Follow-Me Cloud: When Cloud Services Follow Mobile Users

Tarik Taleb, Adlen Ksentini, and Pantelis A. Frangoudis

Abstract—The trend towards the cloudification of the 3GPP LTE mobile network architecture and the emergence of federated cloud infrastructures call for alternative service delivery strategies for improved user experience and efficient resource utilization. We propose *Follow-Me Cloud (FMC)*, a design tailored to this environment, but with a broader applicability, which allows mobile users to always be connected via the optimal data anchor and mobility gateways, while cloud-based services *follow* them and are delivered via the optimal service point inside the cloud infrastructure. Follow-Me Cloud applies a Markov-decision-process-based algorithm for cost-effective performance-optimized service migration decisions, while two alternative schemes to ensure service continuity and disruption-free operation are proposed, based on either software defined networking technologies or the locator/identifier separation protocol. Numerical results from our analytic model for follow-me cloud, as well as testbed experiments with the two alternative follow-me cloud implementations we have developed, demonstrate quantitatively and qualitatively the advantages it can bring about.

Index Terms—Cloud computing, cellular mobile networks, 3GPP LTE, Markov decision process, software-defined networking, network function virtualization

1 Introduction

To cope with the explosive growth in mobile data traffic [1], which challenges both their core and radio networks, mobile operators are pushing towards a new architectural solutions to decentralize the user plane of their networks. Such approaches involve moving data anchor gateways towards the edge of the network and carefully serving IP traffic via selected points close to their Radio Access Network (RAN) nodes by mobile data offloading techniques [2]. At the same time, computation offloading over heterogeneous wireless network infrastructures also attracts attention, in view of the availability of cloud computing resources [3].

At the service delivery end, the success of cloud computing has led content/service providers to consider deploying more regional data centers (DCs). Furthermore, the dependence of content providers (CPs) and Internet Service Providers (ISPs) on one another for efficient content/service delivery and disruption-free network operation in view of dynamic shifts in traffic demands creates content provider-internet service providers cooperation incentives [4] for joint deployment of network-aware content and service delivery infrastructures, and cloud computing technologies are considered for their implementation.

- *T. Taleb is with the Sejong University and Aalto University, Espoo FI-00076, Finland. E-mail: talebtarik@ieee.org.*
- *A. Ksentini is with the Department of Mobile Communications, EURECOM, Sophia-Antipolis, France. E-mail: adlen.ksentini@eurecom.fr.*
- *P.A. Frangoudis is with the IRISA/University of Rennes 1, Rennes, France. E-mail: pantelis.frangoudis@irisa.fr.*

Manuscript received 20 May 2015; revised 26 Oct. 2015; accepted 23 Jan. 2016. Date of publication 4 Feb. 2016; date of current version 5 June 2019.
Recommended for acceptance by C.H. Foh, S.N. Srirama, E. Benkhelifa, B. Kantarci, P. Chatzimisios, and J. Wu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2016.2525987

Importantly, to more efficiently address the needs of mobile users in terms of geographical coverage and proximity of data centers to them, a new means of cooperative service deployment has emerged in the form of a networked *federated cloud* [5]. This involves allocating virtual resources on a number of data centers dispersed over a specific geographical area, over the infrastructure of potentially heterogeneous federated cloud providers in a transparent manner.

The availability of regional resources and the flexibility of the virtualization technologies upon which federated clouds are built are particularly important to support the modern trend of *cloudifying* the mobile network infrastructure and offering mobile services in an elastic manner, following user demand and presence. In the context of the third Generation Partnership Project (3GPP) long term evolution/evolved packet system (LTE/EPS) [6], a decentralized mobile network architecture would include core network gateways such as packed data network gateways (PGWs) and serving gateways (SGWs) operating as *virtualized network function (VNF)* instances on the cloud [7], [8], and not necessarily running on top of specialized dedicated hardware.

At the same time, it is important to ensure that users connected to the mobile core network through a third Generation Partnership Project base station (eNodeB) or using non-third Generation Partnership Project access, such as Wi-Fi, enjoy acceptable quality of experience (QoE) by always guaranteeing optimal end-to-end connectivity for the services offered over the federated cloud, during the entire course of service consumption. Here lies the main challenge we address in this work: While user connectivity to the mobile data anchor gateway is always optimal, this is not necessarily the case for the end-to-end mobile service delivery, since a user on the move may keep receiving the service from a distant (suboptimal) data center after moving to different physical locations.

To answer to this challenge, we introduced the concept of the follow-me cloud (FMC) [9], a design tailored to an interoperating decentralized mobile network/federated cloud architecture. Follow-me cloud allows not only the content, but also the service itself, to follow a mobile user while moving, ensuring that the latter is always connected to the optimal data anchor and mobility gateways, at the same time accessing a cloud-based service from the optimal data center, in terms either of geographical/network-topological proximity or any other service- or network-level metric, such as load, service delay, etc.

To realize the follow-me cloud vision, service continuity, and sophisticated schemes for service migration decisions across data centers are critical. In this article, we present a complete framework defining follow-me cloud from architectural, algorithmic, and implementation perspectives. We explore alternative schemes for ensuring service continuity, which either build on the locator-identifier separation protocol (LISP) [10], [11] or on using software-defined networking (SDN) technologies. We further propose a Markov decision process (MDP)-based algorithm [12] for optimally performing service migration decisions, taking into account user mobility information, and addressing the tradeoff between migration cost and user experience. Our testbed implementation of the proposed architectural alternatives, coupled with an analytic performance evaluation of our system, serve to demonstrate the feasibility and advantages of follow-me cloud, and shed light on the practical aspects of its deployment.

The remainder of this article is structured as follows. In Section 2 we provide an overview of related work. We present the follow-me cloud concept, entities, and high-level functionality in Section 3. Section 4 introduces an analytic model which captures the tradeoff between the benefit and cost due to service migration, and Section 5 presents a Markov decision process scheme building on this model. We describe a locator-identifier separation protocol-based and an software-defined networking-based implementation of follow-me cloud in Section 6, and present analytic and testbed-based performance results in Section 7, before we conclude the article in Section 8.

2 Related Work

2.1 Service Continuity for Mobile Users

In the mobile networking context that we position our work, a major and well-studied challenge is maintaining service continuity during user and, importantly, service mobility. An approach to this problem is decoupling session and location identifiers. A protocol which makes this separation explicit is locator-identifier separation protocol (see Section 2.3), and we apply it in this work.

Nordström et al. [13], on the other hand, present Serval, a networking stack which includes a new service access layer to cater for user and service mobility, providing identifier/location separation. It makes use of service identifiers, which would however require modifications to legacy applications to support the proposed functionality.

Other research works have considered the use of Openflow to hide, through its rules, any changes to IP addresses. Openflow-based solutions often face scalability challenges wrt. the number of flows, number of rules, flow setup rate, bandwidth of the control channel, etc. To reduce the number of control packets, Devoflow [14] moves some of the flow creation work from controllers to switches. Bifulco et al. [15] propose to distribute control plane functions, in order to enhance system scalability, which is an approach that our software-defined networking-based design (see Section 6.1) could follow.

2.2 Service Migration

In a federated cloud context [5], where geographically distributed data centers are connected into a common resource pool, a cloud management procedure for directing service requests to the optimal data centers, satisfying resource, cost, and quality constraints is necessary. If the respective criteria/constraints are not covered, services may need to be migrated across data centers. Malet and Pietzuch [16] propose a cloud management middleware for migrating part of a user's service (represented by a set of virtual machines) between data center sites in response to data center workload variations and in order to move application components geographically closer to users. Agarwal et al. [17] present Volley, an automatic service placement scheme for geographically distributed data centers based on iterative optimization algorithms, which performs service migrations when detecting that data center capacity or user location change. Alicherry and Lakshman [18] propose a data center selection algorithm for placing a virtual machine (VM), modeling the problem as a subgraph selection one, while Steiner et al. [19] demonstrate how services can be placed based on information retrieved from an Application-Layer Traffic Optimization (ALTO) server. The above works mainly focus on the virtual machine migration process rather than on virtual machine mobility management.

Other works [20], [21] integrate IP mobility management directly into the hypervisor, which interacts with a virtual machine before and after its migration to update IP addresses in the virtual machine's routing table, or, as in the work of Li et. al [22], invokes mobile IP functionality each time a virtual machine is created, destroyed or migrated. These solutions perform live virtual machine migration at the expense of potentially long downtimes. Raad et al. [23], on the other hand, achieve subsecond downtimes using a modified version of locator-identifier separation protocol for rapid traffic redirection. Contrary to our approach (see Section 6.2.2), their scheme also requires modifications to the hypervisor, raising deployment issues.

2.3 Locator-Identifier Separation Protocol

With location and identity traditionally coupled, IP mobility becomes a challenging task. To this end, locator-identifier separation protocol [10] separates them using routing locators (RLOCs) and endpoint identifiers (EIDs). locator-identifier separation protocol does not impose any constraints on the Endpoint Identifier and routing locators identifier format; IP addresses are typically used. Routing locators are needed to forward packets to/from the internet, while endpoint identifiers are local to an IP subnet. At the data plane level, locator-identifier separation protocol maps the endpoint identifier address to an routing locator, and encapsulates the packets into other IP packets before forwarding them through the IP transit. Usually, a locator-identifier separation protocol site is managed by at least one tunneling locator-identifier separation protocol router (xTR), having two functionalities: IP packet encapsulation (packet received by a terminal; ingress functionality, or ITR) and decapsulation (packet received by the network; egress functionality, or ETR).

To guarantee endpoint identifier reachability, the locator-identifier separation protocol mapping system includes a map resolver (MR), a Map Server (MS), and a cache table at each tunneling router. When a station has a packet to transmit, the endpoint identifier of the remote station is used in the destination address. Once reaching the ingress part of tunneling router (ITR), the latter encapsulates the transmitted packet by adding three headers (locator-identifier separation protocol, UDP, and IP) and fixing the fields "Source Routing Locator" and "Destination Routing Locator" of the locator-identifier separation protocol header to the source and destination tunneling router routing locators, respectively. The mapping between the endpoint identifiers and the corresponding destination tunneling router routing locator is first looked up in the local cache. If lookup fails, a map_request message is sent to the Map Resolver, which responds with a Map_Reply if the mapping is found. Otherwise, it redirects this request to the Map Server. The map server searches in its local database to find an tunneling router that would correspond to this endpoint identifier, and replies with a map_reply if it exists. Otherwise, it replies with a negative_map_reply. Note that the map server receives map_register messages from ETRs and registers endpoint identifier-to-routing locator mappings in the mapping database.

