AN IMPROVED LOCATOR IDENTIFIER SPLIT ARCHITECTURE (ILISA) TO ENHANCE MOBILITY

Musab Muhammad Isah

School of Computing and Communications

Thesis submitted for the degree of

Doctor of Philosophy

October 2017.

DEDICATED TO

my parents, my wife, and my children.

ABSTRACT

The increased use of mobile devices has prompted the need for efficient mobility management protocols to ensure continuity of communication sessions as users switch connection between available wireless access networks in an area. Locator/Identifier (LOC/ID) split architectures are designed to, among other functions, enable the mobility of nodes on the Internet. The protocols based on these architectures enable mobility by ensuring that the identifier (IP address) used for creating a communication session is maintained throughout the lifetime of the session and only the location of a mobile node (MN) is updated as the device moves.

While the LOC/ID protocols ensure session continuity during handover, they experience packet loss and long service disruption times as the MN moves from one access network to another. The mobility event causes degradation of throughput, poor network utilisation, and affects the stability of some applications, such as video players. This poor performance was confirmed from the experiments we conducted on a laboratory testbed running Locator Identifier Separation Protocol MN (LISP-MN) and Mobile IPv6 (MIPv6). The MIPv6, as the standardised IETF mobility protocol, was used to benchmark the performance of LISP-MN. The poor performance recorded is owed to the design of the LISP-MN's architecture, with no specific way of handling packets that arrive during handover events.

Our main aim in this thesis is to introduce an Improved Locator/Identifier Split Architecture (ILISA) designed to enhance the mobility of nodes running a LOC/ID protocol by mitigating packet loss and reducing service disruption in handovers. A new network node, Loc-server, is central to the new architecture with the task of buffering incoming packets during handover and forwarding the packets to the MN on the completion of the node’s movement process. We implemented ILISA with LISP-MN on a laboratory testbed to evaluate its performance in different mobility scenarios. Our experimental results show a significant improvement in the mobility performance of MNs as reflected by the different network parameters investigated.

DECLARATION

I declare that the work in this thesis is my own work and has not been submitted either in whole or in part for the award of a higher degree elsewhere. Any sections of the thesis, which have been published, are clearly identified.

[illegible]

…………….

Musab Muhammad Isah

October 2017

ACKNOWLEDGEMENT

All praise and thanks to Allah, the One, who by His blessing and favour good works are accomplished. The five years of PhD are without a doubt the most intense I have experienced in my life and gratitude must be expressed to the individuals that helped in making the period a less-difficult and an enjoyable one. Special thanks to my parents and siblings who encouraged me, prayed for me, and provided moral and financial support throughout my life and especially during my PhD. I am ever grateful to my supervisor, Dr. Chris Edwards, for guiding me through the years, taking me under his wings and showing no sign of tiredness up to the end.

Another pillar of support through this journey is my wife, Nadia Bawa, to whom I owe a lot of thanks and gratitude. There are no words to describe how much she contributed to this PhD by ‘holding up at the Homefront' while I kept endless hours in the lab. She took care of our beautiful children in the most loving way a good mother would. The calming effect and significance of her words of encouragement to lift up my spirit whenever I expressed dismay over my progress cannot be over-emphasised.

I also have to thank my colleague, Dr. Yusuf Sani, for helping me to get the grips of some of the tools necessary for my research, and for different collaborative works we did, as well as papers we published together. Special thanks to Dr. Steven Simpson, a senior researcher at Infolab21, for his tutorship on system programming and for taking time to review my code. Steve was always there to point me in the right direction towards solving the many complex programming problems that arose in the course of building the new LOC/ID split architecture.

Finally, thanks to all that supported me to achieve the PhD in one way or another, especially my mentor, Dr. Kabir Ahmed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PUBLICATIONS

Workshop

1. An ILNP-Based Mobility Solution for Future Heterogeneous Wireless Networks; PGNet2013, Liverpool, UK.

Conferences

2. Inter-Domain Mobility with LISP-MN – A Performance Comparison with MIPv6; IFIP WMNC 2015, Munich, Germany.
3. Towards Zero-Packet-Loss with LISP-MN, IEEE International Conference on Computing, Networking and Communications, ICNC 2017, Silicon Valley, USA.

Journals

4. Experimental Evaluation of the Impact of Mobility Management Protocols on HTTP Adaptive Streaming, IET Networks Journal, August 2017 (Published).
5. An Improved LISP Mobile Node Architecture, Elsevier Journal of Networks and Computer Applications (under review).

LIST OF ACRONYMS

3GPP	Third Generation Partnership Project
ABC	Always Best Connected
AN	Access Network
AP	Access Point
AR	Access Router
B_Ack	Binding Acknowledgement
BAD	Binding Authorisation Data
BEX	Base Exchange
BST	Base Station
BU	Binding Update
CN	Control Module
CN	Correspondent Node
CoA	Care-of Address
DAD	Duplicate Address Detection
DASH	Dynamic Adaptive Streaming over HTTP
DF	Diffie-Hellman
DFZ	Default Free Zone
DHCPv6	Dynamic Host Configuration Protocol Version Six
DM	Data Module
E1…E7	Expression 1 to 7
EID	Endpoint Identifier
EO	Evaluation Objectives
ETR	Egress Tunnel Router
FIB	Forwarding Information Base

HA	Home Agent
HAS	HTTP Adaptive Streaming
HeWE	Heterogeneous Wireless Environment
HFG	Handover Failure Guard
HI	Host Identity
HIP	Host Identity Protocol
HIT	Host Identity Tag
HITr	Handover Imminent Trigger
HoA	Home Address
HoWE	Homogeneous Wireless Environment
IHP	Initial/Handshake Packet
ILCC	Identifier-Locator Communication Cache
ILNP	Identifier-Locator Network Protocol
IRTF	Internet Research Task Force
ITR	Ingress Tunnel Router
IVIP	Internet Vastly Improved Plumbing
LISP	Locator-Identifier Split Protocol
LISP-MN	LISP Mobile Node
LMS	Local Mapping System
LOC/ID	Locator Identifier Split
Loc-Server	Location Area Server
LSI	Local Scope Identifier
MAB	Map Address Block
MD	Movement Detection
MIPv6	Mobile IPv6

MN	Mobile Node
MSS	Microsoft Smooth Streamer
PETR	Proxy ETR
PI	Provider Independent
PITR	Proxy ITR
PXTR	Proxy XTR
QoS	Quality of Service
RAdv	Router Advertisement
RRG	Routing Research Group
RLOC	Routing Locator
RO	Route Optimisation
RSlt	Router Solicitation
RTR	Re-encapsulating Tunnel Router
RVS	Rendezvous Server
SDT	Service Disruption Time
SPI	Security Parameter Index
SMR	Solicit Map Request
TTR	Translating Tunnel Router
XTR	Egress/Ingress Tunnel Router

CHAPTER ONE

INTRODUCTION

Mobile devices – especially smartphones – are increasingly becoming the primary source of access to the Internet as the number of wirelessly connected devices exceeded the world population in 2014 [1]. Due to the heterogeneity of wireless access technology, most of these devices are equipped with more than one radio interface, Cellular and Wi-Fi (Wireless-Fidelity) interfaces being the most popular. Telecom operators and standardisation bodies have come up with several specifications, architectures, and standards [2-5] to enable the operators and the service providers to cope with the burgeoning wireless traffic on the Internet as a consequence of, among other factors, an increase in smartphone usage. The other factors responsible for the global growth of data traffic from mobile nodes (MNs) include increased multimedia use, growth in average traffic per device, an increase in devices' speed, bandwidth-hungry applications, impact of 4G connections, among other things. More than half of this traffic comes from video applications, which accounted for 60% of the total mobile data traffic in the year 2016 according to Cisco Visual Networking Index (VNI) [6].

1.1 Overview

In today's networks, mobile users have several wireless networks available to which their devices connect to and disconnect from automatically depending on the devices' network needs, configuration and subscription. The most common four scenarios today include the change of point of attachment (PoA) to the Internet from one cellular base station (BST) to another as a user moves within their operator's coverage area. The second scenario involves the movement of a user's device between access points (AP) under the same administration in a Wi-Fi hotspot, such as university campuses and office environments.

The third scenario involves changing connectivity between 3/4G cellular and home/office/public Wi-Fi. The fourth also involves the movement of wireless devices between Wi-Fi APs, as the second scenario, but the networks are under a different administrative control in this case as can be seen in places such as train stations, airports, shopping malls, and in areas where many households provide free public Wi-Fi access. The third and fourth scenarios will usually require a change in the device's IP address.

Mobility events requiring MNs to change IP address during the process are quite common today. For example, Gao et al. [7] have shown that twenty percent of MNs have at least ten IP address changes per day, which suggests roaming between networks under different domain and/or administrative control. We define a domain to mean an independent wireless network in which connecting to its link requires IP level mobility (inter-domain mobility). Seamless inter-domain mobility is vital for achieving an ‘always best connected (ABC)’ [8] service, i.e. the best possible connection to support applications in use, while remaining globally reachable, anytime, anywhere. The possibility of having this ABC, largely, depends on the ability to utilise all the available access networks (AN) in any particular area at a time. In fact, seamless mobility between heterogeneous wireless networks (wireless links of different technologies) is one of the IP mobility management requirements for mobile communications operators as specified by 3GPP [9].

1.1.1 Mobility Management

When the Internet was first developed as a DARPA program named ARPANET [10], the idea was to connect stationary devices. There was no anticipation of a change in a device’s IP address during a communication session. The IP address in the original setup – and mostly the case until today – performs the dual role of identifying the station and providing its location. This becomes a problem when MNs that are connected to the Internet switch between ANs of different domains, which normally requires a change in the IP address. This change in IP address affects session continuity of all active connections because the transport-layer session states are identified using sockets – a combination of source and destination IP addresses; application port number

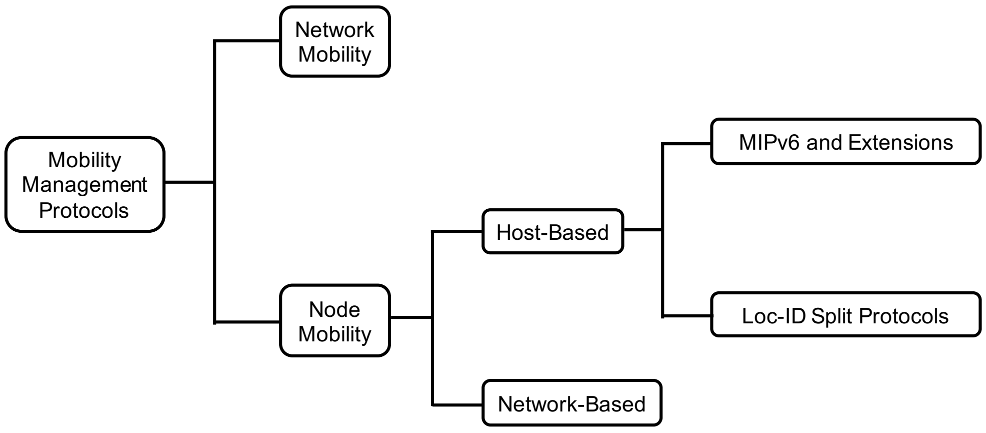


Figure 1.1-1 Categorisation of Mobility Management Protocols

of the transport protocol in use; and an ephemeral port number chosen by the source device protocol stack. As soon as any of the IP addresses changes, any communication session using that address has to be discarded and another session initiated using the newly acquired/configured IP address.

Mobility management protocols (MMP) are designed to solve the problem of session continuity at the point of a device movement. The protocols enable an MN to change its PoA, i.e. to handover from one AN to another, while maintaining all ongoing data sessions on the device. As shown in Figure 1.1-1, the mobility of a host can be broadly classified into node mobility or network mobility [11]. Node mobility deals with a host moving from one point to another; network mobility deals with an edge router (termed access router or simply AR) moving with the network or subnetwork (of nodes) attached to it. Node mobility can be further classified as host-based or network-based mobility. In host-based techniques, the mobile device handles all mobility signalling and the edge router does not participate[1] in any aspect of mobility. On the other hand, mobility signalling is handled by network component(s), usually the AR, in network-

[1] Network components are involved in mobility signalling in one of Fast MIPv6 configurations.

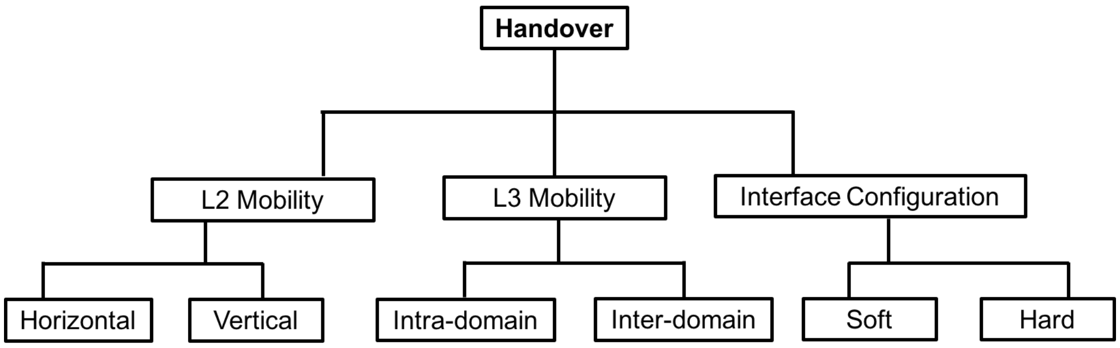


Figure 1.1-2 Handover Classification

based techniques. The host-based Mobility protocols can be subdivided to include most of the Mobile IPv6 (MIPv6 [12]) based techniques on the one hand, and some Locator/Identifier (LOC/ID) Split Protocols [13] on the other.

MMP come into play at the point of changing PoA and the action to take depends on the type of handover that the MN or the AR is executing. Handovers are usually defined based on the processes that take place at the different layers of the TCP/IP protocol stack (see Figure 1.1-2) during the movement event. As it relates to layer two (L2), handovers can be classified as horizontal handover, when an MN (or AR[2]) moves within ANs of the same technology, e.g from one Wi-Fi access point (AP) to another or one cellular BST to another – only one type of wireless interface is involved in the process. This is also referred to as intra-radio access technology handover. Vertical handover (also inter-radio access technology handover) is the other category, when the MN moves between links of different technologies, as it is common with Wi-Fi and

[2] At the egress interface of the AR, all the handover processes are similar to that of the MN. Hence, only the MN's procedures will be discussed.

cellular or Wi-Fi and small cells (e.g. Femtocell) today – where two types of wireless interfaces are involved.

These two handover techniques could also be intra or inter-domain with respect to layer three (L3) or IP-layer handover management. When both the previous and the target networks are owned by the same service provider, then the handover is considered intra-domain. Inter-domain handover happens when the previous and the target ANs are in different domains and under different administrative control. Inter-domain handover is usually associated with a change in the IP address. Host-based mobility protocols, such as MIPv6 and LISP-MN[3] [14], by their design support inter-domain mobility with the use of immutable IP address for setting up communication sessions.

Handovers could also be considered as hard or soft depending on the point in time that the active interface is configured during the movement event. When the interface is configured before the previous connection is released or when the interface needs no configuration because the target network has similar configuration information to the previous, the handover is considered soft. If there is a total disconnection from the previous PoA before the interface is configured to connect to a target PoA, then the handover is considered hard.

Handovers mostly cause degradation of service quality, due to the latency involved in the process and the associated packet loss caused by the delay in switching the connection.

[3] Locator Identifier Separation Protocol Mobile Node

1.1.2 Locator/Identifier Split Architecture

Concept

The idea behind LOC/ID split architectures [15, 16] is to come up with a new reference model that solves the limitation of current Internet addressing architecture. The protocols designed based on this concept proposed decoupling the dual role of IP address (identifying and locating a node) to form a separate Locator (LOC) for determining MN's position on a network and an Identifier (ID) to provide a (unique) identity for the node. The IRTF routing research group (RRG) acknowledges that decoupling the IP namespace into distinct LOC and ID namespaces is vital towards finding the solution to the problems of routing scalability, multi-homing, and inter-domain traffic engineering (ITE) faced by the Internet today [17, 18]. Site renumbering and Internet transparency are also some of the challenges that decoupling of the IP namespace is set to address [19].

Edge networks today want to be able to change their Internet service providers (ISPs) without renumbering their networks – as site renumbering can be costly to organisations. When a network is renumbered, IP addresses hard coded in access control lists, firewalls, web server configuration files, or the addresses stored between sessions (e.g., in P2P applications) could be forgotten in the different files and cause downtime to the network. Avoiding site renumbering is usually achieved with the use of provider independent (PI) addresses, which allows a network to maintain its address space regardless of its top-tier ISP. PI addresses also enable site multi-homing as a network advertises its prefixes (the PI addresses) to all its ISPs as a way of endowing the network with fault tolerance and load balancing capabilities.

The efficiency of using PI addresses to avoid site renumbering and allows for multi-homing of networks comes at the cost of injecting more routes into the BGP with ISPs not being able to aggregate the PI addresses. Advertising these addresses in the default-free-zone (DFZ) affects the scalability of Internet routing by making the forwarding information base of the DFZ to be growing at a greater than linear rate [16]; this is not likely to scale. ITE in the current Internet is limited by the fact that BGP by design will only advertise one route for each prefix.

Separating the core routing address space – using LOCs – from the edge networks – using IDs – will enable high-level aggregation of addresses, reducing the number of globally announced prefixes in the DFZ. It will also allow for distribution of traffic using the different gateways of a domain as advertised by the mapping systems for ITE and multi-homing. For mobile networks, separation of LOC and ID of a mobile device ensures that changing the PoA to a network or changing the active interface on a device does not affect ongoing sessions. The transport (and upper) layer sockets are bound to the device's ID in this case, and routing is achieved using the device's LOC. This is necessary to ensure session continuity when the MN moves to a new AN.

IPv6 provides enough addresses to identify the billions of devices on the Internet for end-to-end connectivity, which is a desirable aspect of Internet transparency as intended by the original design of the global network. Another aspect of Internet transparency, 'immediate hosts not essentially modifying packets' [19], has today being defeated with the presence of middle-boxes such as NATs. LOC/ID protocols are architectures that are geared towards providing

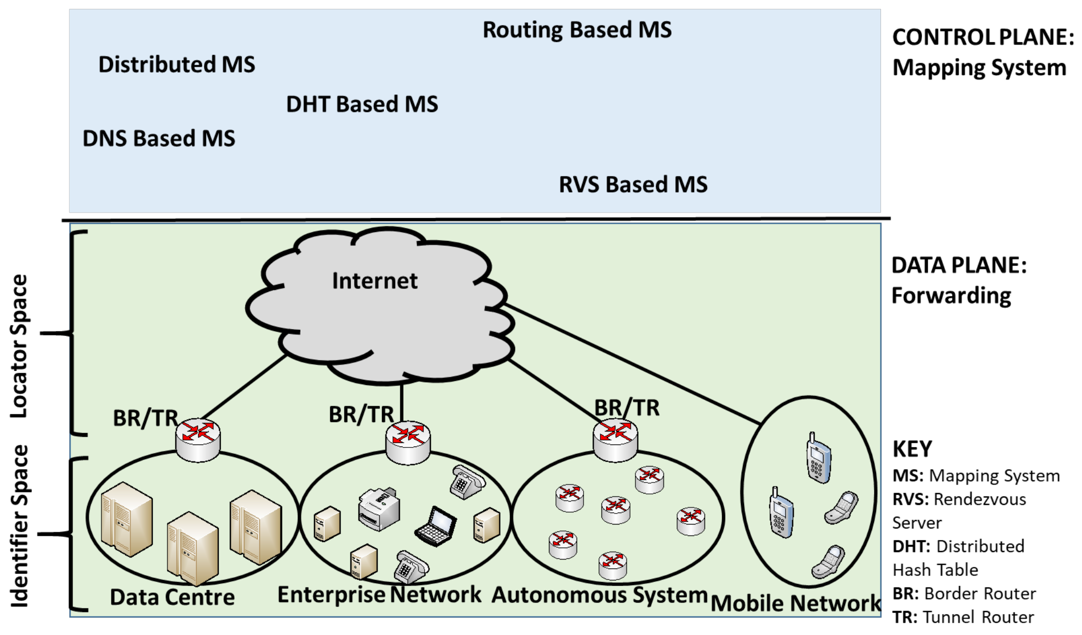


Figure 1.1-3 LOC/ID Split Architecture Showing Control/Data Planes and Address Hierarchy

transparency of the Internet with all hosts being universally addressable and removing the need for the middle-boxes that hinder the end-to-end connectivity.

Components

The LOC/ID split architecture is composed of two main components, the control, and the data planes [19], as shown in Figure 1.1-3. The control plane consists of the mapping system, which maintains the relationship between the LOCs and the IDs and capable of propagating LOC updates to ensure reachability of network nodes at any time. The mapping systems could be implemented using different technologies as shown in the figure – distributed database as used in LISP-DDT[4] [20]; use of core routing infrastructure by LISP-ALT[5] [21]; use of DNS infrastructure [22]; use of DHT[6] technology [23]; or the use of rendezvous server (RVS) [24].

[4] DDT: Delegated Database Tree
[5] LISP ALT: LISP Alternate Topology
[6] DHT: Distributed Hash Table

The data plane is responsible for tunnelling/forwarding of packets to their destinations and whether a packet is tunnelled (encapsulated in another packet) or forwarded (change/add the destination address before sending) is determined by the category of LOC/ID split protocol in use. Two main categories are usually defined [25], network-based and host-based LOC/ID protocols depending on where the control and data signals are handled. The common topology of network-based LOC/ID split protocols is to have a border router (BR, sometimes called tunnel router [TR]) serving as the gateway to a network with LOC address on its egress interface and an ID on its ingress.

The nodes behind the BR (see Figure 1.1-3 above), which can be grouped under an autonomous system, an enterprise network, a mobile network – or the likes – communicate with each other using IDs. The BR queries the mapping system to determine the location of target nodes by using the destination ID the router gleans from Internet-bound packets coming through its ingress interface – packets from the local network. The router uses the information in the query reply from the mapping system to encapsulate the original packet with a new header or rewrites the packet's destination address before sending the message to its destinations through the DFZ. Example of protocols in this category include LISP-MN, IVIP[7] [26] and ILNP[8] [27].

For networks with no LOC/ID split capability, the functionality of tunnelling/forwarding of packets is rest with the endpoints, which are in most cases mobile nodes (MNs). These protocols (for which the endpoints handle signalling) are referred to as host-based and include some of the network-based

[7] IVIP: Internet Vastly Improved Plumbing
[8] ILNP: Identifier-Locator Network Protocol (implementation on Site Border Routers)

protocols mentioned earlier but also having host-based versions to work on endpoint devices, for example, ILNP, LISP-MN, and IVIP. HIP[9] [28] is also another protocol in this category.

1.1.3 Wireless Networks

The two most popular networking technologies that enable users to have a wireless connection to the Internet are cellular and Wi-Fi. The cellular network has seen tremendous development over the years with two main competing standards, third generation partnership project (3GPP) [2, 3] and 3GPP-2 [5]. The 3GPP's 4G network, termed the LTE[10], marks the gradual switch to fully packet-switched networks (all-IP networks) and the deprecation of circuit-switched networks of the earlier generation cellular systems – 1G, 2G, and 3G. The LTE's backbone network, termed the evolved packet core (EPC), is responsible for mobility management in the 4G cellular system, and MIPv6 as well as PMIPv6 are some of the mobility protocols adopted in the cellular standard [29]. Major 3GPP-2 developers have adopted LTE as their 4G technology.

While the cellular network is characterised by wider coverage and ubiquitous availability, Wi-Fi, on the other hand, provides high-speed data services within a small area such as an office building, a campus or a public area. It is one of the fastest-growing wireless technologies in the world [30] owing to the necessity of connecting wireless nodes to the network without the need for cables. Wi-Fi initially – and still – serves as extension of LAN networks but it is today used for different purposes: for home broadband services; as extension

[9] HIP: Host Identify Protocol. The original design has not network-based operation specified.
[10] LTE: Long-Term Evolution

to cellular networks in the form of femtocells [31]; as the wireless access technology for data offloading [32]; and as added services from Telecom operators in the form of hotspots [33]; among other uses.

1.2 Problem Domain

To ensure seamless interaction between cellular and Wi-Fi networks, the 3GPP's EPC has some defined components that enable Wi-Fi (and WLANs[11] in general) to plug into the LTE network via evolved packet data gateway (ePDG) [34]. This will ensure a smooth handover of connectivity for an MN as it moves into or out of a Wi-Fi area. However, with most Wi-Fi networks owned privately and outside the control of most telecom operators, a switch of connection from cellular to Wi-Fi and vice-versa is usually inter-domain in nature. Inter-domain mobility management is vital in mobile networks to allow for seamless handovers, and to simplify cooperation between wireless network service providers. Seamless inter-domain mobility could also improve some of the routing techniques designed to reduce potential congestion in LTE networks such as IP Flow Mobility (IFOM) [35].

MIPv6 and its extension have been standardised to support inter-domain mobility on IPv6 networks, but there are calls [13, 18, 28, 36] for the adoption of LOC/ID protocols in future wireless networks not just for superior mobility techniques but for other benefits the protocols claim as highlighted in section 1.1.2. The fact that LOC/ID protocols have no concept of home network as with the MIPv6 and its many extensions, they are more suitable for inter-domain

[11] WLAN: Wireless Local Area Network

mobility since, as is the case with the DNS, for instance, no single provider may own the mobility anchors and mapping system. However, while using a LOC/ID protocol, such as LISP-MN, IVIP, ILNP or HIP, will enhance inter-domain mobility, they share three problems with the standardised host-mobility protocol, the MIPv6: handover delay; packet loss; and throughput degradation at the point of inter-domain handover as will be discussed in the following subsections. Inter-domain mobility can take place with both horizontal and vertical handovers as discussed below.

1.2.1 Horizontal Handover

Horizontal handover happens in a homogeneous wireless environment (HoWE) where all the access links are of similar technology, and an MN moves between them. When an MN moves to a new wireless domain by establishing a connection using the same interface, the process takes a long time because of the procedures involved. These procedures, discussed in chapter three, could be summarised as follows: movement detection (MD), duplicate address detection (DAD) and mapping update.

As stated earlier, all LOC/ID protocols rely on a mapping system that tracks MN's location, and have some defined control messages to ensure that location information is updated after a handover is completed. The number of these messages exchanged between the MN, and the mapping system is dependent on the protocol, but at least two messages are necessarily implemented by most protocols – a movement notification from the MN and an acknowledgement from the mobility anchor. Some protocols also provide a mechanism to tell the correspondent node (CN) of the change in location. LOC/ID split protocols' handover procedure is similar to the technique used by the MIPv6, and the protocol’s handover delay could give us an idea of how long a LOC/ID protocol’s handover could take. And since the first two procedures of MD and DAD are protocol independent, we can deduce that handover delay on LOC/ID protocols is, give or take, the same as the MIPv6.

Horizontal handovers are usually soft and intra-domain when executed in Wi-Fi hotspots and cellular networks, and they are usually hard and inter-domain in places with distinct and independent wireless networks.

1.2.2 Vertical Handover

Vertical handover is usually inter-domain in nature and could also be executed as soft or hard handover. This type of handover only takes place in a heterogeneous wireless environment (HeWE), where there are access links for both Wi-Fi and cellular/WiMAX connectivity. The MN, in such an environment, usually uses its interface with higher priority, commonly the Wi-Fi, for communication and only switches to the cellular when it loses the Wi-Fi connection. This switch is done in a hard handover fashion (Wi-Fi –> cellular) causing abrupt loss of connectivity, since the prioritised link has to be unavailable before the cellular network is configured for connectivity, and the three handover processes highlighted above need to take place before communication is resumed.

Conversely, when the Wi-Fi link becomes available while the MN is connected using a cellular link, the switch is executed in a soft handover fashion (cellular –> Wi-Fi) as the two links become active for some time before the handover. The running sessions are only switched to the Wi-Fi interface after the interface is fully configured and ready to resume communication. Soft handovers may be accompanied by little or no delay since the transition to the targeted interface is done gradually, and the impact may not be felt by active applications. Although soft handovers eliminate the need for the first two procedures of MD and DAD, it is still necessary to update the mapping system before packets are redirected through the new access link.

1.2.3 Impact of Inter-Domain Mobility on Applications

There are research works such as [38-40] that show that the average handover delay for a MIPv6 network is about three seconds (3s). Such a duration is a long enough latency to perturb even some best-effort type applications not to mention delay sensitive ones, such as voice, for instance, for which delay has to be less than 70ms to avoid voice call quality degradation [41]. Another delay and loss sensitive application is video streaming, which according to Cisco's VNI [6], forms the major traffic on mobile networks today. Most video streaming applications use TCP as their transport protocol, and it is common knowledge that TCP is highly sensitive to packet loss; hence frequent handovers by an MN will significantly affect the quality of any video that users may be streaming.

According to Biernacki [42], as small as 0.1% packet loss can cause TCP throughput to oscillate, which in turn affect the quality of video that a user is viewing. Gorius et al. [43] reported that 0.5% loss resulted in up to 25% reduction in TCP throughput in the experiment the authors conducted to measure the impact of loss on TCP performance, which in turn affects dynamic HTTP streaming. It was also shown in the work of Elkhatib et al. [44] that web browsing using HTTP2 is also quite sensitive to packet loss. When handovers are frequent, even if the loss is within what normally could have been an acceptable boundary, it becomes a severe issue. With Dynamic Adaptive

Streaming over HTTP (DASH [45]) being the most popular video streaming technique on the Internet, understanding how it performs on a LOC/ID-based network is essential in broadening the understanding of the mobility protocols' performance.

The impact is even more pronounced with UDP-based applications. TCP's congestion window mechanism will back off from sending packets if no acknowledgements are received after some time; UDP has no such mechanism. Hence, a UDP-based application will continue to send packets to the MN through the duration of the handover resulting in poor utilisation of network resources since all the packets that arrive after an MN has disconnected from an AR will be dropped. It is pertinent to mention here though, some applications that use UDP as transport protocol are equipped with some mechanism to detect when the remote node is no more responding to the datagrams sent.

1.3 Research Hypothesis

The LOC/ID protocols are considered in the IETF as the possible solutions to the inter-domain mobility problem on the Internet [17, 46]. The research in this area as highlighted previously has produced a number of proposed solutions [14, 26-28, 47-49]. From a theoretical review of the LOC/ID protocols' handover operations and the similarity of the process to that of MIPv6, there is a likelihood of high packet losses occurring amid a communication session as an MN loses its connection during handover before reconnecting to a targeted link. We argue that redirecting these packets to a buffer node nearby and forwarding the packets to the MN after reconnection will mitigate the effects of the handover by reducing the performance degradation inherent with the process. The buffer node will serve MNs connected to a group of independent access links that subscribe to the service in a particular area.

Some questions will arise from the preceding discussions as itemised below, and this thesis will try to answer them.

1. How does mobility with LOC/ID split protocols differ from that of MIPv6 in terms of handover delay, service disruption time, throughput, and packet loss?
2. How much impact does the mobility with LOC/ID split protocols have on popular internet applications such as video, voice and file download applications?
3. Will the results in questions 1 and 2 differ when experiments are run in different wireless environments – homogeneous or heterogeneous?
4. Will buffering of packets during handover improve mobility performance of LOC/ID split protocols?
5. Will the introduction of a buffer node affect the architecture of a LOC/ID protocol in such a way that it significantly changes the protocol's mode of operation?
6. Will introducing the additional node mean introducing new control messages; if yes, how much impact will that have on the backbone network?
7. Could LOC/ID split protocols serve as a better replacement for MIPv6 and its many extensions?

1.4 Research Aims and Objectives

The main aim of this research work is to introduce an improved Locator/Identifier Split Architecture (ILISA) designed to enhance the mobility of nodes running a LOC/ID protocol by mitigating packet loss and reducing service disruption in handovers. A new network node, Loc-server, is central to the new architecture with the task of buffering incoming packets during handover and forwarding the packets to the MN on the completion of the node's movement process. ILISA is designed to improve the architectures of some of LOC/ID split protocols as will be discussed in detail in chapter four. We present a highlight of the research objectives below, but the detail aims and objectives of the research are discussed in chapter three.

1. Implement a LOC/ID protocol on a laboratory testbed including all the required components to support its mobility mechanism.
2. Analyse inter-domain mobility performance of MIPv6 and a LOC/ID protocol on the testbed emulating both homogeneous and heterogeneous wireless network environments.
3. Design and implement a buffer node; we called Location Area Server (or simply Loc-server) that will be storing packets for pre-registered MNs and forwarding the packets at the request of the MNs.
4. Implement new signalling mechanisms on the MN to enable the mobile device to interact with the Loc-server.
5. Re-evaluate the performance of the LOC/ID protocol with the support of Loc-server to determine how much improvement the new node brings to the protocol in question.

1.5 Thesis Structure

The introduction chapter is followed by six other chapters as highlighted below:

Chapter Two touches on the history of LOC/ID split from conception to state of the art. It discusses four host-based LOC/ID split architectures, LISP-MN, IVIP, ILNP and HIP, focusing on their control and data planes and especially detail related to mobility. Each protocol's discussion is followed by a review of other research works to improve its operation except IVIP, for which no published research works geared towards improving mobility was found. The chapter also discusses MIPv6 as the protocol serving as the benchmark for evaluating LOC/ID split protocols. The review of research works on the architectures is summarised at the end of the chapter.

Chapter Three discusses the requirements that any system geared towards improving LOC/ID split protocols' mobility process needs to satisfy. It talks about the different inter-domain mobility scenarios in heterogeneous and homogeneous wireless environments. It also discusses the two types of handovers which occur in the two wireless environments under consideration – horizontal handover in homogeneous wireless environments and vertical handover in heterogeneous environments. The chapter also establishes the major requirements of any network architecture based upon a LOC/ID split concept. It also provides a detailed discussion of the aims and objectives of the research.

Chapter Four presents the design description of the newly developed Improved LOC/ID Split Architecture (ILISA). It presents the different components of the architecture and the necessary modules that need to be implemented on the

MN and Loc-server for the architecture to function. The chapter also details how the four LOC/ID split protocols' architectures and MIPv6 discussed in chapter two can be improved with the Loc-server introduction. The chapter concludes with the discussion of Loc-server's deployment in real life networks and identified three potential implementation scenarios – Telecom Operators network, Internet Service Providers and Community Networks.

Chapter Five provides a detail explanation of how the system design described in chapter four was realised. There is a discussion on the LISP-MN and MIPv6 implementations used for the evaluation. The chapter describes the Loc-server implementation as well as the different parts of the LISP-MN's code that were changed in order build the new architecture. It also describes how the different design components are realised in the implementation work.

Chapter Six is the evaluation chapter to analyse the performance of the improved architecture. It begins with describing the evaluation methodology and strategy as well as the objectives of the evaluation. The chapter first analyses qualitative and quantitative features of LISP-MN performance against MIPv6. It then focuses on evaluating the newly built architecture against vanilla LISP-MN. Performance of video applications on the different protocols is also presented.

Chapter Seven concludes the thesis by summarising its contributions. It also talks about the future work to enhance the solution and presents a general discussion on mobility with LOC/ID split protocols. It also discusses our major findings.

CHAPTER TWO

LOCATOR/IDENTIFIER SPLIT PROTOCOLS

In this chapter, we present a brief history of the emergence of the LOC/ID Split protocols and their classification. We also highlight the architecture as well as control and data plane operations of four different protocols in this field. Although there are several areas of interest concerning the deployment of these protocols, our discussion and critique will mostly focus on the mobility aspects as mobility of nodes forms the basis for this research work. We also provide a discussion on MIPv6 as the IETF standardised host mobility protocol that provides us with the baseline performance of an MN’s handover. The chapter ends with a summary of the proposed and implemented LOC/ID split solutions to improve mobility.

2.1 A Brief History

One of the earliest published materials – if not the first – that came close to defining the difference between a name (read ID) and an address (or LOC) as we know them today is the Internet Experiment Note Number 19 [50]. The author stated in his model that “a name indicates what we seek [on a network] and an address indicates where it is”. He further asserted about the name, “The string needs not to be meaningful to all users, and need not be drawn from a uniform name space” while address on the other hand “must be meaningful throughout the domain, and must be drawn from some uniform address space.” LOC/ID protocols are designed based on this idea where the IDs might not be understood or forwarded outside a LOC/ID protocol’s domain in contrast to the LOCs, which are globally routable addresses and understood by the core routers.

The first proposed decoupling of IP namespace by Chiappa [51] suggested the creation of endpoint identifier namespace for a new fundamental object in internetworking, the endpoint. However, O’Dell’s ‘8+8/GSE[12] [52] was perhaps the first LOC/ID split proposal that was based on the concept of “strong distinction between system identity and location”. Although the idea of IPv6 8+8 was said to be first muted from emails posted by Bob Smart (02 Jun 1994) and Dave Clark (11 Jan 1995) on an IETF mailing list [53].

[12] Global, Site, and End-system address elements

The IETF Internet Architecture Board's Routing and Addressing Workshop held in 2006 in Amsterdam [46], the Netherlands, further motivated the emergence of many other proposals to tackle the problem of the scalability of Internet routing architectures. The main discussion in the workshop was the alarming growth of the routing information base (RIB) in the DFZ, which was caused by the addition of more prefixes to the table. The high increase in the number of prefixes advertised was driven by a number of factors as highlighted in chapter one, including customer networks preferring PI addresses to give them the flexibility of selecting their ISPs and avoiding renumbering when changing a provider; the need for multi-homing through different ISPs; and the need for ITE. The workshop also identified the potential of having billions of handheld gadgets on the Internet going forward, based on the trends in mobile technology as it were, that may have a significant impact on global routing scalability.

The routing research group (RRG) of the IRTF, in its 'Design Goals for Scalable Internet Routing' document [17], identified scalable support for mobility as one of the design goals for any architecture that is geared towards solving the problem of Internet routing scalability highlighted above. Furthermore, scalable support for traffic engineering, scalable support for multi-homing, decoupling location and identification, simplified renumbering, are among the other goals listed in the IRTF document. The RRG recommended that mobility mechanisms should take into account the scalability of Internet routing and be designed in such a way that no dynamic updates of prefixes are injected into the global routing system as a result of MNs' movement between PoAs.

2.2 Classification

As highlighted in section 1.1.2, LOC/ID split protocols could be classified as either network-based or host-based protocols depending on the network components responsible for control and data signalling and where changes on the Internet need to be implemented for a protocol to operate. We have also discussed that network-based protocols could be further categorised into map-and-encap and address rewriting techniques. Protocols that tunnel packets to their destination are usually referred to as map-and-encapsulate and include LISP-MN [14], MIPv6, IVIP[13] [26], eFIT[14] [47], and IPvLX [48]. Address rewriting protocols rewrite the destination address in the original IP header after confirming the destination LOC from the mapping system. Protocols in this category include ILNP[15][27], MILSA[16] [54], LIN6 [55], FARA [56], MAT[17] [57], and GSE [52].

As mentioned earlier, an MN is responsible for all its mobility signalling in host-based category of LOC/ID protocols with some of the network-based protocols such as LISP-MN, ILNP and IVIP providing host-based version of the protocols. Other protocols in this category include HIP, Shim6[18] [49], UIP[19] [59] and HRA[20] [60]. Host-based protocols are the focus of our research work for the fact that they require no or very minimal changes to the existing Internet architecture for their operations – potentially easier to be adopted on the current Internet.

[13] IVIP: Internet Vastly Improved Plumbing
[14] eFIT: A Scalable Routing System Design for Future Internet
[15] ILNP: Identifier-Locator Network Protocol (implementation on Site Border Routers)
[16] MILSA: Mobility and Multi-homing supporting Identifier-Locator Split Architecture
[17] MAT: Mobile IP with Address Translation
[18] Shim6: Level 3 Multihoming Shim Protocol for IPv6
[19] UIP: User Identifier Protocol
[20] HRA: Hierarchical Routing Architecture

Secondly, host-based approaches have a more direct impact on the performance of Internet applications on a device, and we are designing a new architecture to improve the performance of Internet applications.

As such, we carefully selected four protocols – LISP-MN, IVIP, ILNP, and HIP – whose architectures will serve as the guide towards the design of an improved LOC/ID split architecture that is compatible with many protocols in this category. The level of current research work on a protocol and its implementation have also informed our decision of selecting the four. LISP-MN is the most popular LOC/ID split mobility architecture with the highest number of research literature and community behind it. IVIP has a similar architecture to LISP (the parent architecture for LISP-MN, to be discussed later) and will work well with our architecture although there is little research work on it. ILNP, in my view, has the most elegant design of all the LOC/ID Split architectures, as it comes with minimal changes to the current TCP/IP architecture. HIP is also pursued in the research community and already deployed in a production environment at Boeing Aircraft company.

Table 2.2-1 below shows the level of conformance to the objectives set by the IETF RRG for any LOC/ID split protocol, and we can see that all the protocols considered have achieved the mobility objective. Objectives such as routing scalability and traffic engineering are associated with network-based deployments of LISP (for LISP-MN), IVIP, and ILNP and not with the host-based approach used in this work.

S/N	Protocol Name/Approach	RS*	Mo	MH	TE	Se	RC on	
							AR	MN
1	LISP-MN	Yes	Yes	Yes	Yes	No	No	Yes
2	IVIP	Yes	Yes	Yes	No	No	No	Yes
3	ILNP	Yes	Yes	Yes	Yes	Yes	No	Yes
4	HIP	No	Yes	Yes	No	Yes	No	Yes

***RS:** Routing Scalability; **Mo:** Mobility; **MH:** Multi-Homing; **TE:** Traffic Engineering; **Se:** Security; **RC:** Require Changes; **AR:** Access Router; **MN:** Mobile Node.

Table 2.2-1 Conformance to the Objectives Set by the IETF RRG

2.3 Locator/Identifier Separation Protocol (LISP)

2.3.1 Architectural Concept

LISP is a map-and-encap tunnelling protocol standardised by the IETF [61] that enables the separation of ID and Locator of a host in the IP network. The protocol proposes two namespaces (both IP addresses), the endpoint identifier (EID) serving as the node identity, and routing locator (RLOC) to determine the position of the node on a network. The EIDs are obtained from the EID block space and are used on the edge network for intra-domain routing (routing within an autonomous system). RLOCs, on the other hand, are globally routable addresses used at the core network for inter-domain routing and assigned to the border router named egress/ingress tunnel router (ETR/ITR), which marks the entry and exit points of a domain [62]. Hence LISP is sometimes described as a core-edge separation technique.

The ITR maps the destination EID of an outgoing packet to its corresponding RLOC using the mapping system, the ETR, on the other hand, receives and delivers packets destined to one of its EID-prefixes. The ingress and egress functionalities may be collocated on a single TR in a domain and simply referred to as XTR. The current mapping system in use is called LISP-DDT[15], a

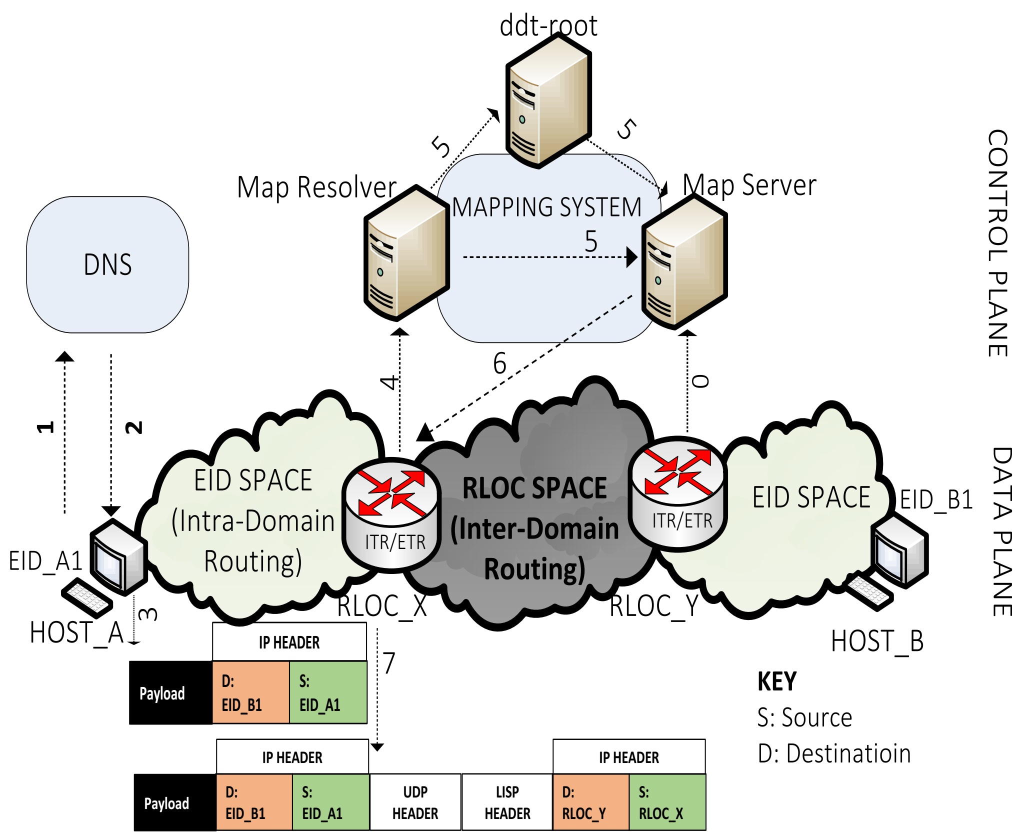


Figure 2.3-1 LISP Data and Control Plane Operations

hierarchical database conceptually similar to DNS. The mapping lookup process could be secured using the proposed LISP-SEC mechanism as detailed in [63] to ensure origin authentication of LISP messages, integrity and anti-reply protection.

