ns-3 Implementation of LISP and LISP-MN

Yue Li*, Luigi Iannone*, Benoit Donnet†
∗ Telecom ParisTech – France
† Université de Liège, Montefiore Institute – Belgium

***Abstract*—The *Locator/Identifier Separation Protocol* (LISP) reconstructs the current IP addressing space to improve the scalability and no interrupt mobility issues. LISP Mobile Node (LISP-MN) is based on the basic LISP functionality to provide seamless mobility across networks. The basic LISP architecture is deployed on LISP Beta Network and LISP-Lab platform to offer the researchers a realistic experimental environment, but both do not support LISP-MN. Some simulation models with LISP extensions are implemented on various simulators, but are unfortunately not open source. Providing a free and flexible simulation model with the basic LISP architecture as well as the extensions so to help researchers quickly test new LISP behaviors motivates our work. This paper introduces the implementation of the basic LISP architecture model and LISP-MN in ns-3. It also provides the evaluation results in mobility scenario to validate the model and shows that, when the current proposal of LISP-MN is behind a LISP-site, it has a very high delay during the handover procedure.**

***Keywords*—LISP, mobility, LISP-MN, ns-3, simulation.**

I. Introduction

The *Locator/Identifier Separation Protocol* (LISP) [1] is initially proposed to solve the scalability and flexibility issues of current Internet architecture, since it decouples the *Routing Locators* (RLOCs, i.e., an attachment point in the Internet topology) and *Endpoint Identifiers* (EIDs, i.e., a communicating end-point). This allows the BGP routing tables in the Internet core to only announce the globally routable RLOCs whereas EIDs are only locally used within the LISP-sites. LISP is under standardization at the IETF for about ten years and, with time, more and more advantages are found such as: seamless mobility at terminal or in the Data Center **[BD: ref here?]**, IPv6 transition and traffic engineering [2]. In particular, the continuous communication without interruption during the handover of terminals becomes a hot topic recently **[BD: ref here?]**.

To test the various features of LISP, two LISP testbeds are built: LISP Beta Network and LISP-Lab platform **[BD: refs for both?]**. Unfortunately, both of them only implement the basic LISP architecture and do not support mobility functionality. Some platform-specific LISP implementations, such as OpenLISP [3], Open Overlay Router (OOR) [4], and Cisco's implementation [5], can provide realistic LISP evaluations but lack the required flexibility for testing new and advanced LISP features. Further, a simulating environment for LISP would be valuable for researches as it would allow them to test, in a controlled and reproductible environment, the tomorrow LISP.**[BD: better but should probably be rewritten a little bit.]** On one hand, LISP has already been implemented in ns-3 [6], a widely used open-source simulator by academic researchers and educational. However, this implementation does not support mobility. On the other hand, LISP and mobility features are already implemented in OMNET++ [7]. However, the source code is not available online. To promote the development of LISP and as ns-3 gradually shows the potential to take the momentum in network simulator domain [8], it highlights the importance to implement LISP and its mobility extensions under ns-3 and motivates our work. **[BD: last sentence unclear]**

In this paper, we introduce the implementation of basic LISP functions and mobility features on ns-3 by both modifying the existent ns-3 modules and integrating new ones. Our implementation is validated through a mobility scenario in which a terminal changes its attachment points while keeping the communication up with the remote node. All the handover procedures are transparent to the terminal. However, the delay to receive the packets is high.

[BD: you mention that the Omnet implementation is not open-source. What about yours? If it is open-source,please provide a link in the paper to the repo.]

The rest of the paper is organized as follows: Sec. II reviews LISP architecture, highlighting the LISP mobility mechanism; Sec. III analyzes the design and implementation of our prototype, and afterwards, Sec. IV presents preliminary evaluation results of our implementations. Sec. V concludes this paper by summarizing its main achievements and discussing potential future works.

II. LISP Overview

A. Basic Architecture

LISP splits the current IP addressing space into two subspaces. One called EID indicating the host identifier is locally used within a LISP-site. The other one, called RLOC, represents the attachment point in the topology, i.e., the border router, and is globally routable between LISP-sites on the Internet core. The binding between EID and RLOC is called *mapping information* and is stored in the *Mapping Distribution System* (MDS) consisting of *Map Resolver* (MR) and *Map Server* (MS) [9]. The border router is in charge of searching for the mapping information either in its own Cache or in the MDS. It then encapsulates the conventional IP packets into LISP packets and forwards them out of the LISP-site on the Internet core. This router is called *Ingress Tunnel Router* (ITR). On the other hand, routers receiving the LISP packets and decapsulating them into traditional packets for the LISP-site behind them are named *Egress Tunnel Router* (ETR). One usually refers to both of them as *xTR*.