Compared to mobile IP, locator-identifier separation protocol avoids triangular routing thanks to decoupling locations and identifiers. A station can move to another location without changing its endpoint identifier ; only the routing locator has to be updated at the map server/map resolver. Furthermore, with few modifications, locator-identifier separation protocol can help achieve short virtual machine migration downtimes [23].

2.4 Our Own Prior Work

This article extends, generalizes, and refines our follow-me cloud concept [9], presenting an evolved design targeting generic decentralized mobile network architectures and making heavier use of NFV technologies, bringing the service closer to the end user. We further complement our locator-identifier separation protocol-based implementation of this scheme [11], which we have updated to match our evolved follow-me cloud design, with an software-defined networking-based one. Finally, on the theoretical front, we extend our service migration decision algorithm [12] to also capture 2D mobility scenarios; our algorithm builds on our analytic model presented in [24], included here for completeness.

3 Follow-Me Cloud Concept

3.1 High-Level Design

In this section, we present the concept and main functionality of our follow-me cloud design for optimized, disruption-free cloud-based services for mobile users. Our high-level architecture is shown in Fig. 1. The two main components of our scheme are the follow-me cloud controller (FMCC) and the DC/GW mapping entity. These can either be two independent architectural components, two

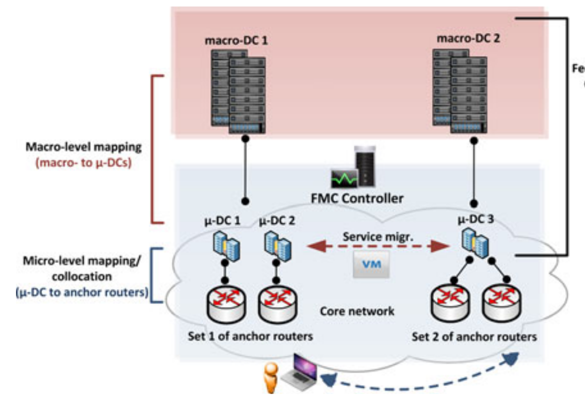


Fig. 1. Follow-me cloud high-level architecture in a federated cloud and distributed mobile network environment.

functional entities collocated with existing nodes, or run as a software on any data center of the underlying cloud.

Follow-me cloud was designed with the third generation partnership project long term evolution/evolved packet system architecture in mind, but is generic and can be applied to other decentralized mobile network access schemes. We assume that data centers are mapped to a set of data anchor routers/gateways. In an long term evolution context, these routers are packed data network gateways, while, for users roaming across federated Wi-Fi hotspots, such as the Fon network [25], the data anchor router can be the access router of the internet service provider to which a public Wi-Fi access point is connected. Depending on the mobile access architecture, other options are possible.

The data center-gateway mapping is based on some criterion, such as location or hop-count distance. This mapping may be static or dynamic. In the latter case, topology information can be exchanged between the follow-me cloud service provider and the mobile network operator (MNO). Alternatively, an mobile network operator function could be in charge of updating the follow-me cloud service provided with such information either in a reactive or a proactive manner. Note, furthermore, that our design includes a single follow-me cloud controller for managing distributed data center instances, but this does not preclude a decentralized self-organized implementation for distributed data center coordination.

Taking advantage of virtualization technologies at the data anchor end, our design extends our first version of the follow-me cloud architecture [9] with the capability to serve users directly from the data anchor router, bringing services closer to the user end. We distinguish between two types of data centers. At a macroscopic level, there are the core (macro) data centers, which can be considered as data origin servers. At a microscopic level, caches implemented at the data anchor gateways operate as microdata centers to serve mobile users more efficiently. The focus of the macrodata center level is persistent service (virtual machine) storage and service instantiation. The follow-me cloud functionality is implemented both at user equipments (UEs) and at the microdata center level and is responsible for service identification and migration procedures. As shown in Fig. 1, a two-level mapping takes place: Macrodata Centers are mapped to a group of microdata centers, each of which is in turn mapped to one or more data anchor routers. Note that these data anchors can be implemented as virtualized network functions hosted in the federated cloud (e.g., collocated with their corresponding microdata center).

Our design allows for various strategies for selecting which virtual machines to cache at microdata centers, as well as for deciding whether a service component will be migrated or replicated at another data anchor following user mobility. However, such strategies are outside the scope of this work; for simplicity and presentation clarity, we assume that a service is deployed at the data anchor router where the user is attached to upon service initiation, and is migrated (i.e., no replication takes place) as a user moves.

From this point on, unless otherwise noted, the term data center will refer to a microdata center (μ-data center).

3.2 Service Migration Process

With the IP address change which takes place when a user equipment changes its data anchor router (e.g., packed data network gateway relocation in a third generation partnership project long term evolution/evolved packet system mobile network architecture), there is a potential need for an follow-me cloud service migration. This change can be detected by the serving microdata center. Whether service migration is worthwhile depends on the service type and requirements (e.g., an ongoing video service with strict QoS requirements may be migrated; a delay-sensitive measurement task for an emergency warning Machine Type Communications service must always be migrated to the optimal data center), content size (e.g., the movie a user is watching is about to finish at the time of packed data network gateway relocation; the user equipment follow-me cloud application layer decides not to initiate service migration), and/or user class. The migration decision relies on several potentially conflicting criteria related with user expectations about the service (QoS/QoE, cost) and network/cloud provider policies (load balancing, maximizing data center resource utilization, microdata center capacity, etc).

Once the user equipment or the current microdata center is consider appropriate to migrate the service, the follow-me cloud plugin available at the microdata center may request the follow-me cloud controller to select the optimal microdata center to initiate the service migration to. As a service may consist of multiple cooperating components, potentially residing at different locations, the decision has to be made indicating whether the service has to be fully or partially migrated, while considering the service migration cost, e.g., the cost associated with the initiation/replication of a new virtual machine at the target microdata center, with the release of resources at the source data center, or with the bandwidth consumption due to traffic being exchanged between the data centers and/or the follow-me cloud controller. An estimate of these costs shall be compared to the benefits for the (federated) cloud in terms of traffic distribution, but also to those for end users in terms of QoE.

4 Analytic Model for Follow-Me Cloud

In this section, we propose an analytic model to establish the relationship and the relevant tradeoff between follow-me cloud service migration cost and benefits in terms of user experience. This model provides insights upon which we base a Markov Decision Process algorithm to derive optimal migration policies (see Section 5).

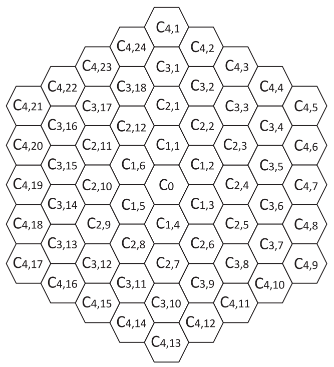


Fig. 2. Typical (3GPP) cellular network.

4.1 Markov-Based System Model

We use Markovian models to represent our system, aiming to be able to derive the user equipment position wrt. the serving data center and predict system evolution. Here, we focus our discussion on a third generation partnership project long term evolution mobile network environment. A third generation partnership project network is typically divided into hexagonal cells (Fig. 2). For the sake of simplicity, we assume that microdata centers and data anchor routers (packed data network gateways) are collocated with eNodeBs. In a real implementation, a microdata center could be mapped to a set of packed data network gateways, which are in turn mapped to a pool of eNodeBs.

We consider a random walk mobility model, where a user equipment can visit any of the six neighboring cells with probability $p = 1/6$. The residence time of a user equipment in each cell follows an exponential distribution with mean $1/\mu$. Fig. 2 shows a service area with $k = 5$ rings of cells. Service migration and data anchor gateway relocation are triggered for a user equipment when its location is k hops away from the serving data center (assumed to be collocated with eNodeBs). Let $X(t)$ denote the distance of the user equipment to the serving data center (in number of hops) at the time instant t. The system $\{X(t), t \geq 0\}$ forms a Continuous-Time Markov Chain (CTMC), with the state space $\{C_{(i,j)} | 0 \leq i \leq (k-1), 1 \leq j \leq 6i\}$.

This chain faces a state space explosion problem, especially if k is high. To address this problem, we reduce the state space by aggregating states that show the same behavior [24], [26], [27], to obtain a new chain $A(t)$ with a lower number of states. In Fig. 2, we see that user equipments in ring 0 can move to any neighboring cell with the same probability. user equipments in ring 1 come back to the cell which hosts the serving data center with probability p, stay in the same ring (same distance from the serving data center) with probability $2p$, and move to ring 2, increasing the distance from the data center, with probability $3p$. Consequently, all states of ring 1 can be aggregated into one state. Regarding ring 2, there are two groups of cells: (i) Cells neighboring three ring-3, two ring-2, and one ring-1 cells, and (ii) cells having two neighbors at each of the three rings. Depending on the ring-2 cell the user equipment is located, e.g., it may either have $3p$ or $2p$ probability to move to ring 3. Therefore, we obtain two aggregated states: State C_2 aggregates states$\{C_{2,1}, C_{2,3}, C_{2,5}, C_{2,7}, C_{2,9}, C_{2,11}\}$ and state $C_2^{(1)}$ aggregates states $\{C_{2,2}, C_{2,4}, C_{2,6}, C_{2,8}, C_{2,10}, C_{2,12}\}$. The

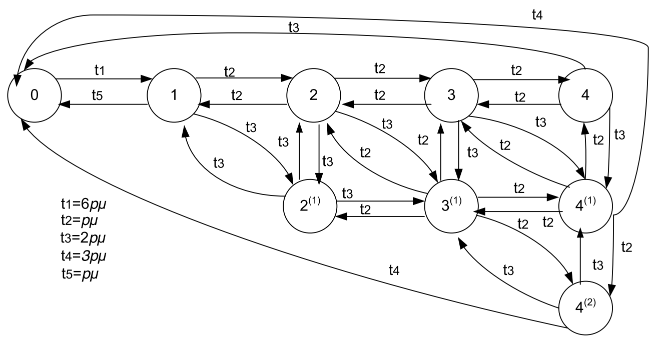


Fig. 3. Markov chain for $k = 5$.

same rationale is applied to obtain aggregated states C_i and $C_i^{(m)}$ for any ring i, where $1 \leq m \leq \lceil \frac{i-1}{2} \rceil$ is the identifier of an aggregated state within a ring.

