Boosting Cloud Communications through a Crosslayer Multipath Protocol Architecture

Matthieu Coudron†, Stefano Secci†, Guido Maier‡, Guy Pujolle†, Achille Pattavina‡
†LIP6, UPMC, 4 place Jussieu 75005, Paris, France.
Email: firstname.lastname@lip6.fr
‡DEIB, Politecnico di Milano, Piazza Leonardo da Vinci 32 – 20133 Milano, Italy.
Email: firstname.lastname@polimi.it

***Abstract*— External reliability in data-center networking is today commonly reached via forms of provider multihoming, so as to guarantee higher service availability rates. In parallel, Cloud users also resort to multihoming via different device access interfaces (Wi-fi, 3G, Wired). Both practices add path diversity between Cloud users and servers, unusable with legacy communication protocols. To overcome this void, we present a holistic multipath communication architecture for Cloud access and inter-Cloud communications, and defend its possible implementation using three promising recent protocols functionally acting at three different communication layers: MPTCP, LISP and TRILL.**

***Index Terms*—Software defined networks, multipath routing, cloud computing, multihoming.**

I. Introduction

Multipath communications represent both a chance and a hassle for the current Internet. On the one hand, legacy Internet protocols have mainly been designed with a single active path paradigm in mind. On the other hand, multihoming practices at both endpoint and network levels can offer path diversity to data connections, potentially allowing effective end-to-end load-balancing and multipath communications [1]. Cloud networking is at the forefront of this trend. Indeed, it increases Cloud availability guarantee via different techniques : the external interconnection of data-centers (DCs) with multiple independent provider links, the deployment of intra-DC multipath layer-2 protocols and IP endpoint multihoming. In this paper, we tackle the challenge of establishing coordinated multipath communications in a Cloud environment composed of multihomed data-center networks and users.

We adopt a protocol interoperability perspective, aiming at increasing throughput and resiliency of Cloud communications, within and across administration domains. We present both stateless and stateful solutions to end-to-end path diversity management, involving novel protocols standardized in the last months: the Multipath Transport Control Protocol (MPTCP) [2], the Locator/Identifier Separation Protocol (LISP) [3] and the Transparent Interconnection of a Lot of Links (TRILL) protocol [4].

II. General Architecture

Our goal is to resort to multipathing to improve cloud access and inter-Cloud performance, more precisely to increase Cloud connections' throughput. Decreasing transfer times improves the user Quality of Experience, and boosts storage consolidation and virtual machine migration as well. As depicted in Figure 1, the envisioned network environment involves Cloud service users, potentially mobile, and data-center networks, with user-to-Cloud, intra-DC and inter-DC communications. Under this perspective, there are major challenges to address:

- How to send and receive data packets along different paths without disturbing applications?
- How to select disjoints paths in order to provide higher resiliency and to avoid bottlenecks?
- How to ensure packets really follow disjoint paths?

It is worth noting that using different paths in a uncoordinated way, TCP performance can be decreased rather than increased, namely because packet arrival disorder may trigger retransmission that may disturb the application workflow. Moreover, selecting Internet-wide end-to-end disjoint paths is commonly considered as an unrealistic dream, given the high heterogeneity and versatility of the Internet ecosystem. However, in the following we present rather simple functional blocks and protocol coordination mechanisms that are one step toward this goal, under realistic assumption and partially already available protocol functionalities.

In the following, MPTCP is considered as the transport layer protocol, as it is undoubtedly more scalable than other proposed alternatives and already deployable, even if it is deployed mostly at an experimental extent for the moment. MPTCP [2] is a TCP extension making use of several TCP subflows when possible to improve throughput and resiliency. It first asserts if the destination is MPTCP compliant (middleboxes such as firewalls might prevent the use of unknown TCP extensions), otherwise it falls back to legacy TCP. Once an MPTCP connection is established, endhosts can advertise their IPs, add or remove MPTCP subflows at anytime.

