PACAO: A Protocol Architecture for Cloud Access Optimization in Distributed Data Center Fabrics

Patrick Raad*†, Stefano Secci*, Chi-Dung Phung*, Pascal Gallard†
* Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France. E-mail: first-name.last-name@upmc.fr
† Non Stop Systems (NSS), 56 boulevard du Courcerin, 77183 Croissy Beaubourg, France. Email: {praad, pgallard}@nss.fr

***Abstract*—In spite of their rapid growth, cloud applications still heavily rely on the network communication infrastructure, whose stability and latency directly affect the quality of experience. In fact, as mobile devices need to rapidly get real-time information and files from the cloud, it becomes an extremely important factor for cloud providers to deliver a better user experience. In this paper, we specify a cloud access overlay protocol architecture, based on traffic engineering extensions of the Locator/Identifier Separation Protocol (LISP), to improve the access performance for Cloud services delivered by a distributed data center fabric. The distributed fabric offers the possibility to access the services through multiple routing locators and to migrate server virtual machines (VMs) to different locations improving access performance. We address the problem of jointly switching VM routing locators and migrating VMs across data-center sites. We propose an adaptive control framework that allows satisfying agreed-upon levels of quality of service. We evaluate the architecture on a real distributed data-center network, involving four distant LISP-enabled data-center sites in France, as compared to legacy situations with no Cloud access optimization. By emulating realistic situations we show that, by only switching the data-center routing locator, we can guarantee a better user experience with a transfer time decreased by 80%. Moreover, we show that, to react to situations when the Cloud access link between sites is disrupted or suffers excessively from packet loss, the adaptive VM migration policy can further decrease the transfer time by 40%.**

I. Introduction

Cloud computing has witnessed a rapid growth over the last decade, with companies of all sizes 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 [1]. While cloud operators offer a plethora of applications and services (e.g., web hosting, storage services), end-users still have limited control over hardware and software resources. Currently, there is a clear demand to overcome this limitation and extend users' control beyond the cloud operator boundaries.

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 DC facilities. On the one hand, a recent study in [2] shows that about 25% of collocation data-centers (DCs) have three or more sites, and that about 5% have more than 10 sites. On the other hand, the so-called cloudlet solution ([3], [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 [5].

Cloud applications are increasingly accessed on the move: when the distance 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 migrate services (server virtual machines) to the closest data-center according to user's movement [6]. 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 [7] and Apple Siri [8] on mobile phones and wearable devices such as the Google Glasses [9], is revolutionizing the way mobile devices interact with the cloud. This type of services requires 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 Protocol Architecture for Cloud Access Optimization (PACAO) solution based on the Locator/Identifier Separation Protocol (LISP). PACAO's 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 through two actions: switching the entry data-center in the distributed cloud fabric, and migrating 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 V 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 multiple DC facilities [2]. 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. Many modular DC architectures have been designed and evaluated to support this evolution. The common design goal is to allow building DCs incrementally starting by regular small basic building blocks, grouping a few switches to interconnect a number of virtualization servers, using regular wiring schemes [10] [11]. 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 and storage synchronization.

The conception of small local cloud facilities is at a good experimental and design stage today. Commonly called 'cloudlets' [4], [12], they can support computational offloading [13]: running whole applications or part of applications out of the device, granting energy gains for the mobile device, and the execution of otherwise too computational intensive applications for a mobile device, such as for instance remote desktop or gaming applications. The decision to offload application and computing tasks can be a mere remote decision or local decision taken by the device. As explained in [14], the decision making can take into account a variate 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 [14] to network-level metrics, which can be largely important in geo-distributed cloud deployments.

B. Cloud access performance

User's quality of experience (QoE) typically refers to the tangible, visible performance the user experiences when consuming a digital service [15]. It is commonly presented as a desirable goal measurable with an orthogonal set of metrics (e.g., glitches, waiting times) with respect to legacy Quality of Service (QoS) metrics (e.g., jitter, delay, throughput) used by network operators. Obviously, providing good QoS-enabling mechanisms in the network can ensure smooth transmission of services (i.e., audio and video), which denotes that generic QoS techniques directly induce QoE metric levels [16]. The analytical relationship between QoE control mechanisms and QoS parameters is derived in [17]. In our work we refer to the following limited set of measurable QoS goals that directly affect user's QoE:

- *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 metric in service-level-agreements (SLA) related to network services, especially when the customer is a business entity that requires a very high level of reliability. For a data-center 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 [18]. 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 are the following: 100 ms is the boundary under which the user feels that the service is reacting instantaneously; between 100 ms and 1 s the user starts perceiving a non-negligible delay; above 1 s there is a risk that the user abandons the service [19]. For more recent and forthcoming mobile services, related to augmented reality, video/voice recognition, remote desktop, network gaming, 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 QoE especially for real-time services.

