MOBILITY AS FIRST CLASS FUNCTIONALITY: ILNPv6 IN THE LINUX KERNEL

Ditchaphong Phoomikiattisak

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews

2015

Mobility as First Class Functionality: ILNPv6 in the Linux Kernel

Thesis by

Ditchaphong Phoomikiattisak

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

University of St Andrews
School of Computer Science

2015

Abstract

Mobility is an increasingly important aspect of communication for the Internet. The usage of handheld computing devices such as tablets and smartphones is increasingly popular among Internet users. However, the current Internet protocol, IP, was not originally designed to support mobility over the Internet. Mobile users currently suffer from connection disruption when they move around. Once a device changes point of attachments between different wireless technology (*vertical handoff*) e.g. from WiFi to 3G, the IP address changes, and the bound session (e.g. TCP session) breaks. While the IETF Mobile IPv4 (MIPv4) and Mobile IPv6 (MIPv6) solutions have been defined for some time, and implementations are available, they have seen little deployment due to their complexity and performance.

This thesis has examined how IP mobility can be supported as *first class functionality*, i.e. mobility can be enabled through the end hosts only, without changing the current network infrastructure. Current approaches such as MIPv6 require the use of proxies and tunnels which introduce protocol overhead and impact transport layer performance. The *Identifier-Locator Network Protocol (ILNP)* is an alternative approach which potentially works end-to-end, but this is yet to be tested. This thesis shows that ILNP provides mobility support as first class functionality, is implemented in an operating system kernel, and is accessible from the standard API without requiring changes to applications. Mobility management is controlled and managed by the end-systems, and does not require additional network-layer entities, only the end hosts need to be upgraded for ILNP to operate. This work demonstrates an instance of ILNP that is a superset of IPv6, called ILNPv6, that is implemented by extending the current IPv6 code in the Linux kernel. A direct performance comparison of ILNPv6 and MIPv6 is presented, showing the improved control and performance of ILNPv6, in terms of flow continuity, packet loss, handoff delay, and signalling overhead.

Declaration

I, Ditchaphong Phoomikiattisak, certify that this thesis, which is approximately 37,000 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

Date .... Signature of candidate ....

I was admitted as a research student in September 2012 and as a candidate for the degree of Doctor of Philosophy in September, 2015; the higher study for which this is a record was carried out in the University of St Andrews between 2012 and 2015.

Date .... Signature of candidate ....

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date .... Signature of supervisor ....

Copyright

Acknowledgements

To Prof. Saleem Bhatti, my supervisor, for your wisdom, guidance and support of everything.

To my parents, for your love, care, support, and many other things which cannot be explained here.

To Simon Dobson, my second supervisor, for all kind suggestions.

To Tristan Henderson, for your ‘straightforward’ reviews and comments that help lift up the standard of my work.

To Chusin Mateechaipong, my best friend, who always be there when I need.

To Chonlatee Khorakhun, Yuchen Zhao, Luke Hutton, Khawar Shehzad, my (ex)office-mates, for sharing ideas, happiness and time together.

To Bruce Simpson, for sharing tons of useful stuff for my research.

To the school system team and administration team, for your assistance in every tiny single matter.

Published Research

Some of the work presented in this thesis has been published.

D. Phoomikiattisak, S. N. Bhatti. Mobility as a First Class Function. Proceedings of *the 11th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob2015).* Abu Dhabi, UAE. Oct 2015.

S. N. Bhatti, D. Phoomikiattisak, R. J. Atkinson. Fast, Secure Failover for IP. Proceedings of *the 33rd IEEE Military Communications Conference (MILCOM 2014).* Baltimore, MD, USA. Oct 2014.

D. Phoomikiattisak, S. N. Bhatti. IP-Layer Soft Handoff Implementation in ILNP. Proceedings of *the 9th ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch2014).* Maui, Hawaii, USA. Sep 2014.

D. Phoomikiattisak, S. N. Bhatti. Network Layer Soft Handoff for IP Mobility. Proceedings of *the 8th ACM workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N2013).* Barcelona, Spain. Nov 2013.

