Mobile Edge Computing Potential in Making Cities Smarter

Tarik Taleb, Sunny Dutta, Adlen Ksentini, Muddesar Iqbal, and Hannu Flinck

The authors propose an approach to enhance users' experience of video streaming in the context of smart cities. The proposed approach relies on the concept of MEC as a key factor in enhancing QoS. It sustains QoS by ensuring that applications/services follow the mobility of users, realizing the "Follow-me-Edge" concept.

ABSTRACT

This article proposes an approach to enhance users' experience of video streaming in the context of smart cities. The proposed approach relies on the concept of MEC as a key factor in enhancing QoS. It sustains QoS by ensuring that applications/services follow the mobility of users, realizing the "Follow Me Edge" concept. The proposed scheme enforces an autonomic creation of MEC services to allow anywhere anytime data access with optimum QoE and reduced latency. Considering its application in smart city scenarios, the proposed scheme represents an important solution for reducing core network traffic and ensuring ultra-short latency through a smart MEC architecture capable of achieving the 1 ms latency dream for the upcoming 5G mobile systems.

INTRODUCTION

Over the years, technology has served humanity by providing sustainable technical solutions to the social problems faced by society. In recent years, the research communities have been working on optimizing the technological infrastructure and maximizing the efficiency of services for citizens to meet their changing needs for smarter living. Society has evolved, and in the present era of smartphones, we have a new concept, the smart city," which is increasingly gaining in importance. Smart cities are expected to improve the quality of life for their citizens, leveraging advanced information and communications technologies (ICT). Smart cities are also expected to provide their citizens with a variety of innovative services, ranging from education and healthcare to augmented and immersive reality; for example, for the support of tourism. Indeed, deployed services in smart cities will involve not only smartphones and tablets, but also utility meters, washing machines, thermostats, refrigerators, sensors for environmental monitoring, and so on; in short, the different components of the Internet of Things (IoT) ecosystem.

The next generation mobile systems, commercially known as fifth generation (5G), aims to accelerate the development of smart cities, by not only increasing the data delivery rates but also accommodating the expected high numbers of IoT devices to be used by smart city services and applications [1, 2]. Besides, thanks to its elasticity and agility, 5G will be able to support numerous smart services, which cannot be supported by current network architectures [3, 4]. This includes immersive reality and tactical applications, and services with highly strict requirements in terms of ultra-short latency and high responsiveness.

5G systems will rely on technologies such as Network Function Virtualization (NFV), Software Defined Networking (SDN), and cloud computing to attain system's flexibility and true elasticity [1, 4]. Among these technologies, cloud computing has tremendously advanced enabling diverse services. However, it remains limited against emerging applications (e.g., tactile Internet and augmented reality) that require ultra-short latency. Cloud is also limited against computation-intensive applications running on power/CPU-constrained user equipment (e.g., mobile gaming) that need to partially run their computation in the cloud while ensuring response times (i.e., for other parts of the code running on the user equipment) in the range of milliseconds. These limitations are principally due to the centralized cloud computing architecture. Mobile edge computing (MEC), interchangeably known as fog computing (originating from the cloudlet concept [5]), represents a vital solution to these limitations. Indeed, it reforms the cloud hierarchy by pushing computing resources in the proximity of mobile users (i.e., at the mobile network edge). There are high expectations for MEC and 5G, when efficiently integrated, to improve the quality of life of residents in smart cities. This underpins the focus of this article, wherein we show how MEC will enable emerging services for smart cities, focusing on an augmented reality use case involving streaming of high definition (HD) video, which is for the support of tourism in smart cities. The overall objective is to demonstrate how high quality of service (QoS) can be maintained regardless of the mobility of users through the use of MEC, more particularly through the concept of Follow Me Edge (FME — similar in spirit to the Follow Me Cloud concept [1, 6]). FME ensures that the service constantly follows the user and that the user is always serviced from the closest edge. As discussed later, the fundamental observations made about the envisioned use case are highly applicable to other services requiring ultra-short latency, such as immersive reality and tactical applications.

The remainder of this article is organized as follows. The following section presents the state of the art. Then we describe our proposed FME framework along with the supporting mecha-

Digital Object Identifier:
10.1109/MCOM.2017.1600249CM

Tarik Taleb and Sunny Dutta are with Aalto University; Tarik Taleb is also with Sejong University; Adlen Ksentini is with Eurecom Institute; Muddesar Iqbal is with London South Bank University; Hannu Flinck is with Nokia Bell Labs.

nisms. For the sake of performance evaluation, in the next section we portray the experimental setup and discuss the obtained results. The article concludes in the final section with a summary recapping the main findings.