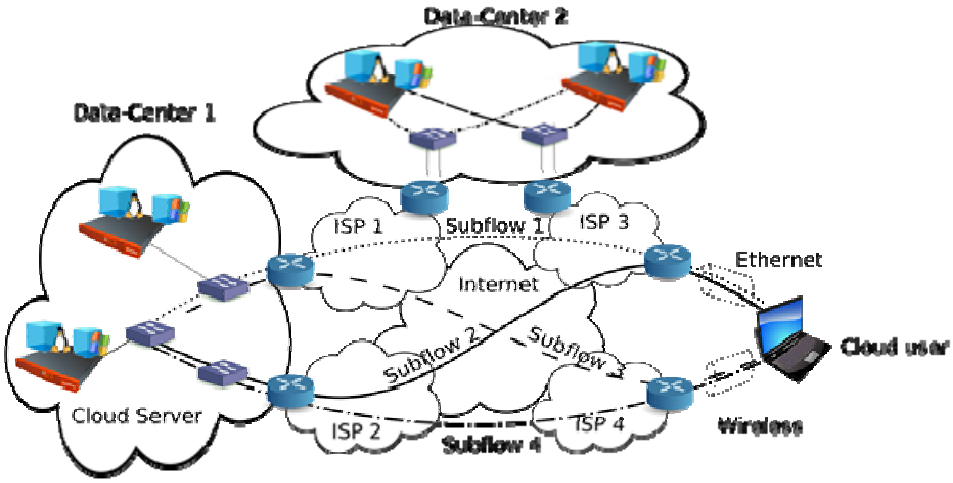

Figure 1 : Representation of the Cloud Networking Context

Basically a subflow could be defined as a TCP connection embedded in a more comprehensive TCP connection. MPTCP can mark a subflow as « backup only » to use it only if the other subflows stop transmitting. More interestingly, joint congestion control techniques using many subflows are documented in standardization documents. Performance gains are achievable if the round-trip-time (RTT) gap among subflows is not too high, so that packet disordering can be absorbed by TCP end-point buffering.

A. Cloud Network Elements

The Cloud fabric we envision should be sufficiently flexible to be extended and adapted to different technologies. The key role is taken by four node types:

- Virtualization Cloud servers, with MPTCP enabled at the hypervisor level.
- Cloud users, able to establish MPTCP connections even when single-homed.
- Border nodes, i.e., routers at the DC and user borders with its Internet Service Providers.
- Cloud Controllers, managing a DC network, enabling path discovery and establishment.

The implementation of MPTCP at the hypervisor level allows its scalable deployment: it somehow represents a TCP optimizer and it allows deploying MPTCP agnostic virtual machines. MPTCP can open multiple TCP subflows for a single TCP connection. These subflows differ with respect to their source or destination IP (if multihomed), or via the subflow TCP port (in case one server is singlehomed). Our proposition is an advanced yet simple protocol architecture that basically stitches MPTCP subflows to Ethernet- and IP-level paths, which are as much disjoint and RTT-similar as possible. The delivered network service improves resiliency and throughput, but comes at a cost: finding diverse and RTT-similar paths adds an overhead in processing time as well as in latency that may impede performance and scalability. Therefore, we foresee a special treatment only for flows for which the pros outweighs the cons, as described hereafter.

B. Functional blocks

A generic crosslayer multipath protocol architecture should implement the following blocks .

Flow Qualification Service: at the Controller or hypervisor level, it identifies which connection benefits from a multipath communication. The online classification ranks different metrics of a flow, e.g., jitter, latency, throughput, security, so that if needed we can sort flows according to policies. For instance, it appears appropriate to distinguish « elephant » flows (i.e., long-lived flows) from « mice » (i.e., short-lived) flows. The hypervisor triggers signaling for multipath communication for elephant flows only.

Flow Monitoring Service: at the Controller or hypervisor level, the state of either the network or the application might evolve during a communication, thus becoming obsolete. If a physical path becomes unreachable or underperforms, one may want to update the multipath configuration. With respect to RTT variations across subflows, the result of this service can allow virtually compensating the variations with artificial delays at the hypervisor software level.