Our protocol architecture is such that the DC entry and VM migration decisions are made accordingly to metrics such as the availability, the latency and the jitter that are made measurable and observable by a dynamic protocol overlay.

C. Locator/Identifier Separation Protocol (LISP)

The basic idea of the Locator/ID Separation Protocol (LISP) [20] 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. It includes two elements for that: Map Server (MS) and Map Resolver (MR). The mapping resolution protocol currently adopted in public test beds and commercial services is based on the Delegated Database Tree (DDT) [21], 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 [22]. Its optimization in support of fast IP mobility have been proposed, to manage both VM mobility [23] and user mobility [24]; LISP offers a more efficient and expressive framework for this with respect to alternative solutions such as mobile IP or DNS-based solutions. A basic proposal to use LISP for localizing VMs as a function of user mobility is also investigated in [25]. Our protocol architecture builds over this overall research interest in using the versatile LISP architecture for mobile Cloud network environments.

III. Mobile Cloud Protocol Architecture

A basic reference 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 have a dedicated private interconnection network. 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 is migrated 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. The 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 design choices accordingly.

- *IP Addressing continuity*: it is important to maintain user's session when user changes its access point. Keeping a session alive when changing user's IP address is not obvious. The same applies to a VM migrating from DC 1 to DC 2. In fact, in absence of layer 2 continuity across IP endpoint attachment points (access point 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.

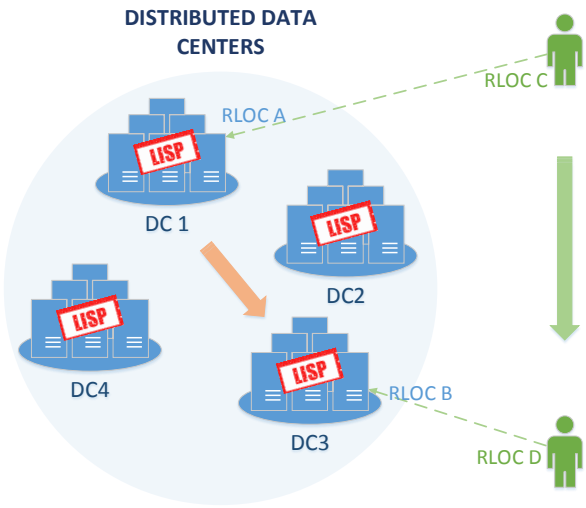


Fig. 1: Mobile cloud access reference scenario.

- *Access DC switching*: the user endpoint should be able to be configured remotely by the cloud operator so that it changes the access DC toward the service VM.
- *VM mobility*: in order to bring a VM closer to its user in terms of QoS distance, the cloud operator must be able to trigger a VM migration across multiple DC sites.
- *Cloud access link monitoring*: in order to support the decision-making related to access DC switching and adaptive 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 migration decisions need to be taken at a logically centralized controller.

Among the different overlay protocol alternatives, a few can satisfy the above requirements. Waiting for a mature specification and implementation of a network virtualization overlay protocol (NVO) by the NVO3 IETF working group [26], among the most promising implemented virtual network overlay protocols quickly reviewed in [2], the single ones supporting addressing continuity, capability to change the location of users and VMs, and capability to monitor the routing link between users' and VMs' locators with a distributed yet logically centralized control-plane, are the LISP and the Virtual eXtensible LAN (VXLAN) [27] protocols, the latter performing Ethernet over UDP-IP encapsulation. The PACAO architecture makes use of both VXLAN and LISP, proposing LISP as Cloud access overlay protocol and VXLAN as inter-DC overlay protocol[1].

[1]Using VXLAN as Cloud access protocol would generate a much higher signaling due to the multicast operations of VXLAN, while using LISP as inter-DC overlay protocol would be less effective given the need of managing VM location states via multicast signaling to all possible VM locations, as far as multicast extensions to LISP are not defined and implemented.

