Result

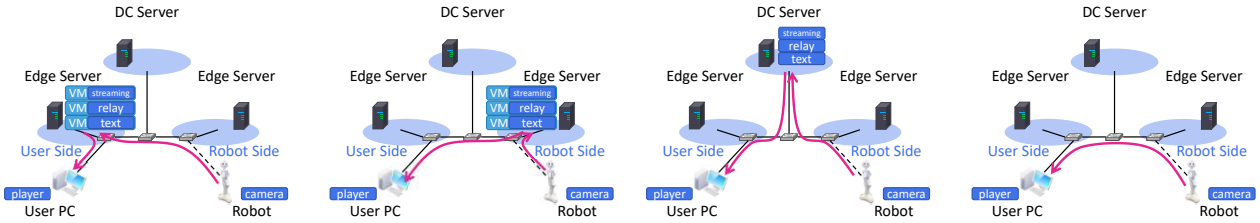
To measure the application-level delay of video live streaming, we display a millisecond-precision digital clock in front of the robot as shown in Fig. 3. Video captured by the robot is live-streamed and displayed on the user PC. The digital clock shown to the robot is also displayed on the user PC for time synchronization. Next, this digital clock and live-streaming video from the robot are arranged on the user PC display, and a screenshot is taken per second for 100 seconds. We calculate the difference in capture times between each screenshot, and calculate the average delay time to measure application-level delay of video live streaming in the four scenarios. Table I shows the application-level delays and their the factors for video live streaming in the four scenarios. The delays are the averages of two measurements.

We also measure server processing times in the Edge-User-Side, Edge-Robot-Side, and Data-Center scenarios. Since it is difficult to directly measure server processing times, we regard differences between incoming and outgoing time packets as the server processing time. We therefore repeat the start and stop of live streaming while capturing packets at the server's network interface. Using those captured packets, we can calculate differences in capture times between first incoming and first outgoing packets. We also calculate differences in capture times between last incoming and last outgoing packets. Table II shows average server processing times of 10 measurements.

C. Evaluation

The results show the following (Table III):

- The processing time at the end device is 425.19 ms.
- The time required for text insertion and streaming server processing is 28.85 ms. During that time, the server



(a) Edge-User-Side (b) Edge-Robot-Side (c) Data-Center (d) Direct

Fig. 2: Location of service functions and communication paths in each scenario

TABLE I: The application-level delays and their factors for video live streaming in the four scenarios

Scenario	Factors	Delay [ms]
Edge-User-Side	End device, Text & Streaming, Virtualization, 2 TCP path length units	479.48
Edge-Robot-Side	End device, Text & Streaming, Virtualization, 4 TCP path length units	491.88
Data-Center	End device, Text & Streaming, 3 TCP path length units	472.64
Direct	End device	425.19

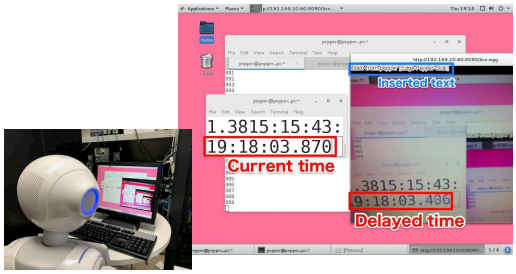


Fig. 3: The way of the application-level delay measurement

TABLE II: Server processing time

Scenario	Server name	Time [ms]
Edge-User-Side	User-side edge server	11.35
Edge-Robot-Side	Robot-side edge server	11.90
Data-Center	DC Server	7.60

processing time is 7.60 ms, and the remaining 21.25 ms is protocol overhead.

- The delay increase due to virtualization is 13.04 ms. During that time, the increase of server processing time is 4.00 ms, and the remaining 9.04 ms is increased protocol overhead.
- The delay increase per TCP path length unit is 6.20 ms.

The RTT caused by distance from the end device to the data center is about 100 ms or less to a domestic data center in Japan, about 100 ms to a data center in the U.S., and about 200 ms to a data center in Europe [22]. Therefore, delay in the Data-Center scenario of 472.64 ms is actually considered to be about 100–200 ms larger. Comparing delay due to geographical factors, the increased delay, 6.84 ms, due to virtualization is sufficiently small. That is, providing services using edge servers will reduce application-level delays between end devices by 15–30%. Application-level quality can thus be improved by relocating service functions from remote data centers to an edge server near the user.

TABLE III: The delay of each factor

Factor	Delay [ms]	Server processing time [ms]
End device	425.19	–
Text & Streaming	28.85	7.60
Virtualization	13.04	4.00
TCP path length unit	6.20	–

Note that processing at the end device takes a long time, because the operational speed of the Intel E3845 Atom processor used in the Pepper robot is about one-tenth that of an Intel Core i7, which is now widely used. When Core i7 or its equivalent CPUs are applied to robotic products in the future, processing times are expected to be reduced to about 40 ms, increasing the proportion of delay occurring in the network. Assuming improvement of end-device performance, it will be effective to provide services using edge servers.

