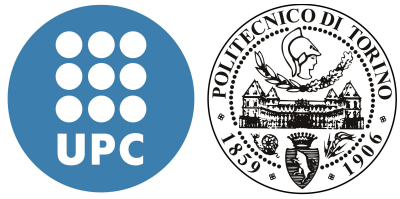
POLITECNICO DI TORINO

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Master degree course in Computer Engineering
Facultat d'Informàtica de Barcelona

Master Degree Thesis

Support for Network-based User Mobility with LISP



Supervisors:
Albert Cabellos Aparicio
Fulvio Giovanni Ottavio Risso

Candidate:
Andrea Galvani

Academic year 2012-2013

If you always put limit on everything you do, physical or anything else, it will spread into your work and into your life. There are no limits. There are only plateaus, and you must not stay there, you must go beyond them.
(B. L.)

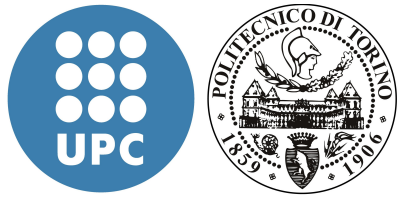
POLITECNICO DI TORINO

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Master degree course in Computer Engineering
Facultat d'Informàtica de Barcelona

Master Degree Thesis

Support for Network-based User Mobility with LISP



Supervisors:
Albert Cabellos Aparicio
Fulvio Giovanni Ottavio Risso

Candidate:
Andrea Galvani

Academic year 2012-2013

Summary

This work focuses on overcoming some of the typical limits imposed by the structure of the Internet in a mobility scenario. The increasingly need for continuous connectivity is not corresponded by the development of supporting solutions, and nowadays has become a critical matter. Besides user mobility, it's mandatory to take in account the different network infrastructures accessible to the users - for example wired networks, Wi-Fi or cellular networks - and the continuous switching of one to the other during the same operation (e.g. video streaming or file downloading). The goal to reach is developing the most possibly abstract solution for making a user roam in different networks, without dropping his active connections. Better, design a network architecture in charge of maintaining user's connections and being transparent to the user at the same time.

The research for this thesis has been done at Universitat Politècnica de Catalunya (BarcelonaTech), in Barcelona, mainly collaborating with the LISPmob research team. The work has been further followed and helped by Cisco Systems.

Acknowledgements

I would like to use this space to thank all the people that made this work possible.

First, a huge thank you to prof. Risso for giving me the opportunity to develop my thesis at Universitat Politècnica de Catalunya in Barcelona, and mostly for having followed me step by step even at a distance, and also for having continuously motivated me during my work. Another big thank you to prof. Cabellos Aparicio, for constantly keeping an eye on my project and for having been available to follow and advise me every single day. I would like to especially thank Alberto Rodriguez Natal, for having really supported me since the first times during my staying at UPC and most of all for bearing me and my questions at - literally - any time of the day. I want to thank Albert Lopez too, for having helped me a lot in the implementation part of my thesis, easily solving problems that would have instead taken me ages. Thanks a lot to Fabio Maino and Preethi Natarajan from Cisco Systems, which followed the developing of my work from time to time giving me great advice and a lot of interesting cues.

Above all the others, I want to deeply thank my parents, and all my family, for providing me the possibility to study and build my future. Thank you for having believed in me, I'm lucky to have you. I want to reserve a special thank you to my sisters who always supported me at a distance.

What really matters to me is to thank all of my friends that I met during all these years of study, and that accompanied me during my adventure across Forlì, Bologna, Torino and Barcelona. I'm profoundly thankful to those people that I consider part of my "home", which I always feel near me, no matter where I am. I don't want to make a list of names. You know who you are. I just hope that I added a +1 to your life too.

In the end, a "thank you" is not enough to Sonia, my constant reference and motivation in all I do.

Contents

List of Tables

List of Figures

Part I

Introduction and scenario overview

Chapter 1

Introduction

1.1 Scenario

Mobility in recent years has become an important issue to deal with, and a big concern regarding user's network experience. Users and devices are becoming less tied to just one network, and are starting to move across multiple networks or even on different supports. Smartphones and tablets represent the main actors of the Internet scenario nowadays, and their behaviour is mandatory to be taken in account, while deploying networks. The constant increase of the amount of these devices is leading to problems in users' connectivity: due to networks' heterogeneous topologies and infrastructures, users are unable to keep their connections alive while roaming across networks. This creates a conflict between the user actions' flow and the support given by networks, which does not follow this paradigm; better, it remarks a huge difference between the user's and the network's point of view. It's been more than a decade since IETF took this problem in account and, consequently, working groups started to put big effort in deploying solutions for supporting user's mobility. This work will focus on proposing different solutions able to guarantee continuity in user's connection, using LISP, a protocol designed by Cisco.

1.2 Roaming

The term "roaming" ensures that the wireless device is kept connected to the network, without losing the connection. In wireless telecommunications scenario, "roaming" refers to the capability of extending the connectivity service in a location that is different from the "home" location, that is where the service was

first registered. Roaming services in wireless networks allow the user to seamlessly continue data transfer while changing networks. Usually roaming applies in the case of connecting to the same network while changing Access Point: effectively, only the physical route is updated but the IP address remains unchanged. Nowadays, roaming services provide the user with transparent continuous network connectivity while the network infrastructure provides seamless roaming services. For example, a mobile phone call remains connected while the user is changing location and hence changing the radio network. However, the existing roaming solutions are still bound to their underlying infrastructure. This is mostly typical of 3G networks, which provide seamless data roaming service while the device migrates between cells. But it is unable to provide the same type of service when the device disconnects from the 3G network and connects to a Wi-Fi access point, in which case the existing connections are dropped and have to be reestablished.

It's possible to distinguish two types of roaming

1. Seamless roaming

2. Nomadic roaming

Seamless roaming follows the scenario of a cellular phone call: a user is making a phone call while walking, or driving. Above him there's a cellular communication system (i.e. GSM or TDMA) which provides a huge are of coverage, and the user is roaming between different base stations of this system. The user can't tell when he's switching base stations, since there's no degradation or disturbance. This is because roaming prevents network availability. Therefore, "seamless" roaming is the required when the network application requires constant network connectivity during the roaming process.

Nomadic roaming, instead, can be better pictured in a Wi-Fi scenario: a user is using his laptop is his office, connected to his office Access Point, then puts his laptop in standby for moving in another room, where it connects again, to another Access Point. This implies the term "nomadic", because the user is not connected to the network while roaming, but only before and after the move.

The case related to this work can be considered even wider than nomadic roaming, since we also have to take in account what happens when the user roams between different LANs, not only different Access Points. For example, when he connects to his home network then, after a while, to the office one.

The latter case impacts more on applications, since there is an actual period of network inactivity while roaming to an attaching point to another. Further more, when a user connects to a completely different network (LAN), even without any network unavailability time, he drops all of his active connections.

This big constraint is imposed by the nature of the Internet network stack.

1.3 TCP/IP constraints

802.11 roaming is referred as "break before make" implying that a station interrupts the communication associated with one AP before creating an association with a new one. This paradigm was introduced, for example, to facilitate MAC protocol management, even if leads to possible data losses during roaming. If we consider a "make before break" approach, that is making a station able to associate to a new AP before disassociating from the old AP, you would need to add security checks on MAC addresses avoid conflicts. A station connected to the same Layer 2 broadcast domain via simultaneous network connections has the potential to trigger broadcast storms. A "make before break" architecture would necessitate additional algorithms to resolve any potential loops, hence adding overhead to the MAC protocol. In addition, the client radio would have to be capable of listening and communicating on more than one channel at a time, increasing the complexity of the radio [24].

TCP was conceived for wired, fixed topologies which are clearly reliable. Therefore it was assumed that data losses could only be caused by congestion, which can happen in a reliable infrastructure. In a mobility scenario, data losses can be due to the unmanaged bit error rate of wireless links or also temporary network availability, which can come from link errors but also the moving of the host.

We can summarize the scenario we have in mind detailing the nomadic roaming example presented above: A user is working in his office with his laptop and has to move to another room, which has a different Access Point that corresponds to a different network (therefore, different IP addresses). FTP and Telnet connections, for example, can remain alive for the time required to move. The goal is make these open connections being retrieved in the most seamless way possible despite the change of the network. This scenario complies with the TCP paradigm since it assumes that the loss was due to congestion and resizes the transmission by bringing the congestion window down to the minimum size. Further more, the TCP's slow-start mechanism makes the sender unnecessarily hold back, and then slowly grows the transmission rate [24].

A TCP connection is represented by a 4-tuple `<source IP, source Port, destination IP, destination port>`. Connecting to different networks changes the source IP address. For example, a mobile phone may have an IP address that was assigned to it by the mobile operator network. Should the mobile phone connect to a Wi-Fi Access Point it will acquire a new IP address, assigned by the Access Point. When a device changes its source IP address, local TCP connections need to be reestablished since the existing 4-tuple no longer identifies the current connection. Broken connections can either be implicitly handled by the running applications or explicitly handled by the user.

1.4 Objectives of this work

Given the scenario, the goal of this work is to find a way to overcome these limits, adding further constraints.

First of all, we do not want to modify the network stack of the user's host. This means avoiding to install any additional software that modifies the behavior of the network card. It's assumed that we're dealing with hosts using the standard TCP/IP stack. This helps us for abstracting the type of host device that we're considering: no distinctions will be made between PCs, tablets, smart phones, etc. Consecutively, we will operate on the network components, such as access points, routers, servers, etc.

Another objective is to try to limit at a minimum rate the amount of modifications that need to be made to the components involved: most of the effort will be put in changing the behavior of Edge routers, while the other components just need to be configured in the correct way.

In the end, the goals proposed to reach can be summarized as below:

1. Guarantee user mobility without dropping active connections
2. Do not modify host's stack
3. Use the most standard possible network components

Figure 1.1 represents the real scenario and network architecture considered, which is definitely similar to the examples of roaming we presented above. We will refer to the case of the depicted scene as "Handover", which is a term mostly used in cellular communications for the transferring of an ongoing call (or in general, data session) from one channel to another without interruption.

The user connects his mobile device (laptop, tablet, etc.) to a network A, e.g. the university network, then he decides to connect to another Wi-Fi, e.g. his office's. His device won't see any changes in the IP address (the handover process is transparent to the host) and all the connections will be resumed normally. The two networks considered can belong to different domains, for example two different Internet service providers.

The type of mobility considered can be called "inter-domain", which is different from "intra-domain" which is the case when a host changes only its attaching point remaining in the same network (N.B. Inter-domain mobility is sometimes also referred as "macromobility", instead intra-domain is referred as "micromobility").

The work will focus on delivering a solution oriented to roaming between Wi-Fi networks. Assumptions will be made in order to clarify how this scenario can be extended for cellular networks in the future.

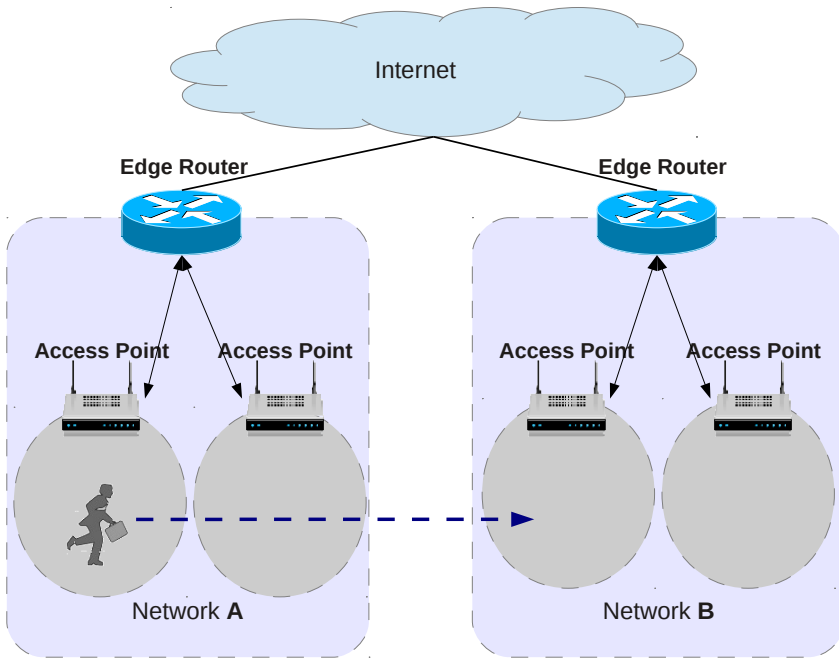


Figure 1.1. Intra-domain roaming scenario

Chapter 2

State of the art

At the time of writing, many solutions for overcoming this obstacles have been proposed. Some of these have been implemented in different environments, but still today there's not a standard approach to user mobility. Before briefly explaining the most notable solutions, we can make a distinction regarding the stack level at which these solutions operate.

1. Application Layer: mobility provided by application layer protocols intends to allow communication end systems to support mobility, heterogeneity, and multihoming. Terminal mobility also allows a device to move between IP subnets, while continuing to be reachable for incoming requests and maintaining sessions across subnet changes.

2. Transport Layer: mobility is intended to maintain TCP's end-to-end reliability and correctness semantics while allowing redirecting the endpoints of an existing transport session to arbitrary addresses.

3. Network Layer: All mobility management and signaling is carried out by L3 protocols.

4. Link Layer: This class includes mobility protocols that use link layer information, when the point of attachment changes, to provide mobility management while the node preserves its network-layer (L3) address. This type is usually involved in intra-domain scenarios.

2.1 Mobile IP

Mobile IP (MIP) [28] uses a stable IP address assigned to moving hosts (from now on, mobile nodes or MN). This "home" address is used to allow the MN to be

reachable by having a stable entry in the DNS service, and to hide the IP layer mobility from upper layers. A consequence of keeping a stable address independently of the node's location is that all correspondent nodes (which are the hosts communicating with the mobile node) can reach the mobile node at that address, without knowing its current location. Therefore, if there are packets forwarded to the home address, and the mobile node is in another network, there will be another node (called "home agent") which will be responsible for tunneling packets to the mobile node's new location. MIPv4 (Mobile IP for IPv4 networks) solves the mobility problem by allowing the mobile node to use a second IP address: the Care-of-Address (CoA). This address changes every time the mobile node moves and changes its point of attachment. The CoA indicates the network prefix, identifying the MN's point of attachment with respect to the network topology. The CoA is composed of a valid prefix in a foreign network: the MN will have a home address and one or more CoAs when moving between networks.

The two main components of Mobile IP are:

1. Home agent, which stores information about MNs whose permanent home address is in the home agent's network.

2. Foreign agent, that instead stores information about MNs visiting its network. Foreign agents also advertise CoA. If there is no foreign agent in the host network, the mobile device has to take care of getting an address and advertising that address by its own means.

When the mobile node moves in a Foreign network, it has to discover the CoA he's assigned. After, this address has to be registered through MIP, in order to update the host's binding. In the end, the Home Agent will establish a tunnel with the MN through the updated CoA.

Going deeper, the operations of Mobile IP can be summarized as following:

1. The mobility agents (Home Agent and Foreign Agent) announces their presence through messages called Agent Advertisement (optionally, these messages can be requested by mobile agents through messages called Agent Solicitation)

2. A MN receives these messages and determines whether it is on its home network or on a foreign network

3. When a MN detects it moved to a foreign network, it obtains a CoA in that network. The CoA can be allocated by the foreign agent or some other address configuration mechanism, such as DHCP (Dynamic Host Configuration Protocol)

4. When the MN is operating in the new network, it needs to register its CoA with its HA, through the exchange of Registration Request and Registration Reply messages

5. Datagrams sent to the MN's home address by a CN are intercepted by the local HA and tunneled to the MN's CoA. The datagram is received at the exit of the tunnel, and finally delivered to the mobile node in the new network

6. Datagrams sent by the MN are generally delivered to the destination using standard routing mechanisms, not necessarily through the HA.

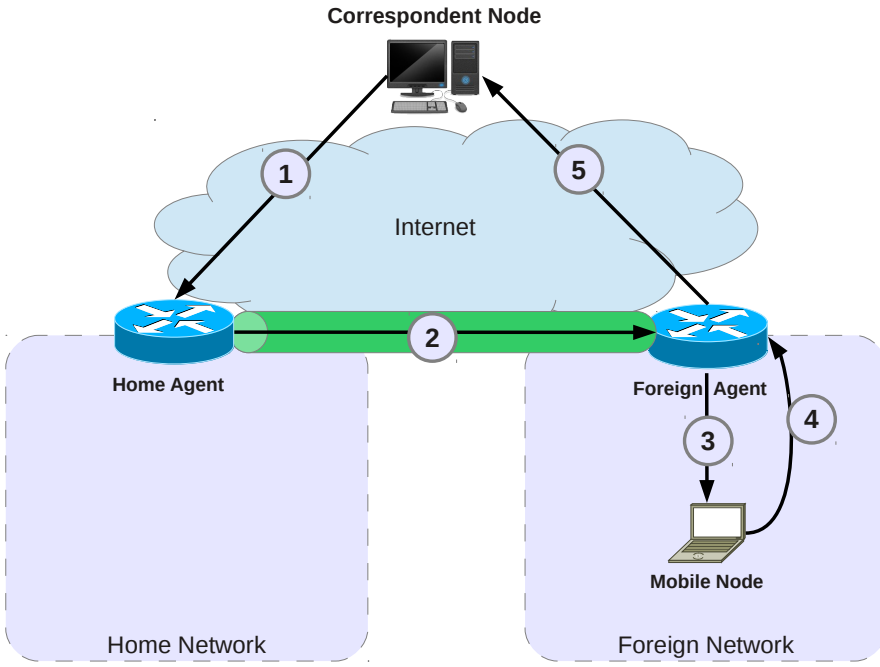


Figure 2.1. Mobile IP architecture and routing

The resulting interaction between MN, HA and CN is called *triangular routing*, as you can see in figure 2.1, which summarizes the MIPv4 paradigm itself.

Triangular routing generates a processing overhead on HA, and in addition it becomes a single point of failure in the network. This problem has been solved with MIPv6, by optimizing the route.

Mobile IPv6 ([10]) was made for supporting mobility support in IPv6 networks. Like in MIPv4 the MN has to establish a binding between home address and care-of-address. This combination of CoA is made by the MN and the HA, and it's possible by a binding registration in which the MN sends Binding Updates (BU) messages to the HA, which then answers with a Binding Acknowledgment (BA). The correspondent nodes are able to store the bindings between the home address of the MN they refer to and its care-of-address. The MN can send information about its location to the Correspondent Nodes, with the Correspondent Binding Procedure, which is a mechanism for authorizing the establishment of binding, called the return routability procedure. If also CNs support MIPv6 they can use the *Route Optimization* process, which is possible if the MN register with the CN. In this case the CN, before sending a packet, look-ups its local cache to find a binding between MN's HA and CoA. If there is an association, the package will be routed to the CoA of mobile node directly, avoiding the triangular routing seen in MIPv4. Hence this is useful to lower the congestion at the home link and the HA.

One drawback (of both versions) of Mobile IP is that it only defines ways of managing macromobility but doesn't take in account micromobility separately. This leads to use the same mechanism in both scenarios. Mobile IP is really not suited for micromobility, since it introduces a high signaling load and considerable delay required for handover. Indeed, the time usually required for the combination of movement detection, new CoA configuration and Binding Update, can not ne always considered acceptable and reliable for applications that require real-time interaction.

2.2 Proxy Mobile IPv6

Proxy Mobile IPv6 (PMIPv6) proposes another approach for IP mobility ([27]). Mobile IP already enables a host to change the point of attachment in an IP network while keeping session continuity, but this ability is not sufficient for true mobility. Enabling efficient handover is an additional and critical requirement. In MIP, triangular routing is a clear bottleneck for performance. Latency is affected by the time required to exchange signaling between the MN and the HA, and there was no way to optimize micromobility. Therefore, newer solutions started proposing the idea of having a "local" Home Agent to provide mobility in a local domain; that is, to provide localized mobility support. This approach is useful also because users typically move in localized environments (for example, they commute between their living homes and their work places) that can be covered with localized domains. Examples of these types of solutions are "Regional Registrations for IPv4" or "Hierarchical Mobile IPv6 for IPv6". In general, the term "localized" refers to a particular area from the point of view of the IP network topology, but depending on the access technology, geographically the area can be large, as happens when

applying a localized mobility approach to cellular networks. PMIPv6 perfectly applies to 3G scenario but it can also be considered for wide WLANs, like campus networks.

Another important change made with PMIP is the introduction of the possibility to support mobility for IPv6 nodes without modifying the host. Mobile Nodes must signal themselves to the network when their location changes and must update routing states in the Home Agent. Therefore, IETF decided to work on a network-based solution, unifying the advantages of a network-based approach with the benefits of localized mobility management strategies. The network will provide mobility support, although the Mobile Node does not participate in IP mobility procedures. That is, network operators can provide mobility support without requiring additional software and complex security configuration in the Mobile Nodes.

PMIPv6 introduces a new architecture (figure 2.2) which resembles MIP in some points:

1. Localized Mobility Domain (LMD): the localized area in which PMIPv6 provides mobility support.

2. Mobile Access Gateway (MAG): the network component that takes care of all the signaling on behalf of the MN attached to its access links. Most of the time it's the access router for the MN, better, the first-hop router in the LMM (Localized Mobility Management infrastructure). It is also responsible for keeping track of the movements of the Mobile Node in the LMD. An LMD has multiple MAGs.

3. Local Mobility Anchor (LMA): a component deployed in the core network, that maintains mappings for every MN. The mapping contains a pointer to the MAG which is managing the MN currently. Packets inside the LMD are routed through tunnels established between the LMA and all the MAGs. The LMA attracts the traffic directed to the addresses assigned to Mobile Nodes in the LMD, meaning that packets with those addresses as destination are routed to the LMA.

When a MN enters a PMIPv6 domain, it attaches to an access link provided by a MAG. The MAG proceeds to identify/authenticate the Mobile Node, and checks if it is authorized to use the mobility management service. If it is, the MAG starts performing mobility signaling on behalf of the Mobile Node. The MAG sends to the LMA a Proxy Binding Update (PBU) associating its own address with the an information of the Mobile Node which represents its identity (it can be just the MAC address or a more complex authentication). After receiving this, the LMA allocates a prefix to the Mobile Node. Then the LMA sends to the MAG a Proxy Binding Acknowledgment (PBA) including the prefix allocated to the Mobile

Node. It also creates a Binding Cache entry and establishes a bidirectional tunnel to the MAG. The MAG sends Router Advertisement messages to the Mobile Node, including the prefix allocated to the Mobile Node, so the Mobile Node can configure an address (through stateless autoconfiguration). DHCP can be used instead of host's autoconfiguration if the MAG implements a DHCP relay.

So when a handover event occurs, the new MAG updates the location of the Mobile Node in the LMA and gives the same prefix to the Mobile Node (using Router Advertisement). In this way, IP mobility is made transparent to the Mobile Node. Hence the Mobile Node keeps the same address configured when it first enters the LMD, even after changing its point of attachment within the network, and from the Mobile Node's point of view the LMD appears as a single link. Further more, all the MAGs have to configure the same link local address for the same MN. So the MN won't see any change in its default route configuration.

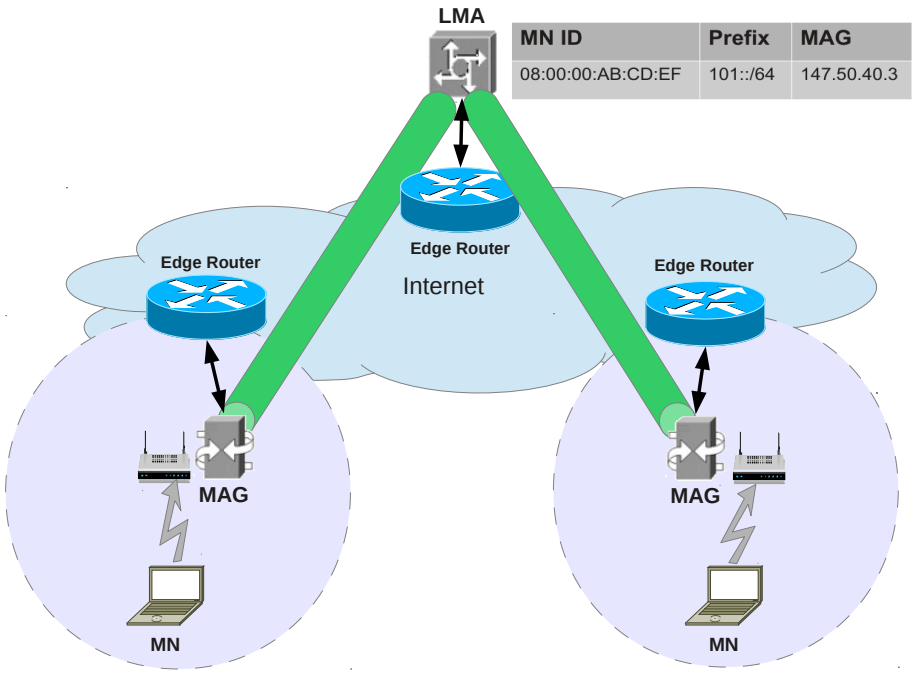


Figure 2.2. Proxy Mobile IPv6 architecture

It's worthy to explain how the tunnel between LMA and MAG works:

The LMA establish a tunnel with every MAG, therefore every packet sent to or from the Mobile Node is routed through the LMA. For example, packets arriving

from outside of the LMD are delivered to the LMA which will forward them in the tunnel to the correspondent MAG. The MAG performs *decapsulation* of this packet and then sends to the correct MN through direct access link. Packets going in the opposite direction (from MN to outside the LMD) are routed to the MAG which this time perform *encapsulation*, sent to the LMA, which will take care of forwarding them to the destination.

We can, at this point, clearly mark the difference between the two approaches seen in MIP and PMIP. Especially the need to modify the host involved in mobility. We will refer to:

1. Host-based mobility, which requires the host to implement specific additional parts (hardware or software) in order to take part in mobility (like MIP)

2. Network-based mobility, which expects to deal with a normal network stack on the host, therefore moving all the workload on network components (like PMIP)

Like stated before, this solution will consider normal hosts (provided with standard TCP/IP stack) as the actors, therefore it will be Network-based.

Chapter 3

Locator/ID Separation Protocol

In recent times, the Internet architecture is beginning to present problems which could not be foreseen in the past and which are strictly related to its nature. One of the biggest regards routing scaling.

This is because of overloaded IP address semantic, which makes efficient routing impossible: the addressing mechanism used in the Internet still follows a paradigm which limits route aggregation compactness. Further more, this problems won't be solved with the introduction of IPv6. What's happening is that routers require bigger memory to store and keep the Internet Routing Table in the forwarding plane, and the amount of memory needed will always be increasing in the future. As a result, people will be replacing equipment to be able to hold the expanding routing table instead of investing resources in implementing new features and satisfying bandwidth requirements.

This problem has been addressed at the Routing and Addressing Workshop that was held by the Internet Architecture Board (IAB) on October 18-19, 2006.The primary goal of the workshop was to develop a shared understanding of the problems that the large backbone operators are facing regarding the scalability of today's Internet routing system. The content discussed has been wrapped up in a RFC document ([11]).

The current Internet routing and addressing architecture uses a single namespace: the IP address, to simultaneously express two aspects of a device: its identity and how it is attached to the network. One clear result of this single numbering space is manifested in the rapid growth of the Internet's default-free zone (DFZ) as a consequence of multihoming, traffic engineering (TE), non-aggregatable address allocations, and business events such as mergers and acquisitions.

