Protocol architecture and algorithms for distributed data center networks

Patrick Raad

UNIVERSITÉ PIERRE ET MARIE CURIE
Laboratoire d'Informatique de Paris 6

Protocol Architecture and Algorithms for Distributed Data Center Networks

Doctoral Dissertation of:
Patrick RAAD

Advisors:
Dr. Stefano SECCI
Dr. Pascal GALLARD

2015

Thèse

Présentée pour obtenir le grade de docteur
de l'Université Pierre et Marie Curie
Spécialité: Informatique

Patrick RAAD

Protocoles et algorithmes pour les réseaux de centres de données distribués

Soutenue le 14 décembre 2015 devant le jury composé de:

Rapporteurs:	**Dr. Thierry COUPAYE**	**Orange Labs, France.**
	Prof. Olivier FESTOR	**Inria, France.**
Examinateurs:	**Prof. Bijan JABBARI**	**George Mason University, USA.**
	M. Christian JACQUENET	**Orange Labs, France.**
	Prof. Guy PUJOLLE	**Univ. Pierre et Marie Curie, France.**
	Prof. Jean-Louis ROUGIER	**Télécom ParisTech, France.**
	Prof. Djamal ZEGHLACHE	**Télécom Sud-Paris, France.**
Invité:	**M. Nicolas AUBÉ**	**CELESTE, France.**
Directeur de thèse:	**Dr. Stefano SECCI**	**Univ. Pierre et Marie Curie, France.**
Encadrant:	**Dr. Pascal GALLARD**	**Non Stop Systems (NSS), France.**

UNIVERSITÉ PIERRE ET MARIE CURIE
Laboratoire d'Informatique de Paris 6

Protocol Architecture and Algorithms for Distributed Data Center Networks

Author: Patrick RAAD

Defended on December 14, 2015, in front of the committee composed of:

Referees:	Dr. Thierry COUPAYE (Orange Labs, France).
	Prof. Olivier FESTOR (Inria, France).
Examiners:	Prof. Bijan JABBARI (George Mason University, USA).
	Mr. Christian JACQUENET (Orange Labs, France).
	Prof. Guy PUJOLLE (Univ. Pierre et Marie Curie, France).
	Prof. Jean-Louis ROUGIER (Télécom ParisTech, France).
	Prof. Djamal ZEGHLACHE (Télécom Sud-Paris, France).
Invited:	M. Nicolas AUBÉ (CELESTE, France).
Advisor:	Dr. Stefano SECCI (Université Pierre et Marie Curie, France).
Co-advisor:	Dr. Pascal GALLARD (Non Stop Systems, France).

Remerciements

J'adresse mes remerciements aux personnes qui m'ont aidé et soutenu dans la réalisation de cette thèse et durant ces trois dernières années.

En premier lieu, je remercie mon directeur de thèse M. Stefano SECCI, docteur à l'université Pierre et Marie Curies (LIP6) et mon encadrant M. Pascal GALLARD, directeur de recherche à Non Stop Systems (NSS). En tant qu'encadrants, ils m'ont guidé vers le droit chemin afin de trouver des solutions et avancer dans une aventure qui a duré 3 ans.

Je remercie M. Guy PUJOLLE, professeur à l'université Pierre et Marie Curie (LIP6), qui m'a accueilli à bras ouvert au sein de son équipe PHARE.

Je remercie M. Marc TRIBOULET, président de Non Stop Systems (NSS), qui a cru en moi pendant toutes ses années en me donnant une opportunité de s'épanouir au sein de son entreprise.

Je remercie mes deux amis Matthieu COUDRON et Dung PHUNG CHI de l'équipe PHARE, qui sans eux ce travail n'aurait pas été achevé.

Je remercie tous mes collègues de l'équipe PHARE et de Non Stop Systems, qui m'ont soutenu tant sur le plan personnel que sur le plan professionnel.

Je remercie spécialement toute ma famille au Liban et en France, qui ont toujours cru en moi et m'ont poussé à garder la tête haute.

Enfin, je remercie ma femme chérie, qui m'a supporté pendant que je fesais des nuits blanches en compagnie de mon ordinateur. En clair, merci d'avoir supporter un GEEK mon amour.

Abstract

While many business and personal applications are being pushed to the cloud, offering a reliable and a stable network connectivity to cloud-hosted services becomes an important challenge to face in future networks. In this dissertation, we design advanced network protocols, algorithms and communication strategies to cope with this evolution in distributed data center architectures. We propose a user-centric distributed cloud network architecture that is able to: (i) migrate virtual resources between data centers with an optimized service downtime; (ii) offer resilient access to virtual resources; (iii) minimize the cloud access latency. We identify two main decision making problems: the virtual machine orchestration problem, also taking care of user mobility, and the routing locator switching configuration problem, taking care of both extra and intra data center link states.

We evaluate our architecture using real test beds of geographically distributed data centers, and we also simulate realistic scenarios based on real mobility traces. We show that migrating virtual machines between data centers at negligible downtime is possible by enhancing overlay protocols. We then demonstrate that by linking cloud virtual resource mobility to user mobility we can get a considerable gain in the transfer rates. We prove by simulations using real traces that the virtual machine placement decision is more important than the routing locator switching decision problem when the goal is to increase the connection throughput: the cloud access performance is primarily affected by the former decision, while the latter decision can be left to intra data center traffic engineering solutions. Finally, we propose solutions to take profit from multipath transport protocols for accelerating cloud access performance in our architecture, and to let link-state intra data center routing fabrics piloting the cloud access routing locator switching.

Résumé en Langue Française

De nos jours les données ainsi que les applications dans le nuage (*cloud*) connaissent une forte croissance, ce qui pousse les fournisseurs à chercher des solutions garantissant un lien réseau stable et résilient à leurs utilisateurs. Dans cette thèse on étudie les protocoles réseaux et les stratégies de communication dans un environnement de centre de données distribués. On propose une architecture *cloud* distribuée, centrée sur l'utilisateur et qui a pour but de: (i) migrer des machines virtuelles entre les centres de données avec un temps d'indisponibilité faible; (ii) fournir un accès résilient aux machines virtuelles; (iii) minimiser le délai d'accès au *cloud*. On a identifié deux problèmes de décision: le problème d'orchestration de machines virtuelles, prenant en compte la mobilité des utilisateurs, et le problème de basculement et de configuration des localisateurs, prenant en compte les états des liens inter- et intra-centre de données.

On évalue notre architecture en utilisant une plate-forme de test avec des centres de données distribués géographiquement et en simulant des scenarios basés sur des traces de mobilités réelles. On montre que, grâce à quelques modifications apportées aux protocoles d'*overlay*, on peut avoir des temps d'indisponibilité très faibles pendant la migration de machines virtuelles entre deux centres de données. Puis on montre qu'en reliant la mobilité des machines virtuelles aux déplacement géographiques des utilisateurs, on peut augmenter le débit de la connexion. De plus, quand l'objectif est de maximiser le débit entre l'utilisateur et sa ressource, on démontre par des simulations que la décision de l'emplacement des machines virtuelles est plus importante que la décision de basculement de point d'entrée du centre de données. Enfin, grâce à un protocole de transport multi-chemins, on montre comment optimiser les performances de notre architecture et comment à partir des solutions de routage intra-centre de données on peut piloter le basculement des localisateurs.

Contents

List of Figures

List of Tables

Acronyms

AFI	Address Family Identifier.
API	Application Programming Interface.
AS	Autonomous System.
BGP	Border Gateway Protocol.
CAPEX	CApital EXpenditure.
CIDR	Classless Inter-Domain Routing.
CP	Change Priority.
CPU	Central Processing Unit.
DC	Data Center.
DDT	Delegated Database Tree.
EC2	Elastic Compute Cloud.
EID	Endpoint Identifier.
ELP	Explicit Locator Path.
ETR	Egress Tunnel Router.
GRE	Generic Routing Encapsulation.
HMAC	Keyed-Hash Message Authentication Code.
IaaS	Infrastructure as a Service.
IETF	Internet Engineering Task Force.

IP	Internet Protocol.
ISIS	Intermediate System to Intermediate System.
ISP	Internet Service Provider.
IT	Information Technology.
ITR	Ingress Tunnel Router.
LAN	Local Area Network.
LCAF	LISP Canonical Address Format.
LIG	LISP Internet Groper.
LISP	Locator/Identifier Separation Protocol.
LISP-TE	LISP Traffic Engineering.
LSP	Link State Packet.
MAC	Media Access Control.
MPTCP	Multipath TCP.
MS	Map Server.
NIC	Network Interface Controller.
OPEX	OPerating EXpenditure.
OS	Operating System.
OSPF	Open Shortest Path First.
QoE	Quality of Experience.
QoS	Quality of Service.
RLOC	Routing LOCator.
RTR	Reencapsulating Tunnel Router.
RTT	Round-Trip Time.
SCTP	Stream Control Transmission Protocol.
SDN	Software Defined Network.
SLA	Service Level Agreement.
SMR	Solicit Map-Request.
SPB	Shortest Path Bridging.

STP	Spanning Tree Protocol.
TCP	Transmission Control Protocol.
TRILL	Transparent Interconnection of a Lot of Links.
TTL	Time To Live.
UDP	User Datagram Protocol.
VM	Virtual Machine.
VPN	Virtual Private Network.
WAN	Wide Area Network.
xTR	LISP Tunnel Router.

Chapter 1

Introduction

Before the emergence of cloud computing, network resources, storage and computing were separated on individual physical machines. For security reasons, systems and applications that interacted with such resources were kept physically directly accessible by the Information Technology (IT) administrators. This practice started to change with the introduction of inexpensive resources in data center environments. At first, data centers were used to host basic database servers, email servers and other types of application servers. They provided the enterprises an easy way to manage their resources at lower costs and share it among their users. When the concept of hosting a complete virtual Operating System (OS) on an another incompatible OS (i.e., running Linux on Windows OS) was materialized in software virtualization solutions, a new communications paradigm shift started to emerge. System virtualization became ubiquitous in data center environments, where hundreds of virtual machines can be controlled by a single logical console. Hence, physical servers were pushed into individual isolated virtual machines which can be cloned, paused and destroyed on-demand. This elasticity allowed network operators to start relocating data center resources based on energy consumption metrics. For instance, an operator could pack all the active virtual machines together and idle all the rest of the unused data center space in order to optimize the cooling.

With the increasing demand of power-harvesting resources (i.e., CPU, memory, storage, etc.), data center equipment continued to evolve at a much faster pace than network equipment. In legacy networking systems, operators needed to configure individually and separately each network device using vendor-specific equipment and tools. In addition, besides the management complexity of IP, automation of network management, despite huge research on the topic, saw only marginal deployment. In this context, network virtualization represented a necessary evolution to allow operators to dynamically express network policies on-the-go and at low cost.

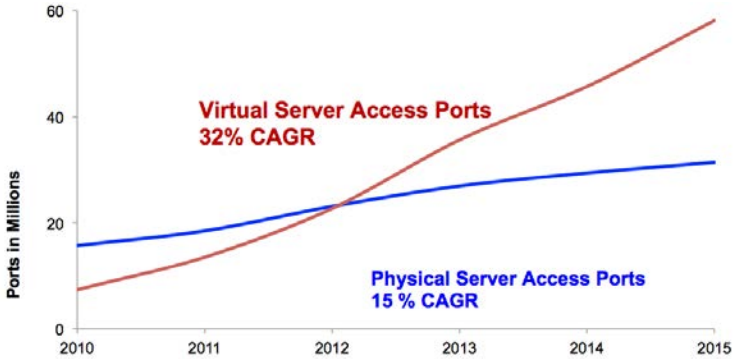


Figure 1.1: Starting point for network virtualization. Source [9]

Network interface virtualization is also pushing the adoption of software defined networking solutions that increase network programmability by separating the control plane (the brain logic of routers) from the data plane (the forwarding logic). In this way, virtual networking equipment can become simple forwarding devices [10, 11], simplifying network reconfiguration and facilitating the management of data centers. According to Figure 1.1, since a few years there are more network interfaces deployed in software rather than in hardware: the number of physical network interfaces deployed in the world is growing 15% per year, whereas the number of virtual network interfaces growing 32% per year and it is estimated to already overcome the number of physical interfaces.

A rapid evolution of the "Cloud", including behind this term both the data center network infrastructure and the computing system platforms, has been happening since a decade, and had been particularly important in the last five years. Users, henceforth, can potentially possess infinite storage space, infinite CPU capacity and infinite networking resources at high reliability and availability. Such a potentially infinite cloud capacity encouraged cloud consumers to outsource their storage and computation to public providers at competitive costs [13].

With this growth, cloud users and providers began to have new requirements: on one hand users want to get faster, reliable and more reactive access to their cloud applications at a minimal cost, and on the other hand providers seek to increase their profit. Therefore, to guarantee high performance to cloud applications, providers started to take service level agreements into account. Such an agreement between cloud users and providers can ensure that agreed-upon quality of service levels are being respected. However, even until now cloud providers rarely offer service level agreement to users when selling cloud services, such as namely infrastructure as a service solutions [14, 136]. Indeed the service level agreement provisioning can be limited to meet provider needs without taking into consideration the experience nor the expectations of cloud users [15]. Therefore it becomes challenging for providers

to build a cloud network architecture that offers seamless connectivity to users without violating service level agreements nor excessively affecting providers cost.

One way to cope with the rising users requirements in terms of network performance, system performance and cloud service availability is to geographically distribute the cloud network architecture. Disposing of many sites undeniably can increase service availability if services are adequately orchestrated over the different sites. It can also increase the network performance if techniques are implemented to decrease the physical channel length between the user and the cloud.

With the goal of designing a flexible and efficient distributed data center network protocol architecture, in this dissertation we aim at answering the following research questions:

- How to perform efficient cloud service migration in a distributed data center environment?
  Migrating virtual machines from one data center to another can allow to meet various goals, such as reducing the energy consumption, increasing the network reliability and optimizing the access to the cloud. Existing solutions essential rely on triangular routing between user, source data center and destination data center locations, inducing a certain overhead to communications besides undermining nominal service availability. In this dissertation we propose a cloud network overlay protocol architecture supporting transparent virtual machine migration with minimum service downtime.

- Is it reasonable to adapt virtual machine migrations to user mobility and cloud access network states in distributed data center infrastructures?
  Cloud users have become increasingly mobile over the years thanks to the emergence of connected mobile devices. Many mass mobile cloud applications (e.g. Google Voice Search [87] and Apple Siri [88]) heavily rely on cloud hosted platforms and necessitate a stable and a reliable network connection. In distributed data centers, cloud users can benefit from multiple data center entry points (routing locators) to their applications. Cloud access could in this context be optimized so that (i) routing locators are switched based on the quality of the network links, and to (ii) virtual machines are adaptively migrated across data center sites as the user-to-cloud link quality varies. In this dissertation we propose solutions to manage these new requirements and to evaluate these features.

- How can we take profit from data center multi-homing of distributed cloud infrastructure to boost cloud access performance?
  Providing an enhanced experience to users by reducing the latency, increasing network stability and throughput, is an important challenge for cloud

providers. While most of web-based applications rely on application level acceleration, the network stack remains important in increasing data transfer rates. In this dissertation, we propose a cross-layer transport-network solution to accelerate cloud access connections in distributed data center architectures.

- How can internal data center network states automatically guide routing locator selection in distributed clouds?
  In distributed data centers users can be made able to access their virtual machines from several entry points, which can increase network resiliency and availability to cloud applications. In this dissertation we propose a cross-layer solution to automatically link the routing locator decision to intra data center network states, under the assumption that these states can dynamically change as a function of cloud network operation.

In order to provide solutions to address the presented questions, we present in this dissertation a protocol architecture for distributed clouds, providing solutions for the algorithmic, protocol design and system design challenges it brings.

Our proposal is a multi-layer network architecture that fills the gap between the three network layers involved in distributed cloud communications: the Transport Control Protocol layer at the user-side, the network Internet Protocol layer at the cloud access network segment, and the datalink network segment at the data center network segment, as represented in Figure 1.2. Each layer needs to be complemented with new features we design, experiment and evaluate. These features involve the network stack, the virtualization system stack and the cloud orchestration decision-making stack.

Structure of the Dissertation

The remainder of this dissertation is as follows:

- **Chapter 2** describes the broad cloud and network context of the dissertation, presenting major network protocols and cloud technologies.

- **Chapter 3** sheds the light on the cloud users' expectations and provider strategies, and describe our reference user-centric cloud network architecture.

- **Chapter 4** proposes a cloud access overlay protocol framework that can allow cloud providers to migrate public virtual machines with optimized downtime, and without affecting user's connectivity.

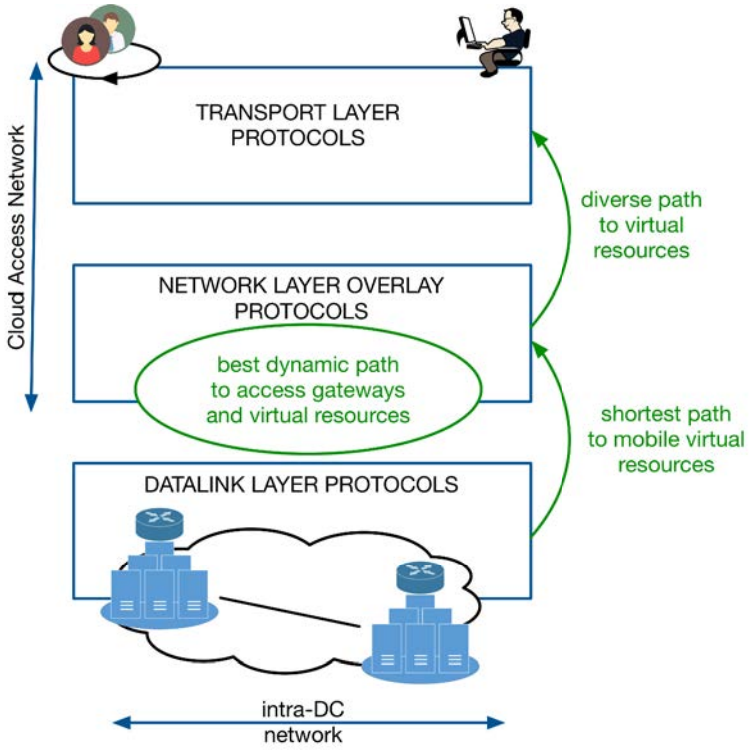


Figure 1.2: Representation of our reference distributed cloud architecture.

- **Chapter 5** describes how to integrate the technique presented in **Chapter 4** with a distributed data center controller to adaptively orchestrate virtual machine migration and routing locator switching as a function of user mobility and cloud access network states.
- **Chapter 6** further extends our architecture to support cloud access acceleration based on smart multipath transport.
- **Chapter 7** describes a proof-of-concept prototype of a cross-layer solution to automatically set routing locators as a function of intra-data center link-states in our distributed cloud network architecture.
- **Chapter 8** concludes the dissertations and opens up new perspectives for further work.

Chapter 2

Background

In this Chapter we introduce the necessary background to concepts and technologies we refer to in the reminder of the dissertation. We start by briefly introducing the Internet routing protocol, and explain how the Internet of today is evolving, including recently standardized protocols such as the Locator/ID Separation protocol (LISP). We discuss how the data center network and protocol architectures have evolved as well as how their infrastructure challenges are yielding to new networking paradigms that started shaping today's cloud and networking industry. We trace the evolution of cloud technologies, by describing their basic architectures, to end up with explaining how the quality of experience of cloud users can be qualified. We shed the light on the role of LISP and other tunneling solution in the migration of virtual machines. Finally, we introduce various network-based acceleration solutions that are being designed for data-center networking.

2.1 Internet Routing

Internet routing needs to be operated by an external gateway protocol that is implemented by all Autonomous Systems in the Internet that use it as common language to exchange IP routes. Since the commercialization of Internet access in 1992, the single Internet external routing protocol is Border Gateway Protocol (BGP), which exposes the necessary metric to perform policy routing as needed to model different interconnection schemes.

In this section we briefly introduce the BGP protocol and its impact on today's Internet functioning.

BGP [28] is an essential Internet protocol for Internet Service Provider (ISP)s to operate Internet services, with several hundred thousand of IP prefixes that are exchanged on a daily basis. An ISP with an independent routing domain is also called an Autonomous System (AS). Each AS uses BGP to exchange routing information on the Internet with neighboring AS and eventually the whole Internet.

When BGP is used between two different ASs it operates using an external BGP mode, while when BGP is used between routers of a same network it is referred to as internal BGP. Internally to an AS networking, protocols such as Open Shortest Path First (OSPF) [132] and Intermediate System to Intermediate System (ISIS) [135] are used to achieve fast internal routing convergence and also to instruct some of the BGP routing decisions. Indeed, the choice of the route between different ASs can be more complicated than finding the shortest path. Full routing information is exchanged between different BGP neighbors when the connection is established. If a route change is detected on a BGP router, adjacent neighbors are notified of this modification. Routes are affected by attributes that determine the best path to a destination when multiple routes exist to the same destination.

The attributes used by the BGP decision protocol include:

- **local preference**: is the degree of preference for one path or one neighbors over the others. If multiple exit points exit from an AS, the local preference is used for selecting the exit point for specific paths;
- **AS_path**: is the complete AS path length toward the destination;
- **origin** attribute is the internal path cost toward the egress router on the way to the destination;
- **multi-exit discriminator** is an attribute that gives the degree of preference over multiple incoming links from the same AS;
- **community** is a set of administrative policy information associated by ingress and ingress BGP filters to outgoing and incoming routes.

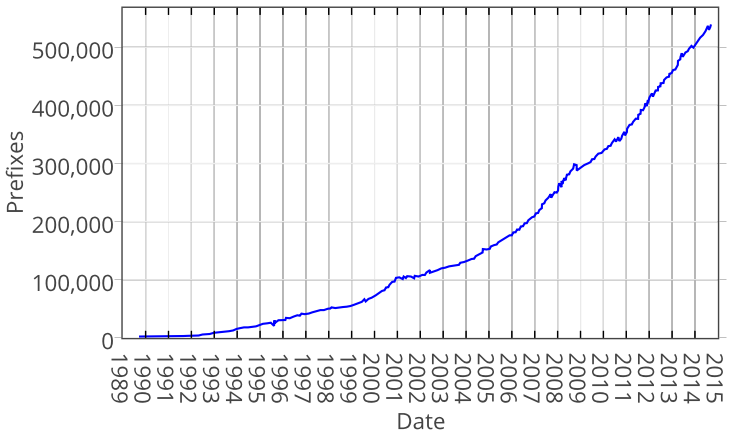


Figure 2.1: Prefix announced on the Internet.

Whenever a BGP router receives multiple route advertisement from different sources, it only chooses one path by preferring: (i) the route with the largest local preference; (ii) the route that has the shortest AS_path; (iii) the route with the lowest origin type; (iv) the route with the lowest multi-exit discriminator; (v) the external route over the internal one; (vi) the route with the lowest IP address.

BGP suffers today of severe security issues, some of which have been partially solved by standards that are, however, not widely implemented. As an example, on June 12th, 2015, a massive route leak caused a large portion of the Internet to slowdown for 2 hours [134].

Another major issue affecting BGP resiliency is its routing scalability. The growth of the Internet routing table has been exponential since the middle of the nineties. It is putting pressure on old routers that risk to no longer possess the required resources (i.e., Central Processing Unit (CPU), memory, etc.) to maintain large routing tables. The number of advertised Internet Protocol (IP) prefixes on the Internet is summarized in Figure 2.1. Despite the introduction of classless interdomain routing (Classless Inter-Domain Routing (CIDR)) in 1997, the growth started to be exponential a few years later. Old router maximum capacity (510k [73]) has already reached its limit at 2014, when the IPv4 BGP routing table exceeded 510k announced prefixes [83], causing Internet service interruptions in many segments of the Internet.

2.2 Locator/Identifier Separation Protocol

In this section we explain how the Locator/Identifier Separation Protocol (LISP) works and why it has been introduced, then describing specifically its traffic engineering features.

2.2.1 Protocol Aspects

The rapid growth of the routing tables remains one of the most important problem in today's Internet. LISP [54] is an approach that tries to slow down and limit the routing table growth. It implements an additional routing level on the top of BGP, separating the IP location from the identification using Routing Locators (Routing LOCator (RLOC)s) and Endpoint Identifiers (Endpoint Identifier (EID)s). An EID is an IP address that identifies a terminal, whereas an RLOC address is attributed to a border tunnel router. LISP uses a map-and-encap scheme at the data plane level, mapping the EID address to an RLOC and encapsulating the packet into another IP packet before forwarding it through the Internet transit. At the control plane level, multiple RLOCs with different weights and priorities can be associated with an EID: for uni-path communications, the least-priority RLOC corresponds to the one to be selected for encapsulation; when a subset or all of the RLOCs have the same priority values, load-balancing is performed on the equal-priority RLOCs. RLOC priority and weight are assigned by the destination EID space owner using its LISP routers.

A LISP site is managed by at least one LISP Tunnel Router (xTR), which has a double functionality: IP packet encapsulation (packet received by a terminal; ingress functionality, or Ingress Tunnel Router (ITR)) and packet decapsulation (packet received by the network; egress functionality, or Egress Tunnel Router (ETR)). The IP-in-IP encapsulation includes a LISP header transporting control functionalities, and a UDP header allowing differentiation between data and control plane messages. For a better understanding, consider the example in Figure 2.2: the traffic sent to the 2.2.2.2 host is encapsulated by the source's ITR toward one of the two destination's RLOCs. The one with the best (lowest) priority metric is selected, which at reception acts as ETR and decapsulates the packet, before sending it to the destination. On the way back to 1.1.1.1, RLOC4 queries a mapping system and gets two RLOCs with equal priorities, hence performs load-balancing as suggested by the weight metric (RLOC1 is selected in the example's packet).

In order to guarantee EID reachability, LISP uses a mapping system that includes a Map Resolver (MR) and a Map Server (MS). As depicted in Figure 2.2, a Map Resolver holds a mapping database, accepts MAP-REQUESTs from xTRs and handles EID-to-RLOC lookups; a particular Map-Request message, called Solicit Map-Request (SMR) can have a flag set (S bit) to solicit a Map-Request to self by the receiver (passing via the MR). A Map Server receives Map-Register messages from ITRs and registers EID-to-RLOC in the mapping database [16]. The EID-to-RLOC mapping resolutions can be based on two protocols: LISP-ALT (Alternative Topology) [38] and DDT (Delegated Database Tree) [22], the first relying on BGP primitives, the second being inspired by DNS. It is worth noting that, in

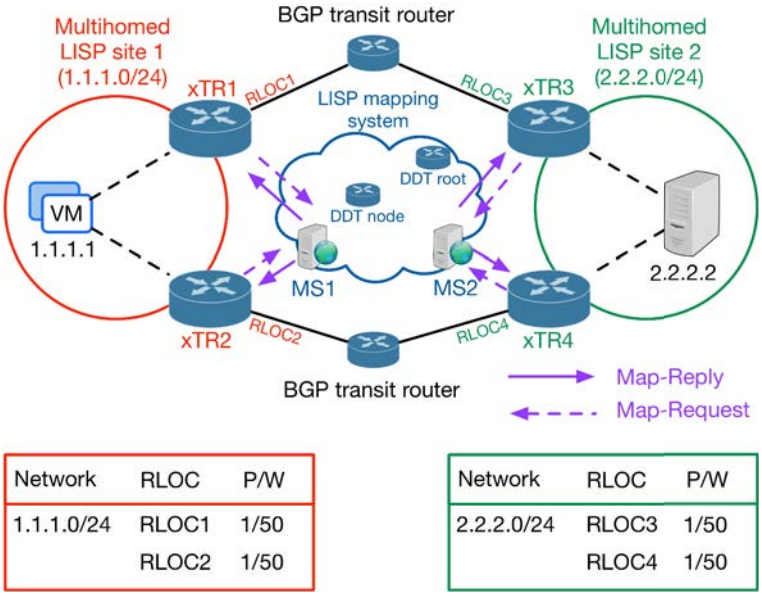


Figure 2.2: LISP communications example.

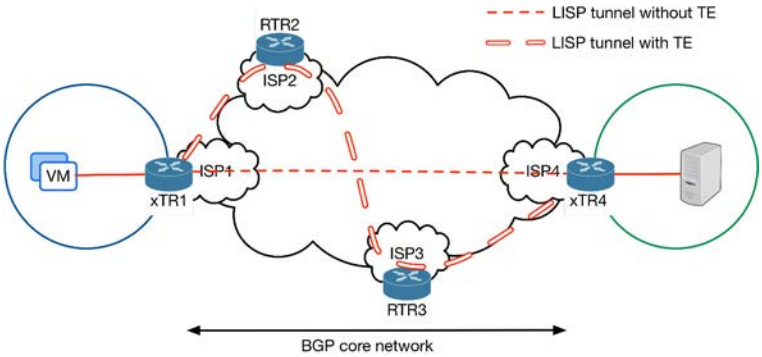


Figure 2.3: LISP-TE example.

a given configuration, if two xTRs, exchanging traffic, use the same MS/MR, when an xTR sends a Map-Request message for an EID that belongs to the other xTR, the MS/MR does not need to use DDT and hence DDT does not add an additional mapping latency to the xTR-MS/MR path latency.

2.2.2 LISP Traffic Engineering

While the packets are encapsulated from one LISP site to another LISP site, the direct path is still determined by the metrics of underlying protocols (i.e., BGP). LISP Traffic Engineering (LISP-TE) [142] is introduced to enable traffic engineering capabilities over LISP, allowing packets to be routed across multiple different desired paths. To understand better how traffic engineering works in LISP we first start

by defining the following terms: (i) an Explicit Locator Path (ELP) is as ordered list of Reencapsulating Tunnel Router (RTR) RLOCs that contains explicitly the path that a LISP packet should follow. It is worth mentioning that the format of ELP is defined in the LISP Canonical Address Format (LCAF) draft [149] which also describes specific encoding for arbitrary Address Family Identifier (AFI) values (i.e., AS number, geo coordinates, multicast information, etc.); (ii) a RTR is a router that decapsulates packets and then encapsulates them to the next hop RLOC based on the ELP. Note that the path between RTRs is determined by the metrics of the underlying protocol.

Let us consider the example in Figure 2.3: xTR1 receives the packets from the Virtual Machine (VM); it retrieves the ELP from the mapping database, then encapsulates the packets to RTR1; after decapsulating the packets the RTR1 uses the destination RLOC address to retrieve the ELP from the mapping database, then encapsulates them to the next hop RTR; RTR2 repeats the same steps as RTR1, then encapsulates the packets to their final hop (xTR4).