V. Effect of Service Function Relocation on Application-Level Quality

In this section, we evaluate the impact of live-migrating VMs on application-level quality. We show the packet loss penalties resulting from VM live migration, and show the effectiveness of service function relocation by changing the amount of background traffic.

In this section, we only considers a function on an edge server that inserts text into video. Furthermore, live migration of VMs is performed using the management network. Since background and application traffic are transferred via the data-plane network, it does not interfere with live migration. Note that live migration is performed in a LAN environment in our experiments, so the impact of live migration in a MAN/WAN environment will be larger than what is shown in this section [15].

Fig. 4: Application-level quality for different levels of background traffic

A. Application-level Quality and Packet Loss

One virtual machine that inserts text into video from the robot and ten other virtual machines that generate background traffic are deployed on the user-side edge server. Background and video live-streaming traffic, about 3.37 Mbps, use the 1 Gbps network interface of the edge server. When the total amount of traffic exceeds 1 Gbps, packet loss occurs since it exceeds the capacity of the interface. As a result, the receiving rate of the user PC decreases. In our experiment, application-level quality clearly worsened when the receiving rate fell below 80% (Fig. 4). Network measurement researcher shows a 20% packet loss caused errors in about 90% of MPEG frames [11].

B. Penalty of VM Live Migration

In this paper, the OpenStack compute nodes as the edge servers uses KVM as a hypervisor. OpenStack instructs KVM to live-migrate a VM, initiating the pre-copy live migration [14]. In pre-copy live migration, the new VM is started at the migration destination, and the 4 GB of memory of the source VM is copied to the new VM. Memory pages that changed at the source VM during copying are repeatedly copied to the new VM. When the number of uncopied memory pages becomes small enough to minimize their transmission time, the hypervisor stops the source VM and instantly copies the uncopied memory. External communication is paused during reconnection. Repeated measurements show that this downtime is about 0.5 s, causing frame errors in live-streaming video. Storage of the VM has already shared by NFS.

C. Demonstration and Evaluation

Results of the experiment show that when live migration is performed just before the receiving rate decreases to 80%, the effect of service function relocation on application-level quality is maximized. However, it is actually difficult for controller nodes to measure the receiving rate at the user PC, because it is considered that the user PC is not under the control of the network operator that provides MEC. Also, it is not currently possible to know the packet loss rate for a specific service by metering the network interface of the machine. We thus developed a program that meters edge server network interfaces using SNMP (Simple Network Management Protocol), and performed VM live migration before packet loss occurred due to monotonically increased background traffic. This is a simple

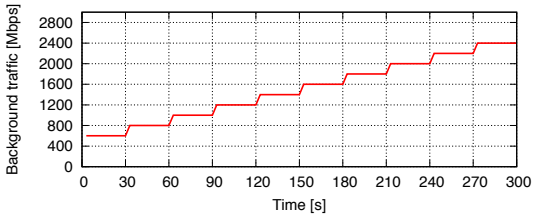


Fig. 5: Background traffic

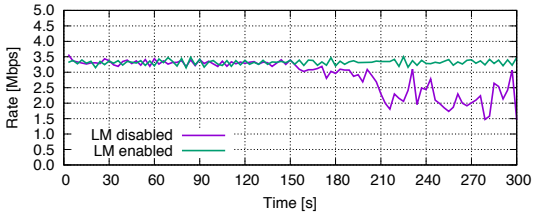


Fig. 6: Receiving rate at the user PC

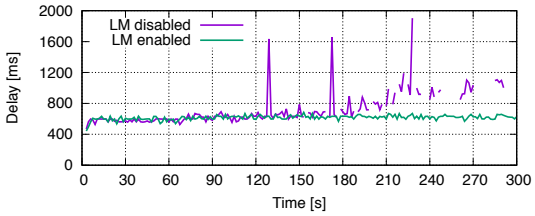


Fig. 7: Application-level delay

program for automated resource management that dynamically relocates service functions based on the network load of edge servers. The program, running on the controller node, instructs the OpenStack controller to perform VM live migration when the traffic rate exceeds 950 Mbps. The program issues SNMP query to obtain the traffic volume on the network interface. Then, the traffic rate is calculated based on the difference of traffic volume between two successive queries. The interval of queries is set to 3 seconds, and the VM live migration starts only when the traffic rate exceeds 950 Mbps over three intervals.

We set up scenarios where VM live migration is enabled and disabled, and measured receiving rates and application-level delay at the user PC under conditions of continually increasing background traffic. We monotonically increase the background traffic by 200 Mbps every 30 seconds so that VMs are not repeatedly migrated (Fig. 5). In the live migration enabled scenario, the VM is migrated from the user-side edge server to the robot-side edge server, as instructed by the program we developed. Application-level delay is measured as described in Section IV. Figures 6 and 7 show the results.

The results for the live migration disabled scenario show that the video receiving rate at the user PC decreases as the background traffic increases. Intermittent, extremely large application-level delays and quality degradation start when the video receiving rate at the user PC decreases to about 2.7 Mbps, which is about 80% of the normal rate. Places where

the application-level delay is not plotted in Fig. 7 indicate that the application-level delay could not be measured, because video frames were damaged and camera capture times could not be read.