Path Discovery Service: this service can be considered as a Traffic Engineering Database (TED) service; at the Controller or hypervisor level, it collects various information about the available path diversity, such as DC Link States, MTUs, load-balancers, DC and Internet topology, multipath capabilities, between the Cloud servers and users through border nodes . While DC-level discovery can be performed using existing operational tools, retrieving Internet path information can be difficult due to various factors (high versatility, unreliable knowledge of the topology, loose paths, etc.). In practice, the

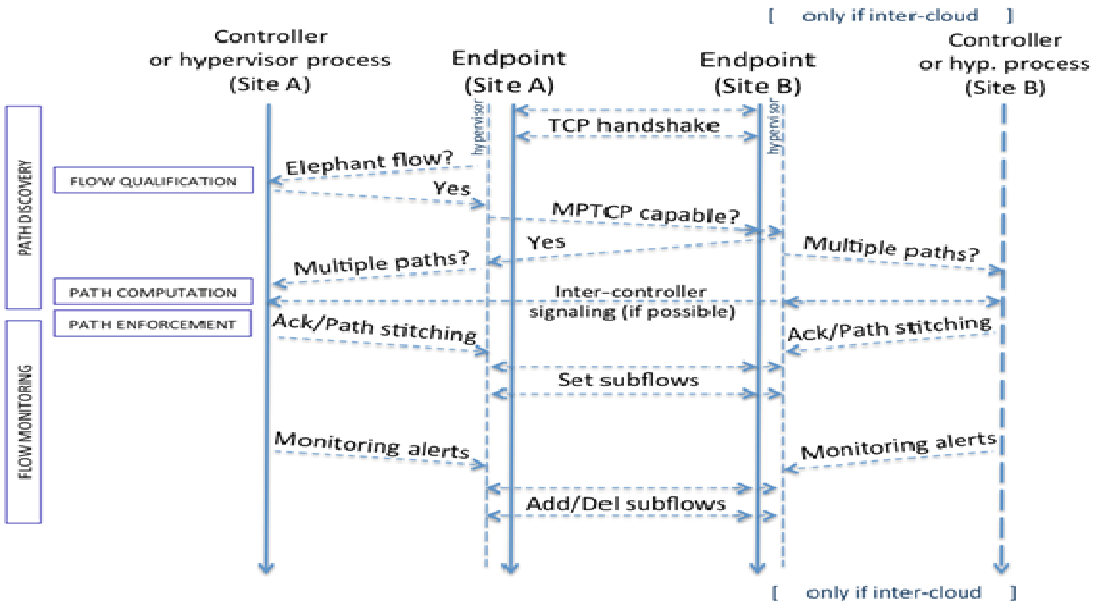

Figure 2. A signaling example in the case of extra-DC multipath communication

service is to be decomposed into an Intradomain Discovery Service (i.e., a controlled network scope, such as a DC network), and an Interdomain Discovery Service. One might also want to replicate the TED as a distributed database or cache, splitting it per layer or per subdomain, each using a pull or push discovery (these considerations are out of the scope of this document).

Path Computation Service: at the Controller or hypervisor level, this service is in charge of selecting paths to assign to flows when requested. The decision process takes into account constraints specified in the request (e.g., a jitter-sensitive path), and uses the TED to resolve matching paths and node capabilities.

Path Enforcement Service: once paths are computed, packets have to use these paths. This is possible through stateful and stateless modes:

- Stateful signaling: network path setup on per flow basis, e.g., using Software Defined Networking (SDN) or MPLS-based architectures.
- Stateful explicit forwarding: the source (i.e., the Cloud server's hypervisor, actively or passively via the Controller) lists in each packet the nodes the packet must pass through.
- Stateless forwarding: the source exploits network load-balancing algorithms and crafts packets' headers so that they follow foreseen paths [6].

C. *Multipath Communication Signaling*

Figure 2 highlights the signaling process and the associated functional blocks. For example, a TCP connection is established between two VMs in two different DCs in site A and site B. The hypervisor hosting the VM at site A detects the flow and queries the Flow Qualification process (could be a process running at the hypervisor, or an external controller). If the flow qualifies for a multipathed connection, the hypervisor acting as an MPTCP proxy verifies if endpoint B is MPTCP capable (or its hypervisor is); if yes, it queries the Controller for multipath capabilities to its destination. Controllers at the two sites may be able to collaborate in the computation of the paths. The Controller(s) compute(s) and enforce(s) the paths based on the information recovered by the Path Discovery Service (running ex-ante in a push mode or ex-post in a pull mode). In the stateful mode, intra-DC paths are virtually stitched with MPTCP subflows by the hypervisor, and at border nodes inter-DC (loose) paths are stitched with intra-DC paths. In the stateless mode, the hypervisor could be informed about the presence of load-balancers and possibly of the available path diversity at intra-DC and/or inter-DC segments. Data is eventually transferred using the multiple end-to-end paths. Guaranteeing a level of disjointness at the intra-DC and/or inter-DC level can allow boosting throughput, thus reducing transfer times. Appearance of network events such as node/link congestions/failures/additions could trigger respective hypervisors, causing subflow deletion or addition, possibly new path computation and enforcement.