2.3.2 LISP Control Plane

As shown in Figure 2.3-1, to communicate with HOST_B in a LISP network, HOST_A resolves the IP of HOST_B using the DNS, as per a normal IP session (1); EID_B1 is returned in the process (2). The packet is then forwarded towards the default gateway, which is the ITR (3). If the ITR does not have the mapping of the EID (to its RLOC – EID_B1 in this case) in its cache, it sends a map-request to its pre-configured map-resolver asking for the RLOC(s) of the ETR serving the requested EID (4). The map-resolver queries the mapping system for the mapping, and the map-request is routed, based on the EID, to the map-server[21] serving that particular prefix (5). A map-reply is sent to the ITR from the appropriate map-server (6), and the message constitutes the EID and the RLOC(s) of the target node; the weights and priorities of the RLOCs (if more than one is found); a nonce (a 24-bit value randomly generated) to avoid unsolicited map replies and to ensure integrity of the packet; and the time to live (TTL) that indicates the amount of time the mapping is valid for.

The ITR then encapsulates the packet to its destination (7). Note that the map-server that handled the map-request must have been earlier delegated (0) by the destination ETR using a map-register message to respond to map-request on behalf of the ETR. Hence the map-server publishes the EID-Prefixes in the mapping system on behalf of the tunnel routers that it is serving. Otherwise, the map-request query from the source ITR is directed by the mapping system to the destination ETR for the tunnel router to send the map-reply itself. LISP uses UDP encapsulation for control messages on port 4342.

2.3.3 LISP Data Plane

The ITR encapsulates and sends the packet, after resolving the EID-to-RLOC mapping, to the ETR discovered. The ETR de-encapsulates the incoming packets destined to an EID within its control and forwards it to the host. Replies would normally be reverse-tunnelled to their destination as the ETR caches EID to RLOC mappings based on what it ‘gleaned’ from the source packet. To optimise performance, the ITR may also cache some routes to speed up the

[21] Map Resolver and Server are usually collocated but shown differently here for the sake of explanation.

routing process and avoid querying the mapping system every time a communication channel is to be established. The caches are refreshed after a pre-defined timeout to avoid storing stale routes. LISP uses UDP encapsulation for data messages on port 4341.

For communication with a non-LISP-domain (when an ITR receives no response from the mapping system), the ITR would encapsulate and forward the packets to a proxy ETR (PETR), which then forwards the packet to its final destination. The PETR would normally be located in networks that do not have ingress filtering to ensure that outbound packets from the sending ITR redirected to the CN by the tunnel router are not dropped by the router's network due to the unknown source address. PETR would also have the sending domain's EID-prefix pre-configured in its database for the router to serve nodes in that domain. Replies from non-LISP-domains are sent to the proxy ITR (PITR) serving the LISP-domain, which encapsulates the packet and forwards it to its destination. As with ITR and ETR, PITR and PETR are also usually collocated to form a PXTR.

2.3.4 Mobility with LISP – The LISP Mobile Node (LISP-MN)

LISP-MN [64] is an approach defined to enable mobility with the LISP protocol. An MN is equipped with a lightweight version of ITR/ETR functionality and behaves like a single LISP-domain. The MN is configured statically with an EID, which is used by the transport and application layer to identify communication sessions. The map-server serves as the mobility anchor and tracks the location of the MN at any given time. For communication with a non-LISP CN, the MN sends and receives all packets via the PETR and PITR respectively.

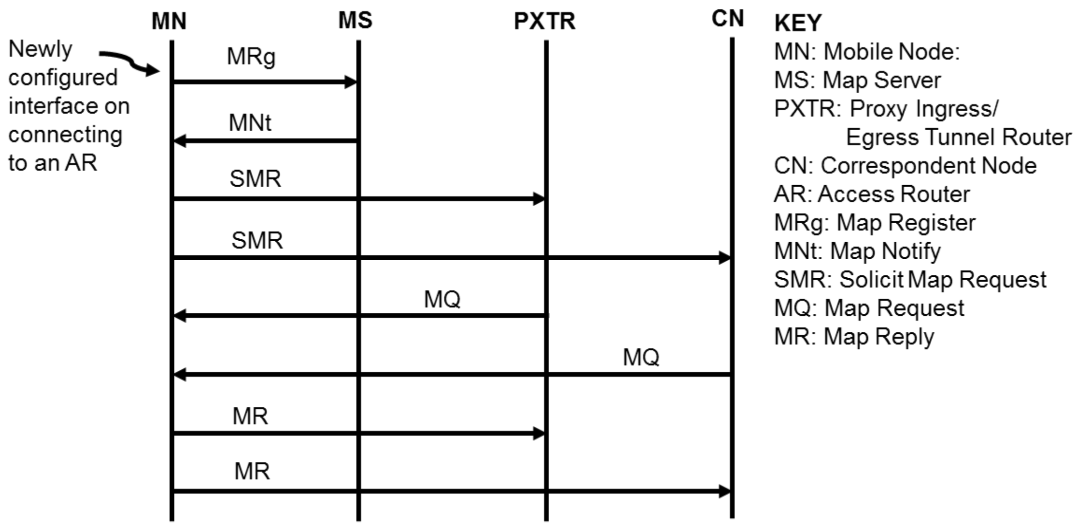


Figure 2.3-2 LISP-MN Control Plane Operation

Once an MN comes online or moves to a new network, it configures a new IP address (RLOC) on (one of) its interface(s) and sends a map-register message to its map-server to register the RLOC (its location) as shown in Figure 2.3-2. The server will authenticate the EID and reply with a map-notify message confirming that the EID-RLOC registration has been successful and an up-to-date mapping is published on the mapping system.

The MN will also send a solicit map request (SMR) message to its PITR, and to any LISP-based CN, if in the midst of communication, to invoke mapping update. Consequently, the PITR and the CN would send a map-request to the MN, to which the MN replies with a map reply (MP) containing the MN's new RLOC. This ensures that the PITR and any LISP-based CN have an up-to-date mapping of the MN's location.

2.3.5 LISP-MN Mobility Scenarios

As highlighted in the protocol's specification [65], LISP-MN can be deployed to work in five distinct connectivity scenarios:

i. A LISP-based MN establishing a communication session with a stationary node (SN) in a LISP domain.
ii. A LISP-based MN establishing a session with a non-LISP domain.
iii. A LISP-based MN session to another LISP-based MN.
iv. A node in a non-LISP domain communicating with a LISP-based MN.
v. A node in a LISP-domain communicating to a LISP-based MN.

We believe that the prevalent scenario in the near future will be a LISP-enabled MN establishing a communication session with a non-LISP correspondent node (CN) as the MN communicates to the legacy Internet. We take this position for two reasons: the different servers on the Internet that form the bulk of the CNs today are mostly located in non-LISP-domains and are themselves not LISP-enabled; secondly, the MNs themselves are likely to connect to a wireless network with no LISP capability and will have to tunnel its packets to a PETR in order to communicate. MN to non-LISP domain communication shown in Figure 2.3-3 below is in line with scenario (ii) and replies to the MN tally with scenario (iv).

To establish a session with the CN, an MN queries the DNS for the IP address of the host as explained in section 2.3.2. If the MN does not have the mapping of the returned EID in its cache, it sends a map-request to the mapping system, and a negative reply indicates that the CN is not located in a LISP domain and it is a non-LISP device. An MN encapsulates the outgoing packets to the PETR[22] as shown in Figure 2.3-3, and the proxy router de-encapsulates the

[22] All outgoing packets in this scenario are encapsulated to the PETR except for management protocols, such as DHCP

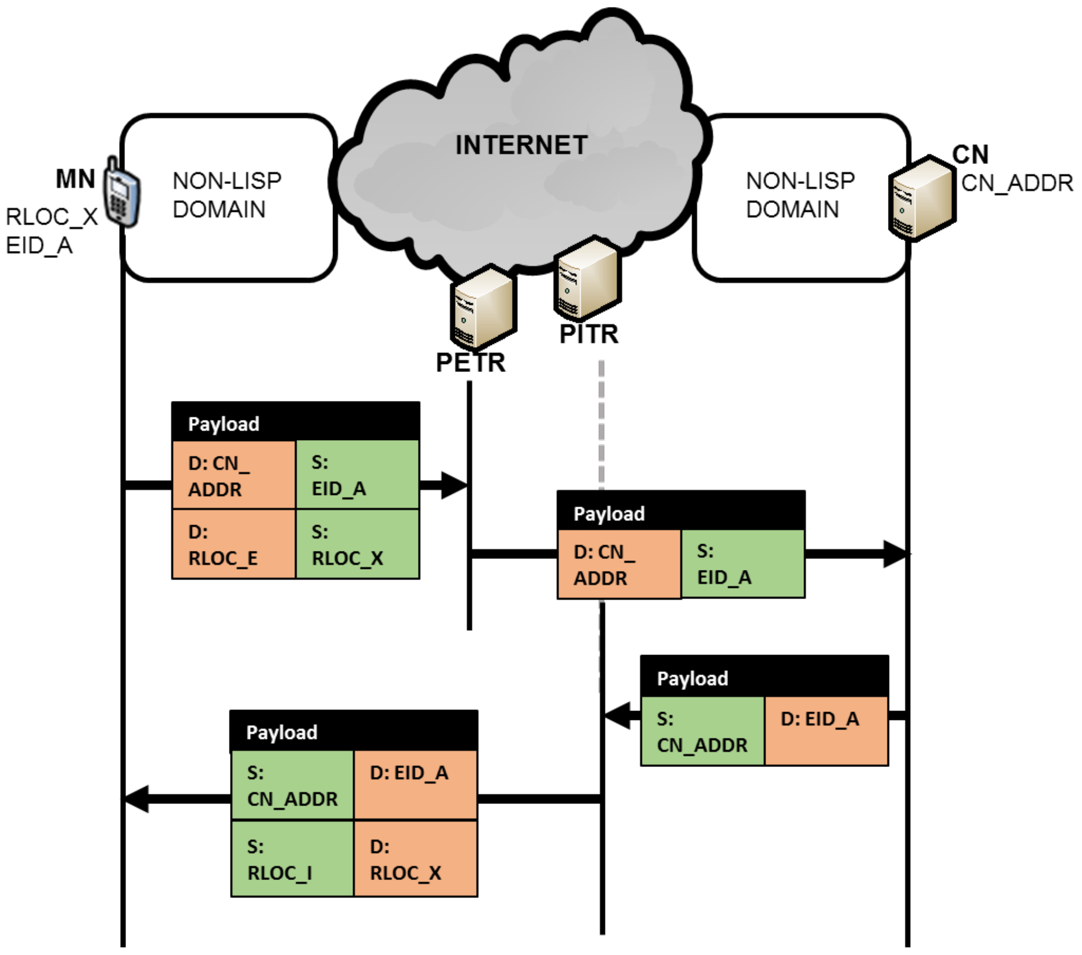


Figure 2.3-3 LISP-MN Data Plane Operation

packets and forwards using the conventional Internet routing system. Replies are sent by the CN using the MN's EID but would be delivered by the Internet routing infrastructure to the PITR serving the EID, which then forwards the packets to the MN. The PITR would advertise reachability of the MN's EID-Prefix in the default-free zone, to enable communication between an MN in a non-LISP domain to another legacy host. The PITR learns of anychange in the MN's location by either contacting the map-server or through the SMR messages explained earlier.

2.3.6 MN behind a NAT

To enable an MN’s reachability from behind a NAT, LISP specifies a re-encapsulating tunnel router (RTR), which allows an MN to be reachable from behind a NAT and serves as both signalling and data proxy for the MN [66]. On connecting to an access link, MN needs to check whether it is behind a NAT by sending a NAT Info-Request message to its map-server, a negative reply is received if the MN is not behind a NAT device. Otherwise, the map-server replies with a list of available RTRs and the actual source address and port of the MN. The MN then sends a data-map-register to an RTR with port number 4341. The message is an encapsulated map-register containing the RLOC of the RTR instead of the MN’s private IP address. This opens up the port number 4341 for LISP data on the NAT device.

The RTR de-encapsulates the message and forwards it to the specified map-server, and the map-server publishes RTR’s RLOC on the mapping system as the current location of the MN. All packets to the MN are routed to the RTR which forwards the traffic using the MN’s NAT state (MN’s EID, NAT device IP address, and the NAT port) it cached earlier, and the NAT device uses the port information in the received packets to determine the targeted MN.

2.3.7 Critique of LISP-MN

For an MN to maintain communication after a handover, it sends and/or receives at least five control messages with the map-server and PXTR for anchoring and forwarding services respectively. This ensures that packets could be rerouted to, and new sessions could be established with, the MN via its new location. In one of our previous pieces of work [67][23], we found LISP-MN's inter-domain handover in a homogeneous wireless environment to take up to 5.6 seconds (s) causing up to 8.4% loss (or 468 datagrams) of 5,571 datagrams sent in one minute [67]. In the event of one or more of the control packets being dropped along its path, the transmission will take a longer time to resume as the MN tries to recover by resending map-register to the map-server or SMR to the PITR. This additional delay will cause even more loss of packets especially for UDP based applications with no mechanism to detect the non-delivery of packets.

One of the most comprehensive analytical reviews of LISP-MN and some proposed improvements to the protocol was presented by Menth et al. [68]. The authors outlined three major problems with the LISP-MN as follows:

1. Unnecessary mapping lookups for an MN in a LISP domain: when an ITR performs a mapping look-up using the target MN's EID and gets the local locator (LLOC) and then looks up again using the LLOC to get the RLOC before forwarding the packet.
2. Double encapsulation headers as packets are encapsulated with the LLOC and further encapsulated with the RLOC in some communication scenarios.
3. Path stretch by routing via PITR and PETR which causes triangular and even quadrangular routing in some communication scenarios.

[23] This work is also presented in the evaluation of the research (chapter six), although there are some differences with the results in the paper as the scripts used were updated to provide more accurate measurements.

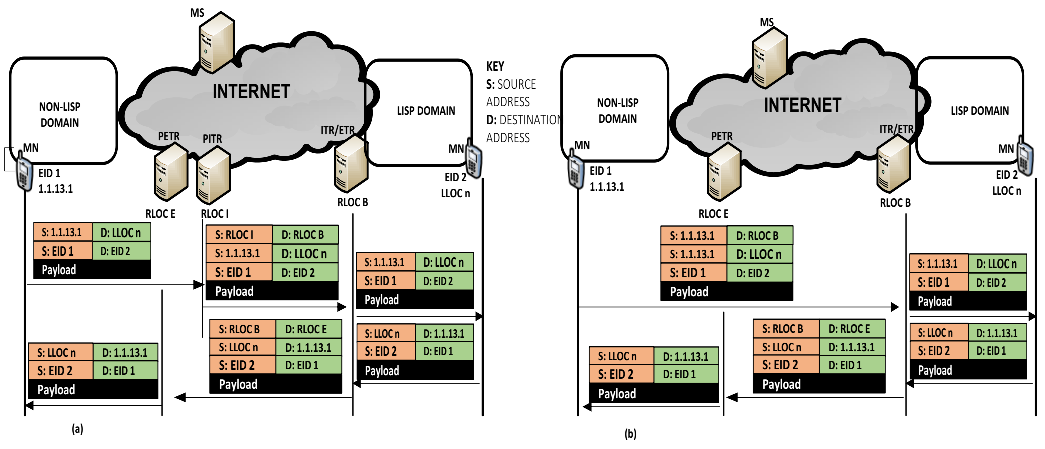


Figure 2.3-4 Packet Encapsulation and Forwarding in Scenario 3, before (a) and after (b) Improvements

To mitigate the problems identified, the authors proposed some improvements to the LISP architecture through the introduction of a local mapping system (LMS) and making the MN be 'location-aware.' On coming onto a network, the MN should query the mapping system with its assigned care of address (CoA) and a negative reply indicates that it is in a non-LISP domain and the CoA is an RLOC. If an RLOC is returned, the CoA is an LLOC, and the MN is in a LISP domain. The MN may then doubly encapsulate the packets directly to the ETR of the targeted MN and in the process avoid sending the packets to the PITR as shown in Figure 2.3-4 (depicting scenario iii of section 2.3.5). This removes the need for the triangular routing when an MN in a non-LISP domain communicates with another MN in a LISP domain.

The introduction of an LMS ensures that an MN in a LISP domain can easily determine if its targeted stationary node (SN) is in the same domain and forward packets to it without encapsulation and avoiding triangular routing in the process. This is in contrast to the normal traffic flow when an MN in a LISP-domain communicates with an SN in the same domain.

The introduction of a new node in the architecture, the LMS, will improve the data plane operation of the protocol but not the handover. And, although the improvements are likely to improve the protocol, no implementation and testing were presented by the authors.

Gohar and Koh [69] proposed a distributed handover scheme for an MN communicating with a CN in the same LISP-domain. The ARs are equipped with the tunnel router functionality and maintains an EID-LLOC cache of all devices in the network. The MN's new AR performs handover signaling on behalf of the MN by exchanging map-request and MP with the node's old AR, after which a bi-directional tunnel is established between the two. The new AR also exchanges the map-request and MP messages with the AR of the CN, and the AR (CN) updates it EID-LLOC cache to reflect the MN's new location. The CN's packets are afterward sent directly to the MN's new location. A numerical analysis shows that the distributed scheme performs better than other schemes it is compared with in terms of handover latency. These schemes included LISP-MN and other centralized solutions such as LISP seamless mobility support (SMOS) scheme [70], and LISP-PIMP [71].

While the solution takes away most of the signaling from the MN – the solution is silent on MN to map-server messages – the involvement of the routers means that, it could only be deployed if routers are LISP-enabled, thereby limiting its deployment in real life. There is also the need to deploy the Media Independent Handover (MIH) 'link-up' event to enable the routers to know when an MN is attaching to it, which make the new system dependent on another external program.

2.4 Internet Vastly Improved Plumbing (IVIP)

2.4.1 Architectural Concept

IVIP [26, 72] is another map-and-encap solution that proposes the separation of the core network from the edge – as has LISP – by providing two namespaces (addresses): scalable provider independent address space, called map address block (MAB) for edge networks; and core address space used in the DFZ. The MAB is usually divided into groups of networks called micronets and is assigned to user networks according to the size of the user's network. A micronet will have an xTR (IVIP uses ITR/ETR as in LISP) as a means of communicating with other micronets and non-IVIP networks. The core addresses are used to tunnel packets from the ITR to the ETR at the ingress point of targeted edge network. A Query Server database is used as the mapping infrastructure to maintain the relationship between the core and the MAB addresses. An ITR may request for mapping either from a local query server or one or more layers of the caching query server.

IVIP's data plane operations are mostly similar to that of LISP. The ITR resolves the mapping based on the destination address on the packet received from its edge network before it tunnels the packet to the destination ETR, which de-encapsulates and delivers the packets after confirming the MAB to be one of its own. DITRs (Default ITRs in the DFZ, similar to the LISP's PITR) are required to tunnel packets from non-IVIP networks destined to MAB addresses. IVIP uses IP-in-IP tunnelling as oppose to LISP's UDP tunnelling. Regarding the control plane, the ITR resolves the mapping of MAB addresses from a local resolving query server (QSR) in their network or their ISP's network. The query servers query other nearby authoritative query servers using a DNS mechanism to provide the required mapping.

2.4.2 Mobility with IVIP – The Translating Tunnel Router (TTR)

To enable mobility, IVIP uses a translating tunnel router (TTR) [73] to serve as the anchor point and at the same time an ETR for the MN. A two-way tunnel is created using the MN's CoA at one end and the TTR's address at the other end through which packets are sent and received. The inner header contains the MN's MAB as source and the CN address as the destination, while the outer header contains the MN's CoA and the TTR's IP address. TTRs are similar to MIPv6 HAs although an MN can use the closest TTR to it and a change of the anchor point will only be required if an MN moves some 1000km away from its current TTR. A mapping update of the current TTR for the MN is published on the mapping infrastructure. Whittle et al. explained in [73] that TTRs may be located at Internet peering points or within access networks of Telecom operators.

An MN will first connect to a central TTR after resolving the router's IP address from the DNS, and the TTR management system determines whether it is the closest router based on an MN's location, otherwise, it transfers the MN to the closest TTR. The MN will run specialised tunnelling software provided by the TTR Company for setting up the tunnels between the device and the TTR when the device comes online or after its handover. The software could be globally standardised or a propriety implementation since it operates only between the MN and the TTR company's routers. As part of the protocol's specification, a TTR never initiates communication with the MN although TTR service providers

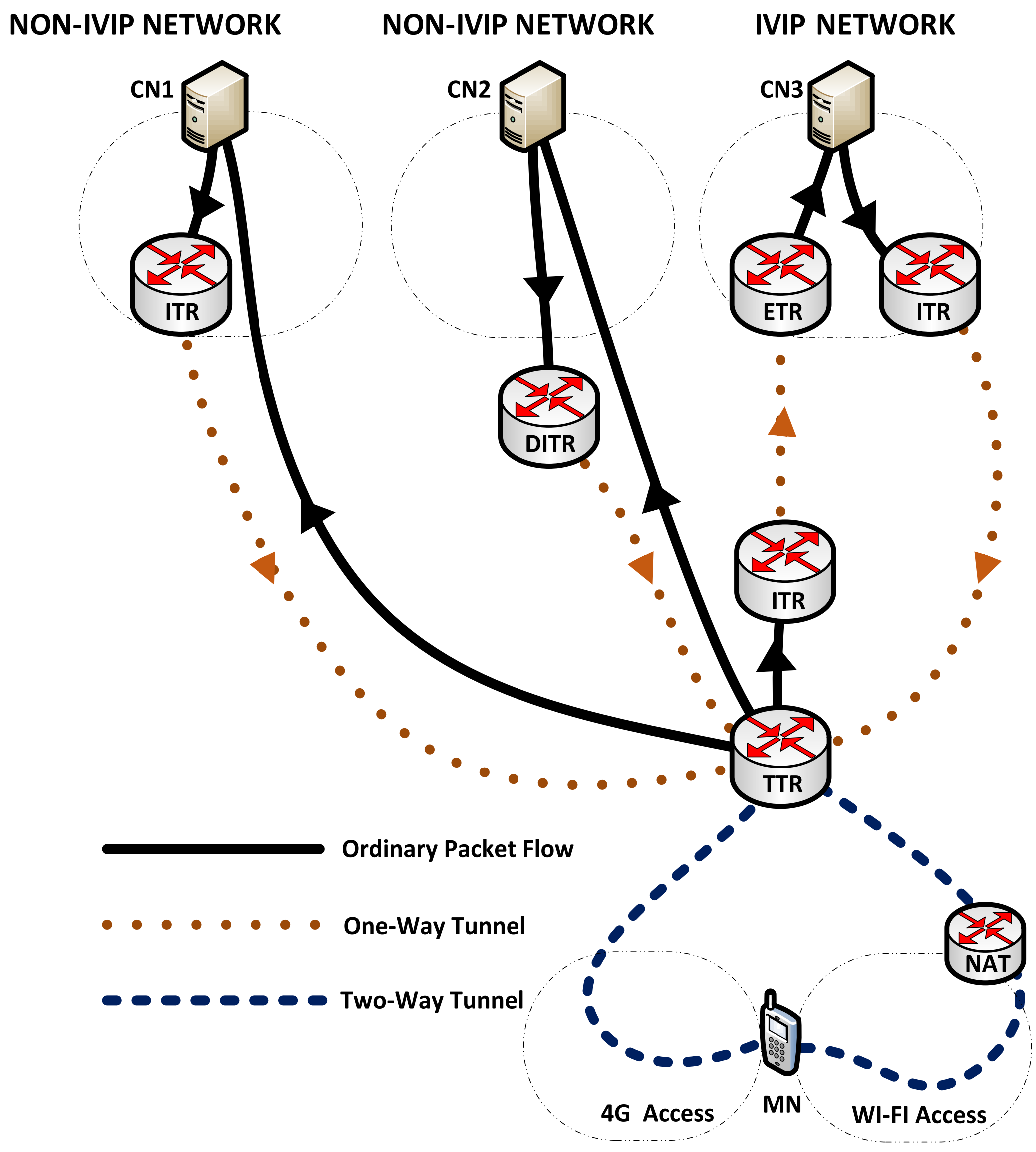


Figure 2.4-1 Packet Flow in IVIP TTR Mobility [73]

are free to implement their software for the tunnel set up. Nevertheless, we assume that at least two messages are necessary to set up a tunnel, the first message will be a binding request/update from the MN and the second will be an acknowledgement from the router. With IVIP mobility, all messages to and from an MN are sent via the TTR as shown in Figure 2.4-1. This also applies to situations where an MN is behind one or more layers of NAT.

As we can see from Figure 2.4-1, an MN tunnels all its packets to the TTR which forwards based on the destination address in the inner packet header. For Packets to CN1 and CN2, the TTR forwards directly (no tunnels) as the CNs have no MAB addresses. Replies from CN1 are sent via the ITR in its network via a one-way tunnel, while CN2 needs to utilise a DITR to send packets to the MN. Although CN1 does not reside in a micronet, there is nonetheless an ITR deployed in its network that enables the CN to communicate with IVIP hosts. CN2, on the other hand, resides in a legacy network and relies on a DITR to reach any IVIP network. CN3 is based in a micronet and receives and sends packets via an ETR and ITR respectively. For communicating with a micronet, the TTR needs to use an ITR service or run the ITR functionality as well, in which case it will remove the need to use the ITR to communicate with CN3 by creating a one-way tunnel to the ETR or a two-way if the remote tunnel routers (xTR) are co-located.

2.4.3 Critique of IVIP mobility

A change in TTR causes an update to the mapping system and with IVIP that is largely eliminated since an MN may only change its TTR when it is a 1000km away from the router. This means that a series of inter-domain handovers by an MN will not have any impact on the IVIP mapping infrastructure and thereby keeping the costly mapping updates at a very low rate. Similarly, IVIP allows the TTR service providers to provide the MN with the necessary software to enable it to connect to the TTR for mobility. The software may be embedded in a mobile phone provided by the operator, for instance, or in the SIM (subscriber identity module) card of the operator.

While having TTR providers using propriety software for mobility encourages competition between the different service providers and will lead to efficiency in the implementations, an MN’s control packets may be blocked by some firewalls for the use of unknown protocols. This will not be the case with well-known and standardised protocols such as MIPv6, for instance. It may also make it difficult to develop a framework to support the protocol’s mobility. A typical example could be developing some IEEE 802.21 media independent handover (MIH) event or command services. IVIP mobility is always involved with triangular routing as packets are always sent via the TTR as we see in Figure 2.4-1. In fact, quadrangle routing is necessary for an MN to communicate with a CN based in another IVIP network. As with LISP-MN, IVIP mobility will involve packet loss as the new interface is configured and packets re-routed via the new access link by the TTR.

Lastly, apart from the Internet draft [72] and Whittle et. al. work [73] both from the original author of the protocol, there is no research work, to the best of our knowledge, that tries to improve TTR mobility or handover with IVIP. We actually could not find any implementation of IVIP nor evaluation of its handover performance. A critique of IVIP is presented in [74] focusing mostly on its mapping infrastructure, adoptability and routing operations. The author of the protocol’s Internet draft has provided a lot of information on the protocol’s operation on his website [26].

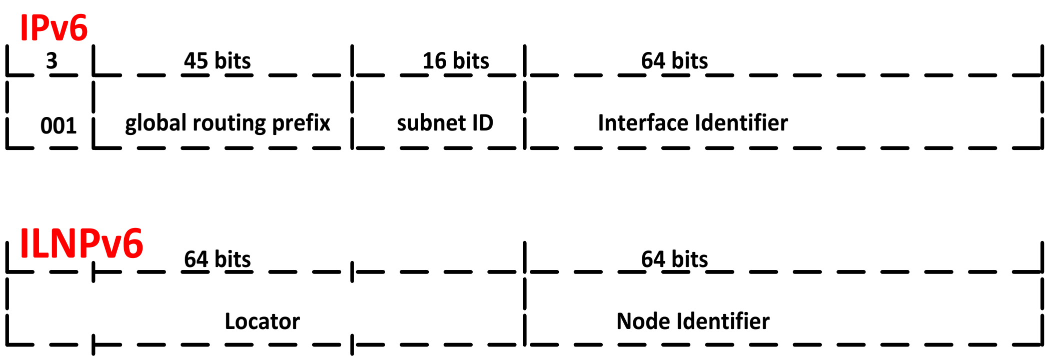


Figure 2.4-2 A Comparison of IPv6 and ILNPv6 Address Formats

2.5 Identifier-Locator Network Protocol (ILNP)

2.5.1 Architectural Concept

ILNP [27] is based on O'Dell's GSE/8+8 [52] concept of routing architecture. The protocol proposes the replacement of the 128 bits IPv6 address with two distinct namespaces of 64 bits each, network locator (L64) and node identifier (NID). The L64 is analogous to the address prefix for routing in IPv6 and serves as the name of a single IP sub-network and not any specific host on the network. The NID (uniqueness can be globally/locally scoped) could be derived from MAC address of the MN's interface and have the same syntax as an IPv6 Identifier, i.e., IEEE Extended Unique Identifier (EUI) 64 address [75].

A pair of I and L values called an identifier-locator vector (or I-LV), which could also be represented as ILNPv6, is needed for communication. This is different from the IPv6 address as shown in Figure 2.4-2; both protocols use the high-order 64 bits for routing purposes (although in different ways), but the low-order 64 bits identifies a node with ILNPv6 and an interface with IPv6. Hence, source/destination I-LV replaces the source/destination address used in IPv6.

Layer	**IPv6**	**ILNPv6**
Application	FQDN/IP Addresses	FQDN, Application specific
Transport	IP address	NID
Network	IP address	L64
Physical Interface	IP address	Dynamic Binding

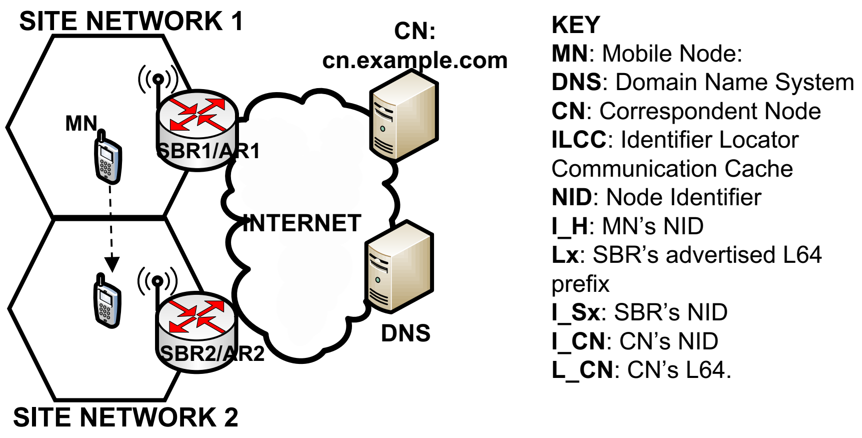
Table 2.5-1 Usage comparison between the IPv6 and ILNP on TCP/IP protocol stack

An MN may have and use more than one NID and/or L64 at a time, but any transport layer session must maintain a single NID throughout the lifetime of the session. All layers above the network will only use the NID or the fully qualified domain name (FQDN), as shown in Table 2.5-1, typically in forming transport and application layer sessions. Routable names, such as the MAC address, X.25 address, or Frame Relay DLCI, on the other hand, are not bound to any interface or sub-network point of attachment. ILNP uses DNS as the rendezvous server to provide the mapping of L64 to NIDs although some resource records need to be added to the DNS to supplement the IPv6's records in the DNS – A, AAAA and PTR records. These required resource records include L, I, PTRL and PTRI.

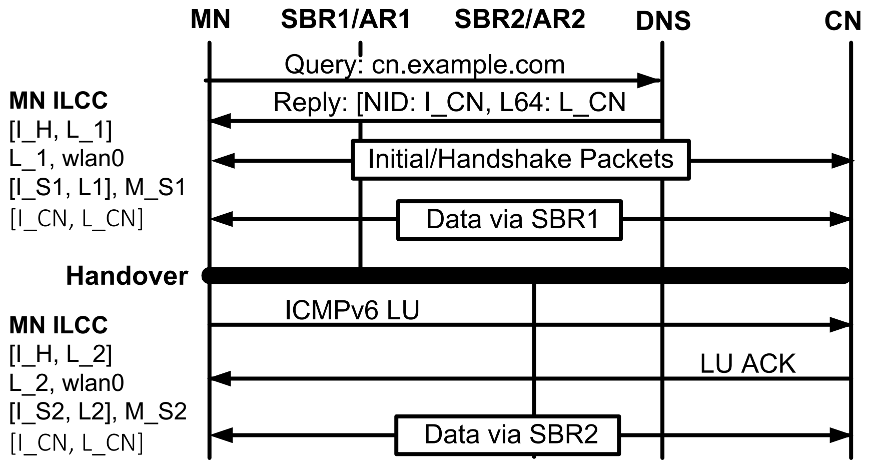
2.5.2 Mobility with ILNP

To enable communication, an MN uses the information contained in its I-L communication cache (ILCC) [76]. This consists of valid IDs (NIDs) and L64s, the bindings between the two addresses, the binding between the L64(s) and interface(s), as well as the ID and MAC address of the site border router (SBR or AR for non-ILNP domain). This information is shown in the expressions E1 to E3 below.

[I_H, L_1] E1
L_1, wlan0............... E2
[I_S1, L1], M_S1....... E3



(A)



(B)

Figure 2.5-1 Architecture (A) and Timeline Diagram (B) for MN's Mobility and ILNP Operations

E1 denotes a binding between the host (MN) ID and LOC, which is physically equivalent to an IPv6 address or the ILNPv6/I-LV discussed earlier. E2 binds the LOC to an interface (wlan0) of the MN and E3 shows the IL-V of the SBR (this will be IP address of the AR for MN in a non-ILNP domain) binding to the router's MAC address. Once these parameters are configured in the MN's ILCC, the device is now ready to establish a communication session. If the MN is to provide a service online, then it uses secure dynamic updates to update these records on the DNS.

As shown in Figure 2.5-1, for communication with the CN, the MN queries the DNS for the I-LV of the FQDN, cn.example.com, and the I_CN and L_CN are returned in the process, which the MN stores in its ILCC as [I_CN, L_CN]. The upper layer protocols use the I_CN to form the transport layer session state. This will be formed of a four-tuple consisting of MN's NID, CN's NID, an ephemeral port number on the MN and a service port number on the CN – represented as I_H, I_CN, P_H, P_CN. The packet is then forwarded as a normal IPv6 packet using [I_CN, L_CN], the CN's I-LV, as the destination and the MN's I-LV as the source. All ILNP communications begin with ILNPv6 nonce destination option included in the initial/handshake packets (IHP) of an ILNP session to indicate that the sending node is ILNP capable. A response to the sent packets indicates that the responder is also ILNP capable, otherwise, it responds with an ICMP error message.

If the MN is within a site network (ILNP's network), its L_1 will be a local LOC, and the SBR applies locator rewriting to change the local LOC value to a global LOC before packets are routed to the outside world. This technique of locator rewriting is effectively network address translation (NAT) without losing the end-to-end transparency between communicating nodes.

In ILNP parlance, this is referred to as localized addressing [77]. If it is a non ILNP network, the ARs will see the packet as a normal IPv6 packet and forward accordingly. When the MN moves to a different SBR/AR, it forms a new LOC,

