Personal Services Placement and Low-Latency Migration in Edge Computing Environments

R. Bruschi*, F. Davoli*‡, P.Lago*, C. Lombardo*‡, J. F. Pajo*‡,
‡ DITEN - University of Genoa - Genoa, Italy
* S3ITI Federated National Laboratory, Consorzio Nazionale Inter-Universitario per le Telecomunicazioni (CNIT), Genoa, Italy

Abstract— **Low latency and integration with edge computing facilities are two of the main enablers of the upcoming fifth-generation mobile network. The geographical distribution of virtualized compute and storage resources (e.g., NFVI PoPs) allows to deploy application/service instances/components closer to mobile subscribers, and to provide much higher performance levels in terms of latency. These new degrees of freedom should be suitably managed in order to place and to migrate application components along with subscribers' move. Given the mass-scale of subscribers and the non-negligible overhead to move application components over resource-constrained edge facilities, this management should be performed in a highly scalable and effective way. In this respect, this paper presents the design of a comprehensive edge computing framework, including an orchestration algorithm, able to place and to move services in few milliseconds (<25 ms). The algorithm provides scalability levels able to support services instantiated on a per-user basis (i.e., personal services) also in the presence of complex network and service environments. The performance evaluation has been carried out by considering different service chains within diverse software live migration technologies.**

Keywords—Service Placement, MEC, SDN, NFV.

I. Introduction

In the recent years, network technologies and architectures are facing a deep revolution in order to meet tomorrow's 5th generation (5G) mobile networks requirements, such as the support for extreme low latency vertical applications and services [1]. To this end, Multi-access Edge Computing (MEC) [2] has been widely accepted as one of the most viable solutions [3], since it enables to host layer-7 applications/services onto computing and storage facilities within Telco Operators' infrastructures, much closer to end users.

By exploiting softwarized infrastructures powered by Network Functions Virtualization (NFV) [4] and Software-Defined Networking (SDN) [5], MEC allows Telco Operators to support services characterized by real-time responsiveness and by a high degree of personalization of networking, billing and features [6] enabled by the knowledge of user location and the network data available within the Telco premises.

However, the deployment of layer-7 services closer to the mobile subscribers leads to additional management complexity that has to be suitably addressed to achieve effective and scalable operations. This complexity becomes manifest in the case of layer-7 personal cloud services/applications [7] for mobile end-users (i.e., modular applications composed of graphs of chained components, to be provided on single separated instances per each user).

Each time an instance of the personal cloud service is activated, or a user moves getting closer to other edge computing facilities, the Service Level Agreements (SLAs) of the application graph components should be checked to assure the requested proximity to end-users and performance levels. In case that these requirements are not satisfied, components should be (live) migrated towards closer facilities allowing to provide the requested resources and performance indicators.

In the absence of proper orchestration mechanisms, it can be clearly expected that huge amounts of data might be moved pointlessly among the edge computing facilities, and end-users might experience frequent performance decays due to the software migration overhead or placement of application components not satisfying the SLA requirements.

In this paper, we address the design of an edge computing framework to support personal cloud services for mobile users in an effective and scalable fashion. The proposed framework allows to enable multiple personal cloud services per user by attaching them to a "Personal Network" (i.e., a virtual network associated to a single subscriber and connecting services deployed in geographically distributed edge facilities). A Virtual Personal Gateway (VPG) is also exploited to logically terminate the Personal Network and to offer a management interface to end-users (e.g., to (un-)subscribe personal services). The VPG can be considered as an extension of the virtual Home Gateway targeted by the ETSI NFV Working Group [8] towards mobile network scenarios where it is crucial to minimize the overhead and the drawback due to migrations among edge facilities.

On top of these entities, the framework includes a very lightweight heuristic algorithm to orchestrate the autonomic placement of service components close to users, both when the service is subscribed and upon user move. To this end, the algorithm checks the required proximity to users declared in the SLA against the actual user and service component positions, it selects the components to be moved and, for each of them, it calculates the new placement by considering the availability of (computing) resources in all the candidate edge facilities.

Tests have been carried out to analyze the performance and the scalability of the orchestration algorithm by highlighting the aspects affecting the delays introduced by the system. In order to provide a comprehensive picture of the operation performance, specific tests have been conducted to further evaluate the delays in the presence of different software migration technologies, such as moving entire virtual machines or by using the application-driven migration of the user state. It is worth noting that this last approach is emerging along with the rise of the cloud-native paradigm [9].