This problem has been further worsen by two additional conditions. The first is IPv4 address space depletion which has led to a finer breakup of IPv4 addresses with less aggregation potential, especially in the case of Provider Independent (PI) addressing. The second is the increasing need of dual-stack routers supporting both IPv4 and IPv6 protocols. IPv6 did not change anything about the use of IP addresses, it still represents the location and the identity of the host at the same time, with no logical separation, and so it still suffers from the same problems as IPv4.

There a lot of challenges for managing and maintaining network nowadays. A common case is that the complexity is increased when multi-homing is required for increased bandwidth and availability and for resiliency. When a site has to be re-numbered or services providers change there a lot of expenses related to this operation, which led to inhibit the development of new services.

After [11] the need for avoiding the big effort required for changing the network topology and all the related costs was clear. The idea that came up was to split in two the logical address space normally used. In particular, mark a separation between the location of a node and its identifier (Loc/ID split).

The basic idea behind the Loc/ID split is that the current Internet routing and addressing architecture combines two functions: Routing Locators (RLOCs), which describe how a device is attached to the network, and Endpoint Identifiers (EIDs), which define "who" the device is, in a single numbering space, the IP address. Proponents of the Loc/ID split argue that this overloading of functions makes it virtually impossible to build an efficient routing system without forcing unacceptable constraints on end-system use of addresses. Splitting these functions, through the use of different numbering spaces for EIDs and RLOCs, will yield several advantages, including improved scalability of the routing system via greater aggregation of RLOCs. To achieve this aggregation, RLOCs must be allocated in a way that is congruent with the topology of the network. Today's "Provider Allocated" IP address space is an example of such an allocation scheme. EIDs, on the other hand, are typically allocated along organizational boundaries. Since the network topology and organizational hierarchies are rarely congruent, it is difficult (if not impossible) to make a single numbering space efficiently serve both purposes without imposing unacceptable constraints (such as requiring renumbering upon provider changes) on the use of that space.

Loc/ID split is already commonly used, both with address translations (e.g. NAT) and tunnels (e.g. GRE, IPsec, MPLS), but these techniques are limited to a local scope. For the goal proposed for this work a global scope for Loc/ID separation is needed.

3.1 Protocol overview

LISP (Locator/ID Separation Protocol) is a specific instance of the Loc/ID split, and so its goal it is to introduce decoupling of location and identity in a network ([9]). This separation will facilitate improved aggregation of the RLOC space, implement persistent identity in the EID space, and increase efficiency of network mobility.

LISP is designed to be a simple, incremental, network-based protocol which implements separation of Internet addresses into EIDs and RLOCs. LISP requires no changes to host stacks and no major changes to existing database infrastructures. This is because it is a *map-n-encap* protocol. In the map-and-encap scheme, when a source sends a packet to the EID of a destination outside of the source domain, the packet traverses the domain infrastructure to a border router (or other border element). The border router maps the destination EID to a RLOC which corresponds to an entry point in destination domain (hence there is a need for a EID-to-RLOC mapping system). This is the "map" phase of map-n-encap. The border router then encapsulates the packet and sets the destination address to the RLOC returned by the mapping infrastructure. This is the "encap" phase of the map-n-encap model. Thus map-n-encap works by appending a new header to the existing packet; the "inner header" source and destination addresses are EIDs, and the "outer header" source and destination addresses are in most cases RLOCs, and are the actual addresses used to route the packet to destination. When an encapsulated packet arrives at the destination border router, the router decapsulates the packet and sends it on to its destination. By definition ([9]), EIDs are not globally routable addresses, unlike RLOCs and should just represent the identity of a host. But we can say that when being the EID an IP address it can be considered routable in a local scope, usually just in the domain where the network operates.

The IP encapsulation scheme adopted in LISP decouples host identity and location, allows having dynamic identity-to-location mapping resolution and is also address family agnostic because it allows all the possible EID-to-RLOC combinations: IPv4-in-IPv4, IPv4-in-IPv6, IPv6-in-IPv4, IPv6-in-IPv6.

LISP has minimal deployment impact: it does not require changes to end systems or core, just minimal changes to edge devices, and it's incrementally deployable day-one. Indeed LISP architecture allows not only LISP-to-LISP communication, but also LISP-to-non-LISP, as we'll see.

3.1.1 LISP components

The LISP specification bases itself on a few fundamental network elements, listed below. This list summarizes the definitions of these components, therefore the descriptions are directly taken from [9].

Egress Tunnel Router (ETR) is a router that receives LISP-encapsulated

IP packets from the Internet on one side and sends decapsulated IP packets to site end-systems on the other side. In particular, an ETR accepts an IP packet where destination address in the outer IP header is one of its own RLOCs. The router strips the outer header and forwards the packet based on the next IP header found.

Ingress Tunnel Router (ITR) is a router that accepts IP packets from site end-systems on one side and sends LISP-encapsulated IP packets toward the Internet on the other side. In particular, an ITR accepts an IP packet with a single IP header (more precisely, an IP packet that does not contain a LISP header). The router treats this inner IP destination address as an EID and performs an EID-to-RLOC mapping lookup if necessary (i.e., it doesn't already have an EID-to-RLOC mapping for the EID). The router then prepends an outer IP header with one of its globally-routable RLOCs in the source address field and the result of the mapping lookup in the destination address field. Note that this destination RLOC may be an intermediate, proxy device that has better knowledge of the EID-to-RLOC mapping closest to the destination EID.

xTR is a reference to an ITR or ETR when direction of data flow is not part of the context description. xTR refers to the router that is the tunnel endpoint. Used synonymously with the term Tunnel Router. For example, an xTR can be located at the Customer Edge (CE) router, meaning both ITR and ETR functionality is at the CE router. A router that performs the actions described can be referred as **LISP Router**.

LISP site is a set of routers in an edge network that are under a single technical administration. LISP routers that reside in the edge network are the demarcation points to separate the edge network from the core network.

Map Cache is a short-lived, on-demand table in an ITR that stores, tracks, and is responsible for timing-out and otherwise validating EID-to-RLOC mappings. This cache is distinct from the full "database" of EID-to-RLOC mappings, it is dynamic, local to the ITR(s), and relatively small while the database is distributed, relatively static, and much more global in scope.

EID-to-RLOC Database is a global distributed database that contains all known EID-prefix to RLOC mappings. Each potential ETR typically contains a small piece of the database: the EID-to-RLOC mappings for the EID prefixes "behind" the router. These map to one of the router's own, globally-visible, IP addresses. The databased locally stored and globally announced by an ETR is called **Local Database**.

Even if it is not the main matter of this work, it is mandatory to mention the components needed in order to allow communication between LISP and non-LISP sites.

Proxy Ingress Tunnel Routers (Proxy-ITRs) are used to provide connectivity between sites that use LISP EIDs and those that do not. They act as gateways between those parts of the Internet that are not using LISP (the legacy Internet). A given Proxy-ITR advertises one or more highly aggregated EID-Prefixes

into the public Internet and acts as the ITR for traffic received from the public Internet.

Proxy Egress Tunnel Router (Proxy-ETR) provide a LISP site's ITRs with the ability to send packets to non-LISP sites in cases where unencapsulated packets (the default mechanism) would fail to be delivered. Proxy-ETRs function by having an ITR encapsulate all non-LISP destined traffic to a pre-configured Proxy-ETR.

LISP Mapping System

In LISP the network elements (LISP routers) are responsible for looking up the mapping between end-point-identifiers (EID) and route locators (RLOC) and this whole process is invisible to the Internet end-hosts. The mappings are stored in a distributed database simply called "Mapping system", which responds to the lookup queries. The Mapping System runs distributed since it's composed by multiple LISP Map-Servers and Map-Resolvers.

Map-Server is a network infrastructure component which learns EID-to-RLOC mapping entries from an authoritative source (typically, an ETR, though static configuration or another out-of-band mechanism may be used). A Map-Server publishes these mappings in the distributed mapping database.

Map-Resolver is a network infrastructure component that accepts LISP Encapsulated Map-Requests, typically from an ITR, and determines whether or not the destination IP address is part of the EID namespace; if it is not, a Negative Map-Reply is returned. Otherwise, the Map-Resolver finds the appropriate EID-to-RLOC mapping by consulting a mapping database system.

A world-wide testbed has been deployed for testing the functionality of LISP on a large scale, called LISP Beta Network ([1]). The Beta Network contains elements such as Map-Servers, Map-Resolvers, Proxy Routers and xTRs. Participants host one or more of these components. It initially used a BGP-based mapping system called LISP ALternative Topology (LISP+ALT) [7], but it has been later replaced by a DNS-like indexing system called DDT inspired from LISP-TREE [31]. The protocol design made it easy to plug in a new mapping system, when a different design proved to have benefits. Some proposals have already emerged and have been compared.

The goal of this work does not require going deeper in the explanation of the deployment of the Mapping System, neither it is mandatory to make assumptions about *which* type of Mapping System is considered. This solution focuses on the interaction between a host device and the LISP router (or xTR), thus the only assumption that needs to be made is that the xTR is able to communicate with a LISP Map-Server in its domain, which is fair considering the scenario we have in mind.

3.1.2 Message flow

When a host in a LISP capable domain emits a packet, it puts its EID in packet's source address, and EID of the correspondent host in its destination address (note that hosts will typically look up EIDs in the Domain Name System). If the packet's destination is in another domain, the packet traverses the source domain's infrastructure to one of its ITRs. Then, the ITR maps destination EID to a RLOC which corresponds to an ETR that is either in the destination domain or proxy's for the destination domain. The ITR then encapsulates the packet, setting the destination address to the RLOC of the ETR returned by the mapping infrastructure or by static configuration.

When the packet arrives at the destination ETR, it decapsulates the packet and sends it on to its destination. As previously stated, this implies that EIDs need to be routable in some scope (likely scoped to the domain).

There are four types of LISP packets that need to be beared in mind:

Map-Request: An ITR may query the mapping system by sending a Map-Request message into the mapping system to request a particular EID-to-RLOC mapping. In order to do this, the Map-Request message is encapsulated before being sent to the Map-Server: the outer IP header containt the RLOC of the requesting ITR and of the Map-Server, in order to route the packet correctly to the destination. As soon as the message is received by the Map-Server, it gets decapsulated and read. The Map-Server will look for the EID prefix requested in the `Record` field. If the Map-Server is not the authoritative one for the EID requested, the Map-Request will be forwarded into the Mapping System.

Map-Reply: This message is used to reply to the requesting ITR, sending back the EID-to-RLOC mapping requested (still, in the `Record` field). It is important to notice that the Map-Reply is directly sent to the ITR, and therefore it is not encapsulated. The Map-Reply can be sent by the Map-Server authoritative for that EID-prefix. Otherwise, the Map-Server can forward the Map-Request to the ETR, which will be the one to send the Map-Reply. This behaviour is decided by the ETR, as soon as it registers its EID-to-RLOC mappings to the Map-Server, as we will see now.

Map-Register: The LISP message sent by an ETR to a Map-Server to register its associated EID-Prefixes. In addition to the set of EID-Prefixes to register, the message includes one or more RLOCs to be used by the Map-Server when forwarding Map-Requests (re-formatted as Encapsulated Map-Requests) received through the database mapping system. An ETR may request that the Map-Server answer Map-Requests on its behalf by setting the `proxy Map-Reply` flag (P-bit) in the message.

Map-Notify: This message is sent by a Map-Server to an ETR to confirm that a Map-Register has been received and processed. In [12] it is not clearly stated that the Map-Notify is sent by the Map-Server also when something about ETR's

mapping has changed, e.g. a EID-prefix previously registered has different RLOCs. This behaviour is described in [5] and deeply explained later.

LISP Tunnel

The most important feature of LISP, which makes it suitable for deploying real mobility solutions, is that when two hosts in different LISP sites are communicating with each other all of the traffic is routed through the ITR/ETR (like normally happens). The difference with LISP is that this traffic gets LISP-encapsulated by the ITR and LISP-decapsulated by the ETR. Due to this mechanism the xTRs establish a so called "LISP Tunnel". The packets are routed in the Internet with the outer IP header set with the xTR's RLOCs, then in the local LISP sites the packets are routed through the EIDs, which are in the inner IP header.

LISP is considered to be an instance of what is architecturally called a "jack-up": the LISP header is put between the two network layers of the packet. The existing network layer is "jacked up" and a new network layer is inserted below it. A LISP data packet is depicted in figure 3.1

The key concept is that when a host moves in a different LISP sites it can theoretically maintain the same EID, changing only its RLOCs, which will be updated (Map-Register) in the Map-Server. Considering the scenario we have in mind:

- Host_A and Host_B in different LISP sites are communicating with each other (e.g. through TCP or UDP)
- All the traffic gets encapsulated by the respective xTRs in a LISP Tunnel
- The outer IP header contains the xTRs' RLOCs, while the inner one contains EID_A and EID_B
- Host_A moves to another LISP site, e.g. changes its Wi-Fi attaching point
- Host_A will get its locators updated, maintaining the same EID_A
- The connections between Host_A and Host_B are still running, because the endpoints <Host_A, Host_B> (and also the ports) have not changed.

Figure 3.2 represents the actions that take place when two hosts want to communicate using LISP.

3.2 Mobility solutions

LISP has some core use-cases, which can be briefly resumed:

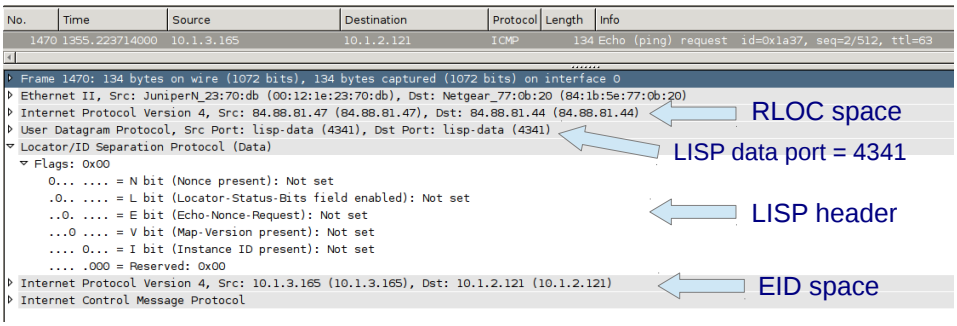


Figure 3.1. LISP data packet example (ping)

- Low OpEx multihoming with ingress traffic engineering (TE) capabilities provides control and management of the utilization of the ingress bandwidth that is being paid for. This is accomplished while eliminating the need for Border Gateway Protocol (BGP) peering with upstream service providers. This case also supports eliminating the need for site renumbering and the associated complexities and costs when changing service providers by decoupling site addressing from core addressing.

- IPv6 Transition support provides inherent, day-one Address-Family agnostic flexibilities. Incorporating LISP into an IPv6 transition or coexistence strategy can both speed and simplify the initial rollout of IPv6 by taking advantage of the LISP mechanisms to encapsulate IPv6 host packets within IPv4 headers (or IPv4 host packets within IPv6 headers). Incorporating LISP into an IPv6 transition strategy has demonstrated quick deployment times, low deployment and operational costs, little or no need for additional equipment or modifications, and high user-satisfaction.

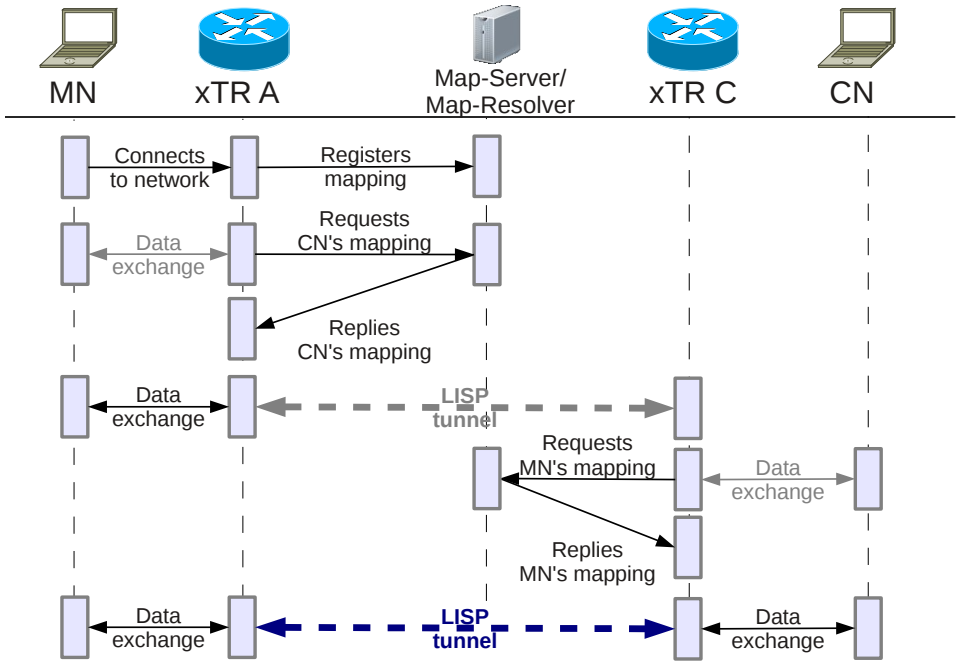


Figure 3.2. LISP data-plane

- Virtualization/Multi-tenancy support provides the capability to segment traffic with minimal infrastructure impact, but with high scale and global scope. Control plane and data plane traffic are segmented by mapping VRFs to LISP "instance-id's", making this overlay solution highly flexible, highly scalable, and inherently low OpEx.

- Data Center VM-Mobility support provides location flexibility for IP endpoints within the data center network and across the Internet due to the servers' identifiers (EIDs) being separated from their location (RLOC). By using Cisco LISP VM-Mobility, you can deploy IP endpoints such as virtual machines anywhere regardless of their IP addresses and can freely move them across data center racks and rows, to separate locations, and globally across organizations.

- **LISP Mobile-Node** support provides a "lightweight" version of LISP's ITR/ETR functionality can be used to provide seamless mobility to a mobile

node. This allows TCP connections to stay alive while roaming, for example, and allows mobile nodes to communicate with other mobile nodes, while either or both are roaming - across the "shortest path" (no home agent).

3.2.1 LISP-MN

Quoting from [6]: The LISP Mobile Node implements a subset of the standard Ingress Tunnel Router and Egress Tunnel Router functionality. Design goals for the LISP mobility design include:

- Allowing TCP connections to stay alive while roaming.
- Allowing the mobile node to communicate with other mobile nodes while either or both are roaming.
- Allowing the mobile node to multi-home (i.e., use multiple interfaces concurrently).
- Allowing the mobile node to be a server. That is, any mobile node or stationary node can find and connect to a mobile node as a server.
- Providing shortest path bidirectional data paths between a mobile node and any other stationary or mobile node.
- Not requiring fine-grained routes in the core network to support mobility.
- Not requiring a home-agent, foreign agent or other data plane network elements to support mobility. Note since the LISP mobile node design does not require these data plane elements, there is no triangle routing of data packets as is found in Mobile IP.
- Not requiring new IPv6 extension headers to avoid triangle routing.

LISP-MN takes advantage of the LISP infrastructure in order to overcome the limits imposed by Mobile IP. LISP-MN is a clear Host-based solution: all the functions usually performed by an xTR are now done by the host itself. Logically the host and the xTR are collapsed into the MN.

Even if the objective of this work is to deploy a Network-based solution there are still some feature that can be abstracted from LISP-MN. The most interesting aspect are the operations that need to be executed when a host (or in this case, a MN) roams in an other network

A roaming event occurs when the LISP-MN receives a new RLOC. Because the new address is a new RLOC from the LISP-MN's perspective, it must update its EID-to-RLOC mapping with its Map-Server; it does this using the Map-Register mechanism.

A LISP-MN may instruct its Map-Server to proxy respond to Map-Requests by setting the Proxy-Map-Reply bit in the Map-Register message. In this case the Map-Server responds with a non-authoritative Map-Reply so that an ITR or PITR will know that the ETR didn't directly respond.

Because the LISP-MN's Map-Server is pre-configured to advertise an aggregate covering the LISP-MN's EID prefix, the database mapping change associated with the roaming event is confined to the Map-Server and those ITRs and PITRs that may have cached the previous mapping.

In order to update ITRs and PITRs mappings a MN/ETR can choose between different techiniques (Map Versioning, Setting Small TTL on Map Replies, Piggy-backing Mapping Data, Temporary PITR Caching) but the one in the interest of this work is using Solicit-Map-Requests.

Soliticit-Map-Request

Soliciting a Map-Request is a selective way for ETRs, at the site where mappings change, to control the rate they receive requests for Map-Reply messages. SMRs are also used to tell remote ITRs to update the mappings they have cached. Since the ETRs don't keep track of remote ITRs that have cached their mappings, they do not know which ITRs need to have their mappings updated. As a result, an ETR will solicit Map-Requests (called an SMR message) from those sites to which it has been sending encapsulated data for the last minute. In particular, an ETR will send an SMR to an ITR to which it has recently sent encapsulated data.

An SMR message is simply a bit set in a Map-Request message. An ITR or PITR will send a Map-Request when they receive an SMR message.

LISP-MN can use *Data Driven SMRs*: An ETR may elect to send SMRs to those sites it has been receiving encapsulated packets from. This will occur when an ITR is sending to an old RLOC (for which there is one-to-one mapping between EID-to-RLOC) and the ETR may not have had a chance to send an SMR the ITR.

We can infer that every MN must know its authoritative Map-Server in order to communicate with it to update its location. In a realistic scenario this can be one of the Map-Servers of the ISP where the MN is subscribed.

3.2.2 LISP VM Mobility

LISP VM Mobility [5] is a solution focused on migrating virtual machines from a LISP site to another, without dropping the running connections. The LISP VM-Mobility solution addresses this issue seamlessly by enabling IP end-points to change location while keeping their assigned IP addresses. A distinction is made between roaming through different subnets or across different locations of a subnet that has been extended with Overlay Transport Virtualization (OTV) or another

LAN extension mechanism. Given the scenario of this work, our interest is focused on seeing how the roaming between different subnets is treated.

The decoupling of Identity from the topology is the core principle on which the LISP VM-Mobility solution is based. It allows the End-point Identifier space to be mobile without impacting the routing that interconnects the Locator IP space. In the LISP VM-Mobility solution, VM migration events are dynamically detected by the LISP Tunnel Router (xTR) based on data plane events. When a move is detected the mappings between EIDs and RLOCs are updated by the new xTR. By updating the RLOC-to-EID mappings, traffic is redirected to the new locations without causing any churn in the underlying routing.

The idea behind LISP VM Mobility is to "move" a virtual machine between distant Data Centers, without changing of the VM. The VM has the role of the "host" in this solution, and it does not have LISP embedded in its stack, unlike LISP-MN. Hence, all the actions to maintain the connections up and running and guarantee the VM to have the same EID are done by the xTRs. LISP VM Mobility is inded a Network-based solution.

There are a lot of differences with the scenario of our work, since here the hosts are virtual machines that are paused, copied to another location, and then resumed. This implies that the moves are detected by comparing the source in the IP header of traffic received from a host against a range of prefixes that are allowed to roam. The xTR realizes a new host has arrived when he sees traffic arriving from a source IP which is not part of its EID prefix (but it's one allowed EID). Obviously, this is not the common scenario of a user moving into a new network: when this happens, the host device restarts the DHCP dialogue in order to obtain an IP suitable for the new network, it does not continue sending traffic maintaing the old IP, since it realizes that the attaching point has changed. The virtual machine is resumed in a new location, and it is not aware of the move, so the network card doesn't see the difference and continues sending packets outside.

There a still a few interesting things related to LISP VM Mobility, which can be taken for deploying a Network-based solution. When a mobile host roams, there are four types of updates that need to take place as a result of the mobility event.

1. The new LISP-VM router needs to detect the newly moved-in VM. The "new" LISP-VM router refers to the xTR where the host lands. It needs to register the /32 for the host's EID using its locator(s).

2. The old LISP-VM router xTR that previously registered the EID needs to be notified of the move.

3. The Map-Server needs to know the new locators for the EID.

4. The remote ITRs and PITRs that have the dynamic-EID cached with old RLOC mapping need to be updated with a new mapping.

This list can be kept as a high-level guideline also for our work.

- For what concerns Step 1, the new xTR will detect the new host as soon as he receives a DHCP packet from it, or any other packets that are automatically sent in order to gain access to the network. When the new xTR learns the EID of the host it will update its location simply though a Map-Register.
- Step 2 is very important, since the new xTR *usually* does not have any knowledge of the host's previous xTR. The previous xTR needs to be notified because it must remove any information regarding the moved host, most of all stop sending Map-Register for its EID. This notification is automatically sent by the Map-Server, as soon as it receives a Map-Register message. Indeed, if the EID that's being updated was mapped to different RLOCs before, the Map-Server will send a Map-Notify to the new xTR (the one who sent the Map-Register) but also to the previous xTR (which RLOCs were previously mapped to the host's EID). In this way, the previous xTR is always automatically informed if some of its hosts moved somewhere else.
- Step 3 is inferred to be obvious after the Map-Register that took place in Step 1.
- Step 4 is similar to the situation shown in the previous paragraph: the host moved into a new network. The Map-Server is informed of the move, so is the previous xTR. But we still need to take care of all the devices that are communicating with our host (called Correspondent Nodes). The problem can be solved with Solicit-Map-Requests messages, as stated for LISP-MN. LISP VM Mobility suggests that the previous xTR should send a SMR to every CN that tries to communicate with a moved host. Another way, which doesn't require the previous xTR to keep a list of the moved hosts, is to automatically send SMRs to every CN (related to the moved host) when the move is detected (see Step 2).

Figure 3.3 resumes the actions that follow a handover event. The actions depicted are the ones required to keep the connections up and running between a MN and its CNs.

3.2.3 State of art

A lot of mobility solutions with LISP have been proposed so far. This paragraph briefly presents the most insteresting features of some of these, which can suit my work. The parameters that must be used to distinguish solutions are two:

- Macromobility vs micromobility
- Host-based vs Network-based

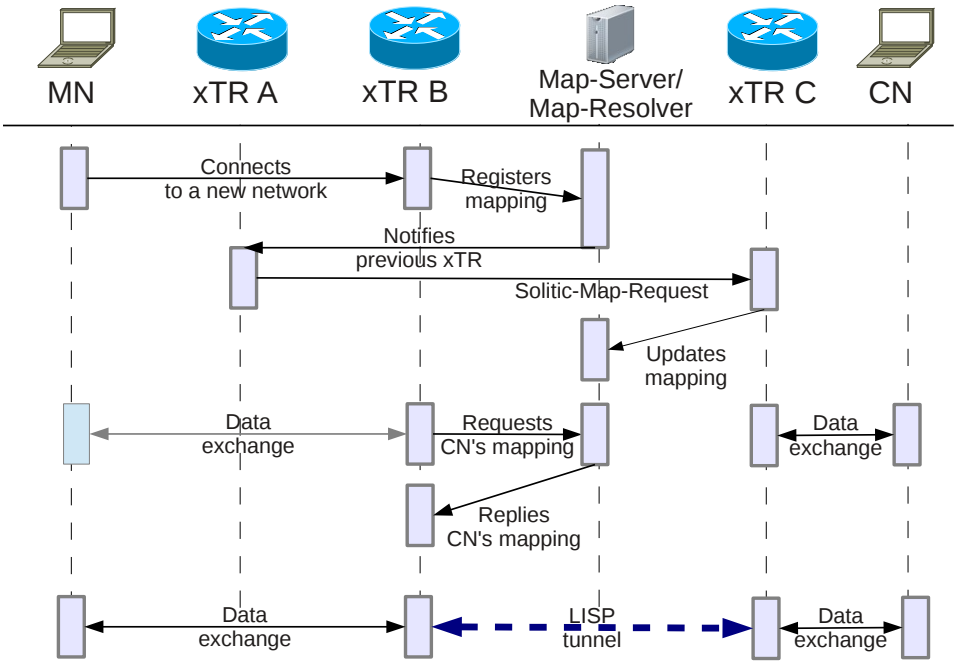


Figure 3.3. Handover action flow

HMM

LISP Hierarchical Mobility Management (HMM) [17] proposes a model of architecture which suits for both macromobility and micromobility. It considers the Internet divided into a number of mapping domains (MDs). An Agent Tunnel Router (ATR) as an agent of each MD manages the Mobile Node's EID-to-RLOC mapping. For the movement within the MD, the ATR keeps the EID-to-RLOC mapping invariable, so it avoids the mapping update in the mapping system and the Tunnel Router (TR) of each correspondent node. For the handover between different MDs, to support fast update and handover, a united mapping table is proposed in the ATR. The goal is to unload the Map-Server trying to keep the maximum load of information inside a local domain. An interesting feature proposed in this draft is that the old xTR (in this case, ATR) should send the list of Correspondent Nodes related to the moved host to the new xTR. This is done as soon as the old xTR is notified of the host's move. This unloads the new xTR to query the Mapping System for the Correspondent Nodes which are communicating

with the host.