The results for the live migration enabled scenario show no degradation in application-level quality. Since the service function was relocated to a robot-side edge server just before the occurrence of packet loss, the receiving rate of the video remained stable at normal levels, despite increased background traffic at the user-side edge server. The results of packet capture show that live migration ends at time 90. The time required for live migration was about 13 s, and the communication downtime was about 0.2 s. Live migration and downtime ended simultaneously. Although the video was momentarily distorted during the downtime, it could not be grasped by the screenshots for application-level delay measurement. This penalty is sufficiently small compared to the degradation of application-level quality when the background traffic is significantly large.

We confirmed that MPEG2 video began to be damaged when packet loss was 20%, and that delay and packet loss due to VM live migration are very temporary. We thus empirically confirmed that service function relocation is useful for maintaining application-level quality.

VI. Conclusion

Standardization of MEC is progressing, and the introduction and deployment of MEC is becoming a reality. It is thus important to understand the application-level delays and their factors in the MEC environments, and to investigate the effects of relocating service functions on application-level quality from the viewpoint of future network and service design. In this paper, we measured application-level delays and investigated the effect of service function relocation on application-level quality in a MEC environment. The results showed that increased processing delays due to software operations are small as compared to delays caused by geographical factors. We showed that providing services using edge servers can reduce application-level delays between end devices by 15–30%. This demonstrates that application-level quality can be improved by MEC. We also revealed that delays and packet loss due to VM live migration are very temporary, and sufficiently small as compared to the degradation of application-level quality when background traffic is significantly large. We thus empirically confirmed that service function relocation is useful for maintaining application-level quality. In future work, we will perform live migration in a MAN and WAN environments to evaluate the effects of service function relocation on application-level quality at larger scales. In that case, we will need to migrate VM storage as well as memory.

Acknowledgment

This work was partially supported by Grant No. 19104 from the National Institute of Information and Communications Technology (NICT) in Japan.

References

[1] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-based Processing: Opportunities and Challenges," in *Proceedings of IEEE SCC 2015*, Jun. 2015, pp. 285–292.

[2] K. Surabhi and M. Saurabh, "Smart buildings: How IoT technology aims to add value for real estate companies," Deloitte University Press, Apr. 2016.

[3] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of big data on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, Jan. 2015.

[4] Z. Huang, W. Li, P. Hui, and C. Peylo, "CloudRidAR: A Cloud-based Architecture for Mobile Augmented Reality," in *Proceedings of ACM workshop on Mobile augmented reality and robotic technology-based systems*, Jun. 2014, pp. 29–34.

[5] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases & Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, Jun. 2017.

[6] "Mobile Edge Computing A key technology towards 5G," ETSI, Sep. 2015.

[7] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Architecture & Orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, May 2017.

[8] "Mobile-Edge Computing (MEC); Framework and Reference Architecture," ETSI GS MEC 003 V1.1.1, Mar. 2016.

[9] OpenStack.org. [Online]. Available: https://www.openstack.org/

[10] M. Claypool and J. Tanner, "The Effects of Jitter on the Peceptual Quality of Video," in *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, Oct. 1999, pp. 115–118.

[11] J. M. Boyce and R. D. Gaglianello, "Packet Loss Effects on MPEG Video Sent Over the Public Internet," in *Proceedings of the sixth ACM international conference on Multimedia*, Sep. 1998, pp. 181–190.

[12] J. Zhang, W. Xie, F. Yang, and Q. Bi, "Mobile Edge Computing and Field Trial Results for 5G Low Latency Scenario," *China Communications*, vol. 13, no. 2, pp. 174–182, Feb. 2016.

[13] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile Edge Computing Potential in Making Cities Smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, Mar. 2017.

[14] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A Survey on Virtual Machine Migration: Challenges, Techniques and Open Issues," *IEEE Communications Surveys & Tutorials*, Jan. 2018.

[15] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. De Laat, J. Mambretti, I. Monga, B. Van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, Oct. 2006.

[16] M. Jammal, H. Hawilo, A. Kanso, and A. Shami, "Mitigating the Risk of Cloud Services Downtime Using Live Migration and High Availability-Aware Placement," in *Proceedings of IEEE CloudCom 2016*, Dec. 2016, pp. 578–583.

[17] S. Winkler and P. Mohandas, "The Evolution of Video Quality Measurement: FromPSNR to Hybrid Metrics," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 660–668, Jun. 2008.

[18] SoftBank Robotics. [Online]. Available: https://www.ald.softbankrobotics.com/en/robots/pepper

[19] FFmpeg. [Online]. Available: https://ffmpeg.org/

[20] "Network Functions Virtualisation - White Paper #3," ETSI, Oct. 2014.

[21] R. Mijumbi, J. Serrat, J.-l. Gorricho, S. Latré, M. Charalambides, and D. Lopez, "Management and Orchestration Challenges in Network Functions Virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

[22] Z. Wu and H. V. Madhyastha, "Understanding the Latency Benefits of Multi-Cloud Webservice Deployments," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 2, pp. 13–20, Apr. 2013.