Integration of LISP and LISP-MN into INET*

Dominik Klein
University of Wuerzburg, Germany
dklein@informatik.uni-wuerzburg.de

Michael Hoefling
University of Tuebingen, Germany
hoefling@informatik.uni-tuebingen.de

Matthias Hartmann
University of Wuerzburg, Germany
hartmann@informatik.uni-wuerzburg.de

Michael Menth
University of Tuebingen, Germany
menth@informatik.uni-tuebingen.de

ABSTRACT

The Locator/Identifier Separation Protocol (LISP) is a new naming and addressing architecture which is currently standardized in the IETF and which is deemed to improve the scalability and flexibility of the current routing architecture. LISP mobile node (LISP-MN) is an extension to the basic LISP architecture and enables mobile nodes to roam into LISP and non-LISP domains. The basic LISP architecture is currently deployed in a beta-network which can be used to test the protocol behavior on a smaller scale. However, a realistic simulation model for the LISP architecture and its various extensions is still missing. Such a simulation model could be used by researchers to quickly test new extensions on a larger scale for different load and network scenarios.

In this paper, we describe the implementation of our model of the LISP architecture and its various extensions in the INET framework for OMNeT++. We present performance results to show the correctness of our model. As a first application, we used the simulation model to assess proposed improvements to LISP-MN and to verify our proposed NAT traversal mechanism for LISP-MN.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol verification, Routing protocols*; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Design, Experimentation, Standardization, Verification

Keywords

OMNeT++, INET, LISP architecture, LISP-MS, LISP-INT, LISP-MN, Mobility and NAT traversal

*This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (support code 01 BK 0800, G-Lab, http://www.german-lab.de/). The authors alone are responsible for the content of the paper.

OMNeT++ 2012 March 23, Desenzano, Italy.

1. INTRODUCTION

The current interdomain routing faces scalability and flexibility problems. More and more edge networks want to do multi-homing, traffic engineering, and provider changes without renumbering their equipment. This requires provider-independent addresses which adds more entries to the rapidly growing BGP routing tables [11]. The maximum allowed IP prefix length is limited by ISPs so that companies with a relatively small provider-aggregateable address space are still very restricted in using such advanced techniques. These are drivers for a more scalable and flexible Internet addressing and routing.

The currently favored solution is the separation of global and local routing and addressing in edge networks [14]. Communication sessions with other nodes are established using endpoint identifiers (EIDs), which might also be used for local routing. EIDs are not advertised in global BGP routing. Therefore, a globally routable locator (RLOC) is added to each packet before sending it to another domain over the Internet. This architecture decouples the combined identification and location functions of today's IP addresses but requires a mapping system to store and distribute the mapping from endpoint identifier to globally routable locator.

The Internet Engineering Task Force (IETF) currently works on LISP [3] as experimental standard. It implements routing separation and fits well into today's Internet routing concept. Interworking between LISP domains and the legacy Internet is supported [9]. Furthermore, an architecture for the integration of mobile nodes (MNs), LISP-MN, is also proposed [4]. It allows multi-homing for mobile nodes and does not need home and foreign agents like in Mobile IPv4 [15] so that triangle routing can be avoided to some extent. To further avoid breaking existing transport connections after roaming events, mobile nodes may use different mechanisms to tell their communication partners about their new location. In principle, these update mechanisms ensure transport connection survivability regardless of the involved domains. However, the mobility extension itself does not support that mobile nodes roam into non-LISP domains behind a NAT box while maintaining their existing connections. To solve this problem, we developed a NAT traversal mechanism for LISP mobile nodes [7].

In this paper, we present the integration of LISP into the INET framework for OMNeT++. We implemented LISP, the LISP-MN architecture with its different update mechanisms, and our NAT traversal mechanism. Our goal was to verify the interoperability of the different LISP protocol mechanisms, and to test the transport connection survivability in various communication scenarios including non-LISP domains behind a NAT box.

The paper is organized as follows. In the next section, we briefly describe the implemented LISP architecture parts, give an introduction to LISP-MN, and explain our NAT traversal mechanism.

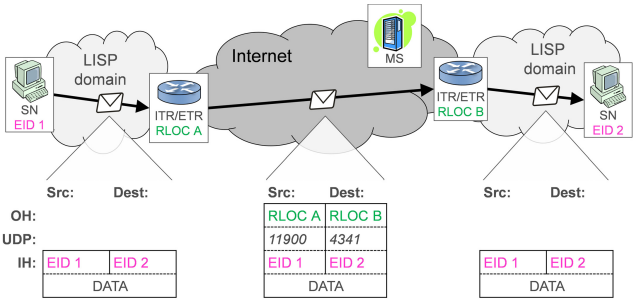


Figure 1: Packet flow sequence with LISP.

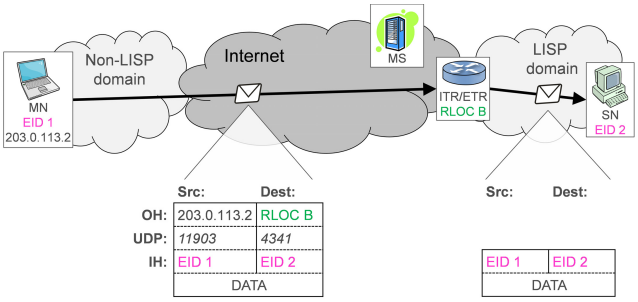


Figure 2: Packet flow sequence with LISP-MN.

Section 3 describes our simulation model in detail and shows the necessary changes to the INET framework. In Section 4, we present an evaluation of the handover performance of the LISP-MN architecture. Finally, Section 5 concludes this work.