Glossary

AD	– Administrative Domian
AH	– Authentication Header
API	– Application Program Interface
BAck	– Binding Acknowledgement
BTMM	– Back to My Mac
BU	– Binding Update
CAPEX	– Capital Expenditure
CGA	– Cryptographically Generated Addresses
CN	– Correspondent Node
CoA	– Care-of-Address
CoT	– Care-of Test
CoTI	– Care-of Test Init
DAD	– Duplicate Address Detection
DHCP	– Dynamic Host Configuration Protocol
DCCP	– Datagram Congestion Control Protocol
DMM	– Distributed Mobility Management
DNS	– Domain Name System
E2E	– End-to-End
EID	– Endpoint Identifier
ESD	– End System Designator
ESP	– Encapsulating Security Payload
ETR	– Egress Tunnel Router
EUI-64	– 64-bit Extended Unique Identifier
FA	– Foreign Agent

FMIPv6 – Fast Handover for Mobile IPv6

FN – Foreign Network

FQDN – Fully Qualified Domain Name

GL – Global Locator

GLI-Split – Global Locator, Local Locator, and Identifier Split

GSE – Global, Site, and End-system address elements

GSO – Generic Segmentation Offload

HA – Home Agent

HAHA – Home Agent to Home Agent

HAWAII – Handoff-Aware Wireless Access Internet Infrastructure

HI – Host Identifier

HIP – Host Identity Protocol

HMIPv6 – Hierarchical Mobile IPv6

HN – Home Network

HoA – Home Address

HoT – Home Test

HoTI – Home Test Init

IAB – Internet Architecture Board

ICMP – Internet Control Message Protocol

ID – Identifier

IEEE – Institute of Electrical and Electronics Engineers

IEN – Internet Experiment Note

IETF – Internet Engineering Task Force

IKE – Internet Key Exchange

ILCC – ILNP Communication Cache

ILNP – Identifier Locator Network Protocol

IL-V – Identifier Locator Vector

IoT – Internet of Things

IP – Internet Protocol

IPv4 – IP version 4

IPv6 – IP version 6

IPSec – Internet Protocol Security

IRTF – Internet Research Task Force

ISP – Internet Service Provider

ITR – Ingress Tunnel Router

L – Locator

L64 – 64-bit Locator

LCoA – Local Care-of-Address

LISP – Locator/Identifier Separation Protocol

LL – Local Locator

LMA – Local Mobility Anchor

LSR – Loose Source Routing

LTE – Long-Term Evolution

LU – Locator Update

LU-ACK – Locator Update Acknowledgement

MAC – Media Access Control

MAG – Mobile Access Gateway

MAN – Metropolitan Area Network

MAP – Mobility Anchor Point

MAS – Mobile Access Station

MILSA – Mobility and Multihoming Supporting Identifier Locator Split Architecture