The remaining of the paper is organized as follows. Section II provides a more detailed description of the proposed edge network framework. Section III describes the solutions designed for resource allocation and management, while results are reported in Section IV. Finally, conclusions are drawn in Section V.

II. Deployment of Personal Cloud Services in the Edge Network

In this work, we consider a MEC infrastructure able to support the deployment of personal services at the edge facilities of telecom operators, which are composed of computing and storage appliances interconnected by physical/virtual OpenFlow switches [10].

We take into account personal services designed in the form of a service chain. Such service chain is stored in a service template managed by a dedicated control plane module [11] and instantiated only upon user subscription. The service chain is composed of individual service components (ServiceApps) potentially deployed in different Points of Presence (PoPs). The appropriate placement and dynamic (re)allocation of ServiceApps, with respect to the user position or to other ServiceApps in the chain, is constrained by a proximity class, which represents the maximum allowed distance able to guarantee the fulfilment of the SLA.

The communication and information exchanged among ServiceApps of the same service chain are handled through Back-End Networks (BNs), which are isolated L2/L3 broadcast network domains, while a Personal Network (PN) is associated to each user and is employed to interconnect the user to the associated service chains with the same level of isolation and security available in the Local Area Network (LAN), independently of the actual user location. An example of this approach can be found in [7].

Figure 1 reports the edge network architecture deployment. PNs are realized by virtualizing typical network functions provided by the user's home gateway and by transferring them into software instances running in commodity computing facilities deployed in the telecom provider edge network. Details on the VPG design are reported in the next section.

A. *The Virtual Personal Gateway*

In order to virtualize the home gateway, we need to take into account the applications and functions needed to cover all of its functionalities. For the control plane, a Virtual Machine (VM) must be deployed for each user. The functionalities to be provided are the user web interface for the configuration and services subscription, and the capability to detect and communicate the changes in the configuration. OpenWRT [12] has been chosen to implement the user interface of the home gateway, as it is a Linux distribution natively created for this goal. Regarding the data plane, the main functionalities to be implemented are the NAT and the firewall; for their deployment, we have exploited a solution, described in Section III.B, which guarantees seamless migrations without any interruptions due to downtimes, as will be shown in the results in Section IV.

Figure 2 shows the deployment of the VPG inside an edge network node. Since the web interface is accessed sparingly and the requirements on its response times are not as restrictive, it is not migrated upon changes in the user location, which only involve the NAT and firewall functions, as described in detail in Section IV.

III. Network and Service Management in MEC

The MEC approach has the potential to increase flexibility and reduce end-to-end latency. However, the deployment of layer-7 services closer to the mobile subscribers has the clear drawback of added scalability issues: since the service chains must be migrated according to the actual user position while respecting the SLAs, huge amounts of data might be moved pointlessly in the absence of proper orchestration mechanisms.

In order to curb this issue, we have realized an orchestration algorithm for the autonomic (i.e., non-tenant-driven) placement of virtual service instances close to users on

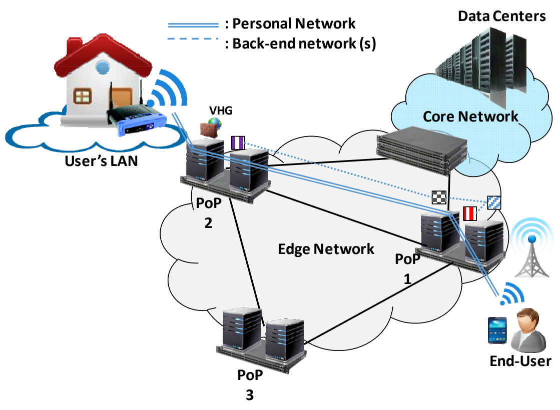


Figure 1. The deployment of personal services in the edge network.