2.3 Data Center Networking: Ethernet Switching and Routing

In this section we explain how data centers are evolving nowadays, and introduce several Ethernet switching and routing protocols that are used in their core network architecture.

2.3.1 Geographical Distributed Data Centers

The current trend in the design of cloud fabrics is to geographically distribute it over multiple DC facilities [110] as shown in Figure 2.4. 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 decreasing the network distance between users and computing facilities. Modular DC architectures [105] 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 [91] [92]. 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’ [60, 97], they can support computational

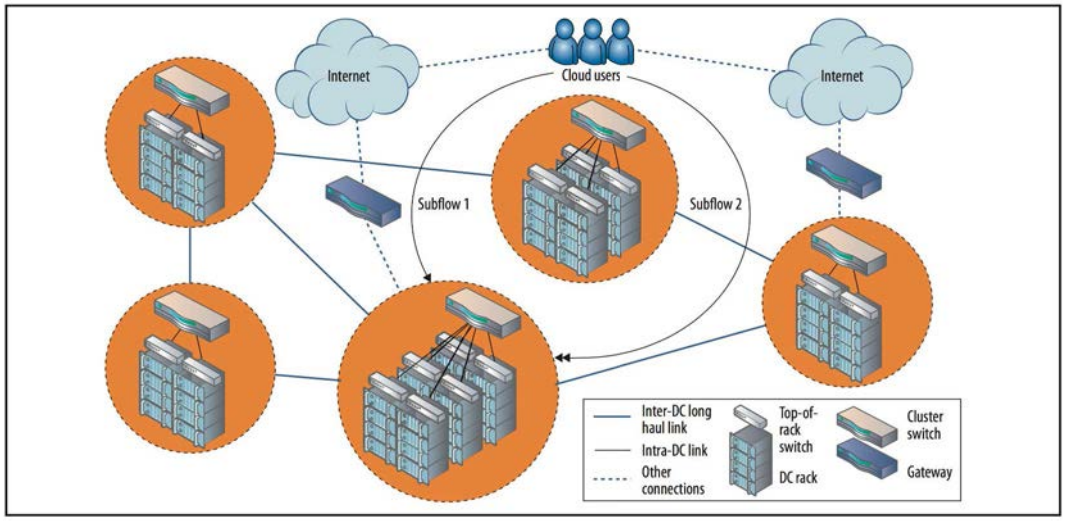


Figure 2.4: Geographically distributed DCs. Source [110]

offloading [98]: a cloudlet common application or part of an application out of the device, granting high network availability and energy gains to computational intensive applications especially for mobile devices, such as for instance remote desktop or gaming applications [109]. The decision to offload application and computing tasks can be a mere remote decision or local decision taken by the device. As explained in [96], 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 [96] to network-level metrics, whose importance become higher for geo-distributed cloud deployments.

2.3.2 Spanning Tree Protcol

One of the major problem in today's Ethernet architectures, is how protocols such as Spanning Tree Protocol (STP) handle loops in the topology. STP is a network protocol designed to prevent bridge loops in Ethernet networks. It prunes multiple links from Layer-2 topology yielding to a loop-free tree topology [17]. Most of the times, STP links are pruned inefficiently by removing the shortest path links and forcing Ethernet frames to strictly follow the spanning tree topology as shown in Figure 2.5. Another disadvantage of STP is that whenever a topology update occurs, it takes several seconds for trees to converge.

In this context, Layer-2 multipath protocols such as Shortest Path Bridging (SPB) and Transparent Interconnection of a Lot of Links (TRILL) were designed to improve Ethernet capabilities in virtualized data center. In the following we briefly introduce SPB and TRILL and then discuss how Software Defined Network (SDN) is gaining momentum in today's network architectures.

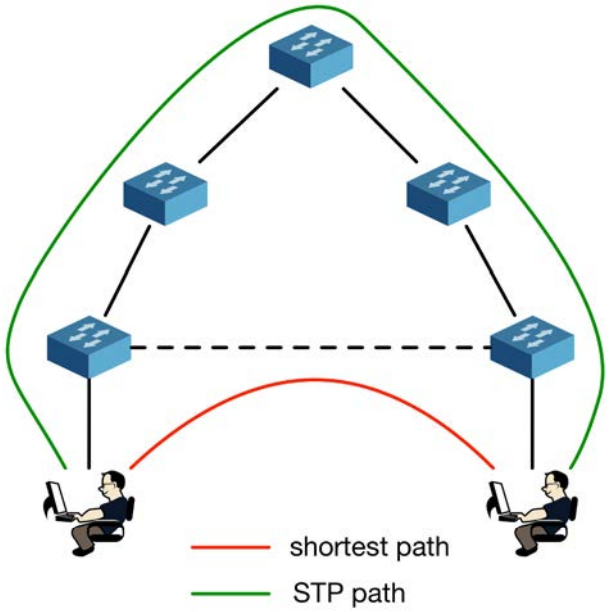


Figure 2.5: An example of inefficient traffic bridging using the spanning tree protocol.

2.3.3 Shortest Path Bridging

SPB is specified in IEEE 802.1aq standard. Unlike STP, SPB guarantees that the shortest paths are always selected in trees. The control plane is based on the ISIS link-state routing protocol which is used to exchange information between bridges in order to generate shortest path trees. Moreover, SPB supports load balancing by calculating trees that have equal costs (euqal cost trees). In SPB packets are generally encapsulated in MAC-in-MAC or in tagged VLAN (Q-in-Q) frames.

2.3.4 Transparent Interconnection of a Lot of Links

TRILL is an Internet Engineering Task Force (IETF) standard that shares some design choices with SPB, i.e., using ISIS for building Ethernet-level routing tables, but which has the key advantage of being incrementally deployable, potentially at only a subset of the switches in a data center.

Devices implementing TRILL are called Routing Bridge (RBridge). The TRILL data plane absolves the role of encapsulating incoming packets to a TRILL network towards a destination Layer-2 locator (or egress RBridge in the TRILL jargon), supported by a partially out-of-band control plane for distributing the mapping information [112]. TRILL uses an adaptation of the ISIS link-state routing protocol to calculate the shortest path at Layer-2. To establish the neighborhood topology, a RBridge sends Hello packets and a Link State Packet (LSP) to all its neighbors

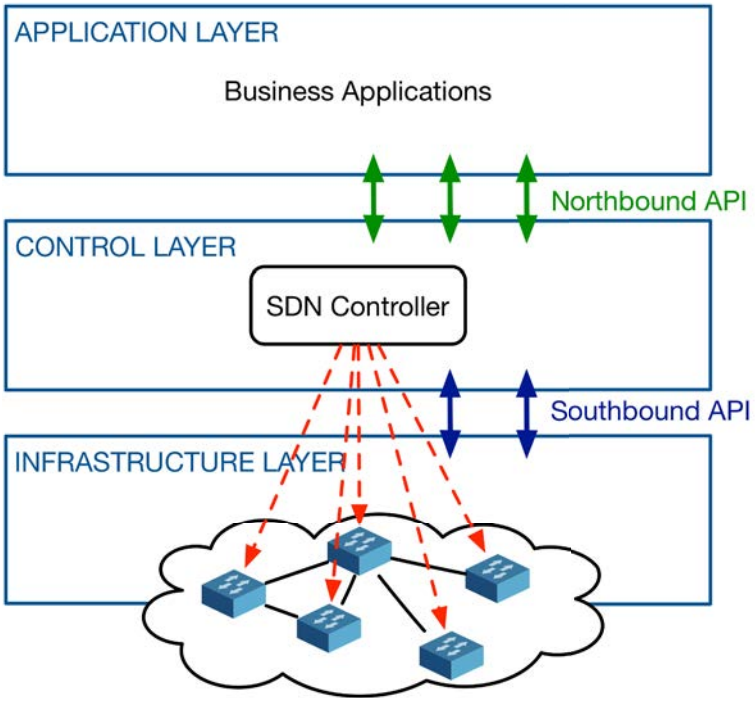


Figure 2.6: Software Defined Network Architecture.

to glean nodes' information, and calculates the path cost between two reachable RBridges. As a result, the TRILL-ISIS forwarding table contains all RBridge 'nicknames'associated to TRILL next hop(s) (as MAC addresses) and the corresponding ISIS shortest path cost.

2.3.5 Software Defined Networking

Software Defined Networking (SDN) [150] is a new approach that consists in optimizing network operations by abstracting software applications from physical devices. Centralized network controllers - also known as SDN controllers - are used to orchestrate, manage and facilitate communication between applications that want to interact with network physical elements and vice-versa. The objective of the controller is to separate network control operations via a programmatic interface. SDN is a software-driven, programmable network that is based on the success of technologies and protocols that preceded it (i.e., IP, BGP, Ethernet, etc.).

The SDN architecture consists of one or many SDN logically centralized controller(s), a southbound Application Programming Interface (API)s and a northbound APIs. The centralized controller offers a general view of the network topology and provides IT administrators the possibility to dictate their network policies and define how the underlying network devices should forward the traffic from one device to another. Southbound APIs are used by SDN controllers to send configuration

information to network devices whereas northbound APIs is used to communicate with the business applications (Figure 2.6). While most of the IT market is switching to virtualization, SDN has the potential on one hand to reduce the CApital EXpenditure (CAPEX) by limiting the need to purchase hardware material, on the other hand it reduces the OPerating EXpenditure (OPEX) by providing an automatic way to provision and optimize virtual network resources. Besides, SDN helps rapidly deploying and integrating new applications, services and infrastructures that evolve around business needs.

OpenFlow as a Southbound Protocol

Originally OpenFlow [130] was designed to test new experimental protocols on the campus of Stanford University. In 2011, the Open Networking Foundation (ONF) was created to standardize and promote the use of OpenFlow in production networks. The OpenFlow have become a part of today's SDN definition, the objective of: (i) separate the data plane from the control plane; (ii) providing network programmability with the use of a centralized controller.

OpenFlow describes the way flows are redirected using agents on the forwarding devices (switches). A flow contains: flow priority, packet processing instructions, packet match fields, flow timeout and a cookie. These flow information are pushed to switches by a centralized controller.

LISP as a Southbound Protocol

Since its standardization at the IETF, LISP did not stop gaining in notoriety in both industry and academic research [75]. It has already become a part of SDN solutions and was implemented in OpenDaylight controller [84]. As presented in [128], there are a number of elements that make LISP a suitable southbound inter-domain SDN protocol.

- LISP can toggle the traffic on-line and on-demand from one location to another pushing all the control messages to the distributed mapping system. It also features an address format, also known as LISP canonical address format (LCAF), that allows mapping from any namespace (i.e., GPS coordinates, AS numbers, etc.) to another and enables rich network programmability and traffic engineering for SDN deployments.

- The LISP data plane is programmed according to control plane policies which are stored in the distributed mapping system. Since the mapping system stores all the EID-toRLOC entries it gives centralized SDN view over the network.

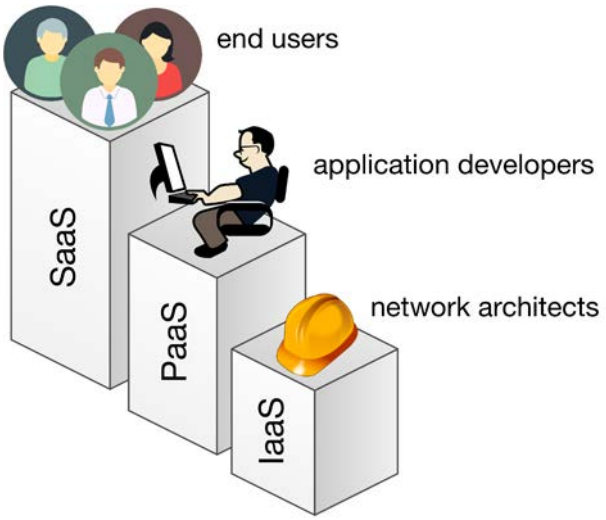


Figure 2.7: Representation of Cloud service models.

- After introducing Delegated Database Tree (DDT) [22] within the LISP mapping system architecture, the network state information and the network entities can hence be retrieved on-line, making LISP a scalable SDN solution over large Internet network topologies.

- One of the key feature of LISP is to decouple the control plane from the data plane by performing edge switching functions at the EID space and traffic forwarding at the RLOC space (i.e., can support IPv6 networks over IPv4 connection or vise-versa), which overcomes some of the SDN shortcomings described in [129]: (i) existing SDN solution such as OpenFlow [130] require switch hardware in order to perform lookups; (ii) less flexibility due to a limited feature set evolution; (iii) switching from IPv4 to IPv6 requires a change in the matching packets in the network core.

LISP is incrementally deployable on inter-domain networks and possesses all the required features to be classified as southbound SDN protocol. It has been designed to be flexible enough to bring to SDN new features and accommodate future protocols by just upgrading xTRs and the mapping system. Last, LISP can push programmable policies to the Internet (i.e., Data Center (DC)-to-DC, user-to-DC) which is more complicated for other SDN solutions to realize.

2.4 Cloud Services and Management

Cloud infrastructures can be classified into three classical categories depending on the type of services they deliver to their customers (Figure 2.7):

Software as a Service (SaaS): in this model applications are hosted on servers or VMs and are provided to customers across the Internet without the need of installing on their local physical machines. In this type of service, customers are restricted to services proposed by the provider.

Platform as a Service (PaaS): it offers a platform that allows developer to create their web applications without buying the software or maintaining the hardware underneath it. This model is still gaining community acceptance as it eases the difficulties concerning the iteration of software development. Google App Engine [155] and Microsoft Azure [156] are still good example for such a model.

Infrastructure as a Service (IaaS): is a way of delivering a variety of Cloud infrastructure on-demand (i.e., network, storage, operating systems) by means of virtual machines (VMs) as basic IaaS unit. Customers can build their own virtual infrastructure freeing them from the burden of costly hardware maintenance. Generally, this kind of service is delivered by providers that own modern data centers. Well-know providers in this domain are Amazon Web Service [158] and Rackspace [157].

2.4.1 Cloud Key Elements

Whether it is a SaaS, PaaS or IaaS, Cloud providers have to deliver to their customers a secure, reliable access to a set of virtual resources with last generation hardware under a pre-defined business model and with the best expected performance. In the following we discuss how most of today's Cloud providers tend to build their own architecture.

Virtual Resources

A key role of Cloud providers is to deliver the latest hardware freeing their users from maintenance burden. They offer pre-installed software packages known as Virtual Private Servers (VPS) hosted on hypervisors (i.e., KVM, VMWare, Xen, etc.). Generally, providers give their customers the choice between the operating system and the software pack for their VPS. In a distributed DC environment, the location of the resources with respect to users' location can have an extreme impact on the network-sensitive applications (i.e., latency, bandwidth, jitter, etc.). Unfortunately, most of Cloud providers have one geographical location available, whereas providers such as Amazon EC2 [136] have multiple distributed data centers, but do not always allow the user to choose the data center on which he needs to start its VPS.

Table 2.1: Amount of downtime under various SLA levels.

Availability	Downtime per year	Downtime per week
90% (one nine)	36.5 days	16.8 hours
99% (two nines)	3.65 days	1.68 hours
99.9% (three nines)	8.76 hours	10.1 minutes
99.99% (four nines)	52.56 minutes	1.01 minutes
99.999% (five nines)	5.26 minutes	6.05 seconds
99.9999% (six nines)	31.5 seconds	0.605 seconds
99,99999% (seven nines)	3.15 seconds	0.0605 seconds

Redundancy

Cloud providers should meet customers' expectations with respect to QoS parameters. One important feature practiced by providers is to load balance the traffic to replicated VMs in order to: (i) avoid overloading the underlying hardware resources (i.e., routers, servers, etc.); (ii) create redundant copies of VMs on different servers/DCs; (iii) minimize the latency through multiple DC access links. We distinguish other important features used by Cloud providers to ensure high resource availability on their infrastructures, such as VM resizing based on the exhibited load and checkpointing. The latter is offered by few providers and consists in creating snapshots of the running VM at any time as a fault tolerance improvement.

Security

One of the important and sensitive subject in Cloud computing is security. Providers should at least be able to offer private connections between the customers and their VMs by generating unique certificates, blocking undesirable traffic and even creating private VLANs. We highlight three common elements used by Cloud providers: (i) a set of static IP addresses is usually attached to a user account allowing a better support for name resolution, a higher reachability over the Internet avoiding address conflicts; (ii) an authentication method (i.e., login/password, X.509 certificates, etc.) improves privacy and protects against some identity theft; (iii) programmable/virtual firewalls prevent undesired connections to be established.

2.4.2 Cloud Service Level Agreements

After discussing most of key elements that a Cloud provider offers, it becomes important to define a business model that describes the payment mode, the price of the proposed services as well as the SLA that should protect both customers

and providers. Among actual IaaS providers a consumption payment model is most of the time applied by charging the customers based on the amount of resources used. The prices may vary and can be a monthly subscription fee based on VPS configuration (i.e., CPU, RAM, storage, etc.). A charge for an account application can be required.

Another important aspect is the service level agreement (SLA), which is a contract that specifies the minimum obligations a provider should offer to a Cloud user. Some Cloud providers limit their QoS terms to VM/server uptime without giving any further details, while a few providers offer complete logging. Advanced techniques such as automatic load balancing, online support 24/7 or remote assistance can be included into Cloud SLAs. Some providers absorb a penalty if an SLA term is breached, by paying back their customers [Amazon EC2]. For instance, if the uptime percentage is not fulfilled users receive a credit to their account compensating the time where the service was unavailable; as a reference Table 2.1 shows the amount of acceptable downtime under different SLA levels.

Despite this practice, SLA is still not clear to users and remain provider-oriented without any focus on customer's quality of service. For example, the quality of a network link between mobile users and their VMs can fluctuate due to mobility, leading to service disruption.

2.4.3 Quality of Service and Quality of Experience

Quality of Service (QoS) technologies help cloud providers offer a reliable connectivity to their services. We highlight in the following some of the measurable Quality of Service (QoS) metrics that can affect today's cloud applications, and network services at large :

- *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-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 [93]. 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 the delay; above 1 s there is a risk that the user abandons the service [72]. 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.

On the other side, user's Quality of Experience (QoE) typically refers to the tangible, visible performance the user experiences when consuming a digital service [68]. It is commonly presented as a desirable goal measurable with an orthogonal set of metrics (e.g., glitches, waiting times) with respect to legacy 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 have direct impact on QoE metric levels [70]. The analytic relationship between QoE control mechanisms and QoS parameters is derived in [71].

2.5 Virtual Machine Mobility

An important feature of Cloud computing system is the capability of migrating virtual services from a physical server to another, potentially also at different distant data centers. Different technologies exist to perform virtual machine migration. We can distinguish tree types of virtual machine migrations: bulk virtual machine migration, live virtual machine migration and checkpointing. While bulk migration consist into stopping the VM for a long time before transferring offline the whole VM stack (RAM and storage) to destination target [30], live migration stops the VM instance for a small amount of time and only transfers the RAM to the destination target. On the other hand, checkpointing consists in saving VM states (RAM and storage disks) and then resuming VM instance on target destination [31] from the last saved state. Recent works on checkpointing for high available cloud systems

exist, but their implementation is still at an immature stage that make them not sufficiently reliable for our purposes.

For the rest of this dissertation, we adopt live migration as virtual machine migration technique as it appears as the one guaranteeing the least possible downtime at an acceptable (low) complexity, and with solid implementations available for common hypervisors. Hence we provide a detailed description hereafter for the typical live migration process[1].

2.5.1 Live Virtual Machine Migration

Live VM migration is a feature introduced in recent hypervisors; it allows moving a running VM instance between two (physical) container hosts without disconnecting application (TCP and UDP) connections of VM's clients. For most of the hypervisors, live migration is limited to situations in which source and destination hosts look like connected to the same local area network. The main reason is that the machine being migrated needs to keep the same routing view of the network (gateway, IP subnet) before and after the migration. Alternatively, in some legacy solutions, upon migration the VM changes its IP address, e.g., via the DHCP, to avoid the additional complexity needed to ensure that the origin IP address is not already used in the destination network, and to transfer the routing table. We distinguish two major approaches [42]:

- **post-copy memory migration:** in this approach the VM is suspended at the source, processor state are then copied to the target hypervisor before resuming the VM and start copying the remainder of memory pages from the source over the network.

- **pre-copy memory migration:** in this approach the memory page are copied from the source to the target while the VM is still running. The VM then stops and starts copying dirty pages - memory pages that are changed during the copy process - before resuming on the destination hypervisor.

To perform Internet-wide migrations with IP continuity, authors in [27] and [46] propose an IP mobility solution. The logic is implemented in the hypervisor, interacting with the VM before and after its migration to update IP addresses in the VM routing table. While [27] succeeds in bringing lower service downtime compared to [46], the hypervisor has to alter VM configuration to support the IP mobility feature, which leads to scalability concerns. Moreover, as the authors state, their solution performance is expected to worsen in large-scale global live migrations,

[1]Nonetheless, from a conceptual perspective, checkpointing-based migrations could also apply as migration technique for our proposal of Chapter 4.

because of the online signaling nature of the proposition and many-way signaling latency.

Authors in [47] propose to adapt the Mobile IP (MIP) protocol [44] to pilot Internet-scale VM migrations, implementing it in the hypervisor. Their solution, called HyperMIP, is invoked whenever a VM is created, destroyed or migrated; as in MIP, it involves Home Agents (HA) to keep the connection alive. Whenever a VM changes a location, a tunnel is established between the HA and the source hypervisor to keep the client connected to the VM. The destination hypervisor then destroys the tunnel when the VM registers its new IP address to the HA. However, HyperMIP still introduces an important signaling overhead due to HA tunnel establishment.

Alternatively, to minimize signaling latency, authors in [37] propose to use an external agent to orchestrate the migration from the beginning to the end, by proactively establishing circuits between the involved containers (source and destination hypervisors) offline, so as to rapidly switch the traffic upon migration. Upon migration, the agent redirects the traffic between the VM and the client by dynamically re-configuring IP tunnels. They achieve a near second network downtime while migrating machines across wide area networks, with a maximum network downtime around 3.8 seconds. Despite being a secure approach, with respect to [27], [46] and [47] their solution involves lower-layer control plane technologies, hence can be much more costly.

2.5.2 Tunneling Cloud Solutions and Virtual Machine Mobility

The above described solutions tackle large-scale VM live migration using Layer 3 tunneling [27], [46] and [47], or Layer 3-Layer 1 interaction [37]. More recently, at the IETF, attention is given to Layer 2 over Layer 3 (L2o3) overlay tunneling solutions, so as to avoid IP reconfiguration to the VM, and service continuity upon migration. Virtual eXtensible LAN (VXLAN) [51], Stateless Transport Tunneling (STT) [52], and Network Virtualization using Generic Routing Encapsulation (NVGRE) [36], are recent propositions, some already implemented, worth being discussed.

VXLAN [51] is a stateless Layer 2 over Layer 3 logic that extends the Layer 2 communication domain over IP networks, extending a VLAN broadcast domain thanks to MAC-to-IP encapsulation between hypervisors, even if communicating VMs and endpoints are in different Layer-3 segments. Basically, when a VM wants to communicate with another VM on a different host, a 'tunnel endpoint' implemented in the hypervisor receives the frame, verifies the target VM is on the same VXLAN segment via standard signaling, and then appends an IP address corresponding to the destination tunnel endpoint, and a VXLAN header. Upon reception, the destination tunnel endpoint verifies the packet, decapsulates it and forwards it to the VM target. Therefore, thanks to the VLAN broadcast domain extension, a VM

belonging to a VXLAN segment can migrate to a VXLAN endpoint in another IP segment, and its traffic is consequently encapsulated by the source VXLAN endpoint toward the destination VXLAN endpoint.

Functionally, NVGRE [36] is a similar L2o3 tunneling solution, with a different header (VXLAN uses UDP and NVGRE does not) and with no specified control-plane to distribute MAC-to-IP mappings (in VXLAN, standard multicast mechanisms do allow resolving these mappings). Encapsulating over Layer 3 Ethernet traffic allows a better bottleneck management in IP-centric DC architectures. Both VXLAN and NVGRE, however, do not allow typical Ethernet physical interfaces to perform TCP optimization (intermediate fragmentation done by the hardware to boost performances). Stateless Transport Tunneling [52] (STT) is another stateless L2o3 tunneling protocol, which uses a TCP-like header inside the IP header so as to allow interface-level TCP optimizations. From a VM mobility and traffic routing perspective, it offers the same encapsulation path than VXLAN and NVGRE.

2.5.3 Virtual Machine Mobility on LISP

In a LISP network, the VM can keep the same IP, provided that the locator change is notified to the mapping system. Two mechanisms at the state of the art can perform this operation. One is a host-based LISP implementation called LISP-mob [20]: the host implements a tiny xTR with basic LISP functions, using the network-assigned IP(s) as RLOC(s) and registering mapping updates for its EID with the mapping servers. Essentially conceived for mobile equipment, LISPmob could also be installed in the VM; there would be, however, a problem with most current hypervisors that impose the VM external address to be in the same subnet before and upon migration, which practically limits the LISPmob usability only to situations where source and destination networks are either LISP sites themselves, or layer-2 over WAN solutions. In the first case, a double encapsulation is needed, which could increase mapping latency, overhead and MTU issues. There may also be scalability issues with a high VM number.

Another method to handle VM mobility via LISP is actually implemented in some Cisco products, only partially documented in [21]. The xTR automatically changes the mapping upon reception of outgoing data-plane traffic from an EID that has been registered as mobile node. The solution has an attracting light impact on IP operations, yet it seems to be weak against EID spoofing, and it seems not to have authentication mechanisms. Moreover, in order to guarantee fast mapping convergence, additional logic may need to be implemented in the VM or in the hypervisor to allow sending outgoing artificial data traffic even if no real outgoing traffic exist.

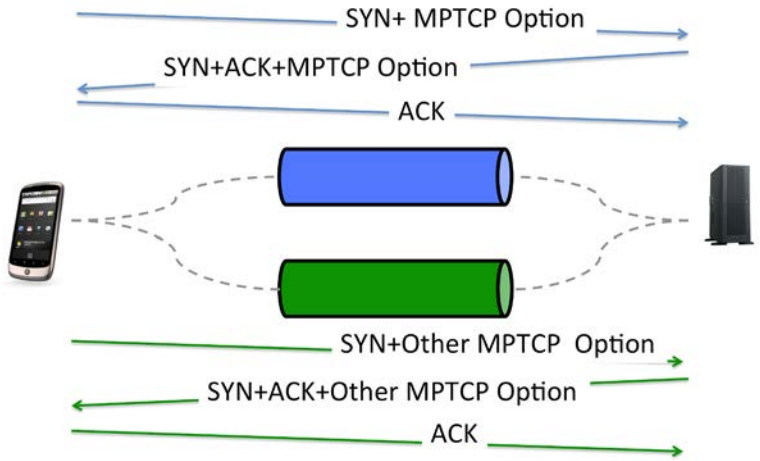


Figure 2.8: Simplified representation of MPTCP signaling. Source [140]

2.6 Cloud Acceleration Solutions

In this section we present different solutions used to accelerate the access to cloud applications.

2.6.1 Layer-7 Acceleration

As the number of web-based applications is increasing, application acceleration and persistent user sessions become important for cloud providers in order to guarantee a seamless user experience. A solution consists in replicating applications by adding multiple servers then let the Layer-4 load balancer do its work. While this works well from a networking perspective, it is harder for developers to maintain and update their applications.

Layer-7 load balancing is a way that helps developers update their applications without having to maintain multiple servers. It balances the traffic by making decisions based on the content of the message. For instance, when a user requests a video and an image from a high-traffic website, the Layer-7 load balancing will allow the traffic to be load balanced based on the contents to the servers that can handle more efficiently multimedia contents.

2.6.2 Layer-4 Acceleration

Layer-7 load balancing is CPU-intensive and requires more resources than a Layer-4 load balancer. One way to efficiently balance the traffic at Layer-4 is by using Multipath TCP (MPTCP). MPTCP is a Transmission Control Protocol (TCP) extension enabling end-to-end multipath communications, with an emphasis on back-

ward compatibility, leveraging on multiple connection subflows concurrently used for connection data forwarding. Without entering excessively into technical details, the MPTCP signaling relies on the use of one TCP option in the three-way handshake, and of another TCP option during the connection to open and close subflows as a function of connection, subflow and network states, as depicted in Figure 2.8.

As explained in [140], many middleboxes (e.g., TCP Optimizers, firewalls, etc.) are able to inspect packets either by modifying their content (e.g., change sequence number) or to dropping them (e.g., if it detects unknown protocols), and thus can hinder TCP extensions deployment. As a consequence, MPTCP refers to IPs with an identifier rather than by the IP in order to prevent the confusion induced by Address Translators (PAT/NAT). If any problem of the kind is detected by MPTCP, it falls back to legacy TCP, as it is the case if the remote endhost is not an MPTCP compliant.

Once an MPTCP connection is established, end-hosts can advertise their IP addresses, add or remove MPTCP subflows anytime. These subflows, which we could define as TCP connections, children of a more comprehensive parent TCP connection, can be used to achieve greater throughput or resiliency (indeed with the "backup" flag, MPTCP can create a subflow used only in case other subflows stop transmitting). It is worth noting that a host can create/advertise subflows with the same IP address, but with a different port number. The main challenge of such a protocol is the congestion control mechanism. It should not be more aggressive than MPTCP, but at the same time it should use the unused capacity; it should balance the load over different paths, but without causing too much packet disordering so that TCP buffering can reorder them.

Chapter 3

User-Centric Clouds

3.1 Introduction

It is indisputable that the large availability of broadband and wireless Internet has pushed users to quit their desktops and invest more time into mobile devices for their daily life. As a result, users are shifting to cloud applications for their personal or their business usages. With the Infrastructure as a Service (IaaS) paradigm, providers such as Amazon Elastic Compute Cloud (EC2) offer virtual machine instances to their customers at a competitive cost without the trouble of having them maintain physical servers [136]. These VMs are highly customizable, allowing users to choose their own configuration such as the number of vCPU, OS type, memory and even the geographical location.

Due to its complexity, the cloud computing industry is facing several uncertainties. It is shown in [137] that the performance of cloud computing can vary a lot during the day. For cloud providers it becomes important to offer a viable service by starting to respect service level agreements, at a risk to start losing valuable customers. This problem arises from the fact that service providers tend to optimize their physical resources in order to increase their profits, at the expense of user experience.