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Num of Locs  |   Operation   |           RESERVED            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                          Locator [1]                          /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Preference [1]         |          Lifetime [1]         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                          Locator [2]                          /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Preference [2]         |          Lifetime [2]         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               .                               |
|                               .                               |
|                               .                               |
```

Figure 2.5-2 ICMP Locator Update Packet for ILNP. Image Source [78]

[I_H, L_2]................. E4
L_2, wlan0................ E5
[I_S2, L2], M_S2......... E6
[I_CN, L_CN].............E7

L_2, based on the prefix advertised by the new SBR/AR. It then updates its CN using a BU message called the ICMPv6 locator update (LU, see Figure 2.5-2) defined in the protocol's specification [78]. The LU packet carries the new locator value of the MN indicating the prefix on which it could be reached on the Internet. The MN updates its ILCC from expressions E1, E2 and E3 into E4, E5 and E6 together with the CN values added earlier, E7, (see the expressions above). And packets are routed accordingly afterward.

2.5.3 Critique of ILNP

As with other LOC/ID protocols, a handover with ILNP will also cause some packet loss as the MN updates its ILCC when moving to a new network. In theory, this could be mitigated, as the ILNP RFC [27] discusses a handover approach in which the MN keeps both the LOC of the previous network while it is still receiving routing adverts from it, and the LOC of the new network it just connected to. It will then send an LU to the CN(s) with the new LOC to move the running sessions to its new link before disconnecting from the previous link. This technique, and the fact that DAD is not required in ILNP thereby reducing handover delay will surely ameliorate packet loss that is inherent with handovers.

Another scenario that could cause significant packet loss in ILNP network is the race condition where the MN is sufficiently mobile and faster than the DNS updates and the time it takes for changes to be propagated by the DNS fetch cycle. Packet loses could also be recorded in the event of simultaneous mobility as the MN contacts and waits for the DNS to provide the new location of the CN. As a new protocol in its implementation stage, there are two major works that are both geared towards the protocol's implementation. These implementations are, to the best of our knowledge, yet to be evaluated by third-party researchers and the briefs presented below are based on the claims made by their authors.

The first work [79], mainly focused on multi-homing, is an implementation of ILNPv6 on FreeBSD achieved by re-using the IPv6 code. It shows how a host with more than one L64 can load balance between the different locators, which is made possible by a simple per-packet volume (load) balancing algorithm. The second ILNPv6 implementation work [80] is based on a Linux kernel by extending the current IPv6 code to support host mobility. Some level of packet loss was recorded with ILNPv6's hard handoff but no packet loss recorded with a soft handoff scenario. The loss is recorded when an MN moves to a new network during communication while packets from the CN are in transit.

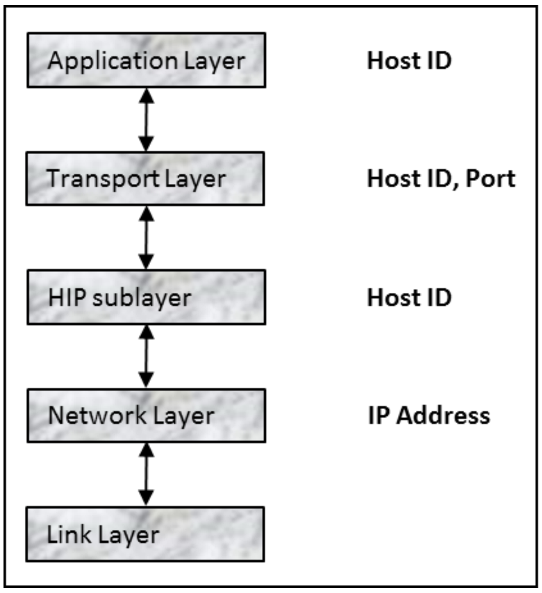


Figure 2.5-3 Location of the HIP sublayer in the TCP/IP Protocol Stack [81]

With soft handover, however, the packets are delivered to the MN via the old AR while it is receiving packets using the new AR. The protocol is shown in [80] to have better all-round handover performance in terms of throughput, packet loss, and handover delay, in comparison to MIPv6.

2.6 Host Identity Protocol (HIP)

2.6.1 Architectural Concept

HIP [28, 81] is a host-based LOC/ID split mechanism that introduces a host identity namespace, for naming IP nodes with a statistically globally unique host identifier (HI). HI is a public cryptographic key from a public-private key pair represented by a 128-bit one-way hash of itself called a host identity tag (HIT). HIT is in the form of an IPv6 address and held by a new layer, the HIP sublayer, in between the network and the transport layers of the TCP/IP protocol stack as shown in Figure 2.5-3. It identifies the device and is used by the transport and upper layers of the stack for communication sessions. It could also be used by

IPv6 address APIs without any modification. HIP also defines a locally unique 32-bit local scope identifier (LSI), which could be used for IPv4 APIs and by the transport and upper layers of the stack. The IP address at the network layer, the LOC, is used for routing purposes and the new HIP sublayer maintains the local mapping of IDs to the LOCs in use for a particular communication session. The presence of HIT and LSI in HIP enables the integration of IPv4 and IPv6 interoperability in the protocol.

The mapping between the HITs and IP addresses on the Internet is provided using an RVS, such as a DNS, although by default HIP uses its own RVS to ensure better performance. The mapping between the FQDN and IP address is configured in a way that the DNS returns, to a requesting application, the HIT (or an LSI) of the target host instead of an IP address. This could be implemented on the DNS by storing HITs (and other information) into the new HIP resource record that allows both HIP and non-HIP hosts to establish a connection with the target host.

HIP's control plane consists of the base exchange (BEX: four-way handshake) session-initiation messages between the two communicating nodes[24], status updates at the point of changing LOCs, and CLOSE & CLOSE_ACK messages for terminating a session. The data plane, on the other hand, is initiated with the establishment of IPSec encapsulated security payload (ESP) security association [82]. The symmetric keys derived during the control plane exchanges protect the ESP, and the security associations are used to transport data securely between the two nodes.

[24] RVS or relay server could be used as intermediaries during the BEX between the two remote nodes.

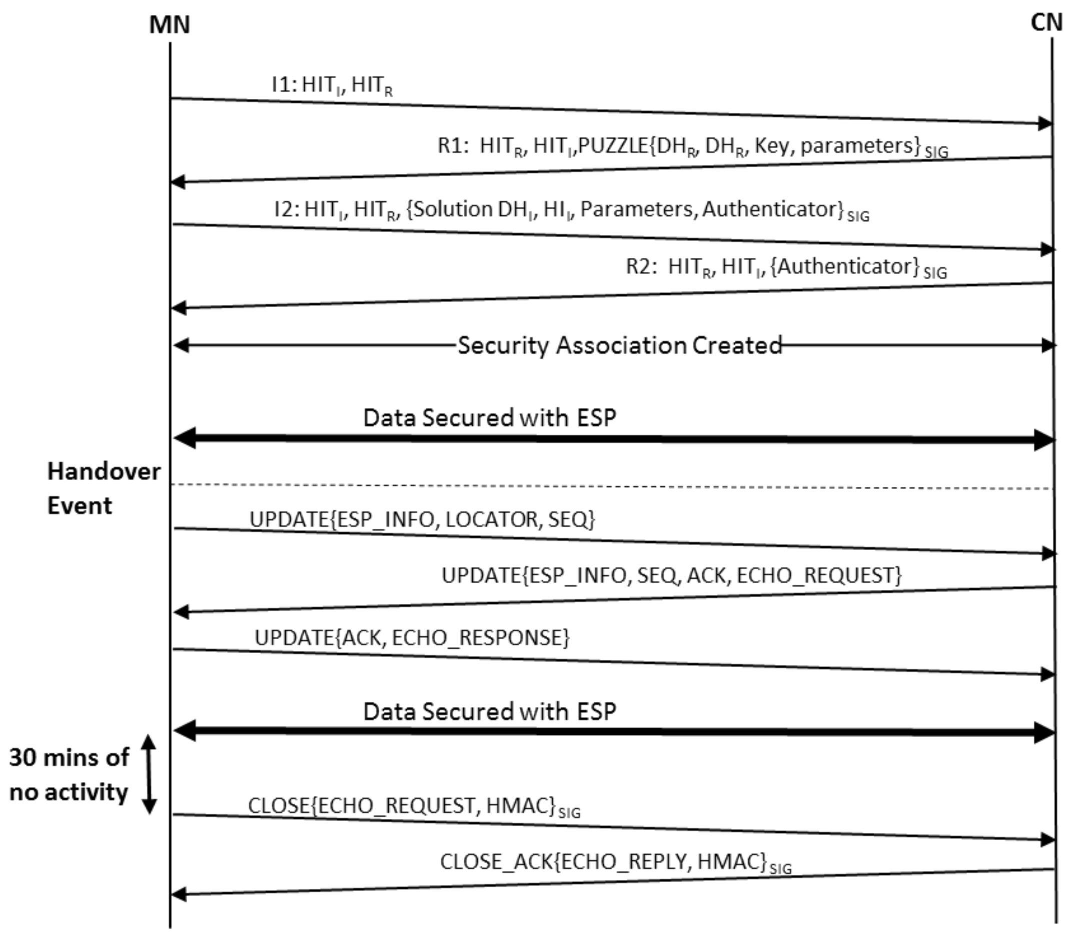

MN
CN
I1: HIT_I, HIT_R
R1: HIT_R, HIT_I,PUZZLE{DH_R, DH_R, Key, parameters}$_{SIG}$
I2: HIT_I, HIT_R, {Solution DH_I, HI_I, Parameters, Authenticator}$_{SIG}$
R2: HIT_R, HIT_I, {Authenticator}$_{SIG}$
Security Association Created
Data Secured with ESP
Handover Event
UPDATE{ESP_INFO, LOCATOR, SEQ}
UPDATE{ESP_INFO, SEQ, ACK, ECHO_REQUEST}
UPDATE{ACK, ECHO_RESPONSE}
Data Secured with ESP
30 mins of no activity
CLOSE{ECHO_REQUEST, HMAC}$_{SIG}$
CLOSE_ACK{ECHO_REPLY, HMAC}$_{SIG}$

Figure 2.6-1 HIP Base Exchange [81]

2.6.2 Mobility with HIP

For an MN to communicate with a remote CN, it queries the DNS for the HIT of the CN and the DNS will either reply with the requested HIT or the IP address of the RVS serving the CN from where the MN can get the HIT. The BEX between the two nodes is initiated by the MN as a means of authenticating the CN and to allow for setting the IPSec ESP SA later.

The messages are carried in a new HIP extension header as depicted in Figure 2.6-1 with the letter I for initiator (the MN) and R for responder (the CN), as well as a sequence number of the messages. The MN will send a trigger message I1 containing the MN's and CN's HITs, and the CN responds with the R1 message. R1 contains a cryptographic puzzle, CN’s Diffie-Hellman (DH) public key to allow the MN to compute the session key, and the CN’s public HI key. The MN solves the puzzle and sends the solution back in the I2 message, which also contains the MN’s public DH key, its public HI key, and an authenticator to show that the MN recently constructs the I2 message.

Once the solution and the authenticator have been verified by the CN, it knows that there is a host that has access to the private key corresponding to the MN’s HI public key that wants to establish a secure communication channel to it. The CN then computes an authenticator and sends it as the R2 packet to the MN. The cryptographic keys generated based on the public keys are used to establish the IPSec ESP SA for secure data communication. A security parameter index (SPI) is carried in each HIP data packet, which together with the IP address are used to locate the ESP SA by the CN and enables the receiving host to decrypt the incoming packets.

When the MN moves to a new AN, it sends a HIP UPDATE message to all its CNs and waits for acknowledgements from the CNs. The update packet contains the Locator parameter which holds the new IP address and the lifetime of the Locator. It also contains an ESP_INFO parameter containing the values of the old and new SPI for security association, as well as a sequence number [83]. The MN may also update the DNS/RVS with its new IP address for proper mapping of its current location if it expects incoming connections from the Internet or in situations where the RVS is used as the intermediary. Initial registration with an RVS is achieved using the BEX messages explained earlier together with an additional REG_REQUEST parameter in the I2 packet. Then the REG_REQUEST parameter is also included in the HIP UPDATE message for subsequent location updates [24].

The CN acknowledges receipt upon validating the packet and afterward updates its local bindings as relates to the MN. The CN will also verify the new address by placing a nonce in the ECHO_REQUEST parameter of the UPDATE message it is sending back to the MN. The MN processes the message and re-echoes the nonce in the ECHO_RESPONSE message to the CN, after which the CN uses the new address for communication.

HIP sessions are terminated using a CLOSE packet typically after 30 mins of inactivity and could be initiated by either of the peer nodes. The packets contain ECHO_REQUEST, HMAC, and HIP_SIGNATURE parameters. The other host replies with CLOSE_ACK containing ECHO_REPLY with the data sent in ECHO_REQUEST to terminate the session. The MN and the CN must verify both HMAC and HIP SIGNATURE in CLOSE and CLOSE ACK packets. [84].

2.6.3 HIP NAT Traversal

Stiemerling discusses the problem with HIP NAT traversal in [85]. The use of HIP extension header, to carry BEX and other control packets, affects HIP operations from behind a NAT as the middle-box could translate or multiplex the IP addresses in the packet, which would affect the integrity of the packets. And since HIP BEX packets do not contain any IP payload, they are likely to be blocked by NAT devices, which only forward IP payloads from well-known transport protocols. The author suggests some HIP extensions where UDP is used to carry HIP packets on a defined port number. The MNs inform the remote hosts of the NAT public IP address, and an RVS is used by MNs behind a NAT to register their IP address and the port that could be used to contact them. HIP data packets may pass through the NAT without any problem although a similar mechanism described for the control plane could also be used for the traversal depending on the type of NAT in use.

2.6.4 Critique of HIP

As stated in [81], HIP attempts to restore the four original characteristics of IP address in an enhanced form. These include non-mutability: the source and destination identities sent are the identities received, and omnisciency: each host knows what identities a peer host can use to send packets to it. This it achieves using the cryptographic keys and the different security measures employed. While these processes no doubt ensure secure communication, it brings with it a large amount of computation especially in a case where a CN, for instance, a web server, continues to process incoming BEX, HIP updates, and termination messages. This may delay a handover update message from an MN being processed in good time for communication to resume.

HIP is designed to be a host-based protocol with no specification for gateways or edge routers as the xTR of LISP & IVIP, or the SBR of ILNP. But as a protocol that utilises IPSec for the data plane, HIP packets could easily be handled by ARs and gateways and should be permitted in routers' default operation modes as recommended in [86]. There are several proposals for improvements to HIP's operation, by moving some signaling to the routers or introducing a local RVS.

Novaczki et al. [87] proposed the use of a local RVS (LRVS) as a domain's gateway (or NAT), which handles an MN's signaling for inter and intra-domain mobility scenarios. An MN learns of the LRVS from a modified ICMPv6 router advertisement (RAdv) messages received from an AR within the domain. The MN registers with the LRVS and thereafter sends a registration message to the RVS. The packets are intercepted by the LRVS, and the IP address is changed to the LRVS' globally routable address. The LRVS, therefore, registers its IP address against the MN's HIT on the RVS. When the MN moves to a new AR within a domain served by its LRVS, it updates the LRVS but not the RVS or the CN thereby reducing the amount of signaling traffic sent on the Internet. If an MN realises that it is moving to a different LRVS upon receiving the RAdv messages, it performs the same registration as described earlier and both the RVS and the CN will be informed of the MN's new location, the new LRVS.

Salmela et al. [88] study the possibilities of providing HIP services to legacy hosts via a HIP proxy. The proxy node is responsible for HIP signaling on behalf of fixed-nodes that it is serving, and establishes the SAs on their behalf. It encapsulates packets from non-HIP nodes to HIP-enabled devices and decapsulates the responses accordingly. A HIP proxy is designed to enable the use of HIP-enabled devices side-by-side with devices on the legacy Internet.

Muslam et al. [89] proposed a network-based mobility management technique with the ARs serving as HIP proxies, where no additional software is needed on the MNs. The architecture of the new system consists of the following: the DNS, which holds the mapping of the MN's HIT against the IP address of its RVS; the RVS, which maps the MN's HIT with the IP address of its LRVS; the LRVS, which maps the MN's HIT with the IP of its AR (proxy); and the AR, which maps the HIT to the MN's IP address. For non-HIP MNs, the proxy router is responsible for registration and other control messages on behalf of an MN.

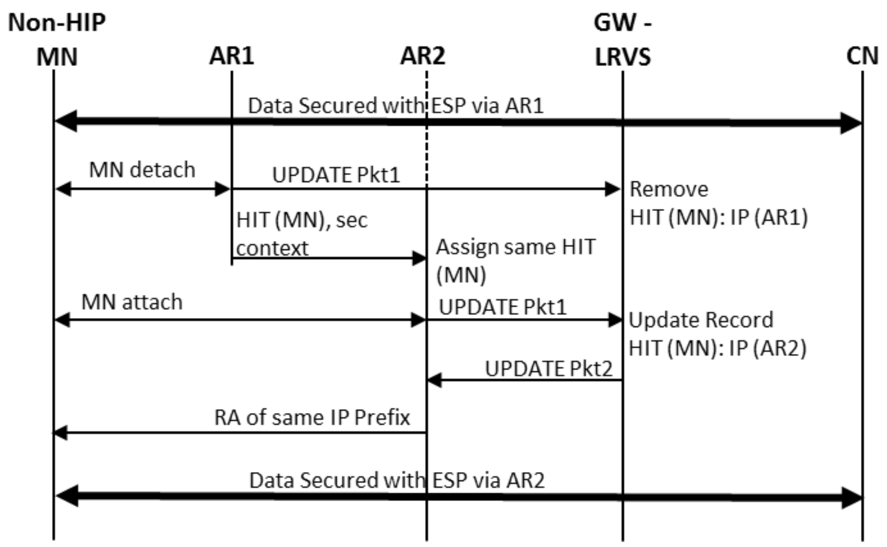


Figure 2.6-2 Network-Based Mobility Management with HIP

As shown in Figure 2.6-2, when an MN moves from AR1 to AR2 both in the same domain, AR1 sends an update packet to de-register the binding of its IP address to the MN’s HIT. The AR defines a HIT for the MN even if it is not a HIP-enabled node. AR2, on the other hand, updates this binding record in the LRVS after learning of the HIP SA context and the HIT of the MN from AR1. As soon as AR2 receives a reply UPDATE packet from LRVS, it sends an RAdv to the MN which will be the same as the RAdv advertised to the MN by AR1, hence the device will maintain the same IP address. All packets that comes in at the completion of the handover are decapsulated by AR2 and forwarded to the MN.

Although the three solutions highlighted will improve aspects of HIP protocol operation, MN’s will still experience packet losses for inter-domain handover scenarios because of the need to change the LRVS as in Novaczki’s and Muslam’s solutions. Salmela’s HIP-proxy solution mainly focuses on enabling inter-communication between HIP and non-HIP enabled network nodes and not about improving mobility.

2.7 Mobile IPv6

MIPv6 [90] is an IETF mobility protocol, and like other mobility protocols, it is targeted at maintaining communication sessions during an MN’s handover. It proposes the use of a non-mutable IP address, termed the home address (HoA), for end-to-end connectivity. While in the home network, the MN uses the HoA as a regular IP address and all communication and routing of packets are done using the same address. On a foreign network, the MN acquires/configures a new IP address called a care-of address (CoA). The relationship between the two addresses is maintained using a mobility anchor, termed a home agent (HA), at the MN’s home network. Henceforth, the HoA is used in forming the transport (and upper) layer sockets, and the CoA is used for routing.

To register the CoA, the MN sends a binding update (BU) to the HA and the HA authenticates the message and replies with a binding acknowledgement (B_Ack) indicating binding completion. The protocol specification mandates the authentication of the binding messages using security mechanisms such as Internet key exchange or IPSec.

As shown in Figure 2.7-1 below, packets to and from the MN, while on a foreign network, are routed via the home network. On receiving these packets, the HA tunnels them to the MN’s current location with the HA’s IP address as the source and MN’s CoA as the destination. The MN replies to the CN with the help of reverse-tunnelling, by sending the packet back to the HA for onward delivery to the CN. Routing can be optimised when communicating with an MIPv6-enabled

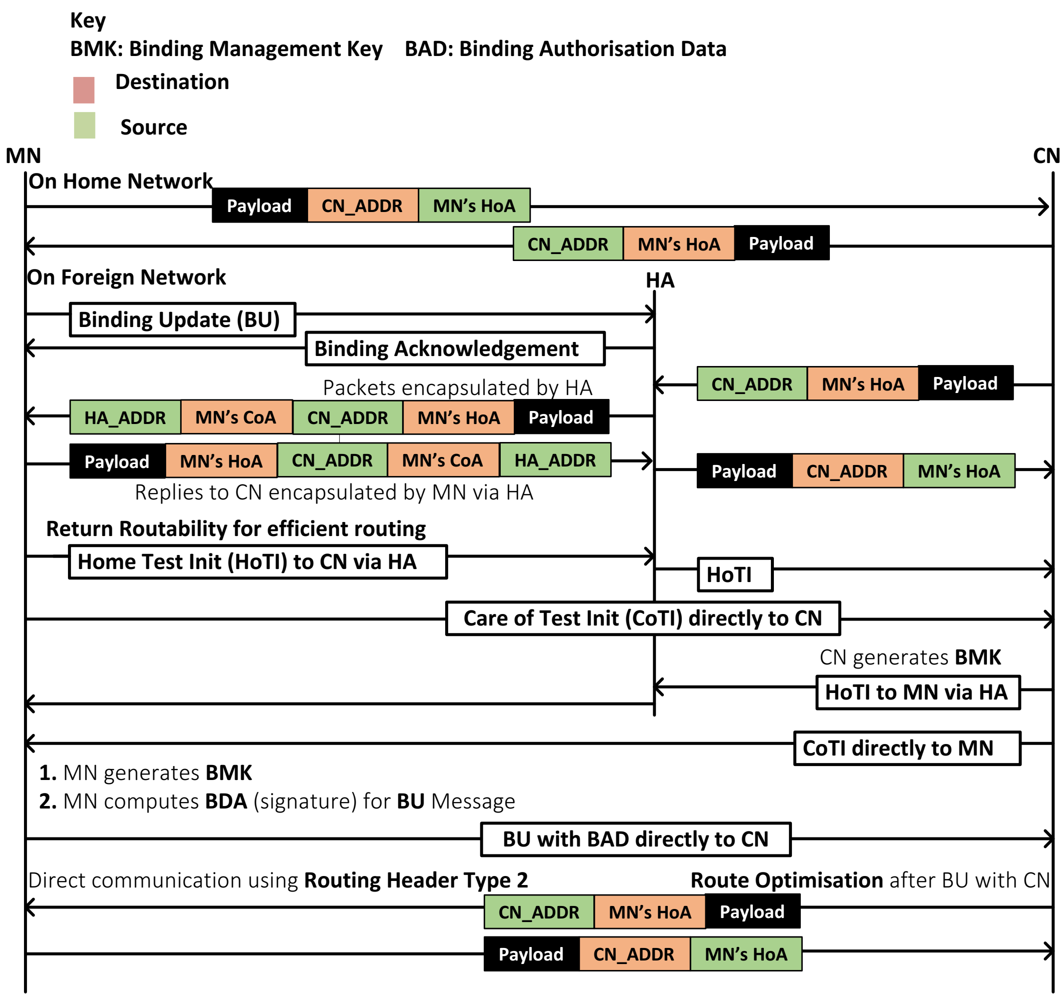

Key
BMK: Binding Management Key BAD: Binding Authorisation Data
🟥 Destination
🟩 Source
MN
CN
On Home Network
Payload | CN_ADDR | MN's HoA
CN_ADDR | MN's HoA | Payload
On Foreign Network
HA
Binding Update (BU)
Binding Acknowledgement
Packets encapsulated by HA
CN_ADDR | MN's HoA | Payload
HA_ADDR | MN's CoA | CN_ADDR | MN's HoA | Payload
Payload | MN's HoA | CN_ADDR | MN's CoA | HA_ADDR
Payload | CN_ADDR | MN's HoA
Replies to CN encapsulated by MN via HA
Return Routability for efficient routing
Home Test Init (HoTI) to CN via HA
HoTI
Care of Test Init (CoTI) directly to CN
CN generates BMK
HoTI to MN via HA
CoTI directly to MN
1. MN generates BMK
2. MN computes BDA (signature) for BU Message
BU with BAD directly to CN
Direct communication using Routing Header Type 2
Route Optimisation after BU with CN
CN_ADDR | MN's HoA | Payload
Payload | CN_ADDR | MN's HoA

Figure 2.7-1 MIPv6 Control and Data Plane Operations with Route Optimisation

CN with the aid of the protocol's route-optimisation (RO) feature as shown in Figure 2.7-1. Once the optimisation process is completed, direct communication can be achieved between the two nodes using the MN's CoA in the packets. And as shown in the figure, a return routability test is necessary to achieve such a level of optimisation.It obvious that MIPv6 shares some characteristics with LOC/ID protocols because it also separates the MN's identity (HoA) to its location (CoA). The major differences with the other protocols are the concept of the home network, with the MN's HA being its only anchor point and the use of a HoA to route packets in the DFZ while on the home network.

2.8 Summary of the Literature

We have highlighted the mobility mechanisms of four different LOC/ID Split mobility protocols – LISP-MN, IVIP, ILNP and HIP – as well as MIPv6. We have seen that all the protocols experience some delay during handover (excluding soft handovers) because, in addition to link layer handover processes, each protocol has a defined signalling message necessary to switch an MN’s traffic to a new link. We have seen the work of Menth et al. [68] that tries to mitigate the three major problems identified with LISP-MN, which include, unnecessary mapping lookups, double encapsulation headers, and path stretch by routing via PITR and PETR. We acknowledged that the introduction of a new node in the architecture, the LMS, will improve the data plane operation of the protocol, but not much regarding handover performance. And, although the new recommendations are likely to improve the protocol, no implementation and testing was presented by the authors.

We have also reviewed the work by Gohar and Koh [69] that proposed a distributed handover scheme for an MN communicating with a CN in the same LISP-domain. While the solution takes away most of the signalling from the MN, the involvement of the routers means that it could only be deployed if routers are LISP-enabled, thereby limiting its deployment in real life. There is also the need to deploy the Media Independent Handover (MIH) ‘link-up’ event to enable the routers to know when an MN is attaching to it. The solution is also silent on MN to map-server control messages and whether these would be taken up by the routers. Although both Menth and Gohar solutions will improve LISP-MN performance, packet loss and service disruption are inevitable during a handover as the CN sends packets to an MN during the handover event.

We actually could not find any implementation of IVIP nor evaluation of its handover performance, although a critique of the protocol is presented in [74] focusing mostly on its mapping infrastructure, adoptability, and routing operations. While we found two implementations of ILNP on UNIX [79] and Linux [80] platforms, and a performance evaluation conducted by the authors, there has been no published research work on this protocol yet from the larger research community.

HIP was originally designed to be a host-based protocol with no specification for gateways or edge routers as LISP's & IVIP's xTR or ILNP's SBR, hence most of its proposed improvements focused on providing network level mobility support. Novaczki [87] and Muslam [89] suggested the introduction of LRVS to move mobility signaling to a gateway, while at the same time enabling NAT traversal for MNs using the new node. Both solutions have shown significant improvement in an MN's handover performance. HIP-Proxy work by Salmela et al. [88], on other the hand, is designed to enable the use of HIP-enabled devices side-by-side with devices on the legacy Internet.

CHAPTER THREE

REQUIREMENT ANALYSIS

Our main task in this thesis is to build a network architecture which supports a LOC/ID split protocol's use in mobile environments to mitigate the packet loss and service disruption experienced with handovers, thereby improving the overall performance of the LOC/ID split protocols. As earlier established, the handover process of LOC/ID protocols is similar to that of MIPv6 and only differ at the point of updating the mappings held by mobility anchors. For this reason, we present the handover performance of MIPv6 to guide us towards drawing the requirements that any system geared towards improving LOC/ID split protocols' handover process needs to satisfy. We start by analysing the different mobility scenarios that these protocols enable and where improvements are needed.

3.1 Mobility Scenarios

As defined earlier, a domain is an independent wireless network in which connecting to its link requires IP level mobility. There are several mobility scenarios that could take place when an MN moves from one domain to another depending on the wireless technologies involved in the handover. We defined horizontal handover earlier as the type of mobility that takes place in HoWE where both the previous and target networks are of the same wireless technology with one interface on the MN involved in the mobility process. If the two technologies are different, hence the MN's two wireless interfaces are involved, it is termed vertical handover. This type of mobility takes place in a HeWE as discussed in chapter one. We illustrated the difference between the two handover types using Figure 3.1-1 below.

The three common mobility scenarios for horizontal handover are Wi-Fi to Wi-Fi; cellular to cellular (or other small cells, such as pico, micro and femto cells); and WiMAX to WiMAX. For vertical handover, on the other hand, it is Wi-Fi to cellular, Wi-Fi to small cells, Wi-Fi to WiMAX, and vice-versa for the three scenarios.

3.1.1 Horizontal Handover Scenarios

A typical example of Wi-Fi to Wi-Fi inter-domain mobility could be found in places with a range of independent Wi-Fi APs to which a pre-configured MN

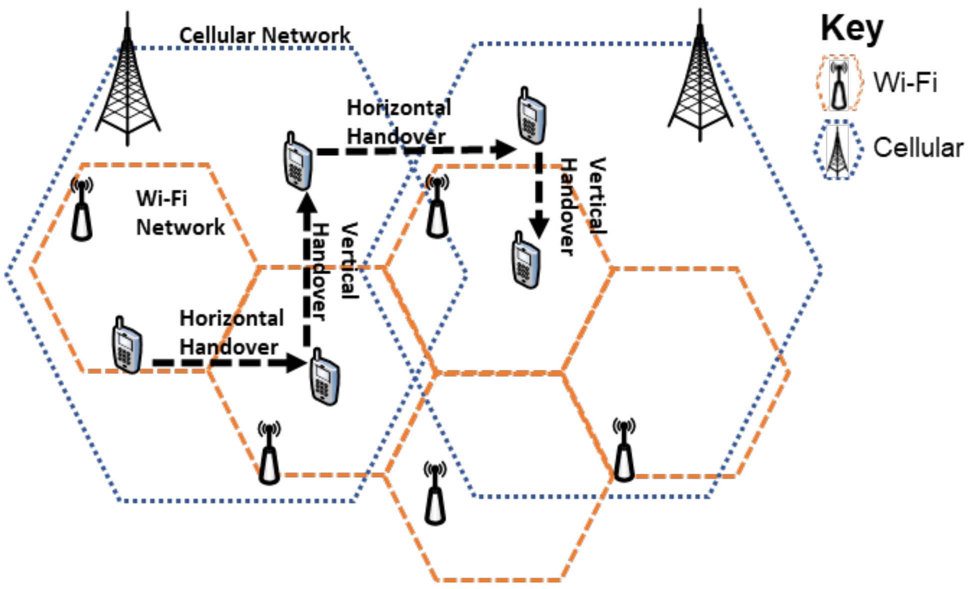


Figure 3.1-1 An Illustration of Horizontal and Vertical Handovers between Wi-Fi and Cellular Access

connects to. This could be found in stations, shopping malls and airports, where a user device could roam from one access network to another and the device configures a new IP address every time it changes its connectivity point. There is also a movement for 'Open Wireless' which pursues the provision of ubiquitous open Wi-Fi Internet in any urban environment with no restriction. This goal is achieved by building technologies that encourage Internet users to securely provide a portion of their Internet connectivity for guests [91, 92]. Such open Internet access allows a user access to several independent networks within a particular neighbourhood or all-around a city. In fact, cities such as New York, Barcelona, Bangalore, Paris, and Copenhagen are already providing free Wi-Fi in some (or most) areas of the city [93].

Another inter-domain mobility scenario is moving from a macro cell of an operator to a femto cell of the same operator. Because femtocells are operated over home broadband, the operator may decide to impose some strict security policies, and that may include using different subnets for the femtocell connectivity which could force the MN to change its IP address upon reconnection.

For a horizontal handover scenario in real life, we can imagine a user with a Wi-Fi capable device accessing a video stream or taking part in a VoIP call, moving across an area where his device has pre-configured access to the available ANs – in cities with free Wi-Fi or Open Wi-Fi areas. As the MN moves from one AN to another in the different wireless environments highlighted above, a new IP address is configured and mobility anchors get updated, which will cause the MN to halt communication for some time while the mobility management process takes place. We highlighted earlier in section 1.2.1 that this type of handover consists of three main stages, MD, DAD, and mapping update.

MD occurs when an MN realizes the presence of another wireless link and decides to connect to the link. This could also happen when the handover is triggered by other means such a media independent handover (MIH) event [37] or due to prompting from an application on, or the configuration of, the device. The data link layer informs the network layer of the presence of a new connection and the network layer sends a router solicitation (RSlt) message to the new AR, as shown in Figure 3.1-2 below. The AR replies with a RAdv providing the MN with the necessary information to connect to the link. The target AR responds to the MN after necessary security checks have been carried out as part of L2 handover procedure and after the MN is cleared to connect by the AR. As soon as the MN learns of its target network from the received RAdv, it configures its LOC using Stateless IP address auto-configuration or begins a DHCPv6 address configuration process.

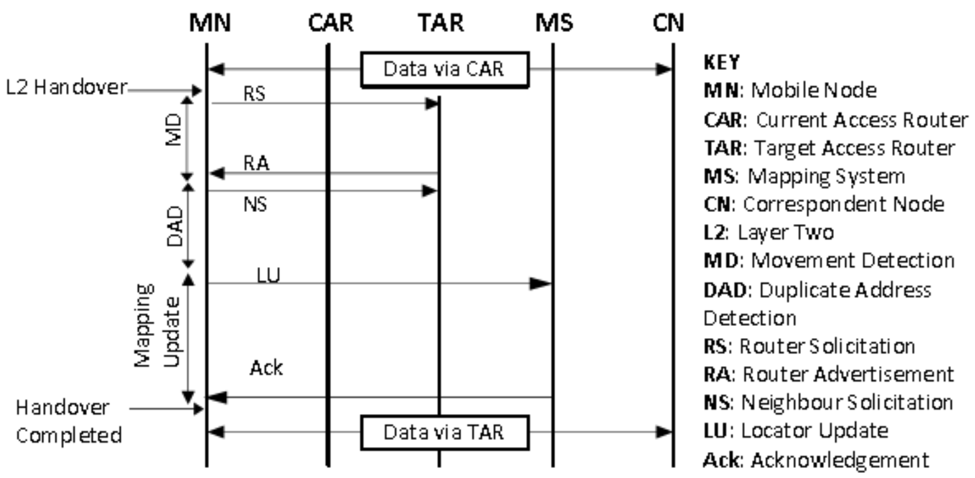


Figure 3.1-2 Three Stages of Handover

The MN will also perform the DAD process by sending a neighbour solicitation message on the link to ensure that no other node has a similar IP address to the one configured. If no neighbour advertisement is received with the same address after some time, the MN is sure that the address is available for it to use as its LOC. Each stage in the handover introduces a certain duration of delay, which culminates and adversely affects the performance of applications running on the device as shown in Figure 3.1-3 below. The figure depicts an average of 10 experimental runs of an MIPv6-enabled MN showing the impact of the handover with a sharp drop in throughput at around the 30-second mark. As will be discussed in detail in chapter six[25], we set up a laboratory testbed with an MN running the MIPv6 protocol as part of a preliminary investigation aimed at understanding the impact of handovers on mobile devices' performance.

[25] Chapter six presents a detail description of the testbed highlighted here, which is used for all the experiments in this research.

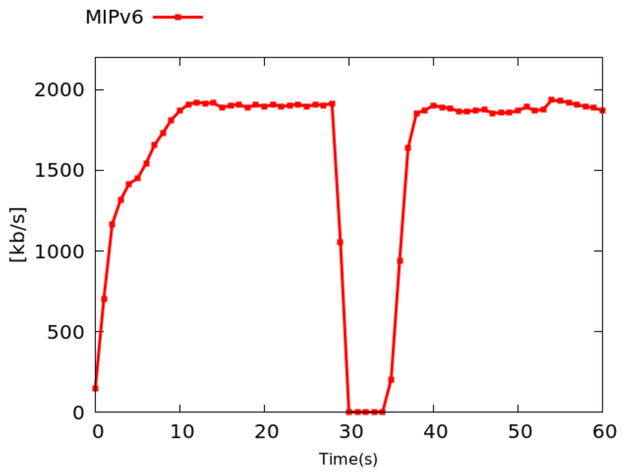


Figure 3.1-3 Inter-Domain Horizontal Handover TCP Throughput

The MN downloaded a file using TCP for a period of one minute, and at some point during the download, it executed a horizontal handover. Figure 3.1-3 also shows a 5-second total break in transmission as the handover was executed. This duration is called *service disruption time* (SDT), defined here as the measured difference between the time when the last packet is received before a handover and when the first packet is received after the device's reconnection. SDT is directly proportional to handover delay – the longer the delay, the longer the SDT.

Similarly, the MN, using the Linux iperf program to emulate voice traffic, conducted a handover during the voice call with a remote client. The handover was executed some 30s into the communication session. The event caused about 360 datagrams loss (6.5%) of the total sent (5,530 datagrams) in one minute.

3.1.2 Vertical Handover Scenarios

Vertical handovers are more often than not inter-domain in nature as the cellular, Wi-Fi and mobile WiMAX services available in a given geographical location are likely to be independent networks. Even where two different technologies available for a user to connect to are owned by the same provider, the two networks could be on different subnets for the purpose of security and billing of subscribers. The Wi-Fi to cellular/WiMAX[26] scenario and vice versa are the most common vertical handover scenarios executed today as stated earlier.

Vertical handovers happen in HeWEs, and a typical example of a real-life scenario is that of a user moving with a cellular and Wi-Fi capable mobile device. While at home, the user connects to their home Wi-Fi, streaming a video or taking part in a VoIP call. As soon as the user steps out and moves away from the Wi-Fi, the device connects to the cellular network of their provider. Similarly, on returning home, the phone disconnects from the cellular network and connects to the higher priority home Wi-Fi. This type of mobility is also found with subscribers of Wi-Fi hotspots and users of public networks as they come into and leave the hotspot/public network area – disconnecting from cellular on coming into the Wi-Fi area and reconnecting to the cellular after going out of Wi-Fi's range.

Inter-domain Vertical handovers can be viewed as hard or soft handovers as discussed in the subsections below.

[26] Due to similarities between cellular and mobile WiMAX's mode of operation, our focus in this work will be on Wi-Fi and cellular networks only.

3.1.2.1 Hard Handover

This is usually seen as break-before-make handover process in which the current interface will go down before the target interface is configured and used. This could happen where an MN has a Wi-Fi and cellular/WiMAX interface. The Wi-Fi is often, by default, the interface with higher priority on most mobile devices. Hence, the Internet connection only switches to the cellular interface when the Wi-Fi connectivity is lost. Because the MN only brings up the cellular interface and configures it after the loss of Wi-Fi link, there is an abrupt loss of connectivity in a Wi-Fi–to–cellular handover. All the three stages of handover depicted in Figure 3.1-2 earlier are required for a hard handover event and are included in measuring its delay period.

3.1.2.2 Soft Handover

This type of handover usually takes place in a make-before-break fashion with the connectivity to the current link maintained while the other interface is being configured after which the transmission is transferred to the new link. A device accessing the Internet via its cellular link and coming into a pre-configured Wi-Fi area will switch all transmissions to the Wi-Fi link as soon as the interface is configured and ready for use. Because the Wi-Fi interface is configured and made ready for use before the cellular interface is downed, the disruption in communication is usually minimal and often not perceived by the users.

The duration of delay in soft handover does not involve MD and DAD since the two processes are conducted before the interface is considered active by the protocol in use. And depending on the mobility protocol, the mapping updates may either be sent using the current (cellular) link prior to the handover or the target (Wi-Fi) link after the handover. Figure 3.1-4 shows MIPv6's inter-domain

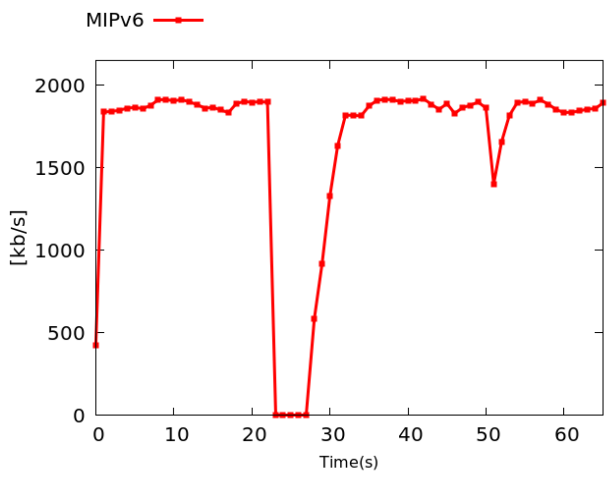


Figure 3.1-4 Inter-Domain Vertical Handover TCP Throughput

mobility for both hard handover (just after 23-second mark) and soft handover (at about 52-second) scenarios as recorded on the testbed settings highlighted earlier. As with the horizontal handover, there was about 5s break in transmission during the hard handover event. This resulted in a sharp drop in throughput of the link in the hard handover scenario going to zero point (on the figure) for the duration .As with handover in HoWE, a similar amount of packets was lost during the movement event as the handover delay was almost the same. The soft handover, on the other hand, recorded only a little dip in the throughput because no break in transmission was recorded for the period, and the little drop in throughput level was because of transferring the communication session to the already configured target interface. This loss is quite insignificant to almost all applications that run on mobile devices.

3.2 Requirements of a New LOC/ID Split System

We have observed from the preceding sections that both horizontal and vertical (hard) handover delays disrupt on-going communication sessions leading to a significant drop in throughput, loss of packets in transit, and an increase in SDT. Hence the major requirements of any network architecture based upon a LOC/ID split mobility protocol is eliminating the negative effect of the performance indicators highlighted above, by mitigating the drop in throughput, reducing the number of packets lost, and reducing the SDT.

3.2.1 Mitigating the Drop in Throughput

It is vital that the drop in throughput level at the point of handover be mitigated to ameliorate the effect that this drop has on the transport protocols and by implication the applications running on a mobile device. As we all know, TCP (cubic) reduces the size of the congestion window by 50% once packets are not acknowledged due to link failure or other reasons. The sender continues to decrease the size of the window if sent packets are not acknowledged by the MN. When the MN finally reconnects via the target access link, it sends for retransmission of the lost packets, and the sender adjusts its window to a slow start before gradually building up to the optimum capacity. That is the reason why, as observed in Figures 3.1-3 and 3.1-4, the throughput picked up slowly after handover before attaining full capacity some few seconds later.

To mitigate this drop in throughput, the followings need to be addressed/introduced:

a) A tunnel needs to be created between the previous and target PoAs for forwarding of packets that are received by the previous device during the handover process. Alternatively,
b) packets sent during the handover process need to be buffered to be forwarded to the MN at the completion of the process. This would be forwarded to the MN's new location at the completion of the handover.

3.2.2 Eliminating/Reducing Packet Loss

When an MN performs a handover amidst a communication session, the sender may continue to send packets depending on the transport protocol in use as explained in the previous section. In the case of TCP, it will stop after sometime when no acknowledgements are received. But with UDP, the sender continues to send throughout the duration of the handover, and many packets will be dropped by the previous PoA since it has no way of knowing the receiver's current location.

To support LOC/ID split protocols' operation in this regard (and with regard to the problem highlighted in subsection section 3.2.1), these packets need to be buffered and forwarded to the MN at the completion of the handover. This will reduce the TCP retransmission requests the MN sends to the sender at the completion of the handover, and in the case of UDP ensures the full receipt of packets in transit.

3.2.3 Reducing Service Disruption Time (SDT)

As mentioned in section 3.1.1, SDT is directly proportional to handover delay and is usually a few seconds longer than the handover duration. Generally, communication resumes after the mapping update is completed when the anchor node can forward packets to the MN’s new location. Longer disruption in a communication session has an adverse effect on the quality of service (QoS) experienced by the user – such as scrambled audio in VoIP calls; re-buffering (temporal stoppage of video playback); low video quality; slow downloads, etc. The delay in handover could be reduced by initiating the movement process while the current interface is still active. This could be done by monitoring the signal strength of the link and initiating the handover process when a defined threshold is attained. The handover process may also be initiated when the change in PoA is triggered by an MIH event or a prompting from an application on the device.

3.3 Research Aims and Objectives

As earlier highlighted in chapter one, the main aim of this research work is to introduce an improved Locator/Identifier Split Architecture (ILISA) designed to enhance the mobility of nodes running a LOC/ID protocol by mitigating packet loss and reducing service disruption in handovers. A new network node, Loc-server, is central to the new architecture with the task of buffering incoming packets during handover and forwarding the packets to the MN on the completion of the node’s movement process. Furthermore, the work is aimed at achieving the following:

1. Provide a thorough understanding of the mobility performance of LOC/ID split protocols.
2. Analyse the performance of both TCP and UDP based applications such as video, voice, and file download over the LOC/ID protocols.

3. Design a new architecture geared towards minimising the drop in throughput and loss of packets thereby improving the performance of LOC/ID protocols.
4. Provide support for multiple LOC/ID split protocols.
5. Enhance the discussion on the use of LOC/ID split protocols in future wireless networks.

To achieve the outlined aims, the following objectives will be pursued:

1. Implement a LOC/ID protocol on a laboratory testbed including all the required components for its mobility mechanism. With most of the protocols having a similar mode of operation, a representative protocol will be carefully selected to test LOC/ID performance by looking at the different network parameters, such as throughput, packet loss, handover delay, SDT and quality of service for multimedia applications.
2. Analyse the inter-domain mobility performance of MIPv6 and a LOC/ID protocol on the testbed emulating both homogeneous and heterogeneous wireless network environments. MIPv6 will be implemented on the testbed to support a performance comparison with the LOC/ID protocol and serve as the benchmark for the latter's performance. The analysis will focus on inter-domain mobility in both homogeneous and heterogeneous wireless network environments.
3. Design and implement a buffer node; we call Location Area Server (or simply Loc-server) that will be storing packets for a pre-registered MN and forwarding the packets at the request of the MN. The Loc-server will perform three main tasks:

   a) On receipt of a buffering request from the MN, it authenticates the device and creates a buffer ready to receive packets destined to the MN’s identifier.
   b) It stores the received packets for a predefined duration by setting up a timer and waiting for the MN’s request for the forwarding of any stored packet. Stored packets are destroyed at the expiration of the timer.
   c) On receipt of the forwarding request from the MN, the Loc-server tunnels the packets to the MN’s current location.

4. Implement new signalling mechanisms on the MN to enable the mobile device to interact with the Loc-server. In addition to the LOC/ID protocol’s control messages, build four new signalling mechanisms on the MN to enable the packets’ redirection at the point of handover:

   a. A ‘handover imminent trigger’ when signal strength reaches a defined threshold. This will notify the LOC/ID protocol on the MN of the need to redirect packets to the Loc-server.
   b. A signalling message integrated into the LOC/ID protocol to tell the anchor node where packets for the MN need to be redirected to.
   c. A second signalling message to tell the Loc-server to receive packets on behalf of the MN.
   d. A third signalling message at the completion of a handover to tell the Loc-server to forward any buffered packets to the MN and for the anchor to resume normal operation.

5. Re-evaluate the performance of the LOC/ID protocol with the support of Loc-server to determine how much improvement the new node brings to the protocol in question. The performance evaluation will be similar to the context used in the first and second objectives.

3.4 Chapter Summary

In this chapter, we discussed the different host-based mobility scenarios experienced by mobile devices in today’s wireless environments and how the handover events affect the mobility experience of the users. We showed the impact of both vertical and horizontal handovers on MIPv6 and the resultant drop in throughput, packet loss, and disruption of on-going communication sessions. We have also discussed the requirements that our solution needs to satisfy to improve on the functionality the LOC/ID split protocols provide. These requirements include:

- Mitigating the Drop in Throughput
- Eliminating/Reducing Packet Loss
- Reducing Service Disruption Time

We concluded the chapter by providing a detailed discussion of the aims and objectives of the research work.

CHAPTER FOUR

DESIGN

This chapter provides a detailed design description of our newly developed Improved LOC/ID Split Architecture (ILISA). ILISA is designed in a way that meets the requirements described in section 3.2 of the previous chapter, which includes mitigating the drop in throughput, reducing packet loss and reducing SDT during handover. The architecture is designed to be compatible with the four LOC/ID split protocols under review as well as MIPv6.

ILISA proposes the introduction of a new network node named Loc-server into LOC/ID split-based architectures to improve the performance of this class of mobility protocols. Loc-server is central to this architecture and designed to ensure that no packets are lost at the point of an MN’s handover. This chapter will discuss the components of ILISA and how the Loc-server could be integrated into different LOC/ID split-based network architectures to enhance the mobility performance of MNs. The chapter provides a high-level view of the system design with implementation specifics provided in the subsequent chapter.

4.1 Overview

Introducing Loc-server is key to improving the LOC/ID split protocols’ performance especially at the point of handover. The node serves as a support entity for packet delivery at the point of an MN’s movement to ensure that no packets are lost because of the mobility process. Loc-server can be owned by a Telecom operator, an ISP or be provided by community networks. It is termed Loc-server because it is designed to serve a defined geographical area with a collection of homogeneous or heterogeneous wireless access technologies as shown in Figure 4.1-1 below. The figure also shows in the magnified area (the oval on the top) that the different networks are connected to each other and to the Loc-servers via the same Internet.

The size of a location area can be determined in a number of ways (e.g., by considering the anticipated number of devices the server will manage). ILISA also proposes the introduction of some functional modules on the MNs to enable the devices to interact with the Loc-server, and for the server to operate on the request of the MN. MNs are configured with an anycast Loc-server address to enable the devices to use the server that is closest to their location,

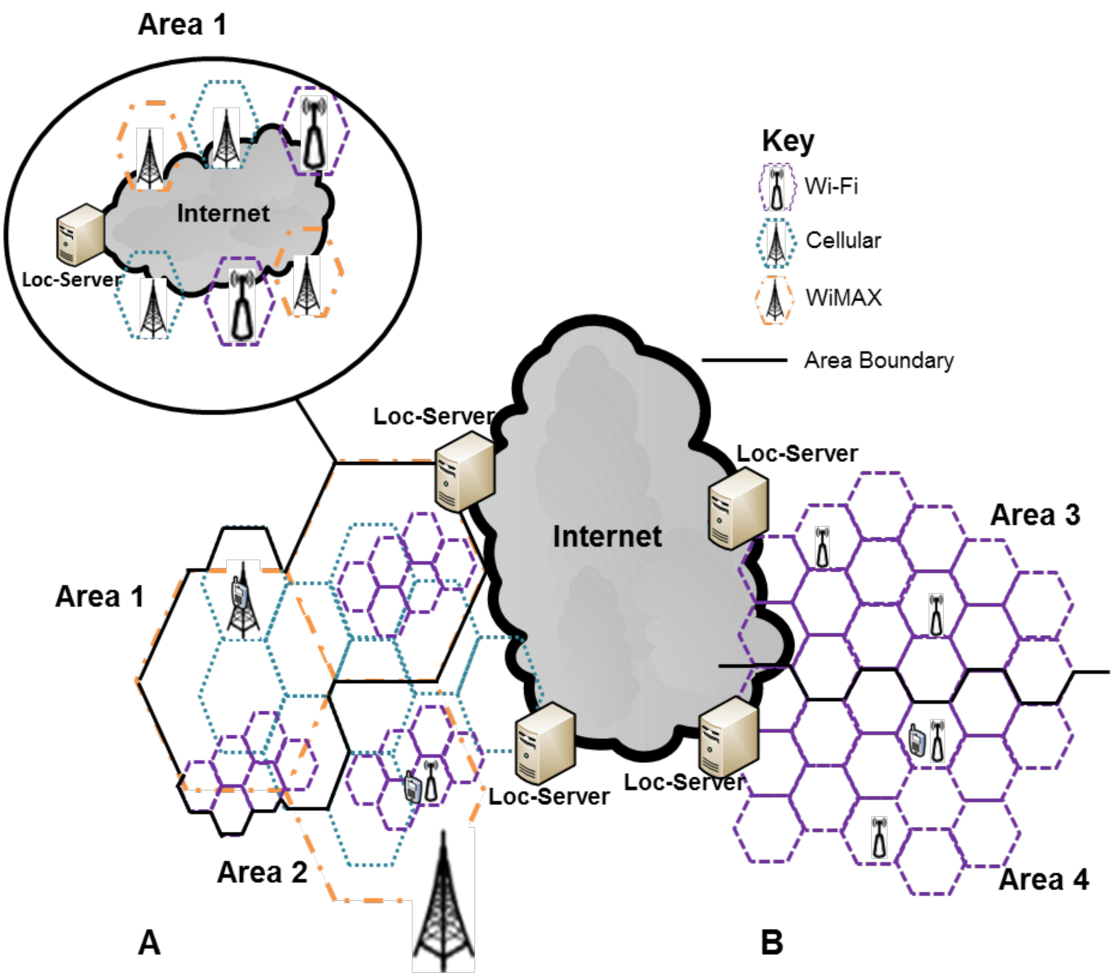


Figure 4.1-1 Location Area Server in both Heterogeneous (A) and Homogeneous (B) Wireless Environments.

which could be any area depending on the PoA that the MN is connected to at the time of handover.

4.2 Operation

When a handover is imminent (more on this in section 4.3.1), the MN sends a request prompting the Loc-server to create a buffer space for its (the MN's) packets. The Loc-server checks to see that the MN's identifier is within the block of IDs allowed and consequently reserves the space and listens for incoming packets destined to that identifier. While sending the buffering request to the

Loc-server, the MN also updates the mobility anchor with the Loc-server's address as its current CoA. This will cause the MN's packets to be redirected to the Loc-server. To update its location at the completion of its handover, the MN provides its new CoA to both the Loc-server and the mobility anchor causing the former to send any buffered packets and the latter to redirect incoming packets to the location. Section 4.5 details the Loc-server operation as integrated with different LOC/ID split protocols.

To enable the communication between the MN and the Loc-server, some modules and functional components need to be introduced as stated earlier. On the MN, there is need for a module to inform the LOC/ID protocol of an imminent handover and some changes to the LOC/ID protocol to enable the protocol to react to this information. The Loc-server, on the other hand, needs a control module, to receive and process the binding/mapping update messages from the MN; a data module, responsible for receiving packets and forwarding to the MN's new location; and a database of prefixes that are permitted to use the Loc-server.

4.3 Mobile Node Design

We highlighted in section 1.1.1 that an MN handles all mobility related signalling in host-based mobility management protocols which the LOC/ID protocols under review fall into. The device informs its mobility anchors or CN of their current location whenever there is a change. For this reason, the MN necessarily needs to know about the existence of a Loc-server to enable communication between the two nodes when necessary. Two modules need to be introduced on the MN to enable this communication: a handover

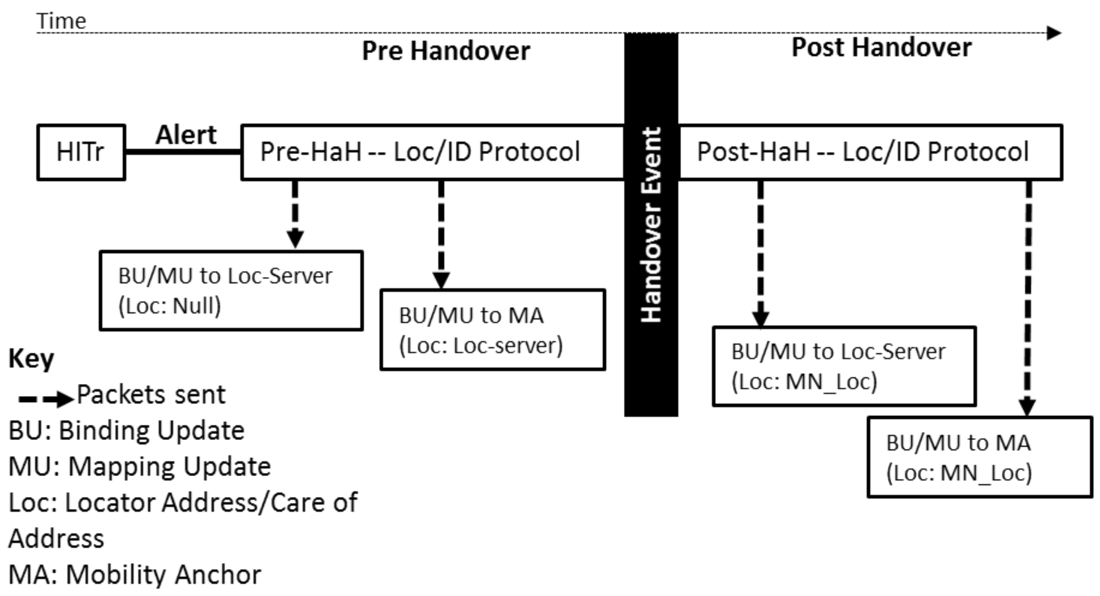

Time
Pre Handover
Post Handover
HITr
Alert
Pre-HaH -- Loc/ID Protocol
Handover Event
Post-HaH -- Loc/ID Protocol
BU/MU to Loc-Server (Loc: Null)
BU/MU to MA (Loc: Loc-server)
BU/MU to Loc-Server (Loc: MN_Loc)
BU/MU to MA (Loc: MN_Loc)
Key
Packets sent
BU: Binding Update
MU: Mapping Update
Loc: Locator Address/Care of Address
MA: Mobility Anchor

Figure 4.3-1 Information and Action Flow between HITr, Pre, and Post Handler

imminent trigger (HITr) and a handover handler (HaH). Figure 4.3-1 shows the interaction between the different modules as discussed in the following subsections.

4.3.1 Handover Imminent Trigger (HITr)

HITr is a module on the MN that tells the LOC/ID protocol that a handover is about to happen and the protocol should prepare for it. The handover could be triggered by different factors depending on the wireless environment involved and/or the configuration of the mobile device. These factors as discussed in [94, 95] include, but are not limited, to (1) traditional (simply received signal strength indicator (RSSI)); (2) user-centric (which network the user of the device wants to connect to); and (3) a cost function-based factor, which considers the monetary cost of using a wireless link, power consumption, data rate, available bandwidth, and other relevant factors.

Complex and advanced algorithms have also been developed to support MNs and network nodes to decide when best to handover and which AN to connect to. These algorithms use a combination of the factors mentioned above in addition to other mechanisms before a handover is triggered and executed [94, 95]. For instance, Lin et. al [96] proposed a context-aware handover decision mechanism that uses context information such as RSSI, signal to noise ratio (SNR) and bit error rate (BER) to calculate packet success rate of a target link as a determinant for triggering a handover. Other researchers [97, 98] proposed the use of fuzzy logic and neural concepts to decide the best network to handover to and when to initiate the handover after considering several factors.

In a HeWE, factors such as cost consideration, RSSI of the access links, or the sudden availability of a wireless link that could best serve the QoS requirements of the running application could be the primary reasons to trigger a handover. Nevertheless, a cost function-based factor is often the default parameter used on most mobile devices in this environment. The often-cheaper Wi-Fi link is prioritised over the more expensive cellular connectivity with most smartphones and tablets automatically switching to – by default – any available pre-configured Wi-Fi on coming into the links' range. For a homogenous environment, the RSSI is the most common factor in determining whether a handover should be triggered or not.

Hence, in designing our solution, we used the two common factors of cost and the RSSI as the handover triggers although other factors discussed above could also have been used. HITr is designed to alert the mobility protocol of an imminent handover less than one second before the process starts. Once an

MN receives the alert, it scans for other wireless links, as it normally would during a regular handover, and connects to a pre-configured AN within its reach.

4.3.1.1 Cost Function-Based Factor

For the purpose of our work, we opted to use this factor as a trigger to handover from a cellular link to a Wi-Fi link in a heterogeneous wireless environment (HeWE). Wi-Fi links are usually cheaper and provide higher bandwidth and data rates, with low battery usage due to the often short distance between MN and the AP providing the Internet connectivity. We utilise a feature available with most LOC/ID protocol implementations, which is the ability to prioritise a given interface on the node to cause a handover whenever the priority interface becomes available during a communication session. This will enable an MN to switch all on-going data sessions to the newly configured interface. Hence an MN using its cellular interface for communication will switch all its on-going data sessions once its Wi-Fi interface connects to any available Wi-Fi network.

4.3.1.2 Received Signal Strength Indicator (RSSI)

The RSSI is the most common factor used by the MNs as a user moves around a HoWE. The MN intermittently scans the area for the available wireless networks and also measures the signal strength of its current link to calculate the threshold against a constant we set. Once that threshold is reached, it sends a signal to our program that a handover is about to happen as shown in Figure 4.3-2 below. The figure highlights the stages involved before an MN informs the running LOC/ID protocol of imminent handover. The MN then connects to the AP with the stronger signal strength after all mobility procedures are completed.

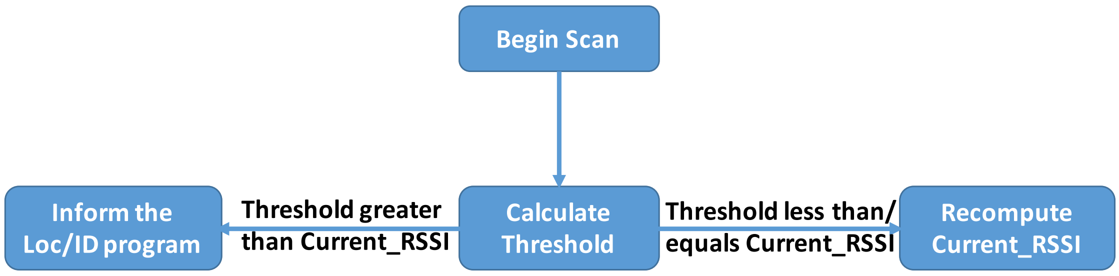

Begin Scan
Inform the Loc/ID program
Threshold greater than Current_RSSI
Calculate Threshold
Threshold less than/ equals Current_RSSI
Recompute Current_RSSI

Figure 4.3-2 Decision Flow on Determining when to Handover

We also used the RSSI to envisage a handover in a HeWE when the signal strength of the priority interface reaches a pre-set threshold. Hence, a handover from a Wi-Fi link to a cellular simply occurs once the MN moves out of the range of the Wi-Fi link. The MN configures the cellular interface and transfers all communication sessions to it.

4.3.2 Handover Handler (HaH)

This is a module to be integrated with a LOC/ID protocol and consists of two components, pre-HaH and post-HaH. The first component handles the RSSI (or any other HITr factor) information received from the HITr module discussed above, and is responsible for involving the Loc-server in the pre-handover procedure, while the second component ensures the involvement of the Loc-server in the post-handover process.

4.3.2.1 Pre-Handover Handler

As soon as the HITr informs the protocol of an imminent handover (see Figure 4.3-1), the pre-HaH causes the LOC/ID protocol to send binding/mapping update message(s) to the MN's mobility anchor and/or CN and then a similar control message to the Loc-server. The pre-HaH module will change the composition of the packets sent to the anchors/CNs when the HITr flag is received with the Loc-server address replacing the MN’s LOC/CoA in the packet. This will cause the anchors/CNs to forward any MN’s packet to the Loc-server from then onwards. The control packet to the Loc-server, as with the one going to the anchors/CNs, contains the ID/HoA of the MN and port numbers where necessary but with a null value in the LOC field of the packet. We termed this control packet, BU-Null. This packet is to cause the Loc-server to establish a secure line of communication with the MN. It also serves as a request to the server to listen to and buffer packets destined to the MN’s ID contained in the packet.

4.3.2.2 Post-Handover Handler

At the completion of the handover, the post-HaH module causes the LOC/ID protocol to yet again send the same binding update message to the Loc-server as the one sent to the mobility anchor. The mobility anchor uses the MN’s LOC in the packet to redirect incoming packets forthwith to the MN’s current location, while the Loc-server will use the address to forward any buffered packet to the MN’s location. We termed this control message BU-MN_LOC.

The relationship between the HITr and HaH was shown earlier in Figure 4.3-1. The figure also depicted the time sequence in which these messages are sent, and the order in which the events happen.

4.3.3 Handover Failure Guard (HFG)

Since there is no algorithm that can predict a handover with 100% accuracy, we introduce a handover failure guard (HFG). When the anchors/CNs are redirecting the MN’s packets to the Loc-server at the point of handover, they are expected to resume normal communication upon BU-MN_LOC receipt after the device has completed the mobility event. In the case that the handover fails to take place because the signal strength of the current link increases or the targeted Wi-Fi link is no more within reach (or for any other reason), the anchors/CNs will continue to send packets to the Loc-server, completely breaking the communication with the MN.

The HFG is designed as a remedy, in such a way that it listens to the wireless interface(s) by setting up a 2-second timer starting from when the BU-Null is sent. We chose two seconds as we expect a handover to happen in less than one second after HITr, and at two seconds, it is safe to assume that the handover did not take place and it is necessary for the MN to resume communication with its remote nodes using its current loc. Hence, when the timer expires and there are no changes on the interface to signify a handover, HFG causes the LOC/ID protocol to send BU-MN_LOC to all the remote devices to ensure the resumption of the data session and the forwarding of any buffered packets. This ensures that even if the handover failed after packets are redirected to the Loc-server, the messages could still be retrieved and communication sessions would resume as normal.

4.4 Loc-server Design

The Loc-server is composed of control and data modules for handling control and data packets respectively and a database of all the prefixes that it is configured to serve.

4.4.1 Control Module (CM)

When an MN sends a BU-Null as described in section 4.3.2.1, the CM on the server receives the packet and checks whether the MN’s ID is within the range of permitted prefixes in its database. If the ID in the BU-Null is not recognised from the available prefixes, the server takes no action and the packet is ignored. Otherwise, the CM passes control to the Data Module (DM) to create a buffer for storing any incoming traffic destined to the MN. The CM will set up a ten-second timeout period to listen for the second control message, BU-MN_LOC from the MN.

When BU-MN_LOC is received within the timeout period, the CM passes the control again to the DM for the forwarding of any buffered packet addressed to the MN's ID. The CM uses the ID in the BU-MN_LOC packets to determine if there is an active session with the MN on the system. If no prior BU-Null is received for the incoming BU-MN_LOC, the packet is ignored. As will be discussed later, the different protocols propose different security measures to ensure that both the BU-Null and BU-MN_LOC are sent by the MN. If the BU-Null is not delivered to the Loc-server, then all MN's packets sent to the server will be dropped and the MN's handover takes place just as it normally would on the vanilla protocol.

The use of a ten-second timeout for the BU-Null is based on our previous work on two different mobility protocols – LISP-MN and MIPv6 [67, 139] – as well as results from other published research works on ILNP [99], HIP [89], Fast MIPv6 [100], Proxy MIPv6 [101]. The works show that a normal handover event rarely go beyond eight seconds, and mostly below four. Hence, after ten seconds of receiving BU-Null without a follow-up message, it is safe to assume that the handover has either failed, or been cancelled by the MN, or the BU-MN_LOC is missing in transit, and there is no need to hold on to the buffer resource already reserved by the Loc-server. The CM in this case will signal the closure of the session to the DM, which in turn will free the buffer and discard any packets therein.

4.4.2 Database

The server maintains a database of all the prefixes/addresses allowed to use the location server and the list is loaded in memory when the program executes. As stated earlier, the CM uses the database to determine whether the ID extracted from the BU-Null is to be provided with location service. Where the database is owned by a telecom operator or an ISP, it will contain mostly list of prefixes allowed to use the service. For Loc-servers configured for use in a community network between a number of Wi-Fi networks in a location area, individual IDs (IP addresses) are likely to populate it. Loc-server provision is further discussed in section 4.6.

4.4.3 Data Module (DM)

The DM is the module responsible for reserving buffer space for the expected packets, receiving the packets, and forwarding them to the MN. No maximum buffer size is specified, but we can predict that 25MB would suffice to hold all packets ranging from 2.5 to 3.5 Mbps of a 720p UDP/TCP video flow [102][27] during an 8-second handover. This is the largest packet flow anticipated during the handover period, which we mentioned earlier that the period is unlikely to exceed eight seconds based on the available literature on performance measurement.

[27] The size of the video chunks used in determining enough buffer for the duration of a handover is based on the work cited.

Once the buffer is created on the prompting of the CM (creating a buffer involves adding the cleared ID from the BU-Null packet in the list of Loc-server destinations), the DM listens to incoming packets addressed to the added ID(s) of the MN(s). If BU-MN_LOC is received and no packet is in the buffer, the locator is saved for a further ten seconds (as is the case if the locator exists) after which it is removed from the list of destinations. The locator is retained to ensure the delivery of any late arriving packets.

4.4.4 Interaction between the Loc-server Modules

As shown in Figure 4.4-1 below, the different modules of the Loc-server interact depending on the control message received from the MN. The CM is central as the receiver of the MN's messages and interacting with the database and the DM. For data packets, however, it is the DM that solely handles the packets.

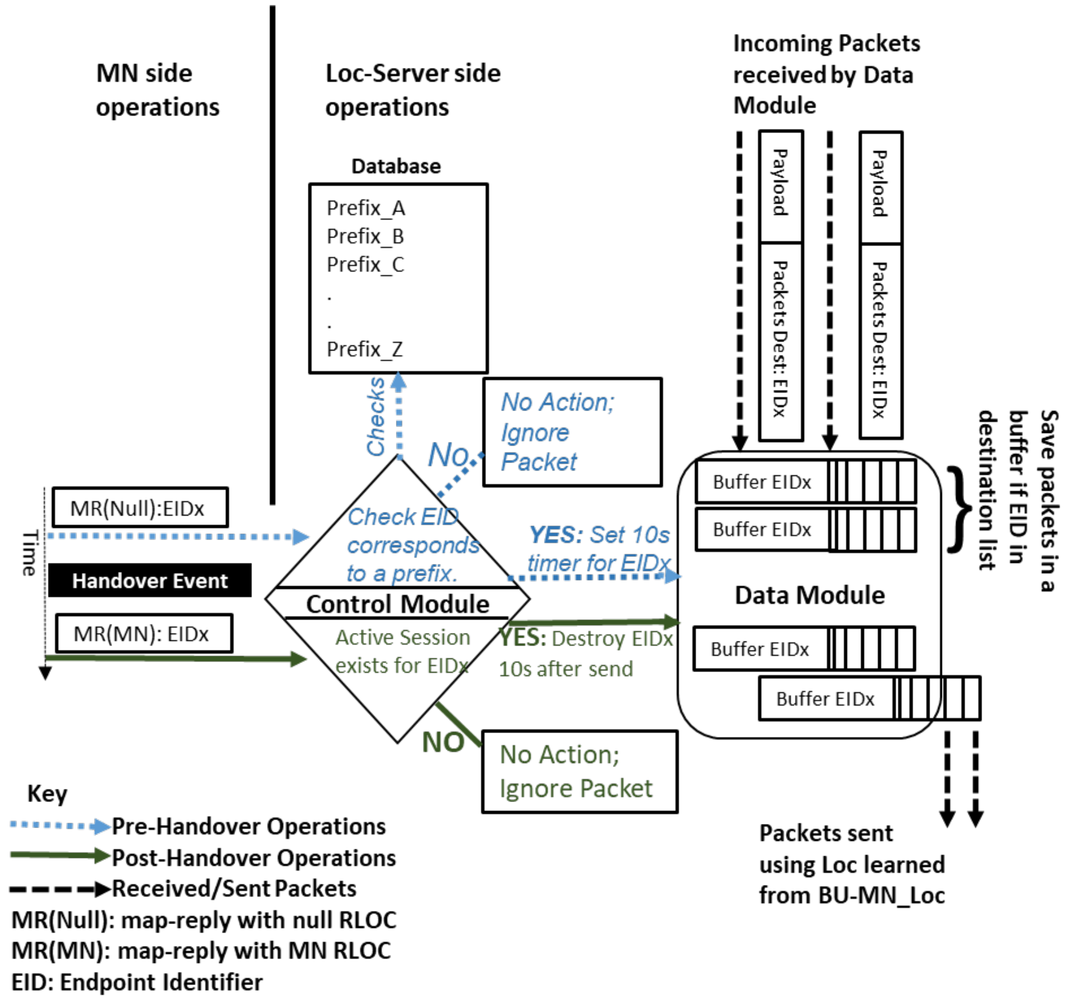
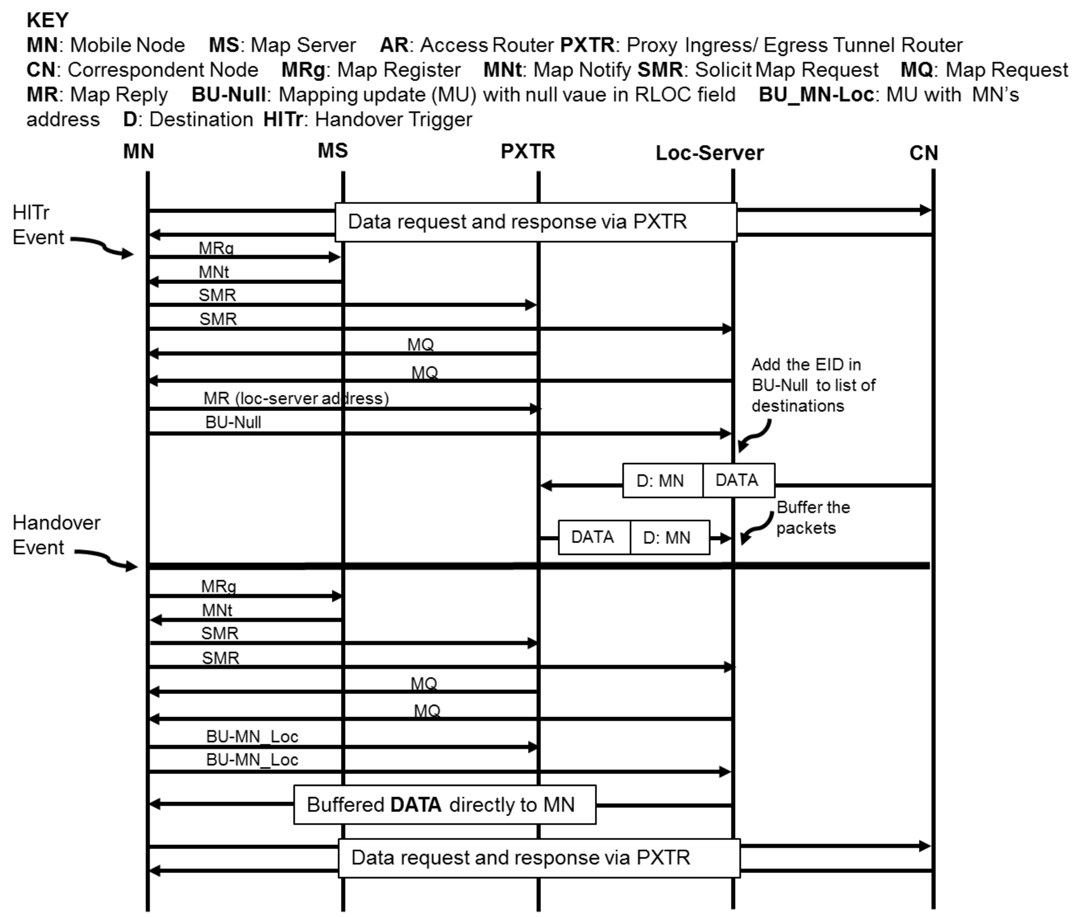


Figure 4.4-1 Interaction Between The Different Loc-server Components

4.5 Loc-server Integration Into LOC/ID split Architectures

As detailed in chapter two, the different LOC/ID protocols have slightly different techniques for mobility management procedures. In this section, we will discuss how the Loc-server can support these different protocols. The mobility protocol running on an MN needs to be able to respond to not just a change in interface configuration but also a new external event of signal quality when a certain threshold is attained. Additional messages are also needed in the control plane since there is a need to send messages to the Loc-server before a handover, as will be discussed below.


KEY
MN: Mobile Node MS: Map Server AR: Access Router PXTR: Proxy Ingress/ Egress Tunnel Router
CN: Correspondent Node MRg: Map Register MNt: Map Notify SMR: Solicit Map Request MQ: Map Request
MR: Map Reply BU-Null: Mapping update (MU) with null vaue in RLOC field BU_MN-Loc: MU with MN's address D: Destination HITr: Handover Trigger
MN
MS
PXTR
Loc-Server
CN
HITr Event
Data request and response via PXTR
MRg
MNt
SMR
SMR
MQ
MQ
MR (loc-server address)
BU-Null
Add the EID in BU-Null to list of destinations
D: MN
DATA
Buffer the packets
DATA
D: MN
Handover Event
MRg
MNt
SMR
SMR
MQ
MQ
BU-MN_Loc
BU-MN_Loc
Buffered DATA directly to MN
Data request and response via PXTR
Figure 4.5-1 Improved LISP-MN Architecture – Interaction with a Loc-server

4.5.1 Improving the LISP-MN Architecture

With LISP-MN, an MN indicates to the Map-Server and the PxTR its movement by sending map-register and SMR–cum–map-reply to the two remote nodes respectively. Since the Loc-server is now added to the architecture, messages are sent both prior and after a handover as shown in the Figure 4.5-1. The figure depicts the interaction between the MN and the Loc-server in both the control and the data planes. In the vanilla LISP-MN architecture discussed in section 2.3.4, updates are only triggered when an MN's interface is (re)configured either due to a handover or when the MN has just come on line.

In our architecture however, mapping updates are also sent on the brink of a handover. The SMR message sent to the PxTR for the proxy router to solicit the current MN’s mapping information (RLOC information) is also sent to the Loc-server. The map-request generated by the two nodes prompt the MN to send map-reply message to the duo with the Loc-server address in the LOC field of the packet going to the PXTR and a null value in the similar field for the Loc-server reply (BU-Null). Similar messages are sent in the two architectures for a newly configured interface with a similar BU_MN- LOC sent to the two nodes. This packet contains the new RLOC configured by the MN and the only difference between the packet going to the PXTR and the one going to the server is the generated nonce. The nonce value is used in LISP to ensure that packets received are from the nodes that claim to be sending them.

Due to these exchanges, the PXTR sends incoming packets to the Loc-server during the handover process as shown in Figure 4.5-1, which are buffered and sent to the MN at the completion of the process. The data plane resumes to its regular operation, with no Loc-server involvement, until another HITr event.

4.5.2 Improving the IVIP Architecture

As discussed in section 2.4.2, the IVIP specification has not provided any procedure for updating the MN’s location during communication, it is left for the TTR provider to provide such software to the customer’s mobile device. We however assumed that at least a binding update to the TTR and an acknowledgement from it must be implemented for the MN to maintain its sessions during mobility. To improve upon this architecture, a binding message will be sent to the TTR and the Loc-server upon a HITr event with Loc-server address in the TTR-bound packet and a null value for Loc-server-bound packet (BU-Null). This causes the TTR to forward incoming packets to the Loc-server and for the server to buffer the packets if the MAB address in the packet corresponds with the ID extracted from the earlier received BU-Null.

On handover completion, the MN sends another binding message to the two network nodes containing its newly acquired CoA to resume normal communication with the TTR and as a request for any buffered packet from the Loc-server.

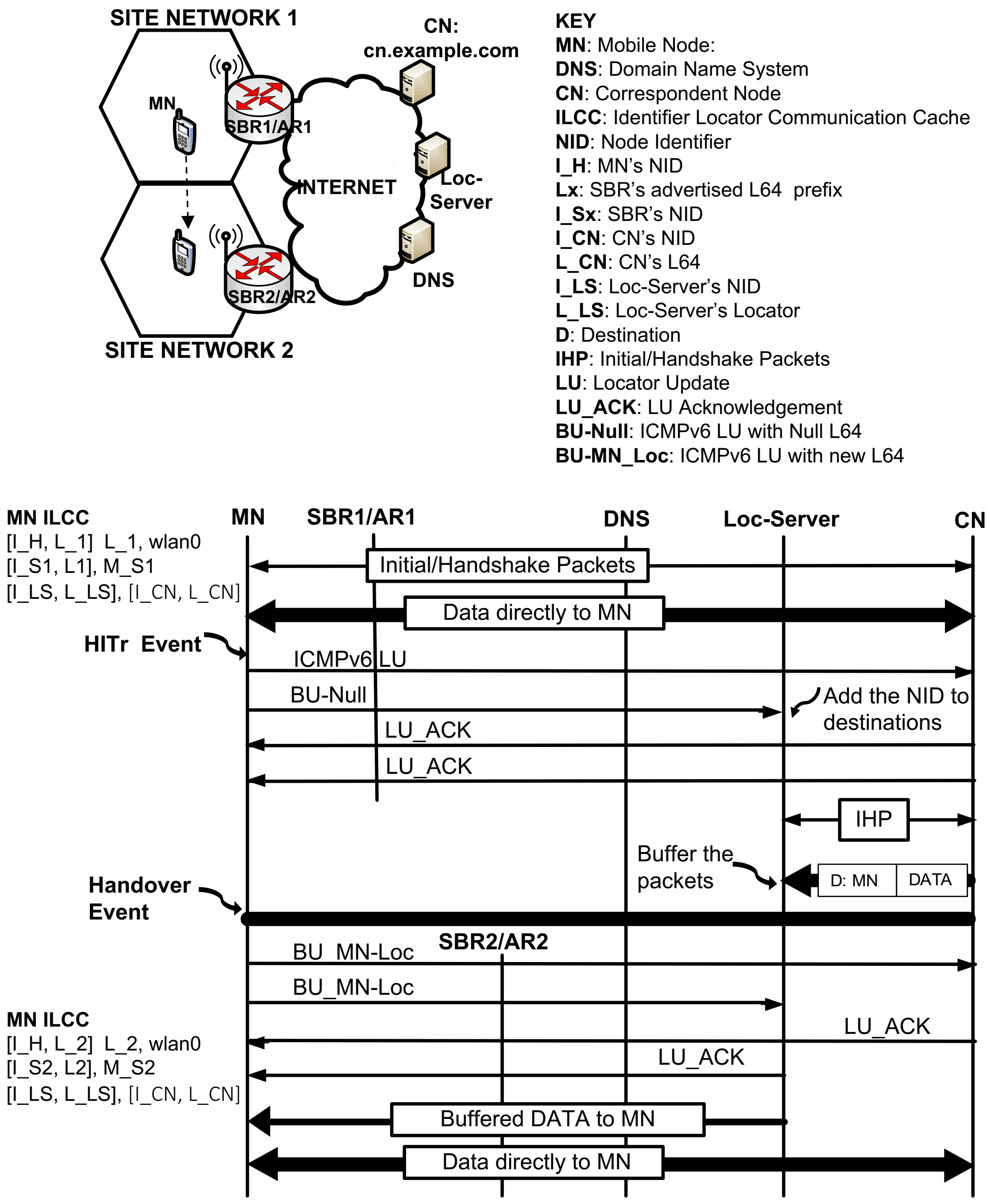
4.5.3 Improving the ILNP Architecture

Loc-server can be integrated to work with ILNP and supports its handover process when an MN communicates with an ILNP-capable CN. We learned in section 2.5.2 that an MN uses the information in its ILCC to establish a connection with other nodes on the Internet. The ILCC contains its ID and L64 value (E1 in Figure 4.5-2 below); L64 to interface binding (E2); SBR information (E3); and IL-V of the CN(s) it is communicating with (E4). This information changes to expressions E5 to E7 when the node moves to a different SBR. This change is communicated to the CN using the ICMPv6 LU message, which contains the new L64 value (LOC) obtained from the visited network.

To introduce the Loc-server, however, an MN is pre-configured with the IL-V of the server with the expression [I_LS, L_LS], which will be added in the ILCC shown in Figure 4.5-2. The new expression added as the Loc-server entry does not change because of a handover. The MN starts the IHP exchange with the server on coming online and includes in the initial packets, its ID and the ILNPv6 nonce destination option to prevent off-path and man-in-the-middle attacks. After the initial exchanges between the two nodes, the server has now authenticated the MN and listens to potential LU packets from it.

[I_H, L_1]	E1	[I_H, L_2]	E5
L_1, wlan0	E2	L_2, wlan0	E6
[I_S1, L1], M_S1	E3	[I_S2, L2], M_S2	E7
[I_CN, L_CN]	E4		

Figure 4.5-2 Identifier-Locator Communication Cache



KEY
MN: Mobile Node:
DNS: Domain Name System
CN: Correspondent Node
ILCC: Identifier Locator Communication Cache
NID: Node Identifier
I_H: MN's NID
Lx: SBR's advertised L64 prefix
I_Sx: SBR's NID
I_CN: CN's NID
L_CN: CN's L64
I_LS: Loc-Server's NID
L_LS: Loc-Server's Locator
D: Destination
IHP: Initial/Handshake Packets
LU: Locator Update
LU_ACK: LU Acknowledgement
BU-Null: ICMPv6 LU with Null L64
BU-MN_Loc: ICMPv6 LU with new L64

Figure 4.5-3 Improved ILNP Architecture - Interaction with a Loc-server

As shown in Figure 4.5-3, when a HITr event occurs, the MN sends an LU message to its CN(s) with the L64 of the Loc-server as the MN's LOC, and also sends a BU-Null (an LU with null LOC) to the Loc-server. The server afterward buffers all packets redirected to it by CNs provided the ID and the nonce values received in the packets are similar to the ones in the BU-Null. Before the CN starts sending packets to the Loc-server, it performs the IHP to confirm the identity of the node.

Once the interface is configured after the handover, the MN sends a vanilla ILNP LU message (BU-MN_LOC) to the CN and the Loc-server. This causes the Loc-server to send all buffered packets to the MN and the CN to resume normal packet delivery. The MN continues to communicate with the CN normally after the remote device acknowledges the MN's LU message.

4.5.4 Improving the HIP Architecture

To start any HIP-based communication, an initial four-way handshake called BEX (discussed in section 2.6.2) needs to take place for the two communicating nodes to confirm the identity of each other. This is also the case when integrating a Loc-server into the HIP architecture. The server needs to be HIP-enabled and has a public and private DH key for the generation of its HIT and for computing the session key for communication. The MN and the server perform the BEX when the MN comes online, creating the ESP-SA and priming the server to listen to the MN's HIP update messages. Since HIP sessions are terminated after thirty minutes of inactivity, the MN needs to re-establish a connection with the server after every termination or the two devices can be configured to retain such sessions for a longer period.

Upon a HITr event, the MN sends HIP updates to both the CN and the Loc-server. This update message carries the IP address of the Loc-server in the

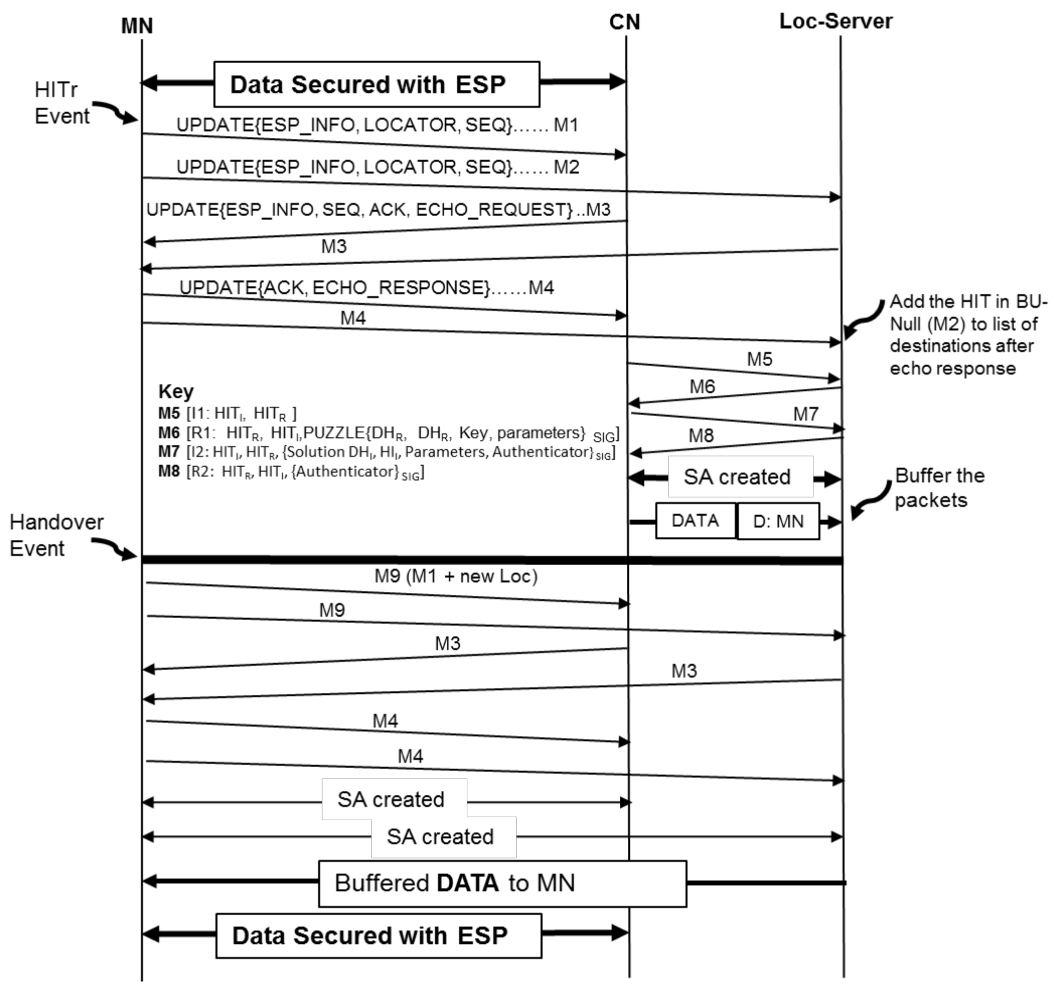

MN
CN
Loc-Server
Data Secured with ESP
HITr Event
UPDATE{ESP_INFO, LOCATOR, SEQ}...... M1
UPDATE{ESP_INFO, LOCATOR, SEQ}...... M2
UPDATE{ESP_INFO, SEQ, ACK, ECHO_REQUEST}..M3
M3
UPDATE{ACK, ECHO_RESPONSE}......M4
M4
Add the HIT in BU-Null (M2) to list of destinations after echo response
M5
M6
M7
M8
Key
M5 [I1: HIT_I, HIT_R]
M6 [R1: HIT_R, HIT_I,PUZZLE{DH_R, DH_R, Key, parameters} SIG]
M7 [I2: HIT_I, HIT_R, {Solution DH_I, HI_I, Parameters, Authenticator} SIG]
M8 [R2: HIT_R, HIT_I, {Authenticator} SIG]
SA created
Buffer the packets
DATA
D: MN
Handover Event
M9 (M1 + new Loc)
M9
M3
M3
M4
M4
SA created
SA created
Buffered DATA to MN
Data Secured with ESP

Figure 4.5-4 Improved HIP Architecture - Interaction with a Loc-server

LOC parameter of the CN-bound packet (Message 1 [M1] in Figure 4.5-4), and a null value in the IP address field of the Loc-server-destined message (M2/BU-Null). Again as discussed in section 2.6.2, the CN (and now the Loc-server) validates the message using the MN's information from the BEX process. Unlike with ILNP where a response to the binding update and BU-Null from the CN and the Loc-server respectively are not necessary, it is vital in the HIP architecture to receive the M3 from the two remote nodes, and the MN responds with M4 to the two. This is necessary because M3 is an echo request used to verify the new address and until an echo reply is received in M4, neither the CN will use the new LOC to redirect packets to the Loc-server, nor the Loc-server will buffer packets for the MN.

If the CN has no recent ESP SA with the Loc-server, it needs to create one using the BEX with the messages M5 to M8 before forwarding any MN's packet to the server. Similar messages to the ones triggered upon a HITr event are exchanged after a handover (M9 in Figure 4.5-4) except that M1 now carries a new LOC. This will cause the server to send all buffered packets destined to the HIT of the MN to the LOC identified from the HIP update message. The update to the CN will cause the node to route subsequent packets to the MN's current location.

4.5.5 Improving the MIPv6 Architecture

The Loc-server can support MNs on an MIPv6 network when leaving or coming into their home networks as well as for roaming between foreign networks. Upon a HITr event, the MN sends a BU to its HA with the IP address of the Loc-server as its new CoA. It also sends another BU to the Loc-server with a null CoA as a buffering request to the server. The HA continues to redirect the MN's packets to the Loc-server during the handover process. Once the handover is completed, the MN sends a BU (BU-MN_LOC) to the two remote nodes for the forwarding of buffered packets by the Loc-server, and resumption of normal communication with the HA (see Figure 4.5-5A). All the BU and acknowledgement messages exchanged between the MN, the HA and the Loc-server are secured using the IPSec mechanism as specified in the MIPv6 RFC [90].

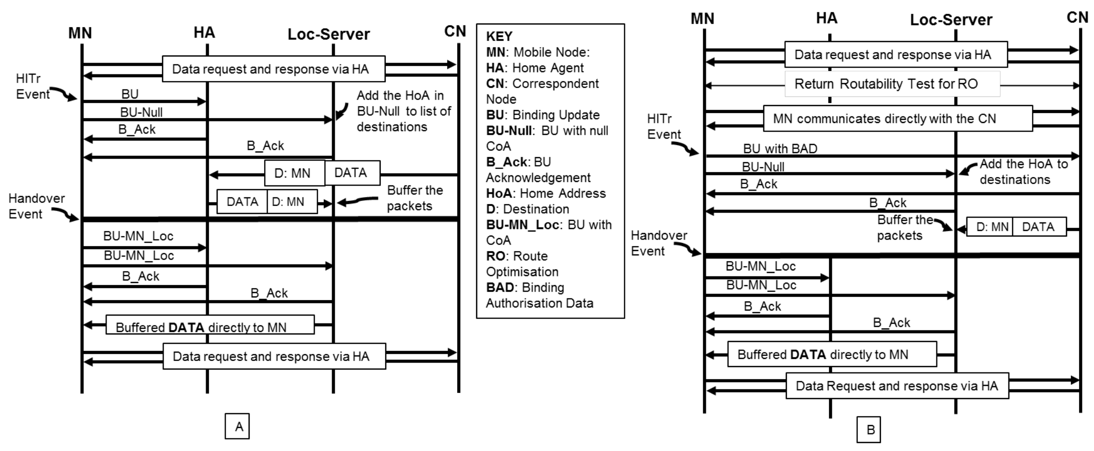

MN
HA
Loc-Server
CN
Data request and response via HA
HITr Event
BU
BU-Null
B_Ack
Add the HoA in BU-Null to list of destinations
B_Ack
D: MN
DATA
DATA
D: MN
Buffer the packets
Handover Event
BU-MN_Loc
BU-MN_Loc
B_Ack
B_Ack
Buffered DATA directly to MN
Data request and response via HA
A
KEY
MN: Mobile Node:
HA: Home Agent
CN: Correspondent Node
BU: Binding Update
BU-Null: BU with null CoA
B_Ack: BU Acknowledgement
HoA: Home Address
D: Destination
BU-MN_Loc: BU with CoA
RO: Route Optimisation
BAD: Binding Authorisation Data
Data request and response via HA
Return Routability Test for RO
MN communicates directly with the CN
BU with BAD
BU-Null
B_Ack
Add the HoA to destinations
B_Ack
Buffer the packets
D: MN
DATA
BU-MN_Loc
BU-MN_Loc
B_Ack
B_Ack
Buffered DATA directly to MN
Data Request and response via HA
B

Figure 4.5-5 Improved MIPv6 Architecture - Interaction with a Loc-server

If route optimisation is in place and the MN was directly communicating with the CN, then the BUs triggered as a result of a HITr event will be to the Loc-server as well as to the CN using the binding authorisation data computed with thebinding management key during the return routability test as discussed in section 2.7. Figure 4.5-5 shows how MIPv6 is improved with Loc-server for the protocol's operation without optimisation (A) and with route optimisation (B).

In scenario B, the MN may default to using the HA after the handover as shown in the figure or update its binding with the CN provided that the maximum return routability binding lifetime of 420 seconds, as defined in the protocol's specification, has not been exceeded. In which case, it has to go through the return routability procedure via the HA once again.

4.6 Discussion regarding the Design

As we have seen so far in this chapter, ILISA is designed in such a way that it is backward compatible with the legacy architectures upon which it is implemented. Where there is a failure in communication between the MN and the Loc-server, the protocols operate in their conventional manner without any interruption. Similarly, if the handover fails to take place after an alert for it has been sent to the mapping system, the HFG ensures that communication is resumed two seconds later. In this discussion, however, we look further at the security aspects of ILISA, the provisioning of the Loc-server in a real deployment, and whether the requirements of the new system discussed in chapter three are met.

4.6.1 Loc-server Security

ILISA focuses on the handover enhancement of LOC/ID protocols and the Loc-server is just as secure and as vulnerable as the mobility anchors and/or CNs of the different protocols discussed. LISP-MN, for instance, specified the use of a 24-bit ‘Nonce’ field in the LISP encapsulation header for the data plane and a 64-bit 'Nonce' field in the LISP control message. The use of nonce makes it difficult for off-path attackers to launch attacks with no prior knowledge of the nonce values – and these values are generated on the fly. As with PXTR, the Loc-server is designed only to accept map-reply sent as a result of the server’s SMR, and the nonce in the reply must match the nonce it generated and sent in the map-request to the MN.

ILNP has also specified the use of nonce serving as the mechanism for identifying ILNP capable nodes. It also serves as a security measure against off-path attacks during communication. Both the MN and the CN exchange the IHP with the Loc-server before any communication can take place. This is a similar case with HIP, for which security is at the core of its working principles. The Loc-server necessarily needs to have DH public and private keys to be able to communicate with HIP-based MN and CNs. The server performs BEX with the remote nodes to secure the communication lines and uses echo requests and responses before accepting HIP update messages. For operating in a MIPv6 setting, the use of IPSec is adopted in the Loc-server design.

4.6.2 Loc-server Service Provision

An important question to answer is, who provides the location area service when these protocols are improved with the introduction of the Loc-server? While the Loc-server can be implemented to work at different levels of network topology, we see three likely implementation scenarios in the future – telecom operators, Internet Service Providers (ISPs) and community networks.

4.6.2.1 Telecom Operators

We envisaged that LISP-MN might be adopted as the mobility protocol of the future. With telecom operators being the ubiquitous source of Internet for billions around the globe, it is the first candidate for loc-server deployment. Telecom operators are the largest providers of wireless Internet connectivity via the cellular networks and are often the owners of home broadband as well as Internet services provided in shops and malls. The 3GPP’s evolved packet core (EPC) is an all-IP backbone network for the 4G LTE. In addition to the LTE, EPC provides the means for connecting the backbone network to the earlier generation of 3GPP’s systems, such as 2G and 3G. It also provides connectivity to non-cellular technologies such as Ethernet, WLAN, and PSTN as well as non-3GPP cellular networks such as WiMAX and 3GPP2’s Cellular systems. The combination of the LTE network, usually termed Evolved UMTS Terrestrial Radio Access Network and EPC is simply referred to as and Evolved Packet System or EPS. As depicted in Figure 4.6-1, the EPC provides access to the backbone network via a number of nodes.

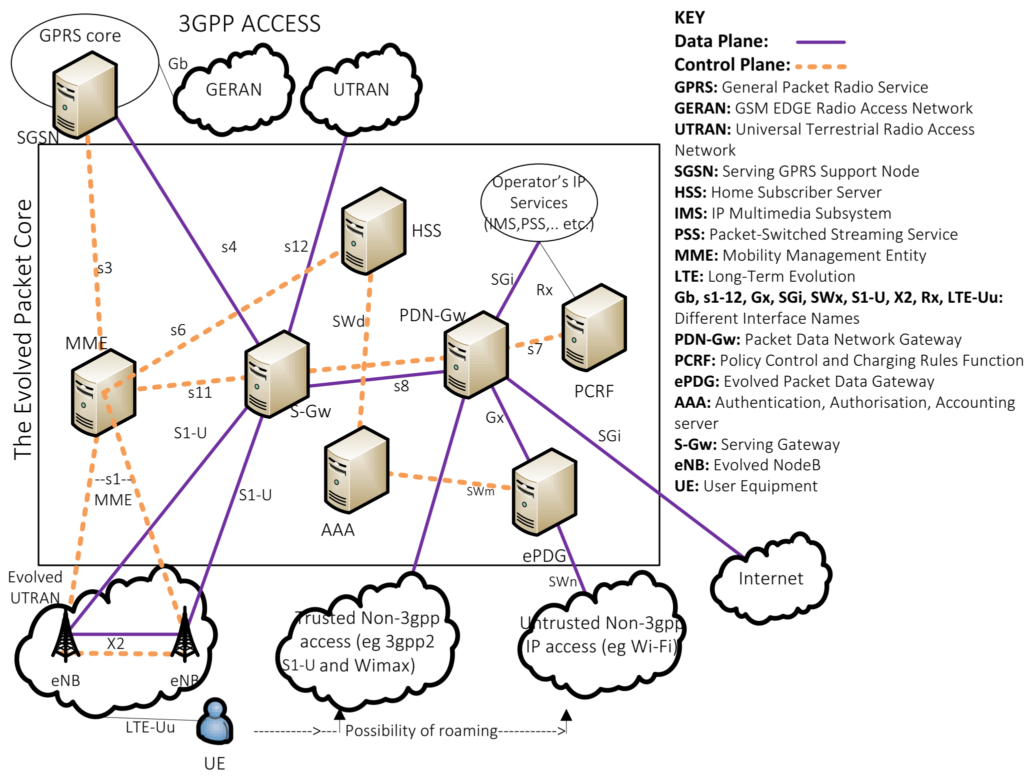

3GPP ACCESS
GPRS core
SGSN
Gb
GERAN
UTRAN
The Evolved Packet Core
s3
s4
s12
HSS
Operator's IP Services (IMS,PSS,.. etc)
SGi
Rx
SWd
PDN-Gw
s6
MME
s7
s11
S-Gw
s8
PCRF
S1-U
Gx
SGi
--s1--MME
S1-U
AAA
SWm
ePDG
Evolved UTRAN
SWn
Internet
Trusted Non-3gpp access (eg 3gpp2 S1-U and Wimax)
Untrusted Non-3gpp IP access (eg Wi-Fi)
X2
eNB
eNB
LTE-Uu
UE
----------->--- Possibility of roaming---------->
KEY
Data Plane:
Control Plane:
GPRS: General Packet Radio Service
GERAN: GSM EDGE Radio Access Network
UTRAN: Universal Terrestrial Radio Access Network
SGSN: Serving GPRS Support Node
HSS: Home Subscriber Server
IMS: IP Multimedia Subsystem
PSS: Packet-Switched Streaming Service
MME: Mobility Management Entity
LTE: Long-Term Evolution
Gb, s1-12, Gx, SGi, SWx, S1-U, X2, Rx, LTE-Uu: Different Interface Names
PDN-Gw: Packet Data Network Gateway
PCRF: Policy Control and Charging Rules Function
ePDG: Evolved Packet Data Gateway
AAA: Authentication, Authorisation, Accounting server
S-Gw: Serving Gateway
eNB: Evolved NodeB
UE: User Equipment

Figure 4.6-1 The Evolved Packet System

The packet data network gateway (P-GW) is the gateway to the Internet and the IMS, and serves as the following network nodes: an MN's default gateway; HA (or Local Mobility Anchor for PMIPv6); and also a DHCP server. It provides access to the different wireless networks and could potentially serve as the Loc-server regardless of the protocol implemented. This set-up could be used to serve MNs in both heterogeneous and homogeneous wireless environments. For instance, an MN switching from home Wi-Fi to cellular could provide the IP address of the P-GW as its Loc-server and the Wi-Fi network forwards these packets to the P-GW, which deliver the packets to the MN on handover completion. This function could also be performed by ePDG[28] in conjunction with the PGW, since the ePDG is designed for untrusted non-3GPP IP access.

[28] Evolved Packeg Data Gateway

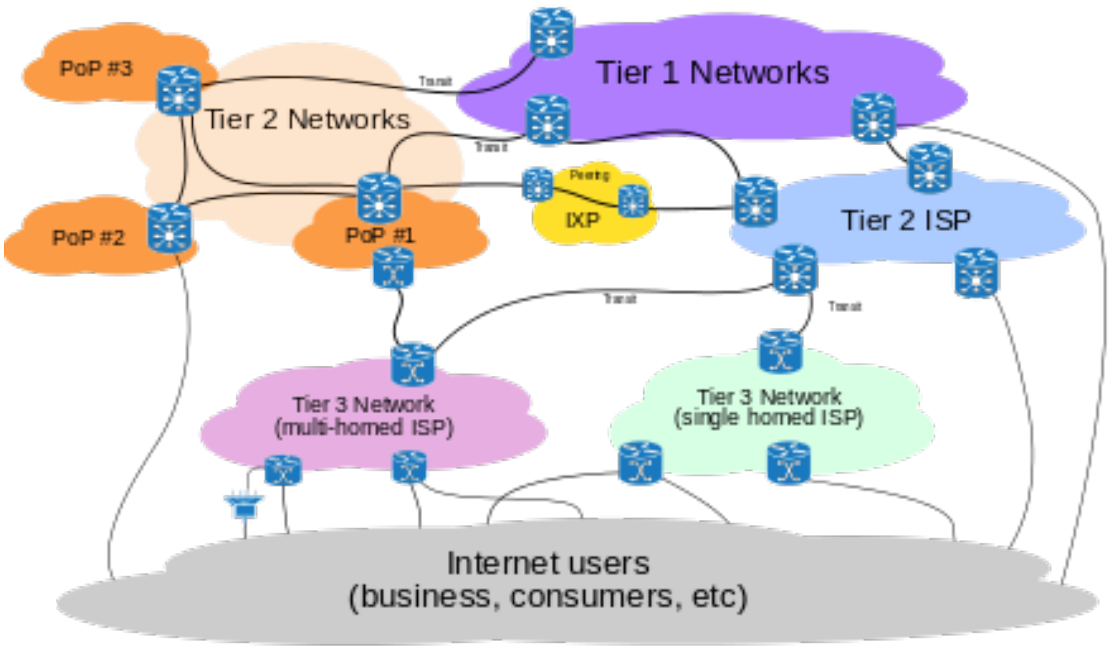


Figure 4.6-2 ISP Hierarchy (Image Source [103])

4.6.2.2 Internet Service Providers (ISP)

As shown in Figure 4.6-2, ISPs are usually classified as tier 1, tier 2 and tier 3. Whilst the tiers are not authoritatively defined, tier 1 ISPs are usually designed to serve tier 2 and tier 3 ISPs as well large enterprises, with their direct connection to the Internet backbone. Whereas tier 2 networks also provide service to companies and tier 3 ISPs, the tier 3 providers are usually the networks that are directly connected to the small businesses, consumers etc. Hence, Loc-servers could be easily cited at tier 3 ISPs servicing a particular geographical area since users' packets are likely to be sent and received via those ISPs.

4.6.2.3 Community Networks

The Loc-server could also simply be provided for use in community networks that are based on Lowest Cost Denominator Networking (LCDNet) – a set of network techniques that enable users to share their home broadband network with the public [104]. One example of an architecture based on LCDNet is the public Wi-Fi access service, or simply PAWS [92]. PAWS enables broadband customers to allow free public access to an unused portion of their Internet by providing a less than best effort service. A Loc-server could be hosted by a local council (or a communal centre) whereby users accessing the public wireless network whilst on the move could have their packets redirected within the same local community since they are moving between adjacent ANs.

The Loc-server could also be installed to serve a group of businesses providing free Internet to their customers. For instance, on a market street (or a station, shopping mall, airport) with a number of shops, a Loc-server could be installed in one of the shops whereby users' packets are routed to the user device targeted AN as a user move and handover while within the premises. It could also be installed in a neighbourhood especially where the 'open wireless' movement [91] is embraced as discussed in the previous chapter. As users move around the neighbourhood and connect from one home network to another, the Loc-server forwards their packet to their new location as they move.

4.7 Chapter Summary

This chapter presented the design description of the newly developed ILISA. We discussed how the new architecture operates, and presented its different components as well as the necessary modules that need to be implemented on the MN and Loc-server for the architecture to function. The chapter also detailed how the four LOC/ID split protocols' architectures as well as the MIPv6 discussed in chapter two can be improved with the introduction of the Loc-server. The chapter concluded with the discussion of Loc-server's deployment in real life networks and identified three potential implementation scenarios – Telecom Operator's network, Internet Service Providers, and Community Networks.

CHAPTER FIVE

IMPLEMENTATION

This chapter provides the details of how the system design described in chapter four was realised. The implementation was carried out on an IPv6 network for the simple reason that IPv6 is protocol of the future, and is already becoming commonplace. As discussed in the research objectives, the system is designed in such a way that it can provide handover support to at least the four LOC/ID protocols discussed, and MIPv6. To test the system, we focus our implementation on working with LISP-MN as a representative of other LOC/ID protocols for reasons we discuss in the next section.

The development environment upon which the architecture is implemented is presented in section 5.1. The section describes LISPmob (LISP-MN) and umip (MIPv6) codebases used for the PhD work. Section 5.2 describes how the LISPmob codebase is enhanced to introduce new features to enable interaction with the Loc-server. Section 5.3 details the implementation of the different Loc-server components discussed in the system design chapter (chapter four).

5.1 Development Environment

To demonstrate the improvement that our proposed architecture brings to LOC/ID protocols' mobility events, we set up an IPv6 laboratory-based network testbed (described below) at Lancaster University consisting of all the necessary LISP-MN and MIPv6 components. We chose LISP-MN for the implementation for two reasons: firstly, all the required network nodes for the protocol's operation can be implemented on a laboratory testbed; secondly, tests could potentially be carried out on the Internet using the open LISP beta network [105]. As far as we know, none of the other protocols have a dedicated Internet overlay network similar to the LISP beta network. We have also mentioned in section 3.3 that we would be using MIPv6 as the benchmark for measuring the handover performance of LOC/ID split protocols. Hence there was an MIPv6 implementation also running on the testbed.

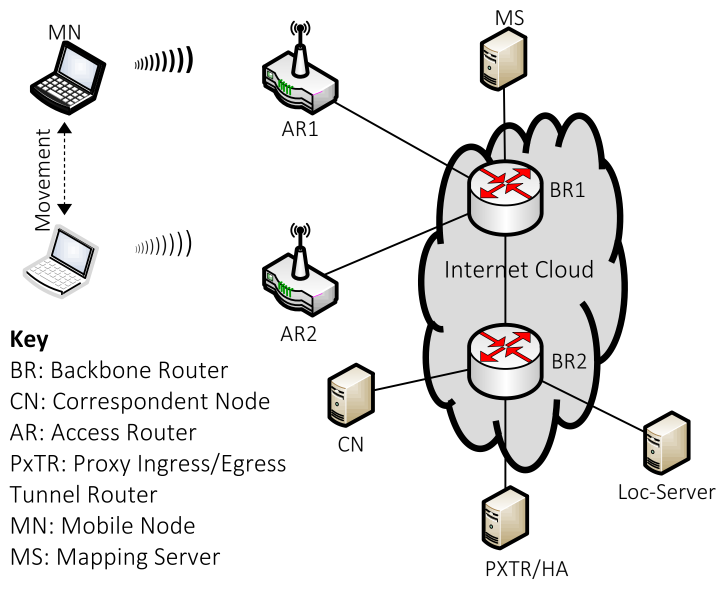


Figure 5.1-1 Development Testbed Showing the Different Components of the Systems

5.1.1 Implementation Testbed

The testbed network as shown in Figure 5.1-1 consists of nine x86 desktop PCs running Ubuntu Linux 3.13 distribution, configured accordingly as one MN, two ARs, one map-server, one PxTR/HA, one CN, two backbone routers, and one Loc-server. The systems are connected via Ethernet except the wireless links between ARs and the MN, and reachability across the network is achieved using dynamic routing with RIPv2.

5.1.2 LISPmob Codebase for LISP-MN

For the LISP-MN functionality on the network, we use an implementation called LISPmob [106], running on the MN, the map-server, and the PxTR. As far as we know, it is the only available implementation for LISP-MN protocol and all its associated components. The LISPmob code was initially developed at CISCO as an implementation of LISP-MN [64], but the project is currently maintained by Barcelona Tech University's Broadband Communications Research Group

[107] with the support of a wide community of researchers, companies, and startups. The code is written in C. To run the protocol, we cloned a copy of the project using git and extracted the content to a folder on the three LISP nodes – the MN, the map-server/resolver and the PXTR. The LISPmob-0.5.1 version was compiled using the GCC compiler on the MN and the map-server, while LISPmob-PXTR version 0.4 was used on the PXTR device – the 0.5 version has no PXTR functionality at the time of our set-up. All the necessary dependencies as detailed on the LISPmob website [106] were installed on the Ubuntu PCs and the changes required in the 'sysctl.conf' system files, as detailed on the project's website, were also effected on the nodes. The LISPmob project has now matured and is commonly referred to as open overlay router [108].

LISPmob consists of three major components as highlighted in its development site [58]:

1. Data Plane: this is implemented as a kernel space program with two modules, *lisp_int.ko*, responsible for creating a virtual interface for the EIDs; and *lisp.ko*, which handles encapsulating/decapsulating of packets and maintaining the mapping cache among other functions.
2. Control Plane: this is implemented as a user space daemon, *lispd*, responsible for sending control messages, map-register, map-notify, SMR, map-request, map-reply, etc, as well as managing interfaces.
3. External tools to support testing and debugging.

Nodes Configuration

The LISP-MN runtime interface and mapping information are provided via a configuration file, which is defined prior to starting the daemon. Each of the three LISP nodes requires certain information and specifications to run as highlighted below:

1. Mobile Node: requires map-server/resolver IP address, key type and password for authentication with the map-server, proxy-reply by the map-server off, PXTR IP address, EID-Prefix of the MN, and RLOC interfaces and their priorities, rloc_probe specification, etc.
2. Map Server/Resolver: requires information on control interface, EID-Prefixes allowed, key type and password of sites allowed to use the server, PXTR IP address, etc.
3. Proxy Tunnel Router: map-server/resolver, EID-Prefixes allowed, rloc_probe, etc.

5.1.3 UMIP codebase for MIPv6

For the MN and HA implementation of MIPv6, we use the open source implementation of the protocol called *umip*, which is developed in compliance with the MIPv6 RFC 6275 specification [12]. The implementation work was originally carried out by the GO-Core project of Helsinki University of Technology [109] in collaboration with the WIDE Nautilus6 working group [110], but it is currently maintained by the umip.org project [111]. To run the umip code, a mobility ready kernel from version 3.0 or later is required, otherwise, patches need to be installed when using the previous kernel versions. Even with the new versions, some relevant kernel modules need to be manually enabled on a vanilla Ubuntu distribution, and the kernel needs to be recompiled afterward.

umip is written in C as a user space daemon, *mip6d*, handling both the control and the data planes. Although designed to work in the user land, umip can interact with the kernel by using the Linux xfrm framework architecture to initiate input and output control – the ioctl – system calls for tunnel processing. It also uses netlink sockets for IPv6 route processing [112]. We cloned the source files from umip's git repository, compiled and installed the files on the MN and the HA together with other necessary programs essential for the program to run. The MN's configuration file contains the HoA and the associated interfaces, the HA's IP address, binding lifetime, and IPsec configuration parameters. The HA, on the other hand, contains a list of allowed prefixes/IP addresses as well as the IPsec configuration.

5.2 Enhancing the LISPmob Code

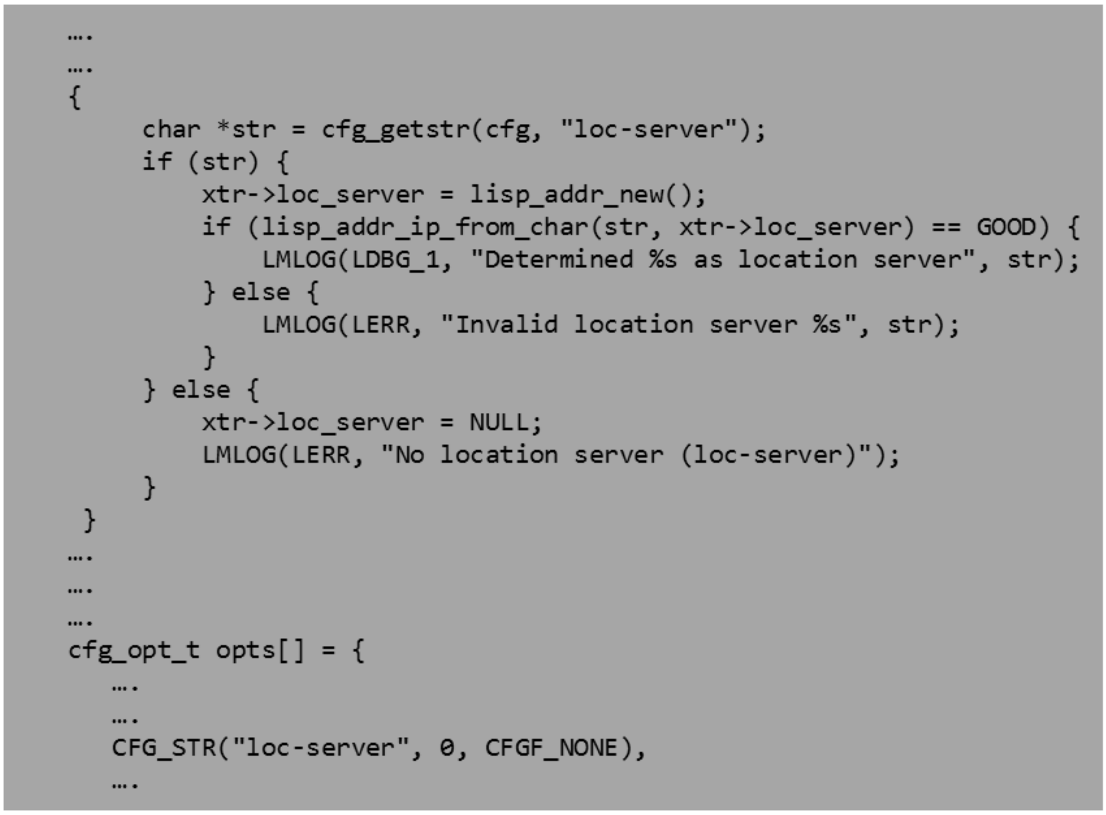
Each LISP-capable node runs an instance of *lispd* to handle LISP signaling and exchange data with the LISP overlay. *lispd* itself contains an xtr object, which holds locally pertinent LISP state, and responds to incoming messages and local events by updating the state and generating new messages. The xtr object is aware of all the nodes that the MN communicates with such as the PXTR and the map-server, which enables it to send the relevant messages, and react based on the system state.

The implementation work on the MN is divided into four different aspects: movement emulation; Loc-server introduction; HITr implementation: and HaH implementation. While movement emulation is entirely external to the LISPmob code operation, the other three aspects of enhancing the LISP-MN involve the modification of the LISPmob code so that the xtr object becomes aware of the new external stimulus of reacting to signal strength, and actions to take upon the event. It also needs to be aware of the new Loc-server and the types of messages to send and receive from it.

5.2.1 Movement Emulation

With all the devices on the testbed being desktop computers, it was impractical to achieve the MN's movement by physically moving the device away from the access point. Instead, we wrote a shell script on the ARs that gradually reduces the signal strength beamed to the MN to trick the mobile device into believing that it is drifting away from the current AR. This causes the LISP-MN protocol to react when the signal strength reaches a certain threshold set in our experiment. The script is run as a super user with the transmission power starting at its default of 20 dBm, and gradually reducing intermittently until the signal transmission is eventually completely diminished, by which time the mobility protocol has already reacted to the low signal strength event.

We did not consider the speed of movement for a walking, cycling, or driving user, in determining how fast or slow the signal strength should diminish. This is due to constraints on the type of devices on our testbed – all PCs. We nevertheless do not expect the results to be different provided that the signal level that triggers handover is maintained for the different speeds of user mobility.