State of the Art

The key idea beneath MEC is to place storage and computation resources at the network edge, in the proximity of users. Accordingly, data processing can be pushed from far remote cloud to the edge. By processing data locally and accelerating data streams through various techniques (i.e., caching and compression), MEC reduces the traffic bottleneck toward the core network. Besides, it helps shorten end-to-end latency, enabling the offload of important computation load from power-constrained user equipment to the edge. As discussed in the executive briefing of the European Telecommunications Standards Institute (ETSI) MEC initiative,[1] edge computing shall enable new computation-intensive services and shall yield promising business models. It also represents a fault resilient solution for its decentralized architecture [7].

Given its potential, MEC has been gaining lots of momentum among industries and within the researcher community [8]. Im-portant standardization activities have been initiated. Indeed, to standardize the specifications of MEC across mobile operators and vendors in the value chain, ETSI formed a new ISG group in 2014 and came up with different industry specifications.[2] The specifications highlight the different service scenarios whereby MEC can be beneficial. For video streaming services, it was recommended to apply intelligent video acceleration schemes using video analytics and video management applications within MEC. The research work in [9] proposes a two-hop network whereby edge architecture enhances data transfer rate and throughput for video streaming compared to remote cloud. The work in [10] exploits network assisted adaptive streaming applications for multimedia content delivery inside MEC to enhance Quality of Experience (QoE). The research study in [11] proposes an architecture with distributed parallel edges to increase QoE for content delivery. The research work in [12] makes use of edges as caches along with proxies to store media content. It also enforces computation offloading to increase the lifetime of mobile devices. In [7], edges function independently as small-scale data centers on their own and are used for video caching and streaming.

In all the above research work, MEC is deemed to be a promising solution for handling video services. Its limitations in terms of resource control and orchestration have also been highlighted as important challenges. In smart city scenarios, users' mobility and the need for dynamic service migration add to these challenges. Most research works on the latter consider traditional cloud environments [1, 6]. In [13], migration of edges has been proposed using a Markov decision process approach to determine optimal solutions for service placement.

To the best of the authors' knowledge, mobility support and migration of service in terms of video content delivery have not been considered yet. In the remainder of this article, we describe and showcase an innovative deployment scenario on how a user's experience on video streaming can be enriched using MEC in spite of the user's mobility.

Follow Me Edge

Use Cases

To support tourism in smart cities, many use cases, involving video streaming from the edge, could be considered. In the following, we consider two representative use cases, one implying edge migration:

Use Case 1: Robert from England visits Helsinki for the first time. He visits the white church, likes it, takes a video of it, comments on it in his native language (i.e., English), and streams it to an edge placed near the white church. Some time later, Eric, also from England and a member of Robert's social network (e.g., Facebook), visits the same church and receives an invitation to view Robert's generated video and hear what Robert said about the location. Eric may further comment on the video, indicating whether he liked it, or post a new video about the location. In this use case, videos about a certain attractive location are cached at edges in the vicinity of that location and streamed to people visiting that location when there is interest or when there is linkage with the video publisher. Mapping the most popular videos with Google Streetview may also be considered. The video streaming as well as the relevant operations (comment, like/dislike, etc.) take place at the corresponding edges near the visited sites.

Use Case 2: Robert visits the city of Hamburg. To explore the city, he takes a sightseeing bus. He uses interactive glasses that recognize historical monuments (e.g., tourist attractions) and accordingly receives introductory video about these monuments in the format of high definition (HD) video. The video can be streamed from either a remote cloud or the edge. As the path to the remote cloud involves multiple hops, some being nearly congested, high resolutions of the video cannot be guaranteed unless it is streamed from the edge. Furthermore, to prevent jitter and the associated degradation in QoE, the video must always be streamed from the nearest edge to Robert. In this use case, Robert's user equipment receives a portion of the video from the nearest edge A. As the bus gets far away from edge A and closer to edge B, the video along with the streaming virtual network function are migrated to edge B, and the remaining portion of the video streams to Robert from edge B. This edge migration occurs in a manner transparent to Robert, who continues enjoying the video without any disruption in the video stream and with no degradation in the perceived QoE.

While the above use cases focus on video streaming services, similar use cases with the same requirements can be derived for augmented reality services. In this work, we consider using lightweight virtualization technologies (i.e., container), and introduce container migration to meet the above mentioned use cases, with more focus on the edge mobility aspect of use case 2.

