Linking Virtual Machine Mobility to User Mobility

Stefano Secci, Patrick Raad, Pascal Gallard

Linking Virtual Machine Mobility to User Mobility

Stefano Secci, *Senior Member, IEEE*, Patrick Raad, *Graduate Student Member, IEEE*, Pascal Gallard

***Abstract*—Cloud applications heavily rely on the network communication infrastructure, whose stability and latency directly affect the quality of experience. As mobile devices need to rapidly retrieve data from the cloud, it becomes an extremely important goal to deliver the lowest possible access latency at the best reliability. In this paper, we specify a cloud access overlay protocol architecture to improve the cloud access performance in distributed data-center (DC) cloud fabrics. We explore how linking virtual machine (VM) mobility and routing to user mobility can compensate performance decrease due to increased user-cloud network distance, by building an online cloud scheduling solution to optimally switch VM routing locators and to relocate VMs across DC sites, as a function of user-DC overlay network states. We evaluate our solution (i) on a real distributed DC testbed spanning the whole France, showing that we can grant a very high transfer time gain, and (ii) by emulating the situation of Internet Service Providers (ISPs) and over-the-top (OTT) cloud providers, exploiting many thousands real France-wide user displacement traces, finding a median throughput gain from 30% for OTT scenarii to 40% for ISP scenarii, the large majority of this gain being granted by adaptive VM mobility.**

***Index Terms*—Virtual Machine Mobility, Mobile Cloud Networking, Distributed Data-Center, LISP routing.**

I. Introduction

Cloud computing has witnessed a rapid growth over the last decade, with small and large companies increasingly migrating to cloud-based Infrastructure as a Service (IaaS) solutions. Experts believe that this trend will continue to develop further in the next few years [2].

Cloud providers are increasingly relying on virtualization to ease network and service management and to decrease expenses by disentangling the software from the hardware. Server virtualization also allows taking control over the guest operating system use of CPU, memory, storage and network resources, and to deploy advanced applications that can balance processing between the cloud and the client device. The common denominator goal in mobile cloud computing research is to build a cloud access infrastructure that is tailored to the mobility and the actual computing capabilities of client device. Very low latency and high reliability requirements are leading to a reduced wide area networks (WAN) segment between the cloud and the user, with a higher geographical distribution of data-centers (DCs) facilities. On the one hand, a recent study in [3] shows that about 25% of collocation DCs have three or more sites, and that about 5% have more than 10 sites. On the other hand, the so-called cloudlet solution [4] is gaining momentum: the idea is to bring cloud servers even closer to users, with small DCs directly in the access network. These concerns by cloud and network providers recently led to the creation of a new Industry Specification Group on Mobile Edge Computing at ETSI [6].

Cloud applications are increasingly accessed on the move: when the distance, in terms of network latency, between the user and the application gets larger, especially for interactive computing-intensive services, the quality of experience starts declining. To overcome this limitation one key approach is to relocate services (server virtual machines) to the closest data-center according to user's movement [7], [8]. Mobile devices using applications such as remote desktop, real-time voice/video recognition and augmented reality heavily rely on the cloud back-end and require a very low cloud access latency, between the user and the computation servers, to guarantee a pleasant user experience. Voice recognition and augmented reality constitute the rising star of this industry; for instance, the introduction of Google Voice Search [9] and Apple Siri [10] on mobile phones and wearable smart devices, is revolutionizing the way mobile devices interact with the cloud. Such services require a cloud network infrastructure that can handle large amount of data on the go, with minimum disruption or loss in the quality offered to the user.

In this paper, we focus on giving mobile users a more efficient access to their cloud applications in a distributed data-center environment making use of our controller solution, named Protocol Architecture for Cloud Access Optimization (PACAO), based on the Locator/Identifier Separation Protocol (LISP). The goal is to satisfy user's needs by improving the Cloud access network latency as a function of user mobility and user-cloud link quality by switching the entry DC in the distributed cloud fabric, and operating virtual machines at a cloud facility closer to its user.

The paper is organized as follows. Section II gives an overview of related work. Section III describes the PACAO architecture. Section IV presents experimental results. Finally, section VI concludes the paper.

II. Background

In this section we overview the state of the art on distributed DC architectures, cloud performance metrics and LISP.

A. Geographically distributed cloud architectures

The current trend in the design of cloud fabrics is to geographically distribute it over multiple DC facilities [3]. Distributing DC facilities allows, from one hand, to increase the reliability of hosted services and, from the other hand, to offer better cloud access performance to customers, thus

Submitted on Dec. 8, 2015, revised on Apr. 28 and July 12, 2016, accepted on July 14, 2016. Editor: Prof. Paolo Bellavista

Stefano Secci is with Sorbonne Universites, UPMC Univ. Paris 06, UMR 7606, LIP6, F-75005, Paris, France. Email: stefano.secci@upmc.fr.

Patrick Raad, Pascal Gallard are with NSS, 215 avenue Georges Clemenceau, 92024 Nanterre Cedex, France. Email: {praad, pgallard}@nss.fr

A preliminary version is in the proceedings of IEEE NETSOFT 2015 [1]. This work was supported by the LISP-Lab [25] (Grant No. ANR-13-INFR-0001), ABCD [26] (Grant No. ANR-INFR-0009), PODIUM (Grant No. 15016552), and FP7 MobileCloud (Grant No. 612212) projects.

decreasing the network distance between users and computing facilities. Modular DC architectures [11] have been designed and evaluated to support this evolution. The common design goal is to allow building DCs incrementally starting by regular small building blocks, grouping a few switches to interconnect a number of virtualization servers, using regular wiring [12] [13]. As opposed to legacy hierarchical architectures, modular DCs better accommodate horizontal traffic between virtualization servers in support of various IaaS operations such as VM migrations/replications and storage synchronization.

The conception of small local cloud facilities is at a good experimental and design stage today. Commonly called 'cloudlets' [**?**], [5], they can support computational offloading [14], granting high network availability and energy gains to computational intensive applications especially for mobile devices, such as for instance remote desktop or gaming applications [15]. The decision to offload application and computing tasks can be a mere remote decision or local decision taken by the device. As explained in [16], the decision making can take into account a number of metrics, including the device energy gain, the VM migration time when VM migration is needed, and other system level metrics. Less attention is devoted in [16] to network-level metrics, whose importance become higher for geo-distributed cloud deployments.

B. Cloud access performance

In our work we address the following limited set of measurable Quality of Service (QoS) goals that directly affect cloud access performance:

- *Availability*: it is a measurable metric that indicates the expected availability rate of a service accessible via a network, i.e., the probability that a service is available when a user tries to access it. It often appears as a binding service-level-agreement (SLA) related to network services, especially when the customer is a business entity that requires a very high reliability level . For DC fabric, the reference availability rates are typically 99.671% for Tier-1 DCs, 99.741% for Tier-2 DCs, and 99.982% for Tier-3 DCs [20]. For long-haul network providers, the carrier-grade network availability rate offered to business services is often higher than 99,99%, especially for critical services. Surrounding a failure affecting the access to one DC of a distributed DC architecture by automatically switching the server routing locator is therefore a desirable property of a Cloud access solution.
- *Network Latency*: it is the delay incurred in the delivery and processing of service data delivered through the network. From the area of usability engineering for legacy Internet services, the time threshold that could affect the user's perception range from a few hundreds of ms, under which the user feels that the service is no longer reacting instantaneously, to a few seconds, when the user clearly perceive the delay [21], and above which there is a risk that the user abandons the service. For more recent and forthcoming mobile services, related to augmented reality, video/voice recognition, remote desktop, network gaming, much more stringent delay requirements are expected - for instance, research on 5G systems actually targets solutions for 1 ms access latency.
- *Network Jitter*: it is the variation in the delay experienced by received packets. Due to congestion, the steady stream could become lumpy and cause packets to arrive out of order. Although the tolerance to network jitter is high, beyond a certain threshold the effects could resemble that of network loss: packets out of order could be discarded at the receiver, which directly affects user experience especially for real-time services.

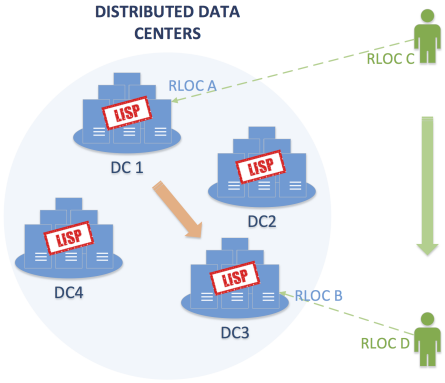


Fig. 1: Mobile cloud access reference scenario.

Our protocol architecture is such that the DC entry and VM mobility decisions are made accordingly to metrics such as the availability, the latency and the jitter that are monitored by a cloud network overlay protocol architecture.

C. Locator/Identifier Separation Protocol (LISP)

The basic idea of the Locator/ID Separation Protocol (LISP) [22] is to split the localization and identification functions of legacy IP into two IP addresses: Endpoint IDentifiers (EIDs) assigned to end-points, and Routing LOCators (RLOCs) assigned to edge routers connected to the global routing system. To separate the two namespaces, LISP uses a map-and-encap scheme: at the data-plane level, edge routers map the identifier to the locator and encapsulate the packet in another IP packet before sending it to the Internet transit network. At the control-plane level, a set of locators with different priorities and weights are affected to an EID-prefix.

A LISP site is managed by at least one tunneling router (xTR), which has typically a double functionality: ingress tunnel router (ITR) and egress tunnel router (ETR), the ITR encapsulating packets and the ETR decapsulating them. LISP has a distributed mapping system that handles EID-to-RLOC lookups, including a Mapping Server (MS) and a Mapping Resolver (MR). The mapping resolution protocol currently adopted in testbeds and commercial services is based on the Delegated Database Tree (DDT) protocol [23], which works similarly to the Domain Name System (DNS).