2. LISP OVERVIEW

In this section, we give a short overview of the implemented LISP architecture parts and its most important extensions. The focus of our implementation is the mobility architecture LISP-MN and hence, we describe this extension in more detail.

2.1 Basic Architecture

LISP separates the current IP address range into two different subsets. LISP domains are edge networks that are connected via LISP gateways to the core of the Internet, where RLOCs are used to forward packets. EIDs identify end-hosts on a global scale and are used to forward packets locally inside LISP domains. Communications between LISP nodes in different domains require tunneling between the different LISP gateways. The gateways either act as ingress tunnel router (ITR) or as egress tunnel router (ETR). ITRs tunnel packets to other LISP gateways which then act as ETRs.

Figure 1 shows a packet flow sequence for the communication between two LISP clients located in different LISP domains. ITR A receives packets addressed to EID 2 from an end-host in its own LISP domain. It keeps the inner header (IH) untouched and adds a UDP header addressed to the default LISP data port *4341* and an outer LISP header (OH) with its RLOC (RLOC A) as source and RLOC B as destination address so that the packets are globally routable. This procedure requires a mapping lookup to learn the appropriate RLOC (RLOC B) for the destination EID (EID 2).

2.2 Mapping Service

The mapping service is the central component of the LISP architecture as it provides the required mapping from EID to RLOC. LISP does not mandate a specific mapping service but instead introduces map servers (MSs) and map resolvers (MRs) [5]. These two entities form an interface which facilitates the operation of LISP with different mapping systems. ETRs register the EID-to-RLOC mapping for all attached LISP nodes at their associated map server on the default LISP signaling port *4342*. If requested by the ETR in the map-register message, the map server acknowledges the registration with a map-notify message.

ITRs query map resolvers for the RLOC of a specific EID. The map resolver sends a map-request message which is forwarded via the mapping service to the authoritative map server. The map server responds with a map-reply message which contains the valid locator set for the queried EID. Map resolvers and map servers can either be deployed in separate nodes or inside ITRs and ETRs.

In our model, each ITR has an integrated map resolver and a mapping cache to store frequently used mappings. This reduces the load on the mapping system and significantly speeds up communication between LISP domains. Every time a mapping is used, its cache timer is reset. Thus, less frequently used cache entries are automatically purged from the cache.

2.3 LISP Interworking

Normal IP nodes usually resolve the DNS name of a LISP node into an EID and use it as destination address. However, EIDs are not globally routable and thus border routers of non-LISP domains discard the packets because of missing forwarding entries for EIDs. To solve this problem, LISP-IW [9] proposes additional middle boxes called proxy-ITRs (PITRs) and proxy-ETRs (PETR). Proxy-ITRs are located outside edge domains and advertise highly aggregated EID prefixes into BGP. This way, packets addressed to EIDs become globally routable and are forwarded to one of the proxy-ITRs. Proxy-ITRs perform the same traffic processing as ordinary ITRs, i.e., they query the mapping system for an RLOC of the destination EID and encapsulate packets towards the returned RLOC.

In the reverse direction, LISP packets destined to non-LISP nodes are not encapsulated by ITRs and the EID remains in the source address field of outgoing packets. Since the EID is not part of the upstream provider's address range, such packets might be dropped when the provider performs source address filtering to ensure that outgoing packets carry only addresses from its own address range. In this case, packets destined to non-LISP nodes are encapsulated by the ITR and tunneled to a proxy-ETR. The proxy-ETR decapsulates the packet and sends it to the destination node in the non-LISP Internet. This way, LISP bypasses the source address filtering of upstream providers.

2.4 LISP Mobile Node

Mobile nodes possess an upgraded stack and act as light-weight LISP domains. They implement ITR/ETR functionality and are configured with the address of a map server that controls the EID-to-RLOC mappings for the mobile node. Mobile nodes register their currently valid locator at their configured map server and refresh this information by sending periodic map register messages. Like ETRs, they may request the map server to acknowledge the map-register message by sending a map-notify message. Mobile nodes also implement map resolver functionality and send signaling traffic to their configured map server without encapsulation. In contrast, data traffic is always encapsulated. Thus, the mobile node requires a proxy-ETR for communication with non-LISP nodes. The map server in this case also serves as proxy-ETR.

Figure 2 shows an example where a mobile node roams into a non-LISP domain and initiates a communication with a remote LISP stationary node (SN) in a LISP domain. The mobile node receives the globally routable non-LISP care-of-address *203.0.113.2*

upon roaming into the new domain. It registers this address as RLOC along with its own EID 1 at its associated map server. This address can be obtained from the map server by ITR B as destination locator upon sending a packet to the mobile node. In [13] we analyzed the encapsulation and forwarding structure of LISP-MN and suggested some improvements.

2.5 Mapping Cache Update Mechanisms

If the mapping of a LISP stationary or mobile node changes due to a certain event, e.g., roaming into a different domain, the cached mappings in ITRs or proxy-ITRs of communication partners need to be updated. This is necessary so that ongoing connections can survive the roaming event. In [3, 4], different mechanisms are proposed. In our model, the following mechanisms are implemented. We explain them by means of a mobile node which roams into another domain.

2.5.1 Solicit Map Request

The solicit map request (SMR) mechanism uses a special map-request message which is sent from the mobile node to ETRs of the most recent communication partners. After a roaming event, the mobile node sends map-request messages with the solicit bit set to all locators in its mapping cache. In the mapping cache, the mobile node stores the RLOCs of ETRs, to which the mobile node has recently sent a packet to. The ITRs, which receive the solicit map request message, first determine whether there is an entry for the EID in their mapping cache. If there is a mapping, they update it by querying the mapping service. This mechanism requires additional signaling messages after the roaming event but the cached mappings of communication partners are quickly updated. The actual timespan between a handover event and the completion of the mapping updates in the cache depends on the connection from the ITR to the map server of the mobile node. If the connection delay to the map server is rather small, this mechanism allows near real-time applications.

