Unifying LISP and TRILL Control-Planes for Distributed Data-Center Networking

Roua Touihri[†*], Patrick Raad[†‡], Nicolas Turpault[*], François Cachereul[*], Stefano Secci[†]
[†] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6. Email: {roua.touihri, patrick.raad, stefano.secci}@upmc.fr
[*]Alphalink, Pornic, France. Email: {r.touihri, n.turpault, f.cachereul}@alphalink.fr.
[‡]Non Stop Systems, Nanterre, France. Email: praad@nonstop.fr

***Abstract*—Nowadays, the explosion of cloud-based applications is leading to a much higher demand on both computing and network infrastructure resources than only a few years ago. Enhancing the user experience, by reducing the latency and increasing network stability, becomes an important challenge for cloud operators. In this paper, we propose a unified protocol architecture, based on the Locator/Identifier Separation Protocol (LISP) and the Transparent Interconnection of a Lots of Links (TRILL) protocol, to enhance the access performance and to minimize the retrieval latency for services hosted in a distributed data center (DC) fabric. LISP is used as a cloud access overlay protocol, while TRILL is used as a 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 (layer 2) to LISP (layer 3) in order to allow cloud providers to let the client access the cloud by the best DC entry point, assuming that the inter-DC site latency dominates over the DC access latency. We tested our architecture in a real testbed. We compared the proposed solution to the legacy situation, highlighting the achievable gains in cloud access latency.**

I. 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 [10] 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 shall be made able to access their virtual machines (VM) through multiple access gateways without affecting the established session. In order to be able 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 this paper, we report experiment results related to an experimental distributed DC network. We present a way to manage a distributed DC control-plane linking the control-planes of two protocols already used in commercial networks (LISP and TRILL) at the cloud access network and at the intra-DC data-link layer. The unified control-plane is instrumental to synchronize routing and mapping information related to VMs, and improve cloud access latency.

The paper is organized as follows. In section II we give an overview of the state of the art. In section III we describe the proposed architecture. In section IV we detail the different modules of our agent. In section V we present experimental results. We conclude our work in section VI.

978-1-5090-0223-8/16/$31.00

II. Background

In this section, we overview the state of the art of protocols for distributed DC architectures, and motivate our approach.

A. Virtual Embedding Protocols for Distributed Data centers

Today, large monolithic DCs can be too complicated to deploy due to the increasing maintenance, logistical and environmental costs of rapidly aging applications and infrastructures. Moreover, the high demand for storage and computational resources can cause severe strains on the underlying hardware, leading to a deterioration of the quality of the user's experience. Hence, distributing DC becomes a solution pushing the cloud boundaries beyond the legacy geographical deployment limits.

To improve cloud network performance and resiliency, decentralized DCs provide a reliable infrastructure as a service increasing the path diversity and the availability to hosted applications. To this extent, several protocols have been implemented to develop advanced control plane features that support VM mobility and virtual network management such as Transparent Interconnection of Lots of Links (TRILL) at the Ethernet level and Locator/Identifier Separation Protocol (LISP) at the IP level, improving the experience of users accessing VMs that can be dynamically displaced as a function of network and system states [16].

1) Transparent Interconnection of Lots of Links (TRILL): TRILL is an Internet Engineering Task Force (IETF) standard. Whenever an update is generated by a topology or a link-state change, TRILL ensures new shortest paths which are computed and taken at the Ethernet level, eliminating the drawbacks of spanning tree protocols in terms of performance and routing efficiency [14].

Devices implementing TRILL are called Routing Bridge (RBridge). The TRILL data plane absolves the role of encapsulating incoming packets to a TRILL network toward 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 [6].

TRILL uses an adaptation of the Intermediate Systems to Intermediate Systems (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 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) Locator/ID Separation Protocol (LISP): LISP relies on the separation of the locator and the identification (in IP networks). They are designated by two IP addresses: the first one is a Routing Locator (RLOC) assigned to the edge router and used for forwarding packets across the Internet. The second address is used as an Endpoint Identification (EID) and is allocated to the endpoint device. Each EID-prefix is associated to a set of locators with different priorities and weights to pilot the routing. One of the main advantages of separating RLOCs from EIDs is to facilitate the mobility, which means that hosts can keep their EIDs while changing RLOCs [8].