In the reference distributed DC fabric, each DC is a LISP site and has at least one LISP tunnel router (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 migrating a VM from one DC to another as it has been shown in our previous work [23]. 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 migrating 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 are LISP-capable mobile nodes that also have unique and static EIDs. By decoupling the locator from the identifier, mobile users are not tied to any geographical location from an addressing and routing perspective. In fact, as illustrated in 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[2]) 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 the user through the EID-to-RLOC mapping, which one should have the highest priority.
- if after switching to a new RLOC, it is worth migrating the VM to another DC.

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 correlated modules:

- *Monitoring Module*: it monitors the connection between the user and the VM, dividing the user-VM link into two links: user-xTR link and xTR-VM link. To monitor the user-xTR link, active online probes periodically collect QoS metrics between VM and users' RLOCs. In order to support all possible RLOC switching and VM migration decisions, all VM RLOCs, i.e., DC sites, are monitored and not only the one that belongs to the site where the VM runs. The probing operation is performed enhancing LISP probing; as of [20], 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 OpenLISP [29], [30] to allow transporting RTT and jitter metrics. To monitor the xTR-VM link, common DC monitoring tools can be used.
- *Optimization Module*: it implements an algorithm that solves the RLOC switching and VM migration optimization problem formulated in section III-C. When used for RLOC optimization, the agent basically takes the metrics collected from the monitoring module, and feed them to the algorithm that calculates 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 lowest LISP priority (best RLOC) is then attributed to the locator with the highest QoS score, and so on. When this module is used to decide the location of the VM, 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 compute the best destination that could host the VM.
- *Decision Module*: based on the optimization solution computed by the optimization module, the decision module absolves the important role in taking decisions on 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 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 orchestrated 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.

C. Cloud Access Optimization Formulation

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 at least one xTR (xTR1, xTR2, xTR3) with at least one RLOC. For the sake of simplicity, we consider that a VM can have one client at the same time (i.e., services such as virtual workspace or virtual desktop solutions - the extension of the following to the model with multiple users per VM is straightforward).

As a matter of fact, cloud providers wish to operate their virtual services at a desired level of QoS. The user often pays the service to the providers for an agreed-upon level of QoS that, if not respected, can lead to monetary compensations to the customer. In this context, in order to provide the agreed-upon level of QoS thus minimizing the penalty, the cloud provider is reasonably interested in trying to switch at first the

[2]It is worth noting that 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 [28].

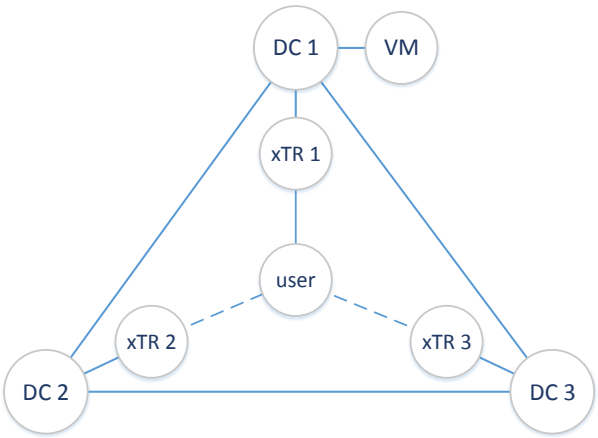


Fig. 2: Distributed DC reference use-case.

traffic of the user to another DC (by changing the priorities of the RLOCs in the EID-to-RLOC mapping), before possibly issuing a VM migration.

The Cloud access optimization problem therefore consists in minimizing the penalties that may arise from not respecting the agreed-upon level of QoS, taking decisions upon switching RLOC and/or migrating a VM, while respecting network constraints.

We give in the following an optimization formulation that is versatile enough to be applied for (i) the RLOC switching problem and (ii) the VM migration 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 network QoS or system performance criterion: $k = 1$ for round-trip-time (RTT), $k = 2$ for jitter, $k = 3$ for RAM, k=4 for CPU, etc. α_k is the weight that measures the importance of 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 each of the described criterion k.

Two important constraints apply to the problem. 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. Therefore:

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

where D is for (i) the set of the future DCs that can host the VM, and for (ii) the set of RLOCs to where a user can redirect its traffic to. $r_d \in [0; 1]$ is a binary variable that can indicate for (i) if a data-center d can host a VM or for (ii) if a the user's traffic can be switched to RLOC d.