As proven by Langar et al. [26], the new aggregated chain $A(t)$, derived from the initial Markov chain $X(t)$, is also Markovian. Fig. 3 shows the transition diagram of the aggregated Markov chain when the service migration is triggered when the user equipment is $k = 5$ hops away from the serving data center. Based on this figure, we can derive the steady state probabilities of the aggregated states. For simplicity, each aggregated state of ring i in Fig. 3 is labeled using the ring number and the superscript used to identify different aggregate states of the same ring, if necessary. The balance equations (Eq. (1)-(6)) to solve the system follow[1]:

$$\begin{cases} \pi_0 = \frac{1}{6}\pi_1 + \frac{1}{2}\pi_{k-1} + \frac{1}{3}\sum_{j=1}^{\lceil \frac{k-2}{2} \rceil} \pi_{k-1}^{(j)} \\ \pi_1 = \pi_0 + \frac{1}{3}\pi_1 + \frac{1}{6}\pi_2 + \frac{1}{3}\pi_2^{(1)} \\ \pi_2 = \frac{1}{6}\pi_1 + \frac{1}{6}\pi_3 + \frac{1}{3}\pi_2^{(1)} + \frac{1}{6}\pi_3^{(1)} \\ \pi_{k-1} = \frac{1}{6}\pi_{k-2} + \frac{1}{6}\pi_{k-1}^{(1)} \\ \pi_i = \frac{1}{6}\pi_{i-1} + \frac{1}{6}\pi_{i+1} + \frac{1}{6}\pi_{i-1}^{(1)} + \frac{1}{6}\pi_{i+1}^{(1)}, \forall 3 \leq i \leq k-2, \end{cases} \tag{1}$$

$$\begin{cases} \pi_2^{(1)} = \frac{1}{3}\pi_1 + \frac{1}{3}\pi_2 + \frac{1}{6}\pi_3^{(1)} \\ \pi_3^{(1)} = \frac{1}{3}\pi_2 + \frac{1}{3}\pi_3 + \frac{1}{3}\pi_2^{(1)} + \frac{1}{6}\pi_3^{(1)} + \frac{1}{6}\pi_4^{(1)} + \frac{1}{3}\pi_4^{(2)} \\ \pi_4^{(1)} = \frac{1}{3}\pi_3 + \frac{1}{3}\pi_4 + \frac{1}{6}\pi_3^{(1)} + \frac{1}{6}\pi_5^{(1)} + \frac{1}{3}\pi_4^{(2)} + \frac{1}{6}\pi_5^{(2)} \\ \pi_{k-1}^{(1)} = \frac{1}{3}\pi_{k-2} + \frac{1}{3}\pi_{k-1} + \frac{1}{6}\pi_{k-2}^{(1)} + \frac{1}{6}\pi_{k-1}^{(2)} \\ \pi_i^{(1)} = \frac{1}{3}\pi_{i-1} + \frac{1}{3}\pi_i + \frac{1}{6}\pi_{i-1}^{(1)} + \frac{1}{6}\pi_{i+1}^{(1)} + \frac{1}{6}\pi_i^{(2)} + \frac{1}{6}\pi_{i+1}^{(2)} \\ \qquad \forall 5 \leq i \leq k-2, \end{cases} \tag{2}$$

$$\begin{cases} \pi_i^{(j)} = \frac{1}{6}\pi_i^{(j-1)} + \frac{b_1}{6}\pi_i^{(j+1)} + \frac{1}{6}\pi_{i-1}^{(j-1)} + \frac{1}{6}\pi_{i-1}^{(j)} \\ + \frac{b_2}{6}\pi_{i+1}^{(j)} + \frac{b_2}{6}\pi_{i+1}^{(j+1)} \\ \qquad \forall 6 < i < k-1 \text{ and } 2 \leq j \leq \lceil \frac{i-1}{2} \rceil - 1, \end{cases} \tag{3}$$

where

$$b_1 = \begin{cases} 1 & \text{if} \quad i \text{ is odd} \\ 1 & \text{if} \quad i \text{ is even and } 2 \leq j \leq \lceil \frac{i-1}{2} \rceil - 2 \\ 2 & \text{if} \quad i \text{ is even and } j = \lceil \frac{i-1}{2} \rceil - 1, \end{cases}$$

and

$$b_2 = \begin{cases} 0 & \text{if} \quad 6 \leq i \leq k-2 \\ 1 & \text{if} \quad i = k-1 \end{cases}$$

1. For details on the state aggregation algorithm and more insight on the derivation of the balance equations for the reduced continuous-time Markov chain , the reader is referred to the work of Langar et al. [26].

$$\begin{cases} \pi_{2l}^{(l)} = \frac{1}{6}\pi_{2l}^{(l-1)} + \frac{1}{6}\pi_{2l-1}^{(l-1)} + \frac{c_1}{6}\pi_{2l+1}^{(l)}, \ \forall 2 \leq l \leq \lceil \frac{k-1}{2} \rceil, \end{cases} \tag{4}$$

where

$$c_1 = \begin{cases} 0 & \text{if} \quad l = \frac{k-1}{2} \\ 1 & \text{otherwise} \end{cases}$$

$$\begin{cases} \pi_{2l+1}^{(l)} = \frac{1}{6}\pi_{2l+1}^{(l-1)} + \frac{1}{6}\pi_{2l+1}^{(l)} + \frac{1}{6}\pi_{2l}^{(l-1)} + \frac{1}{6}\pi_{2l}^{(l)} \\ + \frac{c_2}{6}\pi_{2l+2}^{(l)} + \frac{c_2}{6}\pi_{2l+2}^{(l+1)}, \\ \qquad \forall 2 \leq l \leq \frac{k-2}{2}, \end{cases} \tag{5}$$

where

$$c_2 = \begin{cases} 0 & \text{if} \quad l = \frac{k-2}{2} \\ 1 & \text{otherwise} \end{cases}$$

$$\sum_{i=0}^{k-1} \pi_i + \sum_{i=2}^{k-1} \sum_{m=1}^{\lceil \frac{i-1}{2} \rceil} \pi_i^{(m)} = 1. \tag{6}$$

4.2 Average User Equipment-Data Center Distance and the Probability to be Connected to the Optimal Data Center

Let $E[Dist]$ denote the average distance of a user equipment from the serving data center. $E[Dist]$ depends on the value of k, and the distance (number of hops) of the user equipment from the data anchor router collocated with the serving data center. Recall that a user equipment remains connected to this anchor and all data are consequently routed through the latter until service migration is triggered. Therefore, the average distance is expressed as

$$E[Dist] = \sum_{i=1}^{k-1} i\pi_i + \sum_{i=2}^{k-1} \sum_{j=1}^{\lceil \frac{k-2}{2} \rceil} i\pi_i^{(j)}. \tag{7}$$

The probability that the user equipment is connected to the optimal data center during the system's lifetime is π_0.

4.3 Average End-to-End Delay from the Serving Data Center

We define the end-to-end (e2e) delay as the delay for a user equipment to receive data packets from the serving data center. Similar to $E[Dist]$, the e2e delay depends on the user equipment distance (number of hops) to the data anchor router connecting to the data center. The average e2e delay is denoted by $E[D]$ and is given by

$$E[D] = \sum_{i=1}^{k-1} D_i\pi_i + \sum_{i=2}^{k-1} \sum_{j=1}^{\lceil \frac{k-2}{2} \rceil} D_i\pi_i^{(j)}, \tag{8}$$

where D_i is the e2e delay when the UE is at distance i (cells belonging to ring i).

4.4 Service Migration Cost

MC denotes the cost of migrating part (i.e., some of the components/sessions composing it) or all the service from a data center to the optimal one. It depends on the size of the objects to be migrated, as well as the amount of signaling messages exchanged among the follow-me cloud controller,

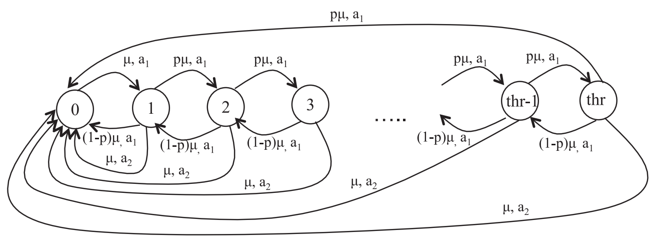


Fig. 4. Continuous time Markov decision process of the service migration procedure: 1D mobility model.