2.5.2 Piggybacking Mapping Data

The piggybacking mechanism uses the ability to add mapping data to map-request messages. After a roaming event, the mobile node invokes the solicit map request mechanism and adds its mapping data to the solicit map request message. Like in the former mechanism, this message is then sent to all ETRs of communication partners. If the receiving ITRs have an entry for that mapping in their mapping cache, they update this mapping with the piggybacked mapping data from the solicit map request message. In addition, appropriate security mechanisms have to be taken into account to avoid malicious insertion of false mapping data in the caches of ITRs. This cache update mechanism requires additional signaling messages and by piggybacking the mapping information, it avoids the lookup for the piggybacked mapping at ITRs of communication partners. Hence, the timespan between handover event and updated mapping cache entries is reduced compared to the plain solicit map request mechanism.

2.5.3 Temporary Proxy-ITR Caching

In case of asymmetric paths between the roaming mobile node and its communication partner, the former two mechanisms are not applicable. A scenario with asymmetric path is the communication with non-LISP domains via proxy-ITRs and proxy-ETRs. To solve this issue, each mobile node keeps track of proxy-ITRs which have recently sent packets to the mobile node in a separate proxy-ITR cache. The cache contains the locators of the proxy-ITRs. After a roaming event, the mobile node sends the solicit map request messages also to these locators. This way, also the mappings in the mapping caches of proxy-ITRs get updated.

2.6 NAT Traversal for Mobile Nodes

When a mobile node roams into a network behind a NAT box, it receives a private care-of-address with only local significance so that this address cannot serve as RLOC for the mobile node. The map server is upgraded to a NAT traversal router (NTR) [8] to make the mobile node behind a NAT reachable from the outside world. The mobile node registers with its NTR when roaming into the private network. Thereby, it punches a hole into the NAT that is maintained by the mobile node and the NTR. The NTR registers its own globally reachable address as RLOC in the mapping system so that traffic destined for the mobile node is sent to the NTR which then forwards it through the established tunnel to the mobile node behind the NAT. Due to the tunnel between the NTR and the mobile node, the NAT traversal mechanism works with every type of NAT, even with symmetric NATs. More details are given in [8].

3. SIMULATION MODEL

In this section, we describe our simulation model of the LISP architecture and its integration in OMNeT++. OMNeT++ [18] is a discrete event simulation environment with a modular, component-based architecture. We built our model on top of the INET framework [17] which extends OMNeT++ with Internet-related protocol implementations and several application models. The nodes in our model are derived from INET nodes and we used the existing IPv4 and UDP implementation as basis for our model. At some places, necessary and non-intrusive modifications were made to the source code of the INET framework.

Initially, our implementation was based on design ideas of OpenLISP. OpenLISP [1] is an open-source implementation of the LISP protocol running in the kernel of the FreeBSD Operating System. However, the OpenLISP implementation does not cover all extensions of the LISP architecture which are described in the LISP standard documents [3–5, 7, 9]. Therefore, we implemented our LISP model mainly based on the standard documents. Starting with the INET framework, we incrementally integrated the basic LISP architecture, map server interface, interworking, mobility, and eventually NAT traversal.

3.1 Modifications to INET

In this subsection, we describe the necessary changes to the INET framework and how we integrated our LISP model.

3.1.1 Integration of DHCP Service

To support a more convenient numbering of IP nodes in larger scenarios and to enable a more realistic handover procedure for mobile nodes, we integrated the DHCP implementation [10] from the INETMANET framework [16]. The integration of the DHCP client and server applications required several changes to the INET framework. These changes were related to the proper handling of the DHCP broadcast messages and the correct propagation of these messages to the DHCP client and server applications inside the host stack. In detail, the `ARP` module, the `IPv4` module, the `RoutingTable` module, the `Ieee80211MgmtAP` module, and the `Ieee80211MgmtSTA` module need to be changed.

The `DHCPClient` module is integrated in all wired and wireless client nodes and the `DHCPServer` module is integrated in a special `DHCPServer` node for wired domains and inside a `Wireless-Router` node for wireless domains. For wired nodes, the DHCP discovery is started at a random time and for wireless nodes, the DHCP discovery is started once the wireless node is associated with

an access point. The information about this event is exchanged between the `Ieee80211MgmtSTA` module and the `DHCPClient` module via the `NotificationBoard`.

3.1.2 Modified Wireless Model

The original model for wireless communications in the INET framework did not support multiple wireless interfaces per client node. However, to test for instance the multi-homing capability of the LISP mobile node architecture, we modified the existing wireless model to support multiple interfaces. In the original model, the `ChannelControl` module kept a list of all wireless nodes in the current scenario and checked for example, whether two wireless nodes are in range. Therefore, each wireless node registered at the `ChannelControl` module. We modified this behavior so that every wireless node registers at the `ChannelControl` module for each wireless interface. The `ChannelControl` module then keeps a list of all wireless entities per interface and not per node like in the original model. This way, a wireless host can be registered with multiple wireless interfaces at the `ChannelControl` module. In detail, this modification required changes to the `AbstractRadio` module, the `ChannelControl` module, the `ChannelAccess` module, and the `BasicMobility` module.

A wireless client node with several wireless interfaces also has several `DHCPClient` modules, one per interface. This way, each `DHCPClient` module is responsible for only one interface. If a `DHCPClient` module is notified via the `NotificationBoard` about an association event of an interface with an access point, it checks whether it is responsible for that interface. Only in that case, it starts the DHCP discovery process over that interface.