```
….
….
{
     char *str = cfg_getstr(cfg, "loc-server");
     if (str) {
         xtr->loc_server = lisp_addr_new();
         if (lisp_addr_ip_from_char(str, xtr->loc_server) == GOOD) {
             LMLOG(LDBG_1, "Determined %s as location server", str);
         } else {
             LMLOG(LERR, "Invalid location server %s", str);
         }
     } else {
         xtr->loc_server = NULL;
         LMLOG(LERR, "No location server (loc-server)");
     }
 }
….
….
….
cfg_opt_t opts[] = {
   ….
   ….
   CFG_STR("loc-server", 0, CFGF_NONE),
   ….
```

Figure 5.2-1 Part of Code Introducing Loc-server into LISPMob code

5.2.2 Loc-server Introduction

To make the xtr object aware of a remote Loc-server, we added some code in the 'lispd/lispd_config_confuse.c' file, as shown in Figure 5.2-1. This enables us to add an extra field that holds the IP address of the Loc-server in the MN's configuration file discussed earlier. LISPmob code uses the *libConfuse* configuration file parser library to enable entries such as that of the Loc-server and other entries in the configuration file. The configuration system recognizes a Loc-server entry and uses it to initialise the field making the server available in the context of the program execution after the configuration file is parsed at the program start.

5.2.3 Handover Imminent Trigger (HITr) Implementation

The most comprehensive framework for improving handover performance of mobile devices is the IEEE 802.21 MIH standard [37], which defines some events, commands and information services that could be used to trigger and execute a handover by an MN. It is designed to be implemented as a '*layer-2.5*' platform in the TCP/IP protocol stack monitoring, among many other functions, the state of the available layer-2 technologies of a device. LINK_GOING_DOWN is one of the primitives of event services defined in the framework to enable an MN monitor the signal strength of a particular link and alert the upper layers of the protocol stack to take appropriate action.

Several complex handover prediction algorithms are proposed in the research literature based on the MIH framework as we see with the works of [113-115]. An implementation of the MIH framework titled 'open dot twenty one,'g or simply ODTONE [116], implements some of the primitives specified in the IEEE working group document referenced above, including the LINK_GOING_DOWN trigger, which is of interest to us.

However, we found the ODTONE implementation to be too cumbersome for our purpose as it implements many other MIH features that are not relevant to our work but that we need to implement them nonetheless in order to use the program. Hence, for HoWE and Wi-Fi to cellular handover in HeWE, we opted to implement our own 'link going down' feature that is used to constantly monitor the signal strength of the current access link using the algorithm shown in Figure 5.2-2 below. The program is external to *lispd* code.