LISP is undergoing an increasing industrial deployment, essentially guided by Cisco Systems integration of LISP in high-end routers, and also in DC switches toward an improved management of VM mobility [24]. An independent world-wide testbed based on open source nodes also exists, coordinated by UPMC [25]. LISP enhancements in support of fast IP mobility

have been proposed, to manage both VM mobility [27] and user mobility [28]; LISP offers a more efficient and expressive framework for the users with respect to alternative solutions such as mobile IP or DNS-based solutions, thanks to lower convergence time and direct mapping update [27]. A basic proposal to use LISP for localizing VMs as a function of user mobility is also proposed in [7], [29]. This paper goes beyond those ideas and introduces a LISP QoS-aware architecture adapted for distributed mobile clouds.

III. Mobile Cloud Protocol Architecture

A reference example scenario is the one represented in Fig. 1: the user is connected to a VM located on DC 1, managed by a Cloud provider that also operates other DCs (in the figure, DCs 2, 3, 4) such that all the DCs of the Cloud fabric use a dedicated private network for inter-DC traffic. The user experience is affected by various QoS factors such as the Round Trip Time (RTT) and the jitter. Depending on whether SLA levels are respected or not, the traffic between the user and DC 1 is susceptible to be switched to the entry of other DCs. If the access DC is switched and if the VM is not located at the new access DC, the traffic is rerouted from within the cross-DC fabric to reach the VM. Eventually if the SLAs are not met or can be further improved the VM can be adaptively relocated to or close to the new access DC. Our proposal consists in defining a Cloud access protocol architecture to orchestrate and manage these Cloud access operations (i.e., adaptive DC entry switching and VM mobility). Our solution, that we named Protocol Architecture for Cloud Access Optimization (PACAO), relies on an adaptation of the LISP architecture as a Cloud access overlay protocol, and on an optimization framework to adaptively determine the best entry DC (VM RLOC) and the best VM location on a per-user basis.

A. Overlay Network Requirements and Features

We express, in the following, the requirements in the definition of a Cloud access overlay architecture, and then we justify our system design choices.

- *VM mobility*: in order to bring a VM closer to its user in terms of SLA distance, the cloud operator must be able to trigger a VM migration (VM state transfer) or relocation (active VM copy switching) across multiple sites.
- *IP Addressing continuity*: it is important to maintain user's session when user changes its attachment point (or Routing LOCator, RLOC). Keeping a session alive when changing user's IP address without changing the application is not obvious. This dually applies to a VM migrating/relocated across DCs. In fact, in absence of layer 2 continuity across IP RLOCs (access or gateway points for users, hosting virtualization server for VMs), layer 3 continuity needs to be guaranteed by forms of data-plane encapsulation able to pass through middle-boxes.
- *Access DC switching*: the user access or gateway endpoint should be able to be configured remotely by the cloud operator so that it changes the access DC toward the service VM (this logic could also be directly implemented in the mobile device [30]).
- *Cloud access link monitoring*: in order to support the decision-making related to adaptive access DC switching and VM mobility, the link between the user and its VM DC location and other possible DC locations needs to be monitored in order to collect network state metrics.
- *VM Orchestration*: based on the collected cloud access overlay link measurements, the DC access switching and VM mobility decisions need to be taken at a logically centralized controller.

Among the different overlay protocol alternatives, a few can satisfy the above requirements. Among the most promising implemented and widely adopted virtual network overlay protocols quickly reviewed in [3], only two protocols, LISP and Virtual eXtensible Local Area Network (VXLAN), offer the necessary data-plane and control-plane features.

From a data-plane perspective, both LISP and VXLAN encapsulate over UDP (in the payload, IP encapsulated packets for LISP and Ethernet encapsulated packets for VXLAN), which facilitates passing through Internet middle-boxes (which typically filter or alter TCP packets, and block uncommon encapsulation protocols not running on top of TCP or UDP); both LISP and VXLAN use a 64-bit shim header after the UDP header and before the encapsulated IP/Ethernet packet. The fact that VXLAN encapsulates Ethernet frames rather than IP packets makes it particularly appropriate to manage traffic between virtualization servers, as traffic from virtual machine to the underneath hypervisor virtual bridge is sent using virtual Ethernet network interface cards.

From a control-plane perspective, both LISP and VXLAN offer distributed signaling (working in a pull unicast mode under the LISP mapping system, or in push multicast mode in the basic VXLAN control-plane), such that (i) no central controller is required and needs to be configured to accept new nodes, (ii) addressing continuity can be guaranteed by changing the IP routing locator (called VTEP, VXLAN Tunnel End Point, with VXLAN and RLOC with LISP) of the destination IP address, and (iii) the virtual link between user and VM locators can be monitored using control-plane probing on top of existing control-plane messages.

Therefore, the PACAO prototype makes use of both VXLAN and LISP, proposing LISP as Cloud access overlay protocol as user access points often work only at the IP layer, and VXLAN as inter-DC overlay protocol as VMs of a same 'cloud' (or IaaS) natively send traffic to each other using Ethernet. The major technical reason for not using VXLAN as Cloud access protocol, is that its basic control-plane functioning, using multicast dissemination of VM positioning to VTEPs of a same VXLAN, would generate higher signaling than LISP, which uses unicast control-plane communication instead. Moreover, the major reason for not using LISP as inter-DC overlay protocol is that it would require multicast extensions to its control-plane functions to synchronize VM location states to a subset of the LISP routers (in fact, a virtual LAN) beyond which a VM could be migrated or relocated.

In our reference distributed DC fabric, each DC is at least one LISP site and has at least one LISP tunnel router

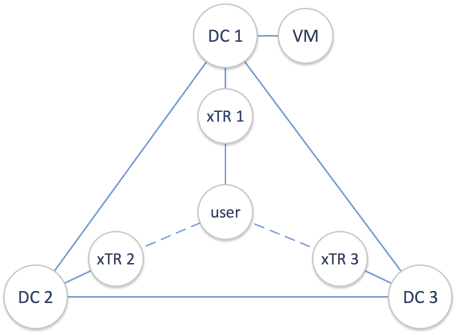


Fig. 2: Distributed DC reference functional scenario.

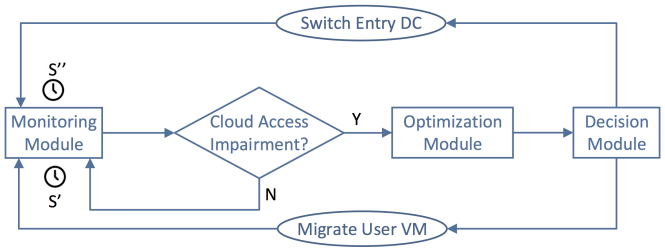


Fig. 3: PACAO policy execution chart.

(xTR). Each xTR can have one or multiple locators (RLOCs) delivered from different Internet service providers (ISPs). The VMs on each data-center typically have unique and static identifiers (EIDs). Indeed, it is possible to maintain the same EID while moving a VM from one DC to another one as it has been shown in our previous work [27]. It is assumed that a VM is also reachable through the RLOCs of the other data-centers. For example, in Fig. 1 a VM on DC 1 is reachable through RLOC A as well as RLOC B without the necessity of moving it to DC 3. Thus, an EID-prefix bound to a VM can have one or multiple RLOCs from different sites with different priorities and weights.

Cloud users can also be LISP-capable mobile nodes that have unique and static EIDs and change their own RLOC when moving across networks. By decoupling the locator from the identifier, mobile users are not tied to any geographical location from an addressing and routing perspective. In fact, in the example of Fig. 1, when a mobile user with RLOC C roams onto a new location, he receives a new locator: RLOC D.

In the proposed architecture, a logically centralized controller (possibly composed of distributed agents[1]) monitors and measures periodically the states of the user-VM link (in terms of round-trip-time, jitter, availability, etc) and decides:

- between the different RLOCs that are sent to users through the EID-to-RLOC mapping, which should have the highest priority (hence be used by users).
- If after switching to a new RLOC, it is worth moving the VM to another DC of the Cloud fabric.

Before formulating the optimization algorithm to be solved by the PACAO controller in order to take the above decisions, we describe its main modules.

B. PACAO controller

Accordingly to the above mentioned requirements and features, the PACAO controller is composed of three modules:

- *Monitoring Module*: it monitors the connection between the user and the VM, distinguishing between the user-xTR and xTR-VM sublinks. To monitor the first one, active probes periodically collect QoS metrics between VM and user's RLOCs. In order to support all possible RLOC switching and VM mobility decisions, all VM RLOCs (and not only the one that belongs to the site where the VM runs), are monitored. The probing operation is performed enhancing LISP probing; as of [22], RLOC probes are used to determine whether a RLOC is available or not; moreover, it is suggested that these probes can also be used for monitoring QoS parameters: accordingly, we implemented RLOC probing in the LIP6-LISP OpenLISP node [33], [34] to allow transporting RTT and jitter metric. To monitor the xTR-VM link, common DC monitoring tools can be used instead.
- *Optimization Module*: it implements an algorithm that solves the RLOC switching and VM mobility online optimization problem (see section III-C). When used for RLOC optimization, the agent basically takes the metrics collected from the monitoring module as input to the optimization algorithm that determines the RLOC that minimizes a fitness cost. The optimization module also maps each user's RLOC to a locators set. The latter contains all the locators of the DCs from where the VM is reachable, sorted by a QoS score: the highest score is attributed to the RLOC that respects and minimizes the SLA and the cost. The best LISP priority is then attributed to the RLOC with the highest QoS score, and so on. When this module is used to decide the VM location, the agent takes the residual capacity of the destination hosts, as well as the network information collected by the monitoring module, and then it lets the algorithm computing the best location.
- *Decision Module*: based on the optimization module solution, the decision module determines whether it is worthwhile for one or a set of target users to keep routing through the same RLOC to reach the VM. It can also decide if it is worth migrating/relocating user's VM to another DC.