When a LISP client wants to reach another LISP site, an edge tunneling router is solicited. This tunneling router is called xTR and has a double functionality: it is an Ingress Tunneling Router (ITR) when it encapsulates packets toward and the destination RLOC; it is an Egress Tunneling Router (ETR) when it decapsulates the packets received from another LISP site.

An EID-to-RLOCs mapping system is designed to support LISP routing. The mapping system is composed of a Map-Server (MS) and a Map-Resolver (MR). The MS learns the authoritative mapping system from the ETR to publish the mapping information in the EID-TO-RLOC database [9]. The MR receives the Map-Requests sent from the ITR and resolves them for the EID-to-RLOC mapping; to do this, the MR can use the LISP Delegated Database Tree (LISP DDT) protocol that works similarly to the DNS caching specification in order to retrieve the correct mapping [12].

B. Orchestration Challenges

In this context, TRILL and LISP protocols share the same philosophy in the method of associating an end-point identifier (Ethernet or IP address) to a routing locator (egress RBridge or egress tunneling router, respectively). This feature is very interesting when support for seamless VM migration is needed at the data-link and network layers. Since live VM migration (a VM is migrated while running and used by clients) and VM redundancy (a VM is kept synchronized at many virtualization servers where it can be restored in case of failure) schemes are available in recent hypervisors (also called Virtual Machine Managers, VMMs) [17], it is interesting to investigate how TRILL and LISP could interact to jointly managing VM mapping to virtualization servers at their corresponding segment, i.e., intra-DC fabric and extra-DC, in a distributed DC fabric scenario. Under the target scenario where a common TRILL domain bridges the distributed DCs of a same cloud fabric, a very interesting feature would be to export the shortest path metric used by TRILL-ISIS into the RLOC priority to allow cloud access traffic to be routed toward the RLOC that corresponds to the best intra-DC path. The TRILL-ISIS path cost toward a given destination VM can indeed change in the following target situations: (i) when an intra-DC link fail; (ii) when an intra-DC link state metric is modified manually, or automatically by traffic engineering toolbox; (iii) once a VM moves (i.e., migrates or is restored) from a virtualization server to another, from a DC site to another site.

The first and the third actions are likely to happen more often in a distributed DC fabric: a link failure between the DC entry point and the VM can interrupt an ongoing session with the user; a VM mobility can sometime decrease the quality of experience especially when the network distance between the user and the application gets larger upon VM mobility. Under such link state changes, we need to establish a control-plane linkage between the LISP and TRILL protocols in order to update LISP mapping system, and improve the quality-of-experience depending on the link and DC chosen by a LISP user [16].

Our reference scenario is depicted in Fig. 1, where two distant DCs use TRILL at the layer-2 intra-DC segment and LISP protocol at the layer-3 extra-DC segment. In this context, the challenge is to implement an agent that can transpose the mapping and routing (also called in the literature virtual network mapping) information from TRILL to LISP protocol, in order to offer the best network support for VMs moving across servers and DC sites. An important assumption justifying our approach, inspired by the reference distributed DC configuration the one of Alphalink[1], is that the inter-DC site latency dominates over the DC access latency, hence mapping over LISP routing locator priority the intra-DC TRILL-ISIS routing metric should lead to better performance.

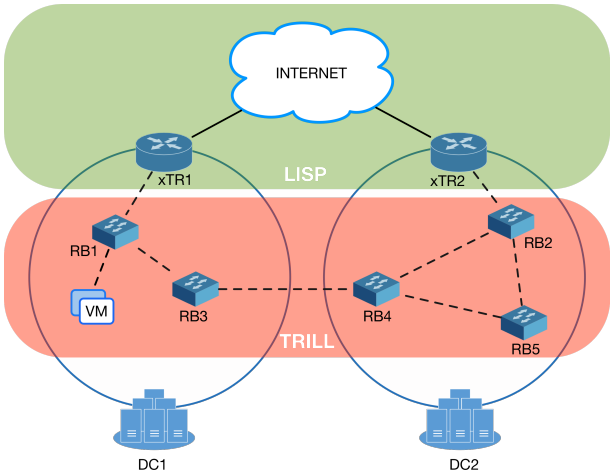

Fig. 1: Reference distributed DC protocol architecture.

III. Protocol Architecture

In this section, we describe the functional diagram of our solution at both the TRILL and LISP protocol levels.