3.1.3 Modified NetworkLayer Module

The basic LISP functionality is implemented in a new `LISPRouting` module. This module is integrated in all LISP nodes and the anchor point is the `NetworkLayer` module and in particular the `IPv4` module. The `LISPRouting` module offers several methods which are called LISP-hooks in the remainder of this document. `lisp_output()` and `lisp_input()` are the most important methods and are explained in the following.

`lisp_output()` handles encapsulation of packets destined to other LISP domains, and schedules mapping requests for unknown EID-to-RLOC mappings. In contrary, `lisp_input()` handles incoming LISP-packets for a LISP domain, i.e., it decapsulates packets, and hands over packets to the `IPv4` module for normal IPv4 routing. The communication from the `IPv4` module to the `LISPRouting` module is done through direct method calls while the communication to the `IPv4` module is done over gates. The `IPv4` module is modified such that the existing INET framework components and nodes still continue to work without any further modifications. That is, the LISP-hooks are only activated if the `LISPRouting` module is also present in the same network node. The LISP-hooks are integrated in the `routePacket()` method and the `reassembleAndDeliver()` method. This enables both the `IPv4` module and the `LISPRouting` module to communicate with each other directly.

3.1.4 Integration of NAT

We also added NAT boxes to test our proposed NAT traversal mechanism for the LISP mobile node architecture. The motivation was to test the interaction with other LISP mechanisms, and not how our mechanism copes with different NAT types.

For our model, we decided to implement a port-restricted NAT as this is the most commonly implemented form of NAT in residential gateways [2]. The NAT functionality has been implemented

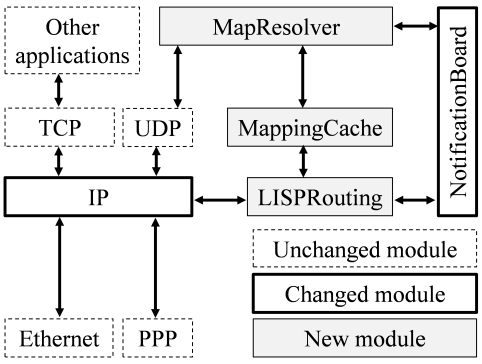


Figure 3: Architecture of a LISP gateway.

in the `PortRestrictedNAT` module and is included in the `NATRouter` node. This module offers hooks which are called by the `IPv4` module similarly to the `LISPRouting` module's LISP-hooks. `nat_input()` processes packets from the outside interface, and translates their destination address and port to the appropriate destination address and port in the local network. In case no mapping can be found in the internal NAT table, the packet is dropped. `nat_output()` processes packets destined to an address in the outside network. The source address of the packet is translated to the outside interface address of the NAT. The source port is translated to a free external port so that returning packets can be mapped correctly to the local source address. It then installs the mapping in the internal NAT table. The hooks had to be placed logically before the LISP-hooks in the `IPv4` module. The `IPv4` module has been modified to be NAT-aware. This was done to preserve backward compatibility with the existing INET framework code base.

3.2 LISP Nodes

In this subsection, we describe the different LISP nodes which were added to the INET framework. All additional LISP nodes are derived from existing INET nodes.

3.2.1 LISPRouter

The `LISPRouter` node is our model of a LISP gateway and the central part of the LISP architecture. It is based on the INET `Router` node. In principle, the `LISPRouter` node acts as ordinary router but with extended functionality as it implements the LISP forwarding behavior. To simplify the implementation, the `LISPRouter` node may both act as ITR and ETR. This way, it is more convenient to build either single-homed or multi-homed domains with several LISP gateways. Depending on the desired scenario, the `LISPRouter` nodes may then be configured as ITR, ETR, or both.

Figure 3 shows a sketch of the architecture of a `LISPRouter` node. Boxes with dashed border represent unchanged INET modules, boxes with bold border changed INET modules, and boxes with regular border and gray-colored background new modules. The important new modules of a `LISPRouter` node are the `MapResolver` module, the `LISPRouting` module, and the `MappingCache` module. The communication between the `LISPRouting` module and the `MapResolver` module is done through INET's `NotificationBoard`. We have chosen the `NotificationBoard` for the inter-module communication because it is also used by several other modules like for example the wireless modules. Hence, using the `NotificationBoard` seemed to be for us the most suitable approach which is in conformance with the INET design principle.

While the `LISPRouting` module is mainly responsible to en-

capsulate and decapsulate the LISP data traffic, the MapResolver module is responsible to send and receive LISP control traffic, e.g. performing the registration process or the mapping lookup. We modeled the MapResolver module as a UDP application because the LISP signaling traffic is sent over UDP.

The MappingCache module in between the MapResolver module and the LISPRouting module is used to cache the returned mappings after a mapping lookup. During a mapping lookup, the MappingCache module may either store the pending packets or drop them like in the current implementation of LISP in the beta-network. After a mapping entry inside the MappingCache module has been used, its timer may either be refreshed or not. The last option may for instance be important for mobile nodes where the mapping regularly changes. The different configuration options can be changed via flags at the parent LISPRouter node. This way, scenarios with different LISPRouter behavior can be quickly simulated.

3.2.2 LISPMapServer

The LISPMapServer node acts as interface to the mapping system and receives the map-register messages and map-request messages from MapResolver modules. These messages are sent over UDP with port 4342. Hence, the LISPMapServer node contains a MapServer module which is modeled as UDP application. The module receives the LISP signaling messages and performs the required operations like adding the mapping to the mapping system or returning a requested mapping.

In real-world implementations, the map server acts as an interface to a distributed mapping storage and not as mapping storage itself. In the current state of our implemented model, the map server acts as both interface and mapping storage. To further improve the accuracy of our model, we think about integrating either the *LISP Alternative Topology* (LISP+ALT) [6] mapping system or our own mapping system FIRMS [12]. Currently, the second option seems more likely as we already have a working implementation of our mapping system in the INET framework.