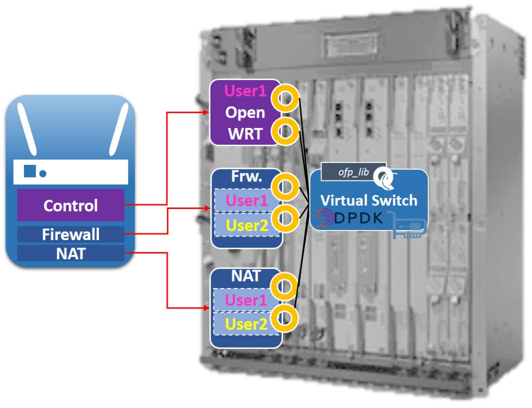


Figure 2. Deployment of the virtual home gateway in an edge node.

the move. Such algorithm is designed and implemented in two phases: first, it is applied to determine the initial placement, and afterwards for the dynamic reallocation of service components. The algorithm is described in Section III.A.

During the initial placement, the amount of resources required for a created task are estimated and provisioned to satisfy the predefined SLA, and the biggest challenge is to allocate the available resources fairly and to make sure no customer is under-provisioned. The second phase, which is the reallocation, can be triggered either by changes in the position of a mobile subscriber or in updated service requirements.

Further reduction in the overall end-to-end latency can be achieved by exploiting the different characteristics of the software migration technologies. This strategy will be analyzed in Section III.B.

A. Static and Dynamic Orchestration Algorithm

The objective of the orchestration algorithm is to quickly solve the service chain placement problem, presented in [13], to deploy service instances upon user's requests, and to dynamically update their position and related networks interconnecting them (e.g., Personal and Back-End Networks, see Section II) according to users' locations and events coming from the infrastructure.

The optimal service placement problem can be considered as the combination of two known theoretical problems, namely the Multi Resource Generalized Quadratic Assignment Problem (MRGQAP) [14] and the Unsplittable Flow Problem (UFP) [15], both proved to be in the NP-hard category. Since orchestration actions are triggered by the user's behavior or by the infrastructure-generated alarms, complex optimization procedures cannot guarantee the required reactiveness (latencies of the order of magnitude of few seconds at maximum) upon user's location change or congestion alarm.

For these reasons, the service placement problem has been decomposed into simpler sub-problems that can be solved through heuristic procedures. For example, an event-based orchestrator has been developed to apply different heuristic algorithms according to the required action. The pseudo code in Algorithm 1 gives a general idea of the implemented orchestration procedure.

```
1. EVENT: NEW_USER, INPUT: UserID, UserHomeLocation
    a. Create and instantiate a new user's Personal Network
2. EVENT: NEW_SERVICE_SUBSCRIPTION, INPUT: UserID, UserLocation, ServiceID
    a. OptDatacenter= Find optimal hosting datacenters for each ServiceApp composing the service
    b. For each ServiceApp, find optimal hosting server in OptDatacenters
3. EVENT: NEW_USER_LOCATION, INPUT: UserID, UserLocation
    a. If required, identify user's services to be migrated
    b. OptDatacenters = Find optimal hosting datacenters for each ServiceApp to be migrated
    c. Find the optimal hosting server for each ServiceApp to be migrated
    d. Update Personal and Back-End Networks topology
4. EVENT: SERVICE_UNSUBSCRIBE, INPUT: UserID, ServiceID
    a. TerminateService(ServiceID, UserID)
```

Algorithm 1: **High-level view of the orchestration algorithm.**

The orchestration actions are triggered by different events. When a new user subscribes, the static placement takes place: the first step consists in instantiating a new Personal Network in the infrastructure. In this initial phase, all the required network functions are instantiated by default in the closest available datacenter to the user's home and specific matching and action rules are configured in all the switches between the user's access network and the instances of network functions. Then, when users subscribe to new services (see line 2 in Algorithm 1), the orchestration algorithm finds the optimal hosting servers for all the ServiceApps composing the services and the optimal paths interconnecting them according to the current users' locations, the services proximity and resource/performance requirements.

Unfortunately, due to the NP-hard nature of this kind of problems, the optimal hosting server cannot be found in acceptable time without compromising the user experience. To avoid this, we simplify and decompose the problem in order to guarantee systems scalability and fast service deployment. In particular, the initial service placement upon user subscription is decomposed in three steps.

In the first step (see line 2.a in Algorithm 1) the optimal hosting datacenter is found for each ServiceApp composing the service. Given a target datacenter for each ServiceApp, the second step (see line 2.b in Algorithm 1) has to compute the destination hosting server; in this procedure, the simple heuristic takes into account the available resources in each server, trying to minimize the number of active servers, by guaranteeing a balanced use of resources. Finally, in the third step, all the required overlay networks between the end-user and related deployed services are created/updated.