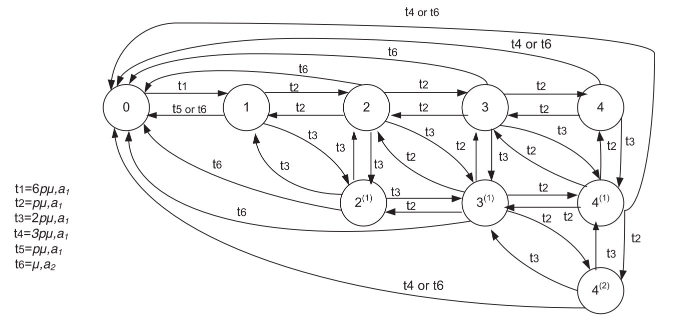


Fig. 5. Continuous time Markov decision process of the service migration procedure: 2D mobility model.

the user equipment and the data centers. In follow-me cloud, there are three signaling messages to trigger service migration. The cost for a service migration thus follows:

$$Cost = Objects_{size} + 3SIG_{size}, \tag{9}$$

where SIG_{size} is the signaling message size. Hence, MC can be derived as follows:

$$MC = \left[3p\pi_{k-1} + 2p\left(\sum_{j=1}^{\lceil\frac{k-2}{2}\rceil}\pi_{k-1}^{(j)}\right)\right] \times Cost. \tag{10}$$

4.5 Service Migration Duration

The service migration duration is the time required to transfer part or all of the service from the current data center to the optimal one. It mainly depends on: (i) the size of the objects to transfer, (ii) the RTT of the TCP connection between the two data centers, and (iii) the time needed to convert a virtual machine to the appropriate format, if the two data centers are not using the same hypervisor. It also represents the time when the service cannot be used, in other words, service disruption time (SDT). Assuming that the data transfer is based on an FTP-like application, we use the empirical TCP latency model of Sikdar et al. [28], and the service disruption time value can be computed as follows:

$$\begin{aligned} SDT = {} & [\log_{1.57}N + f(p_{loss}, RTT)N + 4p_{loss}\log_{1.57}N \\ & + 20p_{loss} + \frac{(10+3RTT)}{4(1-p_{loss})W_{max}\sqrt{W_{max}}}N]RTT \\ & + T_{VM\_conversion}, \end{aligned} \tag{11}$$

where p_{loss} denotes the packet loss rate, N is the number of packets to transfer, W_{max} is the maximum size of the congestion window, $T_{VM\_conversion}$ is the time required to convert a virtual machine and $f(p_{loss}, RTT) = \frac{2.32(2p_{loss}+4p_{loss}^2+16p_{loss}^3)}{(1+RTT)^3}N + \frac{(1+p_{loss})}{RTT10^3}$.

Note that $N = \lceil\frac{Service_{size}}{MSS}\rceil$, where MSS is the maximum segment size used by the TCP connection.

5 MDP-Based Scheme for Service Migration

We model the service migration decision as a Markov Decision Process, capturing the tradeoff between reducing cost and maintaining satisfactory user experience. This model decides whether a service consumed by a user at distance d from the current data center should be migrated to an optimal data center, a decision process carried out by the follow-me cloud controller. To formulate the service migration decision policy, we define a Continuous Time Markov Decision Process (CTMDP) that associates to each state an action, corresponding transition probabilities, and rewards.

Let s_t be the process describing the evolution of the system state and S denote the state space. We assume the cellular network topology model of Fig. 2. Each cell belongs to a Tracking Area (TA) and each tracking area belongs to a Service Area (SA), which is served by one anchor gateway (packed data network gateway or access router). Fig. 4 shows the continuous time markov decision process for the case of a one-dimensional (1D) mobility model: A mobile user has only two possible destinations, i.e., a new service area with probability p, or moving back to a visited service area with probability $1-p$. Higher values of p indicate that a user is moving far from the current data center. Fig. 5 illustrates the case for the 2D mobility model described in the previous section. The vector $A = (a_1, a_2)$ describes the actions available to the follow-me cloud controller at each epoch (i.e., when a user equipment performs handoff and enters into another service area). Action a_2 is used if the service is migrated to an optimal data center, while action a_1 is used if the user equipment is still served by the same data center. Depending on the current state, the set of available actions differs. For the sake of simplicity, we demonstrate the use of markov decision process to solve the service migration problem for 1D mobility. The same reasoning can be applied to the case for the 2D model shown in Fig. 5. Note that, albeit its simplicity, the 1D model is appropriate for vehicular networking environments where users move along predefined trajectories, such as highways or railway tracks.

In the 1D mobility model, the residence time of a user in each service area follows an exponential distribution with mean $1/\mu$. Hence, the state space S is defined as $S = \{0, 1, \ldots, thr\}$, where thr represents the maximum distance (in terms of visited service areas) from where the service must be migrated to the optimal data center.

The follow-me cloud controller observes the current state s of the network and associates a set of possible actions A_s to it, taken upon arrival to it from the previous state. For a given action a, an instantaneous reward $r(s, s', a)$ is associated to a transition from state s to another state s'. The corresponding formal representation of the continuous time markov decision process is as follows:

$$(S, (A_s, s \in S), q(s'|s, a), r(s, s', a)).$$

For particular states, the set of possible actions A reduces to a subset A_s. A policy P associates an action $a(s|P)$ to a state s. Let Q be the transition matrix, with $q(s'|s, a)$ denoting the transition rate between states s and s' in S due to action a, which, in the follow-me cloud scenario, represents a user equipment moving from one service area to another service area. By construction, we define a policy as a function of the actual state. The decision to migrate a service or not is taken by observing only the actual state. Since this process is Markovian (the sojourn time in a service area follows an exponential distribution), the controlled process is also Markovian. To resolve the markov decision process, we use an equivalent Discrete Time Markov Decision Process (DTMDP) for the defined continuous time markov decision process, with a finite state space S. We argue that the state space is finite, since, after a certain distance (thr) from the current data center, the service is automatically migrated to the optimal data center. For each $s \in S$, A_s represents the finite set of allowed actions in that state. This discrete time Markov decision process can be derived by uniformization and discretization of the initial process as follows: When all transition rates in matrix Q are bounded, the sojourn times in all states are exponential with bounded parameters $tr(s|s,a)$. Therefore, a $sup_{(s\in S, a\in A_s)} tr(s|s,a)$ exists and there is a constant value c such that

$$sup_{(s\in S, a\in A_s)}[1-p(s,a)]tr(s|s,a) \leq c < \infty,$$

where $p(s|s,a)$ denotes the probabilities of staying in the same state after the next event. We can now define an equivalent uniformized process with state-independent exponential sojourn times with parameter c, and transition probabilities

$$p(s'|s,a) = \begin{cases} 1 - \frac{([1-p(s|s)]tr(s|s,a))}{c} & s = s' \\ \frac{p(s'|s)tr(s'|s,a)}{c} & s \neq s'. \end{cases}$$

By setting $c = \mu$, the transition probabilities of the discrete time markov decision process procedure are defined as follows:

$$p(j|s,a) = \begin{cases} 1 & j = 0, s \neq 0, s \neq thr, a = a_1 \\ & or\ j = 1, s = 0, a = a_2 \\ p & j = s+1, s \neq 0, s \neq thr, a = a_1 \\ & or\ j = 0, s = thr, a = a_1 \\ 1-p & j = s-1, s \neq 0, a = a_1 \\ 0 & Otherwise. \end{cases}$$

It is important to note that when the system is in state $s = thr$, the only available action is a_1. If the user equipment moves to another service area, where the distance exceeds the threshold thr, service migration is automatically triggered.

In the remainder of this section, we use the discrete time markov decision process version. For $t \in N$, let s_t, a_t and r_t denote the state, action and reward at time t of the discrete time markov decision process procedure, respectively. Let $P^a_{(s,s')} = p[s_{(t+1)} = s'|s_t = s, s_{(t+1)} = s', a_t = a]$ denote the transition probabilities and $R^a_{(s,s')} = E[r_{(t+1)}|s_t = s, s_{(t+1)} = s', a_t = a]$ denote the expected reward associated with the transitions. A policy π is a mapping between a state and an action, and can be denoted as $a_t = \pi(s_t)$, where $t \in N$. Accordingly, a policy $\pi = (\theta_1, \theta_2, \theta_3, \ldots, \theta_N)$ is a sequence of decision rules to be used at all decision epochs. We restrict ourselves only to deterministic policies, as they are simple to implement [29]. When a user equipment hands off a particular service area to another service area, the follow-me cloud controller has to decide either to migrate the service using action a_2 or not to migrate it using action a_1. For each transition, a reward is obtained. This reward is a function of the cost of migrating a service (zero in case of no migration) and the quality obtained from the new state. The cost of migrating a service is defined as follows:

$$g(a) = \begin{cases} 0 & a = a_1, \\ C_m & a = a_2, \end{cases}$$

where C_m denotes the cost of migrating all the service or a part of it. Therefore the reward function is given by

$$r(s,s',a) = Q(s') - g(a),$$

where $Q(s)$ quantifies the quality perceived by a user connected to the source data center when at state s. Note that quality is inversely proportional to the hop distance from the source data center, and $Q(0)$ is the maximum quality that a user equipment can enjoy, when connected to the optimal data center. In general, the quality function can be expressed in the form $Q(s) = Q(0) - K$, where K denotes a predetermined factor. Given a discount factor $\gamma \in [0,1]$ and an initial state s, we define the total discount reward policy $\pi = (\theta_1, \theta_2, \theta_3, \ldots, \theta_N)$ as

$$v^\pi_\gamma = \lim_{N\to\infty} E^\pi_\gamma\left\{\sum_{t=1}^{N} \gamma^{t-1} r_t\right\} = E^\pi_\gamma\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t\right\}.$$

Given the uniformization of the continuous time markov decision process, $r(s,s',a)$ explicitly depends on the transitions between states. The new reward function $r'(s,s',a)$ is obtained as follows [29]:

$$r'(s,s',a) = r(s,s',a)\frac{\alpha + \beta(s,s',a)}{\alpha + c},$$

where $\beta(s,s',a)$ is the transition rate between states s and s' when using action a, and α is a predetermined parameter. With the new formulation of the reward function and the uniformization of continuous time markov decision process, we can use the discounted models as in discrete models to resolve the system [29]. Let $v(s) = max_{\pi\in\Pi} v^\pi(s)$ denote the maximum discounted total reward, given the initial state s. From [29], the optimality equations are given by

$$v(s) = max_{\pi\in\Pi}\{r'(s,s',a) + \sum_{s'\in S} \gamma P[s'|s,a]v(s')\}.$$

The solutions of the optimality equations correspond to the maximum expected discounted total reward $v(s)$ and the optimal policy $\pi^*(s)$. It is worth mentioning that the optimal policy $\pi^*(s)$ indicates the decision as to which network and which data center the user equipment is to be attached and to be connected, respectively, given the state s. There are several algorithms that can be used to solve the optimization problem given by the above optimality equations. Value iteration and policy iteration are two noticeable examples [29]. In our case, we have used the former.