MIP – Mobile IP

MIPv4 – Mobile IP version 4

MIPv6 – Mobile IP version 6

MN – Mobile Node

MOBIKE – IKEv2 Mobility and Multihoming Protocol

MP-TCP – Multipath TCP

MR – Mobile Router

M-SCTP – Mobile Stream Control Transmission Protocol

MSM-IP – Mobility Support Using Multicast in IP

MSS – Maximum Segment Size

MTU – Maximum Transmission Unit

NAR – New Access Router

ND – Neighbour Discovery

NAT – Network Address Translation

NEMO – Network Mobility

NID – Node Identifier

NR – Node Identity Router

OPEX – Operational Expenditure

OS – Operating System

PAR – Previous Access Router

PMIPv6 – Proxy Mobile IPv6

QoE – Quality of Experience

QoS – Quality of Service

RA – Router Advertisement

RANGI – Routing Architecture for the Next Generation Internet

RCoA – Regional Care-of-Address

RG – Routing Goop

RFC – Request for Comments

RLOC – Routing Locator

RM – Realm Manager

RO – Route Optimisation

RVS – Rendezvous Server

RTT – Round-trip Time

SA – Security Association

SBR – Site Border Router

SCTP – Session Initiation Protocol

SHIM6 – Level 3 Multihoming Shim Protocol for IPv6

SIP – Stream Control Transmission Protocol

SLAAC – Stateless Address Autoconfiguration

TCP – Transmission Control Protocol

TMIP – Terminal Independent Mobile IP

TSO – TCP Segmentation Offload

TTL – Time-to-Live

UDP – User Datagram Protocol

ULID – Upper Layer Identifier

VIP – Virtual Internet Protocol

VoIP – Voice over IP

VNA – Virtual Network Address

WAN – Wide Area Network

WLAN – Wireless Local Area Network

RFC document types

BCP – Best Current Practice

DS – Draft Standard

E – Experimental

I – Informational

PS – Proposed Standard

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of portable devices, including laptops, smartphones and tablets has been increasingly popular[1]. Moreover, a new range of products, such as Google glass, smart watches and other wearable devices have increasingly appeared, and might be widely used in the near future. Those devices usually prefer to stay connected to the network continuously. However, there is still no single wireless technology that provides full coverage with satisfactory quality of service (QoS). Hence, changes of point of network attachment between different wireless technologies when the users move is inevitable. Nevertheless, the current Internet architecture does not provide sufficient support for mobile devices. The widely used Internet Protocol, IP, was not designed for hosts that require frequent changes in point of attachment. Therefore, developing new architecture to enable host mobility support is highly desirable.

In this thesis, the term *mobile node (MN)*, defined as *"An IP node capable of changing its point of attachment to the network."* [70], will be used for representing portable/mobile devices in general.

1.1. Host Mobility

According to RFC3753 (I) [70], the term *host mobility support* refers to the *"function of allowing a mobile node to change its point of attachment to the network, without interrupting IP packet delivery to/from that node."* The term *handover* or *handoff*

[1]`http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/`

(the latter will be used throughout this thesis) is used to describe the action when an MN changes, or attempts to change, its point of attachment to the network. Generally, there are two types of handoff: *horizontal handoff* and *vertical handoff*. Horizontal handoff considers a scenario when an MN moves between access points of the same wireless type, such as from WiFi to WiFi. However, if the mobile node moves between access points of different wireless type, such as WiFi to 3G, the movement is considered as a vertical handoff. Also, a movement across administrative domains that results in a change of network layer topology is considered as a vertical handoff. An example scenario of horizontal handoff and vertical handoff is shown in Figure 1.1

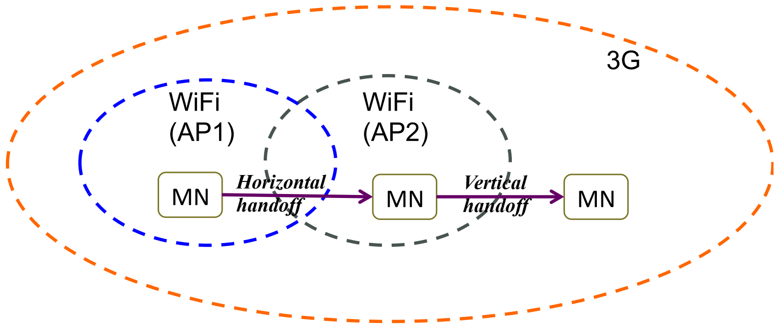


FIGURE 1.1. An example of handoff scenario. When a mobile node (MN) handoff under the same underlying technology, for example WiFi, it does a *horizontal handoff*. However, when it moves across wireless technology, for example WiFi to 3G, it performs a *vertical handoff*.