Each of these modules can logically be implemented on separate nodes or on the same node; in the former case, the PACAO controller can be used by one or multiple agents. For instance, by implementing the monitoring module in xTRs we can easily interact with the EID-to-RLOC map-cache through an application programming interface (API) to probe the cached entries (RLOCs of each user). It should also be noted that separating the role of the agents into modules could allow an easier interoperability with other routing and software-defined network protocols.

[1]Mobile nodes can also feature a lightweight version of an agent that gathers statistics based on user satisfaction. This could help the cloud operator to tweak up its collected QoS data by building a database that maps user satisfaction with the location of the network as it has been proposed in [32].

C. Cloud Access Optimization

Fig. 2 depicts an example for the network model assumed in this paper. We consider three DCs, DC1, DC2 and DC3, hosting service VMs, interconnected to each other by a meshed topology. Each DC has one xTR (xTR1, xTR2, xTR3) with at least one RLOC. For the sake of simplicity, we consider that a VM is used by one client at the same time (i.e., services such as virtual workspace or virtual desktop solutions - the extension with multiple users per VM is straightforward).

As a matter of fact, cloud providers wish to operate their virtual services at a desired SLA. The user often pays the service to the providers for an agreed-upon SLA that, if not respected, can lead to monetary compensations. In this context, in order to provide the agreed-upon SLA thus minimizing the penalty, the cloud provider is reasonably interested in trying to switch the traffic of the user to another DC (by changing the priorities of the RLOCs in the EID-to-RLOC mapping), and possibly also issuing a VM migration or relocation. The Cloud access optimization problem therefore consists in minimizing the penalties that may arise from not respecting the agreed-upon SLA, taking decisions upon switching RLOC and/or moving a VM, while respecting network constraints. We achieve this goal by letting the objective of the decision algorithm to be the sum of the penalties related to possible SLA violations. This also ensures robustness against infeasible configurations, i.e., there will always be a solution including transient cases where the link and or DC performance or capacity is not enough to nullify the penalty, e.g., during failures. It is however worth noting that, in stable conditions, the objective value of stable distributed DC configurations is to be null, and that a non null objective should warn the network management system of the transient violation of the SLA, such that other traffic engineering or network planning actions can be triggered to come back to a stage where all SLAs are met.

We give in the following a polynomial-time linear programming formulation that is versatile enough to be applied for (i) the RLOC switching problem and (ii) the VM mobility problem, executed sequentially, by changing the meaning of variables and parameters. The objective is formulated as:

$$\sum_{k \in K} \alpha_k T_k \tag{1}$$

where K indicates the set of network or system SLA criteria: $k = 1$ for round-trip-time (RTT), $k = 2$ for jitter, $k = 3$ for RAM, $k = 4$ for CPU, etc. α_k is a weight that measures the importance for each criterion, such that $0 \leq \alpha_k \leq 1$ and $\sum_{k \in K} \alpha_k = 1$. T_k is an integer variable representing the penalty that needs to be minimized for criterion K.

Two important constraints apply. The first is a mapping integrity constraint: (i) the VM is only hosted at one DC at a given time, or (ii) the user uses a single RLOC,

$$\sum_{d \in D} r_d = 1 \tag{2}$$

where D is for (i) the set of the DCs that can host the VM, and for (ii) the set of RLOCs a user can redirect its traffic to. r_d is a binary variable indicating for (i) if a data-center d hosts the VM, and for (ii) if user's traffic is switched to d.

Then we need a QoS level enforcement constraint, in order not to exceed a fixed threshold or a residual capacity:

$$m_{d,k} r_d \leq M_k T_k \tag{3}$$

where $m_{d,k}$ is the measured capacity of criterion k and M_k is a residual capacity or a maximum threshold. If the problem is used to represent the RLOC switching (i), then: k can be either the RTT or jitter (or potentially any other QoS metric; please note that availability goal is implicitly enforced by the optimization operations); $m_{d,k}$ represents the measured RTT or jitter between RLOC d and the user; M_k is the maximum tolerated threshold. Note that this last constraint cannot lead to unfeasibility, as the maximum tolerated threshold is rescaled by T_k. When the problem is meant to represent the decision problem about migrating/relocating a VM to another DC (ii), then: $m_{d,k}$ represents the actual capacity of the VM such as the RAM, CPU; M_k is the residual capacity on DC d (it is straightforward to also include RTT and jitter check besides the residual capacity, when needed).

Given the single-hop nature of the cloud access overlay graph, the problem defined by (1)-(3) does not contain flow conservation constraints, and with a single bin to pack it has a polynomial complexity and can be easily solved online.

D. Online Scheduling Procedure

The above presented optimization can be triggered by the monitoring module at each arbitrary duration S' (Fig. 3) to check the supervised links. At the end of a time-slot, the PACAO controller calculates the mean (or any meaningful statistical value) over the collected statistics for the different reference QoS metrics and then run the following algorithm:

- **Step 1:** if the user-VM link measured metric means respect the QoS levels, then go to **1.1**, else go to **1.2**.
  - **1.1**: save the measured data to a database and wait until next scheduling slot $t + 1$.
  - **1.2**: run (1)-(3) as a RLOC switching problem and apply the solution via LISP.
- **Step 2:** monitor the network status between the (possible new) xTR and the VM for another arbitrary duration $S'' < S'$, with a probing interval S''' such that a sufficient number of probes can be collected over S''. If the agreed-upon SLAs are respected then go to **1.1**, else run Eq. (1)-(3) as a VM mobility optimization problem.

There are therefore three time intervals that can potentially have a high impact on the performance of the PACAO operations (see Fig. 3):

- *S'*: the interval at which the optimization and decision modules are sollicited by the monitoring module;
- *S''*: the duration during which the xTR-VM link is probed;
- *S'''*: the probing interval.

S' , S'', S''' can be determined experimentally. In the simulation section, we will perform a sensibility analysis with key values of these time components.

It should be clear that the PACAO architecture addresses both the case when a mobile user roams onto a new location

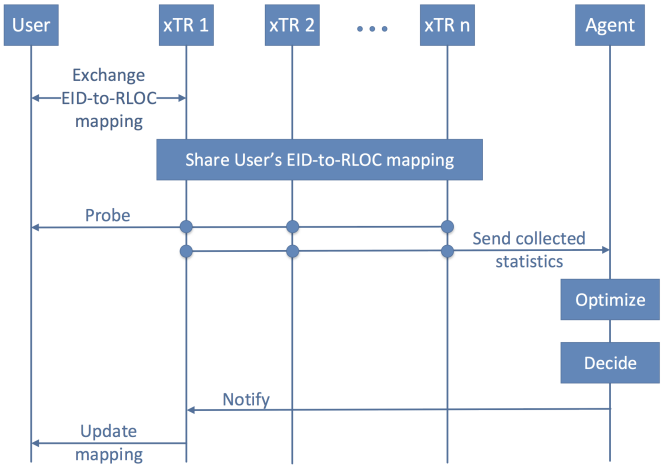


Fig. 4: PACAO sequential functional steps.

with the consequent change of the user-VM link performance, and the case of a sedentary user who could suffer from user-DC network level degradation independent of user mobility (certainly the latter situation occur less frequently than the former one). We implement the monitoring and decision modules described previously in the xTR.

Example: Fig. 4 illustrates the steps of interaction between the different elements of the PACAO architecture[2].

1) The user wants to establish a connection with a VM on DC 1. As of LISP architecture, as soon as a data-plane packet needs to be sent to the VM and if no entry exists in the local mapping-cache, a LISP MAP-REQUEST control-plane message to the mapping system is triggered to get the EID-to-RLOC mapping of the VM as described in section II-C.
2) xTR 1 replies back with an EID-to-RLOC mapping in a MAP-REPLY message. It is worth stressing that the EID-to-RLOC mapping sent to the user contains all the RLOCs of all DCs with different priorities and weights from where the service can be reached. The RLOC priorities are set by the administrator.
3) In order for the VM to communicate with the user, xTR 1 undergoes the dual signaling procedure as in Step 1. It then stores the obtained user EID-to-RLOC mapping in its map-cache, and then actively probes the user, collecting QoS metrics by RLOC probing. As discussed in the previous section, the other xTRs should also gather some metrics. However, only xTR 1 knows about the user's location. To overcome this problem, all xTRs should securely synchronize their mapping cache. Alternatively, the xTR may be data-plane only elements (e.g., OpenVSwitch nodes, which natively support the LISP data-plane) controlled by an external SDN controller (major ones already support LISP control-plane) centralizing data collection and processing.
4) At the end of a time slot (S'), xTRs send the collected QoS metrics to the optimization module where the algorithm described above is implemented. Based on the algorithm first output (RLOC switching results) the agent changes the priority of the RLOCs set in the mapping registrations, and then notifies the user's endpoint and updates the Map-Server.
5) When the agent gets the second output, if needed it triggers a VM migration or relocation to the new DC.

[2]It is worth noting that while the user is meant to be behind xTR1, the VM is meant to be directly behind at least one xTR among xTR 2 and xTR N.