Every time the user moves to another location (see line 3 in Algorithm 1) the dynamic procedure occurs and the orchestration algorithm checks if the proximity levels of the subscribed services are satisfied; if not, all the ServiceApps that must be migrated to another datacenter are identified and the procedures in line 2 are repeated to find an optimal location for these ServiceApps.

Finally, when a user unsubscribes from a service, the orchestration algorithm simply terminates the running service components and releases the occupied resources.

B. Considerations on Software Migration Technologies

In order to curb scalability issues, it is possible to exploit the characteristics offered by the different software migration technologies. Generally, layer-7 services and network functions can be designed based on three different technologies: on a physical machine (what is currently called "bare metal"), on a hosted container, or on a hypervisor. A bare metal implementation consists in exploiting programmable hardware, represented, for example, by a Field Programmable Gate Array

(FPGA) [16]. A container is a wrapper of a piece of software in a filesystem that contains all of the system tools and libraries needed to make it work correctly. Finally, a hypervisor is a VM manager [17] that can be placed on top of bare-metal or of an Operating System (OS). The hypervisor should not run applications natively; rather, its purpose is to virtualize the workloads into separate VMs to gain the flexibility and reliability of virtualization.

Furthermore, in order to more accurately satisfy the application requirements, in the design of the personal services an additional degree of virtualization has been proposed, which can be defined as multi-context process (MCP). In practice, in this environment, applications are in charge of directly providing the possibility of creating multiple instances working on different virtualized "contexts" (i.e., a sort of workspace), and are installed directly on hardware, bypassing both the host operating system and the hypervisor. This bypassing is made possible by using the DPDK libraries [18] and it provides a higher performance level with respect to traditional, kernel-based implementations.

As defined by ETSI [19], the most common required characteristics regarding an application's running environment are isolation, efficient use of physical resources, low to zero performance loss compared to the native OS environment, easy management of application running environments, and portability. Furthermore, resource management is considered crucial, in order to efficiently handle capacity and meet applications requirements. Tools to control resources are available in all types of running environments.

Considering these aspects, the choice of a software migration technology over the other must be made taking into account the desired trade-off between ease of migration and ease of access to the physical resources. In general, the migration of an execution environment includes transferring both the virtual image, which is the actual "body" of the instance, and the dynamic state, representing the data located inside the RAM memory and related to the instance to be migrated.

The main difference between VMs and MCPs resides in the size of their virtual image, with the latter sensibly smaller than the former. This characteristic can be exploited to minimize the migration time of a MCP. In fact, since the virtual image does not occupy much disk space, it can be deployed inside each server of the edge network, and migrations can be then performed by simply moving the dynamic state with a negligible overhead. A quantitative comparison is provided in the following section.

IV. Performance Evaluation

The following tests have the goal of analyzing the performance of the orchestration algorithm presented in Section III. In order to identify the aspects affecting the delays introduced by the system and their extent, Section IV.A reports the computation times obtained for the initial placement and migration of two generic service chains. It is worth noting that these results also include the placement and migration of the VPG described in Section II.A; for this reason, in order to compare how MCP performs with respect to traditional VMs in terms of latency, tests in Section IV.B focus only on the implementation of the VPG.

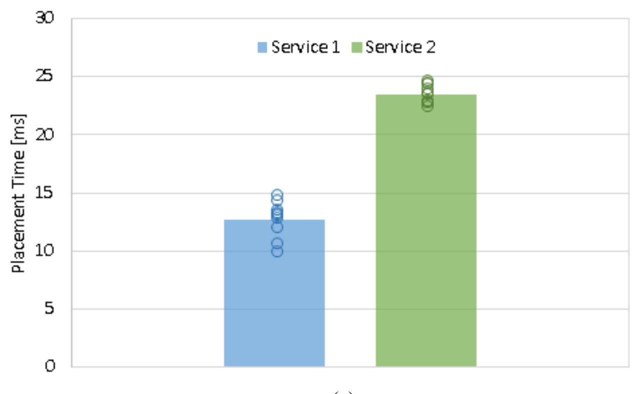
For both test cases, the orchestrator is running on a Linux server equipped with an Intel Xeon E5-2620 v4 2.10GHz processor [20]), and we evaluated a topology from the datasets available in [21].