Figure 3.4 represents the shortcut introduced with this approach compared to the normal flow previously represented in figure 3.3. When the host roams in a new network the CNs are updated through SMRs, but then the new xTR has to populate its Map-Cache with the mappings of these CNs. With the normal approach, this happens gradually as every CN communicates with the host in its new location. With the mechanism proposed here, the new xTR does not have to query the Mapping System in order to obtain the mapping for every CN, which can introduce considerable communication latency and overhead on the Map-Server.

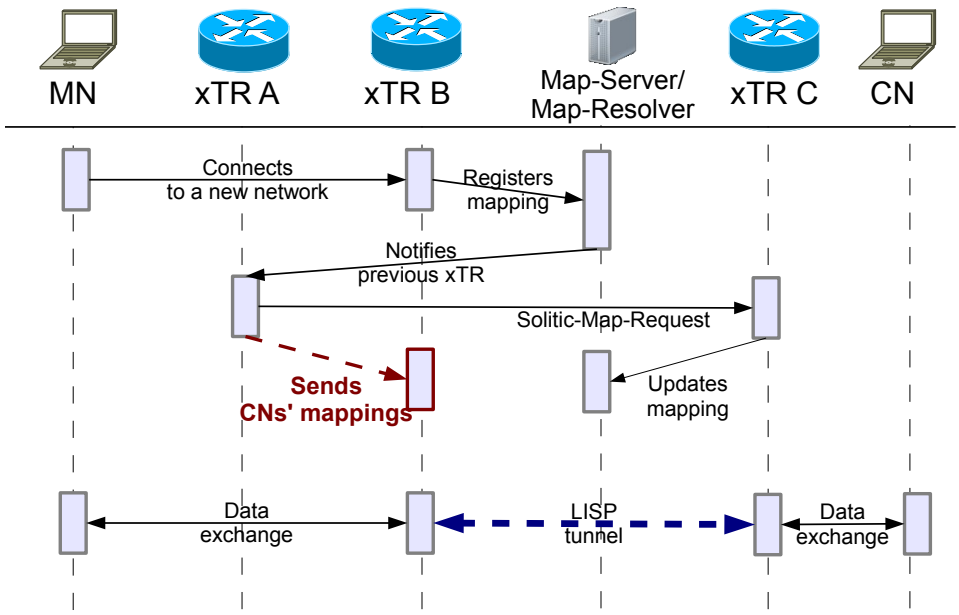


Figure 3.4. Handover action flow proposed in HMM

This idea is also proposed in [26].

LISP-SMOS, LISP-DMC

LISP-SMOS [19] is a seamless mobility support scheme designed to work in Locator/Identifier Split networks. Its goal is to make handover events as seamless as possible. It uses techniques including route optimization, dynamic mapping

and updating based on the mobile node state. It focuses on lower handoff latency, packet loss-free and lower packet transferring overhead. The whole handoff process is transparent to users and completed in the network. The architecture proposed is composed by different layers: Host layer, Edge layer and Core Layer. It decouples the funcionalities of the attaching point (Access Router) and point of access to the core network (xTR). It covers both micromobility and macromobility. The proposal resembles in some way Mobile IP since the home xTR has to keep a state for every host that belongs to its network. It follows the topology proposed in LISP-MN, introducing a LLOC (Local LOCator) for the LISP-MN, which represents its address in the network.

LISP-DMC [22] follows a similar design, but using a distributed mapping system instead of centralized. The distributed mapping system (which works as the LISP Mapping System) stores EID-to-RLOC mappings, instead the Access Routers are in charge of maintaining EID-to-LLOC mappings, which makes micromobility faster. These solutions are heavily host-based since they consider dealing with LISP-MN with a fixed 128 bit EID and that is able to keep mappings in memory. Further more, the MN performs a lot of actions by itself, like updating its mapping when it moves.

If we consider dealing with hosts that are not LISP-MN and don't have any ad-hoc additional software installed in the TCP/IP stack, the scenario is much more difficult. For example, the host can't announce or even know its EID or send a specific message to signal its move.

So far standard hosts have not been taken in account in any of the LISP mobility solutions seen, which creates a big challenge for the work shown in this thesis.

Part II

Proposal design

Chapter 4

Design choices

This list summarizes the main points of this work:

- We would like to enable user terminals to change their network attaching point without impacting their network experience and without dropping any active transport-level session (e.g., TCP/UDP).
- We do not want to install any additional software(e.g., LISP stack) on user terminals. We assume that those devices are equipped with a standard TCP/IP protocol stack.
- As we do not want to modify the user terminals, we assume that the components that may be needed to support the mobility of those devices are provided by the network. In other words, the network will be in charge of any action required to guarantee host mobility (e.g., setup of the appropriate LISP tunnels,etc.).
- We use standard network components in our architecture, keeping modifications needed at minimum, in order to make the proposal easy to implement.
- We assume that the user terminal can move across multiple Wi-Fi networks in different domains. Further more, we want to keep at minimum the requirements needed to support our solution, abstracting from infrastructure / topology / configuration of the networks, in order to guarantee a future extension including mobile operators.

The design has been done assuming the more realistic scenario possible. We're dealing with users roaming in different networks, better in different domains. We will consider users roaming between different Internet Service Provider domains, which are taking part in the same "Mobility Service". We can assume theoretically

to deploy the designed solution in every participating ISP. It goes by itself that there must be a trust agreement between ISPs. From the user point of view, we assume that the user can subscribe to the mobility service as he can normally subscribe to the ISP.

4.1 Overview

First of all, we may clarify the different aspects that must be faced in this work.

4.1.1 ISP topology

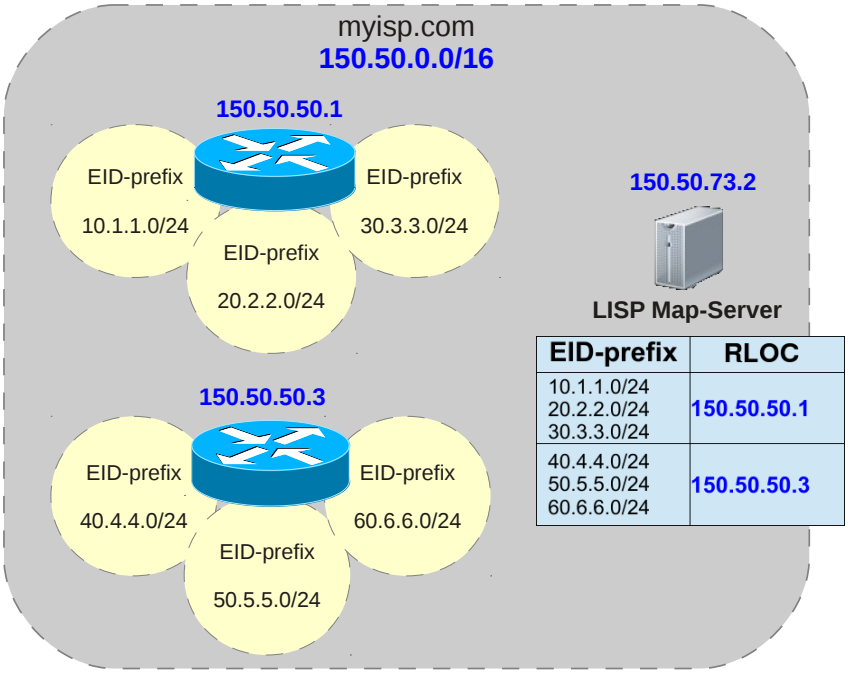


Figure 4.1. ISP topology

The topology to deal with is the one depicted in figure 4.1. We consider adding "LISP sites" to the current architecture of an ISP. A LISP site is composed by one xTR and one or more EID-prefixes below it. Like we said, every xTR has a public address which will be part of the ISP's address space. Every ISP is provided

with (at least) one Map-Server, that maps the EID-prefixes with the xTR's RLOC they're assigned to.

4.1.2 Home and foreign networks

There must be a clear difference between what is a "home" and a "foreign" network. The concept is almost equal to the one used for Mobile IP, like in other mobility solutions. In our solution, when the user subscribes to the mobility service he's registering at a specific ISP, which will become his home domain. When the user roams in a different network, owned by a different ISP, he will be in a foreign domain. For home network we refer to a network that belongs to the home domain (and the same is for foreign network and domain). In general, the home domain is the one in charge of taking care of user's data, and foreign networks should just notify the home domain of the incoming user. It's good to keep at minimum the amount of user's data learned by a foreign domain.

EID-to-RLOC mappings of the users of a domain are maintained by one or more Map-Servers which are also property of the domain. This means not necessarily that the Map-Server is physically in the same network domain but that the credentials to access the Map-Server are known by the domain components. In particular, the components that need to know the secret to access the Map-Server are the xTRs (Edge routers). We will refer to the Map-Server authoritative for a specific user in his domain as "home" Map-Server.

4.1.3 DHCP behaviour

Dynamic Host Configuration Protocol gets involved when a host needs to obtain an IP address in a network. When a host connects to a network it immediately starts the DHCP dialogue, waiting for an answer from the authoritative DHCP Server of that network. Going briefly into the details, the DHCP dialogue is composed by these messages:

1. DHCP Discover: The host broadcasts this message on the physical subnet to discover the DHCP server(s) of the LAN.

2. DHCP Offer: The DHCP Server reserves an IP address for the host and extends an IP lease offer by back this message. This does not only contain the reserved IP but also host's MAC address, subnet mask, lease duration and the IP of the DHCP Server itself.

3. DHCP Request: The host accepts one offer (at most), sending this message to confirm its IP address. This message is broadcasted, to get all the DHCP Servers of the LAN informed of the host's choice.

4. DHCP ACK: This is the last message sent by the DHCP Server to the host. It includes the lease duration and any other configuration information that the client might have requested. At this point, the IP configuration process is completed. Upon receiving this message, the DHCP client will configure its network interface with the negotiated parameters.

There a few notable variants of this dialogue: Sometimes, when the host already knows the network, the dialogue will begin directly with the DHCP Request. And, most importantly, a DHCP client can also request its last-known IP address. If the client remains connected to a network for which this IP is valid, the server may grant the request.

It's been possible to notice two distinguished behaviours regarding this feature. The behavior depends on the type of device and OS installed [1]. A PC device tends to ask for the last IP address obtained in the network it's connecting to. If the network is unknown to it or the host's local cache expired, no specific IP is requested. Instead, most of tablets and smart phones tend to ask for the IP address they obtained last time they were connected, no matter to which network. It usually happens that a mobile device explicitly asks for an address which is not part of the network's address space. In this case, the DHCP Server answers with a DHCP NAK message and the DHCP dialogue restarts from the beginning.

The table 4.1 represents the experimental results obtained studying the DHCP behaviour of different kind of devices.

Table 4.1. DHCP behaviours

OS	**Requests last IP**
Android 4.2.2	yes
Android 4.2.1	yes
Android 4.1.2	no
Android 4.1.1	yes
Android 2.3.6	no
iOS 6.1.3	yes
Windows Mobile 6.1	yes
Windows 8	no
Kubuntu 12.10	no
Ubuntu 12.10	no

[1] It is necessary to say that the assumptions we're making are just the results of experimental tests

As we see, the behaviour is noted in most of the OS deployed on mobile devices. In this case, we can say that the mobile host is "carrying" its old IP address. This information, when present, can help in the design of the solution, or at least can act as a shortcut.

4.2 Procedural steps

It's possible, and necessary at this point, to give a structure to the problems that must be faced while developing the solution.

4.2.1 Host identification

The host is moving in different networks and since it has a normal TCP/IP stack it is not able to send any ad-hoc message to declare its identity or neither its home domain. Hence, host identification (and also authentication, authorization, and accounting) must be done using the information that a host normally *brings*. The host in this scenario is playing a *passive* role, since it does not perform particular actions in order to collaborate with other network components, and it's not providing any specific additional data. So the information that we are able to use are the few that are deductible from the network packets normally exchanged with other network components (most of all, routers).

After having successfully attached to an access link, the host sends a DHCP Request (or Discover) message to obtain an IP address. In this packet. the information related to the host is quite little: the host name and the MAC address. The host name represents the name assigned to the machine and it is not useful for identifying the user. Therefore, we can use the MAC address as a primitive id for the host, representing the physical address of the network which is indeed globally unique. This idea resembles the one seen in [27], and it's just a starting point for the developing of the solution. As we proceed in the writing we will see additional mechanism to identify the *user* instead of the host itself (5.2).

4.2.2 Retrieving host's home Map-Server

In order to update host's EID-to-RLOC binding we must know its home Map-Server for sending him an update (Map-Register) message because, as we said, the home Map-Server is the only one capable of storing host's mapping. There can be two ways to retrieve home Map-Server's address, one using standard DNS and the other which takes advantage of the LISP infrastructure.

DNS approach

It's possible to make use of the well-known DNS service. We suppose that the DNS service (which runs distributed, as each provider is responsible for its domain) can be modified in order to support new DNS records. This should not be a problem because the DNS service was engineered to be extremely extensible, hence we can store additional records for additional purposes. We can think about storing a normal A/AAAA record like:

```
map-server.myisp.com.  IN A 170.60.5.4
```

Remember that, using DNS, we have to assume that every ISP has agreed on the format of the name to store as DNS record (in the example above is `map-server.[domain]`).

A cleaner way should be to use service (SRV) records, which are records that explicitly announce how to reach a particular service of that domain. So, for finding the Map-Server (or a general "LISP service") a DNS query may be issued to find the host name that provides such on behalf of the domain - and which may or may not be within the domain. In our case the DNS entry should be something like:

```
_http._udp.myisp.com.  IN SRV 0 5 80 map-server.myisp.com.
```

DNS architectural choices are up to the implementation.

LISP Mapping System approach

Since we're deploying LISP in the solution, it's clever to use its infrastructure instead of relaying on additional components (such as DNS). For doing this, we must assume that every domain has registered its EID-prefix(es) setting the `Proxy Map-Reply` flag (P-bit) in the Map-Register message, leaving to the Map-Server the responsibility of answering Map-Requests on its behalf. Therefore, we can learn user's Map-Server address sending a Map-Request to the current domain Map-Resolver, asking for the EID of the incoming user (of course we must do this *before* updating user's EID-to-RLOC binding). As we receive the Map-Reply, we can just read the outer source IP address, which will be the home Map-Server address.

4.2.3 Local interface

When the host arrives in a new network it must obtain the same IP address, better, the same EID. The user will have an address that is not part of the address space of the foreign network, so it does not logically belong to the same LAN with the other users. The xTR must act as the default gateway for the host, and the handover must be transparent from the host's point of view. We thought about two possible approaches to overcome this.

Point to point

The xTR can give the host a /30 netmask (255.255.255.252), which is the narrowest possible. In this way, the xTR creates a local subnet that has space only for two hosts, which will be the moved host and the xTR itself. The xTR will set up an additional address on this interface, which will be seen by the host as the default gateway address. It goes by itself that the local subnet is built starting from the EID of the host.

This technique is quite easy to implement but it has a big drawback. Indeed it limits the EIDs that we can deal with to a restricted subset. The /30 netmask forces the local interface to be created starting from addresses that are multiple of four (e.g. X.X.X.120, X.X.X.124, etc.). Also keep in mind that the first and last addresses of a network interface are used respectively for the network itself and for broadcast. Hence we have two addresses out of four that can be used as EIDs. This implies that we are forced to use half of a network's EID-prefix for the EIDs of our mobile hosts.

Proxy-ARP

For not wasting addresses, we can think about deploying Proxy-ARP behaviour on the xTR, which is a technique commonly used in mobility (like in [28]). In this case, the host will be assigned a netmask corresponding to the EID-prefix he belongs to in its home network, not a /30, and every every address of the EID-prefix can be used for mobile hosts.

The mechanism can be explained like this: when a mobile host leaves its home network and moves to a foreign network, the last xTR he was connected to will be notified, as usual. When this happens, the previous xTR uses gratuitous ARP to update the ARP caches of nodes on the home network. This causes such nodes to associate the link-layer address of the xTR with the mobile node's EID. Therefore the nodes under the same prefix of the host will not notice the move of the host because they automatically update their ARP cache with the physical address of the xTR. So all the traffic for the moved host will be directed to the previous xTR which will take care of forwarding it.

The difficult matter is that the new xTR always has to behave like the home xTR for the host. This means that it has to intercept also the traffic that the host is sending on Level 2, which is the traffic directed to the hosts under its home prefix, which are not actually in that physical network. Further more, the new xTR has to forward this traffic outside of its network, so it has to selectively sends ARP replies for every home address request by the moved host.

This behaviour (based on [25]) can be abstracted for every network the mobile host moves to, independently from being home or foreign networks: remember that every time host's LISP binding is updated, the previous xTR the host was

connected to gets notified.

4.2.4 LISP update

The host has been identified (or authenticated), and he maintained its EID. The last step is to update the EID-to-RLOC binding in the Mapping System and notify the Correspondent Nodes. Regarding the host's mapping updates, we found two suitable ways:

Complete trust

Updating is done with a Map-Register directed the host's home Map-Server. The Map-Register message must be authenticated in order to be read by the Map-Server, hence the xTR must know the secret to access the Map-Server. Since we assumed complete trust between the ISP we can imagine having some key exchange mechanism between the parts involved, which are up to the implementation. This implies having ISP sharing keys to their Map-Server, which can result in being quite heavy to manage.

Secret agnostic

Another way can be that the xTR never actually learns the host's home Map-Server key, and therefore it does not send the Map-Register directly to him, but it establishes a dialogue with another xTR that belongs to host's home network. This procedure is explained in deep in 5.1.4.

Updating the correspondent nodes

A critical point of mobility is also to update the bindings of all the hosts that are communicating with the mobile host (correspondent nodes). The *previous* xTR has to be notified that the host moved away from its network, then it has to notify all the CNs.

For achieving this, the xTR has to add another logical level to the LISP Map-Cache it already has. The latter one stores the learned EID-to-RLOC mappings. There should another map, binding host's EID with its CNs' EIDs. The table 4.2 clarifies how this map cache could be deployed.

For what concerns the message that must be sent to the CNs, it's possible to take advantage of the LISP infrastructure already present in the solution, sending a Solicit-Map-Request message to all the CNs for that host. The SMR will trigger an automatic Map-Request which will update the bindings.

Table 4.2. Example of extended Map-Cache on the xTR

Example for Host with EID = 20.2.2.5

$\mathbf{EID_{Host}}$	$\mathbf{EID_{CN}}$	$\mathbf{RLOC_{CN}}$
20.2.2.5	30.3.3.1	160.6.6.1
20.2.2.5	30.3.3.1	160.6.6.2
20.2.2.5	40.4.4.1	170.7.7.5
20.2.2.5	40.4.4.1	170.7.7.6

Chapter 5

Design proposals

This chapter presents all the proposals we were able to develop in this work. All of them have a distinguished scenario, and they're optimized for the environment they're considered to work with. In particular the proposals made are three:

1. LISP-MAC, which introduces a simple but fast roaming mechanism based on the physical address of the host
2. LISP-RADIUS, which introduces the AAA architecture in the solution
3. LISP-ROAM, which improves and simplifies some steps of the other proposals

Every one of these proposals presents different strong points and flaws, making them suitable for different use cases.

5.1 LISP-MAC proposal

The mobile host is roaming between different networks, without any additional software installed on his TCP/IP stack. When he arrives in a new network he does not present any kind of information regarding his previous network (apart from exceptions explained in 4.1.3). What happens in the most common scenario is that the host obtains access to an attaching point of the network, like an Access Point (AP). We can fairly approximate our scenario, considering a host communicating with the xTR (which is the Edge router), even if not directly. This means that, for now, we consider the AP doing nothing but relaying the packets between host and xTR.

As can be seen in figure 4.1, we consider the xTR to be on top of one or more subnetworks which coincide logically with different EID-prefixes. Now it's possible to specify that the scenario considered is a host connecting to one of the xTR's

subnetworks, which means that the mobile host has moved in a EID-prefix which is not its home prefix.

5.1.1 Basic identification with MAC address

The mobile host is connected to a new network, and it's now part of an EID-prefix which does not belong to its home domain (intra-domain mobility). The xTR at the top of the subnetwork does not know anything about the new host.

Like stated in 4.2.1, the first considerable message sent from the host is a DHCP Request (or DHCP Discover). The most relevant information we can extract from this message is the host's MAC address, which acts like a global identifier. Better, the physical address doesn't change depending on the current network, so it can be used for globally mapping the host.

We consider having the DHCP Server of each subnetwork located in the xTR, or at least tied with it. This is an important assumption to make since the xTR must alter the DHCP dialogue with the host and modify the normal behaviour of the DHCP Server. Keep in mind that the home xTR is always keeping the DHCP state of its hosts, no matter where they are.

5.1.2 Host's home xTR

The first approach considered was to not share secrets/keys for LISP Map-Servers between domains. So it was considered that the xTRs of a domain were the only ones able to update the mappings of the users of that domain. This assumption led to consider having a "Home xTR" instead of a home domain/network for the user.

A user subscribes to the mobility service and he's assigned to a specific xTR, which is the only one that can update users' mappings. It's mandatory to bind the user to a specific xTR and not just to a domain, because when the user connects to a foreign network, the foreign xTR has to talk with a specific xTR belonging to user's domain.

Therefore, the user that subscribes to the mobility service gets his specific *host* (identified with its MAC address) registered to a specific xTR, which will be its home xTR.

Why fixed home xTR?

One legit question at this point would be: why there must be a *fixed* home xTR for the host? Why can't the host just be registered by the first xTR it connects to? The point is that we want the hosts to be managed by their own home domain. Which basically means that only xTRs from its home domain can update that host's binding. So if a hosts boots up in a foreign domain, there must be a way

to recognize its home domain, in order to update its location. We could have put more information in the MAC-MS, for example the home domain name, in order to have *dynamic* home xTRs, but we preferred to keep the MAC-MS similar to the LISP Mapping System.

In addition, having hosts with a fixed home xTR allows the ISP to have a more granular security control: we can think of giving the key for a specific EID-prefix only to the xTR that takes care of it. Therefore the keys are not even shared all over the domain, but they are shared at xTR-level.

5.1.3 MAC Mapping System

The mappings linking host's physical addresses to xTR's network addresses can be grouped in a mapping system, which will run distributed like the LISP one. The mapping system obtained can be simply called "MAC Mapping System" (MAC-MS). It's assumed that every ISP is provided with one (or more) MAC Map-Server which is updated everytime a new host is registered to the service. The entry stored in MAC-MS are like:

$\mathbf{MAC_{Host}}$	$\mathbf{IP_{Home\ xTR}}$
80:90:A0:AB:CD:EF	110.10.2.3
0D:05:00:15:F8:AC	110.10.2.4

There are two options regarding *which* address of the xTR should be stored. Indeed, we can just store the RLOC of the xTR, which will be used by the foreign xTR to communicate.

Another more interesting option would be to reserve a specific EID for the xTR, taken from one of its EID-prefixes. So the foreign xTR has to query its domain LISP Map-Resolver for finding not only the RLOC(s) of the xTR but also the priorities and weights of its addresses. It's obvious that the latter option is used mostly in the case of a multi-homed xTR.

For the sake of simplicity, we can think about using a LISP Map-Server also for the MAC-MS, since a LISP Map-Server can store also MAC address, besides other data formats ([8]).

5.1.4 Action flow

Having explained all the components deployed and their behaviours, it's possible to proceed and show how the components interact in our scenario. The flow can be listed like this:

1. The host connects to a new network. In our scenario it can be a user connecting to another Wi-Fi network on purpose or because the signal with the previous one is lost.

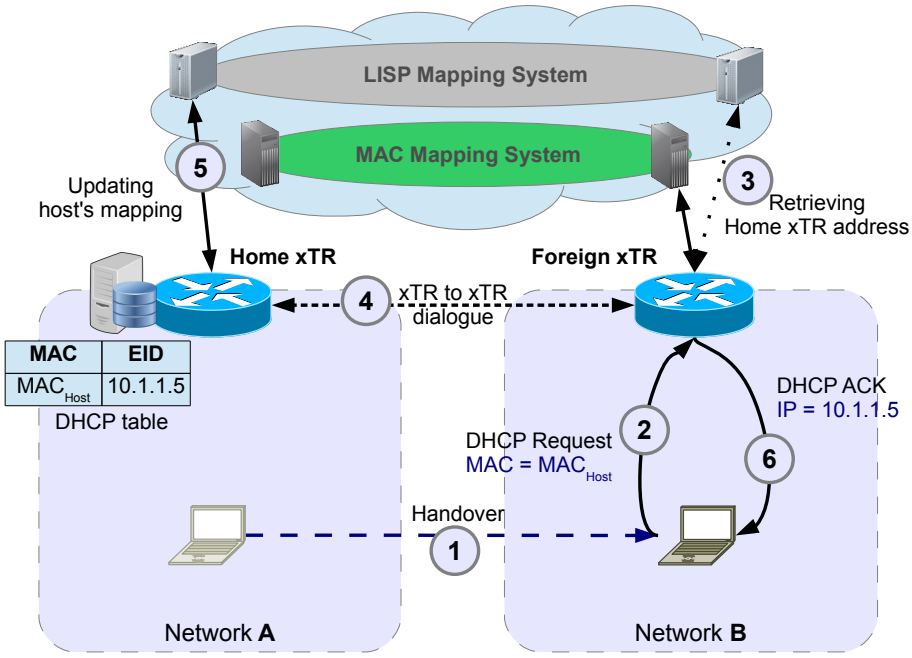


Figure 5.1. LISP-MAC action flow

2. The host sends automatically a DHCP Request, in which it stores the MAC address of the device
3. The message is relayed to the foreign xTR which uses the MAC-MS to find the home xTR of that host, through a Map-Request to its MAC Map-Server (and additionally to the LISP Map-Server, following the procedure explained before)
4. The foreign xTR communicates with the home xTR with a sequence of ad-hoc messages, in order to
   - Obtain host's EID (and other DHCP parameters if needed)
   - Update host's EID-to-RLOC mapping
5. The home xTR updates its host LISP mapping, with a Map-Register
6. At the same time, the foreign xTR is sending back a DHCP ACK message to the host, setting the EID as the IP address assigned for it

7. The foreign xTR finally sets a local interface (as seen in 4.2.3) to isolate the host.

Figure 5.1 represents the whole flow.