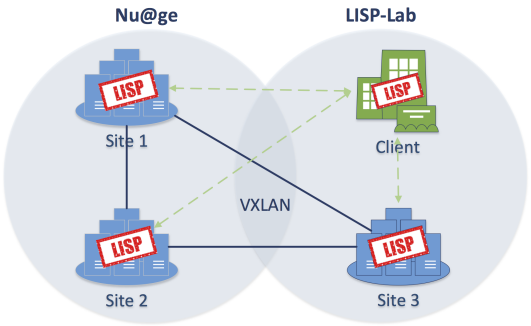


Fig. 5: Testbed network.

IV. TESTBED EXPERIMENTATION RESULTS

We implemented the PACAO architecture, using as VM mobility technology the live VM migration [27][3]. Our implementation relies on the following nodes.

xTR: we used the LIP6-LISP OpenLISP node [33], [34] as xTR software router, to which we added the RLOC probing feature using not only the basic RLOC availability mechanism but also the RTT and jitter probing logic. The xTR sends probes and collects QoS metrics between the xTR's locator and all the other locators in the mapping cache for each EID entry. The statistics are then saved in a JSON type file that is exploited by the controller. Note that the probing frequency can be changed in the OpenLISP configuration files.

Controller: we implemented the PACAO controller, its monitoring, optimization and decision modules, using Python. For the monitoring module, the controller uses the JSON file above to get the user-to-xTR and xTR-to-VM QoS metrics. We used the GNU Linear Programming Kit [35] to solve the optimization problem described in section III-C. To switch RLOCs we used an API provided by OpenLISP in order to get the EID-to-RLOC map cache, then send the CHANGE-PRIORITY message we developed for [27], and update the user's mapping cache; to migrate a VM we use the VIRSH command interface provided by the Libvirt API [36] in KVM. It is worth noting that we chose to put the controller at the hypervisor level to overcome some root restrictions; in general, the controller could be placed in xTRs or even integrated with OpenStack or alternative SDN controllers.

Our testbed is represented in Fig. 5. We used the LISP-LAB [25] experimental network, with three sites in the Paris metropolitan area network (Telcocenter DC in Courbevoie, Marilyn DC in Champs-sur-Marne, and LIP6 DC) and one

[3]As an alternative VM mobility approach to VM state migration, one could use active VM copy switching [41], i.e., multiple VM instances can be made available at different sites and VM mobility corresponds to changing the active VM by turning it on, turning the other VM off, and changing the routing maps.

in Lyon (Rezopole in Lyon-IX), in order to allow for wide VM migrations and emulate a distributed DC fabric using KVM virtualization servers at each site. Using VXLAN [31] for intra-DC fabric reachability, we meshed the 3 KVM servers acting as service VM containers: one at TelcoCenter DC, one at Marilyn DC and the last one at the LIP6 DC, enabling internal traffic redirection. The service VM is an Ubuntu 14.04 server uses a disk mounted on a network file systems. All the inter-DC links use the plain Internet, except the Marilyn-TelcoCenter one that is a dedicated 10 Gbps ethernet over fiber link, yielding to a heterogeneous testbed setting. In order to ensure the isolation with other existing services already running on the DCs, we run our experiments in a IaaS using OpenStack, connecting service VMs to the xTR via VXLAN. VXLAN ensures that the IaaS is reachable by both data-centers. We also created an FTP server on the IaaS in order to measure the user throughput.

A. Routing Locator (RLOC) switching

We first run experiments to evaluate the RLOC switching feature alone, using the TelcoCenter and Marilyn DCs only in the distributed DC fabric. The user runs an xTR located in Non Stop Systems premises behind an 8 Mbps ADSL connection with an average RTT of 35 ms with the Marilyn DC and a 37 ms with TelcoCenter DC. We used the RTT as user-VM QoS metric. We have turned our tests over a period of one month between 8:00 PM and 8:00 AM.

We compare two different cases:

- Legacy: basic configuration with fixed RLOC.
- PACAO-1: our solution limited to RLOC switching optimization feature (VM mobility disabled).

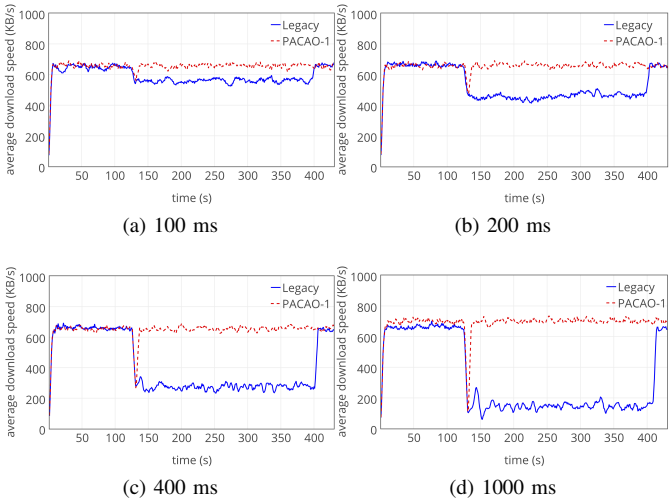
Time (s)	Perturbation actions and PACAO actions
125	Stress link between user and TelcoCenter xTR.
135	PACAO switches traffic to Marilyn RLOC.
385	Stop stressing the link.

TABLE I: First experimentation scenario time line.

The scenario is summarized in Table I:

1) the user downloads a 300 MB file from the server. It connects to TelcoCenter's xTR (whose RLOC priority is set to 1, while Marylin's is set to 2);
2) we set S' to 10 seconds;
3) after 125 seconds we artificially increase the RTT to simulate access point or network impairment between TelcoCenter xTR and the user - the injected RTT varies between 100 ms and 1000 ms;
4) the agent switches the RLOC to Marilyn xTR;
5) we stop stressing the link after 385 seconds.

Each scenario is repeated about 50 times and the following results are averages (95% confidence error bars not plotted because they are not visible). In Fig. 6, we report the average measured bandwidth for four different emulated network impairment cases: we inject delays on the user-RLOC link of 100 ms, 200 ms, 400 ms, 1 s. We notice that for the legacy



(a) 100 ms (b) 200 ms

(c) 400 ms (d) 1000 ms

Fig. 6: Average download speed for different RTT increase impairments on the user-VM link (scenario of Table I).

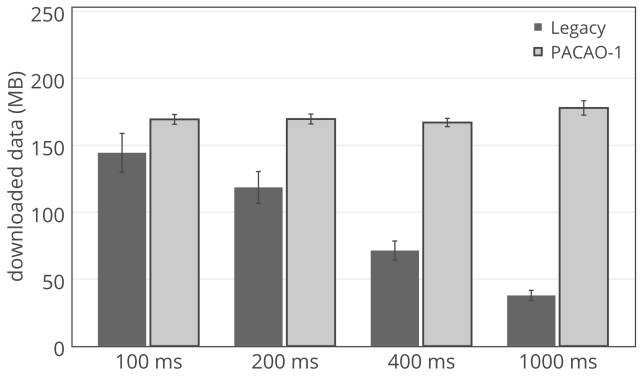


Fig. 7: Total amount of data downloaded in 450 s

case the bandwidth decreases drastically at 125 s. However, in PACAO-1, thanks to the adaptive RLOC switching the bandwidth is quickly restored. It is worth stressing that it takes $S' = 10$s, plus one half of RTT in order to update the map cache of the user with the CHANGE-PRIORITY message [27]. It is worth mentioning that, from the viewpoint of the user, the routing locator switching performed by PACAO-1 could happen almost unnoticed; it can be expected that Quality of Experience (QoE) ratings (see [17]–[19]) would remain stable for low user-VM link latency increases, at least for situations like for case a) in Fig. 6.

In Fig. 7 we report the downloaded volume for the different cases, during 450 s. We clearly see that with PACAO-1 we maintain roughly the same downloaded volume whatever the importance of the network impairment is, improving the download rate by 80%. Moreover, we can notice that the variance is lower with PACAO-1, hence one could expect that QoE levels would be maintained, as drops are prevented and the variability is reduced.

B. RLOC switching and VM mobility

We then run the complete PACAO solution, this time using all DCs with the user host running as a xTR VM on the KVM

server of the Rezopole DC site (this allowed us to run the simulations more frequently during the day and night over a more reliable connection).

With respect to the previous experiment, for which we only monitored the user-xTR latency, we did also monitor this time the RTT on the xTR-VM link too: the sum yields to the global user-VM latency. Note that in order to trigger a VM migration after switching the RLOC we have overloaded the inter-DC links. As a result we generate high load of traffic that is able to jam the bandwidth and the delay.

For a complete statistical measure we have used IPERF for generating traffic, simulating both TCP and VoIP connections and opposing three different cases:

- Legacy situation.
- PACAO-1: PACAO with only RLOC switching.
- PACAO-2: PACAO including VM mobility.

These scenarios are conducted at night between 8:00 P.M. and 8:00 AM, and are repeated 100 times in order to get statistically robust results.

Time (s)	Perturbation actions and PACAO actions
50	Stress link between user and TelcoCenter xTR.
60	PACAO switches traffic to LIP6 RLOC.
67	PACAO migrates VM to LIP6 site (16 s).
187	Stress link between user and LIP6 xTR.
198	PACAO switches traffic to Marilyn RLOC.
204	PACAO migrates VM to Marilyn site (27 s).

TABLE II: Second experimentation scenario time line.

We then apply to each case the steps in Table II:

1) we start IPERF from both the user and the VM;
2) after 50 s we stress the link between the user and TelcoCenter xTR; while for the legacy case nothing changes, for the PACAO cases the controller monitors the link for S' = 10s and takes the decision to switch the traffic to LIP6 RLOC at 60 s.
3) after S'' = 5s, the controller runs only for PACAO-2 case the VM mobility optimization (based on statistics collected between each xTR and the actual position of the VM, TelcoCenter DC), and then issue a VM migration to the LIP6 DC;
4) we repeat the same operation after 120 s, so that PACAO first switches the traffic and then migrates the VM to the Marilyn DC.