B. LISP Mobile Node

The LISP Mobile Node (LISP-MN) implements a subset of the standard xTR functionality [10] supporting a node itself fast roaming and being discovered in an efficient and scalable manner **[BD: unclear]**. It can interact with the MR to get the mapping information associated to the remote host **[BD: which remote host?]**. When a LISP-MN resides in a LISP-site, it is assigned an EID taken from the site's EID-prefix as its RLOC, called *Local RLOC* (*LRLOC*) **[BD: I think I understand the sentence but this is unclear]**. Thus, LISP-MN stores not only its permanent unique EID but also the Local RLOC and registers this mapping information to MS. The conventional IP packets produced by the LISP-MN are encapsulated on itself using the permanent EID as the inner source address and Local RLOC as the outer source address The encapsulated LISP packets are forwarded to xTR and encapsulated again by the method of basic encapsulation of xTR. The procedure of packets being encapsulated two times are called *double encapsulation*. **[BD: this 1st encapsulation is unclear to me. The EDI and LRLOC are used to forward the packet to the xTR. Then, the xTR encapsulates towards the distant sites. How this xTR knows the mapping of the distant site? It is unclear from this paragraph]**.

C. Mapping Cache Update Mechanisms

When the ETRs change their LISP databases **[BD: never heard about the databases before]**, the only way that remote ITR can get the updated mapping information is to re-request the mapping. Thus, Solicit-Map-Request (SMR) is proposed to get the latest mapping. **[BD: the broken English here makes the sentence unclear. SMR cannot be "proposed".]**

Soliciting a Map-Request is used by ETRs to tell remote ITRs to update the mappings they have cached in the Control Plane [1]. When the mappings in the Database of ETR change, the ETR sends the Map-Requests with the SMR bit set (called SMR message) for each RLOC in its Cache. A remote ITR that receives the SMR message will send a Map-Request to the MR. During the sending SMR message and receiving the new Map-Reply, the ITR continues to use the mapping information previously cached, and that may cause the packet loss.

[Bd: globally, this last subsection is unclear.]

III. Simulation Model

ns-3 [11] is a popular and free discrete-event network simulator for networking research. To be closer to the real implementation (in a real Operating System), easily include C-based implementation codes, ease debugging and reduce the cost on maintaining in a long term, C++ is prioritized to be the unique programming language for ns-3 **[BD: I personally don't think C++ is easy to debug... Is it really important to explain why C++ is used for coding in ns-3? I'm not sure]**. Besides, ns-3 offers the possibility to visualize the simulation instance so to allow the users to visually confirm the packets flow as they expect.

A. Modifications to ns-3

Our implementation **[BD: should be open source. Remind here the URL towards the repo]** is under ns-3.26 and based on LISP [12] and LISP-MN standards [13]. The main classes are shown in Fig. 1 in form of UML diagram. The blank blocks refer to the classes that we added into ns-3, while darker blocks are classes already in ns-3. As a design choice, we implement LISP/LISP-MN functionalities by modifying and extending the *internet* module of ns-3, instead of creating a new independent module. The justification of this design is that LISP/LISP-MN and legacy internet module have an interdependent relationship. However, this kind of mutually dependent relationship between modules is not supported by ns-3. Inspired by the OpenLISP design [14], the Data Plane implementation is in "kernel space" (i.e., ns-3 *TCP/IP stack*) and the Control Plane is implemented in "user space" (i.e., ns-3 *Application*). The communication between LISP Data and Control Plane is ensured through a dedicated socket (i.e., *LispMappingSocket*) that inherits from the ns-3 *Socket* class. It should be noted that our implementation only supports IPv4 at time of writing. The IPv6 support is still under construction.