The second constraint is a QoS level verification constraint, used in order not to exceed a fixed threshold or a residual capacity:

$$m_d^k r_d \leq M^k T^k \tag{3}$$

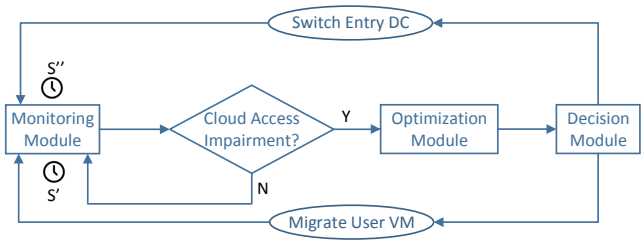


Fig. 3: PACAO policy chart.

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. If the problem represents the decision that must be taken to migrate 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 destination the host of a DC d – note that when migrating a VM, besides the residual capacity, we could easily also include parameters such as RTT and jitter.

Given the full-mesh and single-hop nature of the cloud access overlay graph, the problem defined by (1)-(3) does not contains any complex flow conservation constraint, has a polynomial complexity and can be easily solved online.

D. Algorithmic Online Scheduling Procedure

The above presented optimization can be triggered by the monitoring module that at each time-slot of a preset arbitrary duration S' (Fig. 3) verifies the user-cloud link for each monitored link. At the end of a time-slot, the PACAO controller calculates the mean 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 all 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 newly switched) xTR and the VM for an arbitrary duration $S'' << S'$, with a probing frequency S''' such that a sufficient number of probes can be collected over S''[3]. If the agreed-upon QoS levels are respected then go to **1.1**, else run Eq. (1)-(3) as a VM migration optimization problem.

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

[3] S', S'', S''' can be determined experimentally.

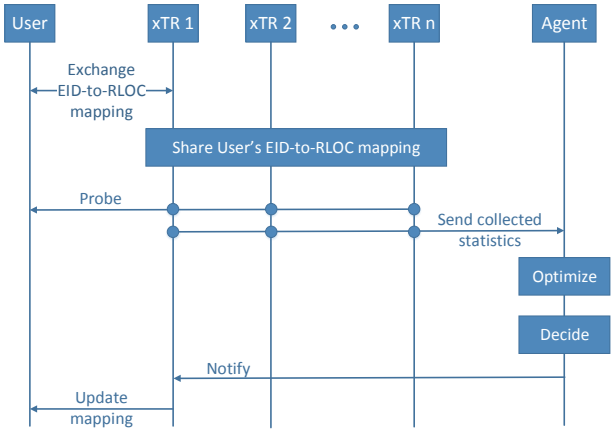


Fig. 4: PACAO sequential functional steps.

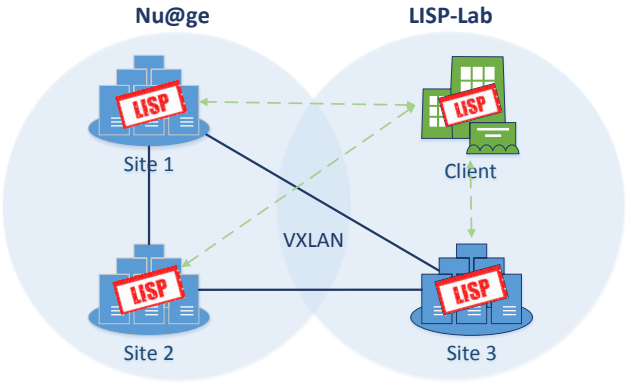


Fig. 5: Test bed network.

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. In order to minimize signaling and facilitate accessing data, we consider that the monitoring and decision modules described previously are implemented in the xTR.

Example: Fig. 4 illustrates seven steps that resume the interaction and procedures between the different elements of the PACAO architecture.

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, 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 using 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 the xTRs should securely share and synchronize their map-cache either by sending specific LISP control-plane messages or using an API to trigger and synchronize between the different xTRs. Alternatively, the xTR may be data-plane only elements (e.g., OpenVSwitch nodes, which natively support LISP data-plane) controlled by an external SDN controller centralizing the map-cache configuration.
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 the agent changes the priority of the RLOCs-set in the EID-to-RLOC mapping, and then notifies the user's endpoint and updates the Map-Server.
5) When the agent gets the second output, it starts migrating the VM to the new hosting DC.