In this Chapter we address the expectations and benefits of the key cloud stakeholders: providers and users. We explain in details what cloud users expect from their cloud services. We highlight the benefits that arise from using virtualization. We then show how we can build holistic networking architectures that leads to user-centric clouds. Finally, we present our reference cloud network architecture.

3.2 Cloud User's Expectations

It is clear that cloud applications have released the user from the obligation of having one device at one location. Therefore users have become more mobile and started using different end-devices to access their web-based cloud applications from different locations. Their demand over high availability, fast connectivity and less service disruption has increased over time, leaving no room for errors for the cloud providers.

To illustrate the expectation of mobile cloud users we give in the following example scenario: employees who spend most of their time roaming outside their office need a flexible solution to stay connected to the cloud infrastructure of their company releasing them from the burden of terminal devices: virtual desktop infrastructures (VDI). VDIs are virtual machines that host desktop operating systems (i.e., Windows, Linux, etc.) connected to other local services on a data center (i.e., database, lightweight directory access protocol (LDAP), etc.). The benefits of such service are numerous: (i) it makes easier to deploy applications that need different operating systems; (ii) it allows staying hooked to the company database in the cloud; (iii) it allows the access to the service anywhere at anytime. However, VDI still depend on the network infrastructure between the user and the virtual machine. Indeed, if a VDI user in France is traveling to the US, he will most probably suffer from an increasing latency which will affect the quality of its experience, making the service sometimes unusable. Knowing that the cloud provider has already a DC in the US or part of a collocation center in the same country, a solution is to migrate the VDI service to a close data center. Besides mobility, VDI users seek a seamless experience with a service continuity. For instance, when a network link fails the service should go on without any noticeable service disruption, by switching the traffic either to another DC access gateway or simply to another link of a multi-homed cloud provider.

Even though there are no formal definitions of what users should/could expect from their cloud providers whether they subscribe for a SaaS, IaaS or PaaS and based on what has been shown earlier in the example above we enumerate the following tangible user metrics:

- **Data Privacy:** when cloud customers are asked about the best service provider they prefer, many of them reply by asking about the offered security tools (i.e., encryption, certificates, etc.) that help keeping their data and connections safe. A cloud user expects rigorous requirements when fetching data that involve personal activities like identity management. Above all, the data they store in the cloud becomes uncompromisingly their own. Securing data should not by any mean affect the quality of the user's experience.

- **Continuity:** most of cloud users nowadays possess multiple smart end-devices (i.e., smartphones, laptops, tablets, etc.), which is challenging for providers to keep their customers' data synchronized. If a user is working on its laptop and wants to switch to a smartphone or a tablet, the data should be synchronized with the latest modifications. For users, cloud services become better if they go seamless and unnoticed thus, invisibility becomes a metric on which a cloud application is based on.

- **Access on-the-go:** whether the users are at their home, in their office or on the train, they want to access their cloud services everywhere with no interruptions. For cloud users distance should feel no different from the local access. They want to always have closer virtual resources to gain more control on the quality of their experience, thus when a service disruption occurs a seamless experience must keep going.

- **Latency:** today's web-based applications and real-time applications have become part of our daily life. How data is transferred from one location to another may add some frustrating delays to such services. Besides the Internet connectivity of end-users, latency can be affected by several flaws in the design architecture of cloud providers. For instance, if a user in France has a running service on a data center in the United States, he may not have the expected latency. Bringing the virtual resources close to the user becomes part of a seamless experience.

3.3 Cloud Provider Needs

Even if a cloud provider has one data center or is part of a colocation center, he needs to ensure network resiliency to its customers by offering them a multi-homed/distributed architecture. For instance, if a network link is cut and the cloud provider doesn't have another backup link, the customer won't be able to access its service anymore, leading to a monetary compensation for the disruption period. A cloud provider should be able to prevent such failovers and put multi-homed services at their customers' disposal.

To build stringent cloud architectures at minimal deployment costs, cloud providers are virtualizing their infrastructure with a small software overhead. This amount is negligible due to the fact that servers are becoming faster and less resource greedy. We highlight in the following the benefits that a provider acquire by virtualizing its infrastructure:

- **Hardware Cost:** cloud providers can typically save much more money by virtualizing their resources. Instead of using multiple physical servers for

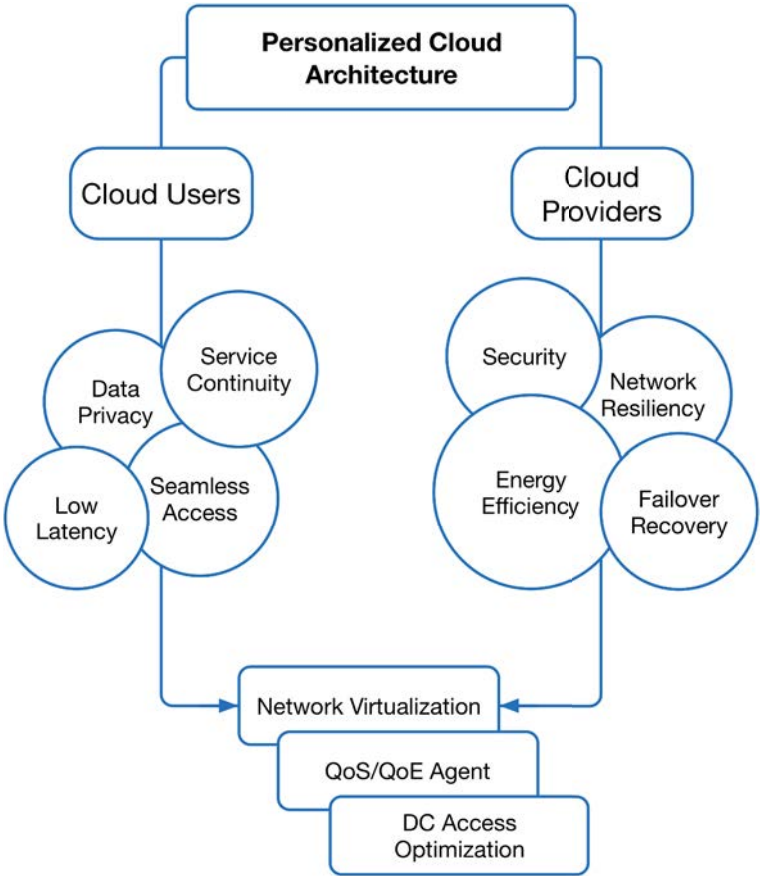


Figure 3.1: Key elements for a user-centric cloud architecture.

each service, they can manage to put a plethora of applications on one server instead. Hence, server consolidation reduces considerably the cost footprint on the data center with less servers to manage, less networking equipments to install and fewer racks to deploy.

- **Energy Consumption:** virtualization reduces the electricity consumption by half. Migrating physical servers to virtual machines and consolidating them into few physical servers lower the cooling costs in the data center.

- **Disaster Recovery:** fast recovery from a natural disaster or hardware failures is one of the biggest advantages of virtualization. In fact, VMs can be migrated from a failing data center to another one instantaneously with almost no downtime at all. Thanks to hardware abstraction, it is no longer needed to keep the same replicated hardware environment in two different locations. Nowadays, most of virtualization platform provide a software that can help recovering from disaster failovers.

- **Network Resiliency and Security:** server consolidation brought a full con-

trol over virtual network equipment lowering the cost. SDN provides a general overview of the network topology making it easier for providers to manage and provision their virtual network infrastructure. It has also an important impact on lowering the operating costs since many of the network administration is centralized and automated. Moreover, it provides a centralized point to distribute the security policies and information automatically throughout the global data center architecture.

3.4 Holistic Cloud Networking Architecture

When subscribing to a cloud service, on one hand users expect at least to have a continuous connectivity with a minimum disruption and on the other hand providers want to maximize their profits with respect to Service Level Agreement (SLA). In the following we identify the key elements for a user-centeric cloud architecture (Figure 3.1):

- **Network resource virtualization:** the combination of network programmability and system virtualization have brought multiple advantages to today's cloud architectures. It certainly reduced the cost of specific hardware appliances (i.e., switch, routers, firewalls, etc.) by replacing them with cheaper generic hardware and advanced flexible software. The data plane has been abstracted, allowing the coexistence of multiple protocols as well as a better evolution without the need of upgrading or replacing the existent hardware stack. When it comes to users accessing their mobile virtual resources in a distributed data center environment, virtualizing the entry points to the cloud becomes important for multihomed providers in order to deliver a seamless connection.

- **QoS feedback:** QoS metrics are defined by providers (i.e., cloud providers, ISPs, data centers, etc.) in order to provide high performance for critical applications. When providers want to prioritize one service over another, QoS metrics (i.e., latency, jitter, bandwidth, etc.) seem to be always essential as a quantitative measure for respecting the agreed-upon level of SLA. Therefore, monitoring the link between users and their VM helps identifying weaknesses and needs over virtual links, which gives the cloud provider the opportunity to offer a stable service to their users. Such information can be gleaned constantly by an agent that is implemented at the cloud infrastructure level.

- **QoE feedback:** whether it is on the cloud or locally, QoE is a subjective metric that measures user's satisfaction while using a particular application. It is necessary for providers to have a feedback on the quality of their offered

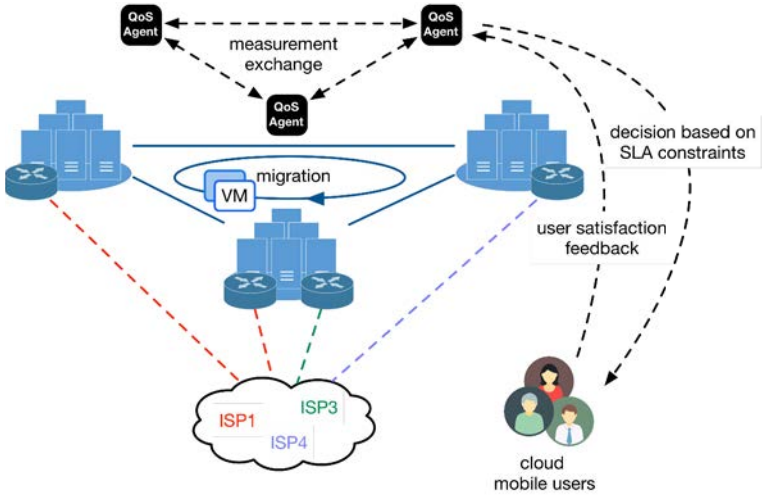


Figure 3.2: QoS/QoE interactions.

cloud services. Even though this kind of experience remains subjective, asking customers about their experience can help cloud providers to better assess and control their virtual infrastructure by adaptively predicting the behavior of their customers. This kind of metric can be implemented at the service level, which gives the user the chance to express his satisfaction before disconnecting from an application (i.e., Skype [78], WhatsApp [79], and other VoIP cloud services always ask about the consumer's experience after each call).

- **Access optimization:** one of the most important challenges is to keep the connection between a cloud application and a user established even when a network link failure occurs. The demand over mutihomed services has increased over the last decade and has become part of every resilient cloud architecture. Whether it is used for load balancing, or for identifying the best DC entry point for cloud applications, multihoming remains the best solution for an optimized access to the cloud.

Figure 3.2 shows the interaction between different elements in a so-called user-centric cloud architecture: (i) a multihomed distributed data center architecture with different entry points and different service providers; (ii) cloud users that can be either mobile or non-mobile; (iii) virtual machines with unique IP addresses, and that can be migrated on-demand across DCs; (iv) distributed QoS agents that are able to glean metric information over the user-to-VM links and find the best entry point(s) to the VM.

3.4.1 Quality of Service and Quality of Experience

QoS and QoE are both important for service providers: QoS technologies provide a solid network base on which cloud applications can run (i.e., online gaming, media streaming, video-conference, etc.), while QoE evaluation processes can ensure that the offered service stays acceptable for the user. Although both terms are sometimes inter-changeably sued, it becomes essential to identify how QoS can be related to QoE in: (i) a single ISP-single data center architecture; (ii) a multiple ISPs-single/distributed data center architecture.

- **Single-homed DC:** in this scope, a cloud provider has one single access gateway to its cloud services. Although this kind of architecture is easier to deploy, the existence of a single point of failure (SPOF) gateway remain a major problem. In fact, users are connected to a single entry point located on a single DC, following passive and not easily controllable BGP routing paths (assuming the Internet is used for the access). When an impairment (i.e., high latency/low bandwidth, disaster failover, etc.) occurs on the user-to-application link will automatically affect the running service, leading to a non-exploitable application. In this kind of situation, QoE stays unique to each user but depends highly on the QoS delivered by the provider.

- **Multi-homed distributed DC:** when introducing multihoming in a single/distributed DC architecture, it is clear that the SPOF problem can be resolved from a networking perspective. It is worth noting that a multiple ISPs-single DC use case architecture still presents a SPOF from a system perspective (if the DC fails the site will go down as well as the services). In these kinds of architecture the QoE remains unique to each cloud user, but the multiple access links to a single/distributed DC(s) present a different QoS proper to each ISP. Therefore, it becomes more difficult to link user's satisfaction for an application to multiple abutting QoS delivered by different service providers without an intermediary (agent) that gleans information from each of these links.

3.4.2 Cloud Network Overlays: Requirements and Specifications

Cloud providers need to manage and optimize their virtual network resources across their distributed fabric at a minimal cost. Several protocol design efforts have been made to: (i) efficiently use the underlying network links to achieve high performance and availability; (ii) accommodate other new protocols and network solutions; (iii) program the network and redirect the traffic on-demand. We identify three protocol design goals that motivate our reference protocol architecture:

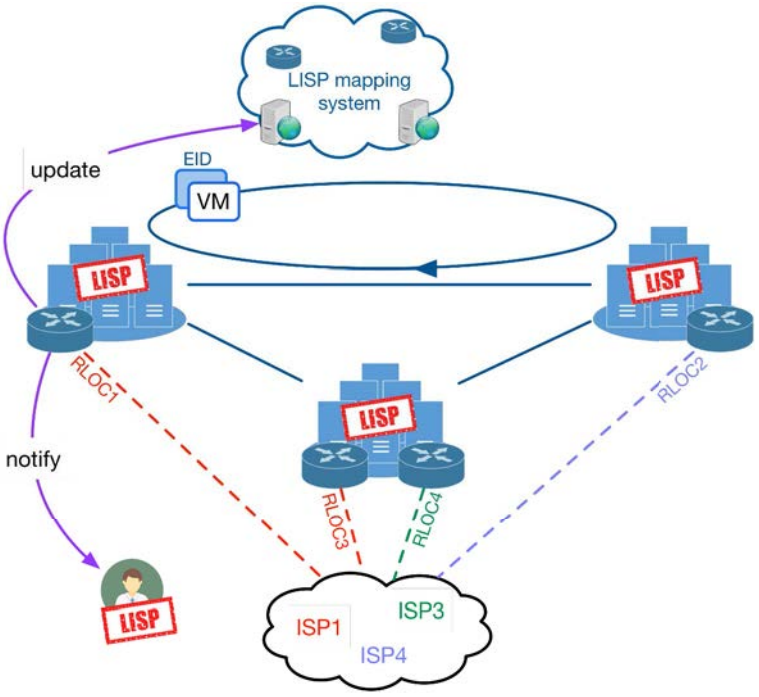


Figure 3.3: LISP virtual machine migration.

- **incremental deployability:** a novel network protocol should be incrementally deployable guaranteeing interoperability with the existing architecture and legacy systems;
- **routing locator/endpoint identification separation:** separating the identifier from the location for networks is required to support universal mobility of both users and services, as well as to simplify distributed edge network facility management;
- **cloud network segmentation:** three network layers are involved in the cloud service provisioning in a distributed data center solution: user-VM layer, the cloud access layer and the intra-DC layer. Protocols acting at each layer can be different but shall share common control plane elements.

Based on these high-level requirements, we identified three recent protocols appropriate to meet the requirements at each layer: LISP, TRILL and MPTCP (already presented in Chapter 2). These protocols have undergone a complete standardization and industrialization in the last few years. They have been designed with incremental deployability and locator/identifier separation in mind. More precisely:

- **Locator/ID Separation Protocol (LISP):** its implementation uses the current IP technology and do not hinder the existing BGP infrastructure [108],

Table 3.1: LISP vs. GRE vs. VPN.

	LISP	GRE	VPN
protocol overhead	medium	low	high
control plane features	yes	no	no
firewall friendliness	high	low	medium
incremental deployability	high	high	medium
open source implementation	yes	yes	yes

Table 3.2: TRILL vs. SPB vs. OpenFlow.

	TRILL	SPB	OpenFlow
multipath forwarding	yes	yes	yes
scalability	high	low	low
incremental deployability	high	medium	low
open source implementation	yes	no	yes

Table 3.3: MPTCP vs. SCTP.

	MPTCP	SCTP
overhead protocol	low	medium
middlebox friendliness	high	low
backward compatibility	yes	no
open source implementation	yes	yes

making it incrementally deployable. It also separates the routing locators from endpoint identifiers by using encapsulations and decapsulations on edge routers.

- **Transparent Interconnection of Lots of Links (TRILL):** with TRILL, Ethernet frames can travel across legacy clouds seamlessly, making it incrementally deployable [113]. By using Layer-2 encapsulations the locator is separated from the end-point identifier. It is suitable for the intra-DC network segment.

- **Multipath TCP (MPTCP):** does not require any modification to the actual TCP implementation [12] and can rely on MPTCP proxies [32], making it incrementally deployable. It is suitable for user-to-application load balancing segment.

It is worth noting that our protocol choice is certainly not binding; new alternative protocols that can meet the same requirements may also be acceptable as

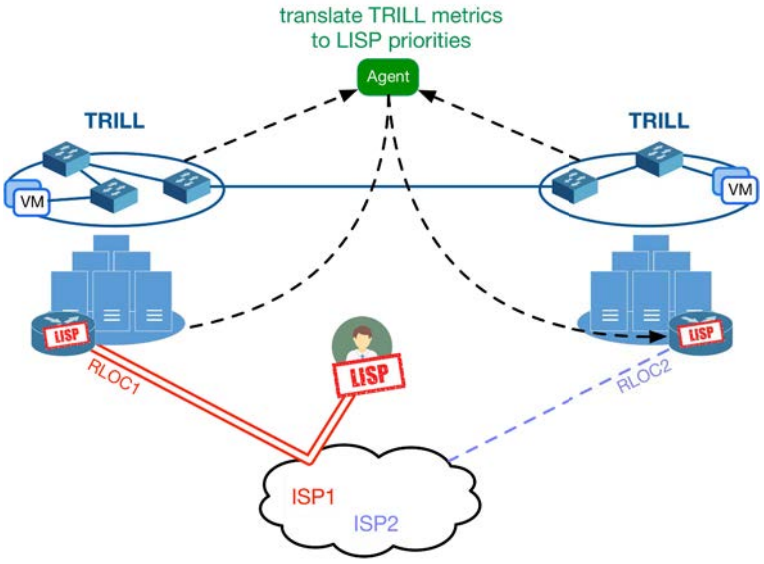


Figure 3.4: TRILL-LISP cloud architecture.

shown in tables 3.1, 3.2 and 3.3: for example, OpenFlow switching for the intra-DC segment can be a good candidate but does not scale well, on the other hand SPB does not scale well and do not have an open source implementation. On the other hand, Generic Routing Encapsulation (GRE) and Virtual Private Network (VPN) lack control plane features in order to be used as a cloud access overlay for mobile users, while Stream Control Transmission Protocol (SCTP) has a larger overhead protocol and lacks network support in current networking equipment to be used as a user-to-application load balancing segment.

In the following we describe the collaborative research context that has supported our reference architecture. We worked on three different projects:

- **NU@GE project:** when the demand over an application vary, VMs can be pushed to new compute nodes locally or across DCs without session disruption to save energy cost [127]. Resource allocations and VM migration were among the challenging objectives in NU@GE project [80] in terms of user's availability and carbon footprint reduction. In this scope, we concentrated our contribution on reducing the downtime when VMs migrate across distributed DCs, by providing and building a framework that allows seamless virtual machine migration as shown in Chapter 4. In Figure 3.3, we use LISP protocol as a cloud access overlay in order to: (i) locate the public identifiers of VMs; (ii) change and update mapping system on-demand; (iii) notify users that the VM has changed location using a single-hop LISP message.

- **ANR LISP-Lab project:** thanks to the LISP-Lab project [75], an exper-

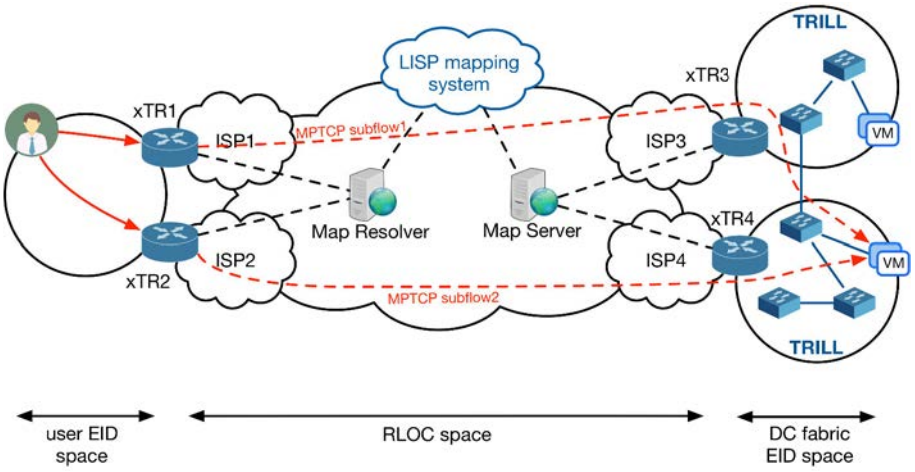


Figure 3.5: Reference cloud networking architecture.

imental platform project on the LISP protocol, we designed and tested a combination of protocols on a distributed international LISP platform. In Chapter 7 we combine LISP and TRILL protocols via an agent that has the role of translating TRILL metrics into LISP priorities giving providers a better way to automatically manage LISP entry point priorities (Figure 3.4).

- **ANR ABCD project:** thanks to the ABCD project [77], on the adaptability of a distributed cloud infrastructure to changing usages and mobility behaviors, we could have access to real user mobility traces to evaluate the impact of our solution to adaptively pilot virtual machine orchestration as a function of user mobility and network states described in Chapter 5.

- **RAVIR project:** most of the times, in a distributed DC fabric, cloud providers have the possibility to offer entry points to their customers especially for mobile users. The challenge is to be able to meet users' expectations by redirecting the traffic and the resources between multihomed DCs. RAVIR project [76] concentrates on the conception of a distributed cloud architecture with multiple operator entry points. In this scope, we provide in Chapter 5 a versatile platform that can be used with any overlay access protocol in order to reconcile both cloud users and providers via a QoS agent. The agent is based on LISP as a multi-operator access protocol and has the following purposes: (i) monitors the users-to-VM connections; (ii) optimize the entry point to the VM over the Internet by switching the routing locator; (iii) optimize the VM location across distributed data centers. In Chapter 6 we use MPTCP as socket overlay on top of LISP in order to offer different physical LISP path to each subflow, hence maximizing the transfer rates.

We represent in Figure 3.5 our holistic user-centric cloud network architecture run and operated by leveraging on the presented protocols. The missing bricks include:

- how to deal with user and virtual machine mobility while guaranteeing seamless cloud service provisioning;
- how to link LISP and TRILL control planes in support of these operations;
- how to link MPTCP and LISP control planes to increase connection resiliency.

We will address these aspects in the remainder of the dissertation.

3.5 Remainder of the dissertation

In the rest of this dissertation we focus our research on the network protocols and algorithms needed to build our holistic user-centric cloud networking architecture. We provide in Chapter 4 a solution that uses LISP for seamlessly migrating IP addresses across distributed cloud fabrics. In Chapter 5, we specify a controller architecture able to orchestrate mobile cloud services and to switch users' traffic and virtual machines from one DC to another, based on the quality metrics of the user-to-VM link. In Chapter 6 we propose an enhancement of our architecture using MPTCP (Layer-4) in order to maximize the user's throughput. In Chapter 7, we describe how to combine TRILL and LISP control-planes in order to automatically manage routing locator switching. Finally, in Chapter 8 we conclude this dissertation.

Chapter 4

Large-Scale Virtual Machine Migration

The rapid growth of Cloud computing services is putting the network communication infrastructure under enormous stress in terms of resiliency and programmability. This evolution reveals several missing blocks of the current Internet Protocol architecture, in particular in terms of VM mobility management for addressing and locator-identifier mapping. In this Chapter, we bring some changes to the LISP to cope this gap. We define novel control plane functions and show how VMs can keep the same IP address while changing locations without service interruption. We evaluate our solution in the worldwide public LISP test bed, involving LISP sites distant from a few hundred kilometers to many thousands kilometers. We show that we can guarantee a service downtime upon virtual machine migration lower than a second across American, Asian and European LISP sites, and down to 300 ms within Europe. Moreover, we discuss how much our approach outperforms standard LISP and triangular routing approaches in terms of service continuity as a function of DC-to-DC and user-to-DC distances.

4.1 Introduction

In virtualization nodes, hypervisors are software-level abstraction modules essential to concurrently manage several VMs on a physical machine. VM migration is a service included in most hypervisors to move VMs from one physical machine to another, commonly within a DC. Migrations are executed for several reasons, ranging from fault management, energy consumption minimization, and quality of service improvement. In legacy cloud networks, VM location was bound to a single facility, due to storage area network and addressing constraints. Eventually, thanks to high-speed low-latency networks, storage networks can span metropolitan and wide area networks, and VM locations can consequently span the whole Internet for public clouds.

Multiple solutions are being experimented to make VM locations independent. The main trend is to allow transparent VM migrations by developing advanced functionality at the hypervisor level [25]. In terms of addressing, the main problem resides in the possibility of scaling from private clouds to public clouds, i.e., seamlessly migrating a virtual server with a public IP across the Internet. Multiple solutions exist to handle addressing issues, ranging from simple ones with centralized or manual address mapping using Media Access Control (MAC)-in-MAC or IP-in-IP encapsulation, or both, to more advanced ones with a distributed control plane supporting VM mobility and location management. Several commercial (non-standard) solutions extend (virtual) local area networks across wide area networks, such as [39], [34], and [43] differently handling Layer-2 and Layer-3 inter-working.

Among the standards to handle VM mobility and addressing issues, we can mention recent efforts to define a distributed control plane in TRILL architecture [18] to manage a directory that pilots Layer-2 encapsulation. However, maintaining Layer-2 long-distance connectivity is often economically prohibitive, a too high barrier for small emerging Cloud service providers, and not scalable enough when the customers are mostly Internet users (i.e., not privately interconnected customers). At the IP layer, the addressing continuity can be guaranteed using ad-hoc VM turntables as suggested in [37], or Mobile IP as proposed in [27]. However, at the Internet scale, these approaches may not offer acceptable end-to-end delay and downtime performance, because of triangular routing and many-way signaling.

More recently, LISP [54], mainly proposed to solve Internet routing scalability and traffic engineering issues, is now considered for VM mobility and has already attracted the attention for some commercial solutions [21]. In order to efficiently handle locator-identifier mappings, LISP offers a distributed control plane, decoupled from the data plane. An advantage of LISP is that it can avoid triangular routing, with encapsulations performed at the first IP-LISP node. Nevertheless,

based on current standards and literature, there are missing functionalities to guarantee low VM migration downtime with LISP. Moreover, those experiences cannot be reproduced in absence of open source public solutions.

In this Chapter we define novel LISP functionalities to obtain high performance in large-scale VM live migrations. We provide all the elements to reproduce the results including reference to an open source implementation of our proposition. Our solution is based on the definition of LISP control plane messages to fast update ID-locator mappings, hence overcoming the long latency of basic LISP mechanisms. We validate and evaluate our solution using the worldwide public `www.lisp4.net` and `www.lisp-lab.org` test beds, piloting five LISP sites in four countries worldwide.

In the following, we describe our protocol extension proposition in section 4.2. In section 4.3, we report experimental results on a real test bed. Section 4.4 summarizes the Chapter.

4.2 Proposed LISP-Based VM Migration Solution

We propose a novel solution to support Internet-scale VM live migration exploiting LISP protocol. We implemented our solution in the open source LISP control plane implementation available at [50], which complements the OpenLISP data plane [23].

In live migration techniques we should be able to move a VM keeping its unique EID, from its actual DC to a new DC maintaining all VM sessions active. As a preliminary step, the source and destination DCs have to share the same internal subnet, i.e., the VM unique EID should be routable beyond its RLOC, wherever it is. LISP supports a large number of locators, and it does not set constraints on RLOC addressing – i.e., the RLOCs can take an IP address belonging not simply to different subnets, but also to different autonomous system networks. The current VM location can be selected leveraging on RLOC metrics. We introduce two main enhancements to LISP:

- a new LISP control plane message to speed up RLOC priority update;
- a migration process allowing hypervisor-xTR coordination for mapping system update, thus allowing a seamless IP migration.

Our solution involves the following network nodes: the source VM host and the destination VM host, both managed by a hypervisor; the VM being migrated from one host to the other; LISP border routers at the source DC and at the destination DC; the cloud user accessing the VM.

Following this section we present the novel LISP control plane message we introduced for communications between the involved LISP nodes, then we describe the VM migration process, and finally we discuss implementation aspects.

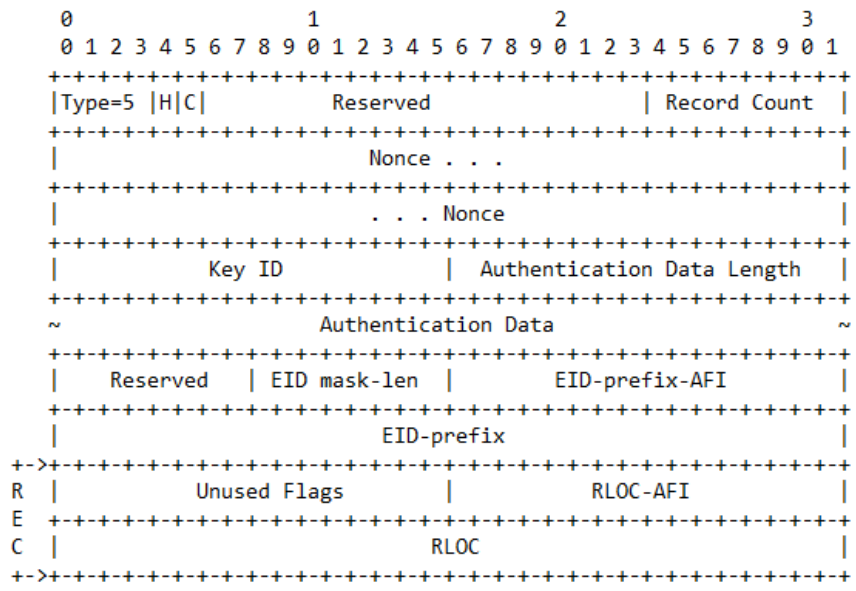


Figure 4.1: Change-Priority message architecture.