3.2.3 LISP Proxy Gateways

LISP proxy gateways are necessary to connect LISP-domains and non-LISP domains. In our model, non-LISP domains are built with untouched INET nodes and LISP-domains have a LISPRouter node as border router. To enable interworking, proxy gateways need to be placed in between LISP and non-LISP domains.

In our model, each LISPRouter node can also be configured to act as either proxy-ITR, proxy-ETR, or both. In case a LISPRouter node acts as LISP proxy gateway, it has to be placed outside of edge domains. All routers are then configured with a route to the proxy-ITR which is used for packets destined to EIDs.

3.2.4 LISPMobileNode

To include the LISP mobile node architecture in our simulation model, we extended the LISPRouting module. Extra conditional code was needed in the IPv4 module to handle both packet sending and delivery correctly. The MapResolver module has to be made aware of care-of-address changes. These changes are signaled through the NotificationBoard. The network node representing a mobile node is based on INET's WirelessHost. The architecture of a mobile node can be seen in Figure 4.

In the following, we describe the function of the important modules by means of a handover example. After a roaming event, the DHCPClient module first starts the DHCP discovery process once the wireless interface is associated with a WirelessRouter node. This node acts as access point and as DHCP server. Once the

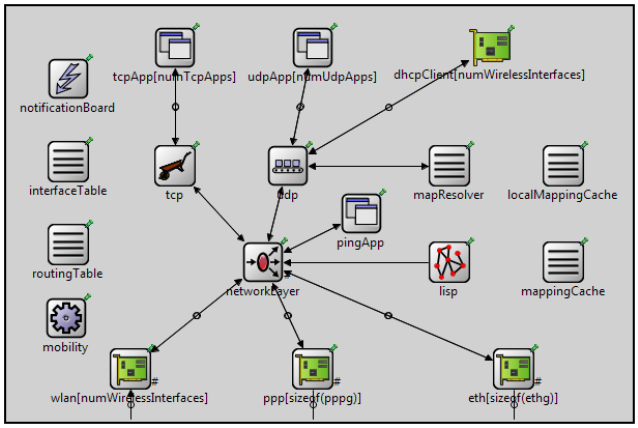


Figure 4: LISP mobile node in OMNeT++/INET.

DHCP process is finished and the wireless interface got a new care-of-address, the registration process is started by the MapResolver module. Again, this event is signaled between the InterfaceTable and the MapResolver via the NotificationBoard. During the registration process, the MapResolver sends a map-register message to the configured MapServer node. The MapServer acknowledges the map-register message with a map-notify message. Once the MapResolver receives this message, it starts the configured update process of remote caches of communication partners, see Section 2.5. This completes the handover process. The also visible localMappingCache module in the above mobile node is part of a set of improvements to the LISP mobile node architecture. See [13] for more details.

3.2.5 NATTraversalRouter

The network node representing an NTR is called NATTraversalRouter. A UDP server application receives map-register messages from mobile nodes and registers a special mapping if it recognizes that the mobile node is behind a NAT. The special mapping is sent to a pre-configured MapServer node while the original mapping is stored in the NATTraversalRouter. Beside the UDP server application, an NTR-hook is installed in the IPv4 module. If a mobile node behind a NAT registered its mapping at the NATTraversalRouter, and other nodes communicate with the mobile node, all traffic is relayed through the NATTraversalRouter. The NTR functionality on the network layer has been implemented in the NTRRouting module. The hook ntr_input() uses the NATTraversalRouter cache to resolve the mapping and to forward the received packet to the correct NAT gateway with the correct address port combination. The NTR-hooks are only used if the appropriate module is present in the network node.

3.2.6 LISPConfigurator

This node is derived from the FlatNetworkConfigurator and used to configure the LISP parameters of the different LISP nodes. In addition, also some parameters of non-LISP nodes like DHCP servers are configured. As an example, the LISPConfigurator assigns the EID prefixes to LISPRouters and DHCPservers in LISP domains. In the current version, we have disabled the feature to also fill the routing tables as the FlatNetworkConfigurator was not intended for hierarchical networks. Hence, we fill the routing tables by routing table files. However, to enable a quicker and more convenient configuration of larger scenarios, we also think about a robust way to automatically fill the routing tables for hierarchical LISP and non-LISP networks.