Dialogue between foreign and home xTR

This part involves the exchanging of four messages between the two xTRs, listed below.

1. Encrypted DHCP Request
2. Encrypted DHCP ACK
3. Authenticated Map-Register
4. Authenticated Map-Notify

These messages are sent inside of a LISP tunnel, which is normally established between xTRs. Note that the DHCP messages should be encrypted in order to not be sniffed, while the Map-Register and Map-Notify just need to be correctly authenticated.

For what concerns the DHCP dialogue, the foreign xTR is simply relaying the DHCP Request received from the host. The home xTR will refresh the binding (if the host was already connected) and will send back a correctly configured DHCP ACK for that host, including its EID. The same dialogue will occur when a host is renewing its DHCP lease from a foreign network.

After this, the foreign xTR sends an encapsulated Map-Register to the home xTR. Since LISP is deployed on the home xTR, it will automatically relay the Map-Register to its Map-Server, changing the authentication data with its own, that is, authenticating the message with its key. The Map-Notify received as answer will be relayed from the home xTR to the foreign xTR. These two LISP messages are authenticated with keys shared by the two xTRs. How these keys are shared it's up to the implementation. For example, they can use certificates, which is suitable for the trust we assume between domains. The same assumption is made for the encryption of DHCP messages.

The home xTR is always keeping the DHCP state of the hosts assigned to it. And the Map-Server as well is the only one keeping the bindings. When a host moves in a foreign network, the foreign xTR has to take care of retrieving host's information talking with the host's home xTR. Given the trust between the ISPs we can assume the dialogue between xTRs is safe, and that this part should not present security architectural flaws.

5.1.5 Previous xTR behaviour

There's a further distinguish between the xTRs that needs to be made. Before roaming into a new network, the host could be connected to another one (which, of course, is not always the home network), that has to be in some way notified of the moved host. The *previous* xTR is the one that was taking care of the host before its move. This means that every connection directed to the mobile host was routed (with LISP) through the previous xTR. When the host moves, the location is updated in the Mapping System, but still the correspondent nodes need to get notified of this change.

As it has been said before, Solicit-Map-Register (SMR) messages are made for this purpose, and upon receiving it the CN triggers an automatic Map-Request in order to update a specific binding. The problem is that the previous xTR is the only one who knows who the CNs are, since it's keeping the bindings in its Map Cache. As we described in [5] the Map-Server, upon receiving a Map-Register, sends a Map-Notify not only to the sender xTR but also to the xTR that registered that EID last time (i.e. the one currently in the Mapping System), which is actually the previous xTR.

So when a xTR receives a Map-Notify message containing an EID corresponding to one of its users and RLOCs that are different from its it understands that the host has moved from its network (and, embedded in the Map-Notify, it also learns where). Upon detecting the host's move, he will send a SMR to all the CNs for that host, in order to get them updated. Alternatively, as explained in [5], the previous xTR will send a SMR only if the CN tries to reach the moved host, in order to not waste bandwidth. We can take further advantage of this message. For example, the previous xTR can perform an additional check, to see if the host really moved away from its network, which can be done for example sending an ARP Request.

5.1.6 Drawbacks

This solution presents two main drawbacks, regarding two different aspects.

The first one is about the MAC-MS and how it is deployed. One great advantage in using a LISP Mapping System is that the EIDs can be aggregated in EID-prefixes, which makes the EID-to-RLOC tables quite light and easy to manage. Fragmentation in the entries is introduced only in the case of mobility, when the foreign network has to register a mapping which is not part of the prefix. With MAC-MS it's impossible to count on addresses aggregation, since we are dealing with MAC addresses, which don't follow a topological schema like IP addresses. It has to be said that there's a part of the MAC address which is not random, instead it's related to the vendor of the network card, but it has nothing to do with our case. So it's clear that LISP-MAC proposal presents problems when deployed on

a large scale.

The main drawback regards security, and it's due to the nature of MAC addresses. The physical address of the host can represent the identity of the user using the device, but it's quite hazardous to use it as an authentication factor. MAC addresses are not secured, and they're meant to be used for routing in local LANs. Relying on these addresses for identifying the hosts outside of the scope of a LAN can be unsafe. This is because MAC addresses can be very easily spoofed (through just one command line using `macchanger`), so an attacker can rapidly take the identity of another user and redirect all the traffic to its host.

We can say that LISP-MAC proposal introduces a basic but efficient architecture for supporting user mobility. But, given the security considerations, we can imagine using this kind of solution not on a wide scale and inside of networks that implement additional security modules.

5.2 LISP-RADIUS proposal

This proposal has the goal to overcome the security limits of LISP-MAC and to put an effort in deploying a safe architecture for user mobility. Like we stated in 4.2.1, the MAC address can be used just as a primitive identification for the user, so we have to see which authentication mechanism can be implemented in our solution. Bare in mind that the mobile host has to be a standard host, so we can not add any additional software to its TCP/IP stack. Using standard network authentication mechanisms suits this case, since they use standard protocols which are already implemented in the host.

5.2.1 802.1X Authentication

IEEE 802.1X is an IEEE Standard for Port-based Network Access Control. It provides an authentication mechanism to devices wishing to attach to a LAN or WLAN network.

There are three actors involved in 802.1X authentication: a supplicant, an authenticator, and an authentication server.

- The supplicant represents the user's device (laptop, tablet, phone, etc.) which is trying to access the network, attaching to a LAN or WLAN
- The authenticator is the network component (usually an Access Point or Switch) which listens to the supplicant's requests
- The authentication server is usually a machine which is capable of understanding and communicating through RADIUS and EAP protocols

Remote Authentication Dial In User Service (RADIUS) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA) management for users that connect and use a network service. Because of the broad support and the ubiquitous nature of the RADIUS protocol, it is often used by ISPs and enterprises to manage access to the Internet or internal networks, wireless networks, or other services.

In the most common scenario, RADIUS is used in order to authenticate the user when he's trying to access a WLAN, which is the scenario we're considering.

RADIUS follows a Client/Server schema ([21]): Network Access Server (NAS) operates as a client of RADIUS. The client is responsible for passing user information to designated RADIUS servers, and then acting on the response which is returned. RADIUS servers are responsible for receiving user connection requests, authenticating the user, and then returning all configuration information necessary for the client to deliver service to the user. A NAS is usually the Access Point itself, which directly communicates with the RADIUS Server it's been assigned to.

For what concerns security ([21]): Transactions between the client and RADIUS server are authenticated through the use of a shared secret, which is never sent over the network. In addition, any user passwords are sent encrypted between the client and RADIUS server, to eliminate the possibility that someone snooping on an unsecured network could determine a user's password.

PPP Authentication Protocols

There are a number of PPP authentication protocols that are supported by the RADIUS protocol.

Password Authentication Protocol (PAP) is the simplest one. It passes a password as a simple string from the user's host to the NAS device. When the NAS forwards the password, it gets encrypted using the RADIUS shared secret as an encryption key. PAP is the most flexible protocol because passing a plain-text password to the authentication server enables that server to compare the password with nearly any storage format.

Challenge Handshake Authentication Protocol (CHAP) was designed to overcome the danger of passing passwords in plain-text. By using CHAP, the NAS sends a random number challenge to the user's computer. The challenge and the user's password are then hashed by using MD5. The client computer then sends the hash as a response to the NAS challenge and the NAS forwards both the challenge and response in the RADIUS Access-Request packet. When the authenticating server receives the RADIUS packet, it uses the challenge and the user's password to create its own version of the response. If the version of the server matches the response supplied by the user's computer, the access request is accepted.

Microsoft Challenge Handshake Authentication Protocol (MS-CHAP) is a variant of CHAP that does not require a plain-text version of the password on the

authenticating server. MS-CHAP passwords are stored more securely at the server but have the same vulnerabilities to dictionary and brute force attacks as CHAP. When using MS-CHAP, passwords have to be well chosen (to avoid a dictionary attack) and long enough that they cannot be calculated readily (to avoid brute-force).

Extensible Authentication Protocol (EAP) is an extension to the Point-to-Point protocol (PPP) that works with dial-up, PPTP, and L2TP clients. EAP allows the addition of new authentication methods known as EAP types. Both the client and the remote access server must support the same EAP type for successful authentication to occur.

Message Digest 5 Challenge Handshake Authentication Protocol (EAP-MD5 CHAP) is a required EAP type that uses the same challenge-handshake protocol as PPP-based CHAP, but the challenges and responses are sent as EAP messages. A typical use for EAP-MD5 CHAP is to authenticate the credentials of remote access clients by using user name and password security systems. You can use EAP-MD5 CHAP to test EAP interoperability.

EAP-Transport Level Security (EAP-TLS) is an EAP type that is used in certificate-based security environments. If you are using smart cards for remote access authentication, you must use the EAP-TLS authentication method. The EAP-TLS exchange of messages provides mutual authentication, negotiation of the encryption method, and secured private key exchange between the remote access client and the authenticating server. EAP-TLS provides the strongest authentication and key exchange method.

In our solution we suggest using EAP for the dialogue between the supplicant and the authenticator. Figure 5.2 depicts which messages are exchanged when a user authenticates in a foreign network.

eduroam

Using 802.1x authentication we change the paradigm we used in LISP-MAC proposal, which now resembles roaming services like eduroam ([30]).

Eduroam provides a world-wide roaming service between networks belonging to school institutions. In our scenario, we consider a user moving between networks belonging to different ISPs. Most of all, our main goal is not to give the user the opportunity to connect to the same mobility service from different campuses, even in different countries. The focus is put more on letting the user connect to different networks that are provided in the same physical place. For example we want to let the user transparently switch between a 2.4Ghz and a 5Ghz Wi-Fi when he's moving in a larger range inside a building, or better, connect to a different Wi-Fi when he's walking to a different floor. The possibility of having access to the same service from different and remote places (like different countries) is not a matter in our scenario, which regards guaranteeing connection continuity when roaming

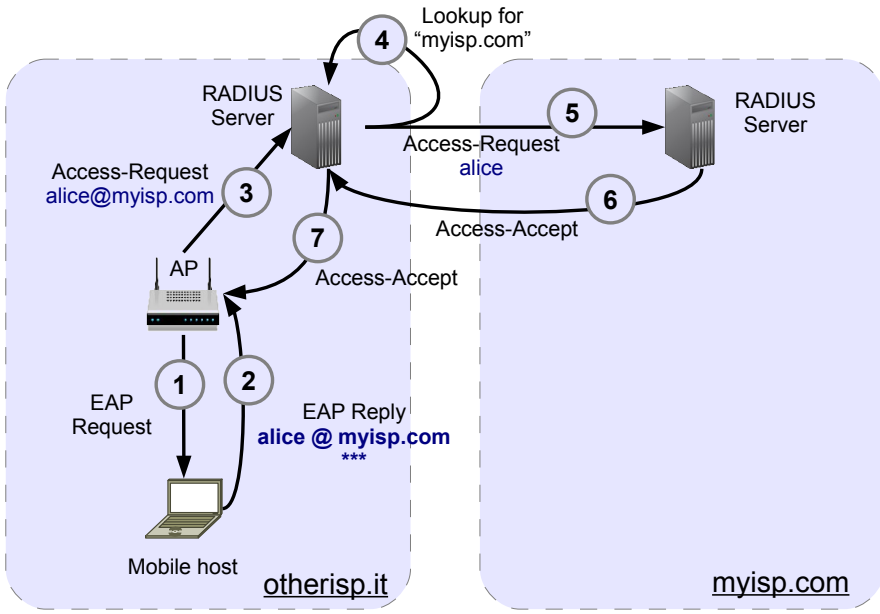


Figure 5.2. RADIUS Proxy message flow

inside a smaller range (e.g. a building) or when moving between close places with a limited period of network inactivity (e.g. from the office to home).

5.2.2 Access-points configuration

When deploying this solution on a widespread scale, we can think about how to ensure seamless roaming between nearby networks. Campus networks that provide connection to eduroam usually deploy a network architecture composed by access points distributed all over the area. When the user obtains access to eduroam for the first time (through username and password) he can move inside the campus without having to re-authenticate every time he changes the attaching point, that is when he connects to a different Access Point. This is because all the access points to the eduroam network have the same Service Set IDentification (SSID), and share the same authentication mechanism. So what happens is that the host's network card has already stored the correspondent authentication data for that wireless network *name*, better, for that network identified with its SSID. Therefore

the host will automatically try to connect to known wireless network with the settings it has stored for that specific SSID. Once on the network, users stay with the same Access Point as long as it is meeting their needs, that is as long as its signal strength is above a sort of quality threshold. The user continuously check if a better connection is provided by another AP on that network, so it will do periodic scans of all channels looking for other APs publishing that same SSID. If a scan turns up a candidate AP that is better than the AP it is currently connected to, it will automatically roam to the other AP. It has to be said that this behaviour is not verified for every host's network card, since it depends on the roaming algorithms and quality thresholds defined. What happens is that sometimes a host doesn't actually roam when it should, ending up being stuck with the first AP they joined even if it can get better performance and reliability with another AP that it's now closer. Assuming both APs are configured similarly and are connected to the same underlying network, roaming is seamless and invisible to the user. Roaming events are invisible at application level, but they can be notified by the lower levels of the network stack. For example, the event is noticed at Level 2, since the host is now connected to a different physical AP. This can trigger a notification directed to upper levels, like IP, e.g. the DHCP client of the network card will restart the DHCP dialogue to check if the underlying network didn't change and so the DHCP lease already obtained is still valid and does not have to be changed.

We can think about make our attaching points provide wireless connectivity sharing the same SSID, and type of authentication required in order to make the roaming between nearby networks as seamless as possible.

The goal of having seamless roaming between wireless networks has been considered also by IEEE and has been standardized in IEEE 802.11F, which specifies the Inter-Access Point Protocol (IAPP). 802.11F ([18]) depicts a possible extension applicable to IEEE 802.11, which provides wireless access point communications among multivendor systems. This extension has been made since, since the beginning, the IEEE 802.11 standard doesn't specify the communications between access points in order to support users roaming from one access point to another and load balancing. IAPP defines a communication protocol between APs beloning to the same network. RADIUS is included is in the architecture and it's used for mapping the SSID of an AP to its IP address and distribution of keys to the APs to allow the encryption of the communications between the APs. Briefly, APs cooperate to provide seamless mobility at Level 2: when the user attaches to another AP of the network, this information is broadcasted to all the other APs, though a message that contains the MAC address of the roaming user and the one of his new location (that is the MAC address of the new AP). Since we're dealing with Level 2 scopes, this type of mobility guarantees mobility only inside the domain of the subnetwork that's implementing it: it does not suits macromobility scenario natively.

What we're trying to achieve is having an efficient intra-domain handover management, instead of the inter-domain case, which is the one usually covered. The

subnetworks we're considering are property of different ISPs, therefore they can not be merged into the same physical network. Using 802.1x authentication assures having a standard and almost ubiquitous authentication mechanism, which is embedded in *most of* the network cards (at least, it's spread quite enough to satisfy the assumptions of this work). Roaming through different domains requires the user to get authenticated each time. Even if transparently (that is, when the network's SSID is already known), this process may require a considerable amount of time, which clearly forbids our solution to be classified as *seamless*. A future extension could be implementing a Single-Sign-On (SSO) system as part of the architecture. With an infrastructure like Kerberos, the user must authenticate only at the beginning of his session, after this he obtains a token which guarantees his identity. It has to be verified how much impact does implementing a system like Kerberos have at application level and, most of all, if it allows a transparent roaming at Level 3, allowing the user to keep the same IP address. Indeed, another benefit of RADIUS is that the authentication part takes place between the host and the access point, before the host obtains an IP address.

5.2.3 Overview

Using 802.1x authentication introduces a big change in the proposal: now it's not the host that gets identified, but it's the *user* that get *authenticated*. There's no need to trust the physical address of the host, since the user gets authenticated with personal credentials. It can be assumed that the user receives this credentials as he subscribes to the mobility service.

As the user moves in a network which supports this service, he will prompted to insert his credentials. After he gets authenticated he will gain access to the network. RADIUS suits perfectly for user's roaming, and for our scenario, since the user declares his home domain while authenticating (`username@domain`). When a RADIUS Server reads a request from a user outside of its domain it acts transparently as a RADIUS Proxy: it forwards the request directly to the RADIUS Server in user's domain and forwards the answer to the user. The RADIUS Server just needs to be configured with the addresses of the other domains' RADIUS Servers, in order to reach them when it's doing proxy. The transparent behaviour of RADIUS eases the deployment of the network architecture, since this one component natively provides authentication and support for user's roaming.

Like in LISP-MAC, the foreign xTR has to learn the user's home xTR address, to start the action flow explained before. Since we're not deploying a Mapping System, like MAC-MS, home xTR's retrieval must be done in other ways.

One possibility is to divide the ISP's logical domain space in many subdomains, which will correspond to the physical LISP-sites. In this way the user belongs to a sub-domain, like `alice@xtr1.myisp.it`, so when it connects to a network he explicitly tells which is the domain name of his home xTR. This introduces a more

granular logical architecture inside the ISP, which is not suitable for scalability in big networks.

A better idea is to take advantage of the RADIUS architecture, better, of the attributes that are possible to attach to RADIUS messages. RADIUS Attributes carry the specific authentication, authorization, information and configuration details for the request and reply. In our case, one of the available attributes can be used to provide the address of the home xTR: the `Framed-IP-Address` indicates the address to be configured for the user. It's possible to use it in Access-Accept packets. Even if we are not really using it for telling the IP to assign to the user, it is still used for carrying important information about the host domain (which will bring us to the user's EID in a second moment). For sending back the home xTR's address (whether it be RLOC or EID, like stated for MAC-MS) we need to configure RADIUS with a storage facility (database or file system) which will keep user's additional data. This approach is very useful because we can think about storing even more information about the user, or his domain, without any security risk, since RADIUS takes care of it ([13]).

These RADIUS behaviour forces an additional change in the architecture. Indeed the xTR must read the Access-Accept message, in order to learn user's home xTR address. What happens normally is that the Access Point takes care of the RADIUS dialogue, and the xTR is only reached by the host when it starts the DHCP dialogue, after being authenticated. It's mandatory to force the RADIUS packet flow to be routed through the xTR. This can be achieved setting the Access Point to ask for RADIUS authentication to the xTR, instead of the domain's RADIUS Server. Upon receiving the RADIUS message from the Access Point, the xTR will be the one who sends the messages to the domain RADIUS Server. In this way, the xTR will be the one receiving the Access-Accept message with the additional attributes, and will also send back the answer (stripped of the attributes) to the Access Point, in order to let the user gain access to the network.

After the authentication part, the host will begin the DHCP process and the dialogue between foreign and home xTR will start as seen for LISP-MAC, and after it all the operations already described above will take place.

Figure 5.3 describes the operations that are followed in order to authenticate the user and retrieve user's home xTR address.

5.2.4 Joined architecture

The two proposals made (LISP-MAC and LISP-RADIUS) are not in conflict or incompatible with each other, instead it is possible to think about deploying both of them in the same domain (figure 5.4).

This combined architecture allows an ISP to guarantee user mobility at different levels of security. For example it may be possible to use LISP-MAC for internal and already secured networks and LISP-RADIUS for a wider scope. Better, we

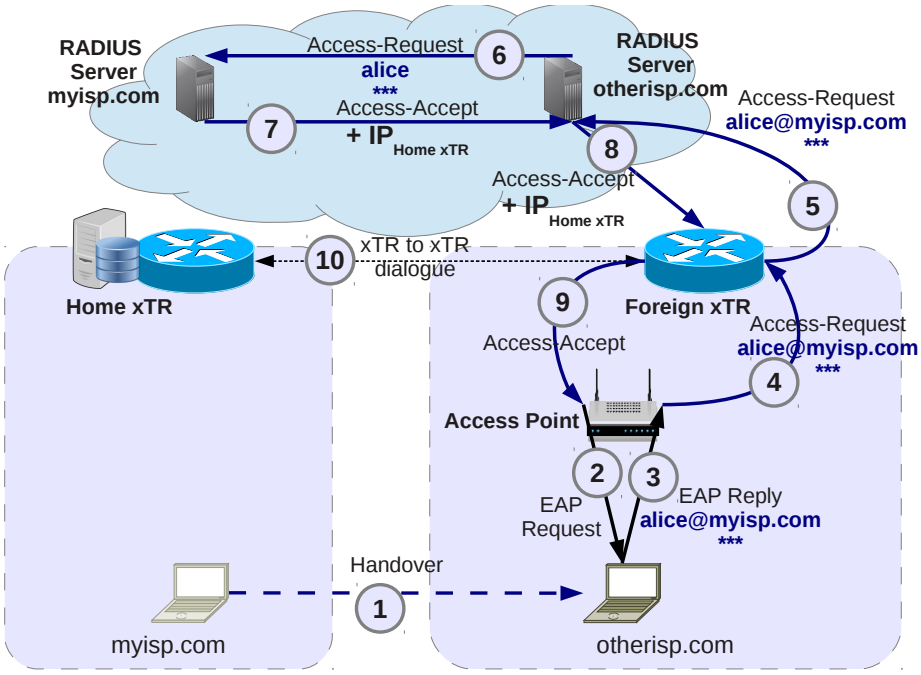


Figure 5.3. LISP-RADIUS (User authentication and Home xTR retrieval)

can think about using LISP-MAC for micromobility inside certain safe networks and LISP-RADIUS for what concerns macromobility.

In the end, there is not one proposal that prevails on the other from all point of views. They can be used alone in different scenarios or they can be joined in the same architecture for taking advantage of both.

5.3 LISP-ROAM proposal

LISP-ROAM follows the proposal presented in LISP-RADIUS, this time introducing assumptions for having an architecture that is more feasible and realistic to implement.

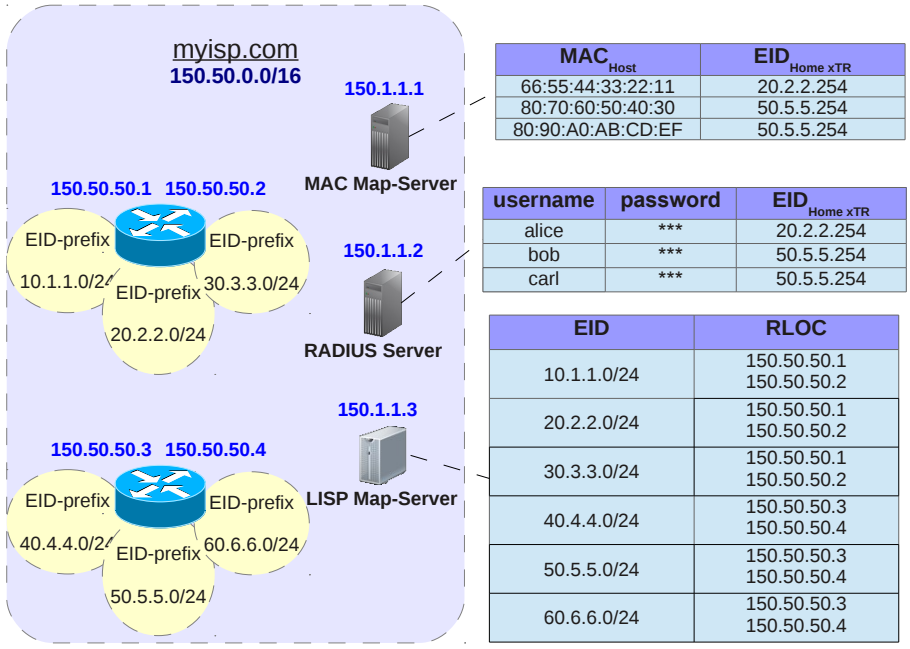


Figure 5.4. ISP supporting both LISP-MAC and LISP-RADIUS

5.3.1 Fixed EIDs

It hasn't been specified until now how is an EID given to the user. Better, how is the EID chosen from the network. The EID must be a globally unique identifier for the user, even if it's not a routable address. In LISP-MN, the EID is fixed and the mobile node takes care of keeping it and registering it into the LISP Mapping System.

In both LISP-MAC and LISP-RADIUS the EID does not have to be fixed. The EID can represent the temporary address assigned by the user's home domain, which is basically what normally happens with standard DHCP. So when a user boots up in his home network the xTR sees that the user is part of its network (through MAC-MS or RADIUS) so it does not start looking for his home xTR. Instead it dynamically assigns an IP address to new host, which will be its EID, and registers the new EID in its domain Map-Server. If the user is booting in a foreign network, the xTR will see that the user does not belong to its domain (again, no matter if we are using MAC-MS or RADIUS) and so it will start the

same operations it does in case of a handover. Indeed, upon receiving a DHCP Request, the home xTR will allocate a dynamic IP (EID) for the mobile host. In case of a handover the home xTR is sending back the EID that the host already has, so what's happening is that the mobile host is actually renewing its DHCP lease. The home xTR just need to embed a DHCP Server to enable this behaviour, which is standard DHCP.

When we talk about a user "booting up" it means that the user starts a new session in a network: in our intention, a new network session occurs each time the user terminal starts using the network after a period of inactivity, such as for example when the user terminal is turned on. In this case, all the previous network activity (e.g., any application-layer communication that occurred in the past) does not affect the data that will be exchanged in the future, because no transport-layer sessions are active. For this reason, when a new session starts, a new EID can be allocated to that user terminal, which may be different from the previous address. Obviously, this assumption should be revised in case the user terminal needs to obtain always the same IP address over time.

In LISP-ROAM user's EID must be fixed. We can imagine that when the user subscribes to the mobility service he's reserved a fixed address, which will always identify him when using that service. Of course this choice is not optimal since it forbids address reuse, and so it affects the availability of the addresses which can't be picked if already assigned, even if the related host is not currently using it. Anyway, this choice allows the xTR to skip the dialogue regarding DHCP with the home xTR, since the foreign xTR will be the one taking care of the mobile host's DHCP state.

Since the EID is now completely tied with the user's identity, RADIUS will be used to store this data. Also, the EID is requested by the foreign xTR in order to take care of the DHCP part, therefore it must be returned as soon as user's authentication ended. In particular, user's fixed EID will be returned embedded in the Access-Accept message, in the Framed-IP-Address attribute, that was already used in LISP-RADIUS for sending the address of the home xTR.

The constraint of having a fixed EID per user may not be considered optimal for a realistic scenario. Indeed it forbids addresses to be reused by the ISP, unlike what normally happens nowadays. One design choice can be having EIDs that are dynamically generated and assigned to the user as he authenticates. LISP-MAC and LISP-RADIUS propose having a DHCP Server, normally deployed in the network, that generates and keeps the EID of a mobile user: but the DHCP Server of the user's home network is the only one aware of the user's EID, therefore it has to communicate this information to foreign networks as the user moves (5.1.4). A way for dynamically generate EIDs as users authenticate would be to put a *global* DHCP Server at domain level, which directly communicates to the RADIUS Server. So when a user authenticates correctly, the RADIUS Server itself will manage the DHCP dialogue with the *global* DHCP Server, obtaining an EID

for the user, which will be attached in the Access-Accept message. The rest of the architecture won't change, so there will be a *local* DHCP Server related to every xTR of the network, which will just keep the current DHCP state of the client in that network. When the user disconnects his device (or after a fixed period of network inactivity) a RADIUS Accounting-Request message is automatically sent, with the attribute `Status-Type` set to `Stop`. This message can be leveraged so that the RADIUS Server tells the *global* DHCP Server that the user's EID is now available again, and it can be reused for another user. The other RADIUS Accounting-Request messages normally sent from the host to the RADIUS Server can be used for keeping the *global* DHCP lease alive.