In case the endpoint B is a user terminal, the right side of Figure 2 does not apply: there is no need for path stitching at site B. On top of that, if the user terminal is multihomed, the more access interfaces there are, the higher number of subflows can be opened. It is worth noting that neither the user terminal nor the DC nor the Cloud virtualization server need to be all multihomed: to open more than one subflow path, just one segment needs to have path diversity, where the segments are the intra-DC segment between Cloud servers and DC border nodes, the extra-DC segment between border nodes, and the end-to-end transport segment between terminal interfaces. ,

At each Cloud network segment, different protocols can act, hence interoperability between them is needed to propagate path diversity across segments. One solution could be having SDN protocols such as OpenFlow as ubiquitously as possible between source and destination, with however important scalability and reliability concerns. A reasonable alternative is to rely on three new distributed protocols recently defined to independently handling path diversity at each segment, with an adequate coordination as proposed hereafter.

.

III. Specific Implementation Using MPTCP, LISP and TRILL

In this section, we present one possible peculiar implementation of the previous architecture making use of three novel protocols: apart the already mentioned MPTCP [2] at the end-to-end transport segment, LISP [3] handles path diversity at the extra-DC segment between border nodes, and TRILL [4] can enable multipath Ethernet-layer communications within the DC. It is important to highlight that, since their standardization, all these protocols have been designed to be incrementally deployable in the existing Internet infrastructure.

Our intention by using these protocols is to address all the previously described use-cases as well as distributing path diversity across layers in a scalable and practical way. A key factor for this purpose is the multipathing ability they all offer (though optional in TRILL). These protocols do have also in common the functionality of mapping one upper layer logical point to many lower-layer logical points: one application port to many IP interfaces for MPTCP, one IP address to many IP routing locators for LISP, one MAC address to many Ethernet routing locators for TRILL. None of them are fully standardized, yet they have all been implemented to some degree: many vendors have started commercializing TRILL since a few months (e.g., Cisco, Huawei, Fujitsu); a TRILL OpenSolaris opensource version exists, and a Linux version is expected to be released closely. FreeBSD (OpenLISP[1]) and Linux (LISPmob[2]) versions of LISP are available; LISP is also already available in Cisco routers. Finally, a stable Linux version of MPTCP exists , adaptable to Android smartphones, so having it implemented at customer end-points and Cloud virtualization servers is not unrealistic. In the following, we first synthetically describe LISP and TRILL, and then discuss cross-layer coordination in the specific architecture. Finally, we present the results of partial experimentations.

A. Locator/Identifier Separation Protocol (LISP)

IP addresses assume today 2 functions : localization and identification of its owner. In LISP [3], each endpoint IP, named Endpoint Identifier (EID), is associated to one or many IP addresses of intermediate IP interfaces, named Routing LOCators (RLOC), typically supposed to be at border routers of the endpoint network. Upon reception of a packet from the local network to an outer EID, the border router acts as an Ingress Tunnel Router (ITR): it retrieves the EID-to-RLOC from a mapping system, and then it prepends to the packet a LISP header and an outer IP header with the RLOC as destination IP address. The receiving RLOC is an Egress Tunnel Router (ETR) that decapsulates the packet and forwards it to the destination EID. As the outer packet is a traditional IP packet, it can be routed on the legacy internet (though there might be Maximum Transfer Unit, MTU problems). If a site is not yet LISP compliant, the traffic might get encapsulated (or decapsulated) by a Proxy ITR (or a Proxy ETR). The usage of RLOC priorities and weights in the mapping system allows inbound traffic engineering, suggesting a best RLOC or an explicit load-balancing. An extension interesting for our architecture is the LISP Canonical Address Format (LCAF) [6] that, among other features, can allow enforcing explicit chains of RLOCs on the way toward the destination, hence boosting the traffic engineering capabilities in the extra-DC Internet segment.