4.2.1 Change Priority Message Format

We introduce a new type of LISP control plane message that we call Change Priority (CP). As depicted in Figure 4.1, we use a new control plane type field value equal to 5[1], and use two bits to define message sub-types to be managed by both xTR and hypervisors:

- **H-bit (Hypervisor -bit)**: this bit is set to 1 when the message is sent by the destination hypervisor (the hypervisor that receives the VM), indicating to the xTR that it has just received a new EID. With the H-bit set, the record count should be set to 0 and the REC field is empty;
- **C-bit (Update Cache -bit)**: this bit is set to 1 when an xTR wants to update the mapping cache of another xTR. With the C-bit set, the record count is set to the number of locators and the REC field contains the RLOC information to rapidly update the receiver mapping cache.

The other fields have the same definition as the Map-Register message fields [54], i.e., with EID and RLOC fields, a nonce field used to guarantee session verification, and Keyed-Hash Message Authentication Code (HMAC) authentication fields useful to secure the communication (the authentication key used for CP messages can be different than the one used by Map-Register; provided by the xTR it is able to handle different keys as shown in [50]).

[1]A preliminary version of this new control plane message has been presented at the first LISP Network Operator Group (LNOG) - `http://www.lisp4.net/lnog`.

Algorithm 1 CP processing.

Ensure: authenticity of CP message
  extract EID from EID-prefix field
  if H-bit is set to 1 **then**
    set own RLOC priority to 1
    send CP to xTR group with H-bit and C-bit set to 0
    register mapping to Map Server
  end if
  if H-bit and C-bit are both set to 0 **then**
    set own RLOC priority to 255
    set locators' priority in RLOC field to 1
    send CP with C-bit set to 1 to all xTRs that have requested the VM EID
    stop registering for EID
  end if
  if C-bit is set to 1 **then**
    update mapping cache according to received message
  end if

4.2.2 VM migration process

The LISP mapping system has to be updated whenever the VM changes its location. Before the migration process starts, the xTRs register the VM EID as a single /32 prefix or as a part of larger EID (sub-)prefix. The involved devices communicate with each other to *atomically* update the priority attribute of the EID-to-RLOC mapping database entries. The following steps describe the LISP-based VM migration process.

1. The migration is initialized by the hypervisor hosting the VM; once the migration process ends, the destination hypervisor (the container that receives the VM) sends a CP message to its xTR (also called *destination xTR*) with H-bit set to 1, and the VM EID in the EID-prefix field.

2. Upon reception, the *destination xTR* authenticates the message, performs an EID-to-RLOC lookup and sets the highest priority to its own RLOCs. Then, it sends a CP message, with H-bit and C-bit set to 0, to update the mapping database of the xTR that was managing the EID before the migration (also called *source xTR*).

3. Before the VM changes its location, the *source xTR* keeps a trace log of all xTRs that have recently requested it (we call them *client xTRs*), i.e., that have the VM RLOCs in their mapping cache.

4. When the *source xTR* receives the CP message from the *destination xTR*, it authenticates it and updates the priorities for the matching EID-prefix entry in its database.

5. To redirect the client session, there are two different possibilities, whether the *client xTR* is a standard router not supporting CP signaling (e.g., a Cisco router implementing the standard LISP control plane [54]), or an advanced router including the CP logic (e.g., an OpenLISP router with our control plane functionality implemented in [50]).

    - For the first case, the *source xTR* sends a SMR to standard *client xTRs*, which triggers mapping update as of [54] (Map-Request to the MR and/or to the RLOCs, depending on the optional usage of proxy mapping, followed by a Map-Rely message to the xTR).
    - For the second case, in order to rapidly redirect the traffic to the VM's new location (*destination xTR*), the *source xTR* sends a CP message with C-bit set to 1 directly to all the *client xTRs*, which will therefore process it immediately (avoiding at least one client xTR-MR round-trip time).

6. Upon EID mapping update, the *client xTRs* renew their mapping cache and start redirecting the traffic to the new VM routing locator(s).

4.2.3 Implementation aspects

The proposed solution has been implemented using open-source software (see [50]), and its implementation involves both the hypervisor and the xTR sides.

On the hypervisor

we integrated a new function that interacts with libvirt (a management kit handling multiple VMs in KVM hypervisor) [120] to trigger CP messages. When a live migration starts, the hypervisor creates a "paused" instance of the VM on the destination host. Meanwhile, libvirt monitors the migration phase from the start to the end. If the migration is successfully completed, libvirt checks if the VM is running on the target host and, if yes, it sends a CP message to its local xTR on the UDP LISP control port 4342. The VM EID is included in the EID-prefix field.

On the xTR

we implemented the Algorithm 1 function in the open-source control plane [35], providing control plane features to the OpenLISP data plane [23]; the OpenLISP data-plane router runs in the FreeBSD kernel, while the control plane runs in the

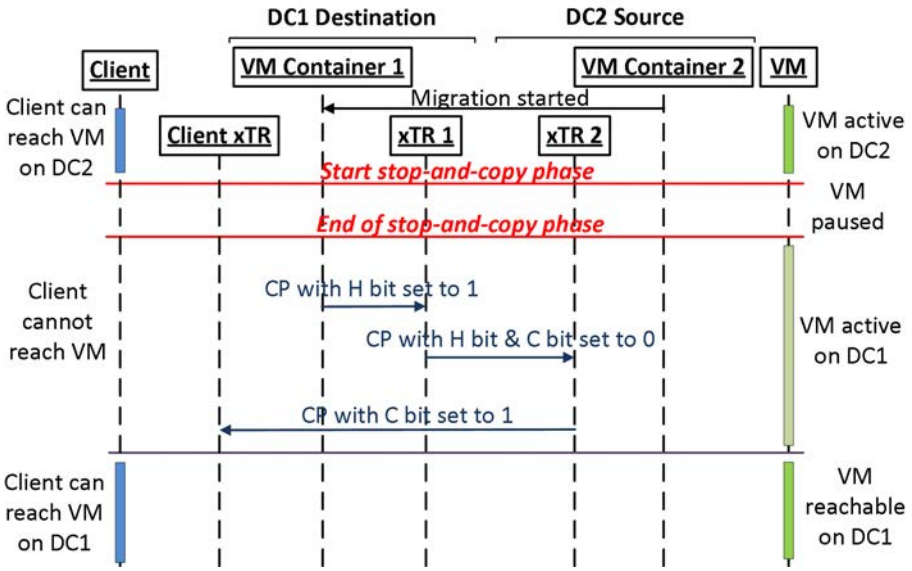


Figure 4.2: Example of CP signaling exchange during a VM migration.

user space. A new feature is implemented to capture control plane message type 5 and the logic to handle CP signaling.

A signaling example

upon client request, or a consolidation engine, a VM needs to be migrated to another public DC. As in the Figure 4.2 example, Container 2 starts migrating VM from DC2 to DC1 while Client is still connected. When the migration reaches the stop-and-copy phase, the VM stops and begins transferring its last memory pages. Meanwhile, Client loses the connection but keeps routing to DC2.

The hypervisor on Container 1 detects that VM is now successfully running, indicating the end of the migration. Then Container 1 announces that the VM has changed its location by sending a CP message with the H-bit set to 1 to xTR 1. Upon reception, xTR 1 sends a CP with H-bit and C-bit set to 0 to notify xTR 2 about the new location of the VM: xTR 1 updates the priorities for VM EID entry in its database.

When xTR 2 receives the CP message, it matches the EID-prefix to the entries within its mapping database, and modifies the priorities accordingly, then it stops registering VM EID. As mentioned in Section 4.2.2, xTR 2 keeps a trace log of all the RLOCs that recently requested the VM EID. In this use case, Client is the only one communicating with VM. As a result, xTR 2 sends a CP with C-bit set to Client xTR.

Finally, Client xTR receives the CP message, maps VM EID, and nenew its cache, then starts redirecting Client's session to the new VM location (DC1).

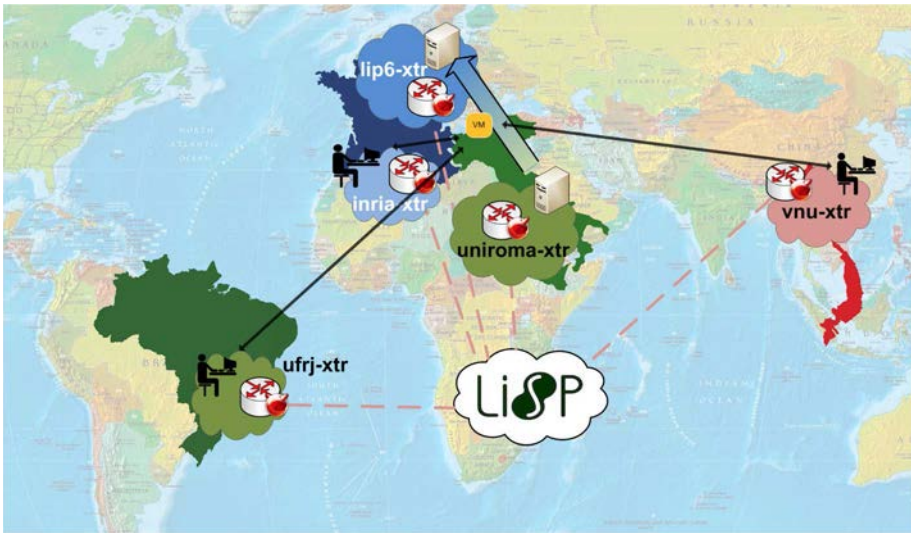


Figure 4.3: LISP test bed topology.

4.3 Test Bed Evaluation

We performed live migrations of a FreeBSD 9.0 VM, with one core and 512 MB RAM (corresponding to a typical service VM like a lightweight web server), from UROMA1 (Rome) to LIP6 (Paris), under KVM [24]. Figure 4.3 gives a representation of the test bed topology. As distributed storage solution, we deployed a Network File System shared storage between source and destination host containers. Hence, only RAM and CPU states are to be transferred during the live migration. The VM containers are Ubuntu 12.04 servers, dual core, with 2048 RAM and using KVM and libvirt 0.9.8.

We measured many parameters during migrations, by 20 ms spaced pings from different clients: distant ones at VNU (Hanoi, Vietnam), UFRJ (Rio de Janeiro, Brazil) LISP sites, and a close one at the INRIA (Sophia Antipolis, France) LISP site. We should note that:

- the clocks on all LISP sites were synchronized to the same Network Time Protocol stratum [49], so that a same VM migration can be monitored concurrently at the different client sites;

- all xTRs register to a same MS/MR located in Denmark (`www.lisp4.net`), hence avoiding the DDT latency in the mapping convergence (as already mentioned in Section 2.2).

We performed hundreds of migrations from UROMA1 to LIP6 sites, over a period of 3 months at different times of the day. We used two possible inter-xTR mapping update modes with the proposed control plane enhancement: SMRs to

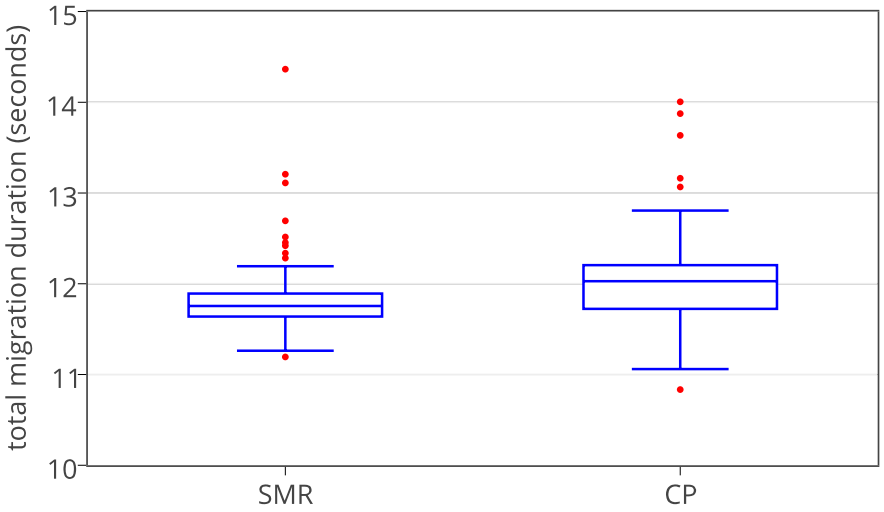


Figure 4.4: Total migration duration (boxplot statistics).

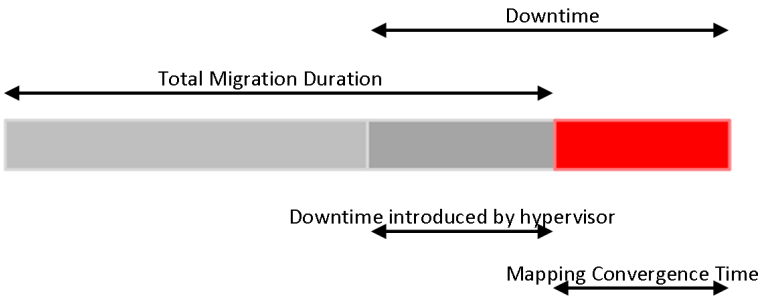


Figure 4.5: Migration duration and downtime composition.

simulate standard client xTRs, and CP to encompass the case with enhanced xTRs at client LISP sites. Given the Internet wideness of the test bed, both the bottleneck bandwidth and RTTs were floating, depending by the time and day, hence we did a statistical evaluation as described hereafter. The average measured RTTs between each site during the migration are reported in Table 4.1; having both close and far client sites allow us to precisely assess the migration performance over LISP protocol.

In order to experimentally assess the relationship between different time components and network situations, we measured these different parameters:

- number of lost packets for each client (i.e., the number of ICMP messages that are lost on each client during migration);
- mapping convergence time for each client: the time between the transmission of CP by the hypervisor and the mapping cache update on each client.
- downtime perceived by each client: the time during which the client could not

Table 4.1: Average measured RTT during migrations.

LISP Sites	Average RTT
LIP6-UROMA1	30.47 ms
LIP6-VNU	299.86 ms
LIP6-INRIA	16.47 ms
LIP6-UFRJ	246.07 ms
UROMA1-VNU	321.27 ms
UROMA1-INRIA	27.27 ms
UROMA1-UFRJ	259.13 ms

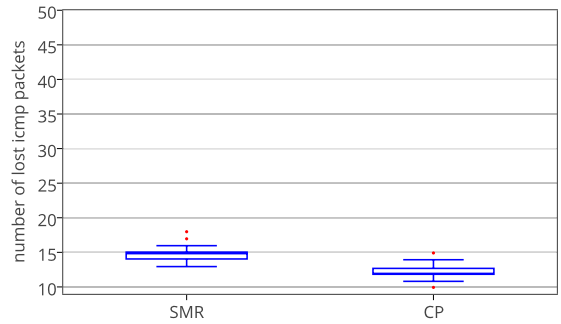
communicate with the VM;

- total migration duration;
- inter-container bandwidth during migration;
- offset for each client: the difference between the clocks on each client and the NTP server;
- RTT between the VM and the clients.

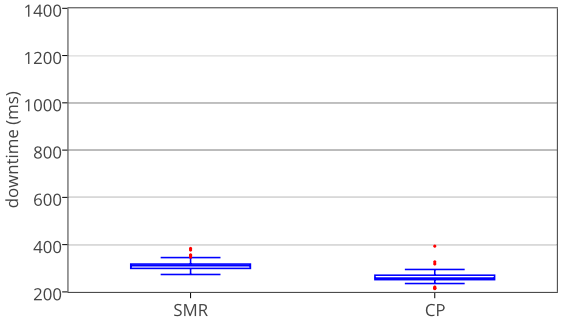
For the sake of completeness, we report in Figure 4.4 statistics about the total migration duration. It has a median around 11.75 s for both signaling modes. It includes the downtime introduced by the hypervisor (pre-copy memory migration: stop-and-copy phase), not including the service downtime. As depicted in Figure 4.5 and as of previous arguments, it is worth underlining that one should expect that: $downtime \geq$ downtime introduced by the hypervisor (stop-and-copy[2] duration) + the mapping convergence time. The stop-and-copy duration depends on the volume of the last memory pages to be transferred that, with standard tools, we do not control. The mapping convergence time reflects our protocol overhead, which is differently affected by the RTT between LISP sites (Table 4.1) depending on the client xTR support of CP signaling.

In order to characterize absolute service downtime suffered by clients, Figures 4.6,4.7, and 4.8 report the boxplots (minimum, 1^{st} quartile, median with the 95% confidence interval, 3^{rd} quartile, maximum, outliers) of the obtained number of lost packets, offset, downtime, and mapping convergence time. We measured the results with the two possible modes for inter-xTR mapping update, using SMR signaling and CP signaling.

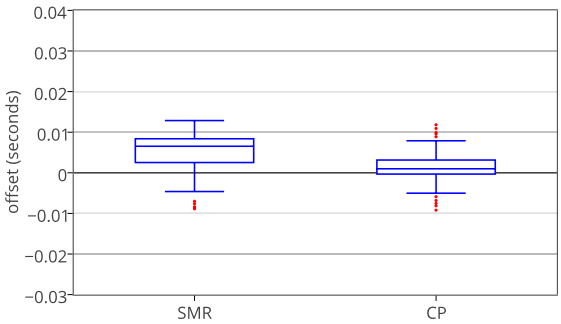
[2]During the stop-and-copy phase the VM stops on the source container in order to transfer the rest of the memory pages, those that have been recently used by the VM (also called dirty pages) [33].



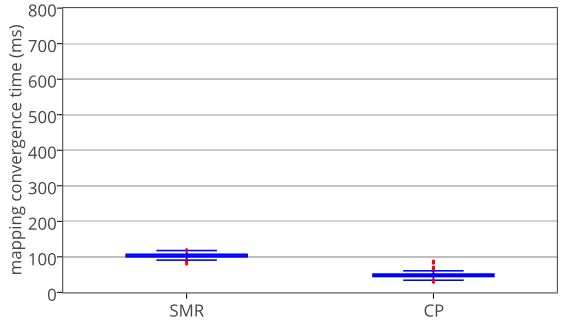
(a) Number of lost packets.



(b) Downtime.

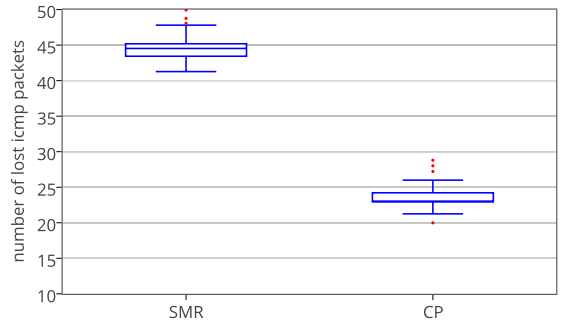


(c) Offset with the NTP server.

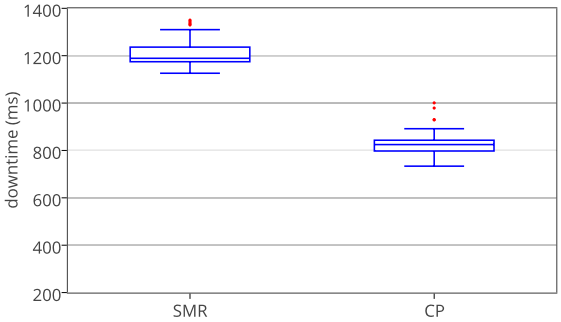


(d) Mapping convergence time.

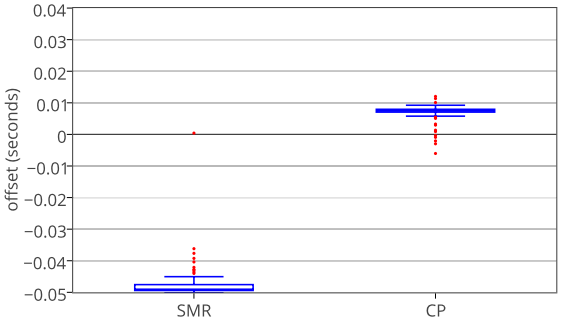
Figure 4.6: INRIA client result parameters (boxplots statistics).



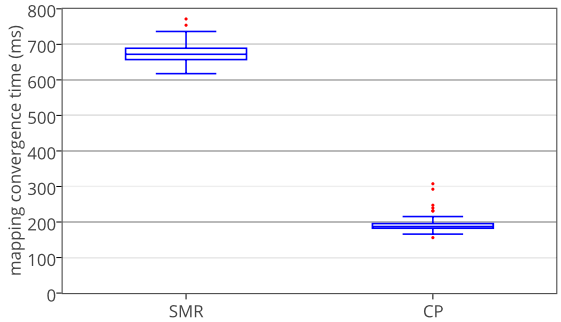
(a) Number of lost packets.



(b) Downtime.

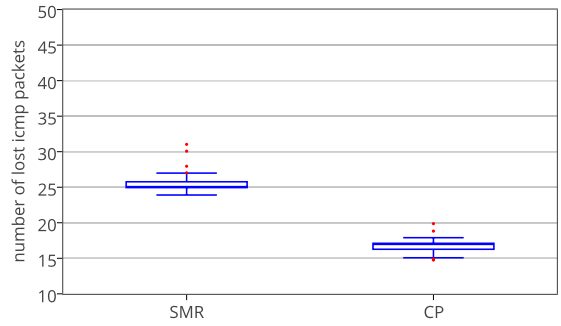


(c) Offset with the NTP server.

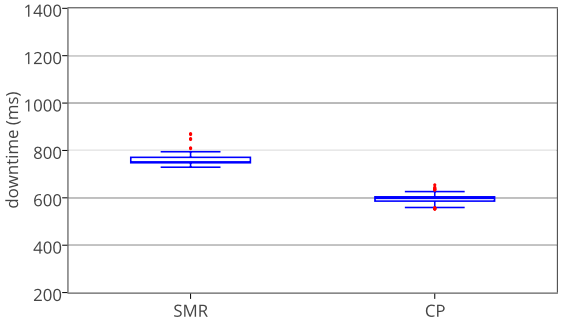


(d) Mapping convergence time.

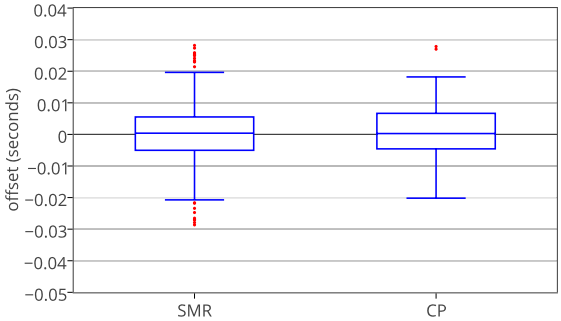
Figure 4.7: VNU client result parameters (boxplots statistics).



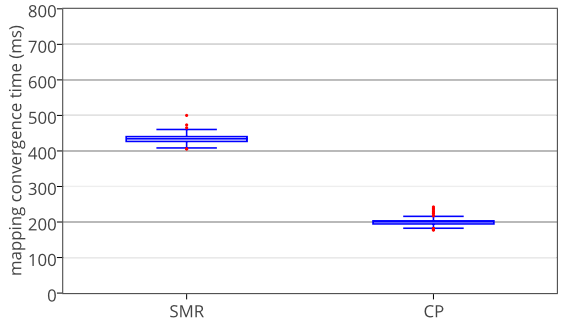
(a) Number of lost packets.



(b) Downtime.



(c) Offset with the NTP server.



(d) Mapping convergence time.

Figure 4.8: UFRJ client result parameters (boxplots statistics).

4.3.1 Using SMR signaling

As explained in Section 4.2.2, as of LISP standard control plane, the SMR message is sent by a xTR to another to solicit mapping update for a given EID. Upon reception of a SMR, the target xTR sends a Map-Request to the Map Resolver that forwards it as an encapsulated message down to the source xTR (if Map-Reply proxy is not enabled as by default In Cisco routers and as is in our simulations), followed by a Map-Repply. The overall SMR signaling time should therefore be lower bounded by one and a half the RTT between the two xTRs, which impacts the mapping convergence time and hence the service downtime. As of our experimentation, we obtained a median downtime of about 320 ms for INRIA client (Figure 4.6b), 1.2 s for VNU (Figure 4.7b) and 600 ms for UFRJ (Figure 4.8b). This large gap between the downtime of close and distant clients can be explained not only by the distance that separates each client from the VM, impacting the propagation delay (see Table 4.1), but also by the fact that the Map Resolver is closer to INRIA than to VNU and UFRJ as mentioned in Section 4.3. We find this gap also in the number of lost ICMP packets, two to three times higher for distant clients than for close ones (Figure 4.6a, Figure 4.7a and Figure 4.8a).

4.3.2 Using CP signaling

As explained in Section 4.2.2, using CP signaling the mapping convergence time can be decreased of at least one RTT between xTRs, with an authenticated one-way message that directly updates xTR cache upon reception. For the INRIA client, we obtain a median downtime of 260 ms gaining a few dozens of ms, whereas we could gain 200 ms for VNU and 400 ms for UFRJ. Moreover, we notice that the number of lost ICMP packets for distant clients has exponentially decreased. This is due to the fact that xTRs have no longer to pass via the Map Resolver to update their mapping cache. Finally, Figures 4.6d,4.7d, and 4.8d show that the mapping convergence time component of the downtime decreases with CP signaling for all cases. While it is roughly between one-third and one-half the downtime with SMR signaling, it falls to between one-sixth and one-third with CP signaling, and this ratio is higher for distant clients. This implies that the hypervisor downtime (stop-and-copy phase) is less sensible to the RTT than the mapping convergence is (likely, the last page transfer profits from an already established TCP connection with an already performed three-way handshake).

It is worth noticing that these measurements may suffer from a small error due to clock synchronization. As mentioned, the xTRs have synchronized clocks over the same NTP stratum. The median of the offsets is represented in Figures 4.6c,4.7c, and 4.8c. While it is negligible for close clients, it is of a few dozens of ms for distant

clients, however less of 5% of the downtime.

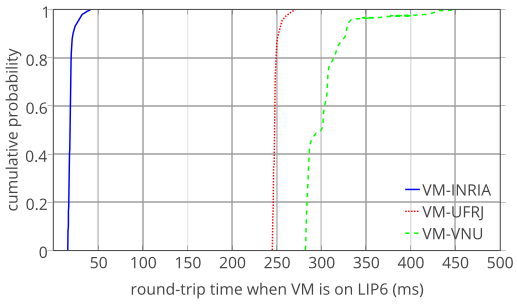
A more precise insight on the simulation parameters is given by the cumulative distribution functions (CDFs) in Figure 4.9. There were no relevant differences in inter-container bandwidth during migrations, just a slightly lower bandwidth for the CP case. The RTT CDFs show us that, from an Internet path perspective, VNU appears as more distant than UFRJ from the VMs, with a similar RTT gap before and after the migration. With respect to the downtime, the relative gap between VNU and UFRJ clients with CP and SMR is similar. In terms of mapping convergence time, the VNU-UFRJ gap changes significantly, the reason is likely due to the amplification of the RTT impact on the convergence time with SMR.

4.3.3 Comparison with alternative solutions

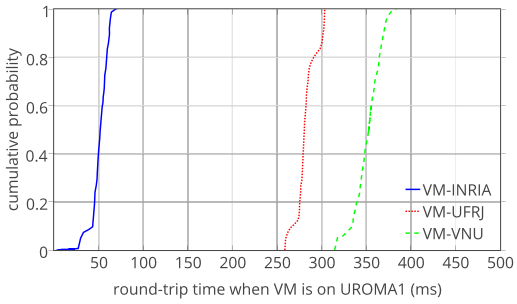
To summarize, our approach offers quite interesting performance, and we show the advantage of extending the signaling logic to clients' xTRs. Our solution offers a median mapping convergence overhead that varies from 50 ms for nearby clients (within a few hundreds km) to 200 ms for distant client (at many thousands km), depending on the signaling scope. With respect to the alternative methods which are fully handled at the hypervisor level, we assess that:

- with HyperMIP [47], authors experienced a 100 to 400 ms of overhead, which is almost two times more than our approach, the main reasons being the usage of Mobile IP and triangular routing;
- similarly in [46] Mobile IPv6 signaling is used to detect VM location change, reaching a minimum overhead around 2500 ms, linearly increasing with the network delay, hence largely higher than our approach;
- authors in [37] went a step further implementing pro-active circuit provisioning, reaching an application downtime varying between 800 and 1600 ms, which is more than 4 times higher than with our approach.

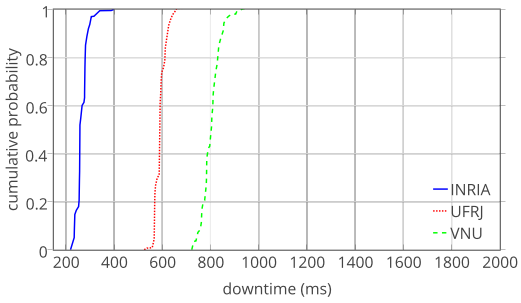
As already mentioned in Section 2.5.2, more recent hypervisor-level Layer 2 over Layer 3 overlay solutions can coexist with our LISP-based VM mobility solution. Alone, they could handle traffic rerouting upon migration, yet, for Internet communications, they would keep depending on the ingress BGP path hence offering triangular routing latency to Internet users. With respect to our solution, legacy triangular routing solutions alone add the source DC - destination DC latency to application connections, hence discouraging long-distance VM migrations. Their integration with our solution can therefore bring advantages in terms of forwarding latency hence transfer time.