FME Architecture

The proposed architecture is based on the two-tier principle, wherein the cloud service provider (CSP) gives access, through an appropriate application programming interface (API) [14], to a content provider or a third party over-the-top (OTT) service

The key idea behind MEC is to place storage and computation resources at the network edge, in the proximity of users. Accordingly, data processing can be pushed from a far remote cloud to the edge. By processing data locally and accelerating data streams through various techniques, MEC reduces the traffic bottleneck toward the core network.

[1] https://portal.etsi.org/portals/0/tbpages/mec/docs/mec_executive_brief_v1_28-09-14.pdf

https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1_18-09-14.pdf

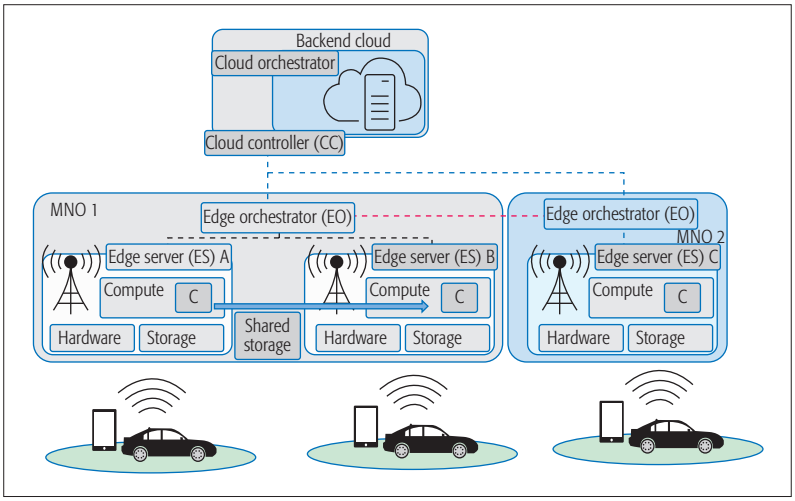

Figure 1. The envisioned mobile edge computing architecture.

to use the cloud resource to deploy its application. The cloud has its own orchestrator to manage the cloud infrastructure and its resources. An additional component is considered: the cloud controller (CC). Considering the business functionality, CC is involved in maintaining the service level agreement (SLA) with the OTT providers and mobile network operators (MNOs). The agreement deals with access rights and policies on the entity's authority. The edge server (ES) belongs to the MNO's network, where it is managed and controlled by the edge orchestrator (EO). Every MNO has its own EO, managing its own set of ES clusters. Figure 1 depicts inter/intra-MNO edge network. The dotted lines represent the agreement level connection among CSP, MNO, EO, and ES.

The ES is hosted on virtual machines on top of the existing server hardware residing in the MNO's edge network node. The ES has its own compute and storage. The compute node is responsible for hosting container-based applications on the edge. The storage is used to keep images of the application containers. For an intra-edge network, additional shared storage is needed to ensure live migration of containers between edge compute nodes. Linux-based containers (LCs) can be employed to make the system lightweight and help in easily deploying service packages. LCs run applications/services provided from the edge, while EO is in charge of deploying, controlling, and migrating containers.

Referring to use case 2, when Robert connects to ES A to watch an HD video introducing Hamburg, he may initially be served from the backend cloud. As stated earlier, this may incur jitter and may limit the video resolution. To cope with this issue, the EO may instantiate a container on the connected ES compute node with built-in streaming and transcoding virtualized functionality [14, 15]. Subsequently, it may store the relative content from the backend cloud into the ES's local storage. Robert will then be served the HD version of the video stream from the ES. Considering the case of HTTP-based video streaming, the EO can fetch the entire media content on one go or may fetch only a certain number of video chunks at a time. Consequently, as the content will be served from one hop away, the user's perceived quality is expected to greatly improve. For applications involving OTT services, the established SLA may help in performing this task with a pre-agreed negotiation between the OTT provider and the MNO. In this case, the EO inside the MNO will get access rights from the OTT service at the beginning of the process. The EO will then create the replica and bring the service to the edge.