B. Transparent Interconnection of Lot of Links (TRILL)

TRILL implements a logic close to LISP's at the Ethernet layer in order to solve switching tables scalability problems and improve network efficiency for DC environments. As LISP, TRILL uses data encapsulation with an outer standard header and a shim specific header, and it tunnels data units from an ingress node to an egress node, with a mapping system to update association of endpoint (MAC) addresses to network locator (MAC) addresses (this is a key feature as VM are migrated across DC racks, hence their location can be updated). On the other hand it differs from LISP since it integrates a multi-hop routing logic between ingress and egress nodes based on IS-IS (though LCAF somehow fills this gap). This is the reason why TRILL bridges are called RBridges (Routing Bridges). It is incrementally deployable: standard Ethernet segments can sit between RBridges, with each RBridge terminating a spanning tree instance. Multipath communications are therefore possible between RBridges at the Ethernet layer. It is worth mentioning that alternative Ethernet routing protocols exist, namely the IEEE 802.1aq Shortest Path Bridging (SPB), Layer-2 Label Switched Path (L2LSP), Provider Backbone Bridge with Traffic Engineering (PBB-TE) and (at some extent) OpenFlow; all could allow Ethernet multipathing too, yet both require a complete deployment at all bridges and especially for this reason they are, for the moment, often considered as less scalable and versatile.

IV. Specific Architecture

In an heterogeneous Cloud environment with MPTCP, LISP and TRILL, the Cloud network has MPTCP enabled at both endpoints, at the DC side either directly in the VM or (more reasonably) as a virtual middle-box in the hypervisor. TRILL

[1] http://www.openlisp.org (data-plane); http://www.lisp.ipv6.lip6.fr (control-plane)

[2] http://www.lispmob.org

is deployed between the Cloud virtualization servers and the DC border nodes, noting that an increasing interest in standardization activities is given to the implementation of RBridges also as virtual bridges at the hypervisor level. LISP is implemented at DC border IP routers, and it can be implemented by user endpoints (see [7]) handling each access interface as an RLOC, hence complementing MPTCP with an inbound traffic engineering control-plane when both are enabled in the user endpoint.

Therefore, from the one hand (DC side) TRILL and MPTCP coexist at the hypervisor level, and from the other hand (access side) LISP and MPTCP coexist at the user mobile node. It is also worth noting that, since a few weeks, the LISP data-plane is implemented in the well-known virtual bridge called OpenVSwitch[3], hence pushing down to the hypervisor LISP encapsulation/decapsulation and letting the three protocols coexist in the same DC node. While MPTCP capability at the endpoints is mandatory in order to enable multipath communications to the application layer, our architecture does not require LISP or TRILL to be implemented concurrently; a partial deployment with at least one of them is needed to associate loosely diverse path to subflows, yet both together would allow reaching higher performance. Finally, about the Cloud Controller, it can be based on classical Network Management Systems and the like, or on an SDN open controller accessed for instance via OpenFlow, or on a Path Computation Element (PCE[4]) [7] generalized for non-MPLS environments, or a mix of these technologies.

In the following, for each component described in the Functional Blocks section of the general architecture, we associate a specific solution based on MPTCP, LISP, TRILL and peculiar functionalities to close the gap between theory and practice.

Flow Qualification: the process can be implemented either as co-located with the hypervisor or as external server. The advantage of the latter case is to exploit network-level information and offload the hypervisor by excessive signaling, and also to implement advanced qualification criteria, for example based on destination's information and white/black lists. However, if simple criteria are used, e.g., a binary classification as elephant or mice flow, the implementation in the hypervisor does not need external information and appears as more appropriate. In such a case, a possible solution could be Mahout [8] implemented at the hypervisor level; basically, whether the socket fills up quicker than its emission rate, it marks the flow as elephant. In this case, we pursue the process, and proceed as explained in Figure 2.