This proposal requires adding a global DHCP Server for every ISP and modifying the RADIUS Server in order to be able to communicate with this new DHCP Server. One of the goals of this work is to use standard components and maintain the necessary modification at minimum, therefore this option can be considered a hint for a feasible future extension (figure 5.5).

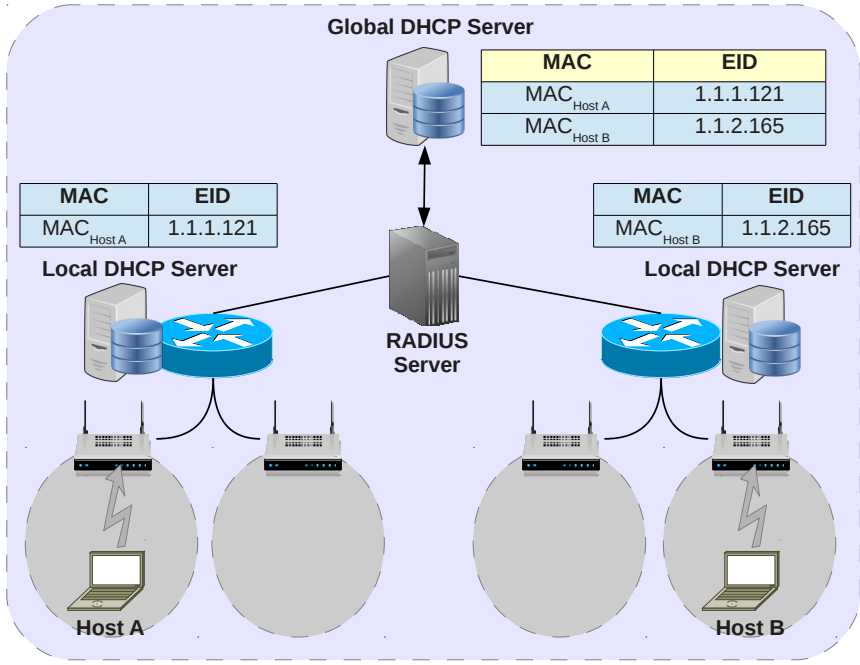


Figure 5.5. LISP-ROAM possible future extension

5.3.2 Full trust

Another important design change regards the supposed trust agreement between the involved ISPs. In LISP-MAC and LISP-RADIUS this agreement implied trusting Map-Servers and RADIUS Servers from other domains and establishing a secret for encrypting the dialogue between home and foreign xTR.

LISP-ROAM goes beyond this and assumes having full trust between the involved ISPs. In particular, what changes is that ISPs are directly sharing the keys for their Map-Servers. This allows the users' EID-to-RLOC bindings to be updated by whichever xTR involved in the mobility solution. These keys can be distributed with out-of-band mechanisms (which are not discussed in this work) or taking advantage of the architecture already deployed for the previous solutions.

As we do with the fixed EID, we can return the secret for the user's domain Map-Server within the RADIUS Access-Accept message (e.g. in the `Reply-Message` attribute). Indeed, the secret is related to the Map-Server but also to the specific EID-prefix, therefore when the ISP is reserving an EID (which is always part of a prefix) for a new user it will easily store also the secret that is necessary for updating it.

Even better, the choice of storing the fixed EID for the user and the Map-Server key together allows us to have a even more precise level of security, since now we can theoretically have one key per user. That is, we can bind keys not to a EID-prefix but also to one single EID, so that the xTR only learns the key necessary for that user. Of course this will introduce storage and management overhead on the Map-Sever, and on the RADIUS Server as well.

Another choice can be to don't store the real key required by the user's Map-Server, but a temporary key. That is a key which is valid just for the current session of the user. After the session expires the key can't be used anymore, resembling a One-Time-Password mechanism. The key related to this session has to be agreed between the Map-Server (which will generate the one-time key) and the RADIUS Server (which will store it). This introduces another additional security level, making the solution more suitable for a realistic scenario. It has to be said that this last proposal requires heavy modifications to the Map-Server, which now is not configured to generate keys dynamically.

In general, the choice of storing the fixed EID with the correspondent key creates a sort of "two-step" authentication: first user's authentication and then, if it's been successful, user's location update.

It goes by itself that the dialogue between foreign and home xTR showed in LISP-MAC and LISP-RADIUS is not needed anymore, since the foreign xTR obtains the EID as the user authenticates and it can directly send a LISP Map-Register to the host's Map-Server.

The whole new dialogue mechanism is depicted in figure 5.6.

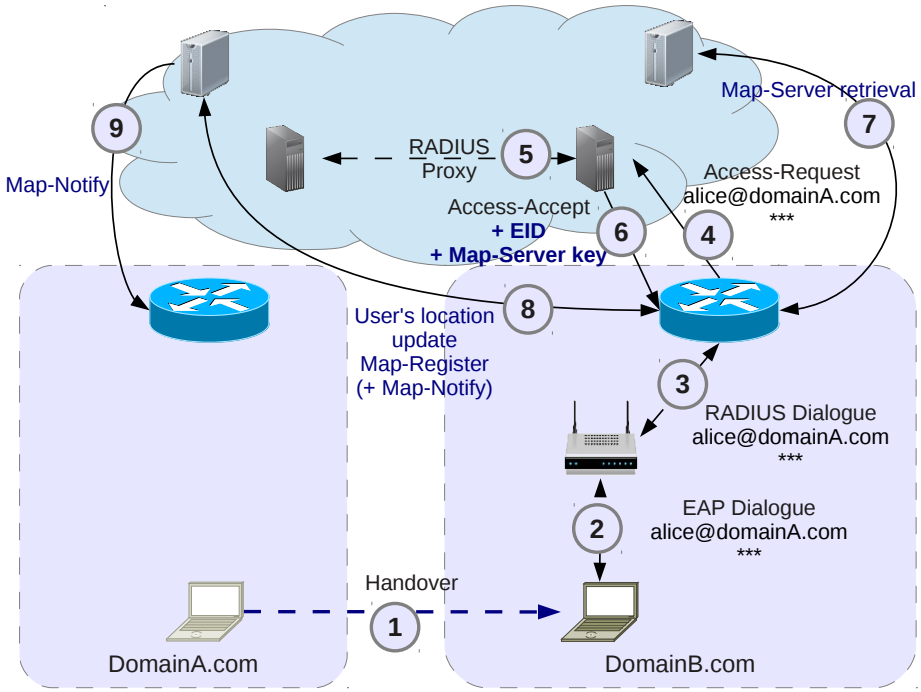


Figure 5.6. LISP-ROAM (user's data retrieval and location update)

Part III

Implementation of a prototype

Chapter 6

Prototype design

In order to gain proof that all the assumptions made are fair and realistic, it has been mandatory to develop a prototype representing the case studied. The proposal developed in the prototype is LISP-ROAM.

6.1 Components

The prototype should resemble a realistic world scenario in the smallest scale possible, using the minimum number of components. It is obvious that the architecture to implement will be used also for making tests and see if everything works smoothly. Therefore only the components necessary for representing a user roaming through Wi-Fi networks will be used.

In particular, the architecture will be composed by

- Three xTRs
  - One represents user's home network
  - One represents user's foreign network
  - One is used by one (or more) correspondent node(s)
- LISP Map-Server(s)
- RADIUS Server(s)

For the sake of simplicity, the xTR will be considered collapsed with the Access Point, i.e. the user will connect directly to the xTR.

Due to practical issues, we consider having only one xTR/AP to represent a domain. Also, every domain *should* have one RADIUS Server and one LISP Map-Server.

For keeping the prototype even more simple and putting more effort in the "core" of the solution, we can consider having just one LISP Map-Server, shared by the three domains. Figure 6.1 gives an idea of the topology implemented.

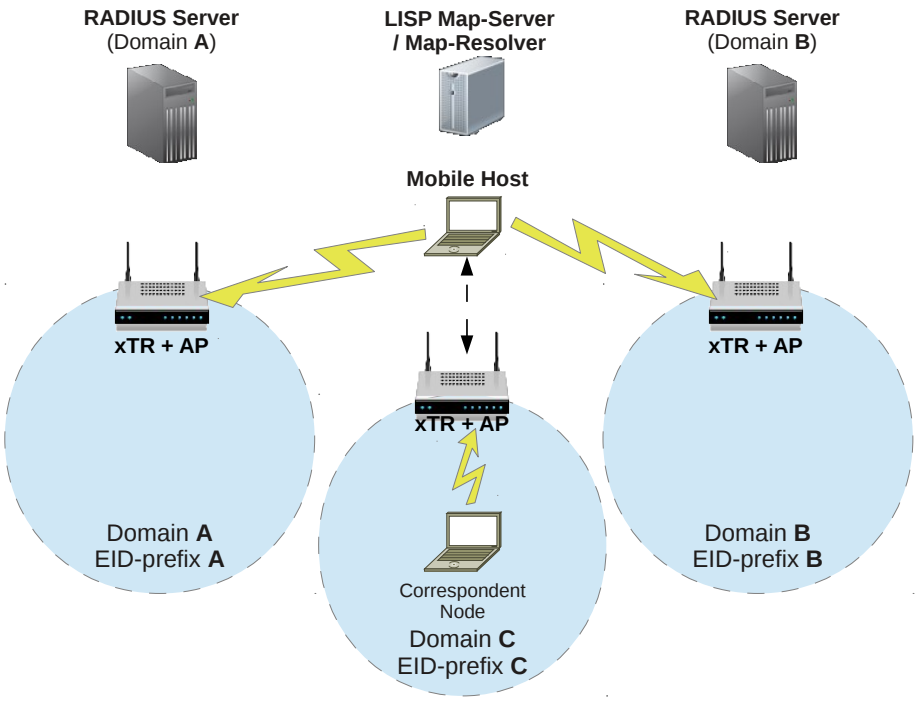


Figure 6.1. Prototype architecture

OpenWRT

In order to be able to modify and manage the behaviour of the routers it's mandatory to use specific firmwares. In the case of this prototype we used **OpenWRT**.

"*OpenWrt is a highly extensible GNU/Linux distribution for embedded devices. Unlike many other distributions for these routers, OpenWrt is built from the ground up to be a full-featured, easily modifiable operating system for your router. In practice, this means that you can have all the features you need with none of the bloat, powered by a Linux kernel that's more recent than most other distributions.*"[23]

Basically OpenWRT lets us use a Linux distribution on a router, providing complete open-source access to the system. As a consequence we are able to fully

configure and re-program the router, as we normally can do with any Linux distribution.

LISPmob

For implementing LISP on the xTRs we used an open-source implementation of LISP, in particular of LISP-MN.

"The LISPmob project aims to deliver a full implementation of both LISP and LISP-MN for Linux-like systems, but parts of the implementation may be reusable on other Unix-like operating systems." [3]

LISPmob provides support for both Data plane and Control plane (and additional tools). LISPmob is composed by a dameon which works by creating a virtual interface for the EIDs, in which all packets are routed and LISP-encapsulated and LISP-decapsulated, and also takes care of the Local DB and the Map-Cache of the node (Data plane). Plus, it manages all the Control plane, that is all the LISP messages necessary for establishing communications. The daemon takes care of both Data and Control plane in user-space. It's possible to avoid dealing with the kernel-space using TUN/TAP, creating a TUN virtual interface to manage the data-plane.

TUN/TAP interfaces are software-only interfaces, meaning that they exist only in the kernel and, unlike regular network interfaces, they have no physical hardware component (and so there's no physical "wire" connected to them). When a program attaches to the TUN/TAP interface, it gets a special file descriptor, reading from which gives it the data that the interface is sending out. In the same way, a program can write to this special descriptor, and the data will appear as input to the tun/tap interface. From the kernel point of view, it would look like the tun/tap interface is receiving data "from the wire". The difference between a TAP interface and a TUN interface is that a TAP interface works with full Ethernet frames, while a TUN interface outputs works with RAW IP packets (and no Ethernet headers are added by the kernel).

LISPmob uses a transient TUN interface, meaning that it's created, used and destroyed by the daemon. When the daemon terminates the interfaces ceases to exist.

LISPmob is available for Linux PC/Servers, Android and OpenWRT. Given its versatility, LISPmob on OpenWRT does not present any big differences with the other versions (just few changes in the configuration file).

As it will be seen further in this work, we modified some parts of the LISPmob code, and consequently its behaviour. The parts of LISPmob modified for our purposes will be sometimes directly referred as "LISProam", because they follow the action flow of the LISP-ROAM proposal.

6.2 Deployment of the architecture

The first part of this chapter regards how to build the architecture used to implement what has been proposed in the design part of this thesis. Every sub-chapter goes deep in the details on how to configure or reprogram the specific network component, trying to be not too much technical when not required.

6.2.1 RADIUS configuration

For the RADIUS part, we had to deploy RADIUS on a server machine. We decided to use **FreeRADIUS** on a Ubuntu Server machine.

"*FreeRADIUS is a modular, high performance free RADIUS suite developed and distributed under the GNU General Public License, version 2, and is free for download and use. The FreeRADIUS Suite includes a RADIUS server, a BSD-licensed RADIUS client library, a PAM library, an Apache module, and numerous additional RADIUS related utilities and development libraries. FreeRADIUS is the most popular open source RADIUS server and the most widely deployed RADIUS server in the world. It supports all common authentication protocols, and the server comes with a PHP-based web user administration tool called dialupadmin. It is the basis for many commercial RADIUS products and services, such as embedded systems, RADIUS appliances that support Network Access Control, and WiMAX. [...] It is also widely used in the academic community, including eduroam. The server is fast, feature-rich, modular, and scalable.*" [29]

FreeRADIUS can be easily found on the apt repository in order to be installed. After this, few configuration files need to be modified. All the configuration files are considered being in the installation directory: `/etc/freeradius/`.

EAP configuration

The file `eap.conf` has to be modified with the type of EAP that we need. For this work we are using **PEAP**, with **MSCHAPv2**.

- Under the `eap { }` section we need to change `default_eap_type = md5` to `default_eap_type = peap`
- Under the `tls { }` section we need to se `private_key_password` to the password we will use to authenticate incoming requests.
- Under the `peap { }` section it must be assured that `default_eap_type = mschapv2`.

In a realistic scenario, it would be mandatory to add custom certificates to the server, and don't use the default ones provided with FreeRADIUS. For doing this,

the fastest way is to follow the procedure described in the `README` file in the directory: `/usr/share/doc/freeradius/examples/certs/`. Before generating the certificates, remember the write the string '01' in the file `serial` and create an empty file called `index.txt`. The procedure will create three files:

- `ca.pem`, which represents the certificate for the Certification Authority, which we can consider having value in our local scope
- `server.pem`, which is the RADIUS Server certificate
- `server.key`, which is the private key used by the server side.

These files need to be copied in `/etc/freeradius/certs/`.

Clients configuration

RADIUS needs to be configured to answer only to authorized requests. Therefore, the list of allowed clients has to be appended to the file `clients.conf`.

```
client 84.88.81.44 {
      secret = ***
      shortname = xTR
}
```

In this work, the xTR will directly query the RADIUS Server. So the entry for the client has to contain the public IP of the xTR, which is the address that the RADIUS Server will see, and the shared secret between the two parts. Also, the `shortname` parameter should be the SSID of the xTR's Wi-Fi network, but this match is not always verified.

Users configuration

For a basic functioning of the RADIUS Server, the users and their authentication data can be stored in a simple text file (`users`), which will store passwords in clear text (`Cleartext-Password`). It's clear that, in the case of this work, a more complex and secure technique has to be used to save users' data. The most obvious choice, and the most common one, is to use a MySQL database. Like FreeRADIUS, MySQL can be easily found on the repository and rapidly installed on the Ubuntu Server machine.

There a few steps that need to be done for deploying a basic MySQL DB for RADIUS.

1. Create a DB for RADIUS and grant privileges to work with it:
   ```
   CREATE DATABASE radius;
   GRANT ALL ON radius.* TO root@localhost IDENTIFIED BY "password";
   ```

2. Edit `/etc/raddb/sql.conf` and enter the server, name and password details to connect to your SQL server and the RADIUS database. The database and table names should be left at the defaults.

3. In `/etc/raddb/radiusd.conf` ensure that the line `$INCLUDE sql.conf` is uncommented.

4. Edit `/etc/raddb/sql.conf`, putting the name of the DB used in the line: `database = "radius"`

5. It's mandatory in `/etc/freeradius/sites-available/inner-tunnel` to uncomment the line starting with `#sql`. If this step is ignored, RADIUS will not read the data obtained from the DB, and will not give Access-Accept even after a correct EAP Handshake.

The standard tables used by RADIUS are:

- `radusergroup`, that contains entries matching a user account name to a group name
- `radcheck`, with an entry for each user account name with a `Cleartext-Password` attribute with a value of their password
- `radreply`, where entries are stored for each user-specific RADIUS reply attribute against their username
- `radgroupreply`, which is filled with attributes to be returned to all members of a given group.

Here is shown the structure of the tables described above. The example below is related to the RADIUS Server of domain B which is storing the credentials for user "alice" (@domainb.com).

- ```
  mysql> select * from radusergroup ;
  +----+--------------------+-----------+----------+
  | id | UserName           | GroupName | priority |
  +----+--------------------+-----------+----------+
  |  1 | alice@domainb.com  | static    |        0 |
  +----+--------------------+-----------+----------+
  ```

- ```
  mysql> select * from radcheck ;
  +----+--------------------+--------------------+-------+-----+
  | id | UserName           | Attribute          | Value | Op  |
  +----+--------------------+--------------------+-------+-----+
  |  1 | alice@domainb.com  | Cleartext-Password | ***   | :=  |
  +----+--------------------+--------------------+-------+-----+
  ```

- ```
  mysql> select * from radreply ;
  +----+--------------------+--------------------+-------------+----+
  | id | UserName           | Attribute          | Value       | Op |
  +----+--------------------+--------------------+-------------+----+
  |  1 | alice@domainb.com  | Framed-IP-Address  | 10.1.2.121  | := |
  +----+--------------------+--------------------+-------------+----+
  ```

- ```
  mysql> select * from radgroupreply ;
  +----+-----------+-----------------+-------------+----+
  | id | GroupName | Attribute       | Value       | Op |
  +----+-----------+-----------------+-------------+----+
  |  1 | static    | Service-Type    | Framed-User | := |
  |  2 | static    | Reply-Message   | *******     | := |
  +----+-----------+-----------------+-------------+----+
  ```

It can be seen that using RADIUS groups suits very well the scenario of this work: it is indeed a feature that eases the managing of big quantity of data, like what happens with users of an ISP. Specifically, we will configure a group for the each EID-prefix, for sending back the same Map-Server's key to every user of that prefix. This is comfortable to be managed (e.g. if the ISP decides to change the Map-Server's key), since it is just one entry in the `radgroupreply` table.

Access Point configuration

As explained, in out small-scope architecture, the xTR will also act as the Access Point, meaning that hosts will directly connect to it. For every host that connects through Wi-Fi the xTR has to authenticate, authorize and account using RADIUS.

The file that must be modified is `/etc/config/wireless`, which contains information on how the wireless network is delivered. It should be edited to contain additional information about the RADIUS Server (IP and shared secret):

```
config wifi-iface
   option device 'radio0'
   option mode 'ap'
   option ssid 'LISP Wi-Fi'
   option encryption 'wpa2'
   # IP of RADIUS server (default ports are OK)
   option auth_server '84.88.81.48'
   option auth_secret 'sharedpsw'
   option acct_server '84.88.81.48'
   option acct_secret 'sharedpsw'
```

```
option nasid 'xTR'
option network 'lan'
```

It's important to mention that we are using the same machine for both authentication and accounting, because they're both managed by RADIUS. We could also use different servers, putting the addresses as in the snippet above.

Proxy configuration

One RADIUS Server per domain is going to be used. We consider having two domains, for simulating the scenario of this work in a minimum scope. Therefore we are deploying two different RADIUS Servers, which have to be able to do RADIUS Proxy with each other. That is, when a user that belongs to domain B is authenticating in a network of domain A, the RADIUS Server of domain A has to communicate and relay all the RADIUS data to the RADIUS Server of domain B.

For doing these, few modifications to the file `proxy.conf` are needed on both RADIUS Servers. For example, on RADIUS Server of domain A:

```
realm "domaina.com" {
    # local queries (username@domaina.com) are treated maintaining the
        string "@domaina.com" in the username
        nostrip
}

realm LOCAL {
    # local queries are not stripped of the domain
        nostrip
}
realm NULL {
}

realm DEFAULT {
}

# configuration for RADIUS Server @ domain B
home_server domainb {
        type = auth+acct
        ipaddr = 84.88.81.48
        port = 1812
        secret = ***
        require_message_authenticator = yes
        response_window = 20
        zombie_period = 40
        revive_interval = 120
        status_check = status-server
        check_interval = 30
        num_answers_to_alive = 3
}
```

```
# we have to define a pool (in this case with just one server)
home_server_pool domainb_pool {
        type = fail-over
        home_server = domainb
        # only one radius server...
}

# binding between domain string ("domainb.com") and server
realm "domainb.com" {
        auth_pool = domainb_pool
        # RADIUS requests are forwarded with "@domainb.com"
        nostrip
}

}
```

The RADIUS Server must be configured also to accept requests incoming from the other RADIUS Servers, apart from the authorized xTRs. So, as done before, we have to add an entry to the `clients.conf` file. For example, for RADIUS Server in domain A:

```
client radiusb {
        ipaddr = 84.88.81.48
        secret = ***
}
```

In this way RADIUS Server of domain A accepts RADIUS Access-Requests from RADIUS Server of domain B, and they authenticate each other through a shared secret.

Start RADIUS

Finally we can start the FreeRADIUS daemon with: `freeradius -X`, which also prints the debug log on the output.

6.2.2 DHCP configuration

After being authenticated, the user gains access to the network, and the first thing that the host device does is requesting an IP address, that is starting the DHCP dialogue. It's necessary first to distinguish the identity of the user against the network, and then to configure the DHCP Server to automatically serve the right configuration. It is required to not modify (reprogram) the DHCP daemon, so the goal is to let the DHCP Server work normally, altering its behaviour dynamically (for every incoming host).

Home and foreign users

LISP-ROAM works by reserving and assigning a fixed EID to every user subscribed to the service. Every xTR is assigned one (or more) EID-prefix(es) to manage in its LISP site. Therefore, when a user is assigned a fixed EID he is also bound to a specific xTR of the domain. The users belonging to a given xTR are called "home users". The concept of "home" is relative to the scope of the xTR, and not to the whole domain. On the other way round, the xTR will be the "home xTR" for the user. When a user roams into another xTR's domain he is a "foreign user", since his EID does not belong to the EID prefix assigned to the xTR. The xTR is a "foreign xTR" for the user.

It is necessary to go deep and understand what this difference implies.

In paragraph 4.2.3 two different approaches have been presented, regarding how to manage moving host with DHCP. The idea for the prototype is to use a sort of mix of both solutions presented:

- When the user is in its "home network", that is the network of his home xTR, his host is given a netmask which corresponds to the mask of the EID-prefix of the network (e.g. a /24). In this way, the host can normally communicate with devices under the same "home network", which is what happens normally.
- When the user moves his device in a "foreign network", it won't get the same netmask of its home network, but a /30 interface (like explained in 4.2.3) which will isolate the device guaranteeing a direct dialogue with the foreign xTR.
- Like shown before in this thesis, the previous xTR will always be notified if a host moved away from its network. If the previous xTR is the home xTR, it will have to perform additional actions apart from the ones shown in the design part of this work:

  The home xTR will broadcast a Gratuitous ARP Reply message (following the Proxy-ARP mechanism) to announce that the moved host will be reachable through the physical address of the home xTR itself. In this way all the hosts that were communicating with the mobile hosts will just redirect the traffic to the home xTR, which will route the traffic (through LISP) to the new location of the mobile host.

  The routing table of the xTR has to be modified, adding a specific /32 route for the moved host. The traffic for the moved host does not have to be routed to the LAN side, but always trough the WAN interface which will be managed with LISP.
- If two hosts, belonging to the same home network, move to the same foreign network they won't be able to reach each other directly at Level 2 (like

what happens in their home network). This is because they will be both assigned a /30 interface, so all the traffic they will send or receive will be routed through the foreign xTR. Note that this mechanism is performed in general for every foreign host under the same network: even if they are in the same physical network, they won't be able to communicate directly with each other, because they are virtually in different /30 networks. So all the traffic gets triangulated with the foreign xTR.

Virtual Network Interface

The hosts are roaming between Wi-Fi networks. The xTRs (which are also Access Points, in this prototype) are receiving all the hosts on the interface that acts as wireless link (WLAN interface). On the WLAN interface the xTR gets first the EAP response from the host, then the DHCP dialogue, then all the traffic it sends. The WLAN interface is the one to modify when the user is foreign, in order to make the handover transparent to the host.

It is fair to assume that the WLAN interface is up and running, and that its network configuration coincides with the EID-prefix of the network. This is in some way mandatory, in order to let every user in the LAN send traffic to the outside through LISP. It has to be said that the necessary condition is that the EID-prefix is *at least* a subset of the network domain, if we want *some* users to be considered in a LISP site. In the scenario considered for this work, the Wi-Fi network of the xTR will be considered a whole LISP site, so the address and netmask of the WLAN interface has to coincide with the EID-prefix configuration. For example:

- `EID-prefix:  10.1.1.0/24`
- ```
  wlan0   Link encap:Ethernet HWaddr **:**:**:**:**:**
          inet addr:10.1.1.254 Bcast:10.1.1.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          ...
  ```

This is the basic configuration, when no foreign users are in the network.

When a foreign user arrives, as explained above, a /30 interface has to be reserved for him. The foreign user will be able to exchange traffic with the xTR, which will get a new address on the new interface. This behaviour is achieved in two steps: correct DHCP configuration (shown in the next sub-paragraph) and **virtual network interfaces**.

The foreign user, like the home ones, will communicate through his WLAN interface with the OpenWRT, which also will receive data on its WLAN interface. The WLAN interface on the OpenWRT is already set to work on a /24 prefix. The WLAN interface has to be able to communicate with a user which EID is not part of the /24 prefix (e.g. 10.1.2.121). We create a virtual network interface, that is a
  ```

interface that logically exists at Level 3 but it's physically represented by the same Level 2 interface (in this case, the WLAN interface). The concept is also known as *IP aliasing*, because it actually adds an IP address to a physical interface.

For our purposes, for every foreign user authenticated in a network, the WLAN interface will set up a virtual interface with a /30 netmask and a new IP address which will be decided based on the user's EID. This is due to what already explained in 4.2.3, which leads to having a user with a fixed EID that is included in certain ranges. In every /30 range, two addresses are actually available to be used: the xTR will pick up the one that is not used by the user and set it as its own new alias IP.

Setting up a virtual network interface is done just through one command:

```
ifconfig wlan0:1 10.1.2.122
 netmask 255.255.255.252
 broadcast 10.1.2.123
```

In this example the foreign user's EID is 10.1.2.121, so the prefix will be 10.1.2.120/30 (10.1.2.120-123):

- 10.1.2.120 is the network address
- 10.1.2.123 has to be the broadcast address
- 10.1.2.121 is the user's EID
- The xTR must pick up 10.1.2.122 as alias on the new interface

As a result, the new interface created will be:

```
wlan0:1   Link encap:Ethernet HWaddr **same as the WLAN**
     inet addr:10.1.2.122 Bcast:10.1.2.123 Mask:255.255.255.252
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

The next step will be giving the user a correct configuration, with DHCP, to be able to communicate with the xTR on its new interface.