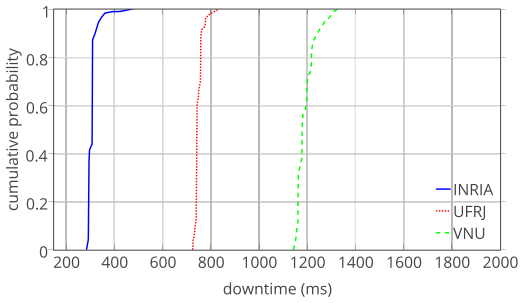
(a) RTT - LIP6



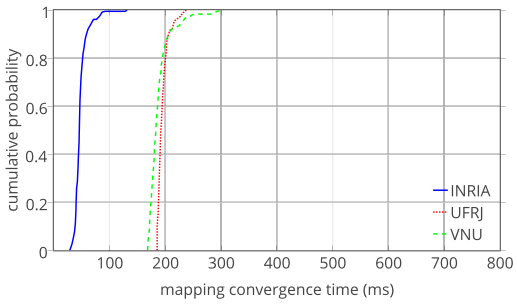
(b) RTT - UROMA1



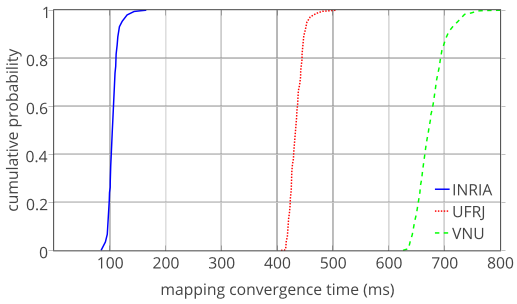
(c) CP approach



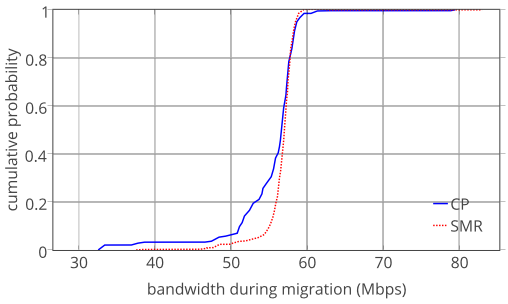
(d) SMR approach



(e) CP approach



(f) SMR approach



(g) Bandwidth

Figure 4.9: Cumulative Distribution Functions of different migration parameters.

4.4 Summary

In this Chapter, we proposed a novel LISP-based solution for IP/EID migration across distant data centers over the Internet. We tested it via the global LISP test bed. We summarize our major contributions as follows:

- we defined and implemented a new type of LISP control plane message to update EID location upon migration, with the interaction between hypervisors and LISP routers[3];
- we performed hundreds of Internet-wide migrations between LISP sites (LIP6 - Paris, UROMA1 - Rome) via the LISP test bed, including the case of clients close to source and destination containers (INRIA - Sophia Antipolis), and the case of distant clients (VNU - Hanoi, UFRJ - Rio de Janeiro);
- by exhaustive statistical analysis on measured relevant parameters and analytical discussions, we characterized the relationship between the service downtime, the mapping convergence time and the RTT;
- we showed that with our we can easily reach sub-second downtimes upon Internet-wide migration, even for very distant clients.

[3]The LISP control plane code with related functionalities is publicly available in [50]. Part of the CP signaling logic was implemented into libvirt.

Chapter 5

Adaptive Virtual Machine Mobility Management

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 Chapter, we specify a cloud access overlay protocol architecture to improve the cloud access performance in distributed DC cloud fabrics. We explored how linking 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 migrate VMs across DC sites, as a function of user-DC overlay network states. We evaluated our solution (i) on a real distributed DC test bed 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.

5.1 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 [53]. Cloud providers rely 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 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 [110] 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 ([60, 85]) 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 [58].

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 [86]. 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 better user experience. Voice recognition and augmented reality constitute the rising star of this industry; for instance, the introduction of Google Voice Search [87] and Apple Siri [88] on mobile phones and wearable smart devices, is revolutionizing the way mobile devices interact with the cloud. This type of service requires a cloud network infrastructure that can handle large amount of data on the go, with minimum disruption or quality loss offered to the user.

In this Chapter, we focus on giving mobile and sedentary users a more efficient way to access their cloud applications in a distributed DC environment making use of our controller solution, named Protocol Architecture for Cloud Access Optimization (PACAO), based on LISP protocol. The goal is to satisfy user's needs by improving

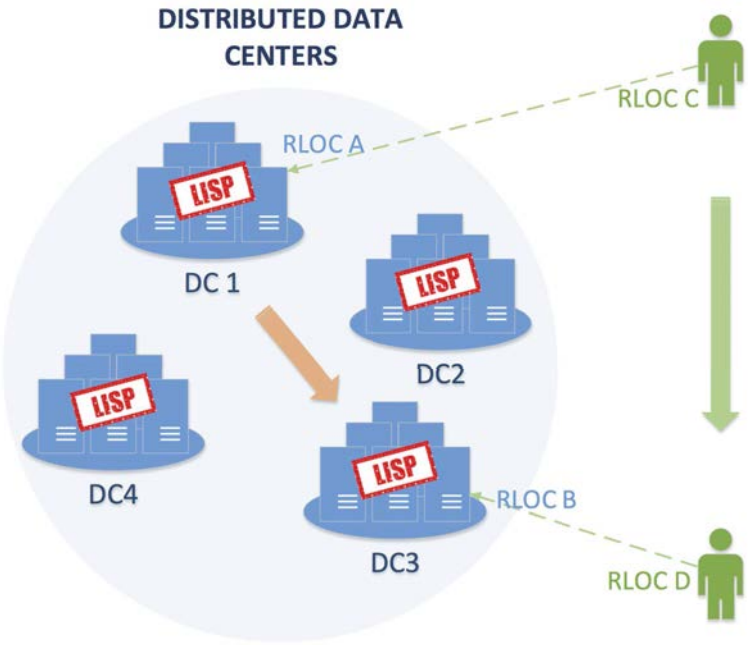


Figure 5.1: Mobile cloud access reference scenario.

the Cloud access network QoS as a function of user-cloud link quality by switching the entry DC in the distributed cloud fabric, and migrating VMs at a cloud facility closer to its user.

The Chapter is organized as follows. Section 5.2 describes the PACAO architecture. Section 5.3 presents experimental results. Finally, section 5.5 summarizes the Chapter.

5.2 Mobile Cloud Protocol Architecture

A reference example scenario is the one represented in Figure 5.1: 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 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 migrated to a closer 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

migration). PACAO relies on an adaptation of 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.

5.2.1 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.

- *IP Addressing continuity*: it is important to maintain user's session when user changes its attachment point (or RLOC). Keeping a session alive when changing user IP address without changing the application 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 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 mobile devices [81]).

- *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 across multiple sites.

- *Cloud access link monitoring*: in order to support the decision-making related to adaptive 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 [102], or other standard development organizations, among the most promising implemented virtual network overlay protocols quickly reviewed in [110], the single ones supporting addressing continuity, capability to change the location of users and VMs, and capability to monitor the routing link between user and VM locators with a

distributed yet logically centralized control-plane, are the LISP and the Virtual eXtensible LAN (VXLAN) [103] protocols, the latter performing Ethernet over UDP-IP encapsulation. The PACAO prototype makes use of both VXLAN and LISP, proposing LISP as Cloud access overlay protocol and VXLAN as inter-DC overlay protocol[1].

In our reference distributed DC fabric, each DC is at least one 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 DC 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 one as it has been shown in the previous Chapter. It is assumed that a VM is also reachable through the RLOCs of the other DCs. For example, in Figure 5.1 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 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, as illustrated in Figure 5.1, when a mobile user with RLOC C moves to 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 RTT, 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 migrating the VM to another DC of the Cloud fabric.

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

[1]Using VXLAN as Cloud access protocol would generate higher signaling due to its multicast operations, while using LISP as inter-DC overlay protocol would require additional variations to it to synchronize VM location states among LISP routers, as far as multicast extensions to LISP are not defined and implemented.

[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 [59].

5.2.2 PACAO controller

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

- *Monitoring Module*: it monitors the connection between the user and the VM, dividing the user-VM link into two links: user-xTR links and xTR-VM links. To monitor the user-xTR link, active online probes periodically collect QoS metrics between VM and user 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 [54], 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 parameter. Accordingly, we implemented RLOC probing in OpenLISP [106, 107] to allow transporting RTT and jitter metrics. 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 migration online optimization problem formulated in section 5.2.3. 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 RLOC to a locator set. The latter contains all 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 lets the algorithm computes the best location for the VM.

- *Decision Module*: based on the optimization solution computed by the optimization module, 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 the VM to another DC.

Each of these modules can logically be implemented on separate nodes or on the same one; in the former case, 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

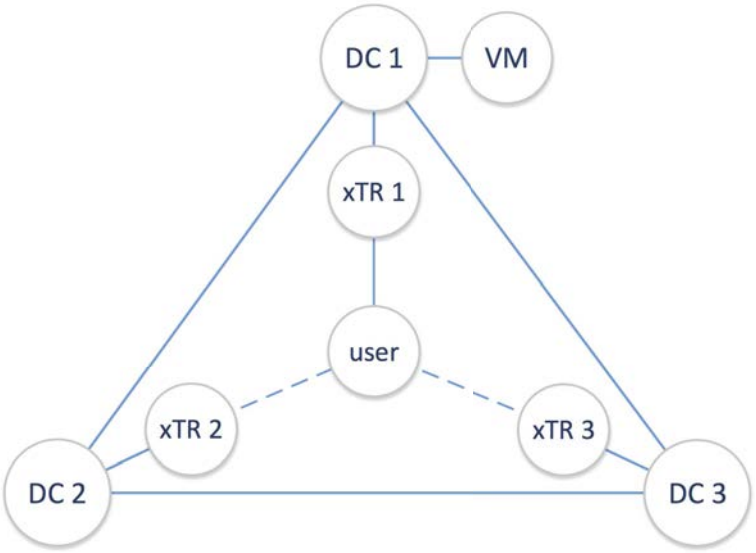


Figure 5.2: Distributed DC reference scenario with functional nodes.

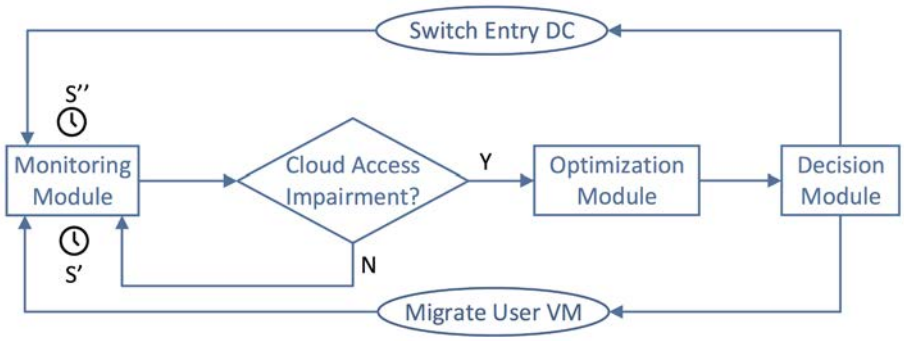


Figure 5.3: PACAO policy execution chart.

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 (SDN) protocols.

5.2.3 Cloud Access Optimization

Figure 5.2 depicts an example for the network model. 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 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. 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 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. 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 a linear programming 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:

$$\sum_{k \in K} \alpha_k T^k \tag{5.1}$$

where K indicates the network QoS or system performance criterion: $k = 1$ for 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 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:

$$\sum_{d \in D} r_d = 1 \tag{5.2}$$

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 DC 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:

$$m_d^k r_d \leq M^k T^k \tag{5.3}$$

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 infeasibility, as the maximum tolerated threshold is rescaled by T^k. When the problem is meant to represent the decision problem about migrating 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 full-mesh and single-hop nature of the cloud access overlay graph, the problem defined by (5.1)-(5.3) does not contains flow conservation constraint, and with a single bin to pack it has a polynomial complexity and can be easily solved online.

5.2.4 Online Scheduling Procedure

The above presented optimization can be triggered by the monitoring module at each arbitrary duration S' (Figure 5.3) to check the supervised links. At the end of a time duration S', 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 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 (5.1)-(5.3) 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 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. (5.1)-(5.3) as a VM migration optimization problem.

It should be clear that PACAO architecture addresses both the case when a mobile user roams onto a new location 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). In order to minimize signaling and facilitate data exchange. We implement the monitoring and decision modules described previously in the xTR.

Example

Figure 5.4 illustrates the steps of interaction between the different elements of the PACAO architecture.

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

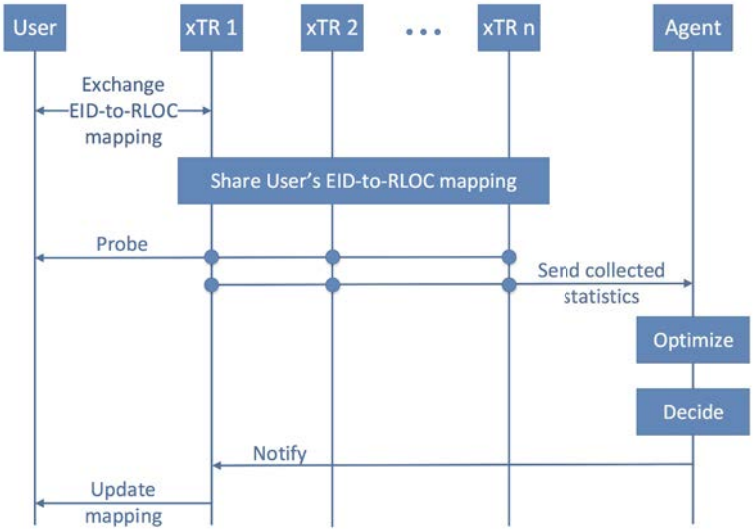


Figure 5.4: PACAO sequential functional steps.

1. A user establishes 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 message is triggered to get the EID-to-RLOC mapping of the VM.

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 RLOCs of all DCs with different priorities and weights from where the service can be reached. 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 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 LISP data plane) controlled by an external SDN controller 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 RLOC 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 to the new hosting DC.

5.3 Test bed Experimentation Results

PACAO architecture has been implemented using the following nodes.

xTR

we used the OpenLISP development version [106] as xTR software routers, to which we added 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 xTR locators and all the other locators in the mapping cache for each EID entry. The statistics are then saved in a log file (JSON format) 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 log file above to get the user-to-xTR and xTR-to-VM QoS metrics. We used the GNU Linear Programming Kit [82] to solve the optimization problem described in section 5.2.3. 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 [99], and update the user's mapping cache; to migrate a VM we use the VIRSH command interface provided by libvirt API [120] 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 test bed is represented in Figure 5.5. We joined the Nu@ge [80] and the LISP-LAB [75] experimental networks, with three sites in 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. Using VXLAN [103], 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. In order to perform seamless VM migrations, we connected them to the same network using an OpenVPN server (disabling authentication and encryption features to reduce the VPN overhead and accelerate the communication). VXLAN,

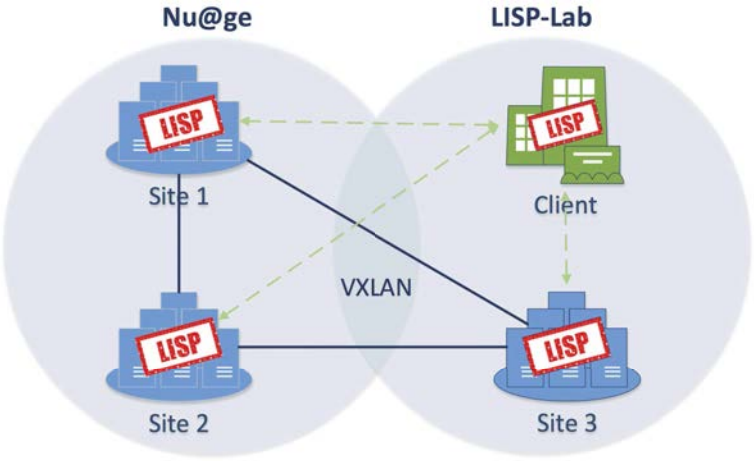


Figure 5.5: testbed network.

running over the VPN, enables internal traffic redirection. The service VM is an Ubuntu 14.04 server that uses a mounted disk on a network file system.

All the inter-DC links use the plain Internet, except the Marilyn-TelcoCenter one that is a 10 Gbps Ethernet over fiber 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 DCs. We also created an FTP server on the IaaS in order to measure the user throughput.

5.3.1 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 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:

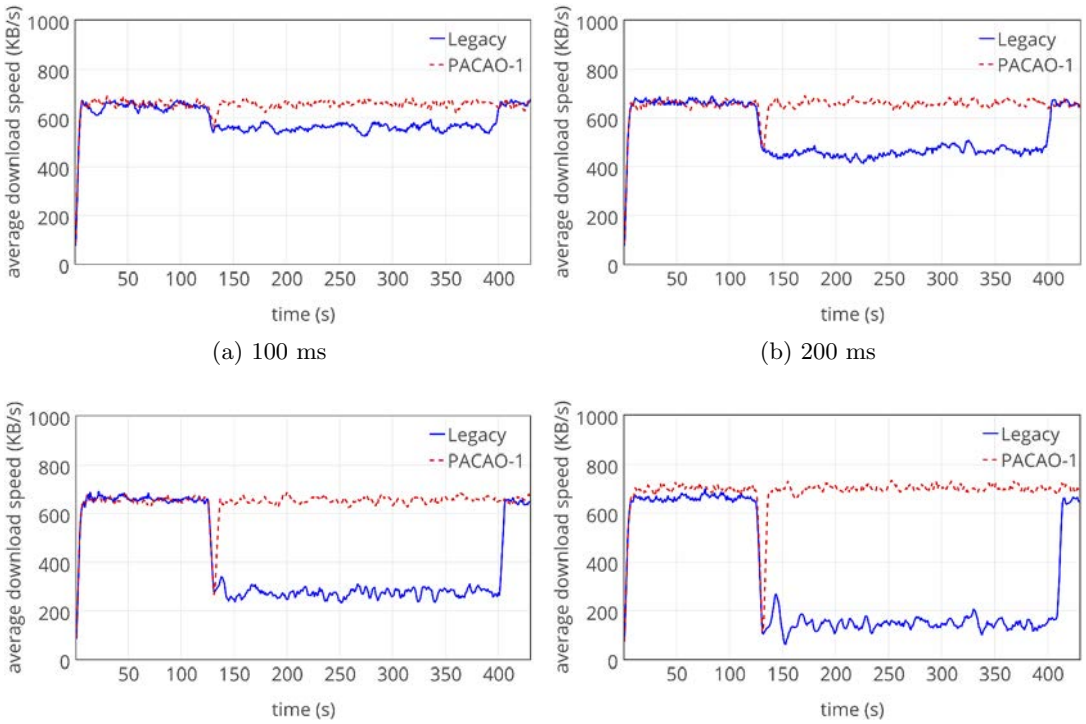
- Legacy: basic configuration with fixed RLOC.
- PACAO-1: our solution limited to RLOC switching optimization feature (VM migration disabled).

The scenario is summarized in Table 5.1:

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);

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 5.1: First experimentation scenario time line.



(a) 100 ms

(b) 200 ms

(c) 400 ms

(d) 1000 ms

Figure 5.6: Average download speed (scenario of Table 5.1).

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 Figure 5.6, we report the average measured bandwidth for four different emulated network

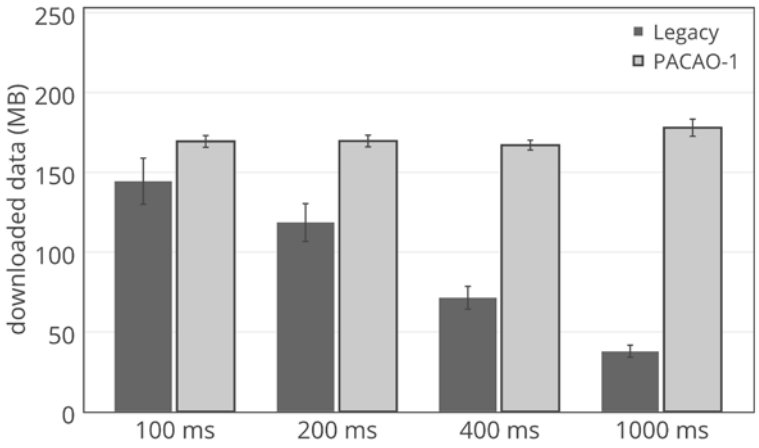


Figure 5.7: Total amount of data downloaded in 450 s

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 case the bandwidth decreases drastically at 125 s as one would expect. 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 [99].

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 Figure 5.7 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%.

5.3.2 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 5.2: Second experimentation scenario time line.

We then apply to each case the steps in Table 5.2:

1. we start IPERF from both user and 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 $S' = 10\ s$ 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.

As depicted in Figure 5.8 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 Figure 5.9, shows we can gain up to 40% by switching the RLOC then migrating the VM in the use case described above.

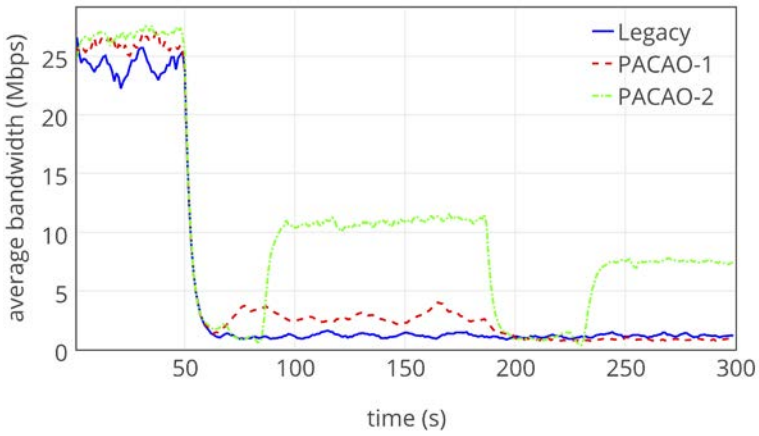


Figure 5.8: Average bandwidth given the scenario of Table 5.2.

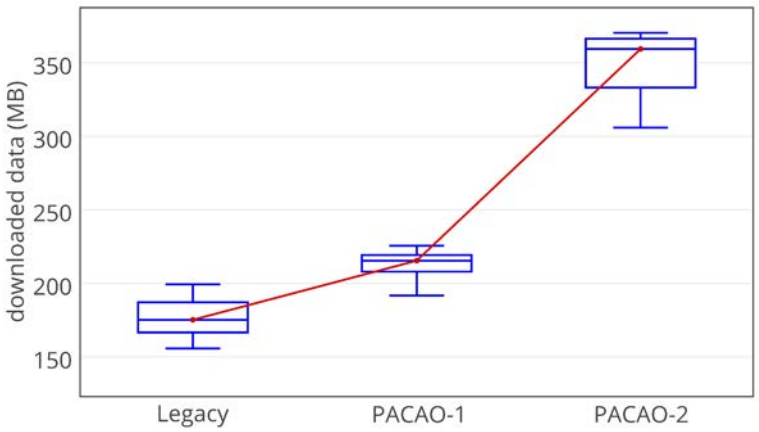


Figure 5.9: Total amount of data downloaded in 300 s.

One concern is whether the proposed PACAO adaptive VM migration is suitable for real-time services, sensible to packet loss and jitter.

Figure 5.10 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 Figure 5.11, which shows no noticeable difference between the two PACAO cases that offer more stable performance (lower dispersion around the median) than the legacy solution.

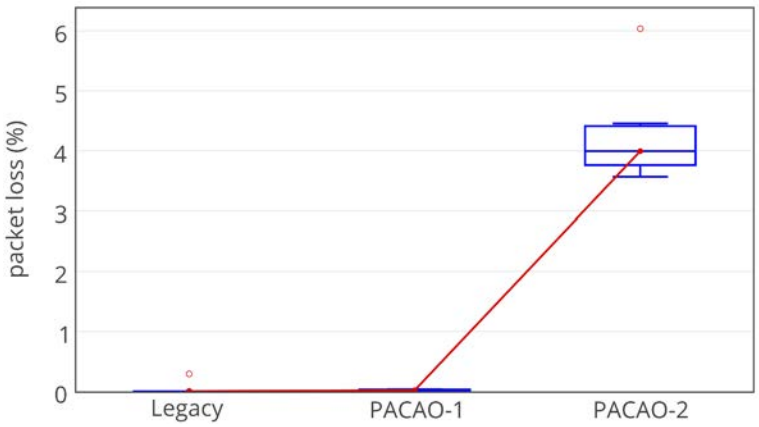


Figure 5.10: Percentage of lost packets.

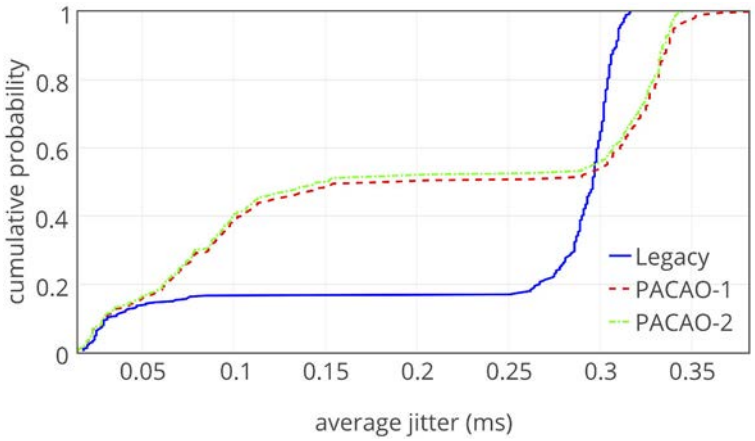


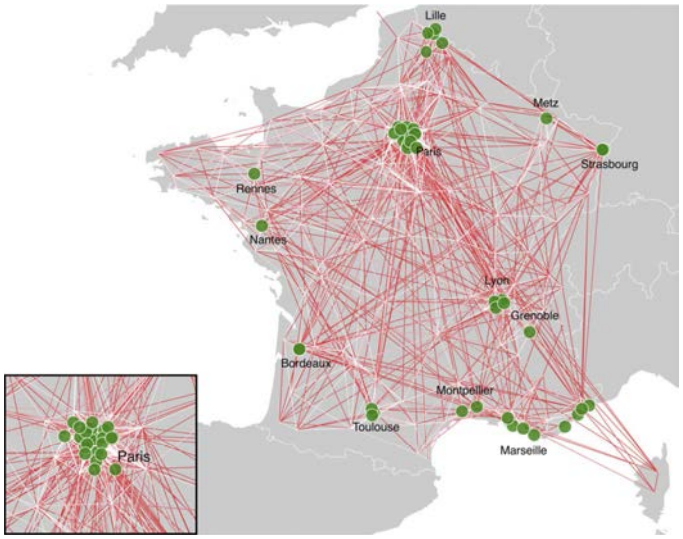
Figure 5.11: CDF of the average jitter.

5.4 Evaluation using Real Mobility Traces

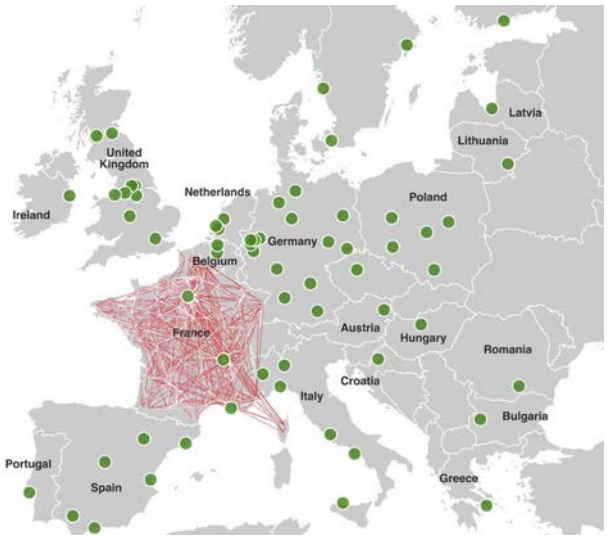
To better understand how our framework would perform in real deployments, we extracted real cellular user's large-scale mobility traces and evaluate the impact of PACAO with different DC geographical distributions. Before detailing the emulation environment, we first provide a description of the data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

5.4.1 User mobility traces and data center distribution

Via a French collaborative research project, we could access large-scale 4G user mobility traces from 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 [55]). The data comes



(a)

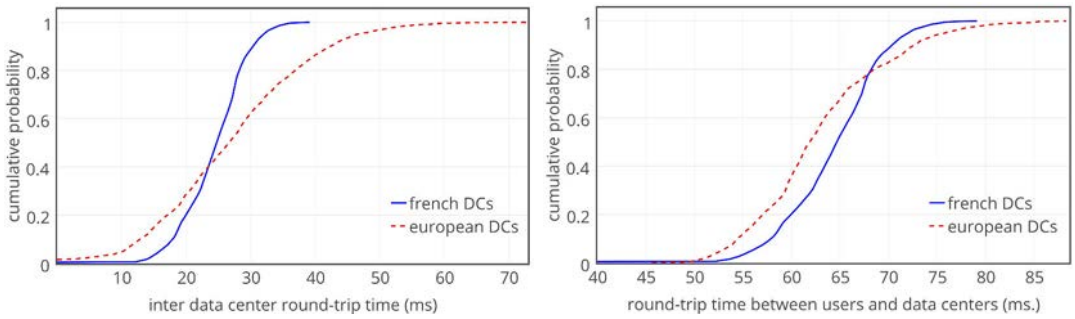


(b)

Figure 5.12: Representation of the employed trajectories and data-center distribution.

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 user density - this is practically implemented as a uniform distribution over the LACs, LACs having a geographical coverage that is inversely



(a) Inter DC links (b) Users to DC links

Figure 5.13: Emulated round-trip-times between network nodes.

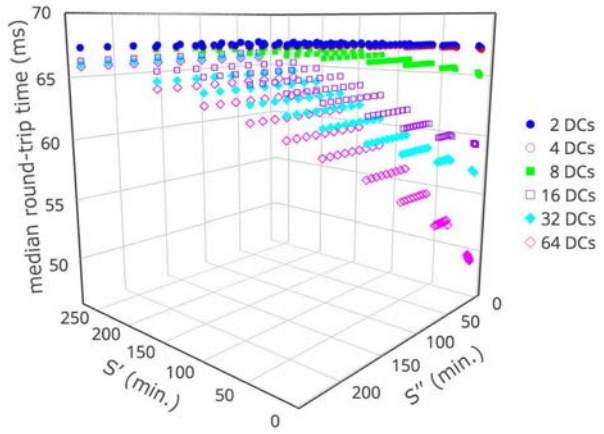
proportional to user density. Each LAC being the possible location of a DC, with no more than one DC per LAC. In the following simulations, we vary the DC distribution from 1 to 64 over French and European territories. The France-wide DC distribution can be considered equivalent to a deployment of the 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, the two use-cases are hereafter referred to 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 today's 4G networks.