Flow Monitoring: it is left to the MPTCP congestion control mechanism using per-subflow windowing. Upon failure of a TCP subflow (i.e. window size inferior to a threshold during a certain period of time), the Cloud server may request a new path to the Controller in order to replace this subflow with a new subflow following a new path. This request may notify the deficient path to the discovery service which may update the states about that specific path. As the MPTCP congestion control mechanism is per subflow, it is important that a subflow keeps using paths of equivalent quality; indeed, a change to a path of lesser quality may decrease the window, yet it takes time to recover the original window value.

Path Discovery: it can be decomposed into two subservices with different constraints.

- *intra-DC discovery service*: in DC environments, the assignment of VMs to servers should be orchestrated by a Cloud Management System that knows where each VM is placed. TRILL support this through an oracle-like system (non mandatory) called « directory system » [9], which returns the destination RBridge associated with a MAC, instead of resorting to ARP flooding, thus reducing L2 broadcast traffic. Associated to this information, the IS-IS substratum readily allows TRILL campus topology synchronization at RBridges, which can be enriched with TE metrics adopting the ISIS-TE extensions defined for IP networks [10] to TRILL-based Ethernet networks.
- *Interdomain discovery service*: MPTCP discovery is quite straightforward; if the distant host does not answer with the MPTCP capable option, then the connection falls back to legacy TCP. If both hosts are MPTCP compliant, they can either advertise their different IP interfaces to the other host, or directly try to open new subflows with these EIDs. As far as LISP is concerned, the hypervisor or the Controller can be easily enabled to query mapping servers to retrieve the RLOCs of the local site and the destination, along with LISP load-sharing information. Moreover, we may also try to rank Internet paths between xTRs, for example adding a TED companion to the LISP mapping information, using providers' services such as [11] or PCE-based [7], knowing however that Internet path information accuracy can be low.

The Controller could manage a single TED built merging TE metrics related to TRILL Ethernet routing tables, LISP RLOC metrics and inter-domain path metrics, as a separate database regularly and on-demand pulling data from TRILL, LISP and external databases.

Path Computation: this service will use the information gathered by the Path Discovery Service to compute a number of paths according to different constraints, which might be passed in the request (e.g., latency, jitter, throughput, bottleneck bandwidth). In the case of inter-cloud communications, the PCE communication Protocol (PCEP) [7] could be used to allow distributed computation between DCs, including also the possibility to involve the PCEs of external ISPs. The extra-DC path computation between LISP endpoints can include the possibility of involving

[3] http://www.openvswitch.org
[4] http://pce.ida-cns-group.net (opensource PCE implementation)

Reencapsulating Tunner Routers (RTR) between the ITR and the ETR [12] if LCAF is enabled.

Path Enforcement: both stateful and stateless methods are conceivable.

- *Stateful mode*: the computed path is enforced in a source-routing fashion. The way this can be implemented in TRILL and LISP does not rely on end-to-end reservation as in MPLS networks, but it implies the hypervisor is able to craft TRILL and LISP packets. This is proposed by [13] for TRILL: the hypervisor, called « TRILL smart endnode » directly queries the TRILL directory and encapsulates the packet with a TRILL header to lighten the load on the next RBridge. Even if not proposed elsewhere, with the above mentioned implementation of the LISP data-plane in virtual bridges such as OpenVSwitch, the hypervisor could become a « LISP smart endnode » as well. Conversely, explicit path prepending is described for LISP with RTRs and LCAF [6][12], allowing the enforcement of multihop xTR chains, and is currently not offered by TRILL. On the other hand, TRILL supports extensions, so we can imagine a similar feature implemented at RBridges.
- *Stateless mode*: with TRILL, we cannot use VLANs anymore to force a physical path, since any VLAN can get encapsulated in shared transport VLANs by the TRILL campus. However, with multipath load-balancing enabled, one can carefully compute the TRILL header's Time To Live (TTL) field so that packets with the same TTL follow the same path as of the result of hashing functions used in load-balancing, similarly to how described in [4]. This technique allows controllable per-flow load-balancing and prevents packet disorder. Similarly, the TTL in the IP header can also be tuned to enforce extra-DC egress load-balancing at ITRs.

To sum up, the stateful method adds LISP and TRILL headers overhead to packets, but in a controlled environment such as a DC, the MTU should not be a problem. As for the stateless method, enough LISP and TRILL nodes need to share the same hashing algorithm to make it interesting, which is a reasonable assumption. However middleboxes may also change the TTL value, thus nullifying the effect. Finally, in this specific architecture, hypervisors can implement all the services, but the heaviest ones such as the Path Discovery and the Path Computation Services that shall be taken on by the Controller.