DNSmasq configuration

OpenWRT comes with **DNSmasq**, which provides two services: a DNS service and a DHCP service. In this work, the DNS service is not necessarily considered. "*Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server. It is designed to provide DNS and, optionally, DHCP, to a small network. [...] The DHCP server integrates with the DNS server and allows machines with DHCP-allocated addresses to appear in the DNS with names configured either in each host or in a central configuration file. Dnsmasq supports static and dynamic DHCP leases and BOOTP/TFTP/PXE for network booting of diskless machines.*" [20]

DNSmasq works with a daemon running in background, so we can say that the DHCP Server is embedded in the xTR.

The file that holds the configuration of DNSmasq is `/etc/dnsmasq.conf`.

This file must store the static bindings of the users (both home and foreign). A static binding is established between the MAC address of the device and the IP (in this case, EID) that has to be assigned. It's mandatory to keep trace of the MAC address of the user, and bind it to the EID learned through RADIUS authentication. Once this information is gathered, it has to be checked if the EID is part of the EID-prefix of the xTR:

- If it is (home user) the static MAC-to-EID binding will be added, and the host will be given the DHCP parameters of the home network (e.g. /24 netmask)
- Else (foreign user) the static binding will be added, and a **new configuration** has to be created specifically for the new user:

  Router address will be the xTR's IP on the new virtual interface

  Broadcast address will be the one reserved on the new virtual interface

  Netmask will be /30

One powerful feature of DNSmasq is the "tag" mechanism. We are able to define DHCP configurations for a particular tag name, and assign this tag to multiple users in the configuration file (it resembles how RADIUS group reply works).

Since all home users will have the same configuration (same router address, netmask, broadcast address and few more), we can tag this configuration as "home", adding this snippet to the configuration file:

```
dhcp-option=tag:home,3,10.1.1.254    # Router
dhcp-option=tag:home,54,10.1.1.254    # DHCP Server Id
dhcp-option=tag:home,1,255.255.255.0    # Netmask
dhcp-option=tag:home,28,10.1.1.255    # Broadcast
dhcp-option=tag:home,6,10.1.1.254    # Domain Server
dhcp-option=tag:home,15,home    # LAN domain name
```

When a user connects to the network, if he's recognized as a home user (checking the EID), the only action to do is to add a static lease for him:

```
# home user 10.1.1.5
dhcp-host=00:aa:bb:cc:dd:ee,set:home,10.1.1.5
```

In the case of a foreign user, we add a piece a configuration for him, along with the static lease. For example, user 'alice@domainb.com' (RADIUS authenticated)

with EID 10.1.2.121:

```
# alicedomainb.com START (10.1.2.120/30
dhcp-host=00:0D:88:65:5A:5D,set:alicedomainb.com,10.1.2.121
dhcp-option=tag:alicedomainb.com,3,10.1.2.122
dhcp-option=tag:alicedomainb.com,54,10.1.2.122
dhcp-option=tag:alicedomainb.com,1,255.255.255.252
dhcp-option=tag:alicedomainb.com,28,10.1.2.123
dhcp-option=tag:alicedomainb.com,6,10.1.2.122
dhcp-option=tag:alicedomainb.com,15,alicedomainb.com
# alicedomainb.com END (10.1.2.120/30)
```

The user will be correctly identified by DNSmasq through the MAC address of the host device. The configuration related to the new /30 WLAN interface will be *automatically* sent back in the DHCP ACK message, and the host will normally set up its connection.

The last step is to enable the WLAN interface to receive DHCP Requests:

```
dhcp-range=wlan0,10.0.0.1,10.255.255.254,255.0.0.0,5m
```

The entry contains the name of the interface, the starting and ending address of the IP address pool, and the expire time of the DHCP lease. Every static IP address is checked before being given to the user: if it is not part of the address pool of the interface it is ignored. So it is obligatory to assign the "widest" IP range possible to the WLAN interface, further more an EID can be *any* IP address since it is routable only in a local scope. What should be done is tell DNSmasq to serve every IP address possible on the WLAN interface.

Unfortunately it is not possible to obtain this behaviour with DNSmasq. If we try to assign the range [0.0.0.0 - 255.255.255.255] to the WLAN interface, this range gets ignored as no IP address will be server on the WLAN.

The widest range to work with is a /8 [1], so the WLAN has been assigned the range 10.0.0.0/8, as can be seen in the above snippet.

We can still assume that a /8 range approximates quite well a realistic scenario, and it is not a hazardous assumption for testing.

[1] It is important to state that this is just an experimental result, and further studies should be addressed to understand if this behaviour is just a constraint of the DNSmasq sofware or if there are additional constraints in the DHCP mechanism that have to be considered.

Chapter 7

LISP-ROAM implementation

The core of the work consists in modifying the LISPmob distribution for the purposes of this work, making the xTR able to correctly handle mobile hosts. As we deeply explained, the flow of the user's interactions with the xTR can be summed up in these parts:

1. RADIUS authentication
   - The xTR learns user's EID
   - The xTR learns user's Map-Server key
   - The xTR learns user's MAC address (needed for DHCP configuration)
2. DHCP configuration
   - The xTR creates a local virtual interface for the new EID
   - DHCP Server sets a static IP (user's EID) bound to user's MAC, and DHCP configuration for the new interface
3. LISP
   - The xTR discovers user's Map-Server address
   - The xTR registers user's new binding

These actions are considered to take place in case of a **foreign** user.
If the user connects to its **home** network, the steps are simplified:

1. RADIUS authentication
   - The xTR learns user's EID
   - The xTR learns user's MAC address (needed for DHCP configuration)

2. DHCP configuration

   DHCP Server sets a static IP (user's EID) bound to user's MAC and tags it as home user

3. LISP

   The xTR registers user's new binding

All of the user data will be stored in a structure representing the important information about the user:

```
typedef struct user_info {
  char *username;
  char eid[INET_ADDRSTRLEN];
  char mac[18];

  char ms_address[INET_ADDRSTRLEN];
  char *ms_key;
  uint64_t ms_nonce;

  int wlan_id;

  int foreign; // 0 = HOME user, 1 = FOREIGN user
} user_info;
```

All the fields will be gradually explained in this chapter. We also use a **vector** (which code will not be reported here) to store the data of all the users connected to the network.

Keep figure 7.1 as a guideline while going in deep in every single steps in each of the following paragraphs. Figure 7.1 depicts the sequence of steps that need to be performed when dealing with a foreign user that connects to the network for the first time.

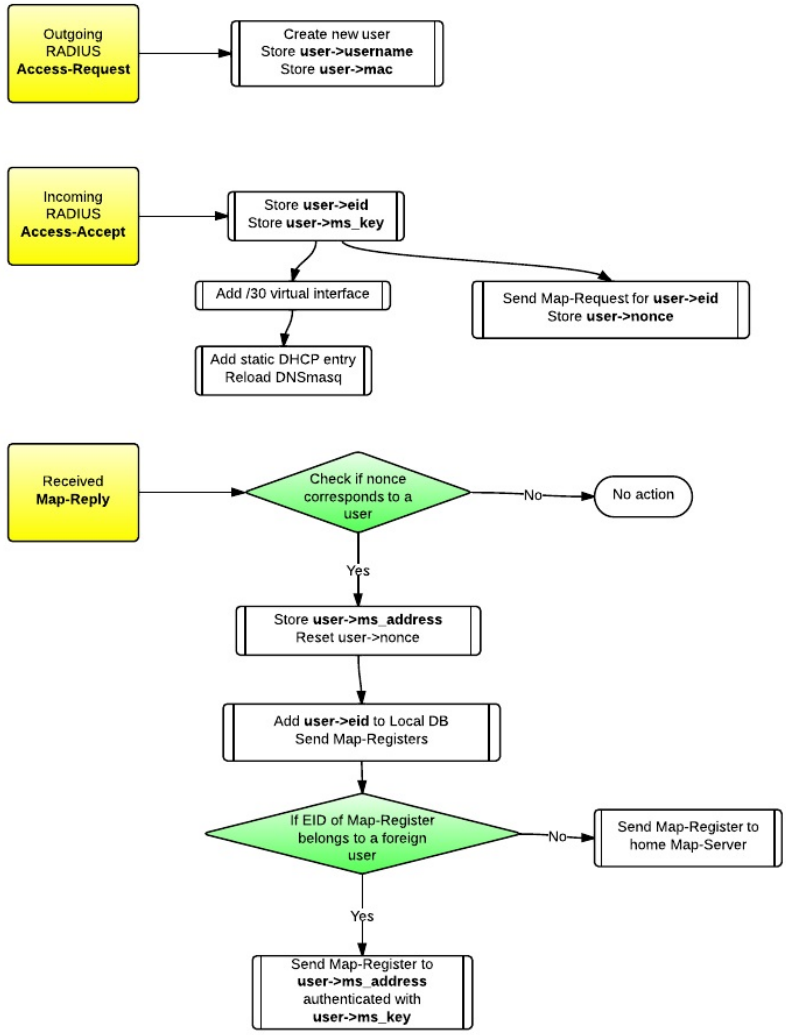


Figure 7.1. LISP-ROAM flow - Foreign unknown user case

7.1 RADIUS outgoing and incoming traffic

After setting correctly the xTR and the RADIUS Server, every time a host is trying to access the network (communicating through EAP with the xTR's WLAN interface), the xTR will exchange RADIUS packets with the RADIUS Server in order to verify and authenticate the supplicant. These packets contain important information about the user, which the xTR has to learn. In our particular interest, the messages are: **RADIUS Access-Request** (sent from the xTR to the RADIUS Server) and **RADIUS Access-Accept** (viceversa).

We take advantage of what's already deployed in LISPmob, most of all the TUN interface. The TUN interface uses RAW sockets, and it is mapped to every address. When LISPmob starts the routing table looks like this:

```
root@andrea: # ip route
0.0.0.0/1 dev lispTun0 proto static
default via 192.168.0.1 dev eth0 metric 100
128.0.0.0/1 dev lispTun0 proto static
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.151
```

As it can be seen, `lispTun0` (the TUN interface used by the daemon `lispd`) is mapped on every address possible, and therefore intercepts every packet that sent or received by the xTR. LISPmob adds two entries to the routing system (`0.0.0.0/1` and `128.0.0.0/1`). These two cover the whole address space, and are automatically removed when LISPmob terminates (as the TUN interface is deleted). This is has been considered an easier approach than re-configuring the default route, which is still there (`default via 192.168.0.1`), as can be clearly seen above.

Better, all the outgoing traffic (the traffic generated from the LAN) will be processed by the TUN interface. The incoming traffic (from the WAN) will be received on the LISP ports. Normal LISPmob only writes the incoming traffic into the TUN interface in order to introduce it to the internal routing system (since the inner packet has a more specific address, it would be routed to the appropriate LAN interface, and would not return to the TUN interface).

So it is already possible to intercept all outgoing and incoming traffic, before it is filtered and modified by LISPmob: it is possible to intercept RADIUS traffic as it takes place, without impacting on the dialogue itself. Like said before, we need to intercept Access-Request and Access-Accept packets, and read the data we care about.

Below are reported the modifications applied to the original LISPmob code. We try not to go too much in the detail, limiting the lines of code shown to the minimum, only to understand what has been done.

7.1.1 RADIUS Access-Request

The user tries to access to the network, and the xTR checks if the authentication data is correct communicating with the RADIUS Server. Hence, the first sensitive information that needs to be obtained is contained in the packets sent from the xTR to the RADIUS Server.

For reading the outgoing traffic, we modify the content of the function `send_packet(...)` in the file `lispd_sockets.c`. These function is called everytime a packet needs to be sent on the WAN side, both LISP or non-LISP.

Every packet received in the RAW socket is stripped of the Level 2 header, therefore it points to the IP header. We have to move the pointer to read the UDP header:

```
struct iphdr iph = (struct iphdr *) packet;
struct udphdr *udph;

if (iph->protocol == 17) // next header is UDP
{
  if(iph->version == 4){
    /* With input RAW UDP sockets in IPv4, we get the whole external
        IPv4 packet */
    udph = (struct udphdr *) (((uint8_t *) packet + sizeof(struct
        iphdr)));
  }
  ...
}
```

We have to check if the transport layer is UDP and if the destination port is the RADIUS one (1812):

```
/* RADIUS outgoing packet - START */
if (iph->protocol == 17 && htons(udph->dest) == RADIUS_PORT)
{
...
```

Then we can parse the RADIUS packet, to check if it is an Access-Request:

```
struct radius_packet *rpacket = (struct radius_packet *) (((uint8_t *)
     udph + sizeof(struct udphdr)));

if (rpacket->code == RADIUS_CODE_ACCESS_REQUEST)
{
...
```

At this point, we have to read the attributes of the packet, which are holding the user data. It is necessary to read:

- The username (attribute `User-Name`)
- The MAC address (attribute `Calling-Station-Id`)

```
char *username;
char mac[18];

struct radius_attribute *rattribute = rpacket->attrs;
while(rattribute != NULL && rattribute->type != 0)
{
  switch(rattribute->type) {
    // User-Name (type=1) in rattribute
    case 1: ;
      strncpy(username, rattribute->value, rattribute->length -2);
      username[rattribute->length -2] = '\0';

      break;

    // Calling-Station-Id (type=31) in rattribute
    case 31: ;
      strncpy(mac, rattribute->value, rattribute->length -2);
      mac[rattribute->length -2] = '\0';

      break;

    default: break;
    }

  // if we read username and MAC address
  if (strlen(username) != 0 && strlen(mac) != 0)
    break;
  else // go on reading...
    rattribute = (struct radius_attribute *) CO(rattribute, rattribute
        ->length);
}
```

The username and MAC address of the supplicant user have to be stored inside a new `user_info` structure, which will be added to the users vector (called **USERS_INFO**):

```
user_info *ui = (user_info*) malloc(sizeof(user_info));
ui->username = (char *) malloc(sizeof(username));
strcpy(ui->username, username);
strcpy(ui->mac, mac);

vector_add(&USERS_INFO, ui);
```

7.1.2 RADIUS Access-Accept

The next packet to be intercepted is the Access-Accept sent back (if the authentication is correct) by the RADIUS Server. Since this is an incoming packet, the function `process_input_packet(...)` inside the `lispd_input.c` file is modified. There are few steps in common with the procedure followed for the Access-Request:

parsing the IP header, then the UDP header, checking if the packet is RADIUS, and if it is an Access-Accept. After this, the list of attributes is read, to find:

- The username (attribute `User-Name`)
- The EID (attribute `Framed-IP-Address`)
- The key of the Map-Server (attribute `Reply-Message`)

```
uint8_t *eid;
char eid_str[20];
char lisp_key[50];
char username[50];
struct radius_attribute *rattribute = rpacket->attrs;
while(rattribute != NULL && rattribute->type != 0)
{
  //lispd_log_msg(LISP_LOG_INFO, "RADIUS attribute type = %d",
      rattribute->type);

  switch(rattribute->type) {
    // User-Name (type=1) in rattribute
    case 1: ;
      ...
    // Framed-IP-Address (type=8) in rattribute
    case 8: ;
      eid = (uint8_t * )ntohl(rattribute->value);
      sprintf(eid_str, "%u.%u.%u.%u", eid[0], eid[1], eid[2], eid[3]);
      break;

    // Reply-Message (type=18) in rattribute
    case 18: ;
      strncpy(lisp_key, rattribute->value, rattribute->length -2);
      lisp_key[rattribute->length -2] = '\0';
      break;

    default: break;
    }

  // if we read everything
  if (strlen(username) != 0 && strlen(lisp_key) != 0 && strlen(eid_str
      ) != 0)
    break;
  else // go on reading...
    rattribute = (struct radius_attribute *) CO(rattribute, rattribute
        ->length);
}
```

The username is used for matching the Access-Request against the Access-Accept. Therefore we add the information obtained to the user already stored in the vector:

```
user_info *user = vector_search_username(&USERS_INFO, username);
user->ms_key = (char *) malloc(sizeof(lisp_key));
strcpy(user->ms_key, lisp_key);
strcpy(user->eid, eid_str);
```

7.2 User's network configuration and location update

7.2.1 Local interface and DHCP

The function `andrea_add_wlan(user_info *user)` performs the steps explained in paragraph 6.2.2 for what concerns the creation of the virtual local interface and the configuration of DNSmasq.

1. Add new WLAN virtual interface (`ifconfig wlan0:X ...`)
2. Add DHCP entry in `/etc/dnsmasq.conf`
3. Reload DNSmasq daemon, to apply the changes

At the end of the procedure the attribute `wlan_id` for the user is filled with the number of the virtual interface `wlan0:[wlan_id]`.

It is important to declare that we made use of the `system(...)` function, using direct system calls to the kernel. A cleaner way would have been using NETLINK to communicate with the kernel ([16]), but since LISP-ROAM is working embedded in the OpenWRT is quite fair to consider the code written just for this type of device, excluding portability.

7.2.2 Retrieve user's Map-Server

If the user is foreign, it will belong to a different network, which means that his bindings are kept by another Map-Server. In order to update user's EID-to-RLOC mapping, the xTR has to learn the address of the foreign Map-Server, and then send him a Map-Register message. In this prototype we take advantage of the LISP infrastructure to do this:

1. xTR sends a Map-Request for the foreign user's EID
2. xTR receives a Map-Reply from the foreign Map-Server
3. xTR reads the outer IP address of the packet, which is the address of the Map-Server

The Map-Server replies to Map-Requests only if the mappings is registered with the `Proxy bit` active, which has to be done by every xTR of the solution. In LISPmob, we have to declare which is the Map-Server the xTR refers to in the configuration file `/etc/config/lispd`. In the same part of the configuration it is possible to set the Proxy-Reply behaviour.

```
config 'map-server'
    option 'address'     '84.88.81.2'
    option 'key_type'    '1'
    option 'key'         '***'
    option 'proxy_reply' 'on'
```

When LISPmob boots up, a Map-Register for each EID-prefix declared in the file is sent to the Map-Server with the P bit on.

In this prototype, after a correct user authentication and the completion of the local interface and DHCP setup. The xTR will spontaneously send a Map-Request, correctly filled. This is done with the function `andrea_send_map_request(user_info *user)`:

```
// retrieve ITR's EID
lisp_addr_t *home_eid = &(get_head_interface_list()->iface->
    head_mappings_list->mapping->eid_prefix);
// build foreign user's EID
lisp_addr_t  *dest_eid = malloc(sizeof(lisp_addr_t));;
get_lisp_addr_from_char(user->eid, dest_eid);
// nonce returned by the Map-Request
uint64_t nonce;

// build the mapping to insert in the Map-Request
lispd_mapping_elt *mapping = (lispd_mapping_elt *)malloc(sizeof(
    lispd_mapping_elt));
mapping->eid_prefix = *dest_eid;
mapping->eid_prefix_length = 32;
mapping->iid = -1;
mapping->locator_count = 0;
mapping->head_v4_locators_list = NULL;
mapping->head_v6_locators_list = NULL;

// send Map-Request
build_and_send_map_request_msg(
    mapping,
    home_eid,
    get_map_resolver(),
    1,
    0,
    0,
    0,
    &nonce);
```

```
// assign nonce to user
user->ms_nonce = nonce;
```

ITR's EID, that is the address of the xTR on the internal LAN (the EID, indeed) is retrieved as the first element of the local EID-to-RLOC mappings assigned to the WAN interface. The user's mapping requested is a normal /32 EID bound to no locators.

After the Map-Request is sent, the nonce of the message is stored in the user structure. This is because the xTR will receive the Map-Reply (from the Map-Server), and will use the nonce to retrieve the user that has to be assigned to that Map-Server. The Map-Reply packet is normally received and processed by LISP, so it is to be intercepted in the `process_input_packet(...)` in `lispd_input.c` function:

```
if(ntohs(udph->dest) == LISP_CONTROL_PORT)
{
    lispd_pkt_map_reply_t *pkt = (struct lispd_pkt_map_request_t *) CO
        (udph, sizeof(struct udphdr));
    // check if it a Map-Reply
  if (pkt->type == LISP_MAP_REPLY)
  {
    // check if the nonce corresponds to a user
    user_info *user = (user_info *) vector_search_nonce(&USERS_INFO,
        pkt->nonce);
    if (user != NULL)
    {
      strcpy(user->ms_address, get_char_from_lisp_addr_t(
          extract_src_addr_from_packet(packet)));
      user->ms_nonce = -1; // Reset after use

      andrea_send_map_register(user);
    }
  }
}
```

A function arranged for the vector is used to retrieve the user related to the nonce. It is fair to assume that the nonce is unique in the vector, since it is a random number, and because it is resetted after the procedure. The Map-Server's address is extracted from the source IP header of the packet and stored in the user structure.

Now the user structure is completed, all the fields are filled with correct data. As it can be seen, the function for updating the location of the user is automatically called.

7.2.3 User's location update

The function `andrea_send_map_register(user_info *user)` includes the actions needed for correctly updating the foreign user's EID. The steps are just two:

1. Add the new EID to the local DB. In this way the EID is really considered to be part of the network.

   This is done with the LISPmob function `add_database_mapping(...)`

2. Immediately send a Map-Register for the new /32 EID

```
// we send a Map-Register ONLY for the new EID
  lisp_addr_t user_eid;
  get_lisp_addr_from_char(user->eid, &user_eid);
  lispd_mapping_elt *mapping = new_local_mapping(user_eid, 32,
      -1);
  lispd_locator_elt *locator = new_local_locator (
      get_head_interface_list()->iface->ipv4_address,&(
      get_head_interface_list()->iface->status),
            1, 100, 255, 0, get_head_interface_list()->iface->
                out_socket_v4);

  add_locator_to_mapping (mapping,locator);
  calculate_balancing_vectors (mapping,&((
      lcl_mapping_extended_info *)mapping->extended_info)->
      outgoing_balancing_locators_vecs);

  build_and_send_map_register_msg(mapping);
```

3. After this, the EID (as included in the local DB) will be periodically registered to the Map-Server (LISPmob function `map_register(...)`)

   This function has been slightly modified in order to not send *every* Map-Register to the home Map-Server: the Map-Registers for foreign users will be sent at the address contained in the related user structure (`user_info->ms_address`):

```
// Look for specific Map-Server (foreign user)
lispd_map_server_list_t *ms = vector_get_map_server(&USERS_INFO,
    get_char_from_lisp_addr_t(mapping->eid_prefix));

...

/* Send the map register */
send_udp_ctrl_packet(ms->address,LISP_CONTROL_PORT,
    LISP_CONTROL_PORT,(void *)map_register_pkt,packet_len);
```

7.3 Flow optimization

All the actions described above are considered to take place in the case of a *foreign* user that connects to the network *for the first time*. We can say this is the worst-case scenario, that is the one that introduces more latency due to the actions that need to be executed.

7.3.1 Known users

Once a foreign user is authenticated, configured for the network, and updated in the LISP Mapping System he is ready for communicating with other LISP sites. His data are kept stored in memory inside a `user_info` structure organized in a vector.

When this user disconnects - or changes network, or in general leaves the network - and comes back after a period of inactivity he does not have to go through the same steps as the first time. His data are still in memory, as long as LISProam is running.

As depicted in figure 7.2 few steps are skipped (compared to figure 7.1).

This is a critical point for what concerns latency and packet loss. When the mobile host is exchanging data with a correspondent node and performing an handover at the time, packets gets dropped before the mobile host regains connection. If the user is already known by the network this accidental loss is sensibly reduced.

The most important missing step is that it is not necessary to retrieve the Map-Server's address anymore: the Map-Register for updating the user's location is automatically sent after receiving a correct RADIUS Access-Accept.

Even the DHCP configuration part does not have to take place: indeed, DNSmasq configuration file is cleaned only when LISProam process is shut down, therefore the static lease for the foreign user is still available.

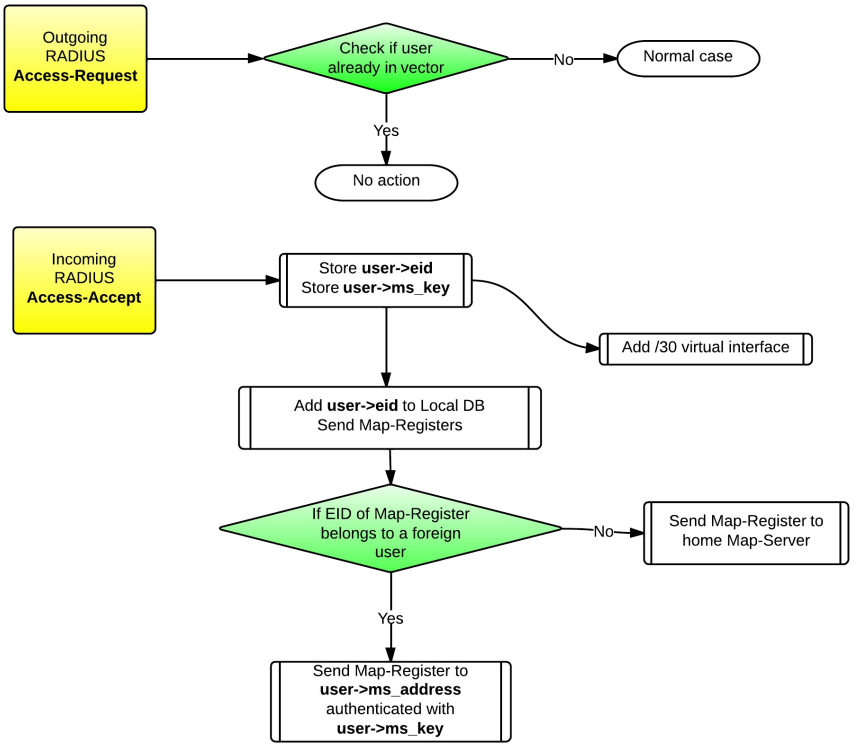


Figure 7.2. LISP-ROAM flow - Foreign known user case

7.3.2 Home users

In the case of home users some steps can be ignored, too. Since the users don't have to be hosted in the network, it is needless to know the address of the user's Map-Server and its key, because they will obviously be the same as the home Map-Server. So there's no more the need the retrieve the address the address of the Map-Server through the Mapping System.

For what concerns the DHCP part, the static lease has always to be added (if not already present). This means reloading DNSmasq, which can introduce latency. Anyway, the configuration to add to the configuration file is just the static lease, and no "tag" section since the user will belong to the "home" tag.

Also no /30 virtual interface will be created, since the user will be part of the home EID-prefix (usually a /24), assigned to the WLAN interface of the xTR.

Since the home user's address is part of the /24 home EID-prefix, his EID (/32) does not have to be specifically added to the local DB. Also, it does not have to be specifically registered in the Mapping System, since it will be reached as part of the /24 EID-prefix. But the xTR does not know if the user was connected to another network before, because it is not sending any Map-Request to the Map-Server (also for optimizing the bandwidth usage). An idea for not introducing latency (as it would be with a Map-Request) is sending a specific Map-Register for the /32 EID of the incoming user, without adding the /32 EID to the local DB. In this way, the specific Map-Register is sent only once the user arrives and then not anymore. So if the user was connected to another network before, the Map-Server will directly send a Map-Notify to the previous xTR that registered the /32 EID of the home user (if there is one).

This does not happen if the home xTR registers just the /24 EID-prefix, since it is wider than a /32 one, so the Map-Server will keep the old binding for the home user.

All of the cases depicted are grouped in table 7.1, that shows which steps can be skipped depending on the type of user: home or foreign, known or unknown.