1) Implementation of LISP Data Plane: A LISP-compatible node (terminal or router) should be capable of determining whether a packet should be passed to LISP-related procedure and retrieving the associated mapping information if necessary. To this end, a new class called *LispOverIp* and its extended classes (refer to Fig. 1) are added to ns-3 *internet* module. This class is in charge of checking whether necessary LISP-related operations (`NeedEncapsulation()`, `NeedDecapsulation()`) must be done, and of encapsulating conventional IP packets (i.e., `LispOutput()`) as well as decapsulating LISP packets(`LispInput()`). It also contains a smart pointer to the LISP database and LISP cache. Both data structures (store the EID-RLOC mapping information) are represented by the class *SimpleMapTable* that inherits from *MapTable*. The inheritance mechanism allows other users to implement their own implementation of LISP database and cache. To support LISP functionalities, the *Ipv4L3Protcol*, which is the IP layer implementation in ns-3, contains one *LispOverIp* object and *Ipv4L3Protcol*'s packet transmission and reception procedures are accordingly adapted.

To process outgoing packets, the adapted `send()` in *Ipv4L3Protcol* first verifies whether the *LispOverIp* object is present. If yes, some checks are then conducted to determine that this packet should be processed by `LispOutput()` (to encapsulate the packets) or by conventional packet transmission routine. For example, if both source and destination IP addresses of this packet belong to the same network, the LISP-related process (e.g., encapsulation) is skipped and this packet is processed as in a non-LISP network. Otherwise, EID-RLOC mapping information is searched from LISP cache and LISP database on LISP-MN node. In case of a cache miss, the packet is dropped and `SendNotifyMessage()` in *LispOverIp* notifies, via a *LispMappingSocket* socket, the *LispEtrItrApplication* that runs on LISP-MN node. At the reception of the

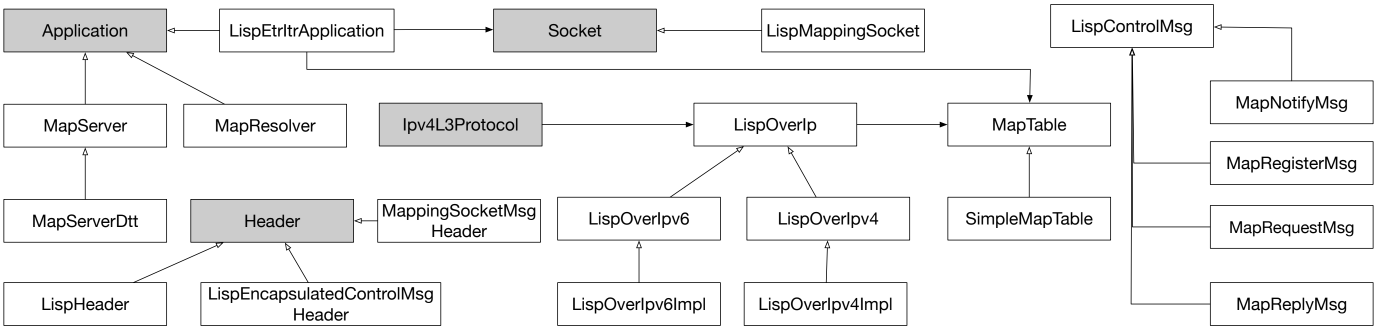

Fig. 1. UML diagram of LISP/LISP-MN implementation. The solid arrow refers to a composition relation, while the blank one refers to a inheritance relation.

cache miss event from LISP Data Plane (i.e., *LispOverIp* object), *LispEtrItrApplication* initiates a Map-Request message to LISP mapping system. At the reception of the Map-Reply, the received EID-RLOC mapping is inserted into LISP cache. It should be noted that as an implementation choice, before the reception of Map-Reply message, all transmitted packets with the required RLOC as destination are dropped. One can also design a buffer to queue these packets and resend them once the required mapping information is received via Map-Reply. The advantage of such an implementation is to reduce the packet loss rate.

For an incoming packet, if the destination of this packet is the node itself, the packet is processed by `LocalDelivery()` in *Ipv4L3Protocol*. Before passing to transport layer, `LocalDelivery()` checks if the packet should be decapsulated. If yes, it is passed to `LispInput()`, in which the packet is decapsulated and reinjected in the IP stack. If the received packet destination is not this node, the packet is processed by patched `IpForward()` method. This packet may be ended up with LISP encapsulation procedure.

2) Implementation of LISP Control Plane: The implementation of LISP Control Plane at least should provide ITR/ETR, MR, and MS. In practice, ETR and ITR functionalities are usually placed on a same router. In our implementation, they are included in the *LispEtrItrApplication* class. A ns-3 node that runs *LispEtrItrApplication* is a LISP-compatible router. It should be able to communicate with *LispOverIp* on the same node (e.g., inform about cache miss) and other LISP-compatible routers (e.g., Map-Request/Map-Reply). To support LISP-MN feature, *LispEtrItrApplication* also communicates with DHCP client application. For example, once a LISP-MN obtains an IP address from the DHCP server, *LispEtrItrApplication* receives the corresponding EID-RLOC mapping and sends a Map-Register message [13].