Although the content is now served from the nearest edge, after some time the connection with the mobile user may begin experiencing degradation as the length of the path to the served MEC increases. To maintain the same quality, it is vital that the content moves along the physical mobility of users in an FME fashion [6]. To realize the FME vision, the EO needs to keep updated information about its resources and the user locations. The latter may be obtained using the MEC's active device location tracking functionality, based on which a user's velocity and direction may be derived. Taking this into consideration, the EO may estimate the latency between the user and the current edge, and compare it with the latency between the same user and the target edge. Once deemed appropriate, the EO may trigger live migration of the container in a proactive manner. This will consist of migrating the video streaming service along with its contents. Upon successful container migration, the user may then be served from the new ES, which will ensure low latency access to the content. The above described migration process will be repeated along the track whenever required.

Migration can happen using various techniques. In the case of live video streaming, service continuity and bare minimum disruption are of prime concern. To perform seamless live migration, the service state has to be maintained in order to ensure that no data is lost. This is achieved by transferring the entire memory content of the running instance (i.e., container) from the source ES to the target ES. The source ES keeps track of which memory blocks are modified while the transfer is in progress. Once this initial transfer is complete, the changes that have occurred in the meantime are transferred again. This continues until the newly built instance becomes exactly identical to the old one. This ensures that after the migration process is complete, the video starts from the exact point rather than overlapping. Indeed, in the case of mishandled memory, data loss happens. This incurs overlap in video playtime, where the user may have to watch the same content again (from the span when the migration started). Moreover, the migration duration should not be too long. If the duration is too long, it might be that by the end of the migration either the user has moved away from the ES location or the played video is almost over. To overcome these constraints, separate shared storage has to be considered. Normally migration takes place by copying memory blocks. Thus, if the blocks are dumped at a shared location attached to the new ES, service transfer becomes faster than in the case considering local storage. Although async mode configuration of shared storage is even faster than sync mode, we propose the use of sync mode to maintain data integrity. In sync mode the data saved in the storage location is confirmed before processing the next request from the ES. In async mode, the requests are processed without proper confirmation. It yields better response time

but at the cost of possible data corruption, which may introduce a glitch in the played video.

So far, the proposed scheme deals with latency reduction and mobility within the network of the same MNO/edge. If the user moves out of the network of an edge provider to the edge of another provider, the SLA shall be used. The SLA should enforce an integrated architecture where the EO handover, shared storage concept, and service migration are considered. In this case, the source EO may hand over the control to the target EO (in a separate MNO's network) and permit service migration. If it is not possible, then during the MNO crossover phase, the content will be served from the back-end cloud temporarily until the new EO again caches the service in its own compute node.

It is worth noting that smart caching and migration can considerably enhance the overall system performance and reduce the migration cost [16]. The caching concept can be enhanced further by considering the remaining duration of the video. If there is no possibility to store the whole content (due to storage space), only the next few chunks of the remaining video may be cached. Moreover, if the video is almost at the end (i.e., the remaining play time less than the migration time), the container/service migration may be simply omitted.

Performance Evaluation

Figure 2 portrays the testbed environment we have built to simulate edge-based video streaming and its mobility considering use case 2. The testbed is built using one Ubuntu 14.04.3 LTS desktop and two laptops with the same host operating system. Virtualbox is used to implement the testbed on a desktop workstation machine. The desktop machine hosts three virtual machines (VMs) inside the virtual box environment. VM1 is used as a gateway for the entire network to access the Internet. VM2 is used to simulate the cloud environment deploying a Devstack-based cloud, which provides all-in-one (i.e., controller, compute, network, and storage on the same node) with an Ubuntu instance running inside it. The Ubuntu cloud instance hosts an HTTP live streaming (HLS) server. The ffmpeg open source server, for both streaming and transcoding, are built in separate VMs. The media contents (HLS fragments) are generated using ffmpeg transcoding servers, and are then streamed using an ffmpeg streaming server (hosted in a separate VM). The floating IP address of the instance was chosen from the same IP subnet range of the edge cluster, so the ES can access the data from the cloud VM. The CC function was omitted, as the SLA level implementation was not considered in the testbed. VM3 was configured using Proxmox VE and acts as edge cluster controller — EO. VM3 also includes a DHCP server with authentication. To automate the orchestration process, a script is used to:

- Monitor the session changeover of the clients from one edge to the other using the authentication server logs
- Handle the container migration

The entire cluster of edges is formed by integrating two additional VMs (i.e., VM4 and VM5) with the EO. VM4 and VM5 are hosted inside laptops to emulate ESs. The connectivity between the VMs is extended using an Ethernet switch. VM4 and VM5 use the same virtual environment as the EO. To ensure that the laptops (VM4 and VM5) act as edge access points, the wireless LAN interface was configured using Host-apd in IEEE 802.11 master mode. The container is created inside VM4. We use Openvz containers for the testbed. The containers are built with Ubuntu cloud minimal image using Nginx as the web server. Nginx is configured to serve as reverse proxy to the back-end cloud HLS server with caching and streaming functionality. It is worth recalling that the objective of these tests is to validate the use of MEC to ensure high-quality HD video streaming service to mobile users. Therefore, the focus of these tests is on caching content and live delivery of the multimedia content closer to the user at the edge.