As depicted in Fig. 8 we run the experimentation for the three cases. One should notice that, even after switching the RLOC (PACAO-1), the user connection is heavily affected due to inter-DC overloaded links. To overcome the problem, with PACAO-2, the controller migrates the VM as well (besides 'migrating' the DC entry point) in order to prevent traffic redirection. It can be noticed that Fig. 8 reports a time lapse of a few seconds between the performance increase with PACAO-1 and the one with PACAO-2 exists: this time is the time needed to migrate the VM states from the source server to the destination server, as described in our previous work [27]. The measured data in the boxplots (minimum, 1^{st} quartile, median, 3^{rd} quartile, maximum) of Fig. 9, shows we can gain up to 40% by switching the RLOC then migrating the VM in the above described use case.

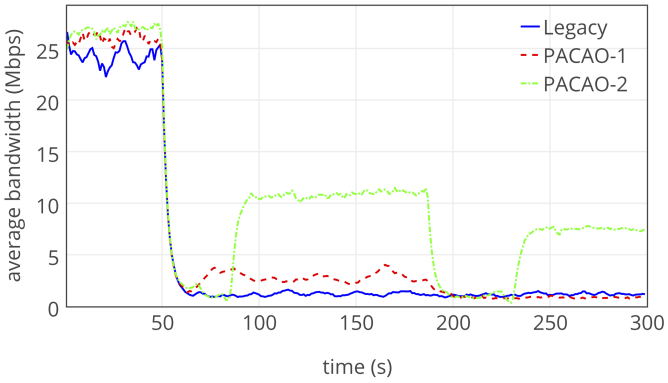


Fig. 8: Average bandwidth given the scenario of Table II.

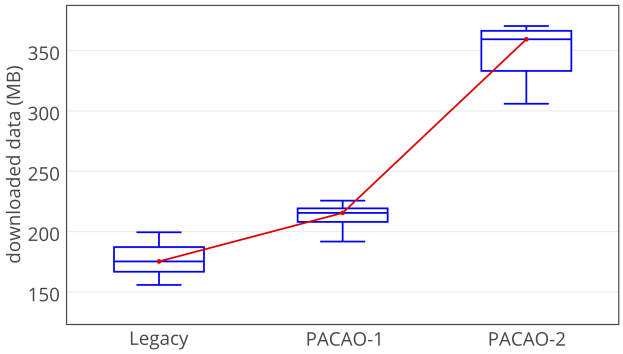


Fig. 9: Total amount of data downloaded in 300 s.

One concern is whether the proposed PACAO adaptive VM migration is suitable for real-time services, sensible to packet loss and jitter. Fig. 10 shows the experienced packet-loss ratio during real-time UDP-based streaming traffic. We get, with the provided long-distance setting, a median of 4% packet loss, which can be considered as marginal, also considered that other advanced techniques are available in commercial products to practically nullify the loss by means of triangulation. The jitter results of these simulations are represented in Fig. 11, which shows no noticeable difference between the two PACAO cases that offer more stable performance (lower dispersion around the median) than the legacy solution.

It is worth noticing that when the time to migrate the VM states is not negligible, i.e., for memory-hungry VMs and applications, that time could be taken into account in the VM migration decision logic in order to avoid unnecessary or counter-productive VM migrations. This can be particularly relevant in use-cases when the VM state migration time could be at the same order of magnitude than the time between two VM migration decisions [4]. For instance, this can be the case for forthcoming metropolitan Mobile Edge Computing (MEC) use-cases where small-scale/micro user mobility could be used to trigger VM migrations, such that the time for a

[4]It is worth noting that in the case VM mobility is implemented using VM replications [41], the migration time would be negligible as merely corresponding to the live memory dirty page transfer time.

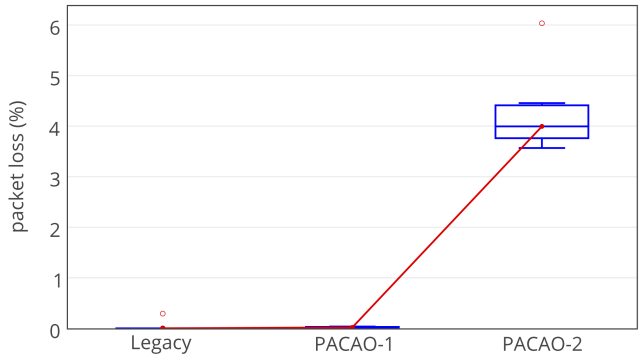


Fig. 10: Percentage of lost packets.

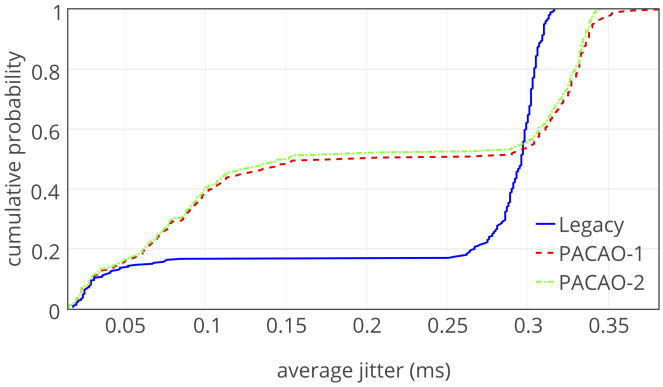


Fig. 11: CDF of the average jitter.

user to displace from a MEC delivery node to another one is comparable to the time needed to migrate the VM states. The VM state migration time can instead be considered as negligible in large-scale/macro user mobility use-case such as ones considering a geo-graphical distribution of computing facility on distances that take a long time to travel. This latter use-case is the one considered in the following analysis.

V. Evaluation using Real Mobility Traces

To better understand how our framework would perform in real deployments, we extracted real cellular user's large-scale mobility traces and evaluate the PACAO impact with different DC geographical distributions. Before detailing the emulation environment, we first provide a description of the datasets.

A. User mobility traces and data-center distribution

Via a French collaborative research project (ABCD [26]), we could access large-scale 4G user mobility traces from the Orange mobile network, made anonymous in their identifiers and in the precise locations. Positioning was indeed at the LAC (Local Area Code) level, a LAC being a macro-region ranging from few kilometers to dozens of kilometers of radius, depending on population density. Moreover, only the LAC-level trajectories shared by at least 2 users are extracted (accordingly to [37]). The data comes from network management tickets, collecting LAC-level positioning and application volume information on a per-session basis, every 6 minutes. We used mobility traces of one single day, giving a total of 2.6 millions useful user entries. Applying 2-anonymity aggregation, and then keeping only those trajectories crossing more than 3 LACs, we get the 36,450 trajectories we used for our simulations.

About the DC geographical distribution, we adopted a gravitational distribution as a function of the user density - this is practically implemented as a uniform distribution over the LACs, LACs having a geographical coverage that is inversely proportional to user density. Each LAC being the possible location of a DC, with no more than one DC per LAC. In the following simulations, we vary the DC distribution from 1 to 64 over the France and the Europe territories. The France-wide DC distribution can be considered equivalent to a deployment of the PACAO architecture into national ISP networks directly offering IaaS services to their users. The Europe-wide DC distribution, instead, can be considered equivalent to the situation of an over-the-top (OTT) IaaS provider, transparently interconnected to ISPs and orchestrating resources based on collected measurements. Accordingly, we refer to the two use-cases as 'France/ISP' and 'Europe/OTT' use-cases.

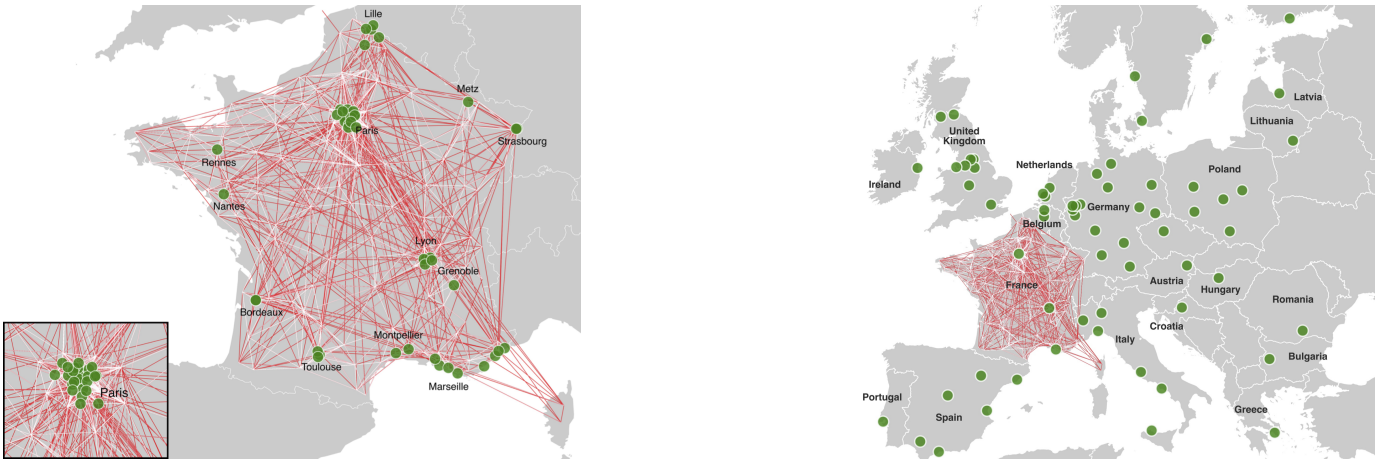
Based on a given DC topology for each simulated instance, we computed the round-trip time (RTT) between DCs, and between users and DCs, as follows. The baseline propagation delay is computed as the rough RTT one would get using a fiber link interconnection (speed of light at 200 000 km/s leading to roughly 1 ms each 100 km) over the Euclidean line connecting the nodes; the RTT is then computed as 3 times the propagation delay (higher than 2 times as the real physical path has to be in fact longer than the direct Euclidean one). For the connection between the user and the DC, we shift the values by 40 ms to reproduce well-known bufferbloat anomalies in todays 4G networks.

