Mobile Edge Computing Potential in Making Cities Smarter

Tarik Taleb[1], Sunny Dutta[1], Adlen Ksentini[2], Muddesar Iqbal[3], and Hannu Flinck[4]
[1] Aalto University, Espoo, Finland
[2] Eurecom Institute, Nice, France
[3] London South Bank University, London, United kingdom
[4] Nokia Bell Labs, Espoo, Finland
Emails: {firstname.lastname@aalto.fi; adlen.ksentini@eurecom.fr; m.iqbal@lsbu.ac.uk; hannu.flinck@nokia-bell-labs.com }

***Abstract* – This paper proposes an approach to enhance users' experience of video streaming in the context of smart cities. The proposed approach relies on the concept of mobile edge computing (MEC) as a key factor in enhancing the Quality of Service (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 Quality of Experience (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, and that is through a smart MEC architecture capable of achieving 1 ms latency dream for the upcoming 5G mobile systems.**

I. INTRODUCTION

Over the years, technology has served the 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 the citizens to meet their changing needs for smarter living. The society has evolved, and in the present era of smartphones, we are yet coming across a new buzzword 'Smart City', which is increasingly gaining high 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 educational, healthcare-relevant, to augmented and immersive reality; e.g., 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 etc.; in short the different components of the Internet of Things (IoT) ecosystem.

The next generation mobile systems, commercially known as 5G, aims at accelerating the development of Smart Cities, not only by increasing the data delivery rates but also accommodating the expected high numbers of IoT devices to be used by Smart City services and applications [11][12]. Besides, thanks to its elasticity and agility features, 5G will be able to support numerous smart services, which cannot be supported by nowadays' network architectures [13][4]. This includes immersive reality and tactical applications, 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 [11][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 (originated from the concept of cloudlet [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 at MEC and 5G, when efficiently integrated, to improve the quality of life of residents in smart cities. This underpins the focus of this paper, wherein, we will show how MEC will enable emerging services for Smart Cities, focusing on an augmented reality use case involving the stream of High Definition (HD) video and that is for the support of tourism in Smart Cities. The overall objective is to demonstrate how high QoS can be maintained regardless the mobility of users and that is through the usage of MEC, more particularly through the concept of Follow Me Edge (FME – similar in spirit to the Follow me Cloud concept [1][14]). FME ensures that the service constantly follows the user and that the user is always serviced from the closest edge. As will be 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 paper is organized as follows. Section II presents the state of the art. Section III describes our proposed FME framework along with the supporting mechanisms. For the sake of performance evaluation, Section IV portrays the experimental setup and discusses the obtained results. The paper concludes in Section V with a summary recapping the main findings.

II. 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 traffic bottleneck towards the core network. Besides, it helps shortening 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 ETSI MEC initiative[i], 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 [3].

Given its potential, MEC has been gaining lots of momentum among industries and within the researcher community [2]. Important standardization activities have been initiated. Indeed, to standardize the specifications of MEC across mobile operators and

[i] https://portal.etsi.org/portals/0/tbpages/mec/docs/mec%20executive%20brief%20v1%2028-09-14.pdf

vendors in the value chain, ETSI formed a new ISG group in 2014 and came up with different industry specifications[ii]. 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 [6] proposes a two-hop network whereby edge architecture enhances data transfer rate and throughput for video streaming compared to remote cloud. The work in [7] exploits network assisted adaptive streaming applications for multimedia content delivery inside MEC to enhance Quality of Experience (QoE). The research study in [8] proposes an architecture with distributed parallel edges to increase QoE for content delivery. The research work in [9] 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 [3], edges function independently as small-scale datacenters 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 been also 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][14]. In [10], migration of edges has been proposed using Markov Decision Process approach to determine optimal solutions for service placement.

To the best knowledge of the authors, mobility support and migration of service in-terms of video content delivery has not been considered yet. In the remainder of this paper, 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.

III. FOLLOW ME EDGE

A. 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 nearby the white church. Sometime 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, indicates whether he liked it, or may 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 top popular videos with Google Streetview may be also considered. The video streaming as well as the relevant operations (e.g., comment, like/dislike, etc.) take place at the corresponding edges nearby 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 either from 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 be always 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 is streaming to Robert from Edge B. This edge migration occurs in a transparent manner to Robert who continues enjoying the video without any disruption in the video stream and with no degradation in the perceived QoE.