A. Orchestration Algorithm Computation Times

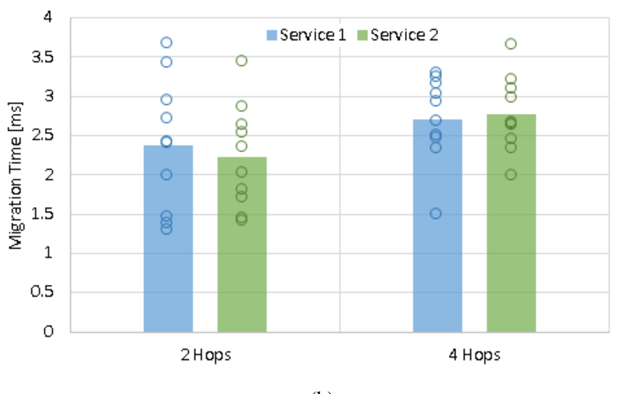
In the reported tests, two service chains have been considered: Service 1 is composed of 15 VMs, and Service 2 of 31 VMs. Computation times have been collected for the initial placement of the service chains and for their migration. For this latter event, we consider two cases: in the first case, a user changing position triggers a migration that takes two hops and requires moving only the VPG, while in the second case four hops are needed, which involves a new placement of the whole service chain.

Figure 3 reports the time needed to conclude (a) the initial service placement and (b) the move service operations, in the presence of the two service chains. In the graph, the columns show the average value, while the circles represent the single values obtained in multiple measures (the tests have been repeated at least ten times).

The initial placement of a service is more time consuming with respect to migrations, with both the number of VMs



(a)



(b)

Figure 3. Computation time, for two service chains, needed to (a) add a service and (b) perform a migration.

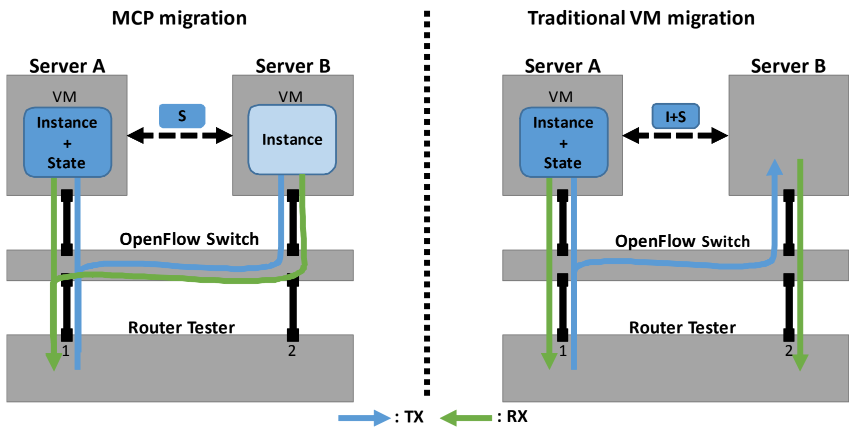


Figure 4. The test-bed used in the evaluation.

composing the chain and the distance triggering the service chain migration influencing the computation time. It is true that these figures are negligible for most applications, however migration overhead must also take into account the time needed to start up again the migrated VMs. Since this factor strictly depends on the service, we have further analyzed this aspect considering only the deployment of the VPG presented in Section II.A and results are reported in the following.

B. Comparison of VM and MCP Migration Times

In the following tests, the performance of a virtual network function deployed for Network Address Translation (NAT) purposes is analyzed in terms of service downtime. We compare the data obtained by deploying it as a VM, which is the typical implementation, and as a MCP according to the design presented in Section III.B. In both cases, the test-bed is the one shown in Figure 4 and is composed of two servers (A and B), and a router tester connected to an OpenFlow switch. In the MCP case, both servers host an instance of the NAT, but only one has the dynamic state enabling the network function. Traffic is sent by the router tester to the active NAT (blue line in Figure 4) that sends it back to the tester (green line). During this transmission, the NAT is initially located in Server A, and is then migrated to Server B.