Fig. 12 represents the 64 DC locations as well as the mobility traces. Fig. 13 reports the resulting distribution of used RTTs, between user and DCs and between DCs, for the France/ISP and Europe/OTT cases.

B. Simulation Results

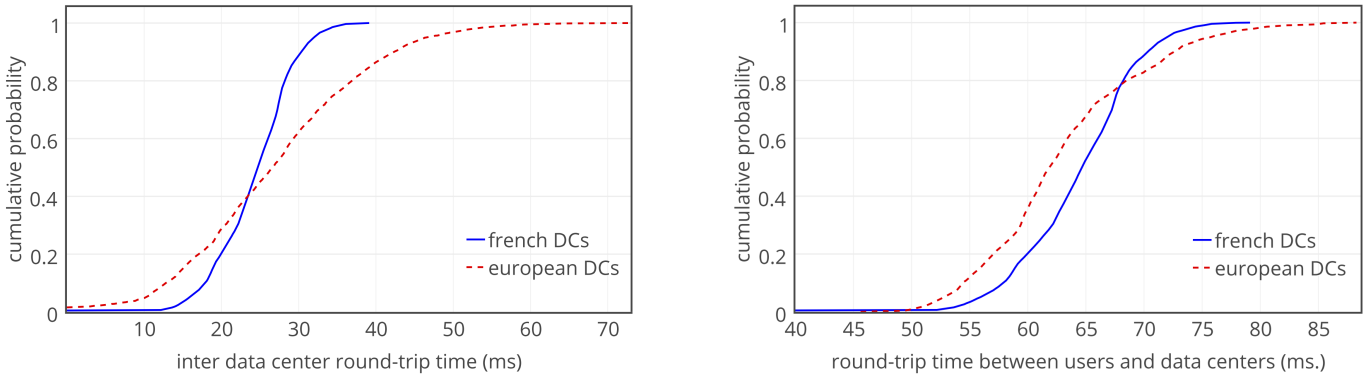
Based on the collected dataset, we wrote a simulator to mimic PACAO functionality and estimate the RTT between users' geographical position and the DCs. The simulator is in python and uses CPLEX as solver.

We want to first determine the regime under which our proposal offers the best results, i.e, which DC distribution and which time-slot (S', S'') setting shall be used for real deployments. As the PACAO policy impact strongly depends on the QoS bounds (3), we evaluated different latency bounds: the 1st decile (strict bound), the median decile bound, and the 9th decile (loose bound) taken from the distributions in Fig. 13. We run it for each of the following DC distributions: 2, 4, 8, 16, 32, 64 DCs. In Fig. 14, 15, we plot the offered median RTT as a function of S' and S'' (see sect. III-D), for the three latency bounds, for the France/ISP and the Europe/OTT use-cases. The range for S' goes from 10% to 90% of the residence time for each user in each LAC. As already mentioned, $S'' < S'$ to ensure fast VM migration decisions with a minimum quality loss - for a given S', S'' ranges in $[0.1S', 0.9S']$.



(a) France/ISP use-case (b) Europe/OTT use-case

Fig. 12: Representation of the employed trajectories and data-center distribution.



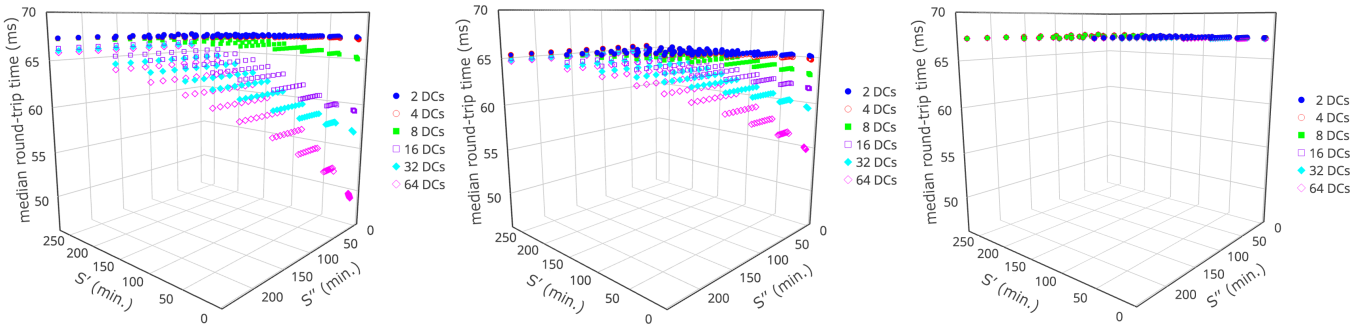
(a) Inter data-center links (b) Users to data-center links

Fig. 13: Emulated round-trip-times between network nodes.

First, we notice that under the loose bound, the results are uncorrelated from the values assigned to S' and S''. Furthermore, we can notice that:

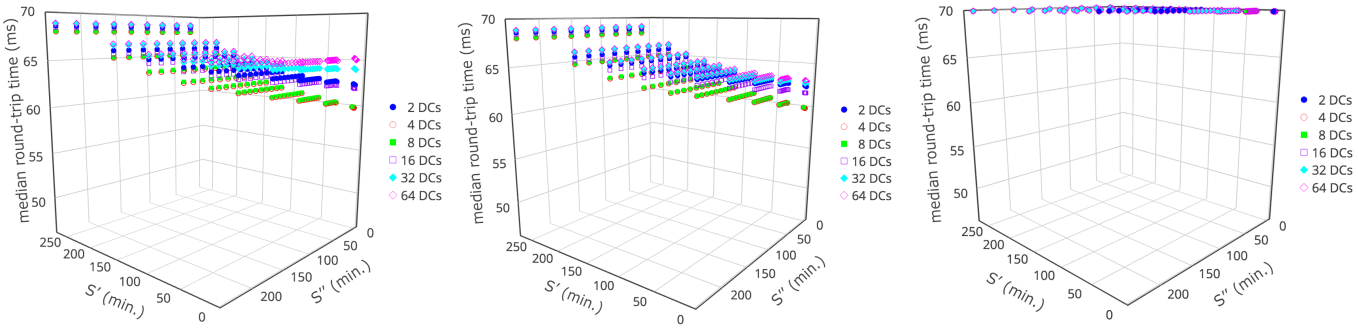
- Passing from the median to the strict latency bound, a decrease from 5% to 10% in the median RTT can be observed in the France/ISP case especially with a high number of DC sites. The reason is that 90% of the customers (Fig. 13b) have a user-DC latency that is greater than the strict bound, i.e., most of the users can first change their access gateway before migrating their VM while they are moving. This is less evident for the median and loose latency bounds because only 50% and 10% of users (loose latency bound) have a user-DC latency higher than the bound, respectively. This phenomenon is less important for the Europe/OTT case, for which a smaller RTT reduction can be noticed only for some DC distribution cases.
- While starting with 16 DCs the S' and S'' values have a significant impact on the RTT for the France/ISP case, for the Europe/OTT case we have better results for the 4-DC and 8-DC distributions. Indeed, in the former case the user mobility is confined within the French frontier and there is no roaming across European cities. Intuitively, when the latency bound plays a relevant weight, the less time we take to act the better the performance is.
- For the France/ISP case, we observe a latency improvement for low S', S'' combinations for a distribution of 16, 32 and 64 DCs. By dispersing the DCs more in France (Fig. 12a), we assign to mobile users access gateways closer to their locations, thus providing higher availability as well as 30% gain in the cloud access latency.
- The lower impact of S', S'' setting for the Europe/OTT case than for the France/ISP case is mostly due to the higher distance of the DCs in Europe for French mobile users. However, we notice that we have a 15% of latency gain (strict and median latency bound) for a DC distribution of 4 and 8 which corresponds to countries abutting France.
- Overall, we get the best performance with 64 DCs for the France/ISP case and 4 DCs for the Europe/OTT case.

Fig. 16 provides the maximum estimated TCP throughput on a lossless path between legacy architecture and PACAO framework, computed dividing the typical TCP window size (163840 bytes) by the RTT [38]; therefore, it does not capture



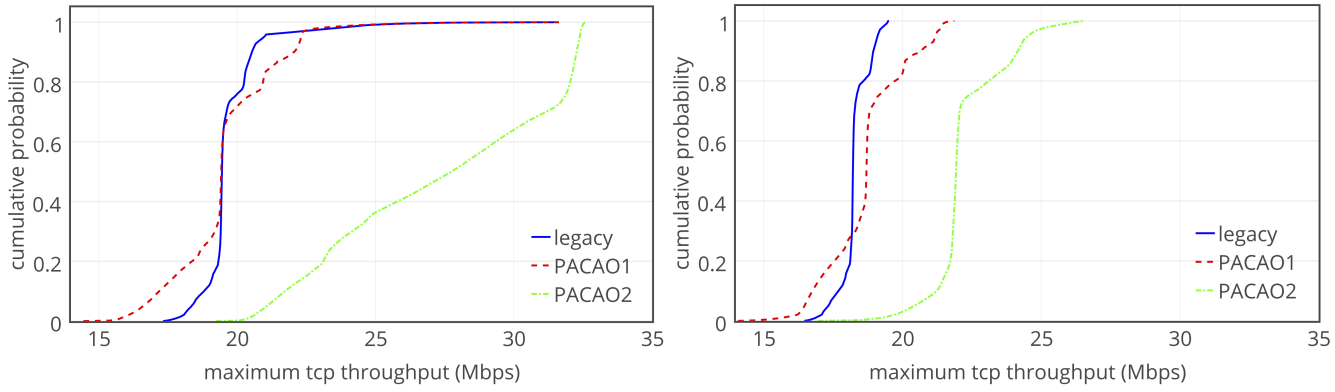
(a) Strict latency bound (58 ms) (b) Median latency bound (66 ms) (c) Loose latency bound (70 ms)

Fig. 14: France/ISP: round-trip time results as a function of PACAO S' and S'' time-slots values.



