Implementation of LISP/MN under ns-3

Yue Li, Luigi Iannone
Telecom ParisTech, Paris, France
yue.li@telecom-paristech.fr
luigi.iannone@telecom-paristech.fr

Benoit Donnet, Lionel Agbodjan
Université de Liège, Montefiore Institute, Belgium
benoit.donnet@uliege.be
lionel.agbodjan@gmail.com

ABSTRACT

The *Locator/Identifier Separation Protocol* (LISP), due to its map-and-encap approach, can bring benefits to mobility. LISP Mobile Node (LISP-MN) is based on the basic LISP functionality to provide the terminal mobility across networks. Assessing the LISP mobility and improving its performance are of paramount importance. However, there exist no open source simulator supporting LISP. Thus, we fill this gap by implementing the basic LISP function as well as LISP-MN on ns-3. In this paper, we describe how these implementations are realized in details.

KEYWORDS

LISP, LISP-MN, ns-3, simulation

1 INTRODUCTION

Aiming at addressing the scalability issue of the Internet Architecture, the Locator/Identifier Separation Protocol (LISP) has been proposed and under standardization at the IETF [1]. The main idea of LISP is the separation of IP addressing space into two sub-spaces: *Endpoint IDentifiers* (EIDs) and *Routing LOCators* (RLOCs). The packets respectively use EIDs or RLOCs according as they are routed within the stub networks or in the core Internet. The inter-domain communications need an additional map-and-encap operation, i.e., need to encapsulate packets using EIDs into packets using RLOCs, hence mapping EID to RLOCs. The mapping system (MDS), which is composed by Map-Resolver (MR) and Map-Server (MS) is in charge of mapping. The xTR, which is the combination of Ingress Tunnel Router (ITR) and Egress Tunnel Router (ETR) is used to complete the encap/dencap operations.

As LISP leverages a map-and-encap mechanism, it can permit the seamless mobility of mobile terminal or VMs in Data Center. In addition, a mobile node implementing LISP functionality is called LISP-MN [2]. It has 2 IP addresses: a permanent EID and a dynamic local RLOC (LRLOC) assigned by its attached router. It can directly communicate with MDS to get the mapping information of the remote host. It is also able to change its attachment point during the communication so to achieve the mobility through the networks without interruption. Thus, the evaluation of LISP mobility and the comparison with other mobility mechanisms become essential.

Although there exist few simulators supporting LISP, they are not open source. It hinders other researchers to modify or adapt the simulator with respect to their own research purposes. Thus we implement basic LISP functions as well as LISP-MN (both of them are denoted as LISP/MN) on ns-3 to facilitate the others to test their new proposals and provide the feedback to move the LISP technology forward.

WNS3 2018, June 2018, NITK Surathkal, Mangalore, India
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 LISP/MN IMPLEMENTATION ON NS-3

Our implementation (see Fig. 1) respects LISP RFC 6830 [1] and LISP mobility standards [3]. Without creating a new ns-3 module, we implement LISP/MN functionalities by modifying and extending two already existing modules of ns-3: *internet* and *internet-apps*. Inspire by OpenLISP [4], our implementation consists of two main parts: LISP Data Plane and LISP Control Plane. The communication between LISP Data and Control Plane is achieved via a dedicated socket (i.e. *LispMappingSocket*) that inherits from ns-3 *Socket* class.

2.1 Implementation of LISP Data Plane

The implementation of LISP Data Plane mainly consists of *LispOverIp* and *MapTable* classes and their subclasses, along side with some auxiliary classes (e.g. *LispHeader*). In addition, to support LISP functionalites, *Ipv4L3Protcol*'s packet transmission, reception, forward and delivery procedures are accordingly adapted.

2.1.1 LISP database and cache. Both LISP database and cache are modelized by the same class *MapTable* that stores and manages mapping information. This class provides CRUD (Create, Retrieve, Update, Delete) operations for mappings. Each mapping in LISP database/cache is an instance of *MapEntry*. For the sake of flexibility, the class *MapTable* is an abstract class. The CRUD methods are implemented in its subclass *BasicMapTable*. The mapping search operation is a straightforward iteration over LISP database/cache. It is possible that for other users to provide their own LISP database and cache implementation by extending *MapTable* class.

2.1.2 Implementation of LISP encapsulation and decapsulation. To integrate LISP/MN into conventional Internet protocol stack, one key technical difficulty is that *Ipv4L3Protcol* should be able to determine when passing a packet being processed to LISP-related procedure and how to retrieve the associated mapping information. To this end, a new class called *LispOverIp* and its extended classes are added to ns-3 *internet* module. This class is in charge of checking whether it is necessary to do LISP-related operations (*NeedEncapsulation()*, *NeedDecapsulation()*), and encapsulating conventional IP packets (i.e., *LispOutput()*) as well as decapsulating LISP packets(*LispInput()*). It contains a smart pointerpointing to the LISP database and LISP cache (e.g. *MapTable*) on which executes mapping search.

2.2 Implementation of LISP Control Plane

The implementation of LISP Control Plane covers xTR, MR and MS. xTR is included into class *LispEtrItrApplication*. The functionalities of MR and MS are respectively implemented by class *MapResolver* and *MapServer*. The LISP Control Plane messages are represented by the derived classes of *LispControlMsg*. In addition, to communicate with LISP Data Plane, a socket class *LispMappingSocket* is proposed.