1.2. Importance of Network Layer Solutions to Host Mobility

As the use of mobile devices and methods of wireless connectivity continue to increase, *seamless mobility* becomes more desirable and important. *Seamless mobility* here means that a device with packet flows in progress can continue its flows even if there are changes to: (a) its physical connectivity (e.g. it moves from 3G to WiFi); or (b) its network layer (domain) connectivity (e.g. it changes from IP network A to IP network B, which might also occur when changing physical connectivity). While mobility using the same underlying technology – *horizontal handoff* (e.g. WiFi to WiFi) – is possible (usually through layer 1 or 2 handoff mechanisms), effective solutions for *vertical handoff* – across different wireless technologies and/or different networks are still being developed. While work is in progress to allow such transfer of communication sessions at the lower layers (e.g. the work by IEEE 802.21 Working Group[2]), the 'natural' place for such interworking across technologies is the network layer.

There are also potential solutions at the transport layer and the application layer. However, they usually have limitations and might not support mobility for every type of service. Transport layer solutions like *Stream Control Transmission Protocol (SCTP)* [110] and *Multipath TCP (MP-TCP)* [46, 47] provide mobility support for only specific transport protocols, which means they do not support applications that use different protocols such as UDP. Application layer solutions like *Session Initiation Protocol (SIP)* [101] provide mobility support by integrating infrastructure with some services, e.g. VoIP. However, not all types of services and applications can be supported by SIP. Therefore, the network layer is the most suitable place to tackle the host mobility issue for IP based applications.

The use of vertical handoff holds many significant challenges. The key issue is managing the change in network-layer connectivity. Typically, different physical connectivity, e.g. changes in network interface, also involve changes to network-layer connectivity – changes in the topological connectivity (*location*) of a device.

Seamless *vertical handoff* is highly desired because, currently (at the time of writing this thesis), there is still no single wireless technology, neither WiFi, 3G, LTE, nor others, that provide a full and ubiquitous coverage. Therefore, a combination of those technologies, and switching among them (i.e. vertical handoff), are indispensable. Hence, *an effective network layer solution is required.*

Seamless host mobility also could be a basis for various future functionalities. For example, it is one of the keys to enable *Ubiquitous Computing* or *The Internet of Things (IoT)*, when everything is always connected and communicating, as Weiser stated that *"sufficient infrastructure for highly mobile devices"* is required [118].

[2]`http://www.ieee802.org/21/`

1.3. Host Mobility and Network Mobility

Apart from host mobility, there is also another type of mobility, called *network mobility* or *site mobility*, where an entire network, instead of just a node, changes its point of attachment. Support of network mobility is usually based on support of host mobility. For example, the IETF NEMO [39] is an extension to Mobile IPv6. For ILNP, network mobility support is possible, essentially, based on the same mechanism that enables host mobility. However, this work focusses on only host mobility; investigation of network mobility is *not* included this thesis, and would be a separate and distinct body of research.

1.4. Challenges in Host Mobility Support

1.4.1. Challenges in Architecture.

(a). *Addressing.*

The key challenge to enable host mobility support is how to address an MN. The current IP address scheme has been identified as not suitable for MNs [22, 31, 74, 102] – see details in Section 2.1. The proposed mobility solutions use different mechanisms to handle the addressing problem. For example, Mobile IP uses two different IP addresses, Host Identity Protocol (HIP) uses public/private key scheme with an IP address, and Identifier Locator Network Protocol (ILNP) uses encoded identifier and locator values in an IPv6 address field. However, each has its advantages and disadvantages.

(b). *Network-based and host-based mobility management.*

Solutions to host mobility can be divided to *host-based* and *network-based* mechanisms, depending on whether the mobility management is done by the end hosts, or by additional network entities. Chapter 2 will provide more details of advantages and disadvantages of those two types of mobility management solutions. This thesis takes the position that host-based solutions are more incrementally deployable.

1.4.2. Challenges in Engineering.

(a). *Difficulty in implementation.*

Both host-based and network-based solutions have their own difficulty in implementation. Network-based solution requires new modules or new equipment in the core network, while host-based solutions need updates to the end-system protocol stack, which means modifying the operating system's kernel.

(b). *Backward compatibility.*

Since the current Internet architecture is quite mature, ideally, mobility support solutions should be backward compatible with the current infrastructure – no changes required for the current Internet landscape. Also, it is beneficial if the current applications need not to be modified to operate over a new architecture. Of course, some additional mechanisms may be required for interoperating between mobility-aware nodes and non-mobility-aware nodes.