```
Begin
    start_rssi = current_signal_strength
    constant = amount_of_change_in_signal
    threshold = start_rssi - constant
    current_rssi = current_signal_strength
    while (threshold <= current_rssi)
        recompute(current_rssi)
    done
    open(FIFO)
end
```

Figure 5.2-2 Pseudocode of Algorithm for Detecting Signal Quality Degradation on a Wireless Interface

At the start of the program, the current signal to noise ratio (SNR) of the received signal on the MN's active wireless interface is read from the Linux '/proc' entries and the result saved in the 'start_rssi' variable. A 'constant' value is defined, which represents the amount of decrease in the SNR after which the link is considered to be going down and at which point the xtr object of the LISPmob code needs to be alerted. The threshold is derived by subtracting the constant from the start_rssi; the value is saved in the variable 'threshold.' The variable 'current_rssi' is derived in a similar way as the start_rssi and used in the loop to constantly monitor the signal strength. The current_rssi is continuously compared with the threshold, and the loop is terminated when their values are the same, or the latter is less than the former. At this point, the script opens and closes a *named pipe* (a FIFO), as a way of alerting the *lispd* daemon. As we will show subsequently, the other end of the pipe is accessed by other *lispd* functions.

As established in chapter three, Wi-Fi connectivity is mostly prioritised in a HeWE against cellular connectivity mainly for the factor of cost and sometimes speed. Hence for cellular to Wi-Fi handovers in HeWE, we utilise the *priority* feature provided in the configuration file of the LISP-MN protocol, which allows the specification of the interface that the MN should always use provided it is available.

5.2.4 Handover Handler (HaH) Implementation

The first aspect of HaH implementation, pre-HaH, is to respond to the alert from HITr when the signal strength reaches the defined threshold. The response, as earlier discussed, is to send an SMR to the PXTR and Loc-server, which is contrary to the protocol's normal operation of only sending the SMR on interface configuration, and only to the PXTR. SMR as explained earlier is a security measure by the protocol to ensure that mapping information is only sent by the MN at the request of the remote nodes.

5.2.4.1 Pre-Handover Handler Operation

When *lispd* is run, an extra Boolean field (signal_dropping, set to FALSE by default) is added in the program's context to guide the action to be taken by the xtr object on HITr alert. On receiving the trigger, the Boolean value is set to TRUE prompting the program to send SMR to the PXTR. Consequently, a map-request is sent by the PXTR, and *lispd* on the MN responds with a map-reply containing Loc-server address to the PXTR and another map-reply with no locator to the Loc-server. The flow of action is shown in the flowchart of Figure 5.2-3 below.

Reporting Signal Quality Degradation to the Program

We defined three additional functions in the LISPmob code to achieve this, *init_qualmon*, *handle_qual_decrease* and *xtr_report_signal_drop*. When invoked at program execution, *init_qualmon* removes any existing pipe before creating a new FIFO. It records the xtr as the object to invoke when the signal level drops. It then opens the FIFO for reading and hands over control to the

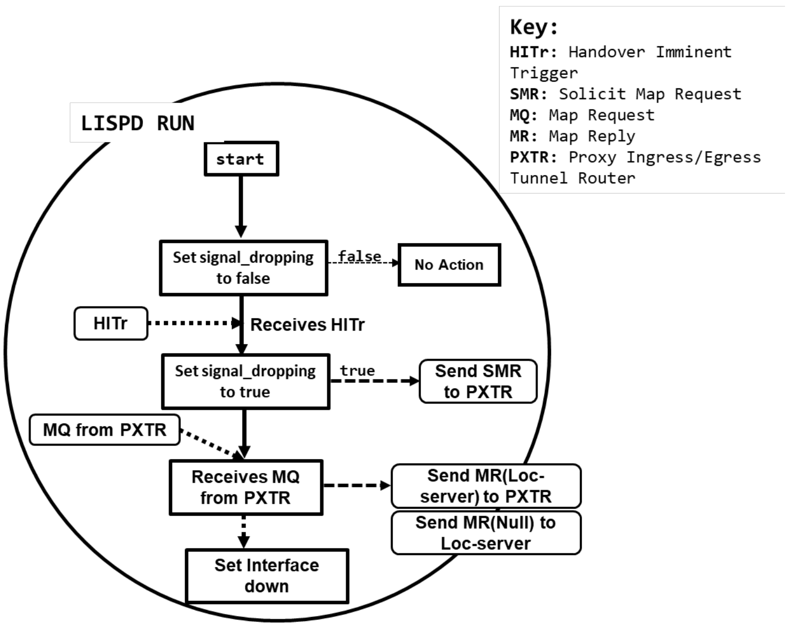
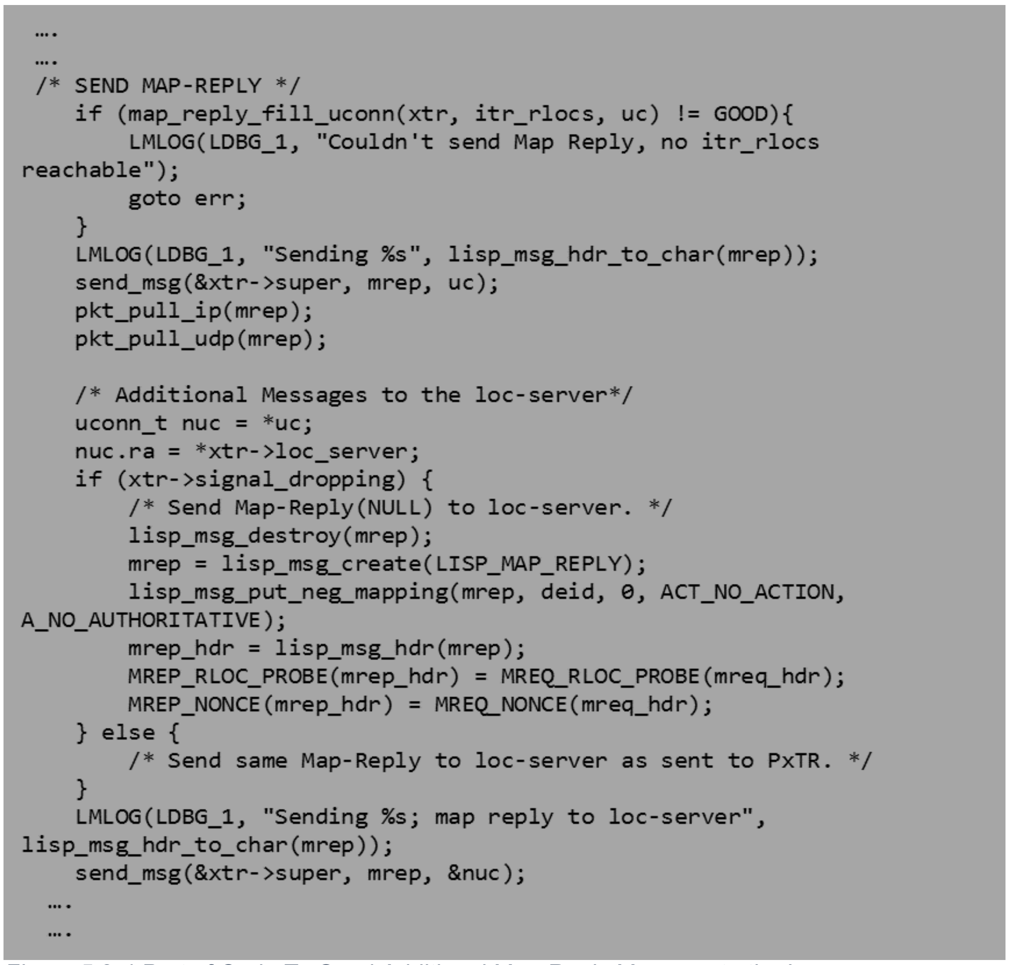

Key:
HITr: Handover Imminent Trigger
SMR: Solicit Map Request
MQ: Map Request
MR: Map Reply
PXTR: Proxy Ingress/Egress Tunnel Router
LISPD RUN
start
Set signal_dropping to false
false
No Action
HITr
Receives HITr
Set signal_dropping to true
true
Send SMR to PXTR
MQ from PXTR
Receives MQ from PXTR
Send MR(Loc-server) to PXTR
Send MR(Null) to Loc-server
Set Interface down

Figure 5.2-3 Pre-Handover Handler Operation

other external function *handle_qual_decrease*. This function repeatedly opens the FIFO file, awaits the closure of the other end of the FIFO, and reopens it. Detection of the closure invokes the *xtr_report_signal_drop*, which is responsible for altering the value of *xtr->signal_dropping* from FALSE to TRUE, causing the program to behave in the manner shown in Figure 5.2-3. *handle_qual_decrease* is registered with *lispd*'s socket master, which detects the FIFO closure on its behalf.

Communicating with the Remote Nodes

The socket master detects the FIFO becoming readable, and invokes *handle_qual_decrease*, which in turn invokes *xtr_report_signal_drop*, which sets the Boolean to TRUE and then sends an SMR to the PxTR. We need to make the xtr object tell the PxTR to deliver future packets to the Loc-server by sending map-reply(Loc-server) to the PXTR, and also make the Loc-server to



```
 ….
 ….
 /* SEND MAP-REPLY */
    if (map_reply_fill_uconn(xtr, itr_rlocs, uc) != GOOD){
        LMLOG(LDBG_1, "Couldn't send Map Reply, no itr_rlocs
reachable");
        goto err;
    }
    LMLOG(LDBG_1, "Sending %s", lisp_msg_hdr_to_char(mrep));
    send_msg(&xtr->super, mrep, uc);
    pkt_pull_ip(mrep);
    pkt_pull_udp(mrep);

    /* Additional Messages to the loc-server*/
    uconn_t nuc = *uc;
    nuc.ra = *xtr->loc_server;
    if (xtr->signal_dropping) {
        /* Send Map-Reply(NULL) to loc-server. */
        lisp_msg_destroy(mrep);
        mrep = lisp_msg_create(LISP_MAP_REPLY);
        lisp_msg_put_neg_mapping(mrep, deid, 0, ACT_NO_ACTION,
A_NO_AUTHORITATIVE);
        mrep_hdr = lisp_msg_hdr(mrep);
        MREP_RLOC_PROBE(mrep_hdr) = MREQ_RLOC_PROBE(mreq_hdr);
        MREP_NONCE(mrep_hdr) = MREQ_NONCE(mreq_hdr);
    } else {
        /* Send same Map-Reply to loc-server as sent to PxTR. */
    }
    LMLOG(LDBG_1, "Sending %s; map reply to loc-server",
lisp_msg_hdr_to_char(mrep));
    send_msg(&xtr->super, mrep, &nuc);
  ….
  ….
```

Figure 5.2-4 Part of Code To Send Additional Map-Reply Message to the Loc-server

prepare itself to buffer packets before handover starts by sending map-reply(Null) to it. Hence:

1. We modify an existing function, *tr_recv_map_request*, that delivers map-reply to the PxTR, to also send a second map-reply to the Loc-server.
2. When signal_dropping is TRUE, the map-reply to the PxTR should contain the address of the Loc-server, and the map-reply to the Loc-server should contain no mapping.

Figure 5.2-4 is part of the *tr_recv_map_request* function in *lispd/control/lispd_xtr.c* where the map-reply message to the Loc-server is sent. The complete LISPmob code as amended can be accessed from the project site [117].

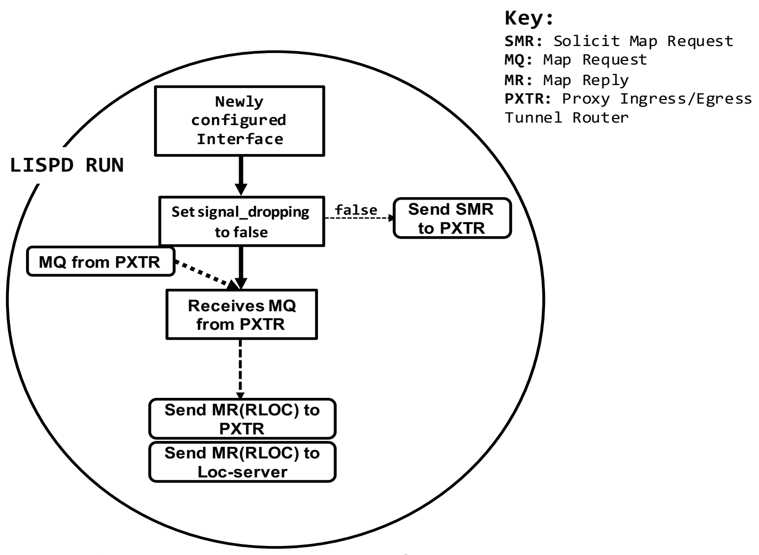


Figure 5.2-5 Post -Handover Handler Operation

5.2.4.2 Post-Handover Handler Operation

Shortly after the pre-HaH, the node's interface goes down, and the xtr object will naturally seek a new local connection, and potentially a new RLOC. When these have been established, it will invoke *send_all_smr_cb*, which sets the Boolean xtr->signal_dropping (which was changed to TRUE prior to handover) back to FALSE, and issues a new SMR to PxTR. Once again, PxTR responds with map-request, which invokes *tr_recv_map_request*, which this time sees that the flag is false, so it must:

1. send a map-reply to PxTR mapping the MN’s EID to its new RLOC (the original behaviour), and
2. send a map-reply to the Loc-server, with essentially the same mapping (an additional behaviour).

The described operation is depicted in Figure 5.2-5. It results in the completion of the handover process, with the intervening packets buffered in the Loc-server additionally being delivered to the MN.

5.3 Loc-server Implementation

As discussed in chapter four, three different modules are needed on the Loc-server, although only two are vital for the server's operation. These include control, data and database modules with the database module being optional since the server could be implemented to provide the location area service with no restriction to any prefix, which is what the database module is implemented to do – provide service to only the prefixes/IP addresses in the database.

Loc-server is designed to sit idle listening to requests for buffering from MNs and subsequently forwarding the buffered packets accordingly, and as such control and data sockets need to be implemented to achieve this functionality. Since we chose LISP-MN for our project and the protocol uses UDP for both control and data packets, the Loc-server necessarily needs to be listening on select UDP ports for the two different types of message. We adapt LISP's 4342 control and 4341 data ports for this listening purpose. The complete Loc-server code can also be accessed from the project site [118] and is also available as Appendix.

5.3.1 Event Reactor

For the server to work in the way described above, we opted to write a single-threaded C program whose behaviour is triggered by diverse external events that can occur in any order. One way of achieving this without resorting to multi-threading is to use a reactor design pattern that enables handling service requests that are delivered concurrently to a service handler by one or more inputs as described by Schmidt [119]. There are several reactor libraries

```
….
….
struct server;

/* A destination represents an IPv6 peer who has registered with us to
   store its packets. */
struct dest {
  struct server *srv; /* which server we belong to (although there's
                         only one) */
  struct dest *next; /* the next destination in the list held by the
                        server */
  struct in6_addr id; // the identifier of this destination
  struct in6_addr careof; // where to forward to, or zero if not set
  struct buffer *buffers; /* the first of buffer in a list of received
                             packets*/
  struct buffer **tail;
  react_event expiry_event; /* an event for detecting if this
                               destination is still needed after a while
*/
  struct timeval expiry; // when to stop needing this destination
};