[1]http://www.alphalink.fr

The reference deployment of TRILL, considered by many companies operating distributed DCs such as Alphalink, is such that TRILL is present in each virtualization server, and possibly also at the DC physical switch level (despite this is not strictly required in TRILL as there is no need in TRILL to have direct physical link between TRILL switches).

A. Functional Requirements

The migration or restoration of a VM for instance, 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 or a link state change (or path cost variation) occurs, to query the TRILL-ISIS control-plane table to determine the next hop RBridge, and to issue an update to the LISP control-plane. More precisely, the LISP RLOC priorities should be set equal or somehow proportional to the TRILL-ISIS path cost.

In order to support at best 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 Fig. 2.

MAC Mapping Table

MAC address	Destination RbridgeID
00-00-19-11-22-33	1
00-f1-e1-44-77-66	1
00-fe-04-77-88-99	2

+

IS-IS Forwarding Table

Rbridge ID	Next Hop Rbridge	IS-IS Metric
1	3	10
2	4	12

↓

MAC	Next Hop Rbridge	IS-IS metric
00-00-19-11-22-33	3	10
00-f1-e1-44-77-66	3	10
00-fe-04-77-88-99	4	12

Fig. 2: Correspondence between TRILL and ISIS tables.

In fact, the MAC mapping table generated by TRILL shows three VMs with different MAC addresses. The first two VMs are trying to reach the RBridge 3 and the last VM is trying to reach the RBridge 4. We suppose that these VMs are directly linked to the RBridge 1. The ISIS database provides us 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 RBridges IDs and the 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 ISIS/TRILL 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's cost linking of the migrated VM to the old and the new destination's xTR. The new generated message is sent to the MS in order to make the required updates to the LISP mapping system The signaling diagram in Fig. 3 represents the corresponding message exchanges.

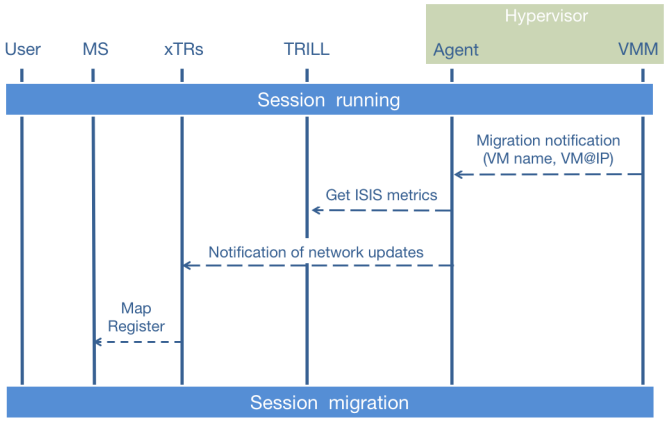

Fig. 3: LISP-TRILL Agent signaling steps triggered upon VM detection.

B. Possible solutions

From a technical perspective, two main approaches could be followed to meet the above requirements.

1) Centralized solution: A centralized solution can be based on a Software Defined Network (SDN) architecture where the SDN controller includes a LISP Mapping Service. For instance, this is the case of the OpenDayLight controller. In such a situation, the agent can be placed as a function of the mapping service and can ensure the notification process when a VM migration or restoration occurs. However, this solution can generate a high load from and towards the SDN controller, which can become a processing bottleneck.

2) Distributed solution: A distributed solution consists in having the agent, creating the glue between TRILL and LISP control-planes and sits at the hypervisor level. In such a way, the agent can be directly notified of a VM migration or an ISIS path cost variation change (e.g., due to network link failure), triggering fast updates to xTRs (by a form of xTR configuration API yet to be specified). When the agent detects a new VM, it determines the cost from the LISP router to the hypervisor and changes the mapping entry in the LISP router. Then, each xTR sends a Map-Register to the MS (i.e., a message needed to register one or many mapping entries for a same EID or EID prefix) with the appropriate EID and the new priority. Ideally, this latter operation requires a form of incremental Map-Register yet to be specified, in order to scale up with the high number of VMs and high mapping granularity for such scenarios.

IV. Agent Implementation

In this section we detail the technical features required for the agent implementation and its working states. We chose to implement the distributed solutions as it better answers the

reliability and retro-compatibility concerns of a distributed DC operator such as Alphalink.