(c). *Handoff performance.*

Many mobility management mechanisms have been proposed during the last few decades. However, none of them are widely adopted, partly due to a poor handoff performance. Most solutions use a *hard handoff* model or the *break-before-make* mechanism: the 'old' network connectivity is dropped before the 'new' network connectivity is initiated. Hence, there may be packets 'in flight' when handoff occurs, and so gratuitous packet loss occurs. Ideally, the *soft handoff* model or the *make-before-break* mechanism should be used: the 'new' connection is created before the 'old' one is disconnected. So, the MN can receive data from both old and new links during handoff, and gratuitous packet loss is minimised.

(d). *Quality of Service (QoS).*

A vertical handoff causes a difficulty in managing QoS of the ongoing communication session. An MN could move between networks with different QoS, e.g. different bandwidth, delay, loss and bit error rate. Therefore, even though a soft handoff solution could minimise gratuitous packet loss during handoff, the application could still have a problem of quality of experience (QoE), for instance, if the MN handoff is from a high speed link to a lower bandwidth, and higher delay link. So, the applications must be adaptable to the change of link quality. For example, there are previous works attempting to improve the QoE of the applications for Mobile IP [67], and for mobile network environments [76]. However, this is an application level issue and is not addressed in this thesis.

1.5. Thesis Outline

Thesis Statement: *IP layer mobility can be supported as 'first class functionality' by using end hosts only, without changing current network infrastructure*

At the moment, mobility support over multiple networks in the IP layer (network layer) is enabled by introducing additional entities, i.e. *Home Agent* and *Foreign Agent* in Mobile IP, which are *supplementary* to the network infrastructure. This is not only difficult to deploy, but also has problems in working with existing systems and protocols such as IPSec [61], Network Address Translation (NAT) [109] and Firewall. *Identifier Locator Network Protocol (ILNP)* [7–16, 18] has good potential to provide IP mobility as *first class functionality* with satisfying performance. Here, *first class functionality* means that mobility management is not supplementary to the architecture: it is controlled and managed by the end-systems, and can work with existing APIs and applications. ILNP could address the challenges mentioned above, as follows.

- Addressing – ILNP uses the concept of identifier/locator split to handle changes in point of attachment of a MN.
- Mobility management – ILNP is a *host-based* architecture, so no additional entities required to the network infrastructure, hosts signal each other directly.
- Backward compatibility – ILNP is backward compatible with the current IPv6 infrastructure, does not require changes to the current socket API (which means legacy applications can operate without changes), and has a mechanism to interoperate with non-ILNP nodes.

- Handoff performance – ILNP could use the *soft handoff* model, which minimises gratuitous packet loss during handoff. Nevertheless, the handoff performace in a real network environment would still need to be evaluated.

Although ILNP has a strong architectural concept, the engineering design for an operating system (OS) is not defined. There is no evidence that it can provide host mobility support in the real network, end-to-end, as part of a real OS. A comprehensive evaluation of any ILNP implementation has not been presented. This work aims to investigate performance of host mobility using ILNP in a real network, not an overlay network, simulation or emulation. The following are 3 research questions of this work to fill the gap between architectural concept and engineering.

(1) Implementation: How can ILNP mobility actually be built as an end-to-end functionality as part of an end-system's network stack? To be answered in Chapter 3 of the thesis.

(2) Performance: How good is the performance that ILNP can provide for mobility support from an application point of view, compared to MIPv6? To be answered in Chapter 4 and 5 of the thesis.

(3) Handoff Signalling: How does ILNP handoff signalling affect the network, compared to MIPv6? To be answered in Chapter 6 of the thesis.

1.6. Thesis Structure

Chapter 1 introduces the concept, importance and challenges of host mobility. The thesis outline and structure are also presented here.

Chapter 2 surveys the past and present state of host mobility support. The chapter starts with the problems of using the IP address in a host mobility environment. Then, a selection of current standardised mobility solutions are overviewed. Finally, descriptive reviews of MIPv6 and ILNP are presented – both are selected for in-depth evalaution in this work.