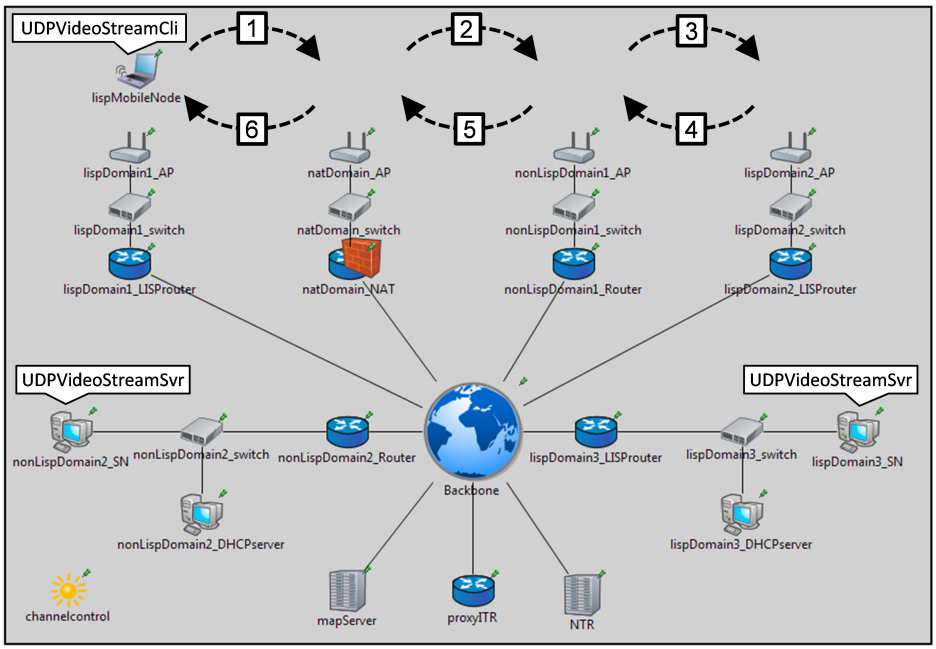


Figure 5: LISP mobile node moving from left-hand side to right-hand side and back. During its movement, the following handover events occur: $(1) : LISP \rightarrow NAT$, $(2) : NAT \rightarrow nLISP$, $(3) : nLISP \rightarrow LISP$, $(4) : LISP \rightarrow nLISP$, $(5) : nLISP \rightarrow NAT$, $(6) : NAT \rightarrow LISP$.

4. EVALUATION

In this section, we investigate the handover performance of the LISP mobile node architecture in different scenarios.

4.1 Simulation Setup

The architecture of the simulation setup can be seen in Figure 5. The objective is to evaluate the handover performance of a LISP mobile node (`lispMobileNode`) which roams between a LISP domain, a non-LISP domain, a non-LISP domain behind a NAT gateway, and another LISP domain. We use a `RectangleMobility` module to move the mobile node from the left-hand side to the right-hand side and back. This way, all possible six handover combinations of source and destination domain type are covered (see Figure 5).

During its movement, the mobile node with the UDP application module `UDPVideoStreamCli` requests a video stream from either the stationary node (`lispDomain3_SN`) inside the LISP domain or from the stationary node (`nonLispDomain2_SN`) inside the non-LISP domain. Both nodes may act as video stream server and have a `UDPVideoStreamSvr` UDP application module. The `UDPVideoStreamSvr` module sends the video stream with a constant bit rate to the requesting `UDPVideoStreamCli` module inside the `lispMobileNode` network node.

The wireless network is modeled as IEEE 802.11g WLAN and we use the standard wireless channel model of the `ChannelControl` module. The link delays for all intra-domain link delays are set to 1 ms whereas the inter-domain link delays are set to 10 ms. In addition, the link delays to the `mapServer`, to the `proxyITR`, and to the `NTR` are set to 50 ms delay. The higher link delays for these nodes were chosen to better see the differences between the analyzed handover scenarios.

4.2 Handover Scenarios

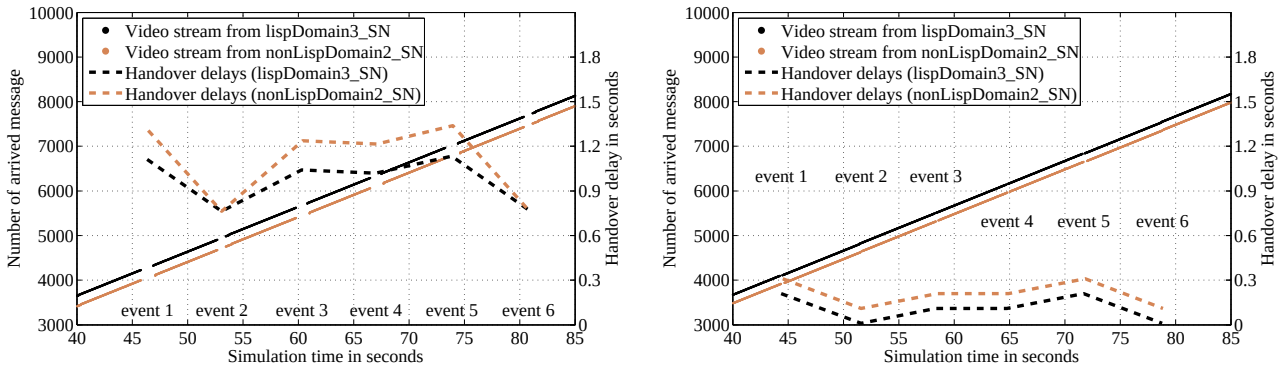
In this subsection, we first show the difference between a single-homed and a multi-homed mobile node. We further investigate the reasons for the handover delays in another evaluation and show how different cache update mechanisms reduce the handover delay.

4.2.1 *Effect of Multi-Homing*

As first evaluation, we investigate how the video stream from LISP and non-LISP domains is influenced during the handover. We further show how a mobile node with several interfaces could use multi-homing to reduce the handover delay. As remote cache update mechanism, we use the solicit map request mechanism described in Section 2.5.1 for the communication with the LISP domain. For the communication with the non-LISP domain, the extended solicit map request mechanism as described in Section 2.5.3 is used.

Figure 6(a) shows the effect of the handover for a single-homed mobile node. The black solid line denotes the video stream from the `lispDomain3_SN` node and the brown solid line denotes the video stream from the `nonLispDomain2_SN` node. The video stream from the `lispDomain3_SN` node was requested earlier than the stream from the `nonLispDomain2_SN` node. Hence, the black line appears on top of the brown line. In addition, the black and brown dashed lines show the corresponding handover delays for the six described events in a different scaling. We see that for both scenarios, the video stream is interrupted during the six handover events but in all cases, the video stream continues after the mobile node has registered its new mapping and has updated the remote caches in either the `lispDomain3_LISProuter` or the `proxyITR`. The duration of each handover is between 0.8 and 1.3 seconds and there is 100 percent packet loss for all scenarios within this time.