A node that runs a *MapServer* application is the MS in a LISP-supported network. This class maintains a LISP database to store the EID-RLOC mapping information, learned from Map-Register message at the initialization stage. In current implementation, the role of MR is to receive the Map-Request message from xTR and forward it to the MS.

3) Integration of TUN net interface card: To support mobility, LISP-MN actually can be regarded as a small LISP-Site, in which xTR functionalities and DHCP service are implemented, as well as configured address of MR and MS. As a LISP-MN node, it has a static permanent EID and dynamic RLOC assigned by the DHCP server. To differentiate with conventional RLOC of xTR interface, such kind of RLOC is referred to as the local RLOC (LRLOC). Different from conventional LISP node, at least two net interface cards (NIC) are installed into LISP-MN. One is *WifiNetDevice*, the other is a TUN type card. The DHCP client application runs on LISP-MN's *WifiNetDevice* and thus the LRLOC is allocated to this card. The permanent EID is assigned to *VirtualNetDevice* net card. We modify the node's routing table so that each packet's inner header contains IP address on *TunNetDevice* and outer header contains IP address of *WifiNetDevice* as source address.

4) Integration of DHCP: To support mobility within conventional LISP node, a modified DHCP client application is integrated into ns-3 node. To be compatible with LISP functionality, DHCP client application is modified. Once the DHCP client receives an allocated IP address (i.e., LRLOC), it notifies the *LispEtrItrApplication* (i.e., LISP Control Plane) by sending a dedicated message that contains the EID-LRLOC mapping. *LispEtrItrApplication* is in charge of populating the received mapping entry into LISP database. During the mobility process, when wireless link is down, the DHCP client flushes the LISP-MN database and populates the database again at the reception of a new LRLOC.

IV. Evaluation

We validate our LISP/LISP-MN implementation by conducting a simulation **[BD: it is unclear to me how, at 1st glance, running a simulation can validate a simulator (while you mention that in the Introduction)?]** and providing a preliminary performance evaluation of handover delay. The topology used for the simulation is shown in Fig. 2. **[BD: you could redraw this figure so that text appears larger]**

A. Simulation Setup

In our simulation, a LISP-MN with permanent EID 172.16.0.1 is initially placed in network 10.1.1.0/24. An *echo* application on LISP-MN sends one packet per second to a remote stationary node CN with EID 10.3.3.2, and the LISP-MN moves into network 10.1.7.0/24 at speed of $7.07m/s$