Whilst 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.

B. FME Architecture

The proposed architecture is based on the two-tier principle, wherein the cloud service provider (CSP) gives access, through appropriate API [15], to a content provider or a third party over-the-top (OTT) service 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 as Cloud Controller (CC). Considering the business functionality, CC is involved in maintaining the service level agreement (SLA) with the OTT providers and the mobile network operators (MNO). 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. Fig. 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 intra-edge network, additional shared storage is needed to ensure live-migration of containers between edge compute nodes. Linux-based containers (LC) 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 Edge Server A to watch a HD video introducing Hamburg, he may be initially

[ii] https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

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 [15][16]. Subsequently, it may store the relative content form the backend cloud into the ES's local storage. Robert will be then served the HD version of the video stream from ES. Considering the case of HTTP-based video streaming, the EO can fetch the entire media content at 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 largely 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 in the beginning of the process. The EO will then create the replica and bring the service to the edge.

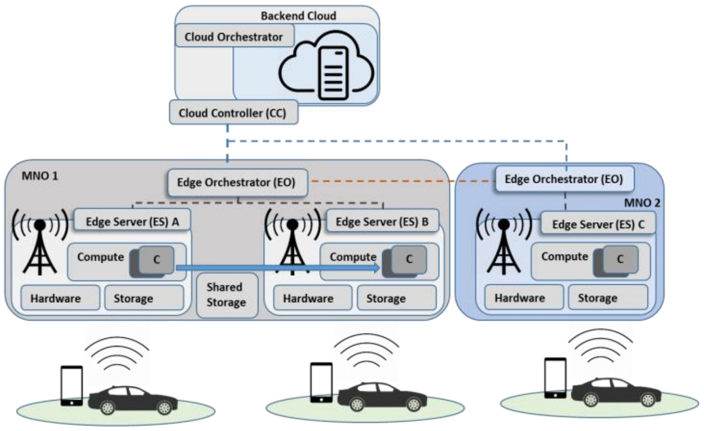

Fig. 1. The envisioned Mobile Edge Computing architecture.

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 a Follow Me Edge fashion [14]. To realize the FME vision, 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 user's velocity and direction may be derived. Taking this in consideration, 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, EO may trigger live migration of the container in a pro-active manner. This will consist in 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 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 occurred in the meantime are transferred again. This continues until the newly built instance becomes exactly identical as the old one. This ensures that after the migration process is complete, the video starts from the exact point rather than overlapping. Indeed, in 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 happen 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 constraint, separate shared storage has to be considered. Normally migration takes place by copying the memory blocks. Thus, if the blocks are dumped on a shared location attached with the new ES, then service transfer becomes faster than in the case of considering a local storage. Though 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 glitch in the played video.

So far, the proposed scheme deals with latency reduction and mobility within the network of the same MNO/edge. In case the user moves out of the network of an edge provider to the edge of another provider, the SLA agreement shall be used. SLA agreement should enforce an integrated architecture where the EO handover, shared storage concept and service migration are considered. In this case, the source EO may handover 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 backend 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 considerable enhance the system overall performance and reduce the migration cost [1]. The caching concept can be further enhanced 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 towards the end (i.e., remaining play time less than the migration time), the container/service migration may be simply omitted.

IV. PERFORMANCE EVALUATION

Fig. 2 portrays the test-bed environment which we have built to simulate the 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 work station 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 Ubuntu instance running inside it.

The Ubuntu cloud instance hosts a HTTP Live Streaming (HLS) server. The ffmpeg[iii] 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 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 that 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: *i)* monitor the session changeover of the clients from one edge to the other using the authentication server logs; *ii)* handle the container migration. The entire cluster of edges is formed by integrating additional two VMs (i.e., VM4 and VM5) with the EO. VM4 and VM5 are hosted inside laptops to emulate ES. The connectivity between the VMs is extended using an Ethernet switch. VM4 and VM5 use the same virtual environment as EO. To ensure that the laptops (namely VM4 and VM5) act as edge access point, 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 webserver. The Nginx is configured to serve as reverse proxy to the backend 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.