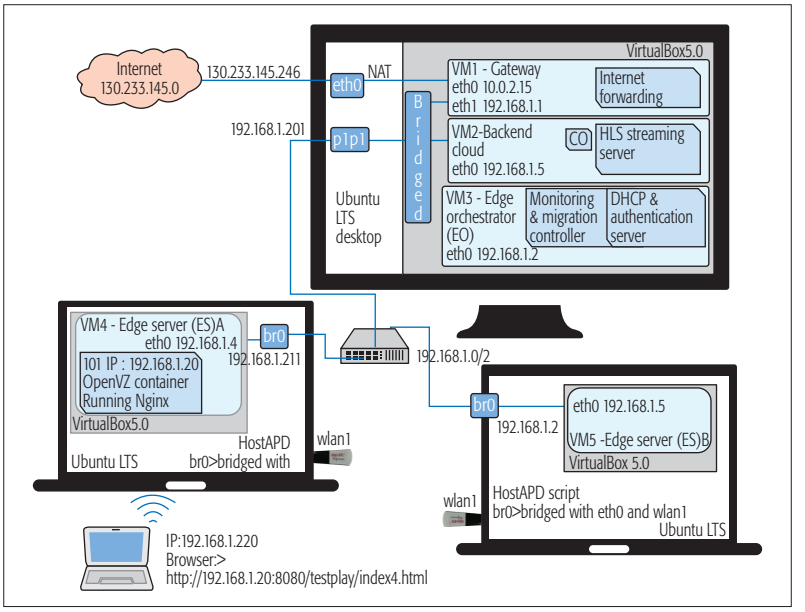

Figure 2. Envisioned testbed setup.

To perform the test, one container is instantiated in Edge1 with all the features explained above. When a user (using a smartphone or laptop) connects to the network through SSID, the user is assigned an IP from the same IP subnet pool of the ES. The user connectivity log along with the MACID of the user are saved in a database of the EO. The user launches a browser and starts browsing the video, using the URL of the streaming sever hosted at the container. To implement minimum security, the user is given authorization to only browse data from the container. Upon connecting for the first time, the container forwards the request to the cloud VM, and the multimedia content is served from the back-end cloud. Simultaneously, it caches the relevant media contents and stores them for further use to the container. For the next requests to the same video, the container makes use of its own streaming functionality to serve the user by using cached contents regardless of the fact that the cloud is accessible or not. Accordingly, this implements the concept of bringing the content closer to the user and making the backend core free from the traffic.

To simulate mobility, laptops are placed at a distance from a multihop network. During the video playback, the user device is deliberately moved from the first edge toward the second edge connected to the next hop in the network. The user automatically connects to Edge2. As soon as the wireless connectivity handover takes place, the logs are generated inside the EO. Upon

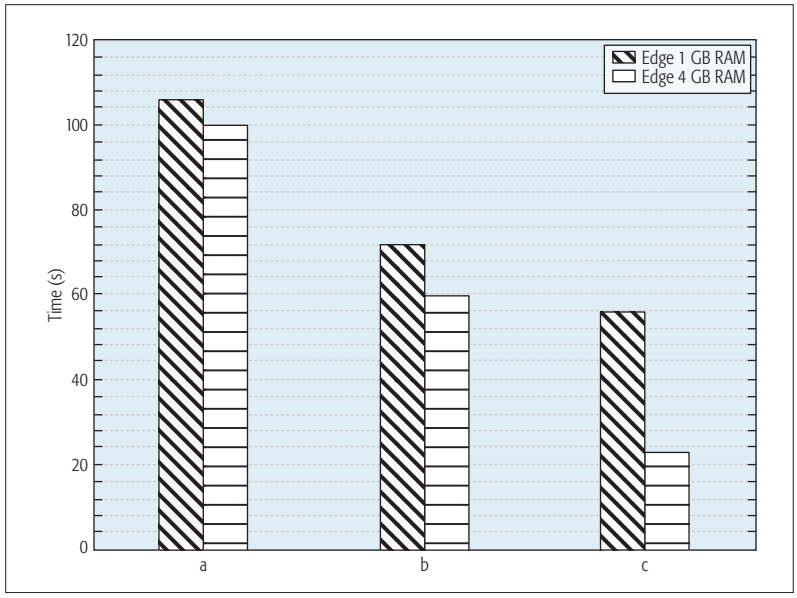

Figure 3. Live migration time using local storage: a) with streaming online mode; b) without streaming offline mode; c) blank container.