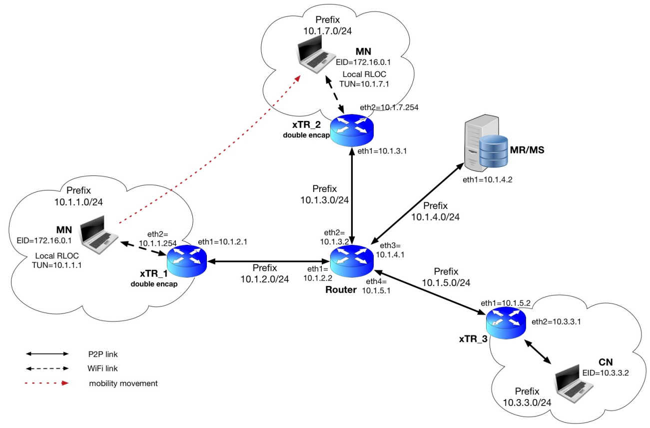

Fig. 2. LISP mobility simulation scenario for double encapsulation

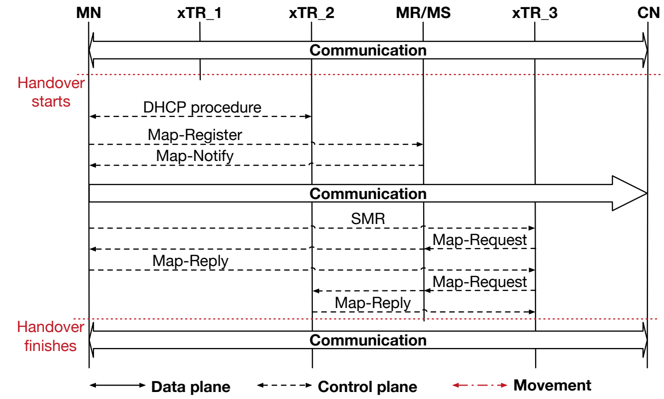

Fig. 3. Schema for LISP-MN mobility

[BD: why that speed?]. The distance between xTR_1 and xTR_2 is $170m$. LISP-MN node uses Wi-Fi to connect to xTR_1. At a certain moment during the move, the Wi-Fi link between LISP-MN and xTR_1 goes down, triggering so the handover procedure. Afterwards, LISP-MN connects to xTR_2 and reestablishes the communication with CN node. The total simulation time is set to $45s$ and the DHCP procedure delay is set to $1s$. We conduct many times of simulations **[BD: how many? Did you average the results? If so, what is your confidence in the average (i.e., you have to compute for each average the confidence interval)?]** with the various beacon interval of Wi-Fi channel in the range of $0.05s$ to $2s$ **[BD: why?]**.

B. Results

As shown in Fig. 3, when MN sends the packets to CN during the simulation, it needs double encapsulation and the packet flow sequences are as follows. The traditional IP packets with EID 172.16.0.1 as source address and EID 10.3.3.2 as destination address are encapsulated by adding Local RLOC 10.1.1.1 as outer source address and RLOC 10.1.5.2 as outer destination address after MN querying the mapping information to MR. The LISP packets are encapsulated and forwarded to xTR_1. The latter gets the mapping information from MR and encapsulates the packets again by adding the RLOC 10.1.2.1 as the outer source address and RLOC 10.1.5.2 as the outer destination address, then sends the packets on Internet core. The xTR_3 needs decapsulate the packets twice after it receiving and verifying the packets, and finally sends to CN. Once LISP-MN lost its Wi-Fi connection with xTR_1, it needs a DHCP procedure (consisting of DHCP Discover, DHCP Offer, DHCP Request and DHCP ACK) with xTR_2 to get a new LRLOC 10.1.7.1 and then triggers LISP SMR to xTR_3. After xTR_3 getting new mapping information of <10.1.7.1, 10.1.3.1> and <172.16.0.1, 10.1.7.1>, the connection between LISP-MN and CN is re-established again.

The overall handover delay in this paper is defined at the moment that LISP-MN sending DHCP Discover message to xTR_2 and end up with xTR_3 receiving the last Map-Reply from xTR_2 **[BD: unclear]**. Precisely, it is made of three parts: the Wi-Fi association delay, the DHCP related delay, and LISP SMR delay:

$$D_{overall} = D_{Wi-Fi}(BI) + D_{DHCP} + D_{SMR}.$$

where D is the delay, BI is Beacon Interval, subscriptions $Wi-Fi$, $DHCP$ and SMR respectively refers to Wi-Fi association, DHCP procedure and LISP SMR.

After several runs **[BD: How much?]** of the simulation, we observe that the overall handover delay changes by the various beacon intervals, in particular the Wi-Fi association delay depends on the different beacon intervals, whereas LISP SMR procedure always cost around $3s$. To get the lower bound of overall handover delay, we can ignore the Wi-Fi association delay when the beacon interval is $500ms$, and the latency due to DHCP procedure is always $1s$. Thus, adopting LISP-MN to conduct the host-based mobility takes at least $4s$. Compared to current most stable solution for host-based IP mobility management MIPv6, which latency including L2 and L3 in a real Wi-Fi testbed is around $3.68s$ [15], LISP-MN has a higher delay caused by the double encapsulation mechanism introduced by LISP-MN behind LISP-Site.

During handover, CN can successfully receive packets from LISP-MN right after DHCP procedure being accomplished, but LISP-MN cannot receive the packets from CN until LISP SMR procedure is also finished. Thus, during DHCP procedure, all bi-directional transmitted packets are lost. To improve the performance, [16] proposes a network-level LISP-MN solution, but has not validated their proposals neither in simulation nor in testbed. Our ns-3 implementation can be used to realize them.

[BD: I'm a little bit puzzled here. I'm not sure what you really evaluate in the sense that you rather provide a description of what happens during each simulation run (i.e., message flow). Do you have some

quantification results (for instance the latency variation over something)?]