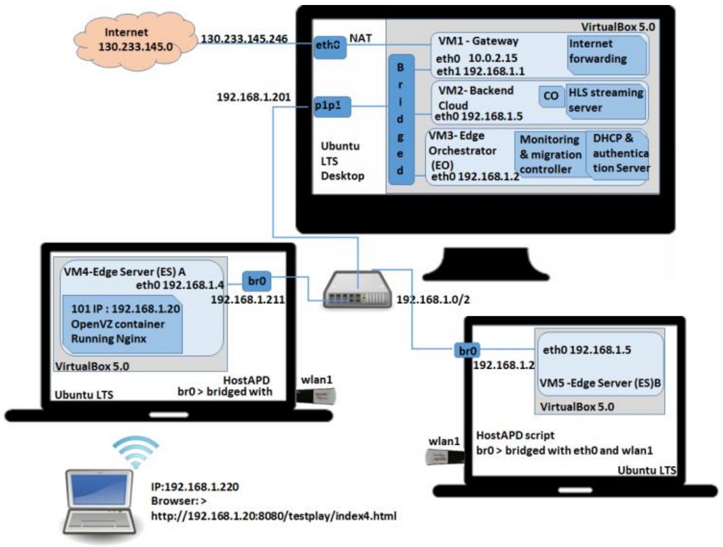

Fig. 2. Envisioned testbed-setup.

To perform the test, one container is instantiated in "Edge1" with all the features explained above. When a user (using 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 backend cloud. Simultaneously, it caches the relevant media contents and stores 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 multi-hop network. During the video payback, the user device is deliberately moved from the first edge towards 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 EO. Upon detecting the client's connectivity, the script in charge of automating the service migration gathers the client info (i.e., MACID), compares it with 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 the migration is initiated. Once completed, the container initialization is started in the new edge along-with post copy. Hereby, post-copy represents transfer of the residual amount of changes that occurred in the memory block during this small interval. As service auto-start is already enabled, so 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, 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. 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: (i) with streaming online mode – streaming being in use and client is watching the video; (ii) without streaming offline mode – streaming being in use whilst client is not watching the video and no changes in the memory blocks; (iii) blank container; and with two different types of ES. The migration latency is plotted considering local storage. The migration latency of a blank con-

[iii] https://ffmpeg.org/

tainer 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 when having higher RAM, leading to a slight decrease in the overall duration.

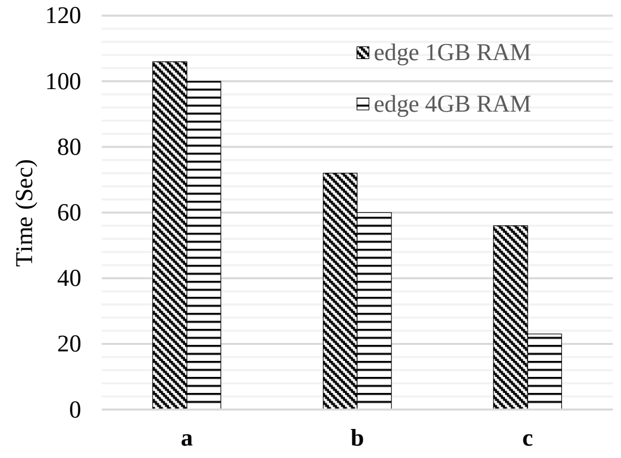

Fig. 3. Live migration time using local storage (a: with streaming online mode; b: without streaming offline mode; c: blank container).

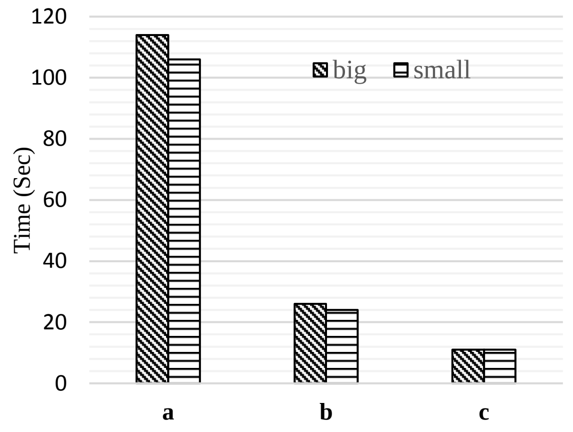

Fig. 4. Live-migration latency (a: local storage; b: shared sync storage; c: shared async storage).