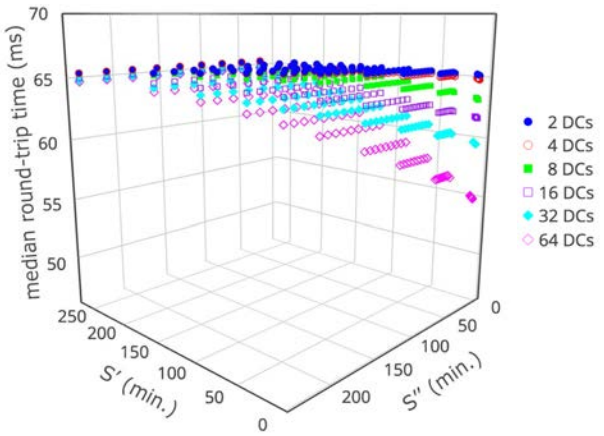
Figure 5.12 represents the 64 DC locations as well as the mobility traces. Figure 5.13 reports the resulting distribution of used RTTs, between user and DCs and between DCs, for the France/ISP and Europe/OTT cases.

5.4.2 Simulation Results

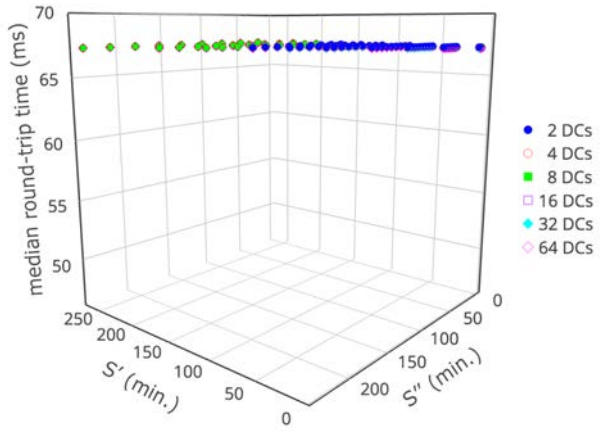
Based on the collected data set, we have written a simulator that mimics PACAO functionality and estimates the RTT between users' geographical position and the DCs. The simulator is written in Python and uses CPLEX to run the optimization module.



(a) Strict latency bound (58 ms)

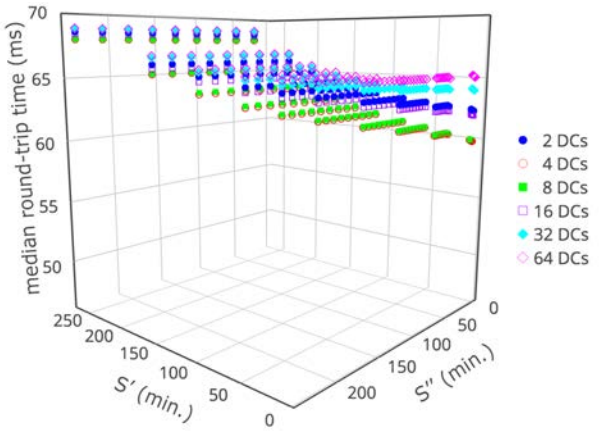


(b) Median latency bound (66 ms)

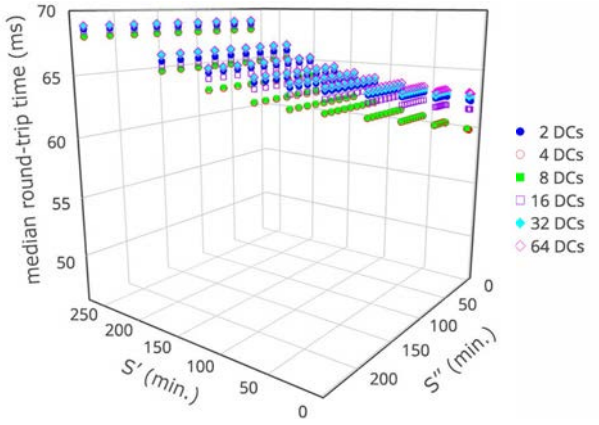


(c) Loose latency bound (70 ms)

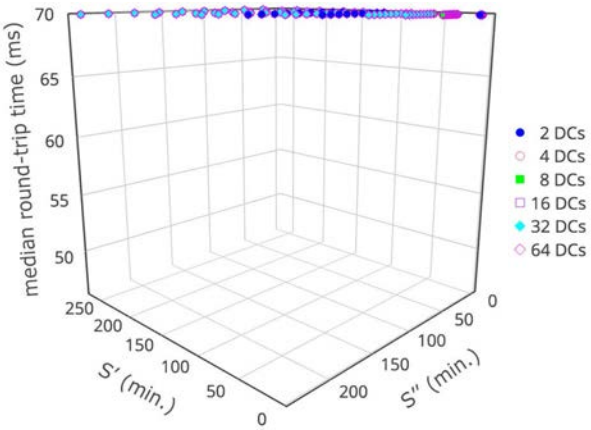
Figure 5.14: France/ISP: round-trip time results as a function of PACAO S' and S'' time-slots values.



(a) Strict latency bound (54 ms)

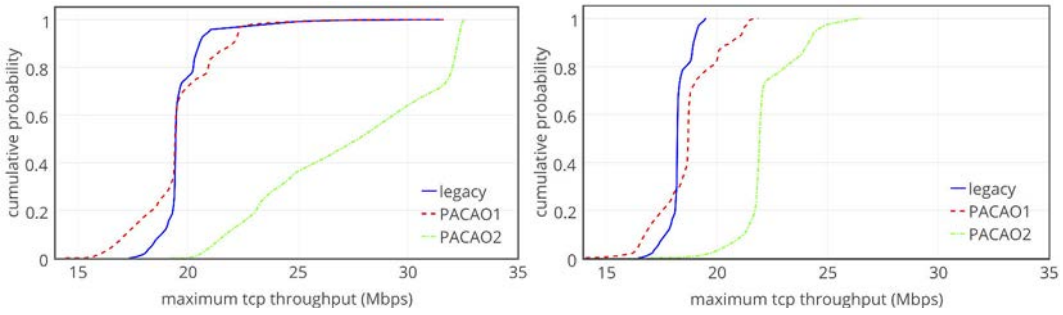


(b) Median latency bound (62 ms)



(c) Loose latency bound (72 ms)

Figure 5.15: Europe/OTT: round-trip time results as a function of PACAO S' and S'' time-slots values.



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

Figure 5.16: Maximum estimated throughput (Mbps).

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 (5.3), 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 Figure 5.13. We run it for each of the following DC distributions: 2, 4, 8, 16, 32, 64 DCs. In Figure 5.14, 5.15, we plot the offered median RTT as a function of S' and S'' (see sect. 5.2.4), 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 $[0.1S', 0.9S']$.

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 of 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 (Figure 5.13b) 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 (Figure 5.12a), 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, the best performance is reached with 64 DCs for the France/ISP case and with 4 DCs for the Europe/OTT case.

Figure 5.16 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 [104], for 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 PACAO-1 step from the PACAO-2 step to better show the impact of VM migration. Besides switching the RLOC (PACAO-1, 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.

5.4.3 Dealing with Cloud Access Congestion

Our cloud access optimization formulation proposed in section 5.2.3 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 affected to a same DC. For such traffic congestion situation, link latency can be expressed by a piece-wise function [56], as depicted in Figure 5.17. We provide in the following a reformulation of the optimization problem to deal with cloud access congestion.

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 5.3.

$$min\ T = \sum_{u \in U} T_u \tag{5.4}$$

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:

$$\sum_{d \in D} r_u^d = 1 \quad \forall\ u \in U \tag{5.5}$$

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

$$\sum_{u \in U} r_u^d + e^d \leq C^d \quad \forall\ d \in D \tag{5.6}$$

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:

$$l^d \geq f_i^d \left(\frac{1}{C^d} \Big(\sum_{u \in U} r_u^d + e^d \Big) \right) - L_{max}(1 - \sigma_i) \quad \forall i \in I,\ \forall d \in D \tag{5.7}$$

$$\sum_{i \in I} \sigma_i = 1 \tag{5.8}$$

where l^d is the additional latency on a DC d. L_{max} is a big-M parameter. I is the set of piece-wise steps. The last constraint is to push and meet the latency SLA:

$$l^d + l_u^d \leq LT_u \quad \forall\ d \in D, u \in U \tag{5.9}$$

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.

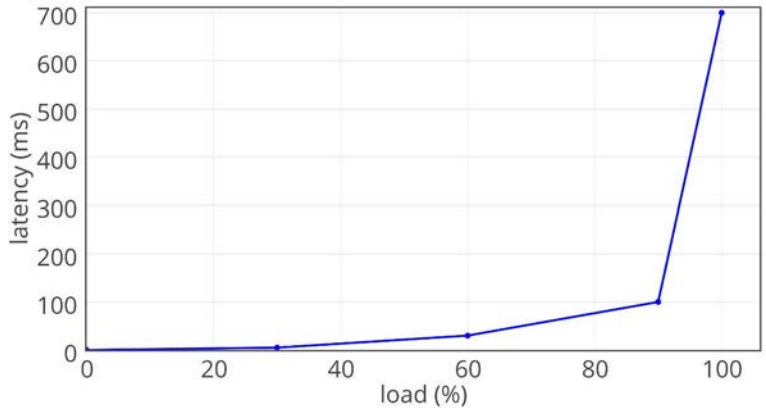
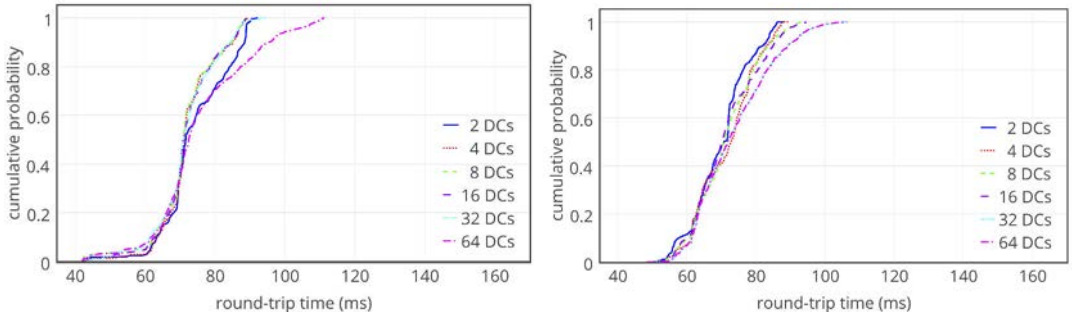


Figure 5.17: Piece-wise latency function of the link load.



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

Figure 5.18: RTT distribution under congestion-aware optimization for different DC distributions.

Simulation Results

In Figure 5.18 we report the CDF of the RTT under (5.4)-(5.9), 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 Figure 5.17. We set the congestion latency component scale as of Figure 5.17 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 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 Figure 5.18 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.

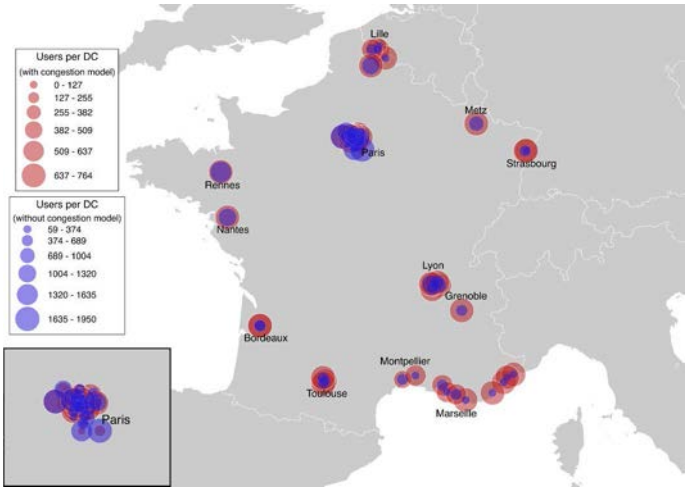
Parameters	
C^d	maximum capacity for DC $d \in D$
L	maximum allowed round-trip time
$f_i^d(x)$	i-th latency function on DC $d \in D$ depending on its access link load x
e^d	number of existing users on DC $d \in D$
l_u^d	latency between user $u \in U$ and DC $d \in D$
Binary variables	
r_u^d	1 if user $u \in U$ 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 5.3: New variables and parameters.

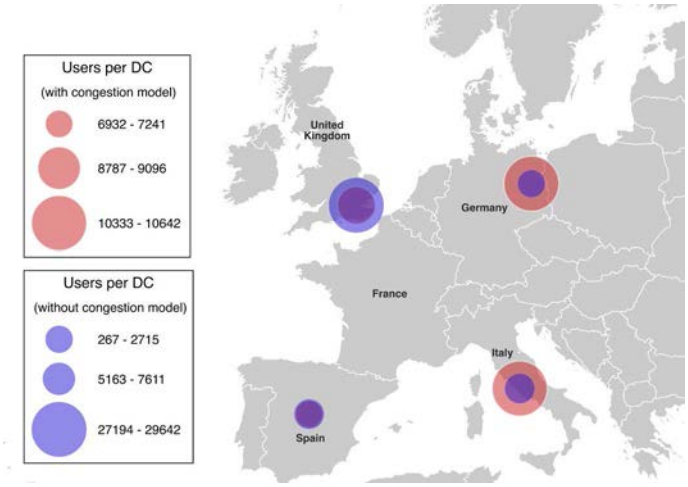
To give an idea on the obtained result, we depict in Figure 5.19 how users get dispatched on the distributed fabric, for the 64 DC France/ISP and 4 DC Europe/OTT cases, as compared to the solution without congestion awareness. The load is better balanced over all the available DCs under congestion awareness. In the Europe/OTT case, this leads to higher load on the eastern DCs, as the western DCs were capturing most of the traffic.

5.5 Summary

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 migrate 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 operations



(a) France/ISP



(b) Europe/OTT

Figure 5.19: Distribution of users over DCs with and without congestion-aware scheduling.

transparently with respect to the underlying network infrastructure.

In this Chapter, 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 migrations 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 test bed including four distant data-centers in France, and by simulation using large-scale mobility traces from a major mobile Internet service provider.

The test bed 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 downside can be represented by a non negligible packet loss, at some extent critical for UDP-based service, due to virtual machine migration, which can be however minimized using advanced commercial techniques than those we could use for testing.

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. The simulation results confirm the test bed results about the importance of linking virtual machine mobility to user mobility, besides simply switching the data-center routing locator.

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 test bed 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. In the next chapter, 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.

Chapter 6

Cloud Access Acceleration with Multipath Transport

The explosion of cloud-based applications is leading to a much higher demand on both computing and network infrastructure resources. Enhancing the user's experience, by reducing the latency, increasing network stability and throughput, using Layer-4 and Layer-2 acceleration protocols becomes an important challenge for cloud operators. In this Chapter we show how MPTCP access can be integrated to our reference architecture in order to load balance the user's traffic efficiently by increasing its throughput. For situations when at least one end-user is multihomed, we propose to enhance its subflow mechanism so that MPTCP creates an adequate number of subflows considering the underlying path diversity offered by LISP. We defined and implemented a cross-layer cooperation module between MPTCP and LISP and we evaluated it with a realistic cloud access use-case scenario involving one multihomed data center.

6.1 Introduction

In Internet-working research, multipath communication capabilities are becoming an increasingly popular subject as Wide Area Network (WAN)-wide multipath protocol features and implementations become more and more tangible. Indeed, to reap the benefit of multipath forwarding - that is in our perspective to gain in resilience and throughput - one need at some point is to have different data flows following physically disjoint paths.

From a physical network perspective, the competition between telecommunication operators resulted in the construction of parallel physical networks till the home, be it wired (xDSL, optical fiber links) or wireless (3G, 4G, etc...). At the same time, enduser laptops and cellphones got equipped with concurrent interfaces such as WiFi, Bluetooth, 4G. On the server side, data centers tend to be connected to the Internet through several ISPs to improve their resiliency, i.e. data centers are more often multihomed. In the end, networks may well provide physically disjoint paths between the servers and its endusers, both ends still need to be able to use these different paths concurrently to improve the performance. The exploitation by the transport layer of this path diversity available end-to-end at the IP layer is actually not easily attainable. What is left to use these paths simultaneously is an adequate protocol stack, flexible and compatible enough with the existing environment, able to stitch transport-layer multipath capabilities to different IP-layer paths.

A significant amount of work has gone into devising the SCTP [138]. Its later equivalent Multipath TCP [139] is more retro-compatible and hence is more likely to be largely deployed than SCTP. Indeed MPTCP is compatible with TCP, both at the header and at the congestion mechanism levels:

- if one of the endhost is not MPTCP compliant or if there are middleboxes preventing the use of MPTCP, then the TCP connection falls back to legacy TCP;
- an MPTCP connection can coexist with other TCP connections, i.e., the MPTCP connection does not get more capacity than the TCP connections.

The congestion mechanism objectives set in the specifications also states that MPTCP must get at least as much capacity its TCP equivalent would get. Thus, the congestion mechanism and the subflow management system appear tightly interlaced; when to create new subflows or to delete them becomes a critical question. The MPTCP stack has a poor knowledge of the underlying network to adopt efficient decisions, but lower layers may share useful knowledge to support MPTCP for an efficient subflow selection. For instance, creating several subflows for a data

center communication with a Layer-2 bottleneck, e.g., due to a shared spanning tree segment, might prove counterproductive in terms of processing overhead vs. increase in throughput; similarly, an Internet cloud-access multipath communication with a server in a data center through a single external path (single provider) could offer no performance gain.

The purpose of this study is to combine IP path diversity to different MPTCP servers and users in a cloud network where at least one side, path diversity is available at the IP layer, but not usable with native protocols. More precisely, we propose a specific cross-layer signaling to allow a MPTCP endpoint to profit from path diversity offered by a LISP network. LISP can give hints to MPTCP about the Wide Area Network (WAN) paths diversity between two MPTCP endpoints. By allowing sharing of such an information among endpoints, we influence the number of subflows to create.

6.2 Supporting Transport Layer Multipath Connections

After formulating a cloud access optimization under traffic congestion situations in Chapter 5, an open question that we want to address is whether the usage of Layer-4 load balancing protocols, such as multipath transport protocols, at the mobile device or access point level can be beneficial to the user under this setting. Therefore, if it is possible to combine multipath TCP with a cloud access overlay protocol, such as LISP, via multiple routing locators in a distributed DC situation, what could the impact be on the end-user's experience?

To take the multipath transmission feature into consideration, the formulation (5.4)-(5.9) in Chapter 5/section 5.4.3 can be slightly modified, changing the r_u^d from a binary variable to a non-negative continuous variable subject to the following two additional constraints:

$$\sum_{d \in D} r_u^d = 1 \quad \forall\ u \in U \tag{6.1}$$

$$0 \leq r_u^d \leq 1 \quad \forall\ u \in U, \forall d \in D \tag{6.2}$$

6.1 can also be relaxed to allow throughput increase when the optimization objective allows r_u^d to take values higher than 1.

6.3 MPTCP a Cross-Layer Solution

The current MPTCP path discovery does not explicitly limit the number of subflows to create, so current implementations create by default a mesh of subflows

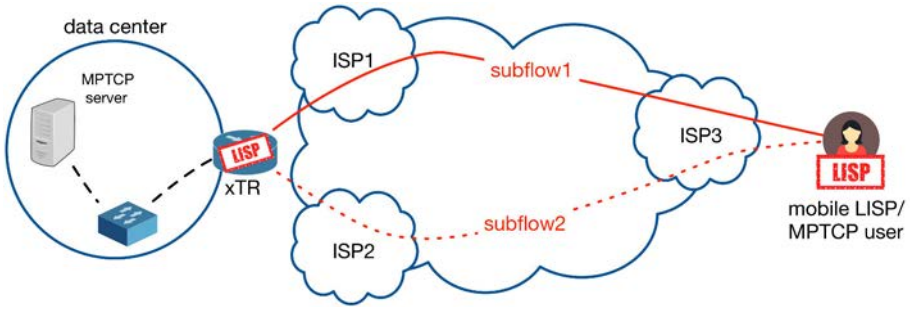


Figure 6.1: Example of a multihomed data center and associated MPTCP subflows.

between two hosts' IPs. Most of the times, the more the subflows, the higher connection throughput, under appropriate congestion control (note that there are also cases in which this default mechanism would prevent MPTCP from increasing the throughput, and cases where fewer subflows could provide the same gain by using fewer network resources). We target the specific case where more subflows could be created if intermediate multipath forwarding nodes (at the border between the local site and the WAN) can be used to split subflows over different WAN paths, under the hypothesis that the connection throughput is limited by WAN capacity rather than by Local Area Network (LAN) capacity.

In the following, we describe how the MPTCP path discovery can be augmented in this sense in a LISP network. Then we present the required signaling between MPTCP and LISP network elements, and the possible methods to perform the required multipath subflow forwarding.

6.3.1 Subflows On-Demand

The MPTCP specification – see the path management section in [139] – states that the path discovery mechanism should remain modular so that it can be changed with "no significant changes to the other functional components ". We leverage on this specific part of the MPTCP protocol to define an augmented subflow discovery module taking profit of LISP capabilities, to augment MPTCP performance while preserving endhost resources.

Figure 6.1 presents a cloud access situation where MPTCP could perform better if its path discovery module were informed of LISP path diversity and forwarding capability when creating subflows. In the example situation, there is one IP on the client host and one IP on the server; as such, the legacy MPTCP mechanism creates only one subflow. Under the assumption that commonly connection throughput is limited by WAN capacity rather than by LAN capacity, limiting to a single subflow prevents from benefiting of the WAN path diversity and the likely greater through-

put achievable if forms of multipath forwarding at intermediate nodes exist. A LISP network offers the signaling capabilities to retrieve path diversity information, and the switching mechanisms to ensure multipath forwarding. In the Figure 6.1 example, this is possible establishing two subflows instead of one, assuming each subflow is forwarded to a different IP transit path (guaranteed as explained next) thanks to the LISP-capable border router. It is worth highlighting that as of the specifications - and as implemented in the MPTCP Linux implementation [141] - different MPTCP subflows can share the same source IP provided that they use different TCP ports. This subflow identification mode should be used to create the additional LISP-enabled subflows.

More generally than the example in Figure 6.1, the number of MPTCP subflows can be significantly increased thanks to LISP capabilities in the case the endpoints dispose of multiple interfaces. We assume communications between hosts are strongly asymmetric, the largest volume being from server to client, so that the client-to-server flow essentially consists in acknowledgments. Let l_1 and l_2 be the number of interfaces of server endpoint (or of the hosting hypervisor in case of VM server) and client endpoints, respectively. A LISP site can be served by one or many LISP routers, each disposing of one or many RLOCs. Let L_1^r and L_2^r be the number of locators of the r^th LISP border router at site 1 (Server side) and 2 (Client side), respectively. Therefore, the maximum number of subflows that can be opened by legacy MPTCP is $l_1 l_2$. Following the design choice to route only one subflow over one RLOC-to-RLOC inter-site path to avoid bottlenecks and other management issue in the WAN segment, the number of maximum number of subflows that could be opened thanks to LISP multipath forwarding is $(\sum_r L_1^r)(\sum_r L_2^r)$. Then we can distinguish two cases:

1. if the LISP site network allows full reachability between endpoints and corresponding ITRs, and endpoint's traffic reaches any corresponding ITR in a non deterministic way, then the maximum number of subflows that shall be opened is:

$$N_a = max\{l_1 l_2; (\sum_r L_1^r)(\sum_r L_2^r)\} \tag{6.3}$$

LISP-based augmentation would likely be effective only if the second term is higher than the first. The non deterministic interconnection between the endpoint and its ITRs can be due, for instance, to adaptive L1/L2/L3 traffic engineering changing routes and egress points from time to time even for a same application flow, or load balancing such as equal-cost multipath L2/L3 routing so that different subflows may exit by different ITRs. It is worth highlighting that 6.3 is only a suggested upper bound; in the case the right term

is the maximum, in such non deterministic situations there is no guarantee the N_a WAN paths will be used, especially if $l_1 < \sum_r L_1^r$ or $l_2 < \sum_r L_2^r$;

2. if each of the endpoint's interfaces can have its traffic routed via one single RLOC each, then we can take a much larger profit from the LISP path diversity. The best situation is when $l_1 \geq \sum_r L_1^r$ and $l_2 \geq \sum_r L_2^r$. So, if the number of the interfaces is at least equal to the number of the RLOCs, then the desired feature is to explicitly link each interface to the right ITR. Such a setting is the de-facto setting when the user is a LISP mobile node MPTCP-capable user; for the Cloud server side, it is technically plausible to create a number of virtual interfaces and bind them to different ITRs via virtual LAN mechanisms and the like. In such configurations, the maximum number of subflows that shall be opened is:

$$N_b = (\sum_r L_1^r)(\sum_r L_2^r) \tag{6.4}$$

The situations described in the previous points can hold only at one site or even only partially within each site. 6.3 and 6.4 are indeed upper bounds. Nevertheless, it does not hurt creating a number of subflows higher than the number of paths than N_a or N_b, at the expense of a higher processing and signaling burden on the network, with a probably limited and in any case non-deterministic gain. Therefore, 6.3 or 6.4 can be used to compute the number of subflows for the MPTCP-LISP augmentation, depending on the situation, upon appropriate customization of the path discovery module settings.

6.3.2 Signaling Requirements and Implementation Aspects

From a mere signaling perspective, the requirement is therefore that the MPTCP endpoint be able to query the LISP mapping system to get the RLOCs of the other endpoint (the number of interfaces being already transported via MPTCP options). Standard LISP control-plane messages can natively assume this task (i.e., Map-Request and Map-Reply messages). Once this information is obtained, the number of subflows to be opened can be computed, and the MPTCP endpoint can either advertise these possible subflows to the other endpoint or initiate the required subflows following a procedure such as the one we propose hereafter.

Let us describe with a higher detail the different steps, depicted in Figure 6.2, needed for MPTCP to retrieve the adequate number of local and remote RLOCs, thus the adequate number of subflows to create.

1. endpoint C (Client) first establishes an MPTCP connection with endpoint S (Server) - it is worth noting that these communications are done at the kernel

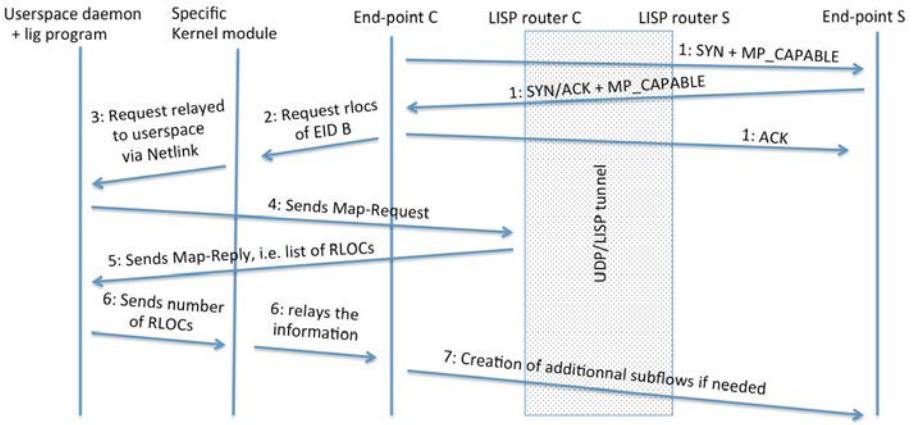


Figure 6.2: Signaling process chronology.

level. Once endpoint S is assessed as MPTCP compliant, we can start the path discovery process.

2. endpoint C calls a function of the previously loaded MPTCP kernel module, with as parameters the current MPTCP connection identifier and the EID we want to find the RLOCs associated to.

3. Upon reception, the kernel module translates it into a netlink message and sends it to the MPTCP netlink multicast group. Netlink being the networking protocol used by the linux kernel space to communicate with user space and vice-versa.

4. We have a specific daemon in user space registered with the MPTCP netlink group that upon reception of this message sends a Map-Request to our LISP router C, i.e., the daemon asks for the RLOCs responsible for the EID (an IP of Server B). Note that xTR should send a Map-Request to a Map Server, but we have modified LISPmob so that it acts as a sort of “proxy Mapping Server”to avoid artifacts related to mapping system latency.

5. This same LISP router C answers with the list of RLOCs responsible for the requested EID to that user space daemon, thus retrieving the number of RLOCs. The daemon then sends the answer via Netlink to the kernel module.

6. Upon reception of the answer, the module relays the information to the kernel, triggering the creation of new TCP subflows. Subflows source ports are chosen so that their port number modulo the number of disjoint paths are all different. The LISP router can then deterministically route each subflow to a specific path. For that mechanism to be fully effective, it should be installed on both

remote and local sites. Another strategy could be to rely on hashing functions and create enough subflows to be sure some will go through different paths. The MPTCP congestion mechanism would then send data only on the best subflows, which is likely to be subflows following different paths.

It is worth noting that the described process is expected to be efficient only for not short-lived flows. Indeed, on the one hand short flows may finish before the procedure; on the other hand, the different requests sent from the hosts to the routers consume bandwidth and add an avoidable load to the router. Caching associations between endpoint identifiers and the number of sublows to create mappings in the source host could alleviate networking traffic. To avoid applying our proposed cross-layer interaction to any flow, a system that triggers the procedure only for long-lived flows would be very reasonable, for example based on a white list or a dynamic recognition system such as described in [148].

6.3.3 LISP Load Balancing Requirements

To ensure that subflows follow different WAN paths, it is possible to implement both stateful and stateless load balancing mechanisms.

- A possible stateful solution would be to resort to the LISP-TE features described in [142]. [142] specifies what is called an ELP which allows to add the list of RLOC to go through to every packet. The source host could craft LISP packets so that each subflow is encapsulated with a different LISP paths. This approach has three main drawbacks. First it implies the source host can craft LISP packets in addition to the routers. Secondly, the source host needs to learn different LISP paths which is not yet straightforward. Finally, it adds the overhead of the list of RLOCs to each packet.

- A stateless solution could be - as described in [143] - to carefully choose IP or TCP header fields so that IP packets follow a foreseen physical IP path. The computation of these fields (for instance the Time To Live (TTL) value in IP packets or source port in TCP packets) require the knowledge of both the topology and the load balancing algorithms running in the nodes to be able to correctly choose the path. Nodes usually use an hash function over IP and some TCP fields to compute a unique flow identifier that decides the egress interface. A stateless solution relying on hashing mechanisms would need to change field values until reaching an adequate hash which is slow. As explained in [143], this approach works best with an invertible load balancing algorithm and can reach important improvements in a data center environment. This technique is not limited to Layer-3 routing but can also apply to Layer-2 technologies with a hop-count such as TRILL.