A. Features

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

- it detects the migration of a VM, recovering its main information such as its name and its EID, or the change of the ISIS path cost due to network link state changes.
- 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 [15].

1) States: The process of the agent across its possible states and actions is represented in the functional diagram of Fig. 4.

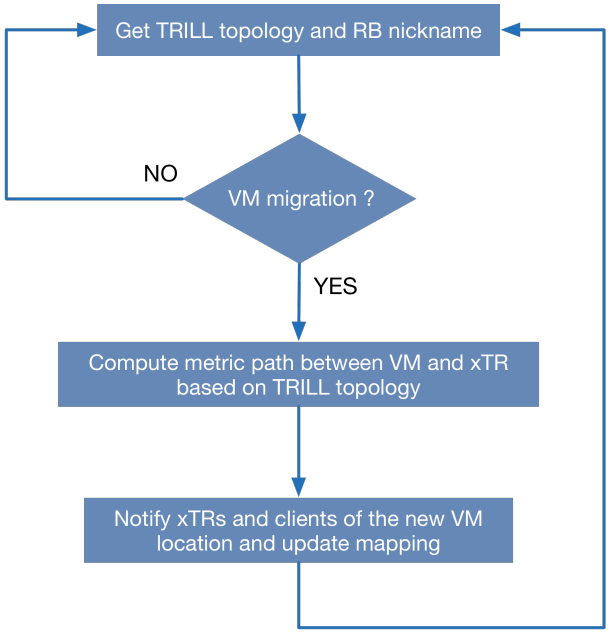

Fig. 4: Diagram of the agent process.

We distinguish four main modules:

a) Module 1: VM detection: several open-source cloud computing softwares are implemented to help administrators monitor cloud network resources. For instance, Openstack [3] is an open-source cloud computing platform that can manage a large number of storage, infrastructure and networking resources throughout a DC; Openstack can manage VM migration and relocation within hypervisors, with shared or dedicated storage, exposing to the administrator a rich API to monitor hypervisors and their attached VMs. Moreover, OpenStack can measure the performance of VM migration according to fitness parameters [13] and can provide VM status (running, stopped, paused, etc.), its addresses (MAC, IP) and the corresponding hypervisor to which it is connected.

In this context, we simulated these information related to the managed active VMs in the network and supplied by the Openstack administrator. The agent, located at the hypervisor level, periodically checks the status of VMs joining the hypervisor local network. It compares the current list of local VMs to an old saved list. Once the new entries are spotted, it indicates the name and the IP address of the migrated VM joining the new local hypervisor. Fig. 5 represents these VM detection tasks. Then, the agent saves the IP address of the detected VM, retrieves the MAC address of the xTR(s), and looks for the Egress RBridges that are directly linked to them in order to calculate the ISIS metric of the path separating the VM from the xTR(s).

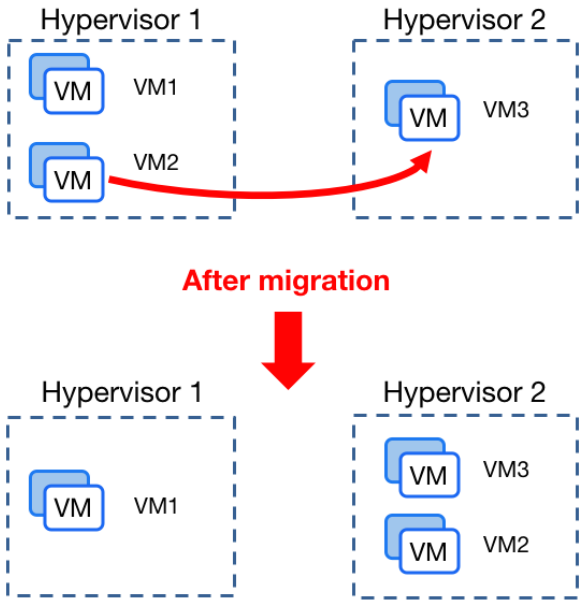

Fig. 5: VM detection process.

b) Module 2: identification of the destination RBridge: In order to find the MAC address of the destination RBridge, the agent solicits the TRILL node configuration and states, retrieving TRILL topology and nicknames of the active RBridges in the network. It can so get the MAC address of the corresponding xTR and the MAC address of the attached RBridges.

c) Module 3: ISIS path cost retrieval: At this stage, the agent retrieves the path cost associated to the identified RBridge from the ISIS routing table.

d) Module 4: mapping policy update: At last, the mapping entry can be changed accordingly to the retrieved ISIS path cost from the xTR(s) to the RBridge beyond which the VM has been detected. The agent has a way to reconfigure the xTR though a dedicated API to set the RLOC priority, triggering an explicit map-register signaling with the mapping system.