To sum up, the stateful method adds LISP and TRILL headers overhead to packets, but in a controlled environment such as a DC, the MTU should not be a problem. As for the stateless method, enough LISP and TRILL nodes need to share the same hashing algorithm to make it interesting, which is a reasonable assumption. However middleboxes may also change the TTL value, thus nullifying the effect. Finally, in this specific architecture, hypervisors can implement all the services, but the heaviest ones such as the Path Discovery and the Path Computation Services that shall be taken on by the Controller.

V. CONCLUSION

Multipathed communications still remain a complex networking research field due to the necessary coordination between network layers and protocols. The architecture we propose in this paper unifies the different control planes thanks to the controller knowledge and coordinated routing mechanisms. Multipathed communications are becoming essential to augment Cloud communications; for instance, very recent experiments have shown that significant throughput could be achieved either locally - a local MPTCP throughput of 51.8Gbit/s between 2 servers containing each 3 dual-port 10 Gig NICs has recently been proven[5] – or on long distances - an intercontinental testbed [14] achieved a throughput of 15Gbits/s from Geneva to Salt Lake City. Our cross-layer multipath architecture can potentially overcome these values so as to further push network innovation at Internet edges. Indeed, coordination between the MPTCP, TRILL and LISP protocols, splitting flows at the transport, network and Ethernet layers, can allow carefully selecting communication paths, while controlling the computational load, and guaranteeing a semi-distributed reliable nature to the communication environment.

ACKNOWLEDGMENT

This article was partially supported by the NU@GE project (http://www.nuage-france.fr), funded by the French «Investissement d'avenir» research programme, and the FUI 15 RAVIR (Réseaux d'Accès Virtualisés au Cloud) project.

REFERENCES

[1] A. Akella et al., On the Performance Benefits of Multihoming Route Control. IEEE/ACM Transactions on Networking, Vol. 16, No. 1, pp : 91-104, 2008.
[2] A. Ford, C. Raiciu, M. Handley, O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses, RFC 6824, January 2013.
[3] D. Farinacci, V. Fuller, D. Meyer, D. Lewis. The Locator/ID Separation Protocol (LISP), RFC 6830, January 2013.
[4] R. Perlman, D. Eastlake 3rd, D. Dutt, S. Gai, A. Ghanwani. Routing Bridges (RBridges): Base Protocol Specification, RFC6325, January 2011
[5] G. Detal et al., Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks, Computer Networks (in press).
[6] D. Farinacci, D. Meyer, J. Snijders. LISP Canonical Address Format (LCAF), draft-ietf-lisp-lcaf-02, March 2013.
[7] A. Farrel, J-P. Vasseur, J. Ash. A Path Computation Element (PCE)-Based Architecture. RFC 4655. August 2006.
[8] A. R. Curtis, W. Kim, P. Yalagandula. Mahout: Low-Overhead Datacenter Traffic Management using End-Host-Based Elephant Detection, in Proc. of IEEE INFOCOM 2011
[9] L. Dunbar, D. Eastlake, R. Perlman, I. Gashinsky. TRILL Edge Directory Assistance Framework. draft-ietf-trill-directory-framework-04, Feb. 2013.
[10] H. Smit, T. Li, IS-IS Extensions for Traffic Engineering (TE), RCF 3784, June 2004.

[5] http://multipath-tcp.org/pmwiki.php?n=Main.50Gbps

[11] D. Saucez, B. Donnet, O. Bonaventure, Implementation and preliminary evaluation of an ISP-driven informed path selection, in Proc. of ACM CoNEXT 2007.
[12] D. Farinacci, P. Lahiri, M. Kowal, LISP Traffic Engineering Use-Cases, draft-farinacci-lisp-te-02, Jan. 2013.
[13] Radia Perlman et al., TRILL Smart Endnodes, draft-perlman-trill-smart-endnodes-01, Jan. 2013.
[14] R. van der Pol, M. Bredel, A. Barczyk , Experiences with MPTCP in an intercontinental multipathed OpenFlow network in Proc. Of SC201