/* A server listens for data packets and control packets, and keeps a
   list of destinations that have registered with it to store their
   packets. */
struct server {
  react_core core; // the reactor to use
  int ctrl_sock; // the control socket
  react_event ctrl_event; // the event for receiving on the control
  // socket
  int data_sock; // the data socket
  react_event data_event; // the event for receiving on the data
  // socket
  struct dest *dests; /* a list of destinations we are holding packets
                         for*/
};
….
….
```
Figure 5.3-1 Core Data Structures for Loc-server Implementation

available for use such as GTK and QT, and the ACE framework. However, we found the Portable C event reactor [120] to be ideal for our purpose since it provides a standard interface suitable for all libraries that do blocking I/O. The reactor enables the creation of a core object for the program and event handler(s) to manage what action to take when control or data packets are received on the specified ports.

5.3.2 Loc-server State

For the server's operation, some core structures are necessarily declared to enable the node operations as described in chapter four. A snippet of these declarations is presented in Figure 5.3-1.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Type=2 |P|E|S|          Reserved               | Record Count  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         Nonce . . .                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         . . . Nonce                           |
+-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                          Record TTL                           |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
R   | Locator Count | EID mask-len  | ACT |A|      Reserved         |
e   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
c   | Rsvd  |  Map-Version Number   |       EID-Prefix-AFI          |
o   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
r   |                          EID-Prefix                           |
d   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  /|    Priority   |    Weight     |  M Priority   |   M Weight    |
| L +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| o |        Unused Flags     |L|p|R|           Loc-AFI             |
| c +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  \|                             Locator                           |
+-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Figure 5.3-2 Map-Reply Message Format (Source [61])

The *dest* structure contains *id*, *care-of,* and *buffers* structures to enable the server to store the EID, RLOC and packets destined to a particular node. The *server* structure, on the other hand, uses the different functions in the reactor library to enable the control and data sockets, and to prime the system to receive the relevant packets on the two sockets as well as the action to take as a consequence.

5.3.3 Control Module (CM) Implementation

When a UDP message is received on port 4342 (the lisp-control packet), the reactor calls a function *on_ctrl_packet* in the program in which the control functionality is implemented. We use this function to determine what type of message is received on the designated port. LISP has defined different message formats which are identified by numbers in the ‘type’ field of the packet header. The map-reply packet, which is of interest to us, is identified with the number 2 as can be seen in the packet header in Figure 5.3-2 above.

Hence the first thing that the CM inspect for packets arriving on port 4342 is the message type and the packet is ignored if it is not 2. Similarly, a ‘Record Count’ of less than or equal to zero will also cause the message to be ignored as it signifies that the packet carries no EID, which is vital for potentially buffering or sending packets destined to the EID.

This functionality is implemented using the *decode_map_reply* function with its flow of execution shown in Figure 5.3-3 below. As our implementation focuses solely on IPv6, we also check the EID-Prefix-AFI, with the value of 2 representing IPv6 being the only accepted value, otherwise, the packet is also ignored. We subsequently check for the ‘Locator Count’ value, with 0 indicating a map-reply(Null) is received or a value of 1 to indicate a map-reply(RLOC) is received. A value of 2 for ‘Locator Count’ also indicates a map-reply(RLOC) with more than one locator. This could happen when an MN moves from a non-priority cellular network to Wi-Fi as the former is brought up and configured before the actual switch of communication sessions to the new interface.

Since we don’t envisage an MN with more than two wireless interfaces, a count of more than two means the packet is somehow corrupted and should be ignored. If the message is found to be a map-reply(Null), we check the cache of stored destinations to see if the EID already exists in which case its expiry timer is reset to 10 seconds otherwise the destination is created, and an expiry timer of 10 seconds is as well set against it. However, if the message is map-reply(RLOC), we pass control to the DM to find the corresponding EID from the cache and forward any buffered packets to the locator that we just learned from the map-reply.

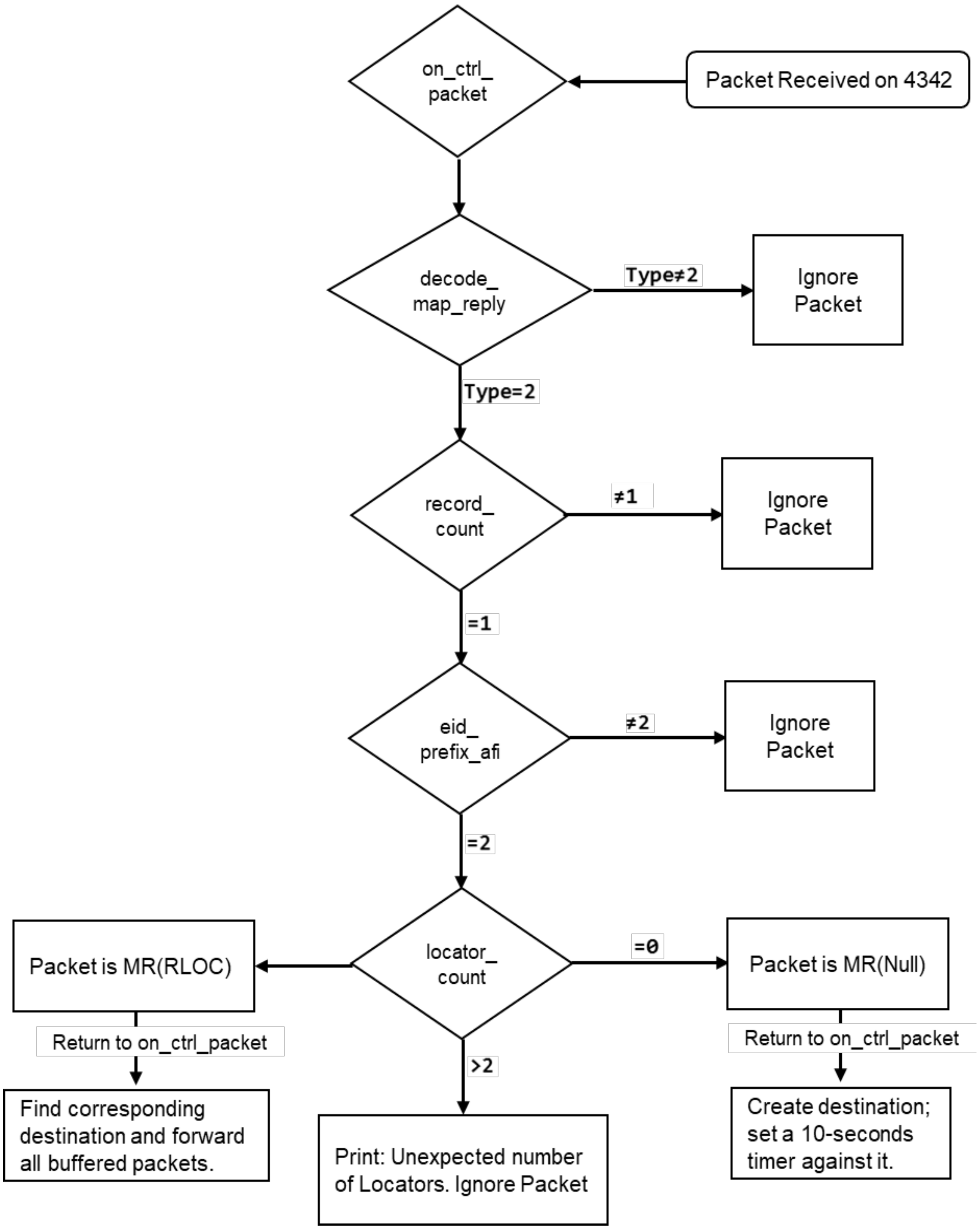


Figure 5.3-3 Decoding a Map-Reply Message

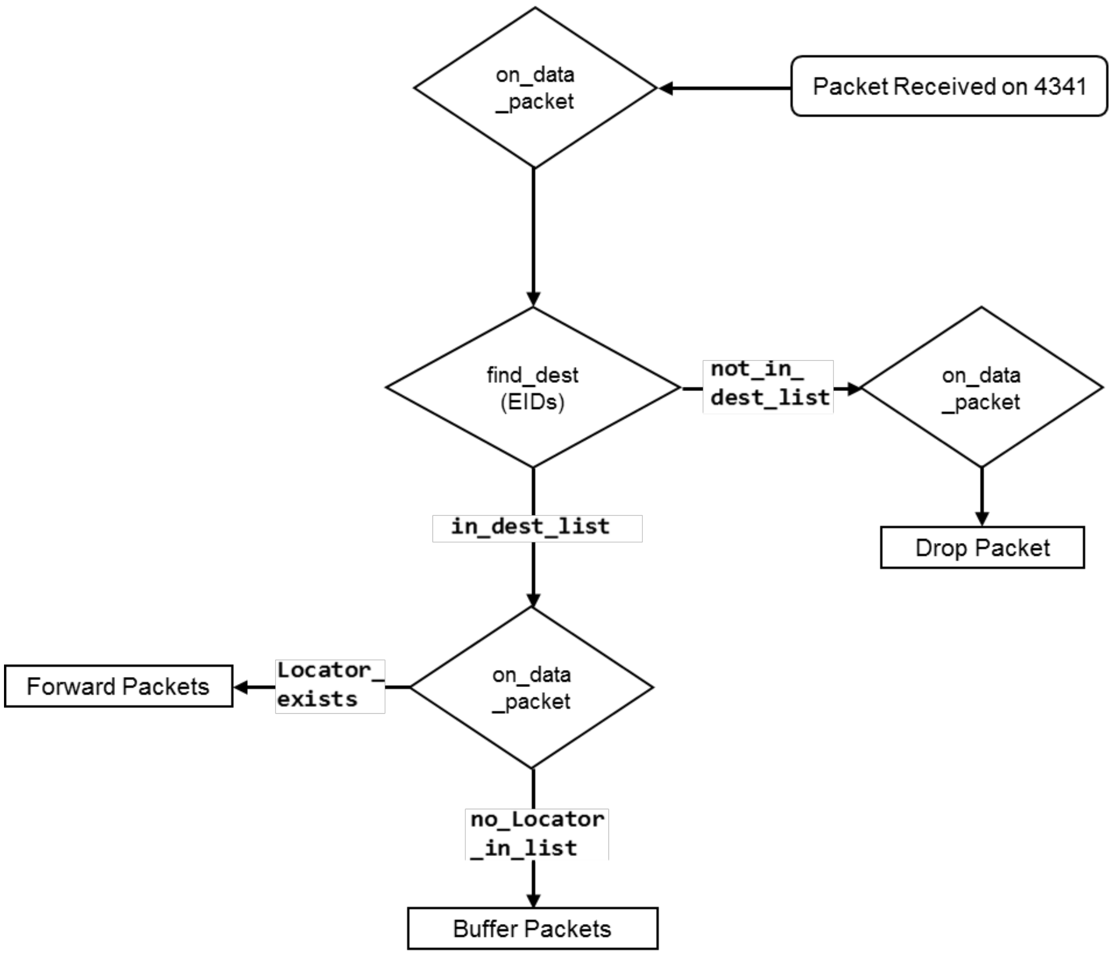


Figure 5.3-4 Data Module Packet Inspection and Actions

5.3.4 Data Module (DM) Implementation

For packets arriving on port 4341 (lisp-data), the reactor calls the *on_data_packet* function, which first of all determines the destination address from the lisp packet header. We then perform one of the following three options depicted in Figure 5.3-4:

1. If the EID is not in the saved destinations, the packet is dropped, and no further action is performed.
2. If the EID is in the saved destinations, then we check for whether a locator exists for the EID, if it does, we forward the packets to the locator address.

In 2 above, if no locator is available for the EID, we buffer the packets received in the name of the EID and wait for a signal, map-reply(RLOC), for 10 seconds from the CM. The packets are dropped if no map-reply(RLOC) for the EID is received after the timer. When the map-reply(RLOC) arrives within the timeout period, the packets are sent to the MN in the order in which they arrived. The lifetime for RLOC is 10 seconds, after which it is deleted together with the corresponding EID.

5.4 Chapter Summary

This chapter described how the newly designed system described in chapter four was implemented on an IPv6 network. The chapter started with the discussion on the LISP-MN implementation used for the work, the LISPmob, and why we chose LISPmob in place of other implementations. We also explained our reasons for choosing LISP-MN to test the new architecture amongst the other LOC/ID protocols. We have also described umip code, as the MIPv6 implementation used on the network.

We also discussed low-level implementation details on how the different modules on the MN and Loc-server described in chapter four were realised. These include how the movement of the MN is emulated on the testbed, how the Loc-server is introduced into *lispd* execution, and how HITr and HaH are realised on the MN. We described the aspects of LISPmob code that were changed to achieve this functionality on the MN. For the Loc-server, we discussed the implementation of ‘event reactor’ which enables the creation of a core object for the Loc-server program and an event handler(s) to manage what action to take when control or data packets are received on the specified ports.

We also provided the implementation detail of control and data modules of the server.

CHAPTER SIX

EVALUATION

In this chapter, we provide a thorough performance evaluation of the newly built architecture. We will be running experiments on a testbed mimicking real conditions through emulated network characteristics and real applications. Whilst running experiments on a real-life network could also be valuable, we limited our work to a testbed because the available IPv6 Internet is tunnelled through IPv4 networks. An additional LISP overlay (yet another tunnel) on top might not give us the actual performance due to the effect that the many layers of tunnelling might have on performance. The experiments might also fail due to link constraints or due to tampering or blocking of the packets by middleboxes. We also want to isolate some network issues and policies that may cause the results of our experiment to not reflect on the actual performance of the new system; network congestion and firewall restrictions are some examples. We also did not consider simulation as testbed experiments are much closer to, and easier to transfer into, reality.

We intend to run the experiments by testing the system in both homogeneous and heterogeneous wireless environments and using different link speeds in some instances. MNs move between networks of similar technology using the same wireless interface; they could also move between links of different technologies, the most common of which is Wi-Fi to cellular or vice versa. We discussed different mobility scenarios of HeWE and HoWE in section 3.1.

6.1 Evaluation Objectives (EOs)

As the IETF standardised host mobility protocol, MIPv6 serves as the benchmark for evaluating other mobility protocols designed to perform similar mobility management functions to the protocol. Hence, the evaluation work begins with a theoretical analysis of LISP-MN against MIPv6 before running the two protocols on the testbed. We perform a qualitative analysis of some mobility features of the protocols before running the two on the testbed for quantitative measurements. The quantitative analysis first looks at TCP and UDP performance before evaluating video applications running on top of the protocols. Based on the findings from this preliminary investigation, we go on to build the new architecture, with the Loc-server at its centre, to address some of the observed limitations of LISP-MN and by extension many LOC/ID protocols.

After building the new system, we again look at the mobility features earlier analysed to understand if there is any change from the original protocol's behaviour in the improved architecture. We then run experiments again on the testbed, focusing on these mobility-related metrics as well as video application performance. In summary, we analyse the qualitative as well as quantitative differences between vanilla LISP-MN architecture and the improved architecture. All the results presented in this chapter are averages of ten experimental runs.

The evaluation is set to answer the performance questions of the LOC/ID protocols as well as the newly developed architecture listed below. These serve as the evaluation objectives (EOs) that this chapter wants to address. Answering the questions will enable us to evaluate the overall improvement to the vanilla LISP.

EO1: In terms of design decisions, what are the differences between LISP-MN and MIPv6, and how do these decisions affect the performance of the protocols?

EO2: How much does LISP-MN's performance differ from the standardised MIPv6 in terms of handover delay, service disruption time, throughput and packet loss?

EO3: How do applications perform on top of the two protocols?

EO4: How do the design decisions taken in developing the new LISP-MN architecture affect the performance of the protocol?

EO5: What are the performance improvements enabled by the improved LISP-MN architecture in terms of the metrics analysed in EO2, in addition to TCP retransmission rate; What is the impact of the control plane overhead introduced by the Loc-server?

EO6: Does the newly developed architecture improve the performance of applications?

EO7: Does the new architecture improve network utilisation; does the improvement differ between networks of different link speeds?

6.2 Development Testbed

This evaluation is carried out on the testbed shown in Figure 6.2-1 below to enable us to answer the questions raised in the previous section. As stated earlier, the testbed consists of nine desktops PCs configured to work as an MN, two ARs, one map-server, one PXTR/HA, a Loc-server, a CN and two backbone routers. The relevant programs running on the different machines to enable our experiments are listed in Table 6.2-1 below.

Composition of the Devices

The MN has two IEEE 802.11bgn wireless interfaces emulating a Wi-Fi and a cellular interface with AR1 as the Wi-Fi AN and AR2 as the cellular. AR2 also serves as a second Wi-Fi network for experiments focusing on homogenous wireless networks only. The ARs run the Linux hostapd program [121] to provide a software-based access point service to the MN. The RADVD program [122] running in the background sends router advertisements, and responds to router solicitations from the MN to enable the MN to configure its IP address and obtain the necessary routing information. The two BRs serve as the Internet cloud connecting the different network nodes together with 100ms delay introduced

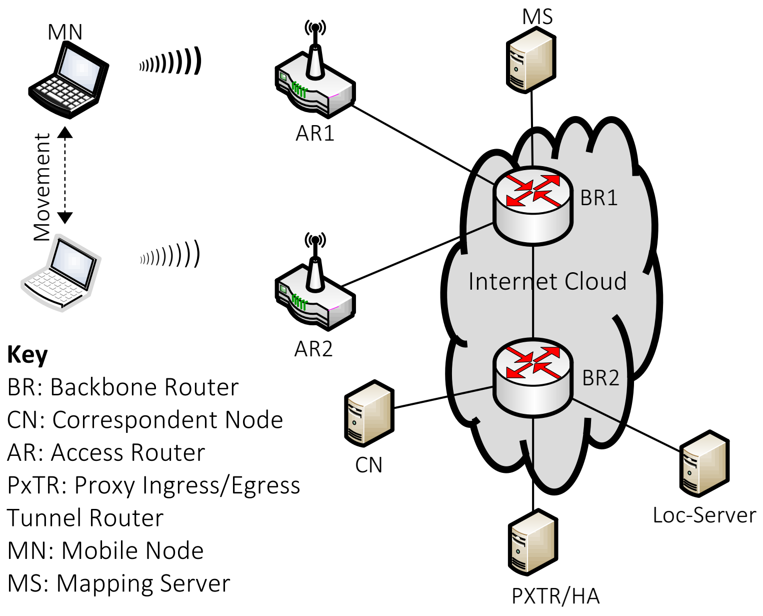


Figure 6.2-1 Development Testbed Showing the Different Components of the Systems

System	Programs running
MN	LISPmob, wpa_supplicant, iperf, DASH players, umip code, client-server download program
AR	Radvd, hostapd, RIPngV2
map-server	LISPmob, RIPngV2
PxTR/HA	LISPmob, umip code, RIPngV2
Loc-server	Loc-server code, RIPngV2
CN	iperf, apache server, client-server download program, RIPngV2

Table 6.2-1 Programs running on the Testbed

in both directions to emulate the delay on the Internet[29]. The CN is the server that provides different online services to the MN as it moves, while the map-server and the PxTR are the two necessary components for LISP-MN's protocol operation. The HA is collocated with the PXTR to enable MIPv6 experiments. The Loc-server as earlier explained is to provide support for the LISP-capable MN during the handover.

[29] Introducing this delay helps us to see the effect of some mobility events, especially soft handover.

6.3 Benchmarking – LISP-MN vs. MIPv6

In the following subsections, we provide a qualitative and quantitative comparison of the two protocols to contribute to the argument of LISP-MN's possible adoption as the mobility protocol of choice. We also provide the performance of video applications on top of the two mobility protocols. The work presented in sections 6.3.1 and 6.3.2 has been presented as a conference paper in [67] but with some modifications on the result here.

FEATURE	LISP-MN	MIPv6
Registration	1. Five control messages: map-register, map-notify with map-server; SMR, map-request, and map-reply with PITR 2. Can potentially be reduced 2 messages.	1. Two control messages: BU and B_Ack with HA 2. Needs the two messages for the handover, although communication can resume after the first
Tunnelling	1. Add 56 bytes to an IPv6 packet by using UDP encapsulation 2. IP version agnostic: capable of IPv4-in-IPv6/IPv6-in-IPv4 tunnelling	1. Uses IP-in-IP tunnelling and adds only 40 bytes IPv6 header. 2. Not IP version agnostic.
Security	1. Uses pair-wise key to authenticate MN<->map-server communication 2. PITR only accepts map-reply carrying the nonce in its map-request 3. Security native to the protocol	1. Uses IPsec as add-on to secure MN<->HA messages. 2. Uses BAD option to secure binding exchanges with CN. 3. Uses nonce for direct communication with CN
Routing	1. Packets sent via the PETR, replies via the PITR. The two nodes usually collocated. 2. Can use more than one proxy tunnel router 3. Can achieve direct communication between MN and remote node but tunnelling is always involved.	1. Packets sent and received via the HA 2. Can also use multiple HAs 3. Route optimisation allows MN<->CN communication without tunnelling.

Table 6.3-1 Analytical Comparison of LISP-MN and MIPv6

6.3.1 Qualitative Analysis

To address the first objective (EO1) of the evaluation, we provide more insight into the operation of the protocols by comparing the two based on features we believe define the quality of a mobility protocol. These include registration, tunnelling, security, and routing operations. A summary of how the two mobility mechanisms compare on the aforementioned features is provided in Table 6.3-1 above.

6.3.1.1 Registration

For the MN's registration on receiving a new RLOC (on coming online or post-handover), a LISP-based MN requires five different control messages to complete its registration with the map-server (two messages) and the PXTR (three). In contrast, an MN running MIPv6 requires only two messages for the same purpose. The need for five messages on LISP-MN is because the MN communicates with two remote components to update its EID-RLOC mapping. It communicates with the map-server using map-register and map-notify messages, and informs the PITR of its new location using solicit-map-request, encapsulated-map-request, and map-reply.

It is important to note that the messages with the PITR are only sent when the MN is in the middle of a communication session. Hence when an MN receives a new RLOC with no ongoing data session only the messages with the map-server are exchanged, and the other three are sent when a session is to be established. The MN can also set the Proxy-Map-Reply flag (P bit) to 1 and Want-Map-Notify (M bit) to 0 in its map-register message. These two flags are to (1) request the map-server to respond to map-requests on its behalf with the P bit and (2) set no map-notify message from the map-server with the M bit when the MN sends map-register; thereby reducing the number of control messages during a handover to two (map-register to map-server and SMR to PXTR). PXTR will send map-request to, and receives map-reply from, the MN

Similarly, MIPv6 also has a feature to enable optimistic handover, which allows the MN to resume communication with the HA even before it receives a B_Ack. In a high mobility environment with many mobile nodes, MIPv6 may show better performance as its handover processes will be less processor-intensive for the APs on whose link an MN is moving to. And with LISP-based MNs sending 60% more control messages (when map-notify and proxy-reply bits are unset and set respectively), a device’s battery life is likely to drain faster in comparison to MIPv6. Even where replies are handled by the map-server and map-notify is disabled, there are more messages in the backbone with LISP-MN, in comparison to MIPv6. Moreover, the LISP-MN architecture requires the implementation of two remote nodes to enable an MN to communicate, in contrast to MIPv6’s one, the HA.

6.3.1.2 Tunnelling

LISP-MN uses UDP for tunnelling LISP data packets on port 4341 and sending LISP control packets on 4342 creating an overlay on the Internet. MIPv6, on the other hand, uses IP-in-IP tunnels for data packets with the payload carried by the inner IP header. Hence a LISP packet adds up to 56 bytes to an IPv6 packet and therefore a higher cost of processing for intermediate devices as compared to MIPv6’s 40-bytes addition. LISP-MN is IP version agnostic and capable of IPv4-in-IPv6/IPv6-in-IPv4 tunnelling, a feature not available with MIPv6. This feature potentially can speed up the adoption of an IPv6 network with LISP, and simplify its interoperability with IPv4.

6.3.1.3 Security

LISP-MN specified the use of pre-shared authentication keys for map-register/notify messages with the map-server. The MN must include authentication data, which is a hash of the message (the payload) using the pair-wise shared key as specified in [123]. Similarly, the additional control messages of LISP-MN improves its security as the PITR will only forward packets after learning of the MN’s new location from a map-reply carrying a similar nonce value as the one in the PITR’s original map-request message to either the MN or the CN. MIPv6 also specifies the use of IPsec SAs or a BAD option for binding messages between the MN and HA to ensure the connectionless integrity and data origin authentication of the IP packets. The protocol also specifies the use of the BAD option for route optimisation with a CN, as well as the use of a nonce for subsequent data exchange with the MN. LISP data messages are also secured using a nonce for communication with a LISP-capable CN.

The difference between the two protocols’ security mechanisms is that LISP-MN’s security is native to it and the protocol cannot work without the minimal security being implemented. Although MIPv6 specification [12] described the use of IPsec to secure binding messages as a “MUST,” the security mechanism is an add-on, and MIPv6 can work perfectly without it – this point is as far as the available implementations of the two protocols are concerned. To optimise routing between an MN and a CN however, it is essential that a return-routability procedure is performed using the BAD option to ensure secure binding is achieved.

Invariably, packet encapsulation reduces Internet security as discussed in [124], since the encapsulation makes it difficult for intermediate routers to filter packets based on the information in the inner-header fields.

6.3.1.4 Routing

While LISP-MN uses the map-server as its anchor point, the actual data delivery is performed by PETR for outbound data and PITR for data destined to the MN. But these will usually be collocated as earlier stated forming a triangular routing in the two LISP-MN communication scenarios under review. This is similar to the MN's triangular routing via the HA in MIPv6. While MIPv6 specified route optimisation for direct communication with the CN, a LISP-based MN in a LISP site may communicate with another MN within or outside its domain directly by encapsulating the packet using the remote MN's local locator (LLOC – within its domain) or RLOC (outside). But in any case, tunnelling is involved. It is pertinent to mention that because LISP is designed to work with the legacy Internet and deployed incrementally, quadrangle routing is necessary for some mobility scenarios to ensure packet delivery.

Furthermore, both protocols allow the use of more than one PXTR and HA accordingly for packets delivery. The LISP-MN architecture increases path-stretch of packets (packets traverses more nodes to get to their destination) in comparison to MIPv6.

6.3.1.5 Discussion

The discussions in section 6.3.1 address EO1 of the seven EOs itemised in section 6.1: *In terms of design decisions, what are the differences between*

LISP-MN and MIPv6, and how do these decisions affect the performance of the protocols?

We have highlighted the differences between the two protocols in terms of registration, tunnelling, security, and routing. In comparison to MIPv6, LISP-MN will use up more resources on the network due to having more control messages. However, the higher number of control messages is meant to improve its security and reachability as an MN moves across the Internet. LISP-MN tunnelling using UDP creates an overlay network allowing its IP version agnosticism feature, which is vital for interoperability between the two different versions of IP. This, of course, is at the expense of longer packets when compared with MIPv6 IP-in-IP encapsulation. While MIPv6 has a mechanism for avoiding triangular routing, LISP-MN by design involves the tunnelling of packets via a network – for scenarios in questions – node whenever inter-domain communication is involved.

6.3.2 Quantitative Analysis

6.3.2.1 Mobility Scenarios and Evaluation Metrics

The real-life heterogeneous mobility scenario as highlighted earlier in the thesis is that of a user moving with a cellular and Wi-Fi capable mobile device. While at home, the user connects to his/her home Wi-Fi streaming a video or making a call using VoIP. As soon as the user steps out and moves away from the Wi-Fi, the device connects to the cellular network of their provider. Similarly, on returning home, the phone disconnects from the cellular network and connects to the higher priority home Wi-Fi. This type of mobility is also found with subscribers of Wi-Fi hotspots and users of public networks as they come into and leave the hotspot/public network area – disconnecting from cellular on coming into the areas and reconnecting on leaving. For homogeneous networks, we are looking at a scenario where a user walks across an area with open independent Wi-Fi networks, such as a train station, shopping mall, etc., connecting and disconnecting from APs as they move.

For performance measurement, we look at the handover delay and SDT experienced by the protocols and the consequent packet loss of UDP datagrams during the handover event. The time between the start of communication and when a handover occurs is randomly set and the activity during and after the handover is the focus of the evaluation. We also look at the TCP throughput achieved by both.

For UDP experiments, we used the Linux *iperf* program to emulate skype VoIP traffic with a packet size of 300 bytes to generate 64 Kbps flows as performed in the works of [80, 125, 126]. We have also implemented a simple client-server program that allows the MN to download a file from the server (CN in Figure 6.2-1) over TCP, and we use the traffic traces from this connection to measure the TCP throughput and the SDT. There is a 100ms delay introduced, using the Linux *netem* program, on the backhaul link between BR1 and BR2. *netem* is also used to regulate the wireless link to a download speed of about 2.3Mbps. The regulation of the download speed is to test the protocols in a relatively average speed network.

6.3.2.2 Handover Delay

In this section, we analyse the different elements that are involved in a handover scenario to understand the contribution of each entity to the delay involved in a handover event. Handover delay is the time during which an MN cannot exchange packets with its CNs. There is actually no handover delay for the two protocols' soft handover events in HeWE, and no service disruption was experienced. This happens because the data session is switched over to the new interface after it is fully configured. It is a different case in a hard handover event and for this reason, we will focus our analysis on this event. We will also discuss a handover event by a single interface MN in HoWE.

For HeWE, we define handover delay as the time between the higher priority interface going down (when all activities of the interface stop) until the last handover message is sent (map_reply from the MN, in the case of LISP-MN) or received (B_Ack from the HA, in the case of MIPv6) by the second interface. SDT, on the other hand, is measured from the time we receive the last packet on the first link to the time we receive the first packet on the second link. For clarity of understanding, we define three different delay variables D1, D2 and D3 for HeWE as follows.

D1: From the current interface being down until first handover message is sent using the newly configured interface – i.e., map_register for LISP-MN and BU for MIPv6.

D2: From the first handover message until the last handover message – map_reply for LISP-MN and B_Ack for MIPv6.

D3: From the last handover message until the data session resumes.

The addition of D1 and D2 produces the handover delay, and the addition of the three delay periods produces the SDT.

For HoWE, we define handover delay as the time between disconnecting from the current link until the last handover message is sent/received on the new link.

SDT remains the time between the last data packet before handover until the first packet after. D2 and D3 are the same as defined earlier while D1 is measured from disconnection of the current link until the first handover message is sent using the new link.

We found the average handover delay of a hard handover event in HeWE and MN's handover event in HoWE to be almost the same. Hence, the result presented in Table 6.3.2 is that of HeWE, as a representative of the handover delay in the two wireless environments.

There are about three seconds of delay before the first handover message is sent in the two wireless environments. The long delay involves the necessary L2 verification messages using the EAPoL protocol [127] as well as layer three address configuration processes. These processes as discussed in chapter three include MD (i.e., sending RSlt and receiving RAdv from the AR), CoA configuration and DAD, bringing the handover delay for the two protocols to a little more than 3s. We can also see from Table 6.3-2 that the D2 of the two protocols is almost the same despite the fact that LISP-MN sends five messages during the handover as against the MIPv6's two, signifying that the former's control messages have very minimal impact on the handover delay. D3 is slightly higher for LISP-MN as compared to MIPv6 resulting in a longer SDT for the former as will be discussed in the following subsection.

Protocol	**Seconds**				
	D1	**D2**	**Handover delay**	**D3**	**SDT**
LISP-MN	3.06	0.205	3.27	2.61	5.88
MIPv6	3.01	0.204	3.21	2.32	5.53

Table 6.3-2 Hard Handover Delay for an MN in a Heterogeneous Wireless Environment.

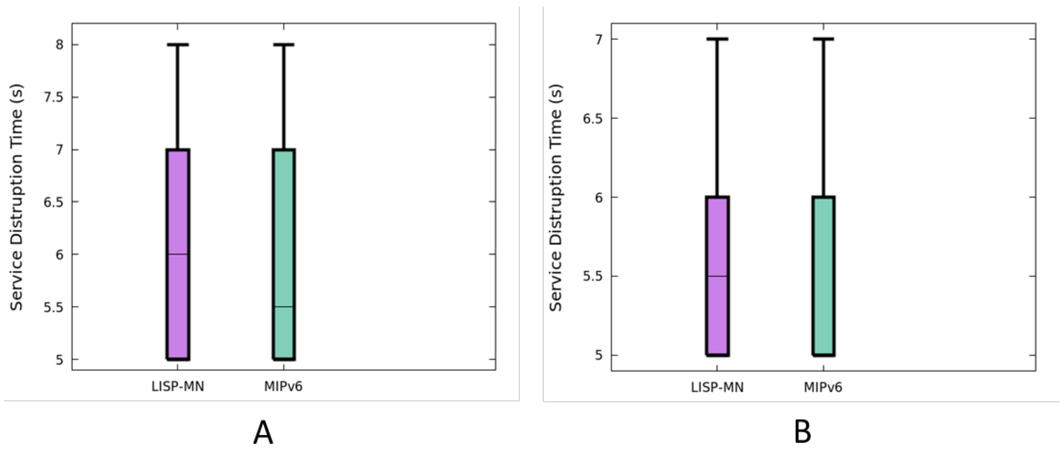


Figure 6.3-1 Service Disruption Time for LISP-MN and MIPv6 in Heterogeneous (A) and Homogeneous (B) Wireless Environments

6.3.2.3 TCP Service Disruption Time (SDT)

Figure 6.3-1(A & B) shows the SDT recorded in the HeWE and HoWE respectively. The mean[30] SDT for LISP-MN in HeWE is 6s and 5.5s for MIPv6, which indicates a slightly better performance for the latter. For HoWE, on the other hand, both protocols recorded a mean SDT of about 6s. Hence we expect to see a similar performance by the two protocols in HoWE. The results show more than 2s difference between the SDT and the handover delay experienced in LISP-MN and MIPv6. This is because, during the course of the handover, the CN stops sending TCP packets to the MN after a period of no acknowledgement.

The additional seconds experienced after the handover is the time it takes the transport layer of the MN to realise the availability of the new connection and to start sending acknowledgements to the CN for communication to resume. And since the SDT varies for the ten runs as we can see from the box plots in Figure

[30] The box plots also show the minimum value (bottom of the candlestick), first quartile, third quartile and the maximum values.

6.3-1, we cannot rule out the behaviour of the server program contributing to the SDT by varying the response to the MN’s acknowledgement packets.

The long SDTs in both protocols will not only affect delay and loss sensitive applications but also ‘best effort’ type applications such as Internet browsing and some mobile apps. Users may experience slow loading of webpages and apps’ response during and after the handover. This may also affect the network performance when many MNs keep sending TCP retransmission requests over the network after a handover event is completed.

6.3.2.4 TCP Throughput

Since the two protocols are run over the same network infrastructure, similar throughput performance is expected as can be seen in Figure 6.3-2. The SDT, as earlier discussed, is similar for the hard handover of HeWE as well as the handover event in HoWE. The most remarkable difference is in the soft handover event of the HeWE (Figure 6.3-2A). With LISP-MN, the two interfaces are both active during the handover, and the control messages are exchanged using the old link before transferring the data session to the new AN. With the MN receiving data on both links for a number of seconds, the throughput of the download increased by 35% for the period just before the newly configured link is subsequently used exclusively.

It is a different case with MIPv6 where the protocol uses the newly configured interface to exchange the handover messages before switching all communications to the newly configured interface. The abrupt switch of interface causes a slowdown in the packet delivery, and some packets in transit

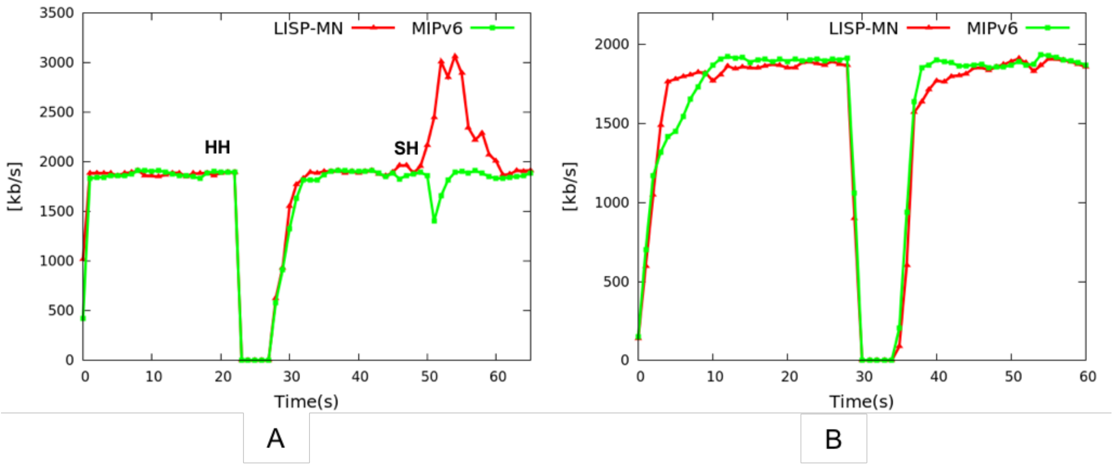


Figure 6.3-2 TCP Throughput Performance for LISP-MN and MIPV6 in Heterogeneous (A) and Homogeneous (B) Wireless Environments

are dropped by the previous access router, hence the little dip in MIPv6's throughput as seen on the figure (6.3.2A).

6.3.2.5 UDP Packet Loss

In a client-server interaction, all packets destined to the client (MN in this case) are dropped at the point of handover by previous ARs unless a policy exists to direct the routers regarding what to do with such packets after the MN has moved. These dropped packets are counted as lost packets and are directly proportional to the handover delay. We measured the loss by sending UDP streams from a CN to the MN and initiating a handover event on the MN during that period. The amount of loss recorded for a hard handover of HeWE and handover in HoWE is almost the same because of the similar handover delay recorded. Figure 6.3-3 shows the performance of LISP-MN and MIPv6 in HoWE as representative of the two wireless environments.

As we can see from the figure, the percentage of lost packets is about 6.7 for LISP-MN and 6.5 for MIPv6. The exact amount of datagrams respectively lost

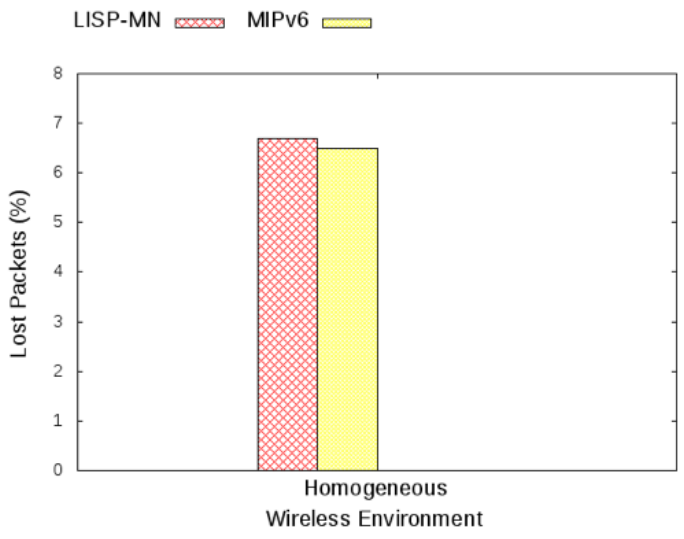


Figure 6.3-3 Packet Loss in Homogeneous Wireless Environment

in the four handover seconds is 368 (6.65%) and 360 (6.50%) of the 5,530 datagrams sent over the period of one minute. Unlike TCP traffic discussed in the previous section, UDP communication resumes as soon as the MN connects to the new link since the CN did not stop sending packets while the handover was taking place.

LISP-MN experiences no loss in the soft handover of HeWE as the two interfaces both stay active for a period before the newly configured interface take full control. Similarly, with MIPv6, a loss of very few datagrams was recorded in the region of 0.1% as communication is switched to the newly configured interface as soon as it is configured. The SDT recorded for UDP traffic is about 4s for the two protocols explaining the reason why there is no significant difference between the protocols in the amount of packets lost during the experiments.

6.3.2.6 Discussion

Section 6.3.2 addressed EO2: *How much does LISP-MN's performance differs from the standardised MIPv6 in terms of handover delay, service disruption time, throughput and packet loss?*

For the four performance metrics investigated, the two protocols demonstrated similar performance for most of the experiments. Despite LISP-MN's five control messages against MIPv6's two, similar handover delay and SDT are recorded for the two protocols. Hence the extra messages have minimal impact on the MN's handover performance. The only significant difference between the two protocols is in the soft handover event as LISP-MN uses two active interfaces for a period before switching all communication to the targeted link. This is in contrast to MIPv6 which move the current data sessions as soon as the new interface is ready for use. LISP-MN's soft handover actually improves throughput performance, whereas MIPv6's recorded a slight decrease in throughput level for the same event.

The amount of loss recorded for hard handover in HeWE and handover in HoWE is quite significant. As described in the works of [42, 43] and [44], this amount of loss will affect voice, video, and HTTP2 traffic performance, and users might experience a break in voice calls and video playback, as well as slow loading of webpages. LISP-MN recorded no loss during soft handover in HeWE, and MIPv6's record of 0.1% loss is so insignificant and will not affect on even the loss-sensitive applications mentioned earlier - voice, video or HTTP2 traffic.

6.3.3 Video Applications

To understand the impact that mobility with these two protocols has on application layer programs, we elected to test video applications because of its high demand for network resources and its sensitivity to changes in the network. Furthermore, mobile video traffic is responsible for 60% of video traffic on the Internet in 2016 according to Cisco [6] making it the most popular traffic generated by mobile devices. The most recent video streaming technology is the Dynamic Adaptive Streaming over HTTP (DASH), which is an approach that ensures the quality of video each user device receives is commensurate to its network context – throughput level, available video quality, etc. The adaptation logic on the MN is designed to download divided video chunks either based on the current throughput [128]; or based on the current buffer occupancy [129, 130]; or based on these two factors.

For our evaluation, we chose to run three different video players based on each of the aforementioned design techniques. DASH players that solely rely on throughput estimation are called throughput-based players, and we chose the open source version of Microsoft smooth streamer (MSS) [131] as a representative of players in this category. For players that rely on buffer occupancy (buffer-based players), we chose the first purely buffer-based player proposed by Huang et al. [129]. And for players that use the two parameters (mixed-mode players), we chose a player proposed by Miller et al. [132].

We set up a web-server using Apache 2.4.17 to host the open source 'Big Buck Bunny' video dataset from [133]. It consists of different quality levels, ranging from 50 to 3500 Kbps, which are available for an MN to download from regardless of the player it is using. To ensure that there is sufficient capacity to sustain the download of the highest video rate throughout the experimentation, we limit the maximum downstream available bandwidth to 4 Mbps. Additionally, all the chunks have a segment length of two seconds, and all players were implemented in Python.

6.3.3.1 Mobility Scenario and Evaluation Metrics

Since a hard handover in HeWE and handover in HoWE have very similar performance levels, we evaluate the handovers of a single interface MN – in HoWE only. We assume a scenario where a user walks across distinct ANs and makes six handovers while streaming video for three minutes. We use the following metrics to evaluate the three players' performance as metrics that are known to affect user's quality of experience [134, 135]:

- Rebuffers is the total number of video freeze per streaming session.
- Average video rate is measured in Kbps, is calculated as $\frac{t_1 q_1 + t_2 q_2 \ldots t_n q_n}{t_n - t_1}$ where t is the time duration that a particular quality level is downloaded and q is the actual quality level downloaded in the corresponding time.
- Instability is the fraction of successive chunk requests by a player in which the requested video rate changes [136], measured at the steady-state.

Hence, we will look at the impact of the mobility protocols on video quality and buffer dynamics, network utilisation, and stability of the players.

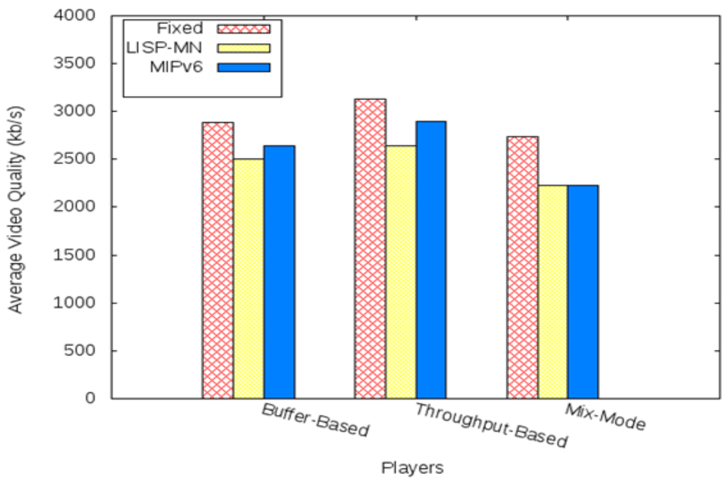


Figure 6.3-4 Average Video Quality of Three DASH players over LISP-MN and MIPv6

6.3.3.2 Average Quality Level

The average quality level attained by the different players on a fixed network (experiments with no mobility) while running the two mobility protocols is given in Figure 6.3-4. While the two mobility protocols have the same effect on a mixed-mode player, buffer and throughput-based players show slightly better performance running over an MIPv6 network. This is owing to the shorter SDT of MIPv6 compared to LISP-MN.

As can be noted from the figure, the mixed-mode player is the least affected by the dynamics of the mobility management protocols, achieving a similar level of average video quality regardless of the mobility protocol being used. However, it achieved the lowest average video rate compared to the two other players under consideration. Hence, as can be seen from Table 6.3-4 below, the mixed-mode player significantly underutilised the available capacity by using slightly above the 50% of the channel capacity in all cases.

	Utilisation (%)		
Players	**Fixed**	**MIPv6**	**LISP-MN**
Buffer-Based	61.54	57.83	56.60
Throughput-Based	76.42	67.66	63.62
Mixed-Mode	55.56	54.81	55.42

Table 6.3-3 Network Utilisation

However, the throughput-based player experiences the highest video quality, as shown in the figure, and is the most sensitive to the change in the context in comparison to the default fixed network. LISP-MN had the highest impact on the player's average video quality, which results in the player losing 15.5% of the video rate in comparison to the fixed network. The throughput-based player experience only 8% drop roaming through the MIPv6 network. In terms of throughput utilisation, the player achieves the highest percentage. As can also be noted from the figure, the buffer-based player is also not very sensitive to the change in mobility protocol, with little difference observed between MIPv6 and LISP-MN's performance in comparison to the fixed network. While the buffer-based player utilises the wireless link better than the mixed-mode player, it falls way behind on the three different experimental scenarios in comparison to the throughput-based player. MIPv6 produces slightly better average quality on the buffer and throughput-based players.

6.3.3.3 Stability

In the event of a changing condition, a DASH player is expected to adjust its video rate request. However, changing video quality level often is known to be detrimental to the quality of experience [137]. Figure 6.3-5 shows the impact the two mobility protocols have on the stability of video quality of the players. It is pertinent to note that the higher the value, the more unstable a player is. It

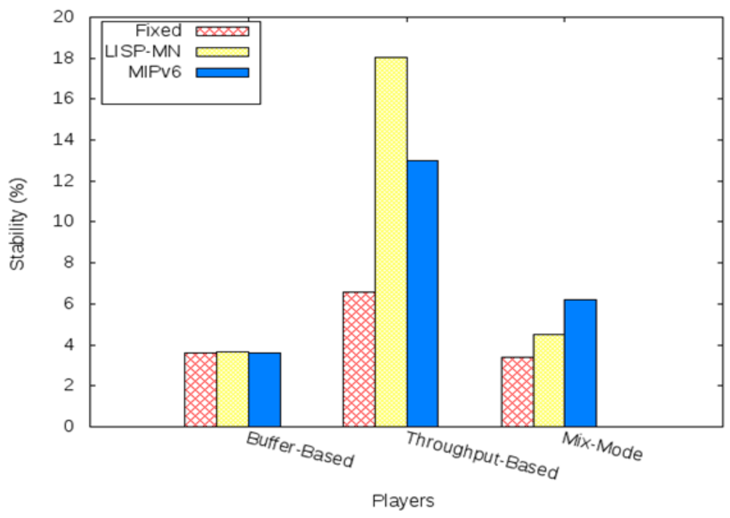


Figure 6.3-5 Stability: Percentage Change in Video Quality

means there is a frequent change in network condition causing the player to continuously change the quality of video it is downloading. As can be seen in Figure 6.3-5, the buffer-based player is the most stable in the face of mobility, in fact, it achieves almost the same stability level when ran on the fixed network as it did on MIPv6 and LISP-MN networks. Hence, the player is not really sensitive to a change of mobility protocol. However, the situation looks different when the video is streamed using the throughput-based player showing a significant increase in the instability level in comparison to buffer-based and mixed-mode players. It also shows better performance on MIPv6 than it does on LISP-MN. Finally, the mixed-mode player saw a slight increase when used within MIPv6 and LISP-MN as compared to the fixed network, however, it is still far less than the case of the throughput-based player.

6.3.3.4 Discussion

Section 6.3.3 answered the third evaluation question (EO3): *How do applications perform on top of the two protocols?*

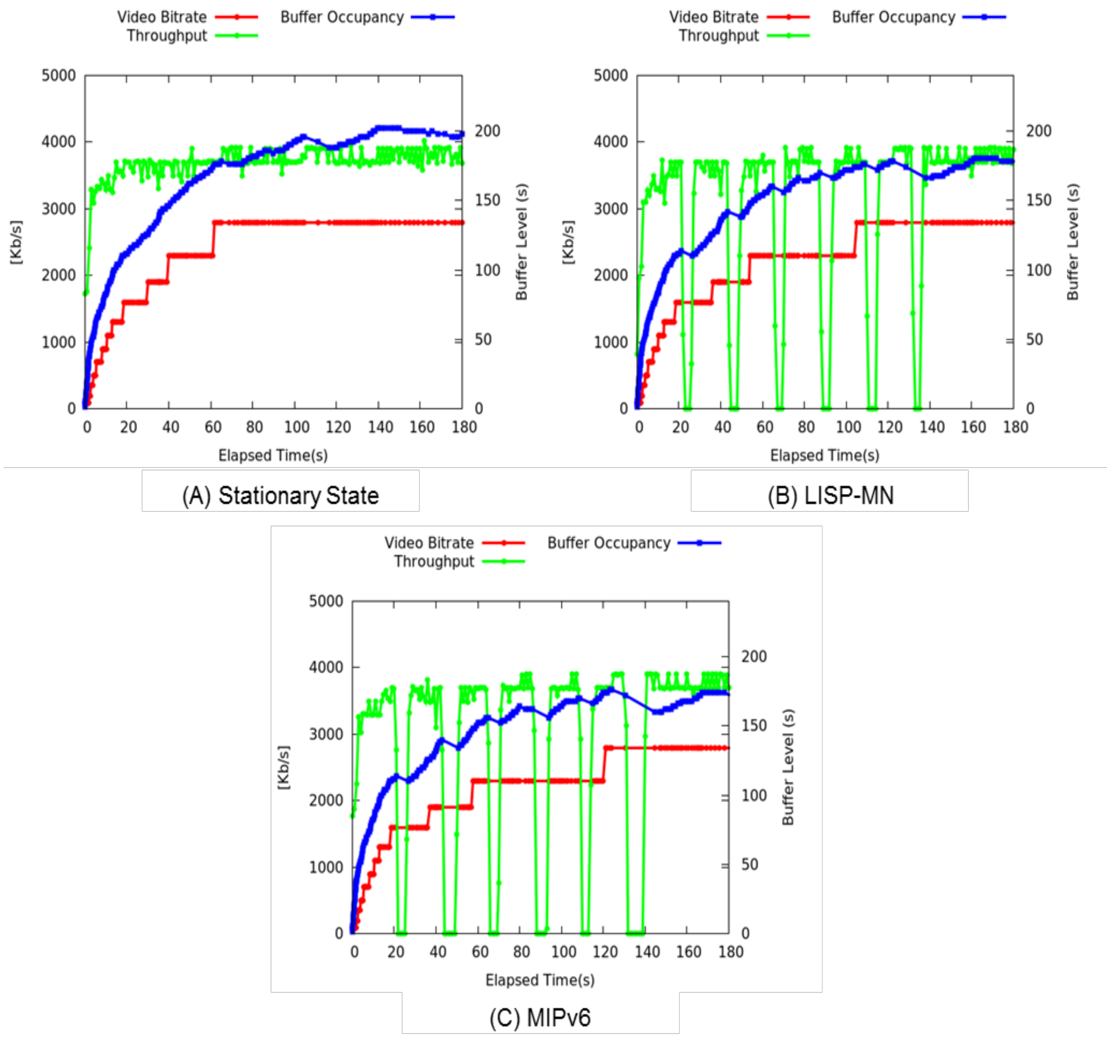


Figure 6.3-6 Buffer-Based Player in both Stationary and Mobility Context

We have seen from the preceding subsections that despite mostly similar performance recorded (section 6.3.2) between LISP-MN and MIPv6, the little differences in performance impacts on the applications running on top of the protocols – video applications in this case. MIPv6 recorded slightly better performance than LISP-MN in terms of SDT and packet loss in HoWE. Hence, its mobility did not affect the players as much as LISP-MN did. Figure 6.3-6 depicts the video bitrate, buffer occupancy and throughput for the buffer-based player running on fixed, LISP-MN and MIPv6 networks. The figures show the impact of mobility on the player with video quality and buffer levels changing as the MN moves from one network to another thereby affecting the player’s performance. While the video bitrate converges at around the sixtieth second for the fixed network, it takes about two minutes into the experiment for the player to converge on the mobile network owing to the effect of mobility.

6.4 Improved LISP-MN Architecture Evaluation

To evaluate the improved architecture, we run experiments on our testbed (Figure 6.2-1) with the addition of the Loc-server to measure the improvements that it brings to LISP-MN. We run experiments using the same settings and parameters discussed in section 6.3 for LISP-MN although we test for both fast and slow links in this case. We consider fast links to be radio links that have more than 2 Mbps download capacity as can be attained with Wi-Fi networks and 3/4G; while slow is considered to be a channel with less than 2 Mbps emulating low-speed Wi-Fi and cellular connectivity. This is based on the work of Riiser et al. [138], which shows that 2 Mbps bandwidth is rarely achieved when streaming media over 3G while commuting on a metro, bus, train, tram or ferry in the suburbs of Oslo.

802.11bgn antennas are used on the Ubuntu desktops serving as the MN and AR. Hence, the bandwidth of fast link channels can reach as high as 25 Mbps. The use of two different link speeds allows us to discover the extent of improvement that the introduction of a Loc-server brings to low and high-speed network environments. As in the previous section, we will look at UDP and TCP performance as well as DASH players over LOC_LISP (Loc-server supported LISP-MN network) and Plain_LISP (vanilla LISP-MN).

6.4.1 Qualitative Analysis

As a starting point, we first look at the features analysed in section 6.3.1 to see if there is any change from the original protocol's behaviour in the improved architecture.

6.4.1.1 Registration

With the introduction of the Loc-server, three new control messages are added to the mobility process. The MN sends an SMR to the Loc-server, and the server's map-request reply is followed by the MN's map-reply. We do not expect the additional messages to cause any difference in performance on the network as we see when calculating handover delay earlier (section 6.3.2.2) where five LISP-MN control messages take almost the same time to be exchanged as MIPv6's two. Any difference is likely to be in microseconds.

6.4.1.2 Tunnelling

In addition to the tunnel between the PXTR and the MN, the improved architecture requires a PXTR->Loc-server and Loc-server->MN tunnel created for forwarding of an MN's packets during the handover session. The Loc-server uses the same tunnelling features as vanilla LISP, i.e., UDP port number 4341 to receive data packets from the PXTR and uses the same port number to forward the received messages to the MN.

6.4.1.3 Security

The Loc-server utilises the security mechanism used by the PXTR to ensure the integrity of messages exchanged by use of a nonce in both control and data packets. This prevents a man-in-the-middle attack as packets are rerouted from the PXTR to the Loc-server and are eventually sent to the MN. As highlighted earlier in section 4.5, the Loc-server will only accept solicited map-replies, and the nonce in the reply must match the nonce the server generated and sent in the preceding map-request to the MN. The packets from the PXTR are forwarded by the Loc-server as they are. Hence, the MN confirms the nonce values in the packets before accepting the incoming messages.

6.4.1.4 Routing

The use of the Loc-server causes quadrangle routing during the handover event as packets move from the CN to the PXTR to the Loc-server and eventually to the MN. But since this only occurs during a handover, it has no impact on the protocol's operation as the normal routing behaviour resumes on handover completion.

6.4.1.5 Discussion

The discussion in this section answers EO4: *How do the design decisions taken in developing the new LISP-MN architecture affect the performance of the protocol?* The major area of concern is the increase in control messages which we believe will not have much impact on an individual MN. But there may be some scalability concerns as the number of MNs increases over a given geographical area and consequently, the service APs might be inundated with 38% more control messages as the MNs move – the additional messages are due to the control messages sent to the Loc-server. The effect of the additional control messages will be offset by the number of packets saved, which are either dropped or retransmitted at the end of the handover.

These retransmitted packets take up more network resources than the 38% more control messages to be used in saving the retransmitted packets. The tunnelling and routing between the PXTR and the MN do not in any way affect the operation of the protocol since the normal operation is resumed post-handover. The integrity of the messages is also ensured with the use of a nonce in both the control and the data packets.

6.4.2 Quantitative Analysis

The mobility scenario and evaluation metrics used in this section are the same as the ones discussed in section 6.3.3.1 although handover delay is not accounted for here since no significant difference is observed in the improved LISP-MN architecture in comparison to vanilla LISP-MN. We have also looked at the retransmission rate on the CN, as an additional evaluation metric, to see if the Loc-server's introduction reduces, in any way, the number of retransmissions by CNs.

6.4.2.1 TCP Service Disruption Time

As can be observed from Figure 6.4-1 below, the SDT for LOC_LISP in HeWE is consistently lower than that of Plain_LISP for the two different links speeds. We have established earlier in section 6.3.3.2 that a LISP-MN's mean handover time for a hard handover in HeWE is 3.27s. We can see from the box plots in the figure that with the new architecture, the data session resumes as soon as the handover is completed in the LOC_LISP scenario. For some of the runs, the SDT is as low as 3*s* against Plain_LISP's minimum value of 5*s* and mean of 5.75*s* on the fast links and 6.15*s* on the slow. The reduction in SDT for an MN in the improved architecture is quite significant with a mean of 3.35*s* on the fast link (Figure 6.4-1A) and 3.8*s* on the slow (Figure 6.4-1B), amounting to 41% and 38% reduction respectively in comparison to the Plain_LISP.

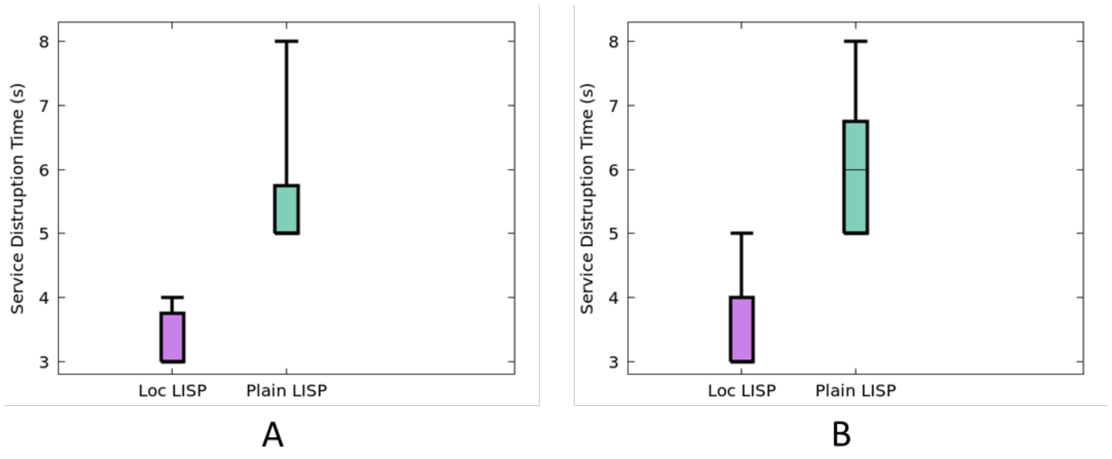


Figure 6.4-1 TCP Service Disruption Time in Heterogeneous Environment for Fast (A) and Slow (B) Links

With the LOC_LISP, the packets buffered by the Loc-server are forwarded to the MN as soon as the handover is completed. Consequently, the MN resumes communication as soon as possible by sending an acknowledgement for the next packets that have not been received. This eliminates the 2*s* delay in Plain_LISP before the MN actually resumes communication post-handover as observed in section 6.3.3.3. As soon as the transport layer handles the incoming packets forwarded by the Loc-server, it requested the subsequent packets as is conventional with the TCP protocol, speeding up the session resumption.

It is a similar story in HoWE (Figure 6.4-2), for which the SDT for LOC_LISP is consistently lower than Plain_LISP. LOC_LISP recorded an average of 3.7*s* on the fast link against Plain_LISP's 6.2*s*; and 4.2*s* on the slow link against Plain_LISP's 6.1*s*. This indicates better performance for LOC_LISP in the different links speeds. The percentage reduction in SDT stands at 39% for the fast link and 32% for the slow in comparison to Plain_LISP. As with the HeWE, the MN resumed communication immediately after handover on LOC_LISP leading to the reduction in SDT.

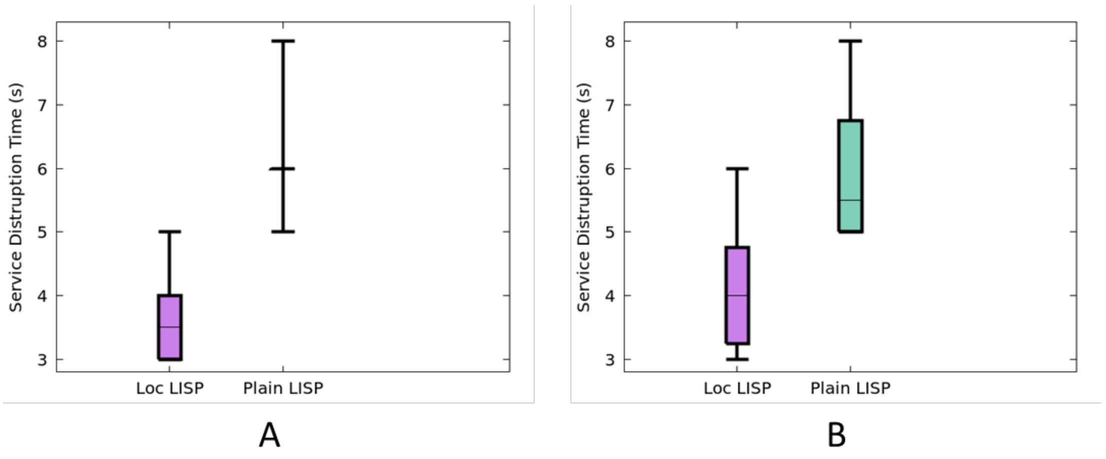


Figure 6.4-2 TCP Service Disruption Time in Homogeneous Environment for Fast (A) and Slow (B) Links

6.4.2.2 TCP Throughput

As discussed in the previous subsection, the SDT is significantly reduced with LOC_LISP in all the scenarios experimented. For HeWE, the average throughput attained by LOC_LISP between 21-30 seconds of the experiment, in the midst of which the hard handover occurred, is 13.4 *Mbps* for the fast link and 0.95 *Mbps* for the slow link against Plain_LISP’s 4.8 *Mbps* and 0.5 *Mbps* respectively. This shows 62% throughput improvement on the fast link and 47% improvement for the slow link (Table 6.4-1). In the HoWE, the throughput attained during the handover period (between 30-40 seconds) for LOC_LISP is 9 *Mbps* on the fast link and 0.45 *Mbps* on the slow link against 1.7 *Mbps* and 0.13 *Mbps* respectively for Plain_LISP. The percentage improvement in this scenario is 81% for the fast link and 71% the slow link. Figure 6.4-3 and Figure 6.4-4 depict the throughput attained in the different scenarios during the experiments. The figures also show the impact that the handovers had on the MN’s throughput. It is obvious from the figures that the introduction of the Loc-server does not in any negative way affect the performance of the LISP-MN

	HeWE (Mbps)		HoWE (Mbps)	
Protocols/Speed	**Fast**	**Slow**	**Fast**	**Slow**