(a) Strict latency bound (54 ms) (b) Median latency bound (62 ms) (c) Loose latency bound (72 ms)

Fig. 15: Europe/OTT: round-trip time results as a function of PACAO S' and S'' time-slots values.



(a) France/ISP (b) Europe/OTT

Fig. 16: Maximum estimated throughput (Mbps).

packet loss, but we believe it is sufficiently realistic for long-lasting file transfers (or 'elephant' flows) such as storage applications (it is less realistic for 'mice' quick flows such as related to web browsing). We consider the best S', S'' configuration 6', 36" with 64 distributed DCs for the France/ISP case and 4 DCs for Europe/OTT case. We separate the PACAO1 step from the PACAO2 step to better show the impact of VM migration. Besides switching the RLOC (PACAO1, which grants limited bandwidth gain to some users), linking VM mobility to user mobility grant a median gain of roughly 40% for the France/ISP case and of roughly 30% for the Europe/OTT case.

C. Dealing with Cloud Access Congestion

Our cloud access optimization formulation proposed in section III-C can be extended to take into consideration the work load on each DC. This can be needed when the gap between global users' traffic potentially entering in a DC is on a comparable scale than the DC access link capacity. In such cases, congestion delay, packet loss and DC unavailability can manifest if too many users are concurrently being assigned to a same DC. For such traffic congestion situation, link latency can be expressed by a piece-wise function [39], as depicted in Fig. 17. We provide in the following a reformulation of the optimization problem to deal with cloud access congestion.

1) Reformulation: The previous formulation needs to be adapted to take into consideration many users at once, instead of a single one. The decision needs no longer to be where to dispatch a user and where to migrate a user's VM, but how to take these decisions for a group of users, simultaneously. The bin-packing problem hence becomes more challenging (NP-hard); therefore, we foresee in the new formulation a math-heuristic approach to iteratively take a subgroup of users, so that by setting the size of each subgroup we can control the time complexity. The new formulation is as follows. The new notation is described in Table III.

$$min\ T = \sum_{u \in U} T_u \tag{4}$$

where U is a set of users, T is the global penalty. Four constraints apply. The first is an integrity constraint - a user can only have one connection to one DC:

$$\sum_{d \in D} r_u^d = 1 \quad \forall\ u \in U \tag{5}$$

The number of users on a DC does not exceed a maximum number defined by the operator:

$$\sum_{u \in U} r_u^d + e^d \leq C^d \quad \forall\ d \in D \tag{6}$$

where e^d is a parameter that represents the number of existing connection (users) on a DC d. The global latency on a DC is set following a piece-wise function of the load:

$$l^d \geq f_i^d\Big(\frac{1}{C^d}\Big(\sum_{u \in U} r_u^d + e^d\Big)\Big) - L_{max}(1 - \sigma_i) \quad \forall i \in I,\ \forall d \in D \tag{7}$$

$$\sum_{i \in I} \sigma_i = 1 \tag{8}$$

where l^d is the additional latency on a DC d. L_{max} is very large integer number. I is the set of piece-wise steps. Using piece-wise latency functions, computed via (7), is a common practice in IP network management since seminal works such as [40]. The last constraint is to push to meet the latency SLA:

$$l^d + l_u^d \leq LT_u \quad \forall\ d \in D, u \in U \tag{9}$$

Similarly, additional additive SLA constraints can be applied, for instance on the packet loss, etc.

The time-complexity can be controlled by dividing the users into sub-groups, executing the optimization iteratively for different groups, by increasing e^d whenever users are added to a DC d. The number of users to treat in each single batch has to be properly determined in order to have the find an acceptable execution time bound for a given setting.

2) Simulation Results: In Fig. 18 we report the CDF of the RTT under (4)-(9), for different DC distributions and for the two use-cases. We set the maximum latency bound to the strict latency bounds previously adopted, and used the piece-wise function of Fig. 17. We set the congestion latency component scale as of Fig. 17 so that it has a non negligible impact on user's latency. The maximum capacity of each DC is set to twice the bandwidth required to collect the portion of traffic

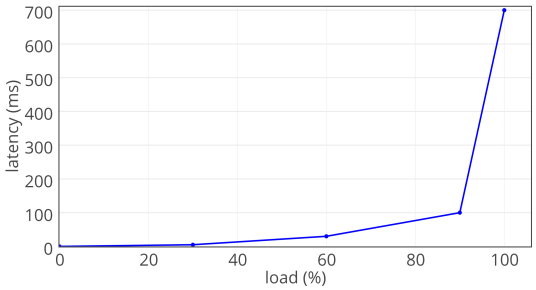


Fig. 17: Piece-wise latency function of the link load.

Parameters	
C^d	maximum capacity for DC $d \in D$
L_{max}	maximum allowed round-trip time
$f_i^d(x)$	i-th latency parameter function on DC $d \in D$ depending on its access link load x
e^d	number of existing users on DC $d \in D$
l_u^d	latency between user $u \in U$ and DC $d \in D$
Binary variables	
r_u^d	1 if user $u \in U$ is connected to DC $d \in D$
σ_i^d	1 if the i-th latency function $f_i^d(x)$ is activated
Non-negative continuous variables	
T_u	≥ 1, penalty of user $u \in U$
l^d	global latency on DC $d \in D$

TABLE III: New variables and parameters.

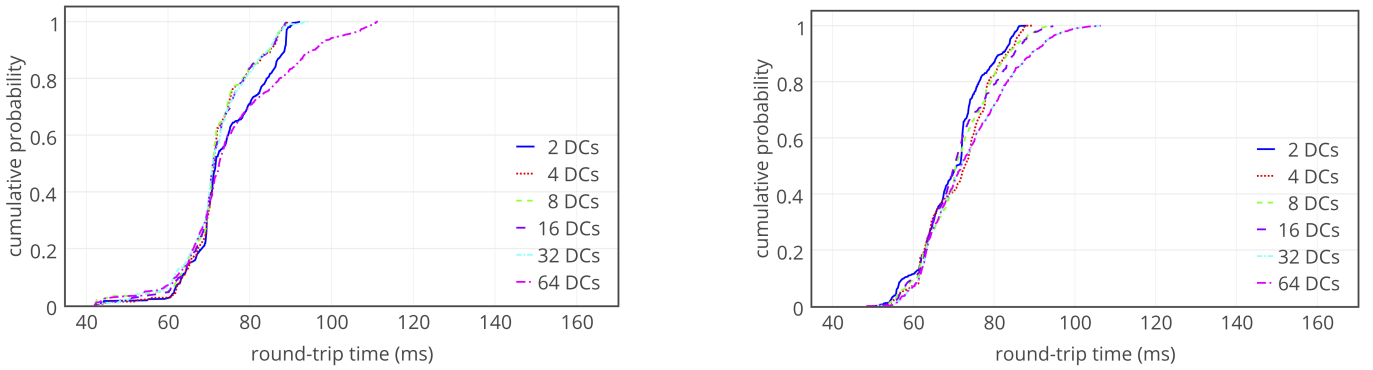
in case of equal traffic proportioning over all available DCs, so that each DC cannot be overloaded attaining 700 ms.

The main result that derives from Fig. 18 is that, under congestion-awareness, the number of deployed DCs do not matter that much any longer, as far as their access capacity is appropriately dimensioned to accommodate all traffic (as it should be for real deployments). Indeed, the CDFs overlap for almost all the cases except with 64 DCs, where 50% of the users have a RTT that is greater than 75 ms.

We depict in Fig. 19 how users get dispatched on the distributed fabric, for the 64 DC France/ISP and 4 DC Europe/OTT cases, as compared to the case without congestion awareness. The load is better balanced over all the available DCs under congestion awareness. In the Europe/OTT case, this leads to higher load on the eastern DCs, as the western DCs were capturing most of the traffic.