In order to avoid service interruption and packet loss during the migration, the OpenFlow switching/routing rules are temporarily configured to duplicate packets destined to the NAT on the move onto both servers. In the MCP case, upon the fulfilment of the migration, the instance in Server A is disabled and the connectivity to and from Server A is removed, while only the connectivity to/from the new position is maintained. Initially, we wanted to compare the migration downtimes obtained in the two cases, and measured as follows:

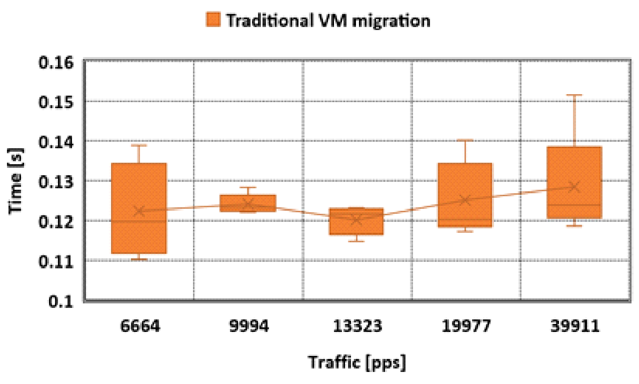
Downtime = Timestamp of 1st packet at Port 2 - Timestamp of last packet at Port 1

However, results showed that no downtime is experienced during the MCP migration; so, for the sake of comparison, we estimated the time ΔT in which the virtual image of the MCP is replicated (which implies additional costs in terms, for instance, of added overhead), by changing the rules in the OpenFlow switch (i.e. from Server B→Port 2 to Server B→Port 1) and counting the number of duplicated packets received at Port 1:

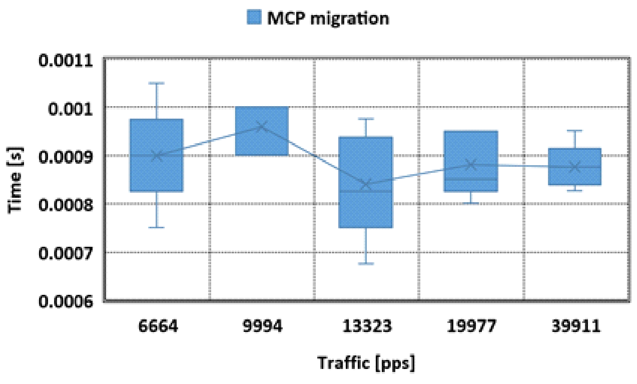
*ΔT = #duplicates * traffic rate*

In both cases, five runs were performed at varying packet rates, to demonstrate the reliability of the results. The packet size is fixed to 64 Bytes.

Figure 5 shows the distributions of the service downtimes and replication times at varying packet rates for the VM and MCP case, respectively, based on quartiles. In more detail, a box is drawn between the 1st and 3rd quartiles, a line along the



(a)



(b)

Figure 5. Measured downtime and replication time for different traffic loads in the case of (a) VM migration, and of (b) MCP migration.

median (2nd quartile), two additional lines indicating the minimums and maximums outside the 1st and 3rd quartiles, respectively, and an x to mark the mean values.

The mean values for the two test cases are compared, as well, considering the line in Figure 5. Results show that the estimates are stable at varying packet rates. Moreover, the migration downtime obtained for the VM migration case is up to over two orders of magnitude greater than the ΔT obtained for MCP migration.

V. CONCLUSIONS AND FUTURE WORK

The possibility of hosting layer-7 applications/services into computing and storage facilities within Telco Operators' infrastructures, much closer to end users, as provided by the Mobile Edge Computing paradigm with the support of Network Functions Virtualization, is widely recognized as a fundamental aspect to fulfil the upcoming 5G mobile network requirements in terms of support for extreme low latency vertical applications and services.

This paper has considered the design of a comprehensive edge computing framework allowing to support services instantiated on a per-user basis also in the presence of complex network and service environments. This is achieved thanks to a very lightweight heuristic algorithm to orchestrate the autonomic placement of service components close to users, and by exploiting the degrees of freedom offered by different software migration technologies.

A thorough performance evaluation has been carried out by testing real application chains, deployed according to diverse software live migration technologies, in the presence of complex network and service environments. Test results obtained on a real wide-area topology show that the proposed framework allows to orchestrate the autonomic placement and migration of services instantiated on a per-user basis in few milliseconds (<25 ms).