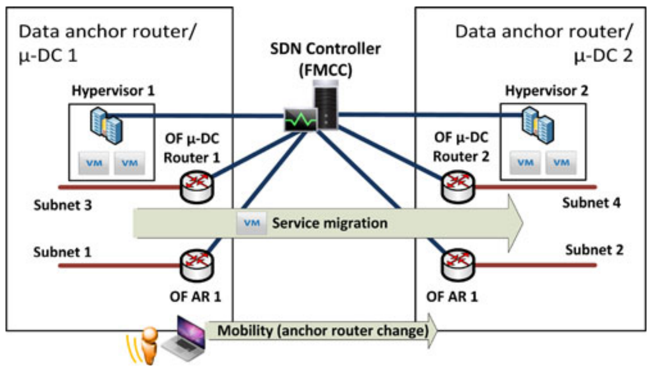


Fig. 6. Openflow-based follow-me cloud architecture.

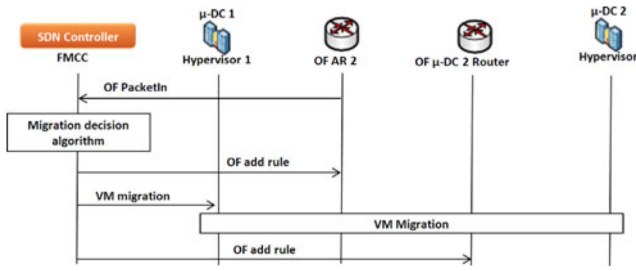


Fig. 7. Service migration procedure in an software-defined networking-based follow-me cloud implementation.

6 Implementation Alternatives

To show the breadth of available options to realize the follow-me cloud architecture, we explore alternative enabling technologies for its implementation. We have identified two candidate approaches, one based on software-defined networking and one on the locator-identifier separation protocol protocol. Note however that, as we have shown [30], the follow-me cloud design can be integrated also with the Mobile IPv6 protocol.

6.1 OpenFlow-Based Follow-Me Cloud Architecture

6.1.1 Design

Our software-defined networking-based scheme is built on the NOX Openflow Controller [31] and is shown in Fig. 6.

Mobile users access cloud-based services over a (mobile/wireless) client network and their traffic is forwarded by Openflow-capable access routers (OF AR). Traffic from/to each federated cloud location (data center), inside which virtual machine instances interconnected via virtual switches are managed by local hypervisors, is routed through Openflow microdata center routers (OF μ-DC x). At the core of this architecture is our NOX-based follow-me cloud controller, with which the components of our architecture communicate. The controller is assumed to be aware of i) the virtual switch instances and their data path identifiers on the physical Openflow switch, ii) the virtual machine identifiers (namely the IP and MAC addresses), iii) the location and IP addresses of each default gateway in the topology, iv) the Openflow switch port identifiers at which the data center, router, and client networks are connected, v) the IP address ranges managed by each DHCP server both for client and DC networks, and vi) the locations of distributed data centers that can either be part of the operator network or could be autonomous domains. The basic functionalities of the software-defined networking-based follow-me cloud controller follow.

Location management. For correctly installing forwarding rules into the Openflow switch, each client and virtual machine are linked to home locations, based on their IP address allocation and gateway settings. The current location of a client in the client network is also maintained. If any traffic from a particular client or virtual machine appears on a different network than the one corresponding to its home location, the follow-me cloud controller updates the status of that entity to indicate that it is in a visited network/location.

The location management process is also responsible for selecting the appropriate microdata center location for the virtual machine serving a client, utilizing information about the geographic location of data centers and characteristics of the client-virtual machine/data center network paths, such as average delay and network load or congestion.

Mobility detection and service migration. The actual virtual machine migration is carried out by the cloud infrastructure software. (See Section 6.1.2 for some details on our proof-of-concept implementation.) The sequence of messages and events is shown in Fig. 7.

When a user changes its point of attachment, the Openflow access router on the visited network sends a PacketIn Openflow message to the controller, which is thus notified of the mobility event. This process can be initiated when a user performs the DHCP message exchange with the visited network. Then, the follow-me cloud controller executes the service migration decision algorithm (see Section 5), and, if migration is necessary, it installs the appropriate OF rule at the Openflow access router of the visited network and launches the virtual machine migration, also adding a rule to the Openflow router serving the microdata center which hosts the migrated virtual machine, so that when traffic for the latter reaches the Openflow switch of a visited network, it can be matched in the flow table of that switch.

Session management. A key functional requirement is for the controller to preserve all ongoing user sessions while a virtual machine is migrated. This implies that no configuration change (e.g., IP address and gateway configurations) is allowed on the virtual machine. Furthermore, the (private) IP address ranges managed by data centers can be overlapping, and the first-hop setting may not be consistent across subnet boundaries. We address this by setting up a tunnel within the visited network segment. The tunnel operates by rewriting the IP address field within the packet IP header for each outgoing packet from the virtual machine to the external network. The original IP header is restored in the packet when the last hop in the visited network segment is reached. The same technique is applied for all the incoming traffic to the virtual machine. This is achieved by modifying the set of Openflow rules installed in the visited network.

6.1.2 Experimental Setup

We have developed an experimental testbed implementing the software-defined networking-based architecture of Fig. 6. Our setup includes two data centers, each one implemented as a single VMWare ESXi hypervisor. Each ESXi host is equipped with two 1 Gb/s Ethernet cards for forwarding management and Openflow traffic over the network. A virtual network topology is defined inside the ESXi host by two vSwitches (soft switches), where each physical NIC is connected with each soft switch instance. The ESXi host manages Windows XP virtual machines and each virtual machine is configured with two virtual NICs (vNIC) connected with the virtual network through the soft switches. One vNIC carries management traffic (vmKernel) and the other carries Openflow traffic. The storage space is shared between the two data centers and is accessed by the standard iSCSI protocol. The data centers are remotely managed by the VMWare vCenter software.

There are two separate WLANs for client connectivity and two Linux-based hosts act as first-hop routers for client traffic, assigning client addresses using DHCP and running Linux traffic control (tc) for controlling path characteristics (e.g., delay and available bandwidth, and thus congestion) between the two network segments. In this simplified setup, each data center host plays the role of a microdata center mapped to a data anchor router (in our case the WLAN first-hop router), and is placed in each of the two client networks (and served by the respective router).

The NOX-based follow-me cloud controller, as specified in Section 6.1, the Linux-based routers, ESXi hosts and the VMWare vCenter host are all connected to ports of an NEC IP-8800 OpenFlow switch. From the physical Openflow switch perspective, four virtual switches (VLAN) are used for separately carrying the traffic of the two data centers and the client network. The follow-me cloud controller manages the forwarding behavior on the four VLAN's and also monitors the path characteristics between a data center and the client network for resource management optimizations. For triggering live virtual machine migration across data centers, the VMotion cloud infrastructure technology and proprietary API from VMWare are used. VMotion traffic is mapped to the management network, while all active communication between the virtual machine and remote users is managed by the Openflow network.

6.2 Locator-Identifier Separation Protocol-Based Follow-Me Cloud Architecture

6.2.1 *Design*

Our second architectural alternative is based on the locator-identifier separation protocol protocol. Each microdata center is connected to the Internet through an tunneling router. The client (mobile) network domain contains IP subnets interconnected through tunneling routers as well, and the architecture includes a locator-identifier separation protocol map resolver/map server element. All locator-identifier separation protocol entities (map resolver/map server and tunneling routers) are implemented as VNFs and are deployed on virtual machines in the cloud. Note that, in practice, the tunneling routers of the mobile transport network domain could either be VNFs or be built directly on top of the data anchor router hardware. This follow-me cloud architecture also includes an follow-me cloud controller in the form of a VNF.

Fig. 8 shows the envisioned locator-identifier separation protocol-based follow-me cloud architecture and scenario. A mobile user accessing a service hosted at microdata center1, and initially connected to Subnet 1, moves to Subnet 2. The tunneling router of Subnet 2 notifies the map resolver/

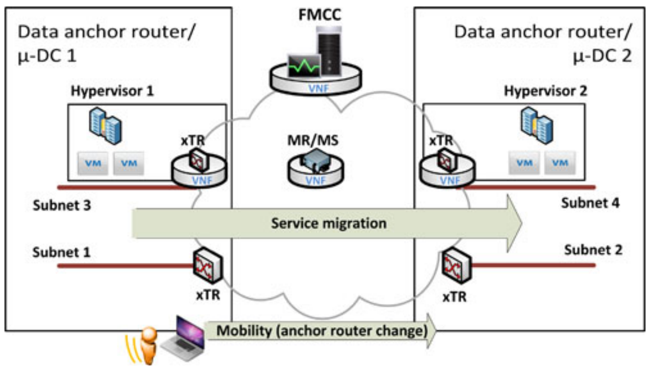


Fig. 8. A locator-identifier separation protocol-based follow-me cloud architecture.

map server about this mobility event. The map resolver/map server updates its cache and informs the follow-me cloud controller about the new location of the user. The follow-me cloud controller executes the service migration decision algorithm (see Section 5) to decide whether to migrate the user's service (that is, the virtual machine(s) hosting the service) to the microdata center corresponding to the new location of the user. If the decision is positive, the follow-me cloud controller requests Hypervisor 1 to launch the migration procedure. Since the virtual machine is migrated to Hypervisor 2, the tunneling router of Subnet 4 is informed of the migration event, and the latter further informs the map resolver and the tunneling router of Subnet 3 (as well as other tunneling routers involved in communication with the virtual machine) accordingly. Finally, the map resolver/map server resolver updates its cache and notifies the follow-me cloud controller about this change.

6.2.2 *Support for Service Continuity*

To ensure service continuity, a locator-identifier separation protocol-assisted live service migration mechanism should be capable of (i) maintaining the virtual machine endpoint identifier when migrating it from its current data center to the target one, (ii) updating the routing locator of the target tunneling router to include the virtual machine's endpoint identifier, (iii) informing the map resolver server and all tunneling routers involved in the communication with the migrated virtual machine to update the routing locator of the migrated virtual machine, and (iv) informing the old tunneling router to erase the virtual machine endpoint identifier from its cache.