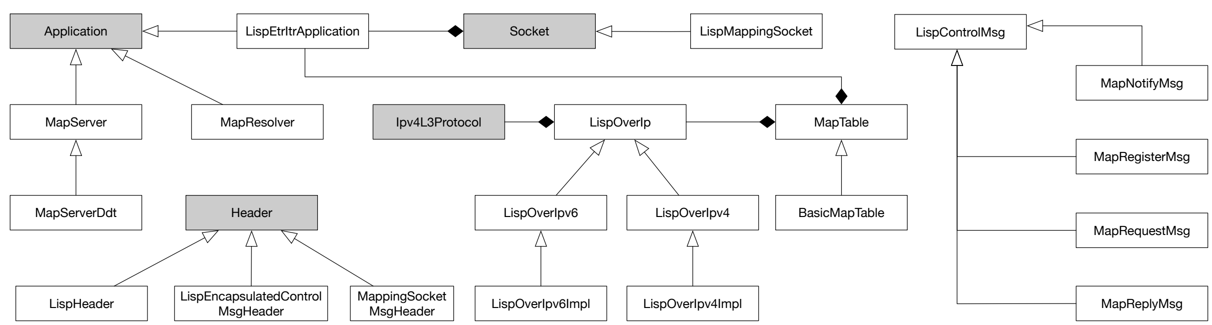

Figure 1: UML diagram of LISP/MN implementation. The darker blocks are classes already in ns-3, while the white blocks are classes added into ns-3.

2.2.1 Implementation of xTR functionalities. 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 cache missing event) and other LISP-compatible routers (e.g. Map-Request/Map-Reply). When destination RLOC is not found in the cache of xTR for a processed packet, the cache miss event occurs and LISP Data Plane (e.g. *LispOverIp*) notifies *LispEtrItrApplication* on the same node of this event. Once reception of cache missing event from LISP Data Plane (i.e. *LispOverIp* object), *LispEtrItrApplication* initiates a Map-Request message to LISP mapping system. Once reception of Map-Reply, the received EID-to-RLOC mapping is inserted into LISP cache. In case of reception of Map-Request, *LispEtrItrApplication* executes a database look up on *MapTable* and generates the corresponding Map-Reply containing EID-to-RLOC mapping. When xTR application starts or LISP database on a node has information update, xTR application sends a Map-Register message to MS and waits for a Map-Notify message. In case of LISP database update, xTR sends a SMR (Solicit-Map-Request) message to all xTR whose RLOC is present in its cache.

2.2.2 Implementation of MR/MS. MapServer class provides MS functionalities. It maintains a LISP database managing EID-to-RLOC mapping information. Once reception of Map-Register message, it updates LISP database and sends a Map-Notify message as response if necessary. When receiving a Map-Request message, if the queried EID-to-RLOC mapping is found within its database, MS forwards this request to the corresponding xTR otherwise sends a Negative Map-Reply message to the querying xTR. In current implementation, the role of MR is to receive the Map-Request message from xTR and forward it to the MS.

2.3 Modification of DHCP client

We adapted ns-3 DHCP client application to support LISP/MN. The modified DHCP client is able to communicate with *LispEtrItrApplication* running on the same node. For example, after IPv4 address assignment, DHCP client checks if the LRLOC is changed. If LRLOC is changed, DHCP client notifies the *LispEtrItrApplication* by sending a dedicated message that contains the EID-LRLOC mapping. *LispEtrItrApplication* is in charge of populating the received mapping entry into LISP database and sending a Map-Register message to MS.

2.4 Integration of TUN net interface card

As a LISP-MN, it has a static permanent EID and dynamic RLOC assigned by the DHCP server. We use the solution based on TUN device. In our implementation, at least two NICs should be installed into the MN. One is normal NIC such as *WifiNetDevice*. The DHCP client application runs on this kind of card and thus the LRLOC is allocated to this card. The other is a TUN type card. The TUN NIC is a virtual card which should actually invoke *Send()* of another real NIC. The permanent EID is assigned to TUN device.

Recall that after the DHCP procedure, the node will be configured a default gateway provided by the DHCP server. Routing table of LISP-MN are modified so that the packets from application layer always use EID as the source IP address of inner IP header.

3 CONCLUSION

Lack of open source LISP simulators, we implement the basic LISP functions and LISP-MN on ns-3. At the moment of writing, we are working on the simulation evaluation of the different mobility scenarios so to present the mobility performance. This work currently only supports IPv4 and the IPv6 support is still in process.

Acknowledgments: Research work presented in this paper 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 and et al. 2013. *The Locator/ID Separation Protocol (LISP).* RFC 6830. Internet Engineering Task Force.
[2] Dino Farinacci, Darrel Lewis, Dave Meyer, and Chris White. 2017. *LISP Mobile Node, draft-ietf-lisp-mn-01.* Technical Report. institution=IETF Internet draft, Apr. 2017.[Online]. Available: https://tools.ietf.org/html/draft-ietf-lisp-mn-01.
[3] D Meyer, D Lewis, D Farinacci, and C White. 2016. LISP mobile node. *draft-meyer-lisp-mn-16. txt, Internet Engineering Task Force* (2016).
[4] Damien Saucez, Luigi Iannone, Olivier Bonaventure, et al. 2009. OpenLISP: An open source implementation of the locator/ID separation protocol. *ACM SIGCOMM Demos Session* (2009), 1–2.