detecting the client's connectivity, the script in charge of automating the service migration gathers the client info (i.e., MACID), compares it to the database, identifies the same client's movement to a new edge, and subsequently triggers the live migration of the container from the old ES to the next one to which the client is directly connected. For Openvz, the live content delivery is done using checkpoint/restore in userspace (CRIU). It performs vz-dump (memory block dump) to save the state and uses rsync (i.e., incremental file transfer utility) to transfer the file to the target location. It performs a dual-level operation to prevent data loss. First, pre-copy starts from the point at which the migration is initiated. Once completed, the container initialization is started in the new edge along with post-copy. Herein, post-copy represents transfer of the residual amount of changes that occurred in the memory block during this small interval. As service autostart is already enabled, once this data transfer operation is done, the container is automatically started in the target edge, and the old one is released. The user remains unaware of this fact and enjoys normal streaming. Throughout the migration time, the container IP address remains the same, ensuring no service downtime (i.e., the session remains active) during this span.

For service migration, the test is performed with two types of storage. In the first type, the whole operation is performed using local storage (i.e., service migration within a federated edge network). The vz-dump files are first copied to the local storage, then synched with the target ES node, the container is initialized in that target node, and after post-copy the migration is completed. However, this method causes high delays. To achieve better performance with minimum response, the second type is implemented. A network file system (NFS) server is used as shared storage for the operation. The NFS server is installed inside the EO, and the shared space is defined for the cluster nodes. The shared storage is used only for vz-dump files. During migration, the copied memory files are stored in the shared location. As the target node can access the shared location directly, it reduces the content delivery to the edge, resulting in faster response.

In Fig. 3, we plot the migration duration of one container for three different conditions:

- With streaming online mode — streaming in use and the client watching the video
- Without streaming offline mode — streaming in use and the client is not watching the video with no changes in the memory blocks
- Blank container with two different types of ES

The migration latency is plotted considering local storage. The migration latency of a blank container is plotted to showcase how much added services impact the migration time. From the results, we can observe that when video streaming is not active, the content migration takes less time compared to the case when the video is being streamed. Moreover, for an ES with higher RAM capacity, the migration duration is shorter. This is attributable to the fact that copying memory pages takes less time with higher RAM, leading to a slight decrease in the overall duration.

Figure 4 plots the migration duration when considering various ways to share the storage among edges. The test is performed with two different sizes of containers, one small and another big, to investigate if container size affects migration duration. We clearly remark that the container size merely affects the migration latency. Besides, we observe that shared-sync mode achieves shorter latency in comparison to local mode. Furthermore, the shared storage, if configured in shared-async mode, reduces the duration of the migration closer to 10 s. In this last mode, the video experienced a single glitch of 1∼2 s. We explain this by the fact that data corruption took place during async mode, resulting in reduced quality of experience (QoE) [17]. The results obtained through this evaluation reveal that the storage type and memory capacity have high impact on the migration latency.

Conclusion and Future Research Directions

In this article, we propose a framework that leverages MEC to support diverse applications in smart city scenarios. To always ensure high QoE, the Follow Me Edge concept is introduced. According to this concept, services move across edge servers as per the movement of their respective users. The proposed framework is validated using a real-life testbed. Edge mobility was tested using different storage types, different container sizes, and different edge resources.

Interesting results were obtained, suggesting migration latency depends on the different techniques used. The obtained results also demonstrate that short migration latency does not necessarily guarantee high QoE. It becomes apparent that the complexity of the system arises as a trade-off between short migration latency at the cost of possible data loss. Based on the obtained results, it can be concluded that a mechanism to select the right combination of techniques to be used for efficiently migrating a service is of vital importance. This defines one of the authors' future research directions in this area.

Acknowledgments

This work was partially supported by the TAKE 5 project funded by the Finnish Funding Agency for Technology and Innovation (TEKES) and in part by the Finnish Ministry of Employment and the Economy. It is also partially supported by the European Union's Horizon 2020 research and innovation programme under the 5G!Pagoda project with grant agreement no. 723172.