In this work, we assume that a virtual machine's endpoint identifier is the first IP address it obtains, and that its routing locator is the IP address of the corresponding tunneling router. Furthermore, we consider that a virtual machine's endpoint identifier is registered at the initial tunneling router with a large IP subnet prefix (in our case, /24). The endpoint identifier is mapped to the routing locator of the source tunneling router at the map resolver, as well as in the caches of tunneling routers communicating with the virtual machine.

When the virtual machine is migrated to the target hypervisor, the endpoint identifier of the migrated virtual machine should be maintained and the destination tunneling router has to be informed about the migration event. Different approaches exist to achieve this. In one approach [32], the

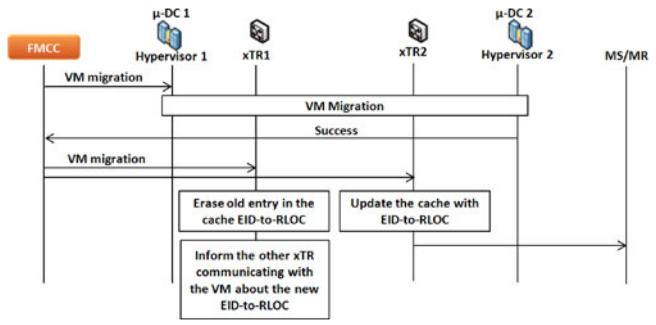

µ-DC 1
µ-DC 2
FMCC
Hypervisor 1
xTR1
xTR2
Hypervisor 2
MS/MR
VM migration
VM Migration
Success
VM migration
Erase old entry in the cache EID-to-RLOC
Update the cache with EID-to-RLOC
Inform the other xTR communicating with the VM about the new EID-to-RLOC

Fig. 9. Service migration procedure in a locator-identifier separation protocol-based follow-me cloud implementation.

tunneling router does not become aware of the new virtual machine until the latter initiates communication, when the tunneling router detects that the source IP address (migrated virtual machine's endpoint identifier) is not belonging to its IP subnet. Although this solution does not require any signaling messages, it can break the current virtual machine's connections and thus does not ensure service continuity; if the virtual machine has no packet to transmit, the current tunneling router communicating with the virtual machine may continue using its old Routing Locator. An alternative to this approach was proposed by Raad et al. [23], where locator-identifier separation protocol is used in the control plane to inform the source and target tunneling routers about a successful virtual machine migration. A new Change Priority (CP) locator-identifier separation protocol message is introduced, which allows (i) the target hypervisor to inform the target tunneling router about the migration of a new virtual machine to the target data (including the virtual machine's endpoint identifier), and (ii) to update the cache (routing locator-endpoint identifier mapping) of other tunneling routers. However, this solution requires modifying the hypervisor, making it hard to implement and deploy, as the hypervisor software is independent from the operator (as well as the locator-identifier separation protocol domain).

In this article, we propose another approach (Fig. 9), where the follow-me cloud controller informs both tunneling routers (handling the involved data center domains) about the change in the virtual machine's routing locator. Since the follow-me cloud controller is integrated in the locator-identifier separation protocol domain (it already communicates with map resolver/map server to track user locations), it could easily be made aware of the tunneling router handling a data center domain. This could be obtained by sending a message to map resolver/map server to know the tunneling router handling the data center's IP domain.

Once the tunneling router becomes aware of the migration of a new virtual machine to its local data center, it notifies the map resolver/map server to update its routing locator by including the migrated virtual machine's endpoint identifier. The endpoint identifier of the migrated virtual machine is in the form of the initial IP address, but with a /32 prefix. Therefore, the routing locator of the target tunneling router is mapped to both its subnet prefix and the virtual machine's endpoint identifier prefix (/32). Furthermore, the former tunneling router erases the old endpoint identifier-to-routing locator entry from its cache. To speed up traffic redirection, the source tunneling router uses a new locator-identifier separation protocol message (as in [23]) to inform the other tunneling router which was communicating with the concerned virtual machine so that the latter accordingly updates the virtual machine's routing locator. Note that the tunneling router should maintain a list of tunneling routers involved with each active connection.

6.2.3 Experimental Setup

The VNFs corresponding to the entities of our locator-identifier separation protocol-based follow-me cloud scheme are implemented using the click modular router framework [33] and run as ClickOS [34] virtual machines deployed on a Xen hypervisor. Data centers are emulated using kvm [35] to take advantage of the virtual machine migration options that it offers, as kvm can migrate only the RAM content between the involved data centers, while the hypervisors have a shared storage via NFS. We emulate a mobile user moving between two IP domains; whenever the user enters a new domain, service (virtual machine) migration is triggered. Each virtual machine is Ubuntu 8.04, with 1 GB of RAM and a one-core processor. Data Centers (kvm managers) run Ubuntu 14.04 with 8 GB of RAM. We emulated varying latencies in the paths between data centers and between the follow-me cloud controller and tunneling routers using Linux tc.

7 Performance Evaluation

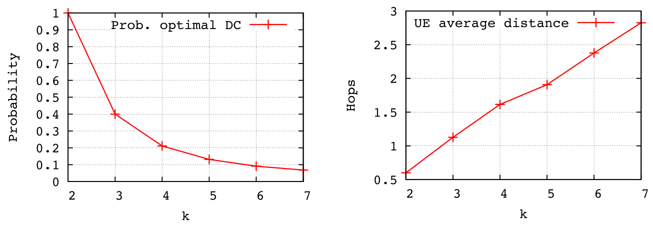
In this section we present a quantitative performance evaluation of the proposed scheme. We begin with numerical results from our analytic model, followed by measurements carried out on our testbed implementation of the software-defined networking- and locator-identifier separation protocol-based follow-me cloud solutions.

7.1 Model-Based Performance Results

We present numerical results obtained by resolving the Markov model defined in Section 4.1. We evaluate the performance of follow-me cloud in terms of the probability that the user equipment is connected to the optimal data center, the average distance of the user equipment from the optimal data center, the user equipment connection latency, the service migration cost and the service disruption time during service migration.

We assume a reliable connection between data centers (zero packet loss), the total size of the service to migrate (i.e., the respective virtual machine) is set to 1 GB, and all data centers are assumed to use the same hypervisor; thus, there is no virtual machine conversion cost when migrating a service. Note that the case for $k = 7$ corresponds to a situation where the follow-me cloud concept is not used, as k is unrealistically high; in practice, the service area is typically limited to a modest value for k.

Figs. 10a and 10b show the probability of a user equipment to be connected to an optimal data center, and the average distance from it for different values of k, respectively. We notice that this probability is a decreasing function of k: High probability is obtained when the service migration is triggered after each user equipment handover, ensuring that the user equipment is always connected to the optimal data center, while the opposite effect appears when delaying service migration to longer distances. On the other hand, the average distance is an increasing function of k. Indeed, if



(a) π_0 (b) Average distance

Fig. 10. Probability to be connected to the optimal data center and the average distance from it.

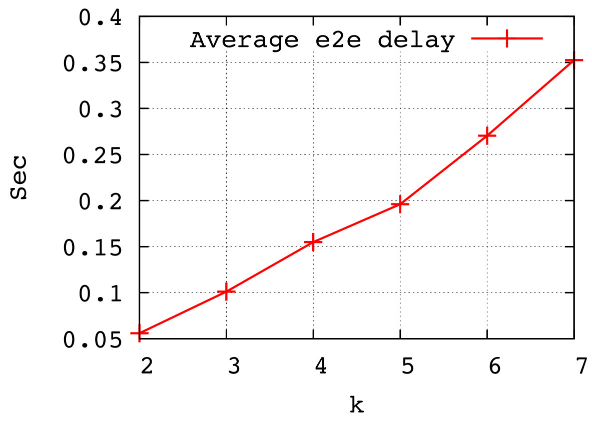


Fig. 11. Average latency.

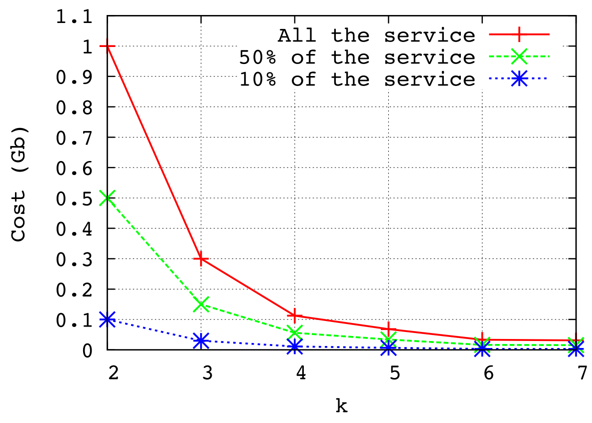
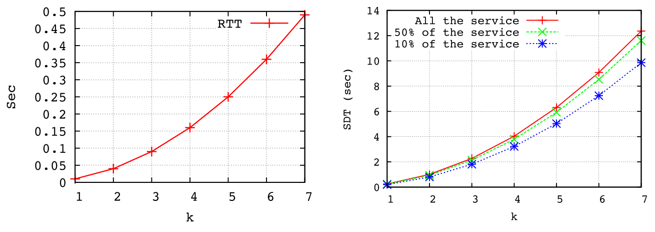


Fig. 12. Cost of service migration.

service migration is delayed, the user equipment is likely connected from a distance higher than one hop. This average distance exceeds two hops when k is higher than six.

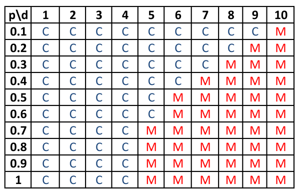
In Fig. 11, we plot the average latency of the user equipment connection for different values of k. Note that latency is given by $Lat_i = 0.02i^2$ (s), where i is the hop distance from the serving data center. Intuitively, the average latency increases with k. If we compare the cases of using $k = 2$ and $k = 7$, we see a significant difference of about 200 ms in delays.

Fig. 12 shows the service migration cost for different values of k. We present results for migrating all, 50, and 10 percent of the service. This cost is a decreasing function of k, and it is high when service relocation is launched at each user equipment handover. Furthermore, the highest cost is reached when migrating all the service, as it critically depends on the object size to migrate.