IV. EXPERIMENTATION AND RESULTS

The PACAO architecture has been implemented using the following nodes.

xTR: we used OpenLISP [29], [30] as xTR software routers, 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 development version [31] has been extended accordingly. 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 frequency of the probes can be changed in the OpenLISP XML 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. For the optimization problem we have used the GNU Linear Programming Kit [32] (GLPK) to solve the optimization problem described in section III-C. For the decision module, the controller must decide when to switch the RLOC and when to migrate the VM: to switch locators we have 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 [23], and update the user's mapping cache; to migrate a VM we use the VIRSH command interface provided by the Libvirt API [33] 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 other more comprehensive SDN controllers.

Our test bed is represented in Fig. 5. We have joined two existing LISP networks: the Nu@ge [34] and the LISP-LAB [35] experimental networks, with three sites in the Paris metropolitan area network (Telcocenter DC in Courbevoie, Marilyn DC in Champs-sur-Marne, and LIP6 DC) and one 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. We have connected the 3 KVM servers acting as service VM containers together: one resides in TelcoCenter DC, one in Marilyn DC and the last one in the LIP6 DC; in order to perform seamless VM migrations, we have connected them to the same network using an OpenVPN server (disabling the authentication and the encryption module to reduce the overhead of VPN and accelerate the migration and the communication between hosts); the three servers are then interconnected using VXLAN over the VPN to enable traffic redirection when switching RLOCs and migrating VMs. The service VM is an Ubuntu 14.04 server that shares a file image mounted on a Network File Systems (NFS). The DC xTRs also run a FreeBSD 10 OpenLISP VMs.

All the inter-DC links use the plain Internet, except the Marilyn-TelcoCenter one that is a 10 Gbps fiber channel link, yielding to a heterogeneous test bed 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 have also created an FTP server on the IaaS in order to measure the throughput with the user.

A. *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 is an OpenLISP FreeBSD 10 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:

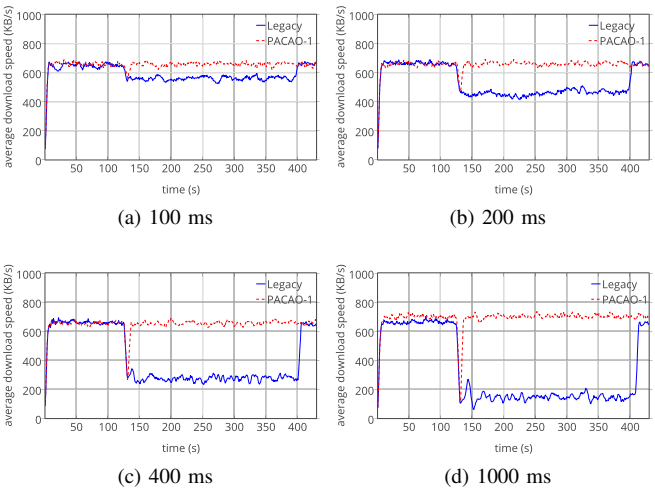
- Legacy: the legacy solution, with fixed routing locator.
- PACAO-1: our solution limited to RLOC switching optimization feature (VM migration disabled).

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 FTP server. It connects to TelcoCenter's xTR (the RLOC priority of TelcoCenter is set to 1 and Marylin is set to 2);



(a) 100 ms (b) 200 ms

(c) 400 ms (d) 1000 ms

Fig. 6: Average download speed given the scenario of Table I.

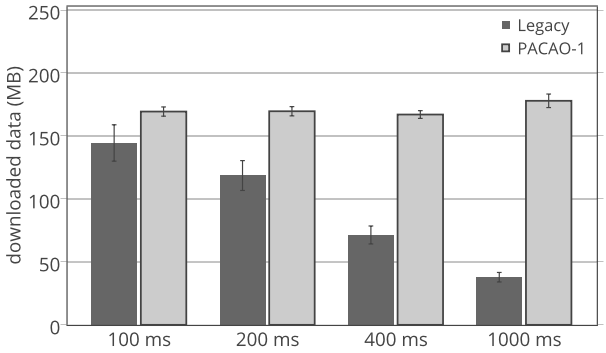


Fig. 7: Total amount of data downloaded in 450 s

2) we divide the time into time slots of 10 seconds. Each 10 seconds the agent calculates the average RTT on the user-xTR link collected by the RLOC probes;
3) after 125 seconds we increase the RTT between TelcoCenter xTR and the user. The injected RTT varies between 100 ms and 1000 ms;
4) the agent decides to switch the RLOC to Marilyn xTR;
5) we stop stressing the link at 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 (e.g., caused by congestion or link failure cause rerouting on a worse path): we inject delays on the user-RLOC link of 100 ms, 200 ms, 400 ms, 1 s. We notice that for the legacy 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 one time slot ($S' = 10\ s$), plus one half of RTT in order to update the map cache of the user with the CHANGE-PRIORITY message [23].