LOC_LISP	13.4	0.95	9	0.45
Plain_LISP	4.8	0.5	1.7	0.13
Improvement (%)	62	47	81	71

Table 6.4-1 Tabular Representation of Throughput Performance of the Two Architectures

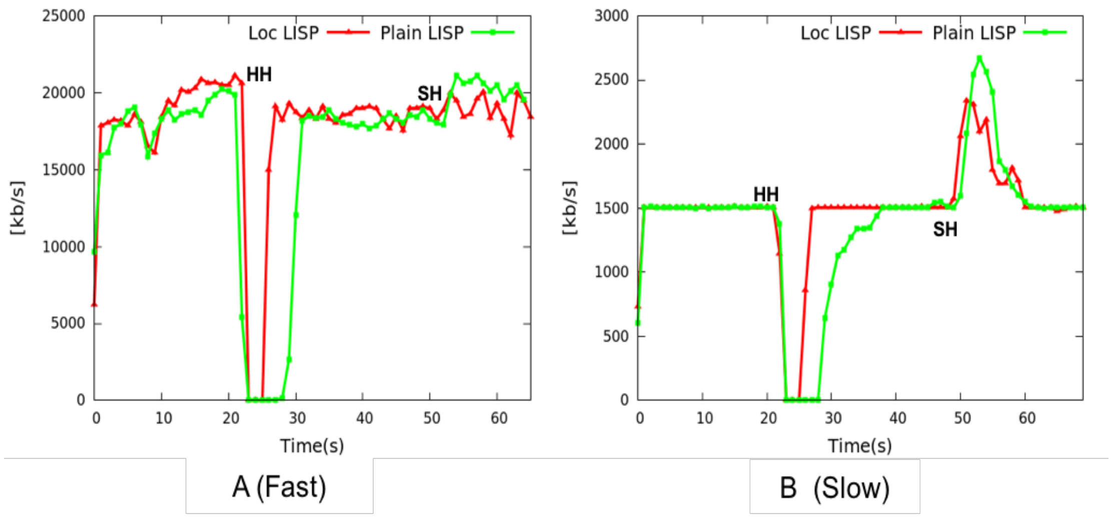


Figure 6.4-3 TCP Throughput in Heterogeneous Wireless Environment for Fast and Slow Links

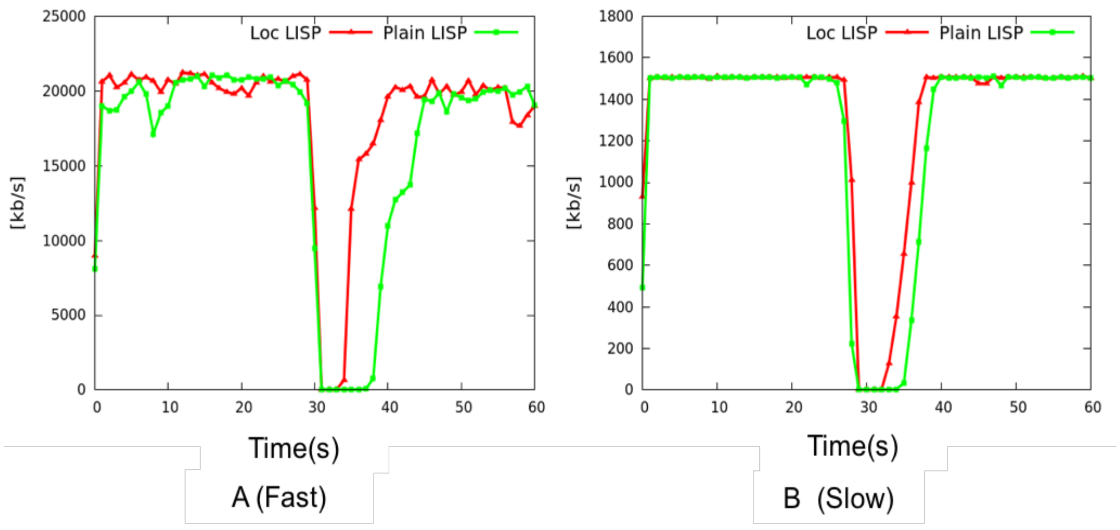


Figure 6.4-4 TCP Throughput in Homogeneous Wireless Environment for Fast and Slow Links

protocol as the two architectures recorded similar throughput performance during the MN download.

The impact of the server introduction is visible at the point of handover with LOC_LISP showing faster recovery and attaining full throughput as soon as the handover is completed. For hard handover in HeWE and handover in HoWE, the throughput level went to zero for some time in all the different scenarios of the two architectures. For soft handover of HeWE however, different throughput behaviour is observed on the slow link. The link experienced a moment of increased throughput during the handover event (Figure 6.4-3B) as the two interfaces were used simultaneously for a period before communication was completely transferred to the targeted access link.

On the fast link, there is no visible effect of soft handover (Figure 6.4-3A) except for the little improvement in throughput which was a result of the targeted wireless link being a little bit faster than the previous link. Even with the two interfaces active for the period, at 20 *Mbps*, the TCP congestion window at the CN and/or the receive window at the MN was already at its maximum, leaving no room for improvement. With the slow link, however, the program sensed the increased bandwidth on the link and utilised it.

6.4.2.3 UDP Packet Loss

The amount of packet loss for a hard handover in HeWE is depicted in Figure 6.4-5A, showing the percentage loss on the LOC_LISP as being far lower – 3.98% for the fast link and 4% for the slow link – than the vanilla LISP-MN which records about 15% loss in the two different links. We also observed similar values for the HoWE for both LOC_LISP and the Plain_LISP as shown in Figure 6.4-5B.

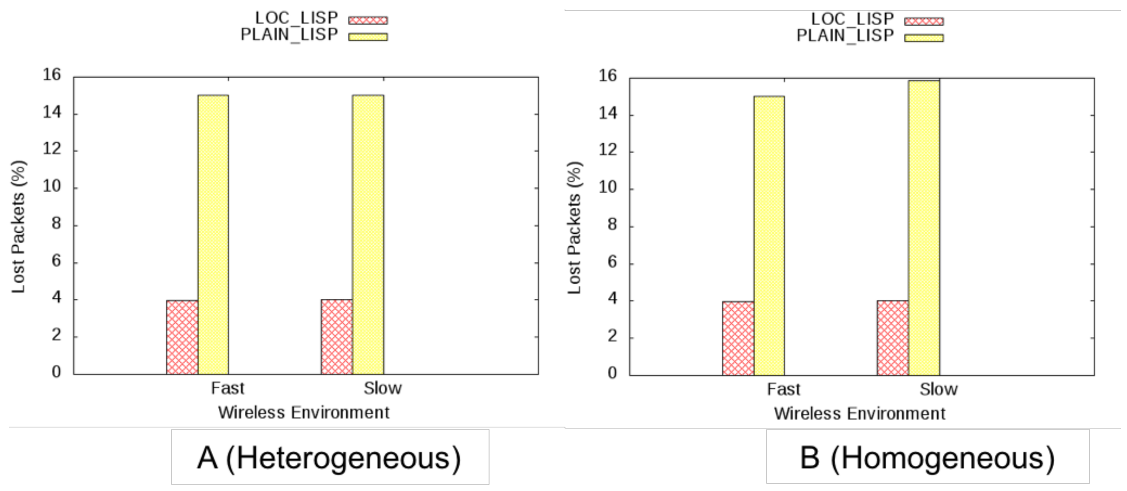


Figure 6.4-5 UDP Packet Loss in Heterogeneous and Homogeneous Wireless Environments

This is because the Loc-server buffered the packets rerouted to it by the PXTR node and forwarded all the packets to the MN on handover completion. This is the reason why the UDP throughput raises to almost double its normal level (Figures 6.4-6 and 6.4-7) immediately after the handover as packets arrived from the Loc-server. The few lost packets in LOC_LISP scenario are likely caused by two events: the burst of traffic from the Loc-server combined with the traffic from the PXTR immediately after the handover being too much for the NIC to handle, and some packets were dropped at that point; and/or there are packets on the wire from the PXTR that arrived at the previous AR after the MN had already begun the handover process and were dropped by the router. We can also observe from the throughput figures (6.3-6 and 6.3-7) that the performance is the same in both fast and slow networks as the voice packets emulated for the experiment, with the maximum throughput of about 80 *Kbps,* were conveniently handled even by the slow link.

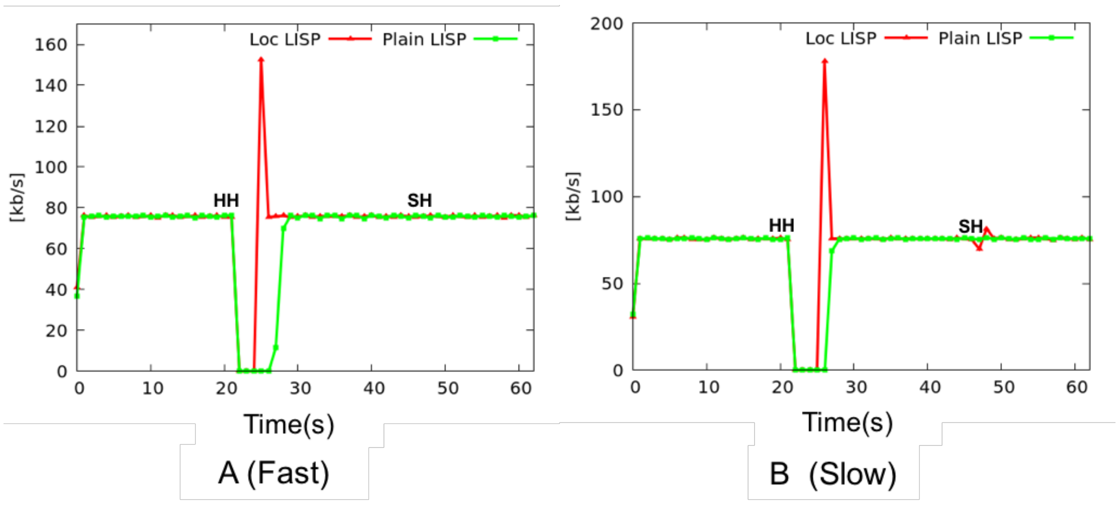


Figure 6.4-6 UDP Throughput in Heterogeneous Wireless Environment for Fast and Slow Links

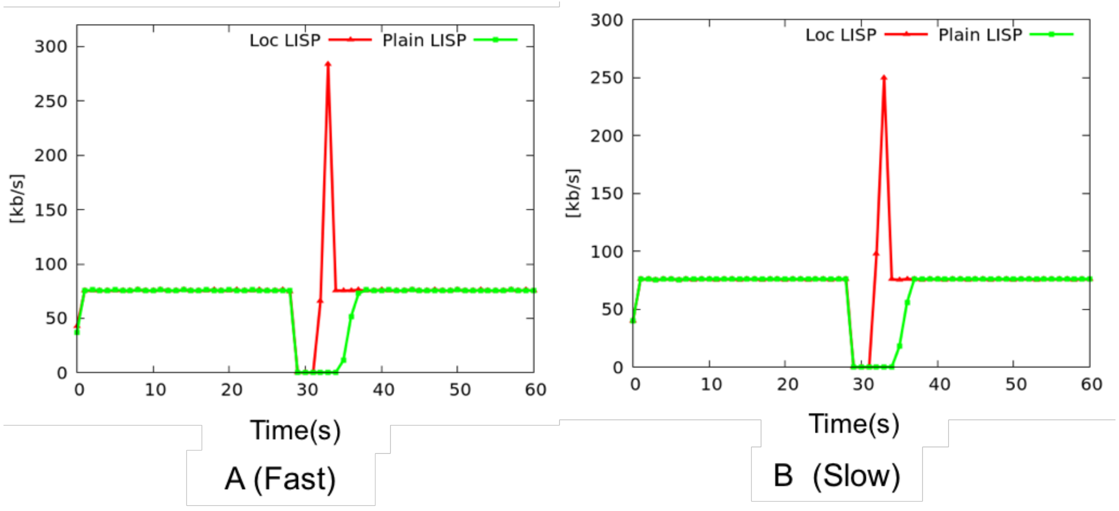


Figure 6.4-7 UDP Throughput in Homogeneous Wireless Environment for Fast and Slow Links

6.4.2.4 TCP Retransmission on the CN

When TCP packets are not acknowledged by the MN, based on the behaviour of TCP, the CN will attempt to retransmit the unacknowledged packets. While this could happen at any point during communication, it is prevalent at the point of handover. On coming back online, the MN requests the CN to continue sending packets by acknowledging the last received message from the server. Because the packets sent by the CN during handover are not dropped with our Loc-server, we can see in Figure 6.4-8 that the retransmission rate is lower in

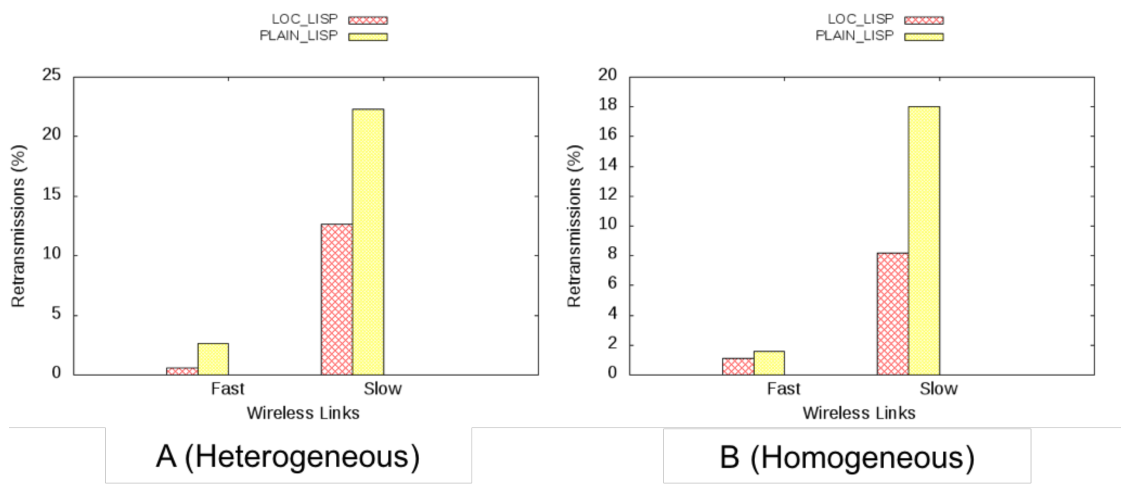


Figure 6.4-8 Retransmission by the CN on Heterogeneous and Homogeneous Wireless Environments

Links	Fast (%)	Slow (%)
LOC_LISP	1.126	8.164
Plain_LISP	1.575	18.001

A

Links	Fast (%)	Slow (%)
LOC_LISP	1.606	12.639
Plain_LISP	2.627	22.266

B

Table 6.4-2 Percentage Retransmission on The CN for Heterogeneous (A) and Homogeneous (B) Wireless Networks

all experiments carried out.

The retransmission rates depicted in Figure 6.4-8 are also presented in Table 6.4-2. The fast links in HeWE and HoWE recorded 45% and 100% reduction in retransmission rate respectively. The slow links, on the other hand, recorded close to 100% reduction in retransmission rate in the two wireless environments. The high reduction experienced in the slow link as compared to fast was due to the higher number of losses experienced with the slower channel.

6.4.2.5 Discussion

Section 6.4.2 addressed the question in EO5: *What are the performance improvements the newly-built LISP-MN architecture brings in terms of the metrics analysed in the second evaluation question as well as the TCP retransmission rate?*

Protocols	Heterogeneous		Homogeneous	
	Fast	Slow	Fast	Slow
LOC LISP	3.3	3.7	3.7	4.1
Plain LISP	5.6	6.0	6.1	6.0

Table 6.4-3 Average TCP SDT in Heterogeneous and Homogeneous Wireless Environments

There are many obvious improvements to the performance that the LOC_LISP architecture brought to LISP-MN. We have seen the improvement in the SDT as summarised in Table 6.4-3 where the improved architecture consistently performed better than the vanilla LISP-MN. Improvements were also recorded with throughput, packet loss, and the retransmission rate.

6.4.3 Video Applications

As with our video analysis in section 6.3.3, we used similar settings as presented in that section to evaluate video applications in HoWE although up to 5000 Kbps video quality is available for the download in this aspect of evaluation and both fast and slow links are observed. The three different players were again used for the experimentation.

6.4.3.1 Average Quality Level

As can be observed from Figure 6.4-9 below, all the players downloaded at a higher quality level on the fast link while running on LOC_LISP in comparison to Plain_LISP. This is simply due to the reduction in SDT and packet loss by sending the buffered packets from the Loc-server. The mixed-mode player shows the best performance followed by the throughput-based player. We can also observe the improvements in the average quality levels of the different players from figures 6.4.10, 6.4.11, and 6.4.12, especially with the slow link for buffer-based, and fast link for throughput-based and mixed-mode players.

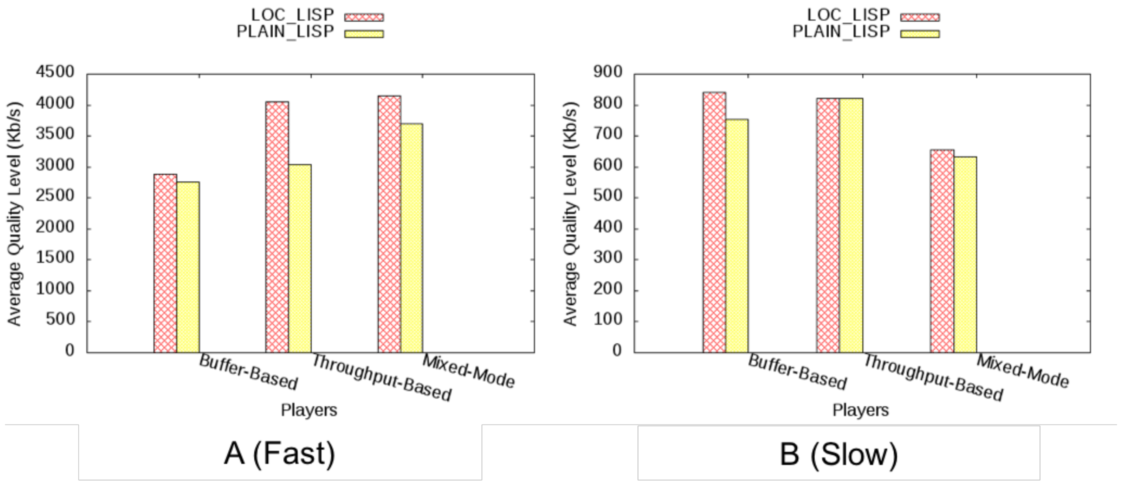


Figure 6.4-9 Average Quality Level Attained by the Players on Fast and Slow Links

We can also notice that buffer draining is faster with Plain_LISP in comparison to LOC_LISP, again due to the shorter SDT and improved throughput performance post-handover. Although the mixed-mode player is found not to be as sensitive to mobility protocol dynamics in section 6.3.3.2, it recorded improvements when run using LOC_LISP as depicted in the figures. The player recorded the lowest video rate as earlier discussed in section 6.3.3.2 as it did on the slow link here, but it came up with the best performance on the fast link.

We can also see that the throughput-based player is the highest beneficiary of the improved architecture on the fast link with an average download of 4,055 *Kbps* against 3,042 *Kbps* for Plain_LISP; it is followed by the mixed-mode player. However, the buffer-based player shows the best improvement on LOC_LISP against Plain_LISP on the slow link, with 840 *Kbps* to 753 respectively.

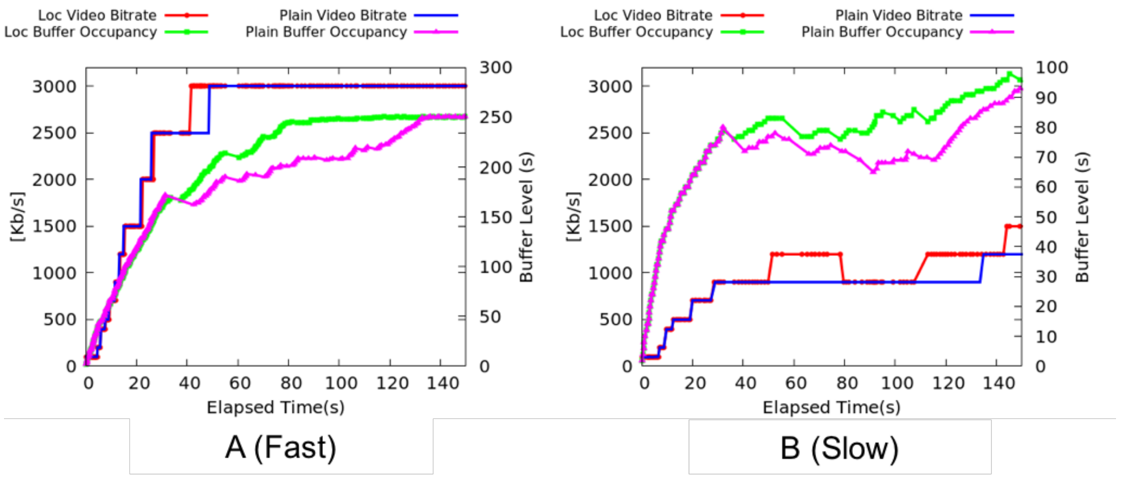


Figure 6.4-10 Buffer-Based Player: Buffer and Average Quality Levels for Fast and Slow Links

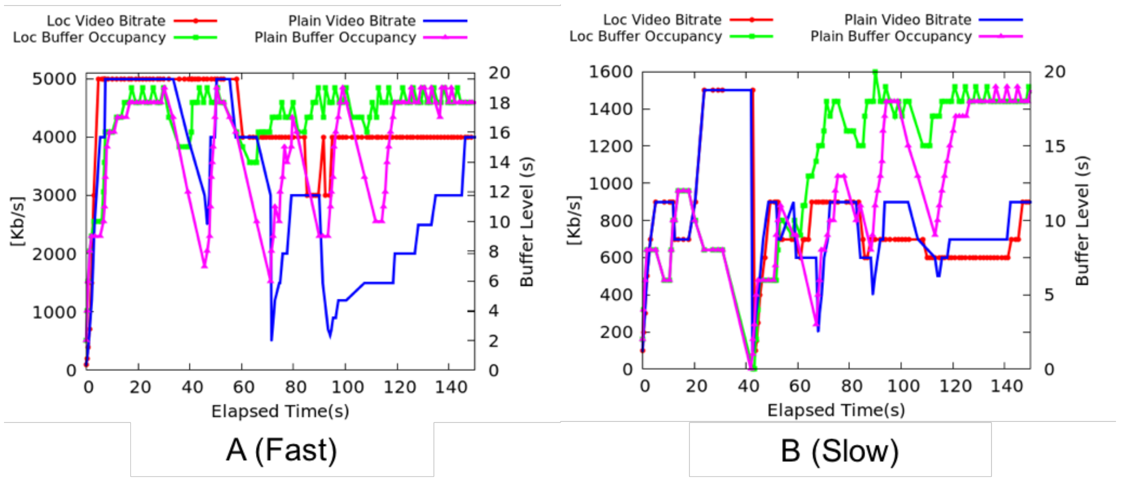


Figure 6.4-11 Throughput-Based Player: Buffer and Average Quality Levels for Fast and Slow Links

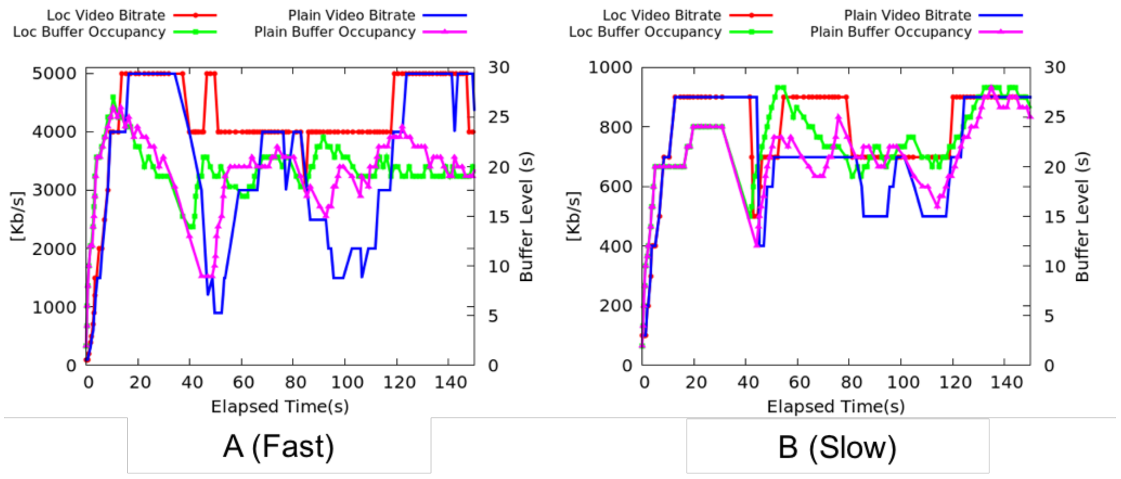


Figure 6.4-12 Mixed-Mode Player: Buffer and Average Quality Levels for Fast and Slow Links

	Utilisation (%)	
Players	**LOC LISP**	**PLAIN LISP**
Buffer-Based	58	55
Throughput-Based	81	61
Mixed-Mode	83	74

Table 6.4-4 Network Utilisation on Fast Link

	Utilisation (%)	
Players	**LOC LISP**	**PLAIN LISP**
Buffer-Based	47	42
Throughput-Based	46	47
Mixed-Mode	36	35

Table 6.4-5 Network Utilisation on Slow Link

6.4.3.2 Network Utilisation

Since the available bandwidth of the fast link is far more than the largest quality level that a user can download, we calculate link utilisation using the difference between the downloaded quality level and the highest available video download, which is 5000 *Kbps* (as stated earlier). The mixed-mode player shows the best utilisation of the three players for the fast Internet connection with 83% for LOC_LISP against 74% for Plain_LISP but performs poorly in slow networks as shown in tables 6.4-4 and 6.4.5. The buffer-based player shows slightly better performance (by only 1% against throughput-based) in slow networks but did poorly on the fast network.

While there is a remarkable difference in utilisation between LOC_LISP and Plain_LISP on the fast link for throughput-based and mixed-mode players, the buffer-based player shows little sensitivity to the improvement in throughput and reduced SDT that comes with LOC_LISP showing only 3% difference in link utilisation. This is because the buffer-based players are designed to download many seconds of a particular video quality, which is played while the interface is down during handover, and unless the SDT extends to a very long time, the user is likely to see the same quality level of download.

6.4.3.3 Stability of players

As stated earlier, maintaining a particular video quality download is vital in enhancing the user experience and frequent changes is detrimental to the quality of experience. It is obvious from Figure 6.4-13 that LOC_LISP produces better stability for throughput-based and mixed mode players for the two different link speeds. There is a tie in performance on the fast link for the buffer-based player but Plain_LISP show better stability on the slow link. This is due to the fact that an MN can maintain a particular quality level while using a buffer-based player even amidst handovers as we saw in Figure 6.4-10. And although receiving buffered packets at the completion of the handover improved the average video quality by 3% on the fast link and 5% on the slow link, it comes at the expense of stability. Note again that , the higher the percentage value, the more unstable a player is.

The most remarkable improvement in stability recorded is that of the throughput-based player on the fast link, with the player being 67% more stable on LOC_LISP than on Plain_LISP, followed by the mixed-mode player. This is expected because as a player that relies on throughput, it is always expected to perform better on a network that provides significantly more of it. Hence, the improvement in throughput brought about by the introduction of Loc-server helps the player reach higher stability levels despite the many handovers during streaming. The mixed-mode player shows better stability than the throughput-

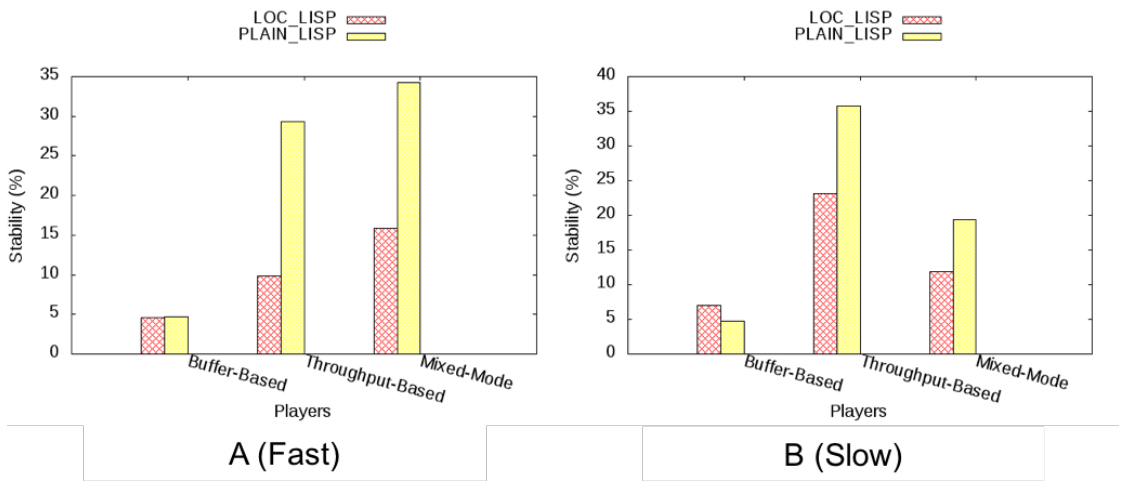


Figure 6.4-13 Stability: Percentage Change in Video Quality For Fast (A) and Slow (B) Links.

based player on a slow link with Plain_LISP showing higher instability for both players in comparison to LOC_LISP.

6.4.3.4 Discussion

Section 6.4.3 discussed the last two EOs: '*Does the newly developed architecture improved the performance of applications*?' and '*Does the new architecture improve network utilisation; does the improvement differ between networks of different link speeds?*'.

The three different players chosen for the evaluation have all demonstrated improved performance when running on top of LOC_LISP with the mixed-mode player showing the best performance on the fast wireless link. Although the mixed-mode player was found not to be sensitive to changing mobility protocol (LISP-MN vs. MIPv6), it recorded better performance when running on top of LOC_LISP. The throughput-based player recording the highest download rate of the three is not surprising as the new architecture improved the throughput experience during handover. The improvement in throughput is also the reason why buffer draining is significantly reduced when the players were run on top of the new architecture.

We have also observed improvements in network utilisation on the fast link with the players running on LOC_LISP utilising the available network capacity to download higher video rates. LOC_LISP did not record any remarkable improvement against the Plain_LISP in network utilisation on the slow link, especially for the throughput-based player. This is generally because with limited download capacity, the players maintain the current quality level despite sensing improvement in throughput and a reduction in SDT, as the next higher quality level could not be sustained with the available network resources.

CHAPTER SEVEN

CONCLUSION

This thesis analysed the inter-domain handover of LOC/ID split-based mobile nodes and how the movement event causes packet loss and disrupted communication sessions. It investigated the different proposed solutions geared towards improving the LOC/ID protocols' mobility performance and highlighted the observed shortcomings. To address the handover issues, the thesis argued that improving the architecture of the protocols with the introduction of a Loc-server to buffer packets during the movement event could help in mitigating the loss of packets and shortening the duration of service disruption. The thesis presented how the architecture of different protocols – LISP-MN, IVIP, ILNP, and HIP – could be improved using the Loc-server. It also showed the implementation and the evaluation of LISP-MN as a representative of the other LOC/ID protocols. This chapter concludes the thesis by looking back at the contribution of each of the chapters as well as the overall thesis. It also talks about the future direction of the research and presents general comments about improving mobility with LOC/ID protocols. We also presented our major findings and conclusions in the closing remark section.

7.1 Thesis Summary

The first chapter of the thesis introduced the reader to the state of mobile communication in the world today and how the widespread use of smartphones impacts on the traffic traversing the Internet. It also provides an overview of mobility management protocols in general and LOC/ID split protocols in particular, and how the developments in cellular networks are driving the widespread use of wireless networks. The chapter introduced the reader to the research hypothesis and research questions as well as the aims and objectives of the PhD.

The second chapter touched on the history of LOC/ID split from conception to state of the art. It discussed four host-based LOC/ID split architectures, LISP-MN, IVIP, ILNP and HIP, focusing on their control and data planes and especially detail related to mobility. Each protocol's discussion was followed by a review of other research works to improve its operation except IVIP, for which no published research works geared towards improving mobility was found. The chapter also discussed MIPv6 as the protocol that served as the benchmark for evaluating LOC/ID split protocols. The review of research works on the architectures was summarised at the end of the chapter.

Chapter three began by describing the two types of handovers in inter-domain mobility scenarios, horizontal and vertical. It detailed the performance degradation that an MN experiences as the result of a handover and the requirements that any system geared towards improving LOC/ID split protocols' handover process needs to satisfy, which included: mitigating the drop in throughput; eliminating/reducing packet loss; and reducing service disruption time.

The forth chapter described the design of the architecture geared towards satisfying the requirements described in chapter three. The chapter provided a high-level view of the improved architecture, detailing the necessary components that need to be implemented to realise the proposed architecture. It also discussed the way the new architecture could be integrated to work with the five protocols reviewed in chapter two.

Chapter five presented the implementation work. It described the LISPmob and UMIP codebases used as the LISP-MN and MIPv6 implementation on the testbed respectively. It also described how the aspects of the LISPmob code were changed to enable the LISP-MN protocol to work in the new architecture. The chapter also presented flowcharts showing how the algorithms for control and data modules of the Loc-server are executed.

The penultimate chapter ascertained the improvements that introducing the Loc-server into one of the protocols brought. It began by presenting a performance comparison between LISP-MN and MIPv6 discussing both the qualitative and quantitative differences and similarities of the two protocols. This is to set a benchmark for LISP-MN's performance in the inter-domain mobility scenarios looking at TCP, UDP and video application performance. The chapter also evaluated the new architecture by comparing its performance with that of vanilla LISP-MN looking at similar performance indicators.

7.2 Revisiting the Objectives of the Thesis

Section 3.3 discussed five objectives of this research work. Below is a discussion of how the objectives of the thesis are achieved.

1. Implement a LOC/ID protocol on a laboratory testbed including all the required components for its mobility mechanism...

2. Analyse the inter-domain mobility performance of MIPv6 and a LOC/ID protocol on the testbed...

These two objectives were fulfilled in chapter five and six with the implementation of LISP-MN using the LISPmob code and MIPv6 using the umip code on the laboratory testbed. The LISPmob code ran on the MN, the map-server and PXTR while the UMIP ran on the MN and the HA. We chose LISP-MN for the implementation for two reasons: firstly, all the required network nodes for the protocol's operation can be implemented on a laboratory testbed; secondly, tests could potentially be carried out on the Internet using the open LISP beta network [105]. As far as we know, none of the other protocols have a dedicated Internet overlay network similar to the LISP beta network. We have also mentioned in section 3.3 (second objective) that we would be using MIPv6 as the benchmark for measuring the handover performance of LOC/ID split protocols. MIPv6 is implemented to serve as the benchmark for measuring the handover performance of LISP-MN as presented in chapter six.

A qualitative and quantitative comparison of the two protocols was presented in chapter six as a contribution to the argument of LISP-MN's possible adoption as the host mobility protocol of the future. We also provided performance analysis of video applications on top of the two mobility protocols.

3. Design and implement a buffer node; we call Location Area Server (or simply Loc-server) that will be storing packets for a pre-registered MN and forwarding the packets at the request of the MN.

In chapter four, we presented the design description of our newly developed ILISA, as a network architecture that proposed the introduction of a new network node named Loc-server into LOC/ID split-based architectures to improve the performance of this class of mobility protocols. Loc-server is central to this architecture and designed to ensure that no packet is lost at the point of an MN's handover. We presented how the new node could be integrated to work with different LOC/ID protocols.

In the fifth chapter, we discussed the modules developed on the MN to make it aware of the Loc-server. We also detailed the development of the server itself with different modules that perform the functions outlined in the objectives, including registering an MN and reserving buffer space for it, buffering the packets as they come in and forward to the new location of the device.

4. Implement new signalling mechanisms on the MN to enable the mobile device to interact with the Loc-server. In addition to the LOC/ID protocol's *control messages, build four new signalling mechanisms on the MN to enable the packets' redirection at the point of handover…*

The four signalling mechanisms (new control messages) introduced on the MN are detailed in the design chapter to include HITr, BU-Null, BU-MN_LOC and a BU message with Loc-server address. While HITr is a newly developed module on the MN, the three other messages were built by making changes to the map-reply message of the LISP-MN as discussed in the fifth chapter on implementation.

5. Re-evaluate the performance of the LOC/ID protocol with the support of Loc-server to determine how much improvement the new node brings to the protocol in question.

In the sixth chapter, we provided a thorough performance evaluation of the newly built architecture by testing the system in both homogeneous and heterogeneous wireless environments. In each of the scenarios, we looked at the performance of the two popular transport protocols, TCP and UDP, as well as the file download and video application running on top of TCP. We performed tests on wireless links with low and high available bandwidth to gain insight into the type of environment that our new architecture will produce better performance, and at the same time determine how much support the Loc-server could give to the LOC/ID protocols.

7.3 Revisiting Requirements

We discussed three main requirements of any network architecture based upon LOC/ID split concept and geared towards improving the mobility of nodes to include the following:

1. Mitigating the Drop in Throughput.
2. Eliminating/Reducing Packet Loss
3. Reducing Service Disruption Time

As shown in the evaluation chapter, both LISP-MN and MIPv6 suffered long SDT, which affected both TCP and UDP-based applications. The handover of the two protocols caused throughput degradation and a high number of packet losses. Hence, the performance of an application running on the two protocols will suffer greatly during handovers. This is demonstrated with the performance of DASH video players as the players experienced low average video quality download, poor network utilisation, and increased instability. The packet loss experienced by the two protocols could potentially increase network congestion as MNs request for retransmission of lost TCP packets at handover completion.

ILISA, as implemented on LISP-MN, has significantly reduced the drop in throughput experienced by Plain_LISP during a handover. With the improved architecture, the throughput attained during the handover period was significantly improved for both fast and slow wireless links. This is regardless of the wireless environment (heterogeneous or homogeneous) that the handover took place, although the new architecture improved throughput experience for MNs in HoWE more than it does in HeWE. LOC_LISP significantly reduced throughput degradation during a handover.

ILISA has significantly reduced the packet losses that are experienced during handover as the Loc-server buffered and forwarded all the packets sent to it during the handover. In fact, an MN in Plain_LISP environment experienced four times more losses in a HoWE than an MN in ILISA. Introduction of the Loc-server in LISP-MN architecture significantly reduced SDT by 41% for the fast link and 38% for the slow link in HeWE and similar improvements were recorded in HoWE. We also observed that sending buffered packets to the MN spurred the device into requesting for subsequent segments and the streaming continued immediately. A reduction in packet loss significantly improved UDP downloads as demonstrated by the newly improved architecture.

There is a significant improvement in the video streaming experience as all the players demonstrated improved performance on ILISA. While the throughput-based and the mixed-mode players recorded improvement in network utilisation running on the new architecture, the buffer-based player did not show any improvement in utilisation even with the Loc-server support. In fact, the improved average quality level recorded by the player is at the expense of stability.

7.4 Future Work

The most important work for the future is to integrate the Loc-server functionality into anchor nodes or remote servers. For instance, in the case of LISP-MN, the Loc-server functionality can be installed on the PXTR or the CN (in scenarios with no PXTR). It could also be on the TTR for IVIP, or the CN for ILNP and HIP. This will significantly reduce the number of control messages sent pre and post-handover. The CN expected to have this functionality will be servers that provide services on the Internet and are running in LOC/ID split environment – we discussed this type of CNs in chapter 2. In such instances, an MN informs the server to hold up sending packets as the MN moves and requests the server to resume on completion of the switch.

Another work is to integrate a predictive handover capability into the new architecture. Although we can save packets to prevent loss during handover and can resume communication much faster after the handover with ILISA, we still suffer some break in communication due to link-switching delay and the associated handover events for hard handover in HeWE and handovers in HoWE. In our current architecture, once the MN discovers that its link is going down, it sends buffering request message to its Loc-server.

We will build a mechanism where the MN scans for and starts association procedure with its target AR by exchanging router solicitation and advertisement messages via its current link. If the MN can associate with the target link before the handover, it provides its new location to its anchors and CNs for redirection of its packet. This will make handover smoother for the MN as we saw with soft handover in HeWE (section 6.4.2.2) where the reduction in throughput was unnoticed on the fast link and even recorded an increase on the slow link. If the MN could not associate with any link prior to handover, it reverts to requesting for buffering of packets by the PXTR, or for remote nodes to hold sending packets.

There is also the need to test the newly improved LISP-MN architecture on the Internet by connecting to the LISP BETA network.

7.5 Closing Remarks

The availability of different wireless technologies enables mobile devices to switch between distinct domains to ensure continuous Internet connectivity as they move around. This inter-domain mobility is characterised by delay, packet loss, and service disruption. The LISP-MN’s long SDT causes degradation in network services and underutilisation of network resources. In this work, we proposed an improved architecture for LISP-MN (LOC_LISP) that significantly reduces the SDT and the packet loss experience by mobile nodes (MN) at the point of handover. The proposed scheme introduces a new node into LISP-MN’s architecture, which is termed Loc-server that is responsible for buffering packets that would have been dropped at the point of handover but the server forwards to the MN at the completion of its handover.

Experiments were run on a laboratory testbed for different link speeds. We found that by introducing the Loc-server, packet losses were reduced and throughput on the links was improved. Also recorded is a reduction in SDT as the MN was spurred by the arriving buffered packets to immediately request for subsequent packets from the server rather than wait a little longer as we observed with Plain_LISP. The effect of this improvement is also visible on DASH video players, which recorded higher average video quality and improved stability due to the shorter SDT experienced in the LOC_LISP environment. We have also realised that while our solution improves LISP-MN performance for slow links, it performs much better for links that are greater than 2*Mbps*.

Our major conclusions from this work is an attempt to answer the questions asked in section 1.3:

- Mobility with LOC/ID protocols, as represented by LISP-MN, is largely similar to mobility with MIPv6 for all the parameters investigated, apart from the few instances where the latter recorded better performance.
- LISP-MN's mobility has some serious impact on Internet application to the extent of severely affecting the applications' performance.
- There is no difference in performance for hard handover of dual interface MNs and handover for single interface mobile node. Similar results were recorded for both LISP-MN and MIPv6.
- Reducing packet loss during handover can significantly improve the performance of transport layer protocols and by extension the applications that run on them. Buffering packets during handover by the Loc-server and forwarding upon completion proved an effective way of improving mobility performance.
- Internet applications benefit significantly from reduced loss and service disruption during handover as we demonstrated by DASH players' performance.
- Introduction of Loc-server changed the LISP-MN's mode of operation at the point of handover, although it improved the protocol's performance at the same time.
- Introducing the Loc-server increased the number of control messages at the point of handover by 38%, but the number of packets saved and delivered after handover would have taken much more network resources if they were to be retransmitted.
- LOC/ID Split mobility protocols have some advantages over MIPv6 family since they have no concept of home networks and have no

centralised mapping. This is in contrast to MIPv6 that uses Has, routing most its data via the home network. When mobility is only the consideration, then LISP-MN does not show any better performance than MIPv6. However, when the overall benefits of using LISP, upon which LISP-MN is built, are taken into consideration – Internet scalability, multi-homing support, Internet traffic engineering, security, etc – then LISP architecture is the better fit for the Internet of the future than the current IP and MIPv6 architecture.

REFERENCES

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper," Cisco Systems Inc., San Jose, CA. 03 February, 2015.

[2] "3rd Generation Partnership Project, Evolved Universal Terrestrial Radio Access (E-UTRA); Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall Description (Release 8)," in *3GPP TS 36.300*, ed: 3GPP TR 21.101, 2009.

[3] "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2 (Release 13)," in *3GPP TS 36.300 V13.2.0 (2015-12)*, ed: 3GPP, 2015.

[4] *WiMAX Forum Network Architecture. Architecture Tenets, Reference Model and Reference Points Base Specification WMF-T32-001-R021v01. WMF Approved (2012-04-17)*, 2012.

[5] "Overview for Ultra Mobile Broadband (UMB) Air Interface Specification (3GPP2 C.S0084-000-0_v2.0)," *3RD GENERATION PARTNERSHIP PROJECT 2,* 2007.

[6] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper," Cisco Inc.2017, Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[7] Z. Gao, A. Venkataramani, J. Kurose, and S. Heimlicher, "Towards a Quantitative Comparison of the Cost-Benefit Trade-offs of Location-Independent Network Architectures," in *ACM SIGCOMM'14*, Chicago, IL, USA, 2014.

[8] E. Gustafsson and A. Jonsson, "Always Best Connected," *IEEE Wireless Communications,* vol. 10, no. 1, pp. 49 - 55, 2002.

[9] J. Laganier, T. Higuchi, and K. Nishida, "Mobility management for all-ip core network," *NTT DOCOMO Technical Journal,* vol. 11, no. 3, pp. 34-39, 2009.

[10] "Defence Advanced Research Projects Agency - A History of the ARPANET: The First Decade (Report)," *Arlington, VA: Bolt, Beranek & Newman Inc.,* 1 April 1981.

[11] K. N. Ashraf, V. Amarsinh, and D. Satish, "Survey and analysis of mobility management protocols for handover in wireless network," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, 2013, pp. 413-420.

[12] C. Perkins, J. Arkko, and D. Johnson, "Mobility Support in IPv6," *Internet Requests for Comments,* vol. RFC 6275, 2011.

[13] C. So-In, R. Jain, S. Paul, and J. Pan, "Future wireless networks: key issues and a survey (ID/locator split perspective)," *Int. J. Commun. Netw. Distrib. Syst.,* vol. 8, no. 1/2, pp. 24-52, 2012.

[14] D. Farinacci, D. Lewis, D. Meyer, and C. White, "LISP Mobile Node," *IETF draft-ietf-lisp-mn-00*, 28-04-2017.

[15] W. Köpp and A. Klein, "Locator/Identifier Split," in *Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, Munich, 2011.

[16] B. Quoitin, L. Iannone, C. De Launois, and O. Bonaventure, "Evaluating the benefits of the locator/identifier separation," in *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, 2007, p. 5: ACM.

[17] T. Li, "Design Goals for Scalable Internet Routing," *Internet RFCs, ISSN 2070-1721, RFC 6227,* 2011.

[18] W. Ramirez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martinez, and M. S. Siddiqui, "A survey and taxonomy of ID/Locator Split Architectures," *Computer Networks,* vol. 60, no. 0, pp. 13-33, 2/26/ 2014.

[19] M. Komu, M. Sethi, and N. Beijar, "A survey of identifier–locator split addressing architectures," *Computer Science Review,* vol. 17, pp. 25-42, 2015/08/01/ 2015.

[20] V. Fuller, A. Jain, D. Lewis, and V. Ermagan, "LISP Delegated Database Tree," *IETF draft-ietf-lisp-ddt-08*, 08-092016.

[21] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)," *IETF RFC 6836*, January, 2013 2013.

[22] S. Yang, H. Luo, Y. Qin, and H. Zhang, "Design and Evaluation of DNS as Location Manager for HIP," *Wireless Personal Communications,* journal article vol. 48, no. 4, pp. 605-619, March 01 2009.

[23] V. Ramasubramanian, E. G, #252, and n. Sirer, "The design and implementation of a next generation name service for the internet," *SIGCOMM Comput. Commun. Rev.,* vol. 34, no. 4, pp. 331-342, 2004.

[24] J. Laganier, "Host Identity Protocol (HIP) Registration Extension," *IETF Request for Comments 5203,* April, 2008 2008.

[25] U. A. E. Mungur, "Scalability of Locator Identity Split Mapping Infrastructure to Support End-Host Mobility," *PhD Dissertation,* Thesis vol. SCC, Lancaster University, 2012.

[26] R. Whittle. (2011). *Ivip - a scalable routing and mobility architecture for the IPv4 and IPv6 Internets*. Available: http://www.firstpr.com.au/ip/ivip/

[27] R. J. Atkinson and S. N. Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description," Internet Research Task Force (IRTF) Request for Comments: 6740November 2012.

[28] R. Moskowitz and P. Nikander. *Host Identity Protocol (HIP) Architecture", RFC 4423 May 2006*. Available: http://www.rfc-editor.org/info/rfc4423

[29] A. A. Aderemi, K. L. Mathew, K. O. Martha, and O. I. Juliet, "3GPP Long Term Evolution: Architecture, Protocols and Interfaces," *International Journal of Information and Communication Technology Research,* vol. 1, no. 7, pp. 306 - 310, Novermber 2011.

[30] B. A. Forouzan, *Data communications and networking*, 5th ed. New York, NY: McGraw-Hill, 2013, pp. xxxvii, 1226 p.

[31] J. G. Andrews and A. Gatherer, "Femtocell networks: a survey," *IEEE Communications Magazine,* vol. 46, no. 9, pp. 59 - 67, September, 2008.

[32] C. B. Sankaran, "Data offloading techniques in 3GPP Rel-10 networks: A tutorial," *Communications Magazine, IEEE,* vol. 50, no. 6, pp. 46-53, 2012.

[33] (2016). *Get wireless Internet | Find wi-fi hotspots | BT Wi-fi*. Available: http://www.btwifi.com/

[34] M. Corici, D. Vingarzan, and T. Magedanz, "3GPP Evolved Packet Core - the Mass Wireless Broadband All-IP Architecture," in *IEEE Telecommunications: The Infrastructure for the 21st Century (WTC), 2010*, 2010.

[35] A. de la Oliva, C. J. Bernardos, M. Calderon, T. Melia, and J. C. Zuniga, "IP flow mobility: smart traffic offload for future wireless networks," *Communications Magazine, IEEE,* vol. 49, no. 10, pp. 124-132, 2011.

[36] H. Tuncer, S. Mishra, and N. Shenoy, "A survey of identity and handoff management approaches for the future Internet," *Computer Communications,* vol. 36, no. 1, pp. 63-79, 12/1/ 2012.

[37] "Standard for Local and metropolitan area networks – Part 21: Media Independent Handover Services. Amendment 1: Security Extensions to Media Independent Handover Services and Protocol.," ed: IEEE 802.21a, 2012.

[38] P. Jokela *et al.*, "Handover performance with HIP and MIPv6," in *Wireless Communication Systems, 2004, 1st International Symposium on*, 2004, pp. 324-328: IEEE.

[39] H. Tuncer, A. Kwasinski, and N. Shenoy, "Performance analysis of Virtual Mobility Domain scheme vs. IPv6 mobility protocols," *Computer Networks,* vol. 57, no. 13, pp. 2578-2596, 2013.

[40] C. Mugga, D. Sun, and D. Ilie, "Performance Comparison of IPv6 Multihoming and Mobility Protocols," presented at the ICN 2014: The Thirteenth International Conference on Networks, Nice, France, 2014.

[41] J. Kempf, "Goals for Network-Based Localized Mobility Management (NETLMM)," *IETF RFC 4831,* April, 2007 2007.

[42] A. Biernacki and K. Tutschku, "Performance of HTTP video streaming under different network conditions," *Multimedia Tools and Applications,* journal article vol. 72, no. 2, pp. 1143-1166, 2013.

[43] M. Gorius, S. Yongtao, and T. Herfet, "Dynamic media streaming over wireless and mobile IP networks," in *Consumer Electronics - Berlin (ICCE-Berlin), 2012 IEEE International Conference on*, 2012, pp. 158-162.

[44] Y. Elkhatib, G. Tyson, and M. Welzl, "Can SPDY really make the web faster?," in *Networking Conference, 2014 IFIP*, 2014, pp. 1-9.

[45] "Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats," *International Standard ISO/IEC 23009-1:2014 Second Edition,* 2014.

[46] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," 2007.

[47] D. Massey, L. Wang, B. Zhang, and L. Zhang, "A Proposal for Scalable Internet Routing & Addressing," 2007-02-27 2007.

[48] F. T. <fred.l.templin@boeing.com>, "IPvLX - IP with virtual Link eXtension," 2007-05-11 2007.

[49] M. Bagnulo and E. Nordmark, "Shim6: Level 3 Multihoming Shim Protocol for IPv6," June, 2009 2009.

[50] J. Shoch, "Inter-network naming, addressing, and routing," in *COMPCON, IEEE Computer Society, Fall*, 1978, vol. 5.

[51] N. J. Chiappa, "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture," Accessed on: 29/06/2017Available: http://mercury.lcs.mit.edu/~jnc/tech/endpoints.txt

[52] M. O'Dell, "GSE - An Alternate Addressing Architecture for IPv6," Network Working Group, Internet Draft1997.

[53] "Re: [RRG] GSE History," 2008, Available: http://www.ietf.org/mail-archive/web/rrg/current/msg02455.html, Accessed 02/03/2016.

[54] J. Pan, S. Paul, R. Jain, and M. Bowman, "MILSA: A Mobility and Multihoming Supporting Identifier Locator Split Architecture for Naming in the Next Generation Internet," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, 2008, pp. 1-6.

[55] M. Kunishi, M. Ishiyama, K. Uehara, H. Esaki, and F. Teraoka, "LIN6: A new approach to mobility support in IPv6," in *Proceedings of the Third Inernational Sympsium on Wireless Personal Multimedia Communications*, 2000, vol. 43.

[56] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: Reorganizing the addressing architecture," in *ACM SIGCOMM Computer Communication Review*, 2003, vol. 33, no. 4, pp. 313-321: ACM.

[57] R. Inayat, R. Aibara, K. Nishimura, T. Fujita, Y. Nomura, and K. Maeda, "MAT: an end-to-end mobile communication architecture with seamless IP handoff support for the next generation internet," *Web and Communication Technologies and Internet-Related Social Issues—HSI 2003,* pp. 171-171, 2003.

[58] (2014, 22/12/2013). *LISPmob - LISP Mobile Node Implementation*. Available: https://github.com/arnatal/lispmob

[59] H. Li *et al.*, "User ID routing architecture," *IEEE Vehicular Technology Magazine,* vol. 5, no. 1, pp. 62-69, 2010.

[60] X. Xu and D. Guo, "Hierarchical routing architecture (HRA)," in *Next generation internet networks, 2008. NGI 2008*, pp. 92-99: IEEE.

[61] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "The Locator/ID Separation Protocol (LISP)," *IETF RFC 6830,* January, 2013.

[62] D. Meyer, "The Locator Identifier Separation Protocol (LISP)," *The Internet Protocol Journal,* vol. 11, no. 1, pp. 23-36, March 2008.

[63] F. Maino, V. Ermagan, A. Cabellos, and D. Saucez, "LISP-Security (LISP-SEC) draft-ietf-lisp-sec-09," *IETF Internet-Draft*, Oct. 2015.

[64] D. Farinacci, D. Lewis, D. Meyer, and C. White, "LISP Mobile Node," *IETF draft-ietf-lisp-mn-00*, 28-04-2017.

[65] D. Farinacci, C. White, D. Lewis, and D. Meyer, "LISP Mobile Node,draft-meyer-lisp-mn-14." doi: urn:ietf:id:meyer-lisp-mn Available: https://tools.ietf.org/html/draft-meyer-lisp-mn-14

[66] A. Rodríguez Natal *et al.*, "LISP-MN: Mobile Networking Through LISP," (in English), *Wireless Personal Communications,* vol. 70, no. 1, pp. 253-266, 2013/05/01 2013.

[67] M. Isah and C. Edwards, "Inter-domain Mobility with LISP-MN--A Performance Comparison with MIPv6," in *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2015, pp. 80-87: IEEE.

[68] M. Menth, D. Klein, and M. Hartmann, "Improvements to LISP Mobile Node," in *Teletraffic Congress (ITC), 2010 22nd International*, 2010, pp. 1-8.

[69] M. Gohar and K. Seok-Joo, "Distributed handover control in localized mobile LISP networks," in *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP*, 2011, pp. 1-7.

[70] J. Hou, Y. Liu, and Z. Gong, "Performance Analysis of a Seamless Mobility Support Scheme in the Locator/Identifier Separation Network," in *Computer Science for Environmental Engineering and EcoInformatics: International Workshop, CSEEE 2011, Kunming, China, July 29-31, 2011. Proceedings*, Y. Yu, Z. Yu, and J. Zhao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 159-165.

[71] C. Nakjung, T. You, J. Park, K. Taekyoung, and Y. Choi, "ID/LOC separation network architecture for mobility support in future internet," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, 2009, vol. 01, pp. 78-82.

[72] R. Whittle, "Ivip (Internet Vastly Improved Plumbing) Architecture," *IETF Internet-Draft, draft-whittle-ivip-arch-04,* 2010-03-08 2010.

[73] R. Whittle and S. Russert, "TTR Mobility Extensions for Core-Edge Separation Solutions to the Internet's Routing Scaling Problem," ed: August, 2008.

[74] T. Li, "Recommendation for a Routing Architecture," *Internet Research Task Force (IRTF), Request For Comments: 6115*, February, 2011.

[75] "IEEE Guidelines for 64-bit Global Identifier (EUI-64)," in *Tutorial*, ed: IEEE.

[76] R. J. Atkinson and S. N. Bhatti, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations," November, 2012 2012.

[77] S. N. Bhatti, R. J. Atkinson, and J. Klemets, "Integrating challenged networks," in *MILITARY COMMUNICATIONS CONFERENCE (MILCOM)*, 2011, pp. 1926 - 1933.

[78] R. J. Atkinson and S. N. Bhatti, "ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)," RFC 6743, IETFNov 2012.

[79] B. Simpson and S. N. Bhatti, "An identifier-locator approach to host multihoming," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, 2014, pp. 139-147: IEEE.

[80] D. Phoomikiattisak, "Mobility as first class functionality: ILNPv6 in the Linux kernel," University of St Andrews, 2016.

[81] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks," *Communications Surveys & Tutorials, IEEE,* vol. 12, no. 2, pp. 186-204, 2010.

[82] P. Jokela, J. Melen, and R. Moskowitz, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)," April, 2015 2015.

[83] T. Henderson, "End-Host Mobility and Multihoming with the Host Identity Protocol," April, 2008 2008.

[84] A. Gurtov, *Host Identity Protocol (HIP): Towards The Secure Mobile Internet*. John Wiley & Sons, 2008.

[85] M. Stiemerling, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication," *IETF Request for Comments 5207,* April, 2008 2008.

[86] J. Woodyatt, "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service," *IETF Request For Comments 6092,* January, 2011 2011.

[87] S. Novaczki, L. Bokor, and S. Imre, "Micromobility support in HIP: survey and extension of host identity protocol," in *Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean*, 2006, pp. 651-654.

[88] P. Salmela and J. Melén, "Host Identity Protocol Proxy," in *E-business and Telecommunication Networks: Second International Conference, ICETE 2005, Reading, UK, October 3-7, 2005. Selected Papers*, J. Filipe, H. Coelhas, and

M. Saramago, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 126-138.

[89] M. M. Muslam, H. A. Chan, L. A. Magagula, and N. Ventura, "Network-based mobility and Host Identity Protocol," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, 2012, pp. 2395-2400.

[90] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," Network Working Group , Request for Comment: 3775June, 2004.

[91] "Open Wireless Movement," 2016, Available: https://openwireless.org/. Accessed 06/05/2016.

[92] A. Sathiaseelan *et al.*, "A feasibility study of an in-the-wild experimental public access wifi network," in *Proceedings of the Fifth ACM Symposium on Computing for Development*, 2014, pp. 33-42: ACM.

[93] "8 Cities with Free WiFi," 2016, Available: http://www.holidayextras.co.uk/travel-blog/wanderlust/8-cities-with-free-wifi.html, Accessed: 18/04/2016.

[94] N. Omheni, F. Zarai, M. S. Obaidat, and K.-F. Hsiao, "A novel vertical handoff decision making algorithm across Heterogeneous Wireless Networks," in *Computer, Information and Telecommunication Systems (CITS), 2014 International Conference on*, 2014, pp. 1-6: IEEE.

[95] A. Bhuvaneswari and E. G. D. P. Raj, "An overview of vertical handoff decision making algorithms," *International Journal of Computer Network and Information Security,* vol. 4, no. 9, p. 55, 2012.

[96] L. Tsungnan, W. Chiapin, and L. Po-Chiang, "A neural network based context-aware handoff algorithm for multimedia computing," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, 2005, vol. 2, pp. ii/1129-ii/1132 Vol. 2.

[97] S.-J. Wu, "Fuzzy-based handover decision scheme for next-generation heterogeneous wireless networks," *Journal of Convergence Information Technology,* vol. 6, no. 4, 2011.

[98] X. Haibo, T. Hui, and Z. Ping, "A novel terminal-controlled handover scheme in heterogeneous wireless networks," *Comput. Electr. Eng.,* vol. 36, no. 2, pp. 269-279, 2010.

[99] D. Phoomikiattisak and S. N. Bhatti, "Mobility as a first class function," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, 2015, pp. 850-859.

[100] V. Visoottiviseth and P. Ngamtura, "On the performance of MIPv6 and FMIPv6 based on real IPv6 applications over IEEE 802.11g testbeds," in *Communications and Information Technologies (ISCIT), 2010 International Symposium on*, 2010, pp. 1217-1222.

[101] Y. Li, H. Su, L. Su, D. Jin, and L. Zeng, "A Comprehensive Performance Evaluation of PMIPv6 over IP-Based Cellular Networks," in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, 2009, pp. 1-6.

[102] A. Zambelli. (2013, 14/04/2016). *H.265/HEVC Ratification and 4K Video Streaming*. Available: http://blogs.iis.net/alexzam/h-265-hevc-ratification-and-4k-video-streaming

[103] (2017, 18/05/2017). *Internet Service Provider*. Available: https://en.wikipedia.org/wiki/Internet_service_provider

[104] A. Sathiaseelan and J. Crowcroft, "LCD-Net: lowest cost denominator networking," *SIGCOMM Comput. Commun. Rev.,* vol. 43, no. 2, pp. 52-57, 2013.

[105] (2016, 13/12/2016). *LISP Beta-Network*. Available: http://www.lisp4.net/beta-network/

[106] (2014). *LISPmob - an open-source LISP implemantation for Linux, Android and OpenWRT*. Available: http://lispmob.org/

[107] (2016). *CBA - Broadband Communications Research Group - lisp*. Available: http://www.cba.upc.edu/lisp

[108] (2016). *Open Overlay Router: Programmable overlays*. Available: http://www.openoverlayrouter.org/

[109] (2012, 01/12/2013). *MIPL Mobile IPv6 for Linux from GO-Core Project at the Helsinki University of Technology*. Available: https://github.com/jlanza/umip

[110] (2013, 29/11/2013). *Nautilus6 Project Overview - Deployment of the Mobile Internet*. Available: http://www.nautilus6.org/

[111] (13/12/2013). *UMIP - Mobile IPv6 and NEMO for Linux*. Available: http://www.umip.org/

[112] I. S. Alsukayti, "The multihomed mobile network architecture," *Thesis (Ph.D.)--Lancaster University*, 2015.

[113] M. Yousaf, S. Bhatti, M. Rehan, A. Qayyum, and S. A. Malik, "An intelligent prediction model for generating LGD trigger of IEEE 802.21 MIH," in *Emerging Intelligent Computing Technology and Applications*: Springer, 2009, pp. 413-422.

[114] C. Ma, E. Fallon, and Y. Qiao, "VOSHM: a velocity optimized seamless handover mechanism for WiMAX networks," in *9th. IT & T Conference*, 2009, p. 21.

[115] P. Payaswini and D. H. Manjaiah, "Performance analysis of link going down threshold for Media Independent Handovers," in *Emerging Trends in Communication, Control, Signal Processing & Computing Applications (C2SPCA), 2013 International Conference on*, 2013, pp. 1-5.

[116] "Open Dot Twenty One - An open-source multiple-plaftorm IEEE 802.21 MIHF implementation," Available: http://hng.av.it.pt/odtone/index.html

[117] (2016, 18/01/2017). *Subversion Directory - lispd*. Available: http://scc-forge.lancs.ac.uk/svn-repos/isah-phd/lispd/trunk/

[118] (2016, 28/01/2017). *Location Server code*. Available: http://scc-forge.lancs.ac.uk/svn-repos/isah-phd/ais/trunk/ais3_react.c

[119] D. C. Schmidt, "Reactor: an object behavioral pattern for concurrent event demultiplexing and event handler dispatching," in *Pattern languages of program design*, O. C. James and C. S. Douglas, Eds.: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 529-545.

[120] S. Simpson. (2014, 22/09/2015). *Portable C event reactor — Single-Threaded Management of I/O Events*. Available: http://www.lancaster.ac.uk/~simpsons/software/pkg-react.htmlz.en-GB

[121] J. Mailnen. (2016). *Ubuntu Manpage: hostapd - IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*. Available: http://manpages.ubuntu.com/manpages/wily/man8/hostapd.8.html

[122] L. Fenneberg, N. Lutchansky, P. Savola, and M. Myllynen. (2016). *Ubuntu Manpage: radvdump - dump router advertisements*. Available: http://manpages.ubuntu.com/manpages/precise/man8/radvdump.8.html

[123] V. Fuller and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface," IETF, RFC 6833Jan. 2013.

[124] C. Perkins, "IP Encapsulation within IP," *IETF RFC 2004,* October,1996.

[125] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei, "Quantifying Skype user satisfaction," in *ACM SIGCOMM Computer Communication Review*, 2006, vol. 36, no. 4, pp. 399-410: ACM.

[126] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: when randomness plays with you," in *ACM SIGCOMM Computer Communication Review*, 2007, vol. 37, no. 4, pp. 37-48: ACM.

[127] "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control," *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004),* pp. 1-205, 2010.

[128] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?," in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, 2012, pp. 9-14: ACM.

[129] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, 2013, pp. 9-14: ACM.

[130] Y. Sani, A. Mauthe, and C. Edwards, "Modelling video rate evolution in adaptive bitrate selection," in *2015 IEEE International Symposium on Multimedia (ISM)*, 2015, pp. 89-94: IEEE.

[131] (2014, March 2014). *Microsoft Smooth Streamer - Open Source*. Available: https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming/

[132] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *2012 19th International Packet Video Workshop (PV)*, 2012, pp. 173-178: IEEE.

[133] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, 2012, pp. 89-94: ACM.

[134] M.-N. Garcia *et al.*, "Quality of experience and HTTP adaptive streaming: A review of subjective studies," in *Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on*, 2014, pp. 141-146: IEEE.

[135] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming," presented at the Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, Hong Kong, China, 2013.

[136] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2013, pp. 19-24: ACM.

[137] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," presented at the Proceedings of the ACM SIGCOMM 2011 conference, Toronto, Ontario, Canada, 2011.

[138] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114-118: ACM.

[139] Y. Sani, M. Isah, C. Edwards, M. Andreas, "Experimental evaluation of the impact of mobility management protocols on HTTP adaptive streaming", in IET Networks, 2017, DOI: 10.1049/iet-net.2016.0119

APPENDIX

Loc-server Code