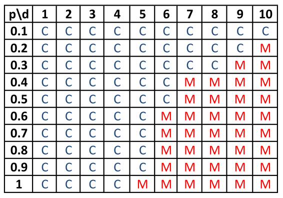
(a) RTT (b) SDT

Fig. 13. Service disruption time – RTT proportional to the square of the distance.



p\d	1	2	3	4	5	6	7	8	9	10
0.1	C	C	C	C	C	C	C	C	C	M
0.2	C	C	C	C	C	C	C	C	M	M
0.3	C	C	C	C	C	C	C	M	M	M
0.4	C	C	C	C	C	C	M	M	M	M
0.5	C	C	C	C	C	M	M	M	M	M
0.6	C	C	C	C	C	M	M	M	M	M
0.7	C	C	C	C	M	M	M	M	M	M
0.8	C	C	C	C	M	M	M	M	M	M
0.9	C	C	C	C	M	M	M	M	M	M
1	C	C	C	C	M	M	M	M	M	M

(a) $\tau = 0.1$



p\d	1	2	3	4	5	6	7	8	9	10
0.1	C	C	C	C	C	C	C	C	C	C
0.2	C	C	C	C	C	C	C	C	C	M
0.3	C	C	C	C	C	C	C	C	M	M
0.4	C	C	C	C	C	C	M	M	M	M
0.5	C	C	C	C	C	C	M	M	M	M
0.6	C	C	C	C	C	M	M	M	M	M
0.7	C	C	C	C	C	M	M	M	M	M
0.8	C	C	C	C	C	M	M	M	M	M
0.9	C	C	C	C	C	M	M	M	M	M
1	C	C	C	C	M	M	M	M	M	M

(b) $\tau = 0.5$

Fig. 14. Optimal policy construction.

Fig. 13 depicts the service disruption time for different values of k. We assume that RTT is proportional to the square of the hop distance i between two data centers, given by $RTT_i = 0.01i^2$ (s). Similar to Fig. 12, we considered three service migration cases: migrating all, 50, or 10 percent of the service. We clearly observe that the service disruption time value is an increasing function of k. This is attributable to the fact that high values of k mean longer distances between the concerned data centers, thus an increased value of RTT. In addition, we notice that the service disruption time value is the highest when migrating all the service, mainly due to the larger size of the objects to transfer. Notably, this difference is not important for low values of k. Since service disruption time is highly dependent on the RTT, accelerating data transfers using tools like FDT [36] is mandatory when the RTT is high.

7.2 Service Migration Policies

Given the tradeoff between migration cost and user experience improvement identified in Section 7.1, we demonstrate the construction of service migration policies obtained using a MATLAB implementation [37] of the value iteration algorithm [29]. We set $thr = 10$, i.e., a service migration is automatically triggered if the user equipment is at a distance higher than 10 (in terms of the number of visited service areas) from the source data center. The factor K of the quality function is arbitrarily set to 1. We introduce a new metric τ representing the ratio between the cost and the maximum quality $Q(0)$. Two scenarios are studied:

- $\tau = 0.1$, which represents a low cost compared to the quality obtained if a service migration is launched. This could be the case when only a part of a service is migrated.
- $\tau = 0.5$, which indicates that the cost is not negligible compared to the quality obtained if a service migration is launched.

Figs. 14a and 14b illustrate the optimal policy constructions for the aforementioned scenarios. The intersection between i and j represents the action (a=C continuation, a=M migration of the service) to be taken by the follow-me cloud controller, where i is the distance from the serving data center and j is the probability p. We remark that the optimal policy construction has a threshold-based form, since beyond a certain distance value, the recommended action is service migration. This distance is inversely proportional to the probability p.

For instance, when $p = 0.8$, the proposed policy recommends service migration, if $d = 6$ and $d = 5$, for the first and second scenarios, respectively, since a high p value means that the user equipment is moving far from the current data center, while low values indicate that the user equipment will most likely remain in the vicinity of the current data center, or will come back to the service area of the current data center. Furthermore, we remark that τ has an impact on the optimal policy construction, since there are less service migrations when τ is high. This is intuitive, as the incurred cost is not negligible compared to the achieved gain when migrating a service. In case of a random walk ($p = 0.5$), for both scenarios, the optimal policy recommends service migration when the distance exceeds 5, which represents a good tradeoff between cost and quality.

7.3 Testbed Experiments

7.3.1 *OpenFlow-Based Implementation*

We present early results obtained from experiments on our software-defined networking-based follow-me cloud testbed described in Section 6.1.2. In particular, we focus on the evolution of service latency during and after the migration process, to demonstrate the advantages of follow-me cloud. To emulate increased service latency due to a user being served from a suboptimal data center, we introduced a 50 ms delay (round-trip) in the path between the client network and the suboptimal data center using Linux tc, while the RTT to the optimal data center was 1 ms. In our experiment, we started with the client originally being served from the suboptimal data center and we monitored the RTT from the user terminal to the virtual machine hosting the service using ping. When follow-me cloud is not used, the client keeps being served suboptimally all along the duration of the experiment (and thus the approximately 50 ms RTT). On the other hand, with follow-me cloud active, when service migration is triggered, and after a period of instability and a short-term increase in latency, the latter converges to a value close to 1 ms upon migration completion.

We further noted an increased delay at the start of the experiment, which is mainly a result of the installation of openflow rules when new traffic arrives at the follow-me cloud controller. This implies a scalability issue for large centralized follow-me cloud deployments. Therefore, decentralization schemes, such as the one proposed by Bifulco et al. [15] could be considered to this end.

7.3.2 *Locator-Identifier Separation Protocol-Based Implementation*

For our locator-identifier separation protocol-based follow-me cloud design, we experimented with the testbed described in Section 6.2.3. Our metrics of interest are (i) the service downtime duration, i.e., the time when the virtual machine is not available, (ii) the RTT between the mobile user and the remote virtual machine, and (iii) the migration duration.

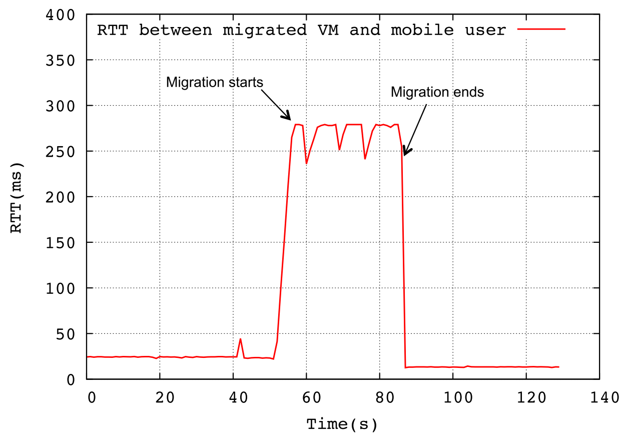


Fig. 15. RTT between the migrated virtual machine and mobile user.

Fig. 15 shows the instantaneous RTT between the migrated virtual machine and the mobile user, measured using ping. Using Linux tc, we introduce a 10 ms RTT between the data centers. The RTT between the follow-me cloud controller and tunneling router 1 is approximately 1 ms, while the RTT between the follow-me cloud controller and tunneling router 2 is set to 10 ms. At the start of the experiment, the user is connected to data center 1, and enjoys good service quality ($RTT \approx 12$ ms). From $t = 55$ s, the user moves to another network and the measured RTT increases to approximately 250 ms. The follow-me cloud controller thus triggers service migration to data center 2, which is considered the optimal one. Migration starts at $t = 58$ s and ends at $t = 84$ s. During this period, the overall service downtime is 7.5 ms (not visible in the figure due to its short duration). This downtime is mainly due to the fact that the follow-me cloud controller does not notify the involved tunneling routers about the change in the virtual machine's endpoint identifier until migration is completed. The kvm hypervisor at data center 1 keeps the virtual machine active until migration is complete and the virtual machine keeps sending back ICMP echo replies from its original location (data center 1). After migration is complete, the user is served by data center 2 (optimal), which brings RTT down to a much lower value.

We further quantify the dependence of the service downtime duration on the latencies in the paths between the entities of our architecture. In Fig. 16, we show the downtime duration when the virtual machine is migrated from data center 1 to data center 2 for different values of the RTT between the follow-me cloud controller and data center 2 for the following testbed configurations: (i) The RTTs between the follow-me cloud controller and tunneling router 1 and tunneling router 2 are set to 100 and 10 ms, respectively (case 1). (ii) The RTTs between the follow-me cloud controller and tunneling router 1 and tunneling router 2 are set to 50 and 50 ms, respectively (case 2).

It appears that the downtime duration is proportional to the RTT between the follow-me cloud controller and the target data center, since the longer it takes to have the information from the data center about the success of a virtual

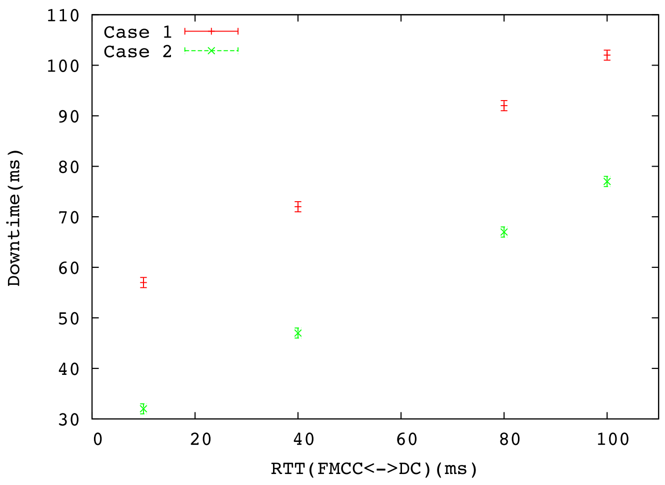


Fig. 16. Downtime duration as a function of path latencies. Each point is the mean value of a few tens of iterations.

machine migration, the longer it takes to accordingly inform the tunneling router routers and thus redirect traffic to tunneling router 2. We draw the same conclusion for the impact of the RTT between the follow-me cloud controller and the tunneling routers on downtime. Service downtime is mainly rooted at the locator-identifier separation protocol mobility management process, and its capacity to rapidly inform the tunneling routers about the migration event. The virtual machine size does not have a strong impact on downtime, since kvm activates the virtual machine in data center 2 only after migration is complete, which confirms the observations made by Raad et al. [23]. Importantly, the maximum downtime experienced is 100 ms (case 2), which is not expected to have a noticeable impact on service quality, especially for non-interactive applications.