Table 7.1. Steps performed based on type of user

	Type of user	
	Unknown	**Known**
Foreign	All steps performed	No Map-Server retrieval No DHCP update
Home	No Map-Server retrieval No virtual interface	No Map-Server retrieval No virtual interface No DHCP update

7.3.3 Different devices

LISP-ROAM works with fixed EIDs. This creates a static and dynamically unchangeable binding between the user and its EID, which introduces a big constraint. In a fairly realistic scenario a user uses his credentials to connect to the same network with multiple devices, which is usually allowed (e.g. eduroam). This is not possible in LISP-ROAM. A future development, with dynamic EIDs, has already been proposed in the design part of this thesis, but it is not deployed in this prototype.

So we are considering a user connecting with one device at a time, which will be assigned the reserved EID.

In order to make this possible there is a little modification that needs to be made, regarding the filling of the fields of the user structure: when the user connects, and he is already known by the network, it is mandatory to check if the MAC address is the same - that is, if the connection is requested by the same device. If not (e.g. the user was connected with the PC and, after shutting down the PC, it is connecting with his smartphone) the MAC of the user structure will be updated with the new one, so all the previous info already stored in memory won't get lost and the user will be automatically recognized by the xTR.

This implies that the old static DHCP entry has to be removed, and a new one will be added with the EID bound to the new MAC.

This feature is a constraint but can also lead to future studies aimed to guarantee seamless mobility through devices. Being sure of having the same EID assigned to different hosts can be exploited to support cross-device session continuity. For example, a video call (TCP flow) started on the PC *may be* moved to the mobile phone that will be given the same EID.

7.4 Handover

The core feature of the LISP-ROAM proposal, and also of this whole work, is being able to maintain user connectivity on a host roaming across networks. In the last paragraph, it has been clearly explained which are the actions (and the sub-cases) when a user connects to a network. But it has not been shown what happens when a user, connected to a network, changes his attaching point.

Like deeply explained in paragraph 5.1.5, what happens when the Map-Server receives a Map-Register for a /32 EID-prefix that was already registered. What has to be done is to intercept the Map-Notify spontaneously sent by the Map-Server, check if it is about a moved host and then start the Correspondent Nodes update part.

Like other messages, the Map-Notify can be intercepted through the function

`process_input_packet(...)`. The code representing how the handover is detected is not reported. The mechanism is quite simple: we have to check if the Map-Notify mappings contain an EID which is one of the users in the network and the locators are different the local xTR ones.

When this condition is verified:

```
// user corresponding to the moved EID
user_info *ui = vector_search_eid(&USERS_INFO, eid_str);

// we must "mark" the interface in order to send SMRs
lispd_iface_list_elt *iface_list = get_head_interface_list();
iface_list->iface->status_changed = TRUE;

// send SMRs to all CNs
init_smr(NULL, NULL);

// remove moved_host's_mapping from local DB
lisp_addr_t moved_eid;
get_lisp_addr_from_char(ui->eid, &moved_eid);
del_mapping_entry_from_db(moved_eid, 32);

// we must add new moved_host_mapping to Map-Cache -> send a Map-
    Request
lispd_mapping_elt *moved_host_mapping = (lispd_mapping_elt *)malloc(
    sizeof(lispd_mapping_elt));
lisp_addr_t host_eid;
get_lisp_addr_from_char(eid_str, &host_eid);
moved_host_mapping->eid_prefix =  host_eid;
moved_host_mapping->eid_prefix_length = 32;
moved_host_mapping->iid = -1;
moved_host_mapping->locator_count = 0;
moved_host_mapping->head_v4_locators_list = NULL;
moved_host_mapping->head_v6_locators_list = NULL;
int nonce;
build_and_send_map_request_msg(
  moved_host_mapping,
  home_eid,
  get_map_resolver(),
  1,
  0,
  0,
  0,
  &nonce);

// Remove virtual WLAN
char command[100];
sprintf(command, "ifconfig %s:%d down", WLAN_INTERFACE, ui->wlan_id);
system(command);
```

It's obvious that this code refers to the case when a foreign user moves in another network. Few steps are simplified in the case of home users (no virtual

interface removal, no deletion from local DB).

When the correspondent node receives a Solicit-Map-Request, it automatically sends a Map-Request, to update the binding. After this, the connections with the moved host are normally restored and running through LISP.

Chapter 8

Test bed

The test bed deployed follows the one depicted for the prototype. Figure 8.1 represents in the detail the real architecture used. xTR A and xTR B represents one domain each (domain A and domain B, respectively). LISProam is running on both the xTRs. The user will move back and forth in these two networks. xTR C is not running LISProam, but normal lispmob (0.3.3), and has just one user in his network, which will be used as correspondent node. Tests are made in order to check if the connection between the mobile host and the correspondent node (which does not move) is maintained after handover between network A and B. The mobile host is "alice@domainb.com", which implies that domain B is considered to be the home network in the test scenario, and domain A will be the foreign.

Like explained previously, we provide one RADIUS Server for every domain and one LISP Map-Server/Map-Resolver shared between domains.

As can be seen in the figure, all the components are placed under the same network (the public IP addresses are part of the same pool) which simplifies a little the realistic scenario considered. In theory, having different domains should imply also having different IP address blocks. We can say that the latency experimented in this tests will be considered lower than the one that would actually be using two physically distinguished domains.

The steps of the test are summarized like this:

1. User (`alice@domainb.com`) connects to the home Wi-Fi network (**LISP-B**)

2. User is assigned the fixed EID (10.1.2.121) and the home DHCP configuration (/24 netmask)

3. User initializes a connection (on whichever transport protocol) with the correspondent node (10.1.3.165)

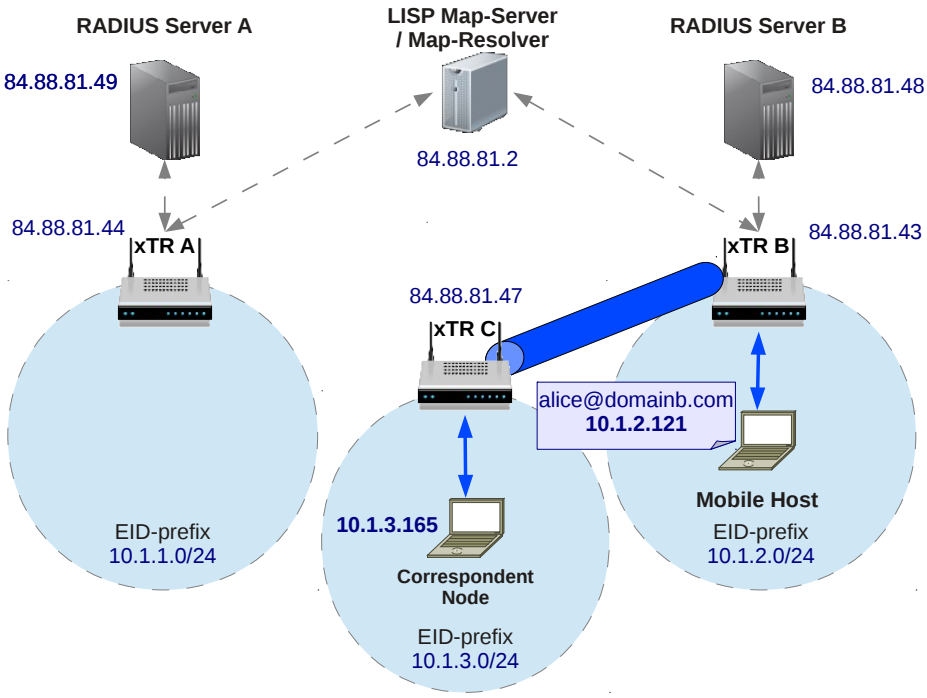


Figure 8.1. Test bed architecture

4. At a certain point, the user decides to connect to the foreign Wi-Fi network (**LISP-A**), obtaining a new DHCP configuration (/30 netmask, new router address, ...)

5. After a short period of inactivity, the connection with the correspondent node is resumed

8.1 Experimental results

In order to understand deeper what really happens in the case we are studying, the output logs of LISProam are shown in this section. In particular, there are the logs taken from xTR A, B and C.

The case tested is user `'alice@domainb.com'` connecting to LISP-B (home network) and then switching to LISP-A.

xTR B (LISP-B)

LISProam starting screen.

```
INFO: LISPmob (0.3.2): 'lispd' started...
INFO:
************
**LISProam**
************
INFO: Waiting for connections on interface: wlan0
```

User's authentication start.

```
DEBUG:  LISProam: Outgoing RADIUS Access-Request packet
DEBUG:  LISProam: Outgoing RADIUS packet -> User-Name: alice@domainb.
    com
DEBUG:  LISProam: Outgoing RADIUS packet -> Calling-Station-Id: 00:0D
    :88:65:5A:5D
INFO:   LISProam: !! Authentication started for user 'alice@domainb.
    com' !!
```

User's authentication completed.

```
DEBUG:  LISProam: Incoming RADIUS Access-Accept packet
DEBUG:  LISProam: Incoming RADIUS packet -> Framed-IP-Address:
    10.1.2.121
DEBUG:  LISProam: Incoming RADIUS packet -> Reply-Message: ***
DEBUG:  LISProam: Incoming RADIUS packet -> User-Name: alice@domainb.
    com
INFO:   LISProam: !! Authentication completed for user 'alice@domainb.
    com' !!
```

DHCP update for new user.

```
INFO:   LISProam: Adding DHCP entry for home user 'alice@domainb.com'

INFO:   LISProam: Reloading DHCP Server

Nov 25 11:56:58 dnsmasq[20608]: started, version 2.62 cachesize 150
Nov 25 11:56:58 dnsmasq[20608]: compile time options: IPv6 GNU-getopt
    no-DBus no-i18n no-IDN DHCP no-DHCPv6 no-Lua TFTP no-conntrack
Nov 25 11:56:58 dnsmasq-dhcp[20608]: DHCP, IP range 10.0.0.1 --
    10.255.255.254, lease time 5m
Nov 25 11:56:58 dnsmasq[20608]: using local addresses only for domain
    lan
Nov 25 11:56:58 dnsmasq[20608]: reading /tmp/resolv.conf.auto
Nov 25 11:56:58 dnsmasq[20608]: using nameserver 8.8.8.8#53
Nov 25 11:56:58 dnsmasq[20608]: using local addresses only for domain
    lan
```

```
Nov 25 11:56:58 dnsmasq[20608]: read /etc/hosts - 1 addresses
```

User is a home user, so there's no need to retrieve the Map-Server.

```
INFO:    LISProam: Map-Server already known for user 'alice@domainb.com
    ' (home user)

         --- User info ---
        > username: alice@domainb.com
        > eid: 10.1.2.121
        > mac: **:**:**:**:**:**
        > HOME user
         --- end ---
```

User's location update. Even if the user is booting up in his home network, his /32 EID has to be registered once in the Mapping Sysem.

```
INFO:    LISProam: Sending Map-Register for new EID (10.1.2.121)

DEBUG: Sent Map-Register message for 10.1.2.121/32 to Map Server
    84.88.81.2
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Notify message
DEBUG: Map-Notify message confirms correct registration
DEBUG-2: Completed processing of LISP control message
```

DHCP dialogue (DNSmasq log).

```
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 available DHCP range:
    10.0.0.1 -- 10.255.255.254
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 DHCPDISCOVER(wlan0)
    10.1.2.121 **:**:**:**:**:**
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 tags: home, known,
    wlan0
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 DHCPOFFER(wlan0)
    10.1.2.121 **:**:**:**:**:**
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 next server: 10.1.2.254
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  1 option:
    53 message-type  2
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    54 server-identifier  10.1.2.254
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    51 lease-time  300
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    58 T1  150
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    59 T2  262
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  7 option:
    15 domain-name  domainB
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    6 dns-server  10.1.2.254
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    28 broadcast  10.1.2.255
```

```
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    1 netmask  255.255.255.0
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 sent size:  4 option:
    3 router  10.1.2.254
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 DHCPREQUEST(wlan0)
    10.1.2.121 00:0d:88:65:5a:5d
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 tags: home, known,
    wlan0
Nov 25 11:57:00 dnsmasq-dhcp[20608]: 751685703 DHCPACK(wlan0)
    10.1.2.121 **:**:**:**:**:**
```

As the connection with correspondent node starts, a Map-Request is sent in order to obtain correspondent node's mapping and then redirect traffic on LISP.

```
DEBUG: No map cache retrieved for eid 10.1.3.165
DEBUG-2: Added map cache entry for EID: 10.1.3.165/32
DEBUG: Sent Map-Request packet for 10.1.3.165/32: Encap: Y, Probe: N,
    SMR: N, SMR-inv: N
DEBUG-2: get_rloc_from_balancing_locator_vec: Source and destination
    RLOCs have differnet afi
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Reply message
DEBUG-2:   Activating map cache entry 10.1.3.165/32
DEBUG-2: add_locator_to_mapping: The locator 84.88.81.47 has been
    added to the EID 10.1.3.165/32.
DEBUG: Balancing locator vector for 10.1.3.165/32:
DEBUG:   IPv4 locators vector (1 locators):   84.88.81.47
DEBUG:   IPv6 locators vector (0 locators):
DEBUG:   IPv4 & IPv6 locators vector (0 locators):
```

Handover: User moves to LISP-A. Solicit-Map-Requests are sent (correspondent node is notified).

```
DEBUG-2: Completed processing of LISP control message
INFO:
        LISProam: !!! User 'alice@domainb.com' moved to a new locator:
             84.88.81.44 !!!

DEBUG-2: EID prefix 10.1.2.121/32 inserted in the database
DEBUG-2: The EID 10.1.2.121/32 has been assigned to the RLOCs of the
    interface br-wan
DEBUG-2: add_locator_to_mapping: The locator 84.88.81.43 has been
    added to the EID 10.1.2.121/32.
DEBUG-2: The EID 10.1.2.121/32 has already been assigned to the RLOCs
    of the interface br-wan
DEBUG: Balancing locator vector for 10.1.2.121/32:
DEBUG:   IPv4 locators vector (1 locators):   84.88.81.43
DEBUG:   IPv6 locators vector (0 locators):
DEBUG:   IPv4 & IPv6 locators vector (0 locators):
DEBUG-2: *** Init SMR notification ***
DEBUG: Start SMR for local EID 10.1.2.0/24
DEBUG: Sent Map-Request packet for 10.1.3.165/32: Encap: N, Probe: N,
    SMR: Y, SMR-inv: N
```

```
DEBUG:    SMR'ing RLOC 84.88.81.47 from EID 10.1.3.165/32
DEBUG: Start SMR for local EID 10.1.2.121/32
DEBUG: Sent Map-Request packet for 10.1.3.165/32: Encap: N, Probe: N,
    SMR: Y, SMR-inv: N
DEBUG:    SMR'ing RLOC 84.88.81.47 from EID 10.1.3.165/32
DEBUG-2: *** Finish SMR notification ***
DEBUG-2: Deleting EID entry 10.1.2.121/32
```

Handover: After sending SMRs, the xTR sends a Map-Request to learn the new user's binding.

```
DEBUG: Sent Map-Request packet for 10.1.2.121/32: Encap: Y, Probe: N,
    SMR: N, SMR-inv: N
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Notify message
DEBUG: Map-Notify message confirms correct registration
DEBUG-2: Completed processing of LISP control message
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Reply message
DEBUG-2: process_map_reply_record:  No map cache entry found for
    10.1.2.121/32
DEBUG-2: Completed processing of LISP control message
```

xTR A (LISP-A)

Here are shown only the difference with the logs of xTR B.

After user authentication, a new virtual WLAN interface must be created, and then new static DHCP entry.

```
INFO:    LISProam: !! Authentication completed for user 'alice@domainb.
    com' !!

INFO:    LISProam: Adding wlan configuration for user 'alice@domainb.
    com' (**:**:**:**:**:**)

INFO:    LISProam: Added interface wlan0:6

INFO:    LISProam: Adding DHCP entry for foreign user 'alice@domainb.
    com'

INFO:    LISProam: Reloading DHCP Server
```

Since the user is foreign, the Map-Server of this user needs to be found.

```
INFO:    LISProam: Map-Server unknown for user 'alice@domainb.com'.
    Retrieving Map-Server.

INFO:    LISProam: Requesting Map-Server for user 'alice@domainb.com'
    (10.1.2.121)
```

```
DEBUG: Sent Map-Request packet for 10.1.2.121/32: Encap: Y, Probe: N,
    SMR: N, SMR-inv: N
INFO:   LISProam: Map-Server address received for user 'alice@domainb.
    com'

        --- User info ---
       > username: alice@domainb.com
       > eid: 10.1.2.121
       > mac: 00:0D:88:65:5A:5D
       > FOREIGN user
              > wlan id: 6
              > MS address: 84.88.81.2
              > MS key: (assigned)
        --- end ---
```

The EID is added to the local DB, before sending the Map-Register to update user's location.

```
INFO:   LISProam: Adding new user's EID 10.1.2.121/32 to Local DB

DEBUG-2: EID prefix 10.1.2.121/32 inserted in the database
DEBUG-2: The EID 10.1.2.121/32 has been assigned to the RLOCs of the
    interface eth1
DEBUG-2: add_locator_to_mapping: The locator 84.88.81.44 has been
    added to the EID 10.1.2.121/32.
DEBUG-2: The EID 10.1.2.121/32 has already been assigned to the RLOCs
    of the interface eth1
DEBUG: Balancing locator vector for 10.1.2.121/32:
DEBUG:   IPv4 locators vector (1 locators):   84.88.81.44
DEBUG:   IPv6 locators vector (0 locators):
```

When periodic Map-Registers are sent (for every EID of the local DB), the EID are check in order to see if they correspond to a foreign user. In this case, the Map-Register is sent to the foreign user's Map-Server with the correct key (in our case the Map-Server is always the same).

```
INFO: LISProam: Map-Server found for foreign user's EID 10.1.2.121
    (84.88.81.2)
DEBUG: Sent Map-Register message for 10.1.2.121/32 to Map Server
    84.88.81.2
DEBUG-2: Completed processing of LISP control message
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Notify message
DEBUG: Map-Notify message confirms correct registration
DEBUG-2: Completed processing of LISP control message
```

After the user moved in this network, the connection with the correspondent node has to be resumed. So xTR A has to learn the correspondent node's mapping, sending a Map-Request.

```
DEBUG: No map cache retrieved for eid 10.1.3.165
DEBUG-2: Added map cache entry for EID: 10.1.3.165/32
DEBUG: Sent Map-Request packet for 10.1.3.165/32: Encap: Y, Probe: N,
    SMR: N, SMR-inv: N
DEBUG-2: get_rloc_from_balancing_locator_vec: Source and destination
    RLOCs have differnet afi
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Reply message
DEBUG-2:   Activating map cache entry 10.1.3.165/32
DEBUG-2: add_locator_to_mapping: The locator 84.88.81.47 has been
    added to the EID 10.1.3.165/32.
DEBUG: Balancing locator vector for 10.1.3.165/32:
DEBUG:   IPv4 locators vector (1 locators):   84.88.81.47
DEBUG:   IPv6 locators vector (0 locators):
DEBUG:   IPv4 & IPv6 locators vector (0 locators):
```

When LISProam is terminated, the virtual WLAN created are removed.

```
DEBUG: Terminal interrupt. Cleaning up...
INFO:  LISProam: Deleted interface wlan0:6
```

xTR C

xTR C looks for mapping of EID 10.1.2.121.

```
DEBUG: No map cache retrieved for eid 10.1.2.121
DEBUG-2: Added map cache entry for EID: 10.1.2.121/32
DEBUG: Sent Map-Request packet for 10.1.2.121/32 to 84.88.81.2: Encap:
     Y, Probe: N, SMR: N, SMR-inv: N . Nonce: 0x3c5ea9a3-0xafbff35b
DEBUG-2: get_rloc_from_balancing_locator_vec: Source and destination
    RLOCs have differnet afi
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Reply message
DEBUG-2: process_map_reply: Nonce of the Map Reply is: 0x3c5ea9a3-0
    xafbff35b
DEBUG-2:   Activating map cache entry 10.1.2.121/32
DEBUG-2: add_locator_to_mapping: The locator 84.88.81.43 has been
    added to the EID 10.1.2.121/32.
DEBUG: Balancing locator vector for 10.1.2.121/32:
DEBUG:   IPv4 locators vector (1 locators):   84.88.81.43
DEBUG:   IPv6 locators vector (0 locators):
DEBUG:   IPv4 & IPv6 locators vector (0 locators):
DEBUG: The map cache entry 10.1.2.121/32 will expire in 10 minutes.
```

xTR C is notified of the mobile host's move (through SMR) and updates its location.

```
DEBUG-2: Completed processing of LISP control message
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Request message
```

```
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Request message
DEBUG: Sent Map-Request packet for 10.1.2.121/32 to 84.88.81.2: Encap:
     Y, Probe: N, SMR: N, SMR-inv: Y . Nonce: 0x7363b874-0x74bfdf77
DEBUG-2: Completed processing of LISP control message
DEBUG-2: Received a LISP control message
DEBUG: Received a LISP Map-Reply message
DEBUG-2: process_map_reply: Nonce of the Map Reply is: 0x7363b874-0
    x74bfdf77
DEBUG-2:   A map cache entry already exists for 10.1.2.121/32,
    replacing locators list of this entry
DEBUG-2: add_locator_to_mapping: The locator 84.88.81.44 has been
    added to the EID 10.1.2.121/32.
DEBUG: Balancing locator vector for 10.1.2.121/32:
DEBUG:   IPv4 locators vector (1 locators):   84.88.81.44
DEBUG:   IPv6 locators vector (0 locators):
DEBUG:   IPv4 & IPv6 locators vector (0 locators):
DEBUG: The map cache entry 10.1.2.121/32 will expire in 10 minutes.
```

8.2 Wireshark captures

Aside from the behaviour and output of LISProam it is in the interest of everyone to see the actual packets exchanged in the case depicted above, captured with Wireshark.

In this case, the user (`alice@domainb.com`) connects to LISP-A, then moves to LISP-B, then again to LISP-A. The first handover is to user's home network. and the second one is the return to the foreign network where the user is now known.

Booting up in foreign network

The user connects to LISP-A, then he gets authenticated. The RADIUS dialogue is skipped, since it has been clarified in previous chapters. Since the user is foreign, an Encapsulated Map-Request is sent to retrieve user's Map-Server. Upon receiving Map-Reply, the Map-Register for the user is sent to the learned Map-Server (figure 8.2).

Communication with correspondent node

Correspondent node (10.1.3.165) in xTR C network starts pinging 10.1.2.121 (figure 8.3). As a result, xTR C automatically queries the Mapping System for user's EID. It can be seen that some Ping requests don't get replied: this is because of the time needed for retrieving user's EID, and also for retrieving correspondent node's EID on xTR A. These requests get lost. It has been experimented that the amount of lost packets in this initial phase of the communication is around 2.

5 63.958198000	10.1.1.0	10.1.2.121	LISP	134 Encapsulated Map-Request for 10.1.2.121/32
6 63.959829000	84.88.81.2	84.88.81.44	LISP	82 Map-Reply for 10.1.2.0/24
7 63.961537000	84.88.81.44	84.88.81.2	LISP	106 Map-Register for 10.1.2.121/32
8 63.965259000	84.88.81.2	84.88.81.44	LISP	106 Map-Notify for 10.1.2.121/32
9 78.392492000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=2/512, ttl=63
10 78.404326000	10.1.2.121	10.1.3.165	ICMP	98 Echo (ping) reply id=0x5f56, seq=2/512, ttl=63
11 78.405236000	10.1.2.121	10.1.3.165	LISP	134 Encapsulated Map-Request for 10.1.3.165/32
12 78.406931000	84.88.81.2	84.88.81.44	LISP	82 Map-Reply for 10.1.3.165/32
13 79.392498000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=3/768, ttl=63
14 79.403698000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=3/768, ttl=63
15 80.394996000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=4/1024, ttl=63
16 80.400503000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=4/1024, ttl=63

Figure 8.2. Capture - User connects to home network and starts pinging

7 73.448678000	10.1.3.165	10.1.2.121	LISP	134 Encapsulated Map-Request for 10.1.2.121/32
8 73.449326000	10.1.3.165	10.1.2.121	ICMP	98 Echo (ping) request id=0x5f56, seq=1/256, ttl=63
9 73.450515000	84.88.81.2	84.88.81.47	LISP	82 Map-Reply for 10.1.2.121/32
10 74.446959000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=2/512, ttl=63
11 75.446603000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=3/768, ttl=63
12 75.458597000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=3/768, ttl=63
13 76.449108000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=4/1024, ttl=63
14 76.458166000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=4/1024, ttl=63

Figure 8.3. Capture - xTR queries Mapping System to start ping

In previous figure 8.2 we can see that the xTR receives a ping request from 10.1.3.165. After this, the xTR automatically looks for that EID in the Mapping System (Map-Request, Map-Reply). When the mapping is saved in the Map-Cache the ping starts working correctly between 10.1.2.121 and 10.1.3.165.

Handover to home network

The user, previously connected to LISP-A, connects to his home network (LISP-B). As seen in figure 8.4, there are no Map-Requests sent in order to learn the Map-Server, since it's the home one. Hence, the first thing that happens is that user's mapping gets updated with a Map-Register for his /32 EID. As shown, xTR B receives few Ping request, to which can answer only after having queried the Mapping System, looking for the correspondent node's mapping.

6 86.223394000	84.88.81.43	84.88.81.2	LISP	106 Map-Register for 10.1.2.121/32
7 86.226860000	84.88.81.2	84.88.81.43	LISP	106 Map-Notify for 10.1.2.121/32
8 86.307915000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=15/3840, ttl=63
9 87.307513000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=16/4096, ttl=63
10 88.307534000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=17/4352, ttl=63
11 88.320131000	10.1.2.0	10.1.3.165	LISP	134 Encapsulated Map-Request for 10.1.3.165/32
12 88.321953000	84.88.81.2	84.88.81.43	LISP	82 Map-Reply for 10.1.3.165/32
13 89.315831000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=18/4608, ttl=63
14 90.309381000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=19/4864, ttl=63
15 90.313256000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=19/4864, ttl=63
16 91.310594000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=20/5120, ttl=63

Figure 8.4. Capture - User connects to foreign network and ping is resumed

At this point, xTR A receives a Map-Notify for 10.1.2.121/32, which contains different RLOCs (84.88.81.43). As explained, an SMR is sent to the correspondent node saved in the Map-Cache (Map-Request for 10.1.3.165, with S bit set), and the binding in the Map-Cache is updated (querying the Mapping System) (figure 8.5).

33	91.320103000	84.88.81.2	84.88.81.44	LISP	106 Map-Notify for 10.1.2.121/32
34	91.322661000	84.88.81.44	84.88.81.47	LISP	102 Map-Request for 10.1.3.165/32
35	91.324500000	84.88.81.44	84.88.81.47	LISP	102 Map-Request for 10.1.3.165/32
36	91.957501000	10.1.1.0	10.1.2.121	LISP	134 Encapsulated Map-Request for 10.1.2.121/32
37	91.959119000	84.88.81.2	84.88.81.44	LISP	82 Map-Reply for 10.1.2.121/32

Figure 8.5. Capture - User moves away from foreign network

xTR C receives a Solicit-Map-Request, that is a Map-Request with the S bit set, containing the EID of the moved user in the "Map-Reply record" part of the message. As a consequence, xTR C updates the mapping with an Encapsulated Map-Request for 10.1.2.121/32. We can already see that some packets are lost, since the xTR is sending Ping requests that are not actually answered.