The agent periodically repeats the whole process in order to guarantee a continuous alignment between the TRILL and LISP control-planes.

We detail in the following the precise software tools we used to build the prototype.

2) Used tools:

a) TRILL and Quagga nodes: Alphalink maintains a fork of the TRILL Linux kernel implementation[2] developed by Gandi[3]. That implementation makes use of a modified version of Quagga [4], and the ISIS version therein, for the link-state signaling and routing table management. Our agent uses telnet to connect to Quagga in order to get the TRILL node configuration and retrieve ISIS information using a simple command line interface offered by the TRILL implementation.

b) Ruby programming language: We chose to develop the proposed agent in ruby language for the following reasons: it is an open source programming language that helps us getting a clear and concise code useful for production; it is an object-oriented language that can be used to generate flexible scripts for embedded systems; it owns an API that can easily write C extensions [5].

c) Libvirt library: It is a software library capable to handle the management of a set of VMs, under several hypervisors [7]. In particular, we used the Libvirt manager to manage the VM configuration and to test the different settings [1].

d) OpenLISP router: We used the OpenLISP development version [2], [11], maintained by LIP6, which includes specific control-plane features to support VM migration with LISP [15], [17].

V. Experimental evaluation

We conducted an experimental campaign on a real distributed DC testbed configuration, involving computing resources in the cities of Pornic and Paris, in France.

The emulated scenario is represented in Fig. 6. We use a simple topology with 4 RBriges, RB1, RB2, RB3 and RB4, a VM directly attached to RB3 that has to stay reachable via RB1. xTR1 represents the xTR of Pornic and xTR2 represents the xTR of Paris. The ISIS metric configuration is represented in Fig. 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 put in place a geographically distributed LISP test bed between Pornic DC (xTR1) and Paris DC (xTR2) to run the scenario described above. We connected a LISP user to the VM and then simulated a network impairment between RB1 and RB3. We compared our solution with the legacy situation where TRILL metrics are not mapped into LISP priorities. We then set the direct link between the VM and xTR1 to down. As a result, TRILL generates the new ISIS metric and redirects the traffic to the available link but with a higher ISIS metric value. Therefore, a LISP client located in Paris still sends its traffic using xTR1 through the new extra-DC path.
We then activated the agent and repeated the same scenario.

[2]http://gitlab.alphalink.fr
[3]https://github.com/Gandi/ktrill

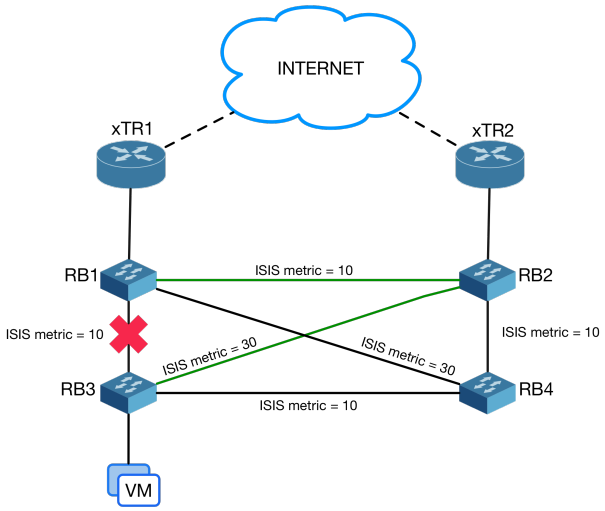

Fig. 6: Representation of a topology change scenario.

After being notified with the new calculated TRILL topology, the agent maps the TRILL metrics into LISP priorities. The user traffic is then switched to xTR2.

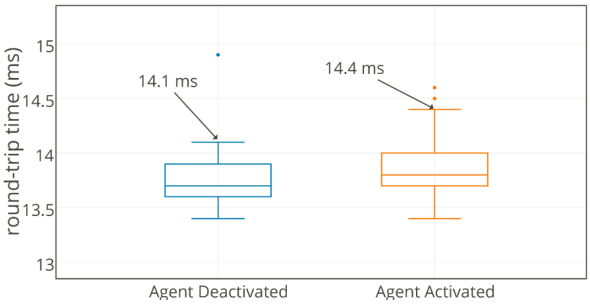

Fig. 7: RTT performance before network impairment.