References

[1] T. Taleb, A. Ksentini, and A. Kobbane, "Lightweight Mobile Core Networks for Machine Type Communications," *IEEE Access*, vol. 2, Oct. 2014. pp. 1128–37.
[2] T. Taleb and A. Kunz, "Machine Type Communications in 3GPP Networks: Potential, Challenges, and Solutions," *IEEE Commun. Mag.*, vol. 50, no. 3, Mar. 2012.
[3] T. Taleb *et al.*, "EASE: EPC as a Service to Ease Mobile Core Network," *IEEE Network*, vol. 29, no. 2, Mar. 2015, pp. 78–88.
[4] T. Taleb, "Toward Carrier Cloud: Potential, Challenges, and Solutions," *IEEE Wireless Commun.*, vol. 21, no. 3, June 2014, pp. 80–91.
[5] U. Shaukat *et al.*, "Cloudlet Deployment in Local Wireless Area Networks, Motivation, Taxonomies, and Open Research Challenges," *J. Network Computer Applications*, vol. 62, Feb. 2016, pp. 18–40.
[6] A. Ksentini, T. Taleb, and F. Messaoudi, "A LISP-Based Implementation of Follow Me Cloud," *IEEE Access*, vol. 2, Oct. 2014. pp. 1340–47.
[7] H. Chang *et al.*, "Bringing the Cloud to the Edge," *Proc. IEEE INFOCOM Wksps.)*, Toronto, Ontario, Canada, May 2014.
[8] A. Ahmed and E. Ahmed, "A Survey on Mobile Edge Computing," *Proc. IEEE 10th Int'l. Conf. Intelligent Systems Control*, India, May 2016.
[9] D. Fesehaye *et al.*, "Impact of Cloudlets on Interactive Mobile Cloud Applications," *Proc. IEEE 16th Int'l Conf. Enterprise Distributed Object Computing*, Beijing, China, Sept. 2012.
[10] J. Fajardo, I. Taboada, and F. Liberal, "Improving Content Delivery Efficiency through Multi-Layer Mobile Edge Adaptation," *IEEE Network*, vol. 29, no. 6, Dec. 2015, pp. 40–46.
[11] W. Zhu *et al.*, "Multimedia Cloud Computing," *IEEE Signal Processing Mag.*, vol. 28 , no. 3, May 2011, pp. 59–69.
[12] Y. Jararweh *et al.*, "Resource Efficient Mobile Computing Using Cloudlet Infrastructure," *Proc. IEEE 9th Int'l. Conf. Mobile Ad Hoc Sensor Networks*, Dalian, China, Dec. 2013.
[13] S. Wang *et al.*, "Dynamic Service Migration in Mobile Edge-Clouds," *Proc. IFIP Networking Conf.*, Toulouse, France, May 2015.
[14] P. Frangoudis *et al.*, "An Architecture for On-Demand Service Deployment over a Telco CDN," *IEEE ICC '16*, Kuala Lumpur, Malaysia, May 2016.
[15] T. Taleb, A. Ksentini, and R. Jantti, "Anything as a Service for 5G Mobile Systems," *IEEE Network*, vol. 30, no. 6, Dec. 2016, pp. 84–91.
[16] T. Taleb and A. Ksentini, "An Analytical Model for Follow Me Cloud," *Proc. IEEE GLOBECOM*, Atlanta, GA, Dec. 2013.
[17] S. Dutta, T. Taleb, and A. Ksentini, "QoE-Aware Elasticity Support in Cloud-Native 5G Systems," *Proc. IEEE ICC '16*, Kuala Lumpur, Malaysia, May 2016.

Biographies

Tarik Taleb is currently a professor at the School of Electrical Engineering, Aalto University, Finland. He has worked as senior researcher and 3GPP standards expert at NEC Europe Ltd. Prior to his work at NEC, until March 2009, he worked as anassistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI. He received his B.E. degree in information engineering with distinction, and his M.Sc. and Ph.D. degrees in information sciences from Tohoku University in 2001, 2003, and 2005, respectively. His research interests lie in the field of architectural enhancements to mobile core networks (particularly 3GPP's), mobile cloud networking, mobile multimedia streaming, and social media networking. He has also been directly engaged in the development and standardization of the Evolved Packet System. He is a member of the IEEE Communications Society Standardization Program Development Board and serves as Steering Committee Chair of the IEEE Conference on Standards for Communications and Networking. He has received many awards for his many contributions to the area of mobile networking.