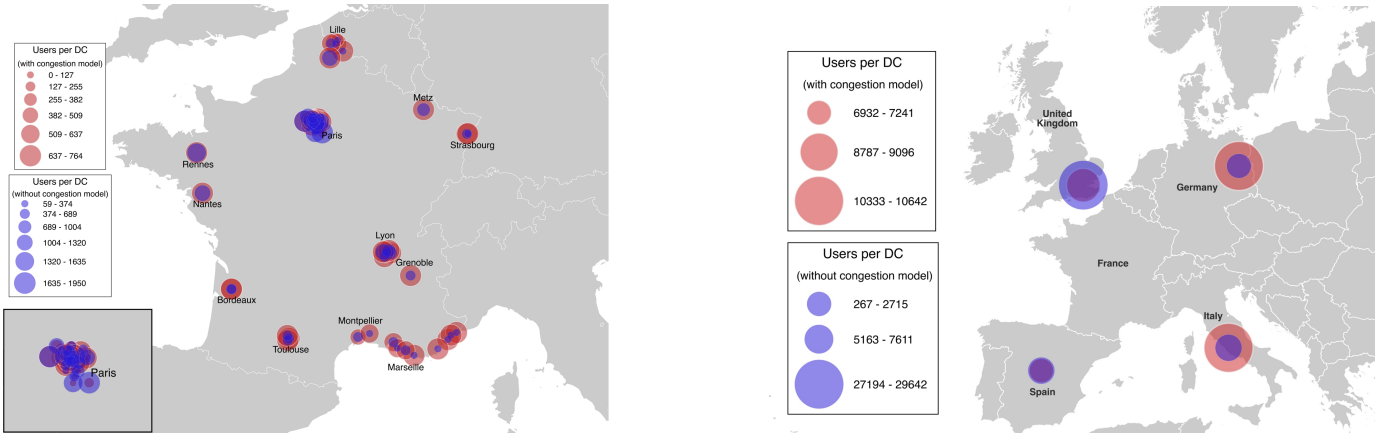
VI. Summary and Future Work

A major concern in mobile cloud networking is finding automated and transparent ways for smart virtual machine mobility taking into account major user displacements, in a cloud network context where the data-center fabric is geographically distributed. Should the user get too far away from its infrastructure as a service (IaaS) virtual machine resources, open challenging research questions are to determine if and how it would be beneficial (i) to move its virtual machines to a closer data-center virtualization facility nearby, (ii) to just switching its data-center routing locator (entry point of the distributed cloud fabric), (iii) to perform both previous operations, and finally (iv) how to technically perform these



(a) France/ISP use-case (b) Europe/OTT use-case

Fig. 18: RTT distribution under congestion-aware optimization for different DC distributions.



(a) France/ISP use-case (b) Europe/OTT use-case

Fig. 19: Distribution of users over DCs with and without congestion-aware scheduling.

operations transparently with respect to the underlying network infrastructure.

In this paper, we designed a cloud access overlay protocol architecture able to support and perform these tasks. It includes a controller able to compute and instantiate, in an online fashion, new user-to-data-center flow assignment as well virtual machine mobility across data-centers. The cloud network overlay is based on marginal improvement of the LISP protocol, which is able by means of IP-over-IP encapsulation to support live virtual machines across distant data-centers. The routing decisions taken by the controller are triggered by changes in the network states, such as increased latency between user and its VMs, and can be run on a per-user fashion or grouping multiple users in the same time to master crowd congestion effects.

We evaluated our proposal with two different methodologies, on a real geographically distributed testbed including four distant data-centers in France, and by simulation using large-scale mobility traces from a major mobile Internet service provider.

The testbed results show that the gain in terms of throughput can be very important, more than double with respect to the legacy situation with a fixed data-center entry point and no adaptive virtual machine mobility. A very high marginal gap is brought by adaptive virtual machine mobility, while data-center routing locator switching alone can grant a less important, yet high, gain. The simulation run using 36,450 anonymized real user mobility traces across the whole France, and simulating different distributed data-center topologies ranging from 4 sites to 64 sites over both France and Europe scales to simulate the use-case of a national ISP and of an OTT cloud provider. The results show a first counter-intuitive result that the largest gain can be granted with many (64) DCs for the France/ISP use-case and with a few (4) DCs for the Europe/OTT case. The gains in terms of median throughput are as high as 40% for the France/ISP case and 30% for the Europe/OTT case with respect to the legacy situation.

Our results are very promising and positively support the technology shift in mobile cloud networking represented by linking virtual machine mobility to user mobility. We proved by both testbed experimentation and simulation results against real traces that disposing of a distributed data-center fabric can bring to major benefits for the users in terms of throughput and cloud access/retrieval latency. Such networking situations are particularly appealing as they also offer higher availability and path diversity to users.

As a future work, we plan to investigate how the usage of multipath transport protocols can grant additional performance gains to the user, and the data-center access network provider. Moreover, an interesting possible further work is to conduct a QoE rating experimentation campaign to asses the performance of the proposed solutions with a perspective closer to Cloud users than to network managers.

REFERENCES

[1] P. Raad, S. Secci, C. D. Phung, and P. Gallard, "PACAO: Protocol Architecture for Cloud Access Optimization," in *Proc. of 2015 IEEE 1st Int. Conference on Network Softwarization (IEEE NETSOFT 2015)*, Apr. 2015.

[2] Gartner Cloud Computing Forcasts Update. [Online]. Available: http://www.gartner.com/newsroom/id/2613015/

[3] S. Secci and S. Murugesan, "Cloud Networks: Enhancing Performance and Resiliency," *Computer*, vol. 47, no. 10, pp. 82–85, 2014.

[4] M. Satyanarayanan *et al.*, "The Case for VM-Based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[5] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet Network Design Optimization," in *Proc. of IFIP Networking*, 2015.

[6] "Mobile-Edge Computing ? Introductory Technical White Paper," ETSI MEC Industry Specification Group 1, 2014.

[7] S. Secci, "Cloud and Mobility: a Castling Move?", *Global Security Magazine*, Avril 2012 (online). http://www.globalsecuritymag.fr/Stefano-Secci-LIP6-consortium,20120426,29908.html.

[8] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[9] Google Voice. [Online]. Available: https://www.google.com/voice/

[10] Apple Siri. [Online]. Available: https://www.apple.com/ios/siri/

[11] J. Hamilton, "Architecture for modular data centers," *arXiv preprint cs/0612110*, 2006.

[12] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[13] C. Guo *et al.*, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[14] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[15] L. Jiao, R. Friedman, X. Fu, S. Secci, Z. Smoreda, and H. Tschofenig, "Challenges and Opportunities for Cloud-based Computation Offloading for Mobile Devices," in *Proc. of Future Network & Mobile Summit 2013*. IEEE, Jul. 2013.

[16] A. Ravi and S. K. Peddoju, "Handoff Strategy for Improving Energy Efficiency and Cloud Service Availability for Mobile Devices," *Wireless Personal Communications*, pp. 1–32, 2014.

[17] T. M. O'Neil, "Quality of experience and quality of service for IP video conferencing," Tech. Rep., 2002.

[18] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[19] Qualinet White Paper on Definitions of Quality of Experience. Output from the fifth Qualinet meeting, Novi Sad, March 12, 2013.

[20] W. Turner, J. H. Seader, and W. Renaud, "Data center site infrastructure tier standard: Topology," *Uptime Institute*, 2010.

[21] A. Bouch, A. Kuchinsky, and N. Bhatti, "Quality is in the eye of the beholder: meeting users' requirements for Internet quality of service," in *Proc. of ACM CHI*, 2000.

[22] D. Farinacci *et al.*, "The locator/ID separation protocol (LISP)," RFC 6830, Jan. 2013.

[23] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree," draft-ietf-lisp-ddt-07, 2016.

[24] Cisco, "Locator ID Separation Protocol (LISP) VM Mobility Solution," Tech. Rep., 2011.

[25] ANR LISP-Lab Project. [Online]. Available: http://www.lisp-lab.org

[26] ANR ABCD Project. [Online]. Available: https://abcd.lip6.fr

[27] P. Raad *et al.*, "Achieving Sub-Second Downtimes in Large-Scale Virtual Machine Migrations with LISP," *Network and Service Management, IEEE Transactions on*, vol. 11, no. 2, pp. 133–143, 2014.

[28] A. Galvani *et al.*, "LISP-ROAM: network-based host mobility with LISP," in *Proc. of ACM MobiArch*, 2014.

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

[30] LISPmob. [Online]. Available: http://www.lispmob.org

[31] M. Mahalingam *et al.*, "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks," RFC 7348, 2014.

[32] T. Taleb and A. Ksentini, "QoS/QoE predictions-based admission control for femto communications," in *Proc. of IEEE ICC*, 2012.

[33] LIP6-LISP open source implementation (website): https://giithub.com/lip6-lisp.

[34] D. C. Phung *et al.*, "The OpenLISP control plane architecture," *IEEE Network Magazine*, vol. 38, no. 2, pp. 34–40, 2014.

[35] GNU Linear Programming Kit. [Online]. Available: https://www.gnu.org/software/glpk

[36] LibVirt virtualization API. [Online]. Available: http://www.libvirt.org

[37] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 1–18, 2008.

[38] B. Constantine, G. Forget, R. Geib, and R. Schrage, "Framework for tcp throughput testing," RFC 6349, IETF, 2011.

[39] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in internet-like environments," in *Proc. of ACM SIGCOMM 2013*.

[40] B. Fortz, M. Thorup, "Internet traffic engineering by optimizing OSPF weights". In *Proc. of IEEE INFOCOM 2000*.

[41] B. Cully, et al, "Remus: High availability via asynchronous virtual machine replication', In *Proc. of USENIX NSDI 2008*.

Stefano Secci is an Associate Professor at the University Pierre and Marie Curie (UPMC - Paris VI, Sorbonne Universites), France, since 2010. He received a "Laurea" degree in Telecommunications Engineering from Politecnico di Milano, in 2005, and a dual Ph.D. degree in computer networks from the same university and Telecom ParisTech, France, in 2009. He is the current Chair of the Internet Technical Committee (ITC), joint between the IEEE Communication Society and the Internet Society (ISOC), since 2013. His works mostly cover network optimization, protocol design, Internet routing and traffic engineering. More information can be found at http://lip6.fr/Stefano.Secci.

Patrick Raad obtained a computer science degree from the Lebanese University in 2011, and the M.Sc. and Ph.D. degree in networking from UPMC, France, in 2012 and 2015, respectively. He is currently senior researcher at Non Stop Systems (NSS), France. His current interests include Internet routing and Cloud Networking.

Pascal Gallard obtained a computer science degree from Rennes University in 2001, and a Ph.D. degree in computer from INRIA and Rennes University in 2004. He was the cofounder of Kerlabs, a company on system virtualization, where he worked from 2006 to 2010. Since 2011 he is a research and development director at Non Stop Systems (NSS), an SME on Cloud computing and virtualization.