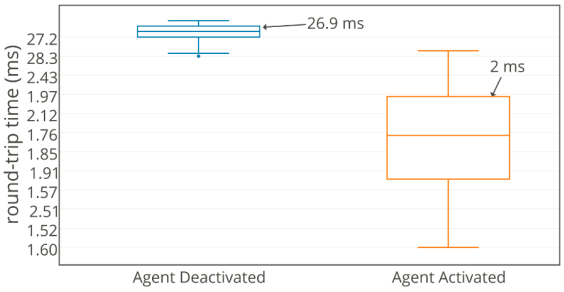

Fig. 8: RTT performance after network impairment.

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 links states: before and after network impairment respectively in (Fig. 7)(Fig. 8 - 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 network impairment, the maximum round-trip time is 14.1 ms for the legacy situation, while it is 14.4 ms for our solution. A similar gap exists also between the median and the other quartiles.

After network impairment, we can better appreciate the benefit of our approach that, for the given setting, offers at much lower round-trip-time.

VI. Conclusion

In this paper, we proposed a unified LISP-TRILL control-plane solution to provide optimized access to Cloud Infrastructure as A Service (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 the 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 testbed in France, testing a proof-of-concept prototype and experimentally proving the interest of our solution.

Acknowledgment

This work was supported by the ANR LISP-Lab (Grant No: ANR-13-INFR-0009), the Systematic FUI 15 RAVIR (http://www.ravir.io) and the PSPC/IA FED4PMR projects.

References

[1] Libvirt The virtualization API (online): http://www.libvirt.org/, Accessed: 2015-08-23.

[2] LIP6-LISP open source project:, https://github.com/lip6-lisp, Accessed: 2015-08-23.

[3] Openstack project (website):, https://www.openstack.org/, Accessed: 2015-08-23.

[4] Quagga project:, http://www.nongnu.org/quagga/, Accessed: 2015-07-25.

[5] Ruby, https://www.ruby-lang.org/en/documentation/, Accessed: 2015-07-16.

[6] A. Amamou, K. Haddadou, G. Pujolle. A trill-based multi-tenant data center network. *Computer Networks* 68:35–53, 2014.

[7] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehorster, A. Brinkmann. Non-intrusive virtualization management using libvirt. In Proc. of *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, March 2010.

[8] D. Farinacci, D. Lewis, D. Meyer, V. Fuller. The locator/id separation protocol (LISP), RFC 6830. January 2013.

[9] V. Fuller, D. Farinacci. Locator/id separation protocol (LISP) map-server interface, RFC 6833. January 2013.

[10] M. Gharbaoui, B. Martini, P. Castoldi. Anycast-based optimizations for inter-data-center interconnections [invited]. *Journal of Optical Communications and Networking* 4(11):B168–B178, 2012.

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

[12] A. Jain, D. Lewis, V. Ermagan, V. Fuller. LISP delegated database tree, internet-draft. April 15, 2015.

[13] R. Kashyap, S. Chaudhary, P.M. Jat. Virtual machine migration for back-end mashup application deployed on openstack environment. In Proc. of *Parallel, Distributed and Grid Computing (PDGC), 2014 Int. Conference on*, Dec 2014.

[14] R. Perlman. An algorithm for distributed computation of a spanning tree in an extended LAN. In Proc. of *ACM SIGCOMM Computer Communication Review* 15:44-53. ACM, 1985.

[15] D.C. Phung, S. Secci, D. Saucez, L. Iannone. The OpenLISP control plane architecture. *IEEE Network*, 28(2):34–40, March 2014.

[16] P. Raad, S. Secci, Chi-Dung Phung, and P. Gallard. PACAO: A protocol architecture for cloud access optimization in distributed data center fabrics. In Proc. of *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, April 2015.

[17] P. Raad, S. Secci, D.C. Phung, A. Cianfrani, P. Gallard, G. Pujolle. Achieving sub-second downtimes in large-scale virtual machine migrations with LISP. *IEEE Transactions on Network and Service Management* 11(2):133–143, June 2014.