28	84.458997000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=12/3072, ttl=63
29	85.460795000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=13/3328, ttl=63
30	86.459813000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=14/3584, ttl=63
31	87.381609000	84.88.81.44	84.88.81.47	LISP	102 Map-Request for 10.1.3.165/32
32	87.383727000	84.88.81.44	84.88.81.47	LISP	102 Map-Request for 10.1.3.165/32
33	87.384806000	10.1.3.165	10.1.2.121	LISP	102 Encapsulated Map-Request for 10.1.2.121/32
34	87.386648000	84.88.81.2	84.88.81.47	LISP	82 Map-Reply for 10.1.2.121/32
35	87.459261000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=15/3840, ttl=63
36	88.458621000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=16/4096, ttl=63
37	89.458423000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=17/4352, ttl=63
38	90.458689000	10.1.3.165	10.1.2.121	ICMP	134 Echo (ping) request id=0x5f56, seq=18/4608, ttl=63
39	90.467583000	10.1.2.121	10.1.3.165	ICMP	134 Echo (ping) reply id=0x5f56, seq=18/4608, ttl=63

Figure 8.6. Capture - xTR queries Mapping System upon receiving SMR

Handover to (same) foreign network

After a while, we reconnect the user to LISP-A. Now the user is recognized by xTR A, so it does not have to go through the steps depicted in figure 8.2. This time, the only thing to do is to update the location of the user, with a Map-Register (figure 8.7). Further more, the mapping of the correspondent node in the Map-Cache has not expired yet, so the ping is resumed instantly without having to query the Mapping System.

The xTR B is notified of the move, therefore sends a SMR to xTR C, and updates the Map-Cache. The interesting fact shown in figure 8.8 is that few Ping requests are still received, even if the user moved. This is because the correspondent node's xTR (xTR C) didn't refresh the binding yet. Packets received during the gap between user's move and correspondent node's update get irreparably lost.

38 116.957618000	84.88.81.44	84.88.81.2	LISP	106	Map-Register for 10.1.2.121/32
39 116.961003000	84.88.81.2	84.88.81.44	LISP	106	Map-Notify for 10.1.2.121/32
40 117.434501000	10.1.3.165	10.1.2.121	ICMP	134	Echo (ping) request id=0x5f56, seq=41/10496, ttl=63
41 117.438829000	10.1.2.121	10.1.3.165	ICMP	134	Echo (ping) reply id=0x5f56, seq=41/10496, ttl=63
42 118.436981000	10.1.3.165	10.1.2.121	ICMP	134	Echo (ping) request id=0x5f56, seq=42/10752, ttl=63
43 118.438706000	10.1.2.121	10.1.3.165	ICMP	134	Echo (ping) reply id=0x5f56, seq=42/10752, ttl=63

Figure 8.7. Capture - User reconnects to foreign network

46 109.329691000	10.1.3.165	10.1.2.121	ICMP	134	Echo (ping) request id=0x5f56, seq=38/9728, ttl=63
47 110.329683000	10.1.3.165	10.1.2.121	ICMP	134	Echo (ping) request id=0x5f56, seq=39/9984, ttl=63
48 111.334639000	10.1.3.165	10.1.2.121	ICMP	134	Echo (ping) request id=0x5f56, seq=40/10240, ttl=63
49 111.867991000	84.88.81.2	84.88.81.43	LISP	106	Map-Notify for 10.1.2.121/32
50 111.876275000	84.88.81.43	84.88.81.47	LISP	102	Map-Request for 10.1.3.165/32
51 111.877203000	84.88.81.43	84.88.81.47	LISP	102	Map-Request for 10.1.3.165/32
52 111.878518000	10.1.2.0	10.1.2.121	LISP	134	Encapsulated Map-Request for 10.1.2.121/32
53 111.880893000	84.88.81.2	84.88.81.43	LISP	82	Map-Reply for 10.1.2.121/32
54 118.313230000	84.88.81.2	84.88.81.43	LISP	106	Map-Notify for 10.1.2.0/24
55 121.355206000	10.1.3.165	10.1.2.121	ICMP	90	Echo (ping) request id=0x5f56, seq=35/8960, ttl=63

Figure 8.8. Capture - User moves away from foreign network

User's point of view

The packet flows on the xTRs are useful to understand what happens on the network side, and for verifying what is the actual cause of the packet loss, but it is even more important to report how is the whole test case seen from the point of view of the mobile host itself.

1. Figure 8.9 depicts the first connection to LISP-A

70 15.043974000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xf9e0fd74
71 15.055589000	10.1.2.122	10.1.2.121	DHCP	353	DHCP Offer - Transaction ID 0xf9e0fd74
72 15.055865000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0xf9e0fd74
73 15.068951000	10.1.2.122	10.1.2.121	DHCP	365	DHCP ACK - Transaction ID 0xf9e0fd74
82 19.059435000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=2/512, ttl=62
83 19.059475000	10.1.2.121	10.1.3.165	ICMP	98	Echo (ping) reply id=0x5f56, seq=2/512, ttl=64
84 20.058795000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=3/768, ttl=62
85 20.058832000	10.1.2.121	10.1.3.165	ICMP	98	Echo (ping) reply id=0x5f56, seq=3/768, ttl=64

Figure 8.9. Capture - User connects to home network

2. Figure 8.10 depicts the connection to LISP-B
3. Figure 8.11 depicts the second connection to LISP-A.

It is clear that the packet flow is quite usual, and there are no trace of uncommon behaviour from this point of view. It is now really evident that the solution deployed is really network-based. The DHCP dialogues are standard. What changes between the two is just the server's IP address: which is xTR B's EID (10.1.2.254) when connecting to the home network (LISP-B) and it is the virtual

140 33.020797000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x1bdf4779
141 33.033736000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=15/3840, ttl=62
142 33.034350000	10.1.2.254	10.1.2.121	DHCP	343	DHCP Offer - Transaction ID 0x1bdf4779
143 33.034525000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x1bdf4779
144 33.051185000	10.1.2.254	10.1.2.121	DHCP	355	DHCP ACK - Transaction ID 0x1bdf4779
145 33.071109000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=16/4096, ttl=62
156 34.078709000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=17/4352, ttl=62
159 34.079970000	10.1.2.121	10.1.3.165	ICMP	98	Echo (ping) reply id=0x5f56, seq=17/4352, ttl=64

Figure 8.10. Capture - User connects to foreign network

252 57.360793000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x98df9773
253 57.389608000	10.1.2.122	10.1.2.121	DHCP	365	DHCP ACK - Transaction ID 0x98df9773
260 58.103628000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=41/10496, ttl=62
263 58.104362000	10.1.2.121	10.1.3.165	ICMP	98	Echo (ping) reply id=0x5f56, seq=41/10496, ttl=64
270 59.105993000	10.1.3.165	10.1.2.121	ICMP	98	Echo (ping) request id=0x5f56, seq=42/10752, ttl=62
271 59.106034000	10.1.2.121	10.1.3.165	ICMP	98	Echo (ping) reply id=0x5f56, seq=42/10752, ttl=64

Figure 8.11. Capture - User reconnects to home network

one (10.1.2.122) added to xTR A in the case of connection to the foreign network (LISP-A). It must be noticed that when reconnecting to LISP-A the DHCP Discover and Offer messages are skipped (figure 8.11), and this has to be taken in account as another factor that speeds up connection between known network and known users.

8.3 Measuring packet loss

We verified the data loss during the handover. What differentiates each handover is how the user is treated in the network he attaches to. The time needed for having the user correctly connected and able to communicate depends on the origin of the user (home or foreign) and on the knowledge the xTR has (known or unknown). Experimental tests brought us to the expected conclusion that the worst case scenario is a foreign unknown user, which introduces a packet loss around 5 packets. If the foreign user is known this loss is already reduced to a mean value of 2 (figure 8.12).

In the case of home users, there is just a slight difference if the user is known or unknown, as seen in figure 8.13.

Further more, it is interesting to verify the amount of time actually needed for the user to perform an handover. We compared the worst case (foreign unknown user), and the best case (home known user) in figure 8.14. We can see that even in the best case the latency introduced by the handover is up to more than 3 seconds. Unfortunately it is difficult to lower this threshold, since it is average the time needed for RADIUS authentication.

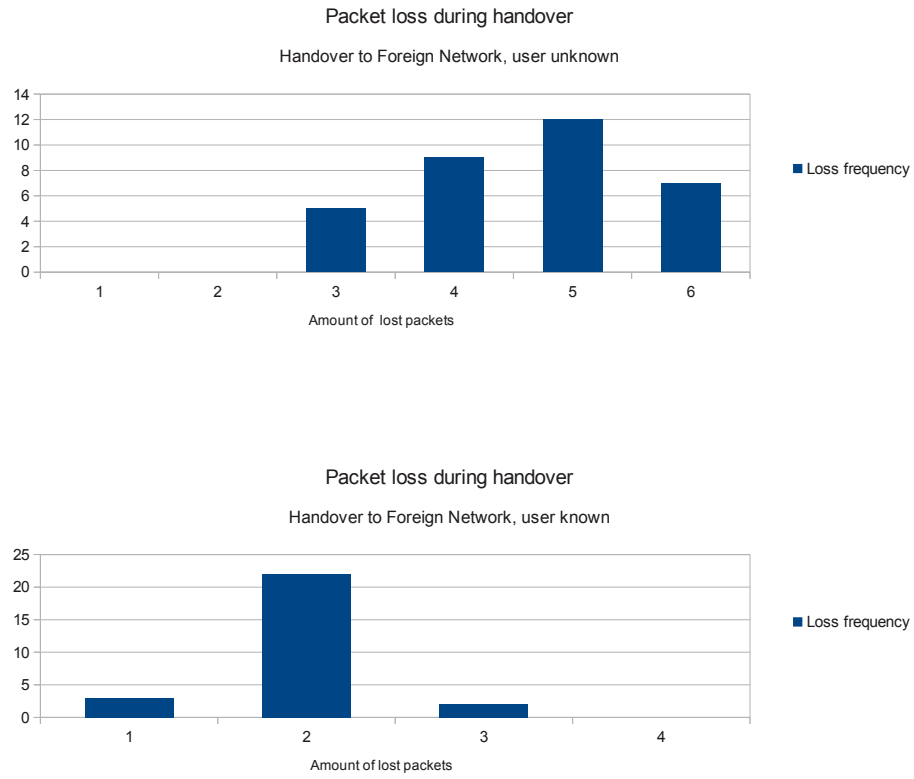


Figure 8.12. Packet loss - Foreign user sub-case

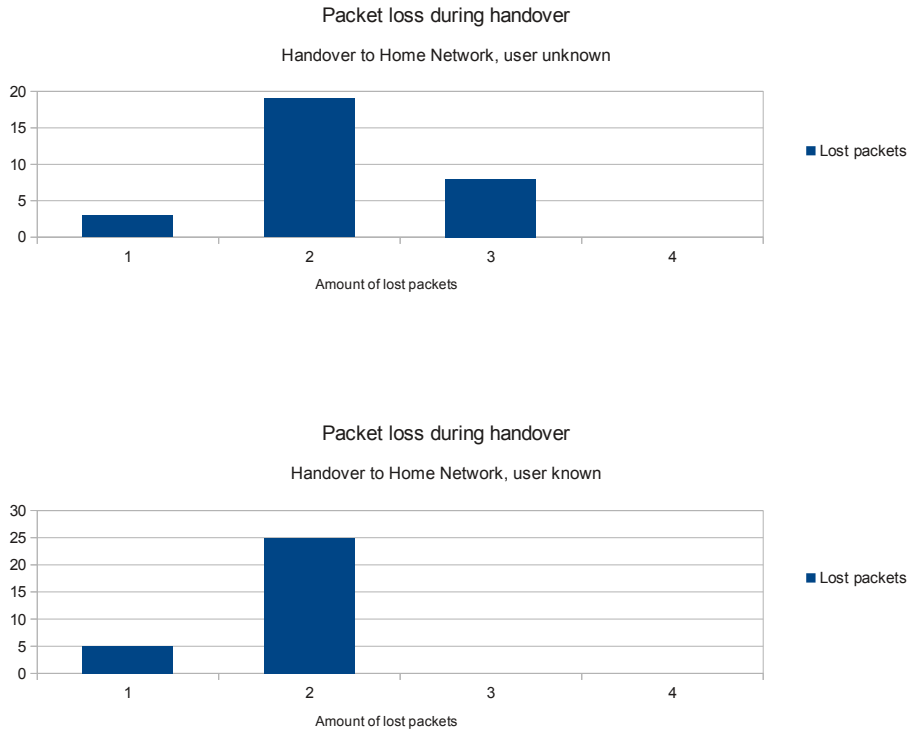


Figure 8.13. Packet loss - Home user sub-case

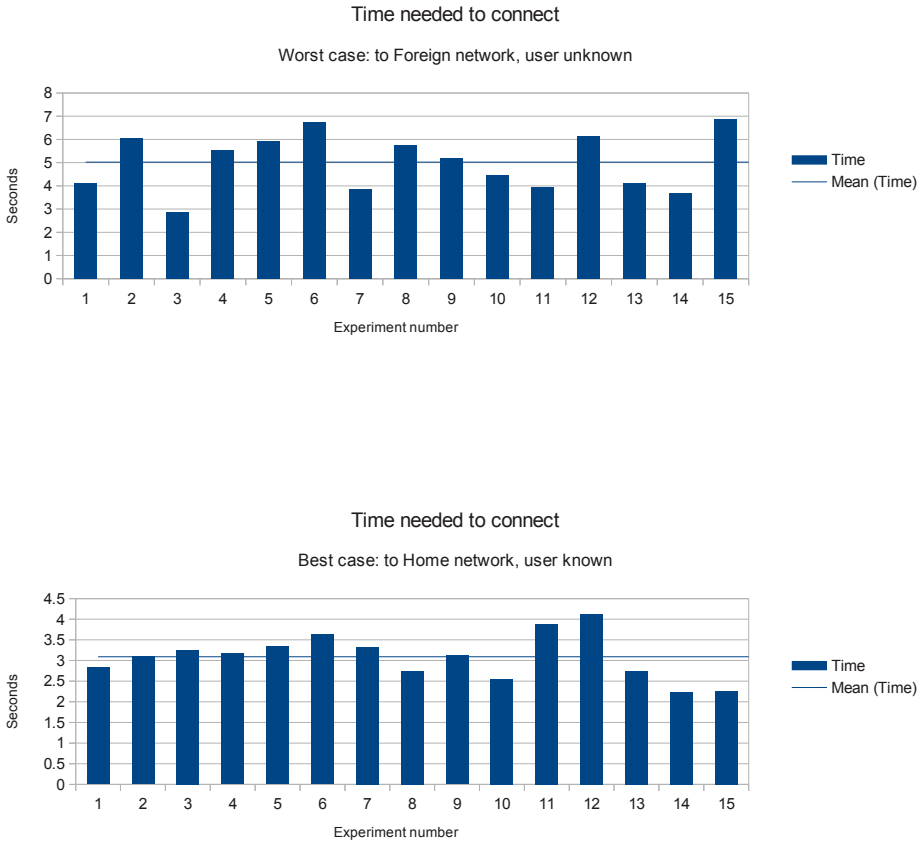


Figure 8.14. Latency - Worst and best case comparison

Chapter 9

Future developments

The results shown in this work can definitely lead to further discussions and ideas on how and where to deploy the solution in a real world scenario, aside from how it can be further extended.

It has to be reminded that since the beginning of the research done in this work the scenario considered was the most realistic as possible. Indeed, the solution has been thought to be implemented by Internet Service Providers and the goal imagined was to build a new mobility service. In this way we first assumed - then proved - that an inter-domain mobility solution is feasible. We did our best to keep the assumptions as generic as possible, never going too much in the details. This has been chosen to not create any kind of constraints that could make the solution possible to work only in a specific scenario. We did not add hazardous assumptions to the scenario not even while implementing the prototype. This let us consider a possible future extensions, including mobile operators.

3G/4G scenario is clearly the most interesting one, since today the Internet is mostly populated by smart phones which continuously switch between Wi-Fi networks (home or office) and mobile networks (when moving from a place to another). In recent times a lot of effort has been put in researching interoperability solutions between the two parties. Without going into the details of how a mobile network is deployed, it has to be reminded that its nature is purely IP, so it is fair to think about creating a way to let the two types of networks cooperate.

Further more, the increasingly need for the user to manage both Internet and mobile connections led to the creation of solutions to support multiple IP flows on a single device. Mobile networks support Mobile IP and Proxy Mobile IP, so a lot of additional specifications to these two protocols has been added in recent times (for example [4]). Other solutions that are gaining ground recently do not work at IP level but for example at Level 4, like Multipath TCP [2], which has been lately implemented in iOS 7.

LISP has given a contribute in this scenario, mostly with LISP-MN. Indeed, installing LISP-MN on a mobile device lets it switch between Wi-Fi and 3G seamlessly, without dropping connections. Our solution achieved the same goal following a network-based paradigm, considering "clean" hosts roaming into networks.

The point of view is completely different. Deploying a host-based solution allows to directly manipulate the traffic in the device and, in the case of mobile devices, the multiple flows that are generated (Wi-Fi and 3G). A network-based solution implies that the other components (routers) have to manage these connections for the client, and in the case of multiple flows is far more difficult.

Considering the architecture we deployed for our work, we can imagine how it can be embedded in a mobile network. The idea is first to make the part of the LISP infrastructure needed for our solution be accessible to mobile networks, e.g. adding Map-Servers/Map-Resolvers to every mobile network provider. Apart from LISP, we use few hardware and software components. Basically they can be summed saying that we need a RADIUS Server and a particular DHCP Server (which has to be embedded in the Edge Router), properly configured in every network.

Mobile networks already use the RADIUS infrastructure. The mobile device authenticates to the cell it is currently attached to using its SIM card. The mechanism simply uses EAP, in particular EAP-SIM ([15]) and EAP-AKA ([14]). These two EAP types use data extracted from the SIM card as credentials, differently from the other types of EAP seen that use user credentials of certificates. In mobile networks a Home Location Register (HLR) is used for gathering data of users belonging to that domain, and it is queried by the RADIUS Server to obtain authentication for users. Therefore, what has to be changed of the authentication part is that we must synchronize user's data (the one relative to our mobility solution) with the data of his SIM card (contained in the HLR). This means that the user has to be recognized both through his credentials and his SIM card, and has to be given the related data (EID, Map-Server key) properly.

It is not clear, at the time of writing, how an IP address is allocated for the user in mobile networks, neither is clear if there is a standard procedure adopted by providers. Still, we can assume that a mechanism DHCP-like is used in mobile networks, mostly considering that solutions like MIP and PMIP are effectively used. Therefore it can be fairly assumed that the DHCP Server (or whichever similar component is used) needs to be configured like we explained in our solution.

Putting this parts together we should have an idea on how to obtain interoperability between Wi-Fi and 3G/4G networks, and also user mobility with session continuity.

Another interesting development for this work (and the most natural one) would be making the solution designed be implemented by Internet Service Providers. Some considerations has to be made, considering this scenario. The solution designed and implemented in this thesis operates in between the Edge layer and the

Access layer of the network. Hence, some modifications are required in these two parts to make the solution adoptable by ISPs. First of all, it has to be clarified how an ISP can actually implement the LISP infrastructure. One of the main advantages of LISP is that is deployable "day-one", which means it does not require any modification to the Core layer of the network. All the effort has to be put in modifying the behaviour of the Edge Routers.

What should happen is that ISPs adopt RLOCs for the Edge Router and use EIDs for the users, which follows the idea of this work. This means that an ISP has to reserve EID-prefixes for its users, just like it reserves public IP addresses (RLOCs) for the routers. It is hazardous to think about switching to this paradigm from a day to another, mostly because it would need address renumbering and part of the functionality available today (NAT, most of all) will be lost. That's why in this thesis we always referred to the solution deployed as a new mobile service, meaning a new infrastructure to be built beside the current one.

It is possible to go briefly in the details of how this could actually work. Today, Internet connection is made accessible to users through Access Points (at home, workplace, etc.) which use NAT most of time to mask the local network to the outside.

NAT can not be considered for our solution, mainly because users' EIDs are fixed, unique and have a global value while an IP under NAT is just used for the LAN and has a local scope. So what should be done is making the Edge Router an actual xTR, with an embedded DHCP Server which is used to give reserved EIDs to users.

This modifications to an ISP can be done for creating a global mobility service. Also establishing agreements with other ISPs, a user could effectively be able to do Wi-Fi roaming without dropping connections. It has to be said that this scenario can be considered realistic when thinking about deploying very wide networks, like WiMAX, letting a user be able to literally move wherever remaining connected, which can be considered possible in a near future.

Another idea is to decrease the scope considered for an actual scenario, and imagine the solution implemented only in Access Routers instead of Edge Routers. That is, deploying LISProam only in the boxes (Access Routers) while ignoring what is behind, that is how we reach the Internet. This is actually possible, and is indeed the case depicted for the prototype and the tests. Of course this will not lead to a wide spread mobility service, but the range of the service will actually be the area covered by the Access Routers' Wi-Fi ranges. We can take eduroam as an example of a network service distributed in quite considerably big area, e.g. university campuses, and think about doing something similar with LISProam. This idea would allow users moving into a certain space (campuses, buildings, different offices) to maintain connections up and running, which is not what happens today.

The latter idea seems more feasible that the other ones, since it does not require modifications to ISPs and represents a more practical use case.

All of the ideas exposed in this chapter seem valid for some scenarios and should be taken in account for further studies and research. It has to be reminded that other extensions regarding design or implementation have already been discussed in the related parts of this thesis and are not reported in this chapter.

Chapter 10

Conclusions

The big constraint of the Internet scenario faced in this work led to a great challenge in finding possible ways to overcome this limit. The problems studied gave the opportunity to deepen the research about which are the main concerns about user mobility in the networks, and a considerable effort has been put in showing the nature of these limits in the detail. A big part of the work focused on proposing theoretical solutions to satisfy the goals proposed, keeping a high level of abstraction in order to allow future developments. In the end, a prototype has been deployed for showing that the solution designed is actually feasible even without making hazardous assumptions. Although the case considered for testing has quite a small scope (for practical reasons) it is fair to assume that the solution designed can really work in a wide spread scenario, even at ISP level. The core of the work, indeed, has been kept as simple as possible, and all the network components used were never heavily modified - at most just configured - making the solution easy to implement and scale in every environment.

LISP can be considered a very powerful network protocol, which will lead to further interesting developments in network research, and its application in the mobility field should be taken in account also from a commercial point of view. The case studied here is just a small use case, but it is enough to prove that user mobility is indeed possible at IP level. We can imagine potentially using the ideas shown in this work in different future scenarios. Having interoperability between ISP networks and connections continuity we can imagine deploying wide spread networks (e.g. city-wide). Or, on the other side, we can imagine using the solution implemented for this work in smaller places (e.g. campuses, buildings), guaranteeing a continuous connectivity taking advantage of the network attaching points that are already present nowadays.

We can fairly say that the goal reached in this work can be considered an important starting point for working on network-based user mobility with LISP.

Bibliography

[1] (online) LISP Beta Network: lisp4.net.

[2] M. Handley O. Bonaventure A. Ford, C. Raiciu. *RFC 6824: TCP Extensions for Multipath Operation with Multiple Addresses*, 01 2013.

[3] Albert Lopez Alberto Rodriguez Natal. (online) LISPmob Documentation: lispmob.org.

[4] CJ. Bernardos. *draft-ietf-netext-pmipv6-flowmob-08: Proxy Mobile IPv6 Extensions to Support Flow Mobility.* UC3M, 10 2013.

[5] Cisco Systems. *Locator ID Separation Protocol (LISP) VM Mobility Solution - White Paper*, 2011.

[6] C. White D. Farinacci, D. Meyer. *draft-meyer-lisp-mn-09: LISP Mobile Node.* Cisco Systems, 7 2013.

[7] D. Meyer D Lewis D. Farinacci, V. Fuller. *RFC 6836: Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT).* Cisco Systems, 01 2013.

[8] J. Snijders D. Farinacci, D. Meyer. *draft-farinacci-lisp-lcaf-10: LISP Canonical Address Format (LCAF).* Cisco Systems, InTouch N.V., 7 2012.

[9] D Lewis D. Farinacci V. Fuller, D. Meyer. *RFC 6830: The Locator/ID Separation Protocol (LISP).* Cisco Systems, 01 2013.

[10] J. Arkko D. Johnson, C. Perkins. *RFC 3775: Mobility Support in IPv6.* Rice University, Nokia Research Center, Ericsson, 6 2004.

[11] L. Zhang D. Meyer, L. Zhang. *RFC 4984: Report from the IAB Workshop on Routing and Addressing.* Internet Architecture Board, 09 2007.

[12] D. Farinacci V. Fuller. *RFC 6833: Locator/ID Separation Protocol (LISP) Map-Server Interface.* Cisco Systems, 01 2013.

[13] RADIUS Extensions Working Group. *draft-ietf-radext-radsec-12: Transport Layer Security (TLS) encryption for RADIUS.* RESTENA, Cisco Systems, 08 2012.

[14] J. Arkko H. Haverinen. *RFC 4187:Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA).* Nokia, Ericsson, 01 2006.

[15] J. Salowey H. Haverinen. *RFC 4186: Extensible Authentication Protocol*

Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). Cisco Systems, Nokia, 01 2006.

[16] Juergen Haas. (online) Linux / Unix Command: netlink.

[17] Huachun Zhou Hongke Zhang, Feng Qiu. *draft-zhang-lisp-hmm-01: A Hierarchical Mobility Management in LISP network*, 12 2012.

[18] IEEE. *802.11F, IEEE Trial-Use Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-AccessPoint Protocol Across Distribution Systems Supporting IEEE 802.11TM Operation*, 2006.

[19] Zhenghu Gong Jie Hou, Yaping Liu. Support Mobility for Future Internet. *Conference of International Telecommunications Network Strategy and Planning Symposium*, pages 1 – 6, 2010.

[20] Simon Kelley. (online) DNSmasq documentation: thekelleys.org.uk.

[21] Livingston Enterprises, Inc. *RFC 2138: Remote Authentication Dial In User Service (RADIUS)*, 04 1997.

[22] Seok-Joo Koh Moneeb Gohar. A distributed mobility control scheme in LISP networks. *Wireless Networks.*

[23] OpenWRT. (online) OpenWRT Documentation: wiki.openwrt.org,.

[24] Jonathan Leary Pejman Roshan. *802.11 Wireless LAN Fundamentals.* Cisco Press, December 2003.

[25] C. Perkins. *RFC 2002: IP Mobility Support.* IBM, 10 1996.

[26] Chi Secci Cianfrani Gallard Pujolle Raad, Colombo. Achieving sub-second downtimes in internet-wide virtual machine live migrations in LISP networks. *Integrated Network Management (IM 2013), 2013 IFIP/IEEE*, pages 286 – 293, 2013.

[27] Ignacio Soto, Carlos J. Bernardos, and MarÃa Calderon. PMIPv6: A Network-Based Localized Mobility Management Solution. *The Internet Protocol Journal*, 13(3):1–32, 2010.

[28] William Stallings. Mobile IP. *The Internet Protocol Journal*, 4(2):2–14, 2001.

[29] FreeRADIUS Development Team. FreeRADIUS website: freeradius.org.

[30] TERENA. (online) eduroam Documentation,.

[31] D. Meyer D Lewis A. Jain V. Ermagan, V. Fuller. *draft-ietf-lisp-ddt-01: LISP Delegated Database Tree.* Cisco Systems, Juniper Networks, 04 2013.