```
// Standard headers
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <assert.h>
#include <errno.h>

// Unix/Linux/POSIX headers
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <linux/ipv6.h>

// Additional libraries
#include <ddslib/dllist.h>
#include <react/core.h>
#include <react/user.h>
#include <react/event.h>
#include <react/conds.h>
#include <react/condtype.h>
//#include "recordpacket.h"

#include "serial.h"

#define DATA_PORT 4341
#define CTRL_PORT 4342

/* Define structure types. */

/* A buffer is a block of memory containing a packet of a known size.
   The structure is an element of a singly-linked list. */
struct buffer {
  struct buffer *next; // next in list
  size_t size; // size of the packet
  void *data; // start of packet, allocated with malloc
};

struct server;

/* A destination represents an IPv6 peer who has registered with us
to
   store its packets. */
struct dest {
  struct server *srv; /* which server we belong to (although there's
                         only one) */
  struct dest *next; /* the next destination in the list held by the
                        server */
  struct in6_addr id; // the identifier of this destination
  struct in6_addr careof; // where to forward to, or zero if not set
  struct buffer *buffers; /* the first of buffer in a list of
received
                           packets*/
  struct buffer **tail;
  react_event expiry_event; /* an event for detecting if this
                             destination is still needed after a
while */
  struct timeval expiry; // when to stop needing this destination
};

/* A server listens for data packets and control packets, and keeps a
   list of destinations that have registered with it to store their
   packets. */
struct server {
  react_core core; // the reactor to use
  int ctrl_sock; // the control socket
  react_event ctrl_event; // the event for receiving on the control
  // socket
  int data_sock; // the data socket
  react_event data_event; // the event for receiving on the data
  // socket
  struct dest *dests; /* a list of destinations we are holding
packets
                       for*/
};

void list_dests(FILE *fp, struct server *srv)
{
  char idbuf[100], cobuf[100];
  for (struct dest *dp = srv->dests; dp != NULL; dp = dp->next) {
    inet_ntop(AF_INET6, &dp->id, idbuf, sizeof idbuf);
    inet_ntop(AF_INET6, &dp->careof, cobuf, sizeof cobuf);
    fprintf(fp, "  %p: %s -> %s\n", (void *) dp, idbuf, cobuf);
  }
}

static void dump(FILE *fp, const void *base, size_t len)
{
  for (size_t i = 0; i < len; i++) {
    if (i % 16 == 0)
      fprintf(fp, "%08zx:", i);
    fprintf(fp, " %02x", i[(unsigned char *) base]);
    if (i % 16 == 15)
      putc('\n', fp);
  }
  if (len % 16 != 0)
    putc('\n', fp);
}

static void on_data_packet(void *);
static void on_ctrl_packet(void *);

/* Indicate that we are ready to receive a control packet. */
static int prime_ctrl(struct server *srv)
{
  react_sockcond cond;
  cond.handle = srv->ctrl_sock;
  cond.mode = react_MIN;
  fprintf(stderr, "\nPriming to read control socket %d\n",
cond.handle);
```

```
  return react_prime(srv->ctrl_event, react_CSOCK, &cond);
}

/* Indicate that we are ready to receive a data packet. */
static int prime_data(struct server *srv)
{
  react_sockcond cond;
  cond.handle = srv->data_sock;
  cond.mode = react_MIN;
  return react_prime(srv->data_event, react_CSOCK, &cond);
}

/* Indicate that we want to be called back in a while. */
static int prime_expiry(struct dest *dst)
{
  return react_prime(dst->expiry_event, react_CTIMEVAL, &dst-
>expiry);
}

/* Initialize the server to listen for data packets on data, control
   packets on ctrl, and to use core as a reactor. */
static int init_server(struct server *srv, react_core core,
                       int ctrl, int data)
{
  /* Set everything to safe defaults. */
  srv->core = core;
  srv->ctrl_sock = ctrl;
  srv->data_sock = data;
  srv->dests = NULL;
  srv->ctrl_event = srv->data_event = react_ERROR;

  // Now start allocating stuff.  If we fail, we just jump to a point
  // where we tidy everything up.

  /* Create event handles for receiving packets. */
  if ((srv->ctrl_event = react_open(core)) == react_ERROR)
    goto fail;
  if ((srv->data_event = react_open(core)) == react_ERROR)
    goto fail;

  fprintf(stderr, "Control socket: %d\n", srv->ctrl_sock);
  fprintf(stderr, "   Data socket: %d\n", srv->data_sock);

  /* Specify what action to take when these events trigger. */
  react_direct(srv->ctrl_event, &on_ctrl_packet, srv);
  react_direct(srv->data_event, &on_data_packet, srv);

  /* Specify that we are ready to receive packets. */
  if (prime_ctrl(srv) != 0 || prime_data(srv) != 0)
    goto fail;

  return 0;

 fail:
  react_close(srv->ctrl_event);
  react_close(srv->data_event);
  return -1;
}

static const struct in6_addr EMPTY_ADDR = IN6ADDR_ANY_INIT;
```

```
/* Compare two IPv6 addresses, and return zero if they match. */
static int cmp_v6(const struct in6_addr *p1, const struct in6_addr
*p2)
{
  char buf1[100], buf2[100];
  inet_ntop(AF_INET6, p1, buf1, sizeof buf1);
  inet_ntop(AF_INET6, p2, buf2, sizeof buf2);
  int rc = memcmp(p1, p2, sizeof(struct in6_addr));
  return rc;
}

/* Determine whether a care-of address has been set. */
static int careof_set(struct dest *dst)
{
  return cmp_v6(&dst->careof, &EMPTY_ADDR) != 0;
}

static struct dest **find_dest(struct dest **ptr, const struct
in6_addr *id)
{
  while (*ptr && cmp_v6(id, &(*ptr)->id))
    ptr = &(*ptr)->next;
  return ptr;
}

static void on_dest_expiry(void *);

static struct dest *create_dest(struct server *srv, const struct
in6_addr *id)
{
  struct dest *dst = malloc(sizeof *dst);
  if (!dst) return NULL;

  /* Initialize all fields to safe values. */
  dst->srv = srv;
  dst->next = NULL;
  dst->id = *id;
  dst->careof = EMPTY_ADDR;
  dst->buffers = NULL;
  dst->tail = &dst->buffers;
  dst->expiry_event = react_ERROR;

  /* Now start allocating resources, tidying everything up on
     failure. */
  dst->expiry_event = react_open(dst->srv->core);
  if (dst->expiry_event == react_ERROR)
    goto fail;
  react_direct(dst->expiry_event, &on_dest_expiry, dst);
  return dst;

 fail:
  react_close(dst->expiry_event);
  free(dst);
  return NULL;
}

/*If we don't know the care-of-address we need to store
  the packet in the right dest structure. Hence we write
  a function to achieve that.
*/
```

```
int record_packet(struct dest *dst, const void *buff, size_t len)
{
  struct buffer *entry = malloc(sizeof *entry);
  if (!entry) return -1;

  // Allocate and copy packet.
  entry->size = len;
  entry->data = malloc(len);
  if (!entry->data) {
    free(entry);
    return -1;
  }
  memcpy(entry->data, buff, len);

  // Link into struct dest.
  entry->next = NULL;
  *dst->tail = entry;
  dst->tail = &entry->next;
  assert(*dst->tail == NULL);

  return 0;
}

union {
  unsigned char raw[65536];
  struct {
    unsigned N : 1;
    unsigned L : 1;
    unsigned E : 1;
    unsigned V : 1;
    unsigned I : 1;
    unsigned flags : 3;
    unsigned noncemv : 24;
    unsigned instidlocst : 32;
    struct ipv6hdr v6hdr;
  } lispdata;
  struct {
    unsigned type : 4;
    unsigned p : 1;
    unsigned e : 1;
    unsigned s : 1;
    unsigned : 17;
    unsigned count : 8;
    unsigned long long nonce : 64;
  } lisphdr;
} packet_buffer;

static ssize_t send_a_packet(struct server *srv,
                             const struct in6_addr *daddr,
                             const void *buf,
                             size_t len)
{
  union {
    struct sockaddr_in6 in6;
    struct sockaddr gen;
  }dest;
  memset(&dest, 0, sizeof dest);
  dest.in6.sin6_family = AF_INET6;
  dest.in6.sin6_addr = *daddr;
  dest.in6.sin6_port = htons(DATA_PORT);
  for (int i = 0; i < 5; i++) {
    ssize_t done = sendto(srv->data_sock, buf, len, 0,
                          &dest.gen, sizeof dest.in6);
    if (done < len) {
      return done;
    }
    if (done >= 0) return done;
    if (errno == ECONNREFUSED)
      continue;
    perror("sendto");
    return done;
  }
  perror("sendto");
  return -1;
}

/* A packet is ready to read from the data socket. */
static void on_data_packet(void *vsrv)
{

  struct server *srv = vsrv;

  // Read the packet from src->data_sock into an array.
  union {
    struct sockaddr gen;
    struct sockaddr_in6 in6;
  } srvStore;
  socklen_t sStore = sizeof srvStore;;
  ssize_t nBytes;

  nBytes = recvfrom(srv->data_sock, packet_buffer.raw,
                    sizeof packet_buffer.raw, 0,
                    &srvStore.gen, &sStore);

  if (nBytes < 0) {
    perror("recvfrom failed");
    exit(4);
  }

  // Determine the destination address.
  struct in6_addr mn_eid = packet_buffer.lispdata.v6hdr.daddr;

  // Look for the destination in src->dests.
  struct dest **dstp = find_dest(&srv->dests, &mn_eid);

  if (*dstp == NULL) {
    // We have no such entry, so ignore the packet.
    fprintf(stderr, "   *** PACKET DROPPED ***\n");
  } else if (careof_set(*dstp)) {
    char locbuf[100];
    inet_ntop(AF_INET6, &(*dstp)->careof, locbuf, sizeof locbuf);

    //Just send the packet on to careof.
    //We use union to allow us to see the data in a packet as
    //both raw buffer as well as IPv6 header
    ssize_t done = send_a_packet(srv, &(*dstp)->careof,
                                 packet_buffer.raw, nBytes);
  }else{
    // Create a struct buffer, copy the packet contents into it,
    // and append it to the dest.
    /*If we don't yet know the care-of address, we need to store
      the packet in the right struct dest structure (which we've
      already found with find_dest)
    */

    if (record_packet(*dstp, packet_buffer.raw, nBytes) < 0) {
      perror("packet buffering failed");
      exit(6);
    }
  }

  // Reprime the event.
  int prc = prime_data(srv);
  assert(prc == 0);
}

static int decode_map_reply(size_t len, struct in6_addr *destid,
struct in6_addr *careof)
{
  dump(stderr, packet_buffer.raw, len);
  struct buf buf;
  buf_init(&buf, packet_buffer.raw, len);
  uint_least32_t word;
  buf_readu32(&buf, &word);
  unsigned msgtype = (word >> 28) & 0xfu;
  fprintf(stderr, "\nMessage type: %u\n", msgtype);
  if (msgtype != 2) return 0;
  unsigned nrecs = word & 0xffu;
  fprintf(stderr, "Record count: %u\n", nrecs);
  if (nrecs != 1) return 0;
  buf_readu64(&buf, NULL); // Skip nonce.

  // Now reading first record.
  buf_readu32(&buf, NULL); // Skip TTL.
  uint_least8_t nlocs;
  buf_readu8(&buf, &nlocs); // Locator count
  buf_readu8(&buf, NULL); // Skip EID mask-len.
  buf_readu16(&buf, NULL); // Skip ACT, A and reserved.
  buf_readu16(&buf, NULL); // Skip Map-Version number.
  uint_least16_t eidpfxafi;
  buf_readu16(&buf, &eidpfxafi);
  if (eidpfxafi != 2) {
    fprintf(stderr, "Not V6.\n");
    return 0;
  }
  buf_readin6addr(&buf, destid);
  fprintf(stderr, "EID Prefix:\n");
  dump(stderr, destid->s6_addr, 16);
  if (nlocs == 0) {
    fprintf(stderr, "no locator\n");
    return 1;
  }
  if (nlocs != 1) {
    fprintf(stderr, "unexpected number of locatord\n");
    return 0;
  }

  // Now reading first locator.
  buf_readu64(&buf, NULL);
  buf_readin6addr(&buf, careof);
  fprintf(stderr, "Locator:\n");
  dump(stderr, careof->s6_addr, 16);
  return 2;
}

static void on_ctrl_packet(void *vsrv)
{
  struct server *srv = vsrv;
  fprintf(stderr, "\n%s:%d on_ctrl_packet start\n", __FILE__,
__LINE__);
  list_dests(stderr, srv);

  // Read the packet from src->ctrl_sock into an array.
  union {
    struct sockaddr gen;
    struct sockaddr_in6 in6;
  } srvStore;
  socklen_t sStore = sizeof srvStore;;
  ssize_t nBytes;

  nBytes = recvfrom(srv->ctrl_sock, packet_buffer.raw,
                    sizeof packet_buffer.raw, 0,
                    &srvStore.gen, &sStore);

  if (nBytes < 0) {

    perror("recvfrom failed");
    exit(4);
  }

  // Determine what type of message it is.
  struct in6_addr destid = IN6ADDR_ANY_INIT, careof =
IN6ADDR_ANY_INIT;
  int msgtype = decode_map_reply(nBytes, &destid, &careof);

  if (msgtype == 1 /* It's an idx-to-NULL message. */) {
    char idbuf[100];
    inet_ntop(AF_INET6, &destid, idbuf, sizeof idbuf);
    fprintf(stderr, "\nLooking to initiate mapping for %s:\n",
idbuf);
    list_dests(stderr, srv);

    // Look up idx in srv->dests.
    struct dest **dstp = find_dest(&srv->dests, &destid);

    if (*dstp == NULL) {
      // No existing entry was found, so create a new struct dest,
and
      // link it into srv.
      *dstp = create_dest(srv, &destid);

      // Check that *dstp is not null.  If it is, we had a
      // problem setting it up, and probably should abort or
      // something.
      assert(*dstp != NULL);
      fprintf(stderr, "\n%s:%d created dest\n", __FILE__, __LINE__);
      list_dests(stderr, srv);
    }
    //  Check that *dstp is not null.  If it is, we had a
    // problem setting it up, and probably should abort or
    // something.
    assert(*dstp != NULL);
    // Compute a new expiry time for it, and set it to time out,
i.e.,
    // prime its expiry event.
    gettimeofday(&(*dstp)->expiry, NULL);
    (*dstp)->expiry.tv_sec += 20;
    int rc = prime_expiry(*dstp);
    assert(rc == 0);

  } else if (msgtype == 2 /* It's an idx-to-new_care_of message. */)
{
    // Look up idx in srv->dests.
    char idbuf[100], cobuf[100];
    inet_ntop(AF_INET6, &destid, idbuf, sizeof idbuf);
    inet_ntop(AF_INET6, &careof, cobuf, sizeof cobuf);
    fprintf(stderr, "\nLooking to complete mapping for %s->%s:\n",
            idbuf, cobuf);
    list_dests(stderr, srv);

    struct dest **dstp = find_dest(&srv->dests, &destid);
    if (*dstp != NULL) {
      struct dest *dst = *dstp;

      // Record the care-of address in case we want to keep this
      // record for a while.
      dst->careof = careof;

      // Send and delete all dst->buffers as packets.
      while (dst->buffers != NULL) {
        struct buffer *curr = dst->buffers;

        // Send data in 'curr' as UDP packet.
        ssize_t done = send_a_packet(srv, &dst->careof,
                                     curr->data, curr->size);
        assert(done == curr->size);

        dst->buffers = curr->next;
        free(curr->data);
        free(curr);
      }

      // Reprime the expiry of this dest, so it will be deleted after
      // a short while.
      gettimeofday(&dst->expiry, NULL);
      dst->expiry.tv_sec += 10;
      int rc = prime_expiry(dst);
      assert(rc == 0);
    }
  } else {
    // Warning message?
    fprintf(stderr, "msgtype:%d\n", msgtype);
  }

  // Reprime the event.
  int prc = prime_ctrl(srv);
  assert(prc == 0);
}

static void on_dest_expiry(void *vdst)
{
  struct dest *dst = vdst;
  struct server *srv = dst->srv;

  // Report to the user that this dest is being deleted.
  char idbuf[100];
  inet_ntop(AF_INET6, &dst->id, idbuf, sizeof idbuf);
  fprintf(stderr, "**** Expiring %s\n", idbuf);

  // Find out who points to us in the list.
  struct dest **dstp = &srv->dests;
  while (*dstp && *dstp != dst)
    dstp = &(*dstp)->next;
  assert(*dstp == dst); // We must find ourselves, or something went
                        // wrong!

  // Unlink from srv.
  *dstp = (*dstp)->next;

  // Delete the buffers without sending any packets.
  while (dst->buffers != NULL) {
    struct buffer *curr = dst->buffers;
    dst->buffers = curr->next;
    free(curr->data);
    free(curr);
  }

  // Delete dst and its remaining resources.
  assert(dst->buffers == NULL);
  react_close(dst->expiry_event);
  free(dst);

  // Don't reprime the event - it should have been
  // destroyed as part of the deletion of dst.
}

int main(int argc, const char *const *argv)
{
  react_core core;
  int data_sock = -1;
  int ctrl_sock = -1;

  struct server srv;

  data_sock = socket(PF_INET6, SOCK_DGRAM, 0);
  ctrl_sock = socket(PF_INET6, SOCK_DGRAM, 0);

  if (data_sock  < 0) {
    perror("creating socket");
    exit(1);
  }

  if (ctrl_sock  < 0) {
    perror("creating socket");
    exit(1);
  }
#ifdef V6ONLY
  {
    int opt = 1;
    if (setsockopt(sock, IPPROTO_IPV6, IPV6_V6ONLY, &opt,
sizeof(opt)) < 0) {
      perror("setting option IPV6_V6ONLY");
      exit(1);
    }
  }
#endif
  struct sockaddr_in6 addr;
  memset(&addr, 0, sizeof addr);
  addr.sin6_family = AF_INET6;
  addr.sin6_addr = in6addr_any;

  addr.sin6_port = htons(DATA_PORT);
  if (bind(data_sock, (struct sockaddr *) &addr, sizeof addr) < 0) {
    perror("bind(data)");
    return EXIT_FAILURE;
  }

  addr.sin6_port = htons(CTRL_PORT);
  if (bind(ctrl_sock, (struct sockaddr *) &addr, sizeof addr) < 0) {
    perror("bind(data)");
    return EXIT_FAILURE;
  }

  // Create the reactor.
  core = react_opencore(1);
  if (core == react_COREERROR) {
    perror("react_opencore");
    return EXIT_FAILURE;
  }

  // Initialize the server.
  if (init_server(&srv, core, ctrl_sock, data_sock) < 0) {
    perror("init_server");
    return EXIT_FAILURE;
  }

  // Now wait for something to happen, handle it, and wait again.
  for ( ; ; ) {
    if (react_yield(core) < 0) {
      perror("react_yield");
      break;
    }
  }
  return 0;
}
```