Sunny Dutta obtained his M.Sc. and Bachelor's degree from the School of Electrical Engineering, Aalto University, Finland, and the West Bengal University of Technology, India, in 2016 and 2006, respectively. Prior to his Master's studies, he worked as an engineer assuming different roles in network administration, energy automation, and smart grid communication network infrastructure. His present research focus includes MEC, NFV, SDN, and multimedia content delivery.

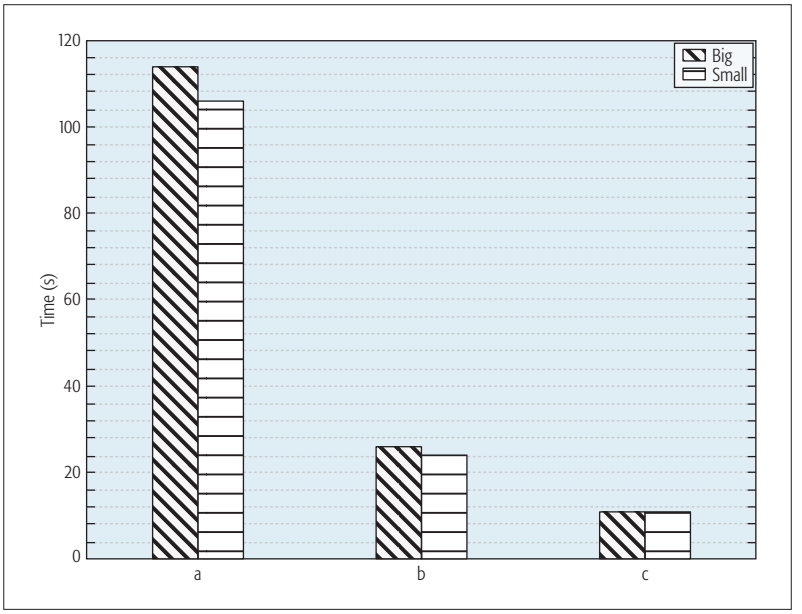

Figure 4. Live migration latency: a) local storage; b) shared sync storage; c) shared async storage.

Adlen Ksentini received his M.Sc. degree in telecommunication and multimedia networking from the University of Versailles Saint-Quentin-en-Yvelines, and his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005, with a dissertation on QoS provisioning in IEEE 802.11-based networks. From 2006 to 2015, he worked at the University of Rennes 1 as an assistant professor. During this period, he was a member of the Dionysos Team with INRIA, Rennes. Since March 2016, he has been working as an assistant professor in the Communication Systems Department of EURECOM. He has been involved in several national and European projects on QoS and QoE support in future wireless, network virtualization, cloud networking, and mobile networks. He has co-authored over 100 technical journal and international conference papers. He received the best paper award from IEEE IWCMC 2016, IEEE ICC 2012, and ACM MSWiM 2005. He has been acting as TPC Symposium Chair for IEEE ICC 2016 and 2017 and IEEE GLOBECOM 2017. He was a Guest Editor of *IEEE Wireless Communications*, *IEEE Communications Magazine*, and two issues of ComSoc *MMTC Letters*. He has been on the Technical Program Committees of major IEEE ComSoc, ICC/GLOBECOM, ICME, WCNC, and PIMRC conferences.

Muddesar Iqbal is working as a senior lecturer in mobile computing at the Computer Science and Informatics Division, School of Engineering, London South Bank University. He has been a principal investigator, co-investigator, technical lead, project manager, coordinator, and focal person of more than 10 internationally teamed R&D and capacity building training projects with total funding of over £1 million from different international organizations. He has co-founded and successfully launched a startup called SwanMesh Networks Ltd that was initially established in the United Kingdom to commercialize his Ph.D. project and now has over six years of design and development experience. Over the last few years, he has been actively involved in R&D projects in the area of mobile cloud computing and other open source networking technologies for applications in healthcare, disaster management, and community networks. He has won two awards from the Association of Business Executives UK while tutoring on computing modules.

Hannu Flinck is a research manager at Nokia Bell Labs Espoo, Finland. Before that he worked with Nokia Research Center and the Technology and Innovation unit of Nokia Networks in various positions. He has been actively participating in a number of EU research projects. He received his M.Sc. degree (1986) and Lic.Tech. degree (1993) in computer science and communication systems from Aalto University (at that time known as Helsinki University of Technology). His current research interests include mobile edge computing, SDN, and content delivery in mobile networks, particularly in 5G networks.