For the sake of completeness, we report the downloaded volume for the two scenario under the four different network impairment cases, during 450 s, in Fig. 7 for both solutions. 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%.

B. RLOC switching and VM migration

We then run the complete PACAO solution, this time using all DCs with the user host running as a FreeBSD 10 OpenLISP 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 migration.

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 service 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 one time slot ($S' = 10\ s$ in our case) and takes the decision to switch the traffic to LIP6 RLOC at 60 s.
3) after $S'' = 5\ s$, the controller runs only for PACAO-2 case the VM migration 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.

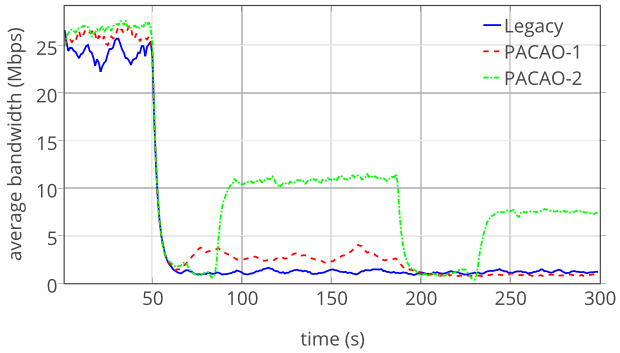


Fig. 8: Average bandwidth given the scenario of Table II.

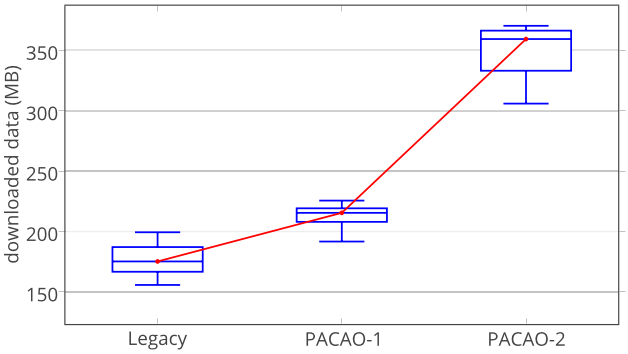


Fig. 9: Total amount of data downloaded in 300 s.

As depicted in Fig. 8 we run the experimentation for the three cases. One should notice that, even after switching the RLOC, the user connection is heavily affected due to inter-DC overloaded links. To overcome the problem the PACAO controller migrates the VM as well in order to prevent traffic redirection. 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 use case described above.

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. As shown, 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.

V. Conclusion

We have proposed a Protocol Architecture for Cloud Access Optimization, named PACAO, to optimize the quality of the

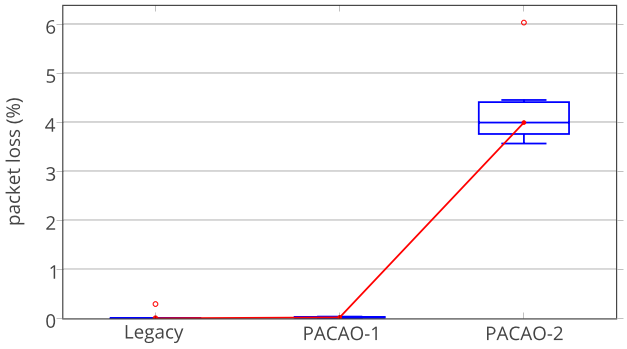


Fig. 10: Percentage of lost packets.

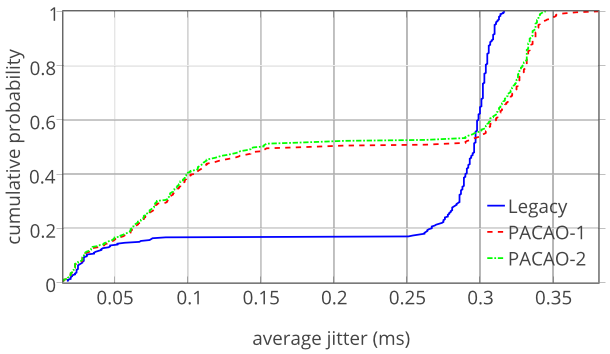


Fig. 11: CDF of the average jitter.

link between users and their virtual machines in distributed cloud fabric environment where user's virtual machines can be displaced across multiple data-center sites of a cloud provider to improve the user's quality of experience.