Figure 6(b) now shows the results for a mobile node which uses multiple interfaces to decrease the handover delay. We see that the handover delay is significantly reduced compared to the single-homed case. As soon as the mobile node detects a new access point on its other currently not connected interface, its starts the association process with that new access point. Once this is done, it requests an address via DHCP and registers this address as its current locator at its map server. During the LISP signaling process,



(a) Single-homed mobile node.

(b) Multi-homed mobile node.

Figure 6: Comparison between mobile node with one and two wireless interfaces.

the mobile node is connected to both access points. Hence, packets via the old access point still reach the mobile node. Once the new mapping is available at either the `lispDomain3_LISProuter` or the `proxyITR`, the packets arrive at the mobile node via the new access point.

4.2.2 *Influences on the Handover Delay*

During the handover in the single-homed case, several elements influence the total handover delay. At first, it takes some time until the mobile node recognizes that it has moved out of the wireless range of its associated access point. In our model, each access point announces its presence with beacon frames which are transmitted every 100 ms. If a mobile node does not receive a beacon frame for 350 ms, it infers that it has moved out of the range of its access point. At this point, it starts the scanning for other wireless access points again. For our model, we use the passive scanning method with a `maxChannelTime` of 300 ms. During this timespan, the mobile node listens for other beacon frames on a certain channel. This is done for all possible channels. In our model, we allow only one channel. Hence after one `maxChannelTime`, if the mobile node has discovered another access point, it starts the association process with that access point. Once it is associated, it starts the DHCP discovery and requests a DHCP lease. After it has successfully obtained a lease, it starts the LISP signaling process. This process comprises the registration of the new mapping and the update of remote caches of communication partners.

In Figures 7(a) and 7(c), we show the mean and 95% confidence interval for the individual delays during a handover event for the single-homed scenario and for all six possible handover events. Again these events occur, when the mobile node in Figure 5 moves from the left-hand side to the right-hand side and back. To calculate the mean and the 95% confidence interval, we have conducted each experiment eight times

The beacon lost detection delay, and the scanning and association delay are as expected, and take about 300 ms. The larger confidence interval for the beacon lost detection delay is due to the exact moment within one beacon frame window (100 ms), when the mobile node leaves the wireless coverage. The other delays are not directly influenced by random numbers and hence, the confidence interval for those is rather small. Comparing the delays of the different events, the LISP signaling delay is the only difference. Roaming into the NAT domain has the longest LISP signaling delay because the packets need to be relayed via the NTR to traverse the NAT (events 1 and 5). This relaying process adds an additional delay of 100 ms. This is not necessary during handover events between LISP and non-LISP domains (events 3 and 4). Hence, these LISP signaling delays are 100 ms lower. Finally, roaming from a NAT domain has the lowest LISP signaling delay (about 100 ms) as the NTR serves as anchor point (events 2 and 6). In this case, it is sufficient to update the mapping at the NTR. Once this is done, the NTR relays the packets to the new domain. The relaying process via the NTR is done until the new mapping is also available at the `lispDomain3_LISProuter` node. At this point, the packets are sent directly as the NTR is not necessary anymore.

For the video stream from the `nonLispDomain2_SN` node, we see a similar behavior except that the detour via the proxy-ITR prolongs the LISP and the total delay for all scenarios except the scenarios where the mobile node roams from the NAT domain to another domain (events 2 and 6). In this case, there is no detour via the proxy-ITR and hence, the delay is the same for both scenarios.

4.2.3 *Different Remote Cache Update Mechanism*

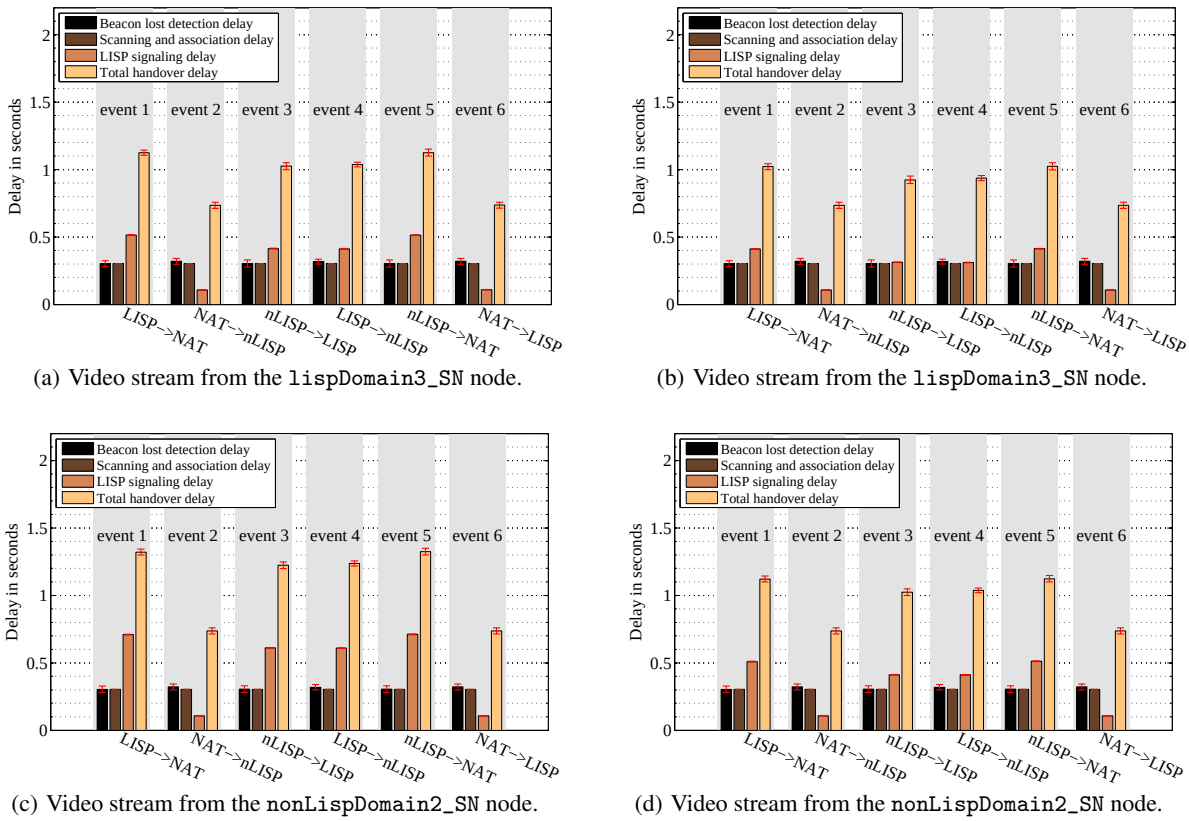
We show the individual handover delays when the piggybacking mapping data mechanism as described in Section 2.5.2 is used. In Figure 7(b) we show the individual delays for the video stream from `lispDomain3_SN` node and in Figure 7(d), we show the delays for the video stream from the `nonLispDomain2_SN` node.