V. Conclusion and future work

As a promising technology for the future Internet architecture, LISP attracts more and more attention **[BD: ref?]**. There exist some LISP implementations, but they do not support LISP-MN or they are proprietary. Further, although measurements on LISP-testbeds can provide real time performance, due to the complicated topological structure, it is somewhat like a black box test which hinders us to find the exact explanation for some results. This highlights the importance to have an open source simulator for LISP in particular to support LISP-MN functionality. In this paper, we present our implementation for LISP/LISP-MN within ns-3, since the latter is a largely accepted simulator in networking research. The simulation results show that our implementation works well, and reveal the current LISP-MN proposal with a double encapsulation that has an high level delay during handover procedure. Our simulator can be a perfect choice to test the improvements of LISP-MN.

There are two possible directions to support IP mobility in LISP: host-based (i.e. LISP-MN) and network-based (i.e., xTR) mobility. We can compare the performance between LISP double encapsulation described in this paper with only host supporting LISP and only router supporting LISP leveraging our proposed simulator. As Map-Versioning [17] is another Mapping Cache update mechanism, we can also compare the performance between it and SMR that we present in this paper by our simulator.

[BD: this is a (very) short paper. I would suggest to shorten the conclusion]

Acknowledgment

Research work presented in this paper is supported by the French Research Agency under ANR-13-INFR-0009 LISP-Lab Project (www.lisp-lab.org) and benefited support from NewNet@Paris, Cisco's Chair "Networks for the Future" at Telecom ParisTech (http://newnet.telecom-paristech.fr). Any opinions, findings or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of partners of the Chair.

References

[1] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The locator/ID separation protocol (LISP)," RFC 6830, Internet Engineering Task Force, January 2013.

[2] D. Saucez, L. Iannone, A. Cabellos, and F. Coras, "Locator/ID separation protocol (LISP) impact," RFC 7834, Internet Engineering Task Force, April 2016.

[3] D. Phung, S. Secci, D. Saucez, and L. Iannone, "The OpenLISP control plane architecture," *IEEE Network*, vol. 28, pp. 34–40, March/April 2014.

[4] "Lispmob (website)." https://www.openoverlayrouter.org/lispmob/.

[5] "Cisco lisp implementation (website)." https://www.cisco.com/c/en/us/products/ios-nx-os-software/locator-id-separation-protocol-lisp/index.html.

[6] D. Lezama, E. Grampin, and R. Gagliano, "Implementation of LISP evaluation tools," in *Proc. Network Operations and Management Symposium*, October 2009.

[7] D. Klein, M. Hartmann, M. Hoefling, and M. Menth, "Integration of LISP and LIS-MN into INET," in *Proc. 5th International Conference on Simulation Tools and Techniques (ICSCT)*, March 2012.

[8] M. K. Rana, B. Sardar, s. MAndal, and D. Saha, "Implementation and performance evaluation of a mobile IPv6 (MIPv6) simulation model for ns-3," *Simulation Modelling Practice and Theory*, vol. 72, pp. 1–22, March 2017.

[9] V. Fuller and et al., "Locator/ID separation protocol (LISP) map-server interface," RFC 6833, Internet Engineering Task Force, January 2013.

[10] D. Farinacci, D. Lewis, D. Meyer, and C. White, "Lisp mobile node draft-ietf-lisp-mn-00," Internet-Draft, Internet Engineering Task Force, 2017.

[11] "ns-3." See https://www.nsnam.org/.

[12] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, and A. Cabellos, "The locator/ID separation protocol (LISP)," Internet Draft (Work in Progress) draft-ietf-lisp-rfc6830bis-05, Internet Engineering Task Force, August 2017.

[13] D. Farinacci, D. Lewis, D. Meyer, and C. White, "LISP mobile node," Internet Draft (Work in Progress) draft-meyer-lisp-mn-16, Internet Engineering Task Force, December 2016.

[14] D. Saucez, L. Iannone, and O. Bonaventure, "OpenLISP: An open source implementation of the locator/ID separation protocol." ACM SIGCOMM Demo Session, August 2009.

[15] V. Vassiliou and Z. Zinonos, "An analysis of the handover latency components in mobile IPv6," *Journal of Internet Engineering*, vol. 3, pp. 230–240, December 2009.

[16] Z. Tang, Y. Zhou, W. Deng, and B. Wang, "LISP-HNM: Integrated fast host and network mobility control in LISP networks," in *Proc. 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2017.

[17] L. S. P. L. Map-Versioning, "Iannone, l. and saucez, d. and bonaventure, o.," RFC 6834, Internet Engineering Task Force, January 2013.