Future work that will be carried out in the 5G Public-Private Partnership (5G-PPP) Innovation Action MATILDA [22] regards the design of an algorithm to evaluate the cost of migrating the image against storing it in multiple servers.

ACKNOWLEDGMENT

This work is supported by the European Commission through the MATILDA Project (Contract H2020-761898). Special thanks go to the Research and Innovation Action INPUT.

REFERENCES

[1] "5G Vision - The 5G Infrastructure Public Private Partnership: the next generation of communication networks and services", URL: https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf.

[2] ETSI GS MEC 002 2016, "Mobile Edge Computing (MEC); Technical Requirements", URL: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf

[3] 5G PPP Architecture Working Group, "View on 5G Architecture (Version 2.0)," URL: https://5g-ppp.eu/wp-content/uploads/2017/07/5G-PPP-5G-Architecture-White-Paper-2-Summer-2017_For-Public-Consultation.pdf.

[4] M. Chiosi et al., "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges and Call For Action," In Proceedings of the SDN and OpenFlow World Congress, Darmstadt, Germany. ETSI White Paper. URL: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[5] "Software-Defined Networking: The New Norm for Networks, Open Networking Foundation (ONF)," White Paper, Apr. 2012.

[6] "The Tactile Internet", ITU-T Technology Watch Report, August 2014.

[7] R. Bruschi, F. Davoli, P. Lago, A. Lombardo, C. Lombardo, C. Rametta, G. Schembra. "An SDN/NFV Platform for Personal Cloud Services," IEEE Transaction on Network and Service Management (TNSM), vol. 14, no. 4, Dec. 2017, pp. 1143 - 1156.

[8] ETSI GS NFV 001 2013, "Network Functions Virtualization (NFV): Use Cases", URL: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf.

[9] "Evolving the Mobile Core to Being Cloud Native", Cisco White Paper, URL: https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/ultra-services-platform/white-paper-c11-739215.pdf.

[10] Open Network Foundation (ONF), "OpenFlow Switch Specification," Version 1.4.0, URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[11] R. Bruschi, G. Genovese, A. Iera, P. Lago, G. Lamanna, C. Lombardo, S. Mangialardi, "OpenStack Extension for Fog-Powered Personal Services Deployment", Proc. of the First International Workshop on Softwarized Infrastructures for 5G and Fog Computing (Soft5 2017), Genoa, Italy.

[12] The OpenWRT Website, URL: http://openwrt.org

[13] F. Wamser, C. Lombardo, C. Vassilakis, L. Dinh-Xua1, P. Lago, R. Bruschi, P. Tran-Gia. "Orchestration and Monitoring in Fog Computing for Personal Edge Cloud Service Support", Proc. of the IEEE International Symposium on Local and Metropolitan Area Networks IEEE (IEEE LANMAN 2018), Washington DC, USA, 25-28 June 2018.

[14] M. Yagiura et al., "Path Relinking Approach for the Multi-Resource Generalized Quadratic Assignment Problem", Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. (2007).

[15] Y. Dinitz, N. Garg, M. X. Goemans. "On the Single-Source Unsplittable Flow Problem". Springer Combinatorica, vol.19, no. 1, pp 17–41, January 1999.

[16] NetFPGA 10G OpenFlow switch, URL: https://github.com/NetFPGA/NetFPGA-public/wiki/.

[17] Understanding Full Virtualization, Paravirtualization, and Hardware Assist. White paper, URL: https://www.vmware.com/techpapers/2007/understanding-full-virtualization-paravirtualizat-1008.html .

[18] The Intel Data-Plane Development Kit, URL: http://www.dpdk.org

[19] ETSI GS NFV-SWA 001. 2014, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture", URL: http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_nfv-swa001v010101p.pdf.

[20] "Intel Xeon Processor E5-2620 v4 (20M Cache, 2.10 GHz)." [Online]. Available: https://ark.intel.com/products/92986/Intel-Xeon-Processor-E5-2620-v4-20M-Cache-2 10-GHz

[21] "The Internet Topology Zoo." [Online]. Available: http://topologyzoo.org.

[22] MATILDA - A Holistic, Innovative Framework for Design, Development and Orchestration of 5G-ready Applications and Network Services over Sliced Programmable Infrastructure, URL: http://www.matilda-5g.eu