In the first scenario, the piggybacking mechanism avoids one mapping lookup (100 ms) at the `lispDomain3_LISProuter` node for all handover events except for those where the mobile node roams from the NAT domain to another domain (events 2 and 6). This is because in this case, the lookup does not affect the handover at all as the traffic is relayed via the NTR.

In the second scenario, the piggybacking mechanism also avoids one mapping lookup at the `proxyITR` which results in 200 ms lower handover delay. Again, this is only true for handover events, where the NAT domain is the source domain of the handover event (events 2 and 6). In this case again, no detour via the proxy-ITR is required and hence no mapping lookup is involved.

5. CONCLUSION

In this work, we presented the integration of the LISP architecture into the INET framework. The simulation model includes the basic architecture with interworking mechanisms, the LISP-MN architecture, and our NAT traversal method for LISP-MN. As an application, we evaluated the interoperability of our proposed NAT traversal method and the LISP-MN architecture during different handover events. We showed that our mechanism worked well in maintaining the transport connection survivability even for NAT domains. Future work could comprise the integration of a mapping system, e.g., LISP+ALT, in order to perform simulations of the entire LISP architecture on a large scale.



(a) Video stream from the `lispDomain3_SN` node.

(b) Video stream from the `lispDomain3_SN` node.

(c) Video stream from the `nonLispDomain2_SN` node.

(d) Video stream from the `nonLispDomain2_SN` node.

Figure 7: Individual handover delays for solicit map request (a, c) and piggybacking (b, d) update mechanism.

6. ACKNOWLEDGEMENTS

The authors would like to thank Alexander Seith for updating the implementation as well as Rastin Pries and Prof. Tran-Gia for the fruitful discussion and the support in this work.

7. REFERENCES

[1] The OpenLISP Project. http://www.openlisp.org/, Oct. 2011.

[2] L. D'Acunto, J. Pouwelse, and H. Sips. A Measurement of NAT & Firewall Characteristics in Peer-to-Peer Systems. In *ASCI Conference*, Zeewolde, the Netherlands, 2009.

[3] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID Separation Protocol (LISP). http://tools.ietf.org/html/draft-ietf-lisp, Feb. 2011.

[4] D. Farinacci, D. Lewis, D. Meyer, and C. White. LISP Mobile Node. http://tools.ietf.org/html/draft-meyer-lisp-mn, Oct. 2011.

[5] V. Fuller and D. Farinacci. LISP Map Server. http://tools.ietf.org/html/draft-ietf-lisp-ms, Jan. 2012.

[6] V. Fuller, D. Farinacci, D. Meyer, and D. Lewis. LISP Alternative Topology (LISP+ALT). http://tools.ietf.org/html/draft-ietf-lisp-alt, Dec. 2011.

[7] D. Klein, M. Hartmann, and M. Menth. NAT Traversal for LISP Mobile Node. draft-klein-lisp-mn-nat-traversal, July 2010.

[8] D. Klein, M. Hartmann, and M. Menth. NAT traversal for LISP mobile node. In *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, pages 8:1–8:6, Philadelphia, USA, 2010. ACM.

[9] D. Lewis, D. Meyer, D. Farinacci, and V. Fuller. Interworking LISP with IPv4 and IPv6. http://tools.ietf.org/html/draft-ietf-lisp-interworking, Feb. 2012.

[10] J.-C. Maureira. DHCP Server/Client for INET/INETMANET. https://github.com/jmaureir/DHCP, Aug. 2010.

[11] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 Address Allocation and the BGP Routing Table Evolution. *ACM SIGCOMM Computer Communications Review*, 35(1):71 – 80, Jan. 2005.

[12] M. Menth, M. Hartmann, and M. Hoefling. Firms: A mapping system for future internet routing. *IEEE Journal on Selected Areas in Communications*, 28(8):1326–1331, 2010.

[13] M. Menth, D. Klein, and M. Hartmann. Improvements to LISP Mobile Node. In *22nd International Teletraffic Congress (ITC)*, pages 1 –8, Sept. 2010.

[14] D. Meyer, L. Zhang, and K. Fall. RFC4984: Report from the IAB Workshop on Routing and Addressing, Sept. 2007.

[15] C. Perkins. RFC3344: IP Mobility Support for IP4, Aug. 2002.

[16] A. A. Quintana. INETMANET Framework for the OMNeT++ Discrete Event Simulator. https://github.com/inetmanet/inetmanet/, 2012.

[17] A. Varga. INET Framework for the OMNeT++ Discrete Event Simulator. http://github.com/inet-framework/inet, 2012.

[18] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, Mar. 2008.