The PACAO architecture includes a Cloud access overlay network, managed and monitored by the LISP protocol, and a controller regularly optimizing the overlay allowing user's traffic to be possibly switched via another DC entry site and user's virtual machines to be possibly migrated to DC sites closer to users.

The proposed solution was implemented by making use of OpenLISP software routers and an ad-hoc PACAO controller. Extensive experiments on a real distributed data-center test bed allowed us to validate our proposal on realistic scenarios.

Acknowledgment

This work was supported by the FUI 15 Systematic RAVIR project, the ANR LISP-Lab and ABCD (Grants No: ANR-13-INFR-0001, ANR-13-INFR-0009) projects, and the FP7 MobileCloud (Grant No. 612212) project.

References

[1] Gartner Cloud Computing Forcasts Update. [Online]. Available: http://www.gartner.com/newsroom/id/2613015/
[2] S. Secci and S. Murugesan, "Cloud Networks: Enhancing Performance and Resiliency," *Computer*, vol. 47, no. 10, pp. 82–85, 2014.
[3] M. Satyanarayanan *et al.*, "The Case for VM-Based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.
[4] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet Network Design Optimization," in *Proc. of IFIP Networking*, 2015.
[5] "Mobile-Edge Computing – Introductory Technical White Paper," ETSI MEC Industry Specification Group 1, 2014.
[6] 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.
[7] Google Voice. [Online]. Available: https://www.google.com/voice/
[8] Apple Siri. [Online]. Available: https://www.apple.com/ios/siri/
[9] Google Glass. [Online]. Available: https://www.google.com/glass/start/
[10] 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.
[11] 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.
[12] M. Satyanarayanan *et al.*, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
[13] 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.
[14] 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.
[15] Empirix, "Assuring QoE on Next Generation Networks," Tech. Rep., 2003.
[16] T. M. O'Neil, "Quality of experience and quality of service for IP video conferencing," Tech. Rep., 2002.
[17] 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.
[18] W. Turner, J. H. Seader, and W. Renaud, "Data center site infrastructure tier standard: Topology," *Uptime Institute*, 2010.
[19] 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.
[20] D. Farinacci *et al.*, "The locator/ID separation protocol (LISP)," RFC 6830, Jan. 2013.
[21] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree," draft-ietf-lisp-ddt-02, 2013.
[22] Cisco, "Locator ID Separation Protocol (LISP) VM Mobility Solution," Tech. Rep., 2011.
[23] P. Raad *et al.*, "Achieving Sub-Second Downtimes in Large-Scale Virtual Machine Migrations with LISP," *IEEE Trans. on Network and Service Management*, vol. 11, no. 2, pp. 133–143, 2014.
[24] A. Galvani *et al.*, "LISP-ROAM: network-based host mobility with LISP," in *Proc. of ACM MobiArch*, 2014.
[25] A. Ksentini, T. Taleb, and F. Messaoudi, "A LISP-based Implementation of Follow Me Cloud," *IEEE Access*, vol. 2, pp. 1340–1347.
[26] D. Black *et al.*, "An Architecture for Overlay Networks (NVO3)," draft-ietf-nvo3-arch-02, 2014.
[27] 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.
[28] T. Taleb and A. Ksentini, "QoS/QoE predictions-based admission control for femto communications," in *Proc. of IEEE ICC*, 2012.
[29] L. Iannone, D. Saucez, and O. Bonaventure, "Implementing the locator/id separation protocol: Design and experience," *Computer Networks*, vol. 55, no. 4, pp. 948–958, 2011.
[30] D. C. Phung *et al.*, "The OpenLISP control plane architecture," *IEEE Network Magazine*, vol. 38, no. 2, pp. 34–40, 2014.
[31] OpenLISP LIP6 dev. version. [Online]. Available: https://github.com/lip6-lisp
[32] GNU Linear Programming Kit. [Online]. Available: https://www.gnu.org/software/glpk
[33] LibVirt virtualization API. [Online]. Available: http://www.libvirt.org
[34] Investissement d'Avenir Nu@ge Project. [Online]. Available: http://www.nuage-france.fr
[35] ANR LISP-Lab Project. [Online]. Available: http://www.lisp-lab.org