Another conclusion from our experiments is that the duration of the service migration itself becomes practically independent of the RTT between data centers, as the latter increases. The time it takes to migrate a virtual machine from data center 1 to data center 2 does not change much as the RTT in the data center 1-data center 2 path increases beyond 10 ms. This is because virtual machine migration is based on TCP, which is impacted more by the link bandwidth (set to 100 Mb/s in this experiment) than by the RTT.

8 Conclusion

We presented our vision towards the enhanced delivery of cloud-based services to mobile users, tackling mobility-related challenges and offering an optimized user experience. We have designed Follow-Me Cloud, a framework that enables cloud services to "follow" users on the move, by performing sophisticated decisions to migrate service resources to the appropriate cloud infrastructure locations. We have defined an analytic model for the behavior of our system, and build on it to propose a Markov-Decision-Process-based algorithm for service migration decisions, which captures the tradeoff between migration cost and user experience. We have further developed two alternative follow-me cloud architecture designs, one which is based on software-defined networking technologies and one making use of the locator-identifier separation protocol protocol. The presented numerical results from our model, as well as experiments with our software-defined networking and locator-identifier separation protocol-based testbeds, demonstrate the potential of our approach for optimized mobile cloud-based service delivery and its feasibility for real-world deployment.

Appendix
List of Acronyms

CTMC	continuous-time Markov chain
CTMDP	continuous-time Markov decision process
DTMDP	discrete-time Markov decision process
EID	endpoint identifier
ETR	egress xTR
FMC	follow-me cloud
FMCC	fMC controller
ITR	ingress xTR
MDP	Markov decision process
MR	map resolver
MS	map server
OF AR	openflow access router
PGW	packet data network gateway
RLOC	routing locator
SGW	serving gateway
UE	user equipment
xTR	tunneling router

Acknowledgments

This research work is partially supported by the TAKE 5 project funded by the Finnish Funding Agency for Technology and Innovation (TEKES), a part of the Finnish Ministry of Employment and the Economy, and by the French FUI-18 DVD2C project.

References

[1] Cisco. (2014. Jun.) Cisco visual networking index: Forecast and methodology, 2013-2018, White Paper. [Online]. Available: http://goo.gl/xoBrTA
[2] T. Taleb, K. Samdanis, and A. Ksentini, "Supporting highly mobile users in Cost-effective decentralized mobile operator networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 7, pp. 3381–3396, Sep. 2014.
[3] L. Lei, Z. Zhong, K. Zheng, J. Chen, and H. Meng, "Challenges on wireless heterogeneous networks for mobile cloud computing," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 34–44, Jun. 2013.
[4] B. Frank, I. Poese, G. Smaragdakis, A. Feldmann, B. Maggs, S. Uhlig, V. Aggarwal, and F. Schneider, "Collaboration opportunities for content delivery and network infrastructures," *ACM SIGCOMM ebook on Recent Adv. Netw.*, vol. 1, pp. 305–377, Aug. 2013.
[5] R. Moreno-Vozmediano, R. Montero, and I. Llorente, "IaaS cloud architecture: From virtualized datacenters to federated cloud infrastructures," *IEEE Comput.*, vol. 45, no. 12, pp. 65–72, Dec. 2012.
[6] 3rd Generation Partnership Project (3GPP), "Technical Specification (TS) 23.002: Network architecture," release 13, Dec. 2015.
[7] H. Wang, S. Chen, H. Xu, M. Ai, and Y. Shi, "SoftNet: A software defined decentralized mobile network architecture toward 5G," *IEEE Netw.*, vol. 29, no. 2, pp. 16–22, Mar. 2015.
[8] 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 deployment over cloud," *IEEE Netw.*, vol. 29, no. 2, pp. 78–88, Mar. 2015.
[9] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep./Oct. 2013.
[10] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. (2011.) Locator/id separation protocol (lisp), Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lisp-13 [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-lisp-13.txt

[11] A. Ksentini, T. Taleb, and F. Messaoudi, "A Lisp-based Implementation of Follow Me Cloud," *IEEE Access*, vol. 2, pp. 1340–1347, Nov. 2014.

[12] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2014, pp. 1350–1354.

[13] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service-centric networking," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, 2012, p. 7.

[14] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for High-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 254–265.

[15] R. Bifulco, M. Brunner, R. Canonico, P. Hasselmeyer, and F. Mir, "Scalability of a mobile cloud management system," in *Proc. ACM SIGCOMM Workshop Mobile Cloud Comput.*, 2012, pp. 17–22.

[16] B. Malet and P. Pietzuch, "Resource allocation across multiple cloud data centres," in *Proc. 8th Int. Workshop Middleware Grids, Clouds e-Sci.*, 2010, pp. 5-1–5-6.

[17] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for Geo-distributed cloud services," *presented at the 7th USENIX Conf. Networking System Design Implementation*, San Jose, CA, USA, 2010.

[18] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 962–971.

[19] M. Steiner, B. G. Gaglianello, V. Gurbani, V. Hilt, W. Roome, M. Scharf, and T. Voith, "Network-aware service placement in a distributed cloud environment," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. protocols Comput. Commun.*, 2012, pp. 73–74.

[20] H. Watanabe, T. Ohigashi, T. Kondo, K. Nishimura, and R. Aibara, "A performance improvement method for the global live migration of virtual machine with ip mobility," *presented at the 5th Int. Conf. Mobile Computing Ubiquitous Networking*, Seattle, WA, USA, 2010.

[21] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The efficacy of live virtual machine migrations over the internet," *presented at the 2nd Int. Workshop Virtualization Technology Distribution Computing* , Washington, DC, USA, 2007.

[22] Q. Li, J. Huai, J. Li, T. Wo, and M. Wen, "HyperMIP: Hypervisor controlled mobile ip for virtual machine live migration across networks," in *Proc. 11th IEEE High Assur. Syst. Eng. Symp.*, Dec. 2008, pp. 80–88.

[23] P. Raad, S. Secci, D. P. Chi, A. Cianfrani, P. Gallard, and G. Pujolle, "Achieving sub-second downtimes in large-scale virtual machine migrations with LISP," *IEEE Trans. Netw. Serv. Manag.*, vol. 11, no. 2, pp. 133–143, Jun. 2014.

[24] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proc. IEEE Global Commun. Conf.*, Dec. 2013, pp. 1291–1296.

[25] (2016). Fon [Online]. Available: https://fon.com

[26] R. Langar, N. Bouabdallah, and R. Boutaba, "A comprehensive analysis of mobility management in MPLS-based wireless access networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 918–931, Aug. 2008.

[27] K.-H. Chiang and N. Shenoy, "A 2-D random-walk mobility model for Location-management studies in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 53, no. 2, pp. 413–424, Mar. 2004.

[28] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "An integrated model for the latency and steady-state throughput of TCP connections," *Perform. Eval.*, vol. 46, nos. 2-3, pp. 139–154, Oct. 2001.

[29] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: Wiley, 1994.

[30] A. Aissioui, A. Ksentini, and A. Gueroui, "PMIPV6-based follow me cloud," presented at the IEEE Globecom, San Diego, CA, USA, Dec. 2015.

[31] (2016). NOX OpenFlow Controller [Online]. Available: https://github.com/noxrepo

[32] Cisco, "Locator ID Separation Protocol (LISP) VM mobility solution," White Paper, Cisco Syst., Inc., Tech. Rep., 2011.

[33] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[34] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the art of network function virtualization," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation*, 2014, pp. 459–473.

[35] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: The linux virtual machine monitor," in *Proc. Linux Symp.*, Ottawa, ON, Canada, Jun. 2007, vol. 1, pp. 225–230.

[36] (2016). Fast Data Transfer [Online]. Available: http://monalisa.cern.ch/FDT/

[37] (2016). Markov decision processes (MDP) toolbox [Online]. Available: http://www7.inra.fr/mia/T/MDPtoolbox/

Tarik Taleb received the BE degree in information engineering with distinction, and the MSc and PhD degrees in information sciences from Tohoku University, Japan, in 2001, 2003, and 2005, respectively. He is currently a professor at the School of Electrical Engineering, Aalto University, Finland. Previously, he was a senior researcher and 3GPP standards expert with NEC Europe Ltd., Heidelberg, Germany (2009-2015). Until March 2009, he was an assistant professor at Tohoku University. From October 2005 to March 2006, he was a research fellow with the Intelligent Cosmos Research Institute, Sendai. His research interests include architectural enhancements to mobile core networks, cloud-based mobile networking, mobile multimedia streaming, inter-vehicular communications, and social media networking. Prof. Taleb has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group.

Adlen Ksentini received the MSc degree in telecommunication and multimedia networking from the University of Versailles, France, and the PhD degree in computer science from the University of Cergy-Pontoise, France, in 2005, with a dissertation on QoS provisioning in IEEE 802.11-based networks. From 2006 to 2015, he was an associate professor at University of Rennes 1, France, and member of the INRIA Rennes team Dionysos. Recently, he joined the Mobile and Wireless Networking Department at EURECOM Institute as an associate professor. His interests include future Internet networks, mobile networks, QoS, QoE, performance evaluation, and multimedia transmission.

Pantelis A. Frangoudis received the BSc (2003), MSc (2005), and PhD (2012) degrees in Computer Science from the Department of Informatics, AUEB, Greece. Currently, he is a post-doctoral researcher at team Dionysos, IRISA/INRIA Rennes, France, which he joined under an ERCIM post-doctoral fellowship (2012-2013). His research interests include wireless networking, Internet multimedia, network security, future Internet architectures, cloud computing, and QoE monitoring and management.