Fig. 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 the migration duration. We clearly remark that the container size merely affects the migration latency. Besides, we observe '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 secs. In this last mode, the video experienced a single glitch of 1~2 sec. We explain this by the fact that data corruption took place during async mode hence resulting in reduced QoE [17]. The results obtained through this evaluation reveals that the storage type and memory capacity have a high impact on the migration latency.

V. Conclusion and future Research directions

In this paper, we proposed a framework which 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 its 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 the migration latency depends on different used techniques. 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 tradeoff 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.

ACKNOWLEDGEMENTS

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 and A. Ksentini, "An analytical model for Follow Me Cloud," in Proc. IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, Dec. 2013.

[2] A. Ahmed and E. Ahmed, "A Survey on Mobile Edge Computing," in Proc. IEEE 10th Intl. Conf. on Intelligent Systems and Control (ISCO 2016), Salerno, Italy, May 2016.

[3] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman,"Bringing the cloud to the edge," in Proc. IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, May 2014.

[4] T. Taleb, "Towards Carrier Cloud: Potential, Challenges, & Solutions," in IEEE Wireless Communications Mag., Vol. 21, No. 3, Jun. 2014. pp. 80-91.

[5] U. Shaukat, E. Ahmed, Z. Anwar, F. Xia, "Cloud Deployment in Local Wireless Area Networks, Motivation, Taxonomies, and Open Research Challenges", on Journal of Network and Computer Applications, Volume 62, February 2016, Pages 18–40.

[6] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, "Impact of Cloudlets on Interactive Mobile Cloud Applications," in Proc. IEEE 16th Int'l Conf. on Enterprise Distributed Object Computing Conference (EDOC), Beijing, China, Sep. 2012.

[7] J. Fajardo, I. Taboada, and F. Liberal, "Improving content delivery efficiency through multi-layer mobile edge adaptation", in IEEE Network Mag., Vol. 29, No. 6, Dec. 2015, pp. 40-46.

[8] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia Cloud Computing ," in IEEE Signal Processing Mag., Vol. 28 , No. 3, May 2011, pp. 59-69.

[9] Y. Jararweh, L.Tawalbeh, F. Ababneh, and F.Dosari, "Resource Efficient Mobile Computing Using Cloudlet Infrastructure," in Proc.

IEEE 9th Int'l. Conf. on Mobile Ad-hoc and Sensor Networks (MSN), Dalian, China, Dec. 2013.

[10] S. Wang, R. Urgaonkar, M. Zafer, and T. He, "Dynamic service migration in mobile edge-clouds," in Proc. IFIP Networking Conf., Toulouse, France, May 2015.

[11] T. Taleb, A. Ksentini, and A. Kobbane, "Lightweight Mobile Core Networks for Machine Type Communications," in IEEE Access Mag., Vol 2, Oct. 2014. pp.1128-1137.

[12] T. Taleb and A. Kunz, "Machine Type Communications in 3GPP Networks: Potential, Challenges, and Solutions," in IEEE Communications Mag., Vol. 50, No. 3, Mar. 2012.

[13] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "EASE: EPC as a Service to Ease Mobile Core Network," in IEEE Network Mag., Vol. 29, No. 2, Mar. 2015. pp.78 - 88.

[14] A. Ksentini, T. Taleb, and F. Messaoudi, "A LISP-based Implementation of Follow Me Cloud," in IEEE Access Mag., Vol 2, Oct. 2014. pp. 1340-1347.

[15] P. Frangoudis, L. Yala, A. Ksentini, and T. Taleb, "An architecture for on-demand service deployment over a telco CDN," in IEEE ICC'16, Kuala Lumpur, Malaysia, May 2016.

[16] T. Taleb, A. Ksentini, and R. Jantti, "Anything as a Service for 5G Mobile Systems," to appear in IEEE Network Magazine.

[17] S. Dutta, T. Taleb, and A. Ksentini, "QoE-aware Elasticity Support in Cloud-Native 5G Systems," in IEEE ICC'16, Kuala Lumpur, Malaysia, May 2016.