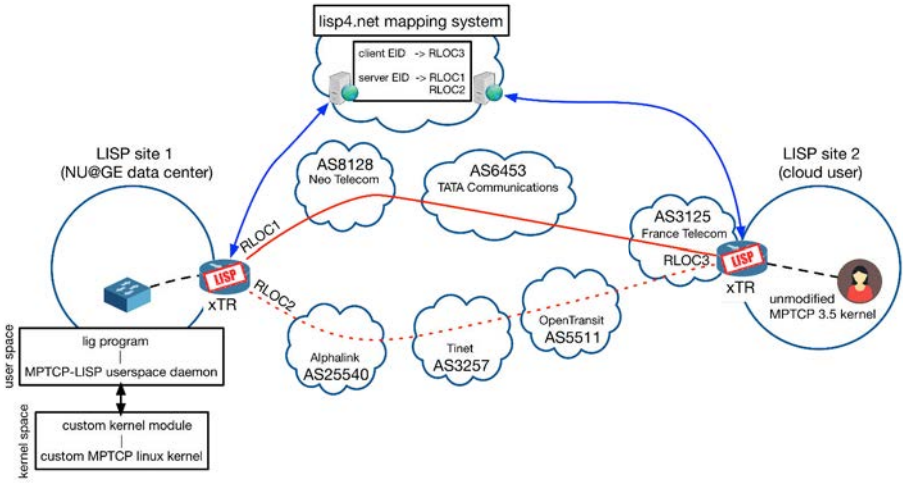


Figure 6.3: Cloud network test bed scenario.

The latter solution is viable in controlled environments, for IP-based intra-DC communications. The computation of the different fields could be to an advanced Layer-2 fabrics allowing DC traffic engineering such as OpenFlow. As for the extra-DC segment, stateless solutions do not allow enough control to choose paths. The LISP-TE features could thus prove helpful since it explicitly encodes some LISP nodes to go through, but requires the usage of a spread enough WAN operational LISP network to prevent packets from making a prejudicial detour because they have to go trough LISP routers.

The requirement is, that the MPTCP endpoint is able to query the LISP mapping system to get the locators of the other endpoint, therefore using the LISP control plane messages for this purpose (i.e., Map-Request and Map-Reply messages). Once this information is obtained, the MPTCP endpoint opens the required subflows following an adequate procedure, as detailed hereafter.

6.4 Experimental Results

We give in this section a synthetic description of the experimental test bed, the related open source developments, and relatively complex required details to reproduce our experimentation.

6.4.1 Network Test Bed

Let us illustrate the experimental test bed we used for the experimentation of our solution, displayed in Figure 6.3. We used a MPTCP-capable virtual machine hosted in the Paris-Est DC of the French NU@GE project [80], disposing of two ISPs, Neo

Telecom (AS8218) and Alphalink (AS25540). On the other side, we use a single-homed and MPTCP-capable cloud user. The Internet-working between the cloud server and user is such that two disjoint Internet paths are used down to the user's ISPs, and such that the two intra-DC paths between the VM hypervisor and the DC border LISP router are disjoint ones.

In such configuration, the highest improvements can be reached when the cloud user's provider access and transit links are not congested. The test bed scenario can be considered as the most representative one for real cases, where typically cloud users are not multihomed and the DC has a few ISPs. Moreover, by targeting the more basic configuration with only two subflows we can more effectively show the benefit of our LISP-MPTCP cross-layer cooperation solution.

6.4.2 Open Source Nodes

In terms of open source software, we used the open source MPTCP Linux implementation [141] and the LISPmob [81] implementation (preferred over the BSD OpenLISP router [23] [106] because more easily customizable). We then applied these necessary modifications to the open source nodes:

- to the MPTCP kernel so that it retrieves the number of local and remote RLOCs for each destination it is talking to;

- to a modified LISPmob [81] router so that it balances two subflows deterministically between its two RLOCs. Instead of resorting to a hash of the tuple (source IP,destination IP, Source port, Destination Port, Time To Live), we replaced the load balancing mechanism by a single modulo (number of remote RLOCs) on the TCP source port and force MPTCP to start TCP subflows with certain source port numbers.

To ease the development and to de-correlate the path discovery mechanism from other MPTCP mechanisms as stated in [144], we minimized the modifications to the kernel and instead created a kernel module called by the kernel each time MPTCP establishes a new connection. We also used an existing user space program named LISP Internet Groper (LIG) [145, 146]) to request EID-to-RLOC mappings, using LISP Map-Request and Map-Reply control plane messages. The LISP nodes were connected to the LISP Beta Network test bed [147]. In order to retrieve a list of RLOCs responsible for an EID (and implicitly their number), we created a user space daemon listening to requests transmitted by the kernel module, which then interacts with the LIG program. In the following, we do not differentiate between the 2 user space programs since it is just a matter of implementation.

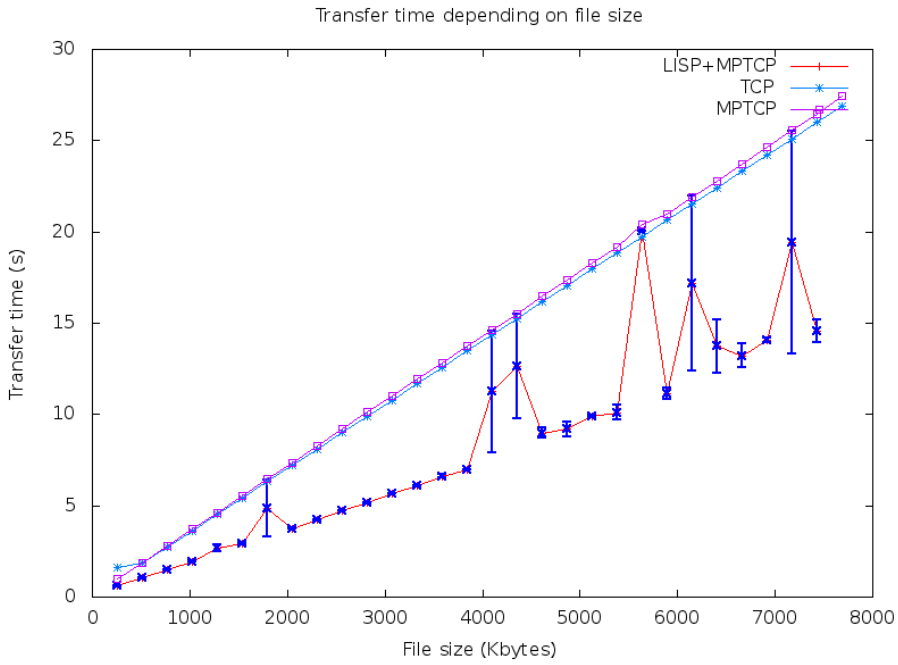


Figure 6.4: Completion times for different file sizes.

6.4.3 Transfer Times

In order to demonstrate the previously described architecture, we worked on the same topology as in Figure 6.1. A single-homed client (with one link to AS3215) downloads 40 files from a single-homed server (one link to a LISP software router) in a dual-homed data center (both links active, one towards AS25540, another one towards AS6453). Each transferred file is set bigger than the previous one by an increment of 256kb to assess the performance for different volumes. We record five transfer completion times for each file, and we repeat this whole procedure using three different configurations:

1. legacy TCP: with no cooperation with LISP, and a single TCP flow.
2. MPTCP: with no cooperation with LISP, and a single MPTCP subflow.
3. LISP + MPTCP: with cross-layer cooperation, creating as many subflows as the product of remote and local RLOCs, i.e., 2 subflows.

One can reasonably expect the cases 1 and 2 to be very similar in the provided configuration since MPTCP should use one flow only. During our tests, the server upload speed could fill the client download speed (8Mbit/sec, corresponds to RLOC 3 in Figure 6.3) with a single TCP connection. In order to exhibit an increase in

throughput via the use of several subflows, we limited RLOC 1 and 2 throughput via the linux QoS utilities so that each each RLOC could send no more then 2Mbit/sec. On Figure 6.4, we see as expected that unassisted MPTCP (i.e., MPTCP without LISP support, marked 'MPTCP') and TCP transfer times are quite close, MPTCP being a little slower than TCP; the cause should be the additional 20 bytes overhead per packet introduced by MPTCP. More importantly, we can see that our solution (marked 'LISP+MPTCP') performs significantly better. For almost all file sizes, we get a gain close to 100%, i.e., with the additional LISP-enabled subflow we can halve the transfer time. This very clearly shows the important benefit we can reach with the MPTCP-LISP cross-layer cooperation module we propose.

Some visible variations show that the performance gap is lower in some cases, i.e., for around 1800 Kbytes, 4500 Kbytes, 5800 Kbytes and higher file sizes. Two possible reasons can be attributed to these point artifacts. One reason could be that the user has a single ISP, and that point congestion situations appear in the provider-user network paths. Another reason could be a long latency in obtaining LISP mappings, which can depend on EID-RLOC mappings with exceeded TTLs (inducing one single subflow until the reception mapping reply). The fact that these point peaks show especially for higher loads, and the measurement of low LISP mapping system latencies during the test, would suggest the reason is the first one.

6.4.4 Data Plane Overhead

From a data plane forwarding perspective, we recall that LISP performs an IP-in-IP encapsulation with shim User Datagram Protocol (UDP) and LISP headers. In order to get an idea of the LISP overhead, we recorded the transfer times in two cases. A first case with two MPTCP subflows enabled via LISP, and a second case with the same two MPTCP subflows, but manually configured, without LISP, hence avoiding data plane encapsulation overheads. The results are shown in Figure 6.5. At a first look one can immediately notice that the overhead is most of the time negligible. It is worth noting that neither our connection or LISPmob allowed us to test higher rates (Nevertheless, we suspect that at higher rates, we might see a more important processing overhead of LISP traffic since the UDP encapsulation prevents the system from offloading TCP segmentation to the Network Interface Controller (NIC)). Comparing Figure 6.4 with Figure 6.5, the same artifacts appear in the same instants. We find that this overhead is largely balanced by the added throughput made available thanks to the MPTCP-LISP cross-layer cooperation.

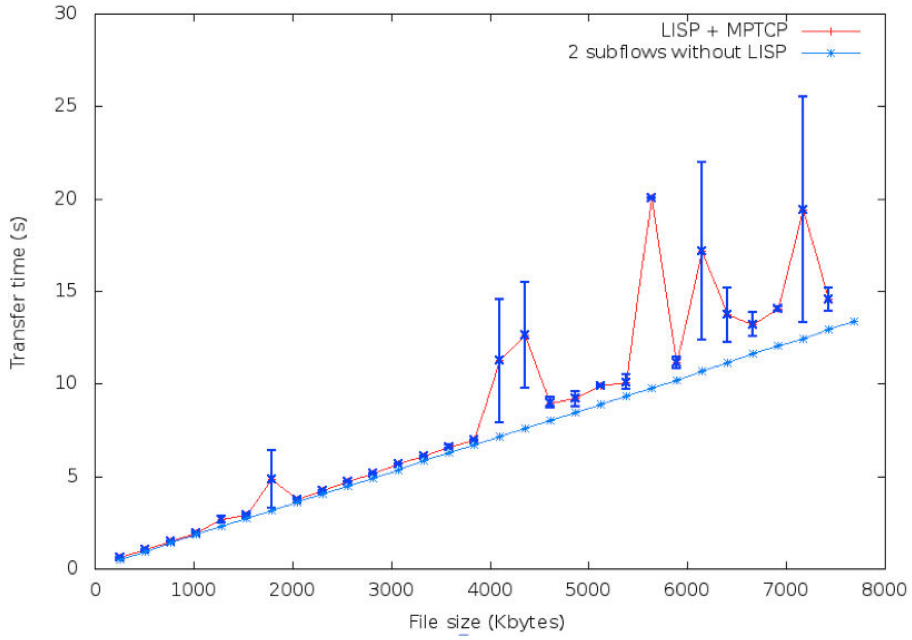


Figure 6.5: Transfer times for two subflows with and without LISP.

6.5 Summary

In this Chapter we have shown that the current that augmented MPTCP can achieve better performance with a better knowledge of the underlying IP topology gathered thanks to LISP. Our experimentation on a real large-scale test bed involving one data center network reveal that the throughput, hence the transfer times, can be greatly improved. Our proposition consists in allowing an MPTCP endpoints to gather information about the IP Routing Locators using LISP control plane messages to query the LISP mapping system. We show that with just one additional LISP-enabled subflow, a transport-layer connection spanning the Internet via disjoint AS paths can terminate file transfers two times faster. It is therefore reasonable to generalize these results stating that, in absence of middle-boxes filtering the MPTCP signaling, the achievable gain is directly proportional to the number of additional LISP-enabled subflows. Hence the higher the multihoming degree of Cloud clients and data-centers, the higher the achievable gain.

Chapter 7

Automated Routing Locator Switching

We propose unified cross-layer framework between LISP and TRILL protocols, to automate the routing locator priority management in a distributed data center fabric. As shown in our reference architecture (Chapter 3), LISP is used as the cloud access overlay protocol, while TRILL is used as the geo-distributed DC virtual network overlay protocol. We specify and design a cross-layer protocol agent able to map virtual network embedding information from TRILL to LISP in order to allow cloud providers to automatically map their intra-DC network metrics into overlay access priorities in order to better manage the access to distributed data centers.

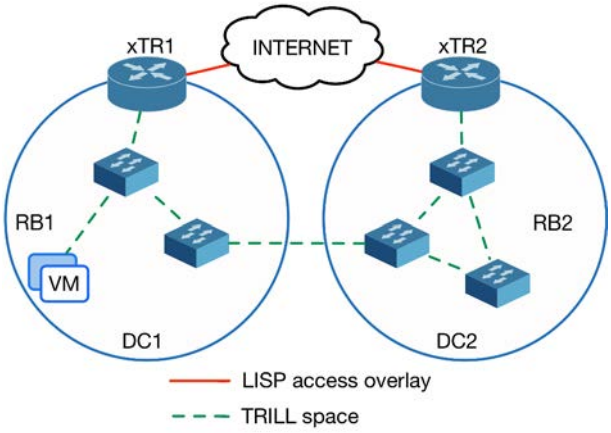


Figure 7.1: Reference distributed DC protocol architecture.

7.1 Introduction

Cloud computing research is recently being re-centered toward the needs of services requiring or benefiting from a continuous and persistent virtual resource orchestration across distributed data centers. Geographically distributing DCs [115] ensures network resiliency and high availability to applications, eventually guaranteeing a better user experience in terms of availability and cloud data retrieval latency. In a distributed DC, users are able to access their VM through multiple access gateways without affecting the established session. In order to orchestrate computing and storage services and to chase resource optimization, several effort have been made since a few years in protocol design and experimentation.

In the following, we report results related to an experimental distributed DC network. We present a solution to manage a distributed DC linking LISP and TRILL control planes at the intra-DC data-link layer. The unified control plane is used to synchronize routing and mapping information related to VMs, thus improve cloud access priority management.

7.2 Protocol Architecture

In this section, we describe the functional diagram of our solution at both the TRILL and LISP protocol levels. The reference deployment of TRILL can be present in each virtualization server, and possibly also at the DC physical switch level (despite this is not strictly required by TRILL as there is no need to have direct physical link between RBridges).

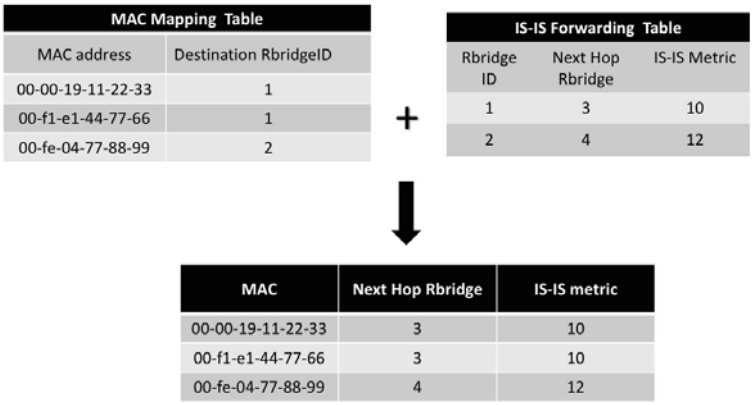


Figure 7.2: Correspondence between TRILL and ISIS tables.

7.2.1 Functional Requirements

The migration of a VM generates an automatic TRILL-ISIS path cost update, essentially because the egress RBridge of a given VM, identified by a unique MAC address, changes. In this context, our agent needs to instantaneously detect whenever a VM migration occurs to: (i) query the TRILL-ISIS control plane table; (ii) determine the next RBridge hop and issue; (iii) update to the LISP control-plane. More precisely, RLOC priorities should be equal or somehow proportional to the TRILL-ISIS path cost.

In order to support these operations, we need to merge the ISIS forwarding table and the MAC mapping table to get a direct correspondence between the computed ISIS metric of the path and the MAC address of the related RBridge, as shown in Figure 7.2.

In fact, the MAC mapping table generated by TRILL shows three VMs with different MAC addresses. The first two VMs try to reach the RBridge 3 and the last VM tries to reach the RBridge 4. We suppose that these VMs are directly linked to RBridge 1. The ISIS database provides the required information about the path cost that links RBridge 1 (i.e., Ingress RBridge) to both destination RBridges (i.e., Egress RBridges). In order to calculate the distance between a VM and the destination RBridge, we use the mapping tables to find the correspondence between destination RBridge IDs and its related ISIS metric.

These new calculated metrics are considered as priorities that can classify RLOCs from the most preferable to the least one with respect to the TRILL-ISIS routing metric. After receiving the notification from the agent, xTRs modify their configuration by changing the content of the Map-Register message and add the ISIS metric that corresponds to the path cost linking the migrated VM to the old and the new destination xTR. The new generated message is sent to the Map Server (MS)

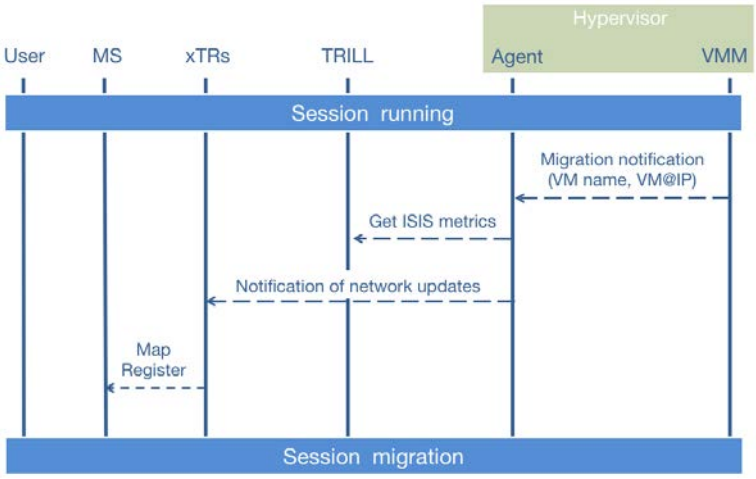


Figure 7.3: LISP-TRILL Agent signaling steps triggered upon VM detection.

in order to make the required updates to the LISP mapping system. The signaling diagram in Figure 7.3 represents the corresponding message exchange.

7.2.2 The Role of a Unifying Agent

The agent has an important role in synchronizing LISP and TRILL control planes. We summarize its main actions as follows:

- it detects the migration or restoration of a VM (or a change in the topology), recovering its main information such as its name and its EID.
- It gets the new ISIS path cost from the local LISP router(s) to the VM virtualization server.
- It solicits to the LISP router a RLOC priority update locally to the xTR and externally to the mapping system [106].

The process of the agent across its possible states and actions is represented in the functional diagram of Figure 7.4. We distinguish four main steps:

1. **VM detection:** the agent, located at the hypervisor level, monitors new VMs joining local hypervisor networks. Once a new VM is spotted, the agent retrieves its EID. Figure 7.5 represents VM detection tasks. Then, the agent retrieves the MAC address of the xTR(s), and looks for the Egress RBridges that are directly linked to it(them) in order to calculate the ISIS metric of the path separating the VM from the xTR(s).
2. **Destination RBridge identification:** in order to find the MAC address of the destination RBridge, the agent solicits the TRILL node configuration

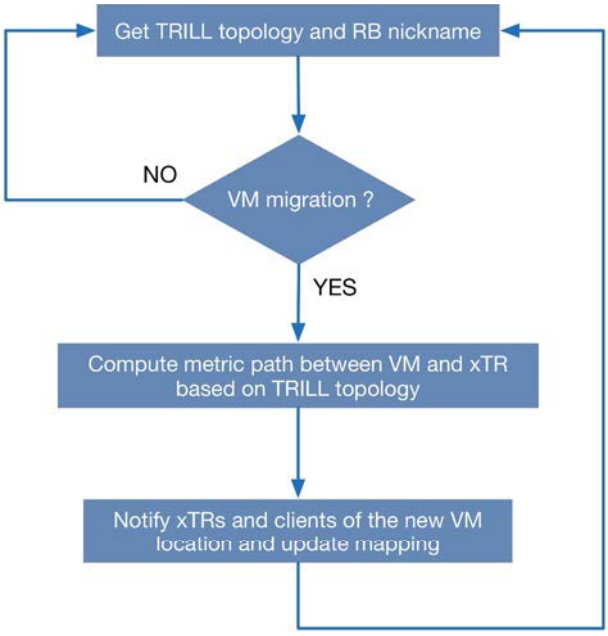


Figure 7.4: Diagram of the agent process.

and states, retrieving network topology and nicknames of the active RBridges. Thus, the MAC address of the corresponding xTR and the MAC address of the attached RBridges can be retrieved.

3. **ISIS path cost retrieval:** at this stage, the agent retrieves the path cost associated to the identified RBridge from the ISIS routing table.

4. **Mapping policy update:** the mapping entry can be changed accordingly to the retrieved ISIS path cost from the xTR(s) to the RBridge. The agent reconfigures the xTR database though a dedicated API to set the RLOC priority, triggering an explicit Map-Register.

This whole process is repeated periodically in order to guarantee a continuous alignment between the TRILL and LISP control planes.

7.3 Test Bed and Experimental evaluation

TRILL has a Linux kernel implementation developed by Gandi[1]. That implementation uses a modified version of Quagga [117] for the ISIS link-state signaling and routing table management. Our agent uses telnet to connect to Quagga in order to get TRILL node configuration and retrieve ISIS metric information using a simple command line interface. The agent prototype is written in Ruby [116] and is able to

[1]https://github.com/Gandi/ktrill

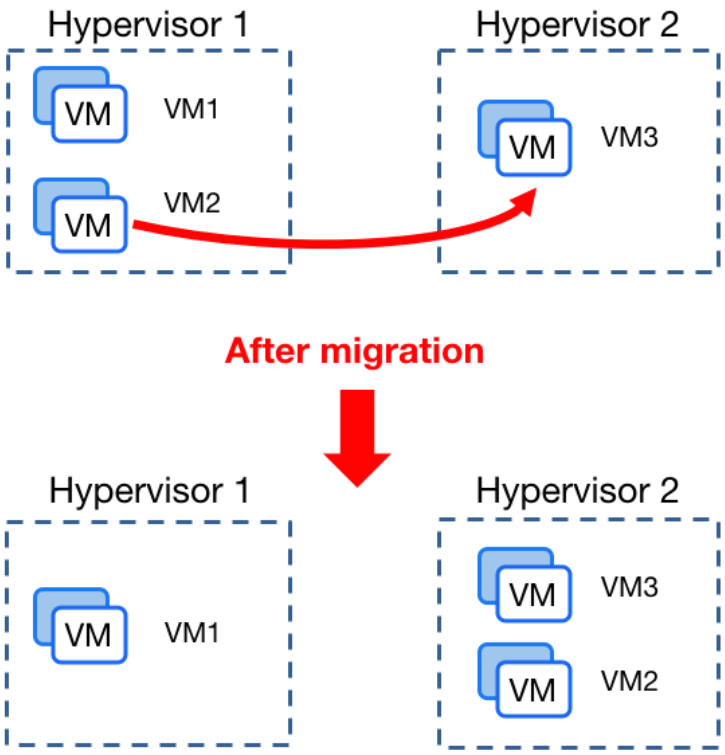


Figure 7.5: VM detection process.

handle the management of a set of VMs by using Libvirt [120]. We used the OpenLISP development version [107, 122], maintained by LIP6, which includes specific control plane features to support VM migration with LISP [1, 106].

We conducted an experimental campaign on a real distributed DC test bed configuration, involving computing resources in cities such as Pornic and Paris, in France. The emulated scenario is represented in Figure 7.6. We use a simple topology with 4 RBriges, RB1, RB2, RB3 and RB4, a VM directly attached to RB3 that must stay reachable via RB1. The ISIS metric configuration is depicted in Figure 7.6. When all links operate normally, the VM reaches RB1 via the direct link separating RB1 from RB3 with an ISIS metric equal to 10. Then, we emulated a network link failure, setting this direct link to down. At this stage, the VM sends traffic to RB1 via RB2 and then the ISIS metric changes and becomes equal to 40.

We built up a complete LISP-TRILL network context to run the scenario on, including therefore xTRs and RBridge in both Pornic and Paris DCs. We open connections to the VM before and after migrating it from Pornic to Paris (changing the topology). We place one LISP client in Pornic, one in Paris, and we compare the latency measurements from both clients.

We compare the legacy situation where the agent performing LISP-TRILL con-

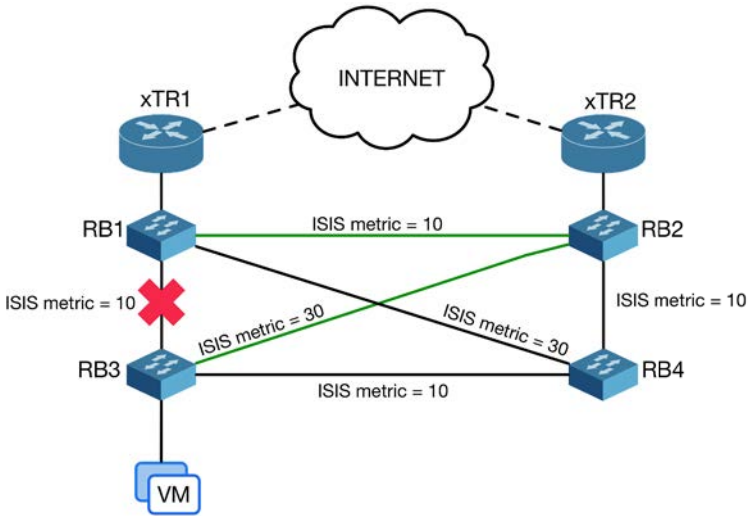


Figure 7.6: Representation of a topology change scenario.

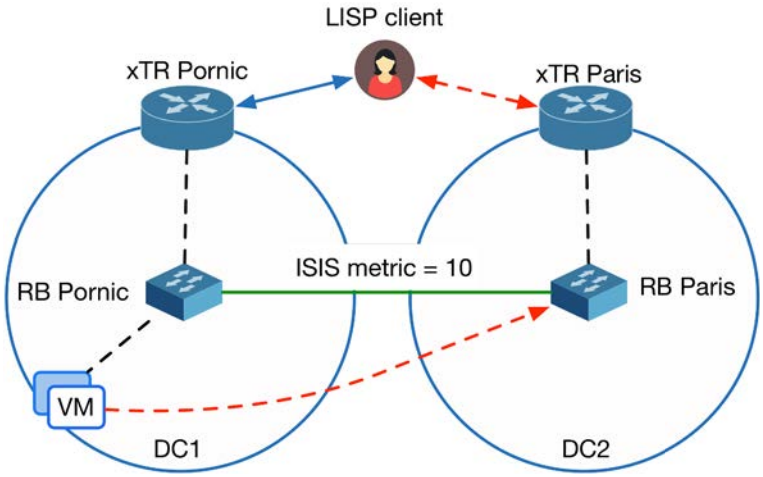


Figure 7.7: Legacy situation a LISP-TRILL control-plane agent.

trol plane synchronization is not present, to the case where it is present, as in Figure 7.7.

To test the efficiency of the proposed solution, we focus on the latency offered to the clients as the most important performance metric. We repeated the scenario a few dozens of times. We compare the legacy situation (agent deactivated) to our solution (agent activated) for the two VM states: before VM migration (Figure 7.8) and after VM migration (Figure 7.9 - note that part of the vertical axis is cut from roughly 2 ms and 28 ms for the sake of readability). The results are shown using boxplots indicating the minimum, first quartile, median, third quartile and maximum values. We can easily note that, before migration, the maximum Round-Trip Time (RTT) is 14.1 ms for the legacy situation, while it is 14.4 ms for our

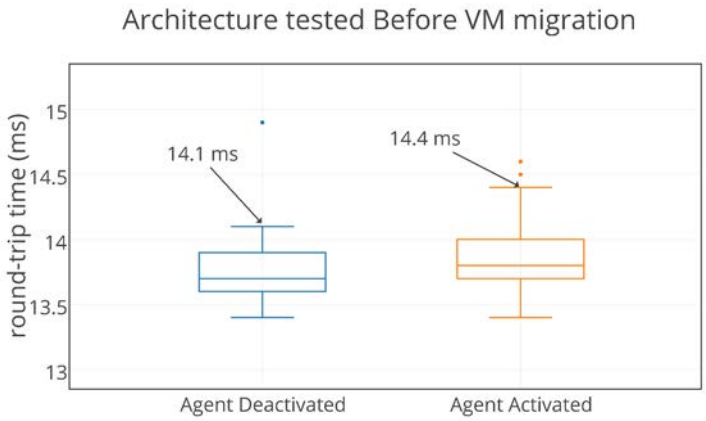


Figure 7.8: RTT performance before VM migration.

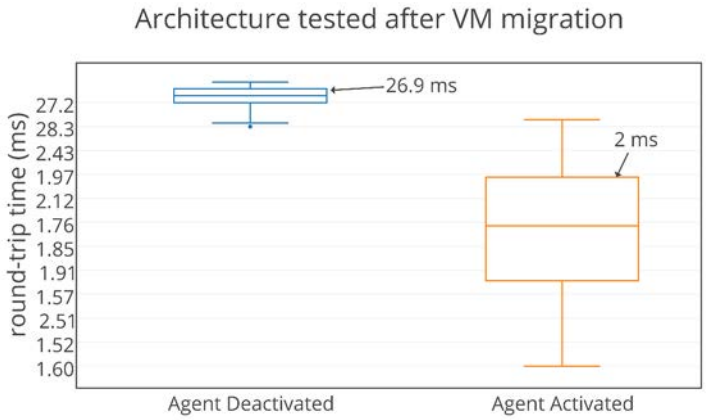


Figure 7.9: RTT performance after VM migration.

solution. A similar gap exists also between the median and the other quartiles. After migration, we can better appreciate the benefits of our approach that, for the given setting, offers at much lower RTT.

7.4 Summary

We proposed a unified LISP-TRILL control plane solution to provide optimized access priorities to the cloud IaaS VM-based services hosted in a distributed DC network. The targeted framework is the one where IaaS users access their hosted services by means of the LISP protocol, across the Internet, when the distributed DC fabric is managed by TRILL protocol. We specified the requirements and the implementation steps of a distributed agent architecture that maps the TRILL-ISIS routing states to reach hosted VMs into the LISP mapping metrics that guide

the choice on the access DC or routing locator in a distributed DC fabric. We implemented the agent and run experiments over a real geographically distributed DC test bed in France, testing a proof-of-concept prototype and experimentally proving the interest of our solution.

Chapter 8

Conclusion and Perspectives

Today, it is undeniable that the cloud industry started to dominate the IT market by offering to users a plethora of applications and services that can be accessed everywhere and anytime. However, rarely users can get what they expect especially when cloud providers seek to increase their own profit at the expense of their customers' satisfaction. In this dissertation, we aimed to find cloud networking solutions (i.e., protocols, algorithms, etc.) allowing providers to conceive user-centric clouds that meet user's expectations by respecting the cloud access service level agreements.

It is still unclear how user's satisfaction is perceived and quantified by cloud providers. In Chapter 3 we argued that even if the user's quality of experience is unique and subjective, it requires a seamless and transparent access to cloud services, with low latency connectivity and high availability. We proposed and motivated a holistic cloud networking architecture on top of geo-distributed and multihomed data center facilities.

In order to offer a seamless connectivity for cloud users when virtual resources (in particular, virtual machines) are migrated between distant data centers over the Internet for maintenance, we proposed in Chapter 4 a framework using LISP as a cloud access overlay (Layer-3) for distributed data centers. By adding several missing functionalities to LISP from one side and to the hypervisor user-space from the other side, we were able to achieve a programmable protocol that allows to automatically relocate virtual machines with a minimum connectivity downtime over large scale geo-distributed data centers.

Another concern is when the link between a user and its cloud application fail to provide a satisfactory experience as a possible consequence of: (i) a traveling user moving geographically farther from its cloud actual delivery point; (ii) bottlenecks on the cloud access networks. In this context, we designed in Chapter 5 a cloud

access overlay protocol and controller for distributed data centers that is able to detect impairments on the user-to-VM link, optimize the data center routing locator switching when needed and optimizes virtual machine migrations to nearby data center facilities. After running experiments on a real test bed and simulations with real anonymized user mobility traces, we found considerable reduction in terms of latency as well as a significant gain of TCP throughput.

Sometimes it might seem interesting to load balance the traffic instead of engaging a resource heavy migration which can be costly to cloud providers. In Chapter 6, we studied whether providing transport layer (Layer-4) load balancing, such as multipath transport, over an access overlay protocol can positively impact the experience of the user. In fact, by combining the path diversity provided by LISP and the platform suggested in Chapter 5 with MPTCP, multiple subflows can be created on-demand thus granting additional performance gain from a throughput perspective.

Furthermore, from a cloud networking perspective, providing resilient core architectures in a distributed data center environment can be beneficial to cloud users. Although the contribution in Chapter 7 is still at its early stages, we developed a proof-of-concept prototype that unifies LISP (Layer-3) and TRILL (Layer-2) control planes in order to optimize the access to the cloud respecting the intra-DC network topology. The first test results that we obtained are encouraging, and showed that the combination of LISP as an access overlay protocol and TRILL as an intra-DC protocol can minimize the latency between the user and its VM by automatically changing priorities to different DC entry points.

In conclusion, the purpose of the above contributions is to be integrated into one and unique platform in order to build holistic user-centric distributed clouds. In fact, the solutions that we proposed in each chapter could be combined together via a common API (Figure 8.1), thus offering to different protocols from different layers an easy way of communication. Layer-2 protocol solutions are combined with overlay Layer-3 protocols in order to automatically manage data center entry points policies. Layer-3 overlay protocol supports the usage of an optimized path based on the user-to-VM link respecting the information communicated by Layer-2 protocols. Finally, Layer-4 cross-layer solutions increase the transfer rates of cloud applications.

This study was conducted from a networking perspective, however from a system perspective several open questions are yet to be addressed:

- robust decision making issues are not addressed in this dissertation and deserve more attention. For instance, optimizing the user's experience may results

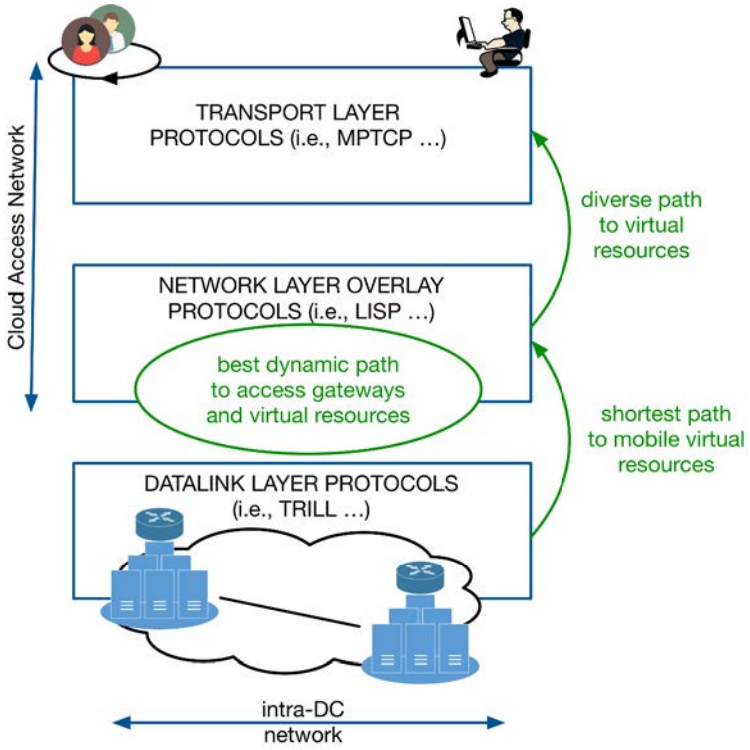


Figure 8.1: Our reference distributed cloud architecture.

into frequently changing access gateways and triggering a virtual machine migration, especially for mobile users. How to make virtual machine migration robust and reliable is an open research challenge.

- Security remains an important issue in today's cloud industry. Cloud users want a reliable and private connection to their applications. While LISP provides basic security measures, they still need to be improved. By applying strict security measures on overlay access protocols, the user's connectivity may suffer from higher latency, low transfer rate, etc. How security measurements can be applied on a user-centric cloud architecture without affecting their level of satisfaction?

- We studied in this dissertation the use case of one or multiple user(s) accessing to their individual virtual machines. However, media streaming applications have more than one user. In this particular case, what can be done to satisfy all the users, and how shared virtual machine migrations should be handled?

- As data centers are getting smaller, cheaper and geographically distributed, new paradigms are emerging. Today, we are talking more and more about hybrid clouds, where micro and proximity data centers are pushed closer to (mobile) users releasing remote data centers from the processing burden and

granting faster and reliable connectivity. Hybrid clouds gave birth to new cloud concepts which are starting to get deployed in our daily life. For instance, "Fog Computing" is a rising model for the Internet of Things (IoT) deployment: data are collected and processed via sensors locally then are pushed to remote data centers for further processing. In addition, nearby cloud affiliations can support device-to-cloud computation offloading. How new data and computation intensive affiliations for the IoT can be designed is another promising research field.

At last it is worth mentioning that this work was supported by several successful research projects: the French "Investissement d'Avenir" NU@GE project, 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.

Publications

International journal with peer review

[1] P. Raad, S. Secci, D. Phung Chi, A. Cianfrani, P. Gallard, and G. Pujolle. "Achieving Sub-Second Downtimes in Large-Scale Virtual Machine Migrations with LISP". *IEEE Transactions on Network and Service Management*, vol. 11 (2), pages 133-143, (ISBN: 1932-4537) (2014).

International conferences with peer review

[2] R. Touihri, P. Raad, N. Turpault, F. Cachereul, and S. Secci. "Unifying LISP and TRILL Control-Planes for Distributed Data-Center Networking". In *IEEE/IFIP Network Operations and Management Symposium*, Istanbul, Turkey.

[3] P. Raad, S. Secci, D. Phung Chi, and P. Gallarad. "PACAO: Protocol Architecture for Cloud Access Optimization". In *IEEE 1st International Conference on Network Softwarization (NETSOFT)*, London, United Kingdom, pages 1-9, (2015).

[4] P. Raad, G. Colombo, D. Phung Chi, S. Secci, , A. Cianfrani, P. Gallard, and G. Pujolle. "Achieving Sub-Second Downtimes in Internet-wide Virtual Machine Live Migrations in LISP Networks". In *IEEE/IFIP International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, pages 286-293 (2013).

[5] M. Coudron, S. Secci, G. Pujolle, P. Raad, and P. Gallard. "Cross-layer Cooperation to Boost Multipath TCP Performance in Cloud Networks". In *IEEE International Conference in Cloud Networking (CloudNet)*, San Francisco, CA, USA, pages 58-66, (2013).

[6] D. Phung Chi, S. Secci, G. Pujolle, P. Raad, and P. Gallard. An Open Control-Plane Implementation for LISP networks". In *IEEE 3rd International Conference on Network Infrastructure and Digital Content (NIDC)*, Beijing, China, pages 266-270, (2012).

[7] P. Raad, G. Colombo, D. Phung Chi, S. Secci, , A. Cianfrani, P. Gallard, and G. Pujolle. "Demonstrating LISP-based Virtual Machine Mobility for Cloud Networks". In *IEEE 1st International Conference on Cloud Networking (CloudNet)*, Paris, France, pages 200-202, (2012).

Submitted

[8] P. Raad, S. Secci, and P. Gallard. "Linking Virtual Machine Mobility to User Mobility". *IEEE Transactions on Parallel and Distributed Systems* (submitted).

References

[8] VMware. `http://www.vmware.com`.

[9] B. Davie (VMware). "Network Virtualization: Enabling the Software-Defined Data Center". *IEEE Cloudnet 2013*, Keynote (2013).

[10] T. Koponen, et al. "Onix: a distributed control platform for large-scale production networks". *The 9th USENIX conference on Operating systems design and implementation* (2010).

[11] S. Jain, et al. "B4: experience with a globally-deployed software defined wan". *ACM SIGCOMM conference* (2013).

[12] A. Kostopoulos, et al. "Towards multipath TCP adoption: challenges and opportunities". *The 6th EURO-NF Conference on Next Generation Internet* (2010).

[13] M. Armbrust, et al. "A berkeley view of cloud computing". *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28* (2009).

[14] GoGrid. "Cloud hosting: Instant windows and linux cloud servers". `http://www.gogrid.com` (2009).

[15] R. Prodan, and S. Ostermann. "A survey and taxonomy of infrastructure as a service and web hosting cloud providers". *The 10th IEEE/ACM International Conference on Grid Computing*, pages 17-25 (2010).

[16] V. Fuller, and D. Farinacci. "LISP Map Server Interface". *RFC 6833* (2013).

[17] "IEEE Standard for Local and metropolitan area networks– Bridges and Bridged Networks - Amendment 23: Application Virtual Local Area Network (VLAN) Type, Length, Value (TLV)". *IEEE Std 802.1Qcd-2015*, pages 1-108 (2015).

[18] L. Dunbar, et al. "TRILL Edge Directory Assistance Framework". *RFC 7067* (2013).

[19] D. Meyer, et al. "Report from the IAB Workshop on Routing and Addressing". *RFC 2439* (2007).

[20] C. White, et al. "LISP Mobile Node". *draft-meyer-lisp-mn-13* (2015).

[21] Cisco, Inc. "Locator ID Separation Protocol (LISP) VM Mobility Solution". `http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/locator-id-separation-protocol-lisp/lisp-vm_mobility_wp.pdf`, white paper (2011).

[22] D. Lewis, V. Fuller. "LISP Delegated Database Tree". *draft-ietf-lisp-ddt-03* (2015).

[23] L. Iannone, et al. "OpenLISP: An Open Source Implementation of the Locator/ID Separation Protocol". *ACM SIGCOMM* (2009).

[24] A. Kivity, et al. "kvm: the Linux virtual machine monitor". *The Linux Symposium* (2007).

[25] M. Nelson, et al. "Fast transparent migration for virtual machines". *The annual conference on USENIX Annual Technical Conference* (2005).

[26] P. Svard, et al. "High performance live migration through dynamic page transfer reordering and compression". *IEEE 3rd International Conference on Cloud Computing Technology and Science* (2011).

[27] H. Watanabe, et al. "A Performance Improvement Method for the Global Live Migration of Virtual Machine with IP Mobility". *The 5th International Conference on Mobile Computing and Ubiquitous Networking* (2010).

[28] Border Gateway Protocol. `http://docwiki.cisco.com/wiki/Border_Gateway_Protocol`.

[29] P. Barham, et al. "Xen and the art of virtualization". *The 9th ACM symposium on Operating systems principles* (2013).

[30] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiberg. "Live wide-area migration of virtual machines including local persistent state ". *in Proc. of ACM CoNext* (2007).

[31] J.S. Plank, Y. Kim, and J.J. Dongarra. "Diskless Checkpointing". *IEEE Transactions on Parallel and Distributed Systems*, vol. 9 (10), pages 972-986 (1998).

[32] M. Boucadair, and C. Jacquenet. "An MPTCP Option for Network-Assisted MPTCP Deployments: Plain Transport Mode". *draft-boucadair-mptcp-plain-mode-02* (2015).

[33] C. Clark et al. "Live migration of virtual machines". *The 2nd conference on Symposium on Networked Systems Design & Implementation* (2005).

[34] Cisco, and VMware. "Virtual Machine Mobility with VMware VMotion and Cisco Data Center Interconnect Technologies" . `http://www.cisco.com/c/dam/en/us/solutions/collateral/data-center-virtualization/data-center-virtualization/white_paper_c11-557822.pdf`, white paper (2009).

[35] D. Phung Chi, et al. “An open control-plane implementation for LISP networks”. *The 3rd IEEE International Conference on Network Infrastructure and Digital Content* (2012).

[36] P. Garg, and Y. Wang “NVGRE: Network Virtualization Using Generic Routing Encapsulation”. *RFC 7637* (2015).

[37] F. Travostino, et al. “Seamless live migration of virtual machines over the MAN/WAN”. *Future Generation Computer Systems*, vol. 22 (8), pages 901-907 (2006).

[38] D. Lewis, et al. “LISP Alternative Topology (LISP+ALT)”. *RFC 6836* (2013).

[39] S. Setty. “vMotion Architecture, Performance, and Best Practices in VMware vSphere 5”. `https://www.vmware.com/files/pdf/vmotion-perf-vsphere5.pdf`, white paper (2011).

[40] Prowess Consulting. “Quick Migration with Hyper-V”. *Microsoft Corporation* (2008).

[41] Hitachi Data Systems in collaboration with Microsoft, Brocade and Ciena. “Hyper-V Live Migration over Distance”. `https://www.hds.com/assets/pdf/hyper-v-live-migration-over-distance-reference-architecture-guide.pdf`, reference guide (2010).

[42] D. Kapil, E. Pilli, and R.C Joshi. “Live virtual machine migration techniques: Survey and research challenges”. *IEEE 3rd International Advance Computing Conference* (2013).

[43] Cisco, Inc. “Cisco Overlay Transport Virtualization Technology Introduction and Deployment Considerations”. `http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DCI/whitepaper/DCI3_OTV_Intro.pdf` (2012)

[44] C. Perkins. “IP mobility support for IPv4”. *RFC 3344* (2002).

[45] S. Gundavelli, et al. “Proxy mobile ipv6”. RFC 5213 (2008).

[46] E. Harney, et al. “The efficacy of live virtual machine migrations over the internet”. *The 2nd ACM international workshop on Virtualization technology in distributed computing* (2007).

[47] Q. Li, et al. “HyperMIP: hypervisor controlled mobile IP for virtual machine live migration across networks”. *IEEE High Assurance Systems Engineering Symposium* (2008).

[48] C. Clark et al. “Live migration of virtual machines”. *The 2nd conference on Symposium on Networked Systems Design & Implementation* (2005).

[49] D.L. Mills. “Internet time synchronization: the network time protocol”. *IEEE Transactions on Communications*, vol. 39 (10), pages 1482-1493 (1991).

[50] OpenLISP control plane. `http://github/lip6-lisp/control-plane`.

[51] T. Sridhar, et al. "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks" (2012).

[52] E.B. Davie, and J. Gross. "LA Stateless Transport Tunneling Protocol for Network Virtualization (STT)". *draft-davie-stt-02* (2013).

[53] Gartner Cloud Computing Forcasts Update. `http://www.gartner.com/newsroom/id/2613015/`.

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

[55] B. Gedik, and L. Liu. "Protecting location privacy with personalized k-anonymity: Architecture and algorithms". *IEEE Transactions on Mobile Computing*, vol. 7 (1), pages 1-18 (2008).

[56] L. Qiu, et al. "On selfish routing in internet-like environments". *The 2003 ACM conference on Applications, technologies, architectures, and protocols for computer communications* (2003).

[57] C. White. "LISP Mobile Node". *draft-meyer-lisp-mn-12* (2015).

[58] ETSI MEC Industry Specification Group. "Mobile-Edge Computing – Introductory Technical White Paper". `https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf`, white paper (2014).

[59] T. Taleb, A. Ksentini. "QoS/QoE predictions-based admission control for femto communications". *IEEE International Conference on Communication* (2012).

[60] A. Ceselli, M. Premoli, and S. Secci. "Cloudlet Network Design Optimization". *IFIP Networking* (2015).

[61] S.H. Zanakis. "Multi-attribute decision making: A simulation comparison of select methods". *European journal of operational research*, vol. 107 (3), pages 507-529 (1998).

[62] J. Ramesh. "Quality of experience". *IEEE MultiMedia*, vol. 11 (1), pages 96-95 (2004).

[63] S. Aldrich, et al. "What kind of the total customer experience does your e-business deliver?". *Patricia Seybold Group* (2000).

[64] X. Xiao, and L.M. Ni. "Internet QoS: A big picture". *IEEE Network*, vol. 13 (2), pages 8-18 (1999).

[65] K. Nichols, and V. Jacobson. "A two-bit differentiated services architecture for the Internet". *RFC 2638* (1999).

[66] E. Rosen, et al. "Multiprotocol label switching architecture". *RFC 3031* (2001).

[67] D. Awduche, O. Daniel, and J. Agogbua. "Requirements for traffic engineering over MPLS". *RFC 2702* (1999).

[68] Empirix. "Assuring QoE on Next Generation Networks" (2003).

[69] A. Van Moorsel. "Metrics for the internet age: Quality of experience and quality of business". *Fifth International Workshop on Performability Modeling of Computer and Communication Systems*, vol. 34 (13), pages 26-31 (2001).

[70] O'Neil, and M. Timothy. "Quality of experience and quality of service for IP video conferencing" (2002).

[71] M. Fiedler, T. Hossfeld, and P. Tran-Gia. "A generic quantitative relationship between quality of experience and quality of service". *IEEE Network*, vol. 24 (2), pages 36-41 (2010).

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

[73] Cisco, Inc. "CAT 6500 and 7600 Series Routers and Switches TCAM Allocation Adjustment Procedures" (2015).

[74] R. Ahuja, et al. "Network flows: theory, algorithms, and applications". *Prentice-Hall, Inc.* (1993).

[75] ANR LISP-Lab Project. `http://www.lisp-lab.org`.

[76] FUI RAVIR project. `http://www.ravir.io`.

[77] ANR ABCD Project. `https://abcd.lip6.fr`.

[78] Skype. `https://www.skype.com`.

[79] WhatsApp. `https://www.whatsapp.com`.

[80] "Investissement d'Avenir NU@GE Project". `http://www.nuage-france.fr`.

[81] LISPmob. `http://www.lispmob.org`.

[82] GNU Linear Programming Kit. `https://www.gnu.org/software/glpk`.

[83] BGP Report. `http://www.potaroo.net`.

[84] OpenDaylight. `http://www.opendaylight.org`.

[85] M. Satyanarayanan, et al. "The Case for VM-Based Cloudlets in Mobile Computing". *IEEE Pervasive Computing*, vol. 8 (4), pages 14-23 (2009).

[86] T. Taleb, and A. Ksentini. "Follow me cloud: interworking federated clouds and distributed mobile networks". *IEEE Network*, vol. 27 (5), pages 12- 19 (2013).

[87] Google Voice. `https://www.google.com/voice/`.

[88] Apple Siri. `https://www.apple.com/ios/siri/`.

[89] Google Glass. `https://www.google.com/glass/start/`.

[90] Iperf. `https://www.iperf.fr`.

[91] G. Chuanxiong, et al. "Dcell: a scalable and fault-tolerant network structure for data centers". *ACM SIGCOMM Computer Communication Review*, vol. 38 (4), pages 75-86 (2008).

[92] G. Chuanxiong, et al. "BCube: a high performance, server-centric network architecture for modular data centers". *ACM SIGCOMM Computer Communication Review*, vol. 39 (4), pages 63-74 (2009).

[93] W.P. Turner, H. John, and W.E. Renaud. "Data center site infrastructure tier standard: Topology". *Uptime Institute* (2010).

[94] A. Klein, et al. "Access schemes for mobile cloud computing". *The 11th International Conference on Mobile Data Management* (2010).

[95] A. Klein, C. Mannweiler, and H.D. Schotten. "A framework for intelligent radio network access based on context models". *The 22nd WWRF meeting* (2009).

[96] A Ravi, and S.K. Peddoju. "Handoff Strategy for Improving Energy Efficiency and Cloud Service Availability for Mobile Devices". *Wireless Personal Communications* (2014).

[97] M. Satyanarayanan, et al. "The Case for VM-Based Cloudlets in Mobile Computing". *IEEE Pervasive Computing*, vol. 8 (4), pages 14-23 (2009).

[98] K. Kumar, and L. Yung-Hsiang. "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?". *Computer*, vol. 43 (4), pages 51-56 (2010).

[99] Cisco, Inc. "Locator ID Separation Protocol (LISP) VM Mobility Solution" (2011).

[100] A. Ksentini, T. Taleb, F. Messaoudi. "A LISP-based Implementation of Follow Me Cloud". *IEEE Access*, vol. 2, pages 1340-1347 (2014).

[101] A. Galvani, et al. "LISP-ROAM: network-based host mobility with LISP". *ACM MobiArch* (2014).

[102] D. Black, et al. "An Architecture for Overlay Networks (NVO3)". *draft-ietf-nvo3-arch-02* (2014).

[103] 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).

[104] B. Constantine, et al. "Framework for TCP Throughput Testing". *RFC 6349* (2011).

[105] H. Hamilton. "Architecture for modular data centers" (2006).

[106] D. Phung Chi, et al. "The OpenLISP control plane architecture". *IEEE Network Magazine*, vol. 38 (2), pages 34-40 (2014).

[107] L. Iannone, D. Saucez, and O. Bonaventure. "Implementing the locator/id separation protocol: Design and experience". *Computer Networks*, vol. 55 (4), pages 948-958 (2011).

[108] D. Saucez, L. Iannone, O. Bonaventure, and D. Farinacci. "Designing a Deployable Future Internet: the Locator/Identifier Separation Protocol (LISP) case". *IEEE Internet Computing* (2012).

[109] L. Jiao, et al. "Challenges and Opportunities for Cloud-based Computation Offloading for Mobile Devices". *Future Network & Mobile Summit* (2013).

[110] S. Secci, and S. Murugesan. "Cloud Networks: Enhancing Performance and Resiliency". *Computer*, vol. 47 (10), pages 82-85 (2014).

[111] R. Perlman, and D. Eastlake. "Introduction to TRILL". *The Internet Protocol Journal*, vol. 4 (3), pages 2-20 (2011).

[112] A. Amamou, K. Haddadou, and G. Pujolle. "A TRILL-based multi-tenant data center network". *Computer Networks*, vol. 68, pages 35-53 (2014).

[113] R. Perlman. "An algorithm for distributed computation of a spanningtree in an extended lan". *ACM SIGCOMM Computer Communication Review*, vol. 15 (4), pages 44-53 (1985).

[114] Z.K. Khattak, M. Awais, and A. Iqbal. "Performance Evaluation of OpenDaylight SDN Controller".

[115] M. Gharbaoui, B. Martini, and P. Castoldi. "Anycast-based optimizations for inter-data-center interconnections". *Journal of Optical Communications and Networking (Optical Society of America)*, vol. 4 (11), pages B168–B178 (2012).

[116] Ruby. `https://www.ruby-lang.org/en/documentation/`.

[117] Quagga. `http://www.nongnu.org/quagga/`.

[118] M. Bolte, et al. "Non-intrusive virtualization management using libvirt". *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 574-579 (2010).

[119] Openstack Cloud Software. `https://www.openstack.org/`.

[120] Libvirt The virtualization API. `http://www.libvirt.org/`.

[121] R. Kashyap, S. Chaudhary, and P.M. Jat. "Virtual machine migration for back-end mashup application deployed on OpenStack environment". *International Conference on Parallel, Distributed and Grid Computing (PDGC)* (2014).

[122] OpenLISP development version. `https://github.com/lip6-lisp/`.

[123] Alphalink. `http://www.alphalink.fr/`.

[124] C. Vázquez, et al. "On the use of clouds for grid resource provisioning". *Future Generation Computer Systems (Elsevier)*, vol. 27 (5), pages 600-605 (2011).

[125] T. Voith, K. Oberle, and M. Stein. "Quality of service provisioning for distributed data center inter-connectivity enabled by network virtualization". *Future Generation Computer Systems (Elsevier)*, vol. 28 (3), pages 554-562 (2012).

[126] E. Byun, et al. "Cost optimized provisioning of elastic resources for application workflows". *Future Generation Computer Systems (Elsevier)*, vol. 27 (8), pages 1011-1026 (2011).

[127] K. Halder, U. Bellur, and P. Kulkarni. "Risk aware provisioning and resource aggregation based consolidation of virtual machines". *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 598-605 (2012).

[128] A. Rodriguez-Natal, et al. "LISP: a southbound SDN protocol?". *Communications Magazine, IEEE*, vol. 53 (7), pages 201-207 (2015).

[129] , M. Casado, et al. "Fabric: a retrospective on evolving SDN". *The first ACM workshop on Hot topics in software defined networks* (2012).

[130] N. McKeown, et al. "OpenFlow: enabling innovation in campus networks". *ACM SIGCOMM Computer Communication Review*, vol. 38 (2), pages 69-74 (2008).

[131] Y. Rekhter, T. Li, and S. Hares. "A border gateway protocol 4 (BGP-4)" (2005).

[132] J. Moy. "Open shortest path first (ospf) version 2". *RFC 2328* (1998).

[133] V. Fuller, T. Li. "Classless inter-domain routing (CIDR): The Internet address assignment and aggregation plan" (2006).

[134] Massive Route Leak Cause Internet Slowdown. =http://www.bgpmon.net/massive-route-leak-cause-internet-slowdown.

[135] R. Callon. "Use of OSI ISIS for routing in TCP". *RFC 1195* (1990).

[136] Amazon EC2. `http://aws.amazon.com/ec2`.

[137] A. Fox, et al. "Above the clouds: A Berkeley view of cloud computing". *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, page 13 (2009).

[138] L. Ong, and J. Yoakum, "An introduction to the stream control transmission protocol (SCTP)". *RFC 3286* (2002).

[139] A. Ford, et al. "Architectural guidelines for multipath TCP development". *RFC 6182* (2011).

[140] O. Bonaventure. "Multipath TCP". *IEEE CloudNet 2012*, Tutorial. `http://www-phare.lip6.fr/cloudnet12/Multipath-TCP-tutorial-cloudnet.pptx` (2012).

[141] C. Raiciu, et al. "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP". *Proc. of USENIX Symposium of Networked Systems Design and Implementation (NSDI'12)*, San Jose, CA (2012).

[142] D. Farinacci, P. Lahiri, and M. Kowal. "LISP Traffic Engineering Use-Cases ", *draft-farinacci-lisp-te-09* (2015).

[143] G. Detal, et al. "Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks". *Computer Networks* (in press).

[144] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. "TCP Extensions for Multipath Operation with Multiple Addresses". RFC 6824, Jan. 2013.

[145] D. Farinacci, and D. Meyer. "The LISP Internet Groper". RFC 6835.

[146] The LISP Internet Groper (LIG) open source version. `https://github.com/davidmeyer/lig`

[147] LISP Beta Network testbed. `http://www.lisp4.net`.

[148] A.R. Curtis, W. Kim, P. Yalagandula, "Mahout: Low-Overhead Datacenter Traffic Management using End-Host-Based Elephant Detection". *Proc. of IEEE INFOCOM* (2011).

[149] D. Farinacci, D. Meyer, and J. Snijders. "LISP Canonical Address Format (LCAF)". *draft-ietf-lisp-lcaf-11* (2015).

[150] Open Networking Fundation. "Software-defined networking: The new norm for networks". *ONF White Paper* (2012).

[151] S. Hares, and R. White. "Software-defined networks and the interface to the routing system (I2RS)". *IEEE Internet Computing*, pages 84-88 (2013).

[152] Leading operators create ETSI standards group for network functions virtualization. `http://www.etsi.org/index.php/news-events/news/644-2013-01-isg-nfv-created` (2013).

[153] Linux Containers. `http://linuxcontainers.org`.

[154] Network Function Virtualization (NFV); Use Cases. `http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf` (2013).

[155] Google App Engine. `http://cloud.google.com/appengine/docs`.

[156] Micosoft Azure. `http://azure.microsoft.com`.

[157] Rackspace. http://www.rackspace.com.

[158] Amazon Web Services. http://aws.amazon.com.

UNIVERSITÉ PIERRE ET MARIE CURIE
LABORATOIRE D'INFORMATIQUE DE PARIS 6
4 PLACE JUSSIEU, 75005 PARIS