Chapter 3 shows how ILNPv6 could be implemented in the Linux kernel, using a dual stack approach. The chapter begins with a feasibility study using an overlay network to show that host mobility using ILNPv6 is possible. The rest of the chapter demonstrates how IPv6 in the Linux kernel could be enhanced to enable ILNPv6 functionalities.

Chapter 4 presents a performance comparison of the implemented ILNPv6 and MIPv6 using a UDP application. The purpose of this chapter is to show that handoff performance of ILNPv6 is better than MIPv6. This is evidence that ILNPv6 is able to provide mobility support over a real network. The chapter also explains how UDP in the Linux kernel could be extended to support ILNPv6.

Chapter 5 studies performance of TCP over ILNPv6 and MIPv6. This is to investigate how a handoff, by ILNPv6 and MIPv6, impacts TCP behaviour. Again, the evaluation here demonstrates that ILNPv6 is able to provide mobility support to TCP applications over a real network, with better performance than MIPv6. However, there may be some fine tuning required to TCP in order to optimise its performance when operating with ILNPv6.

Chapter 6 analyses the handoff signalling used in MIPv6 and ILNPv6. There are three analyses, which aim to verify: i) how does a lossy network impact handoff signals and handoff success rate; ii) what would happen if a signalling packet is lost; and iii) how much extra overhead do ILNPv6 and MIPv6 generate in the network when providing mobility for a high number of MNs. The study here shows that ILNPv6 handoff signalling is more efficient than MIPv6.

Chapter 7 summarises how the work has fulfilled the research questions towards host mobility mentioned earlier. There are also discussions of future work regarding deployment of ILNPv6 in the Internet.

Chapter 2

History of IP Mobility

This chapter surveys the history of IP mobility solutions from past to present. First, the problems of using IP address for MNs are described. Then a selection of mobility solutions is reviewed, and a comparison to ILNP is made where necessary.

2.1. Issues in Use of IP Addresses for Mobility

The reason that a special mechanism is required for host mobility support is that the use of IP addresses alone is insufficient. In fact, not only for host mobility, use of IP addresses also has problems with other IP functions such as multihoming, failover, concurrent sessions via multiple interfaces, and roaming, as Carpenter discusses [32]. Those problems share the same root cause: semantic overloading of the IP address, where a dynamic change of IP address affects the on-going communication session.

The fundamental problem for host mobility using IP can be explained by considering the bindings of an IP address within the protocol stack, as shown in Table 2.1 [12]. The second column of Table 1 shows that the IP address is used in state information for the transport layer, and it is also assigned to a specific physical interface: the IP address acts as an *identifier* at both the transport and physical layer. Effectively, a transport layer communication session is bound 1:1 to a physical interface. Also, if an application flow uses the IP address instead of a Fully Qualified Domain Name (FQDN) for its session state, then that application-layer session is also bound to a specific physical interface. Meanwhile, the IP address is used for routing at the network layer: the IP address acts as a topological *locator*. Hence, when an MN performs a handoff between different networks (e.g. switching from a WiFi interface to a 3G interface), it should, using this model, changes IP addresses, and the change of IP addresses results in the transport session state, and perhaps the application session state, becoming invalid.

Table 2.1. Use of names in IP.

Protocol layer	IPv4 and IPv6
Application	FQDN*, IP address
Transport	IP address
Network	IP address
(interface)	IP address

* FQDN: Fully Qualified Domain Name

In fact, these problems of the use of an IP address were known before the publication of [32]. It was mentioned in the first Internet Experiment Note (IEN) in 1977 [22] that the change of a local address of a host affects the TCP binding. The IEN also proposed, by implication, to separate an identifier of a host from the address.

Later on, in 1993, RFC 1498 (I) [102] discussed naming and binding of end systems. The essence of the document is that a node must preserve identity during communication: that is, the name used for identification and session state binding should not change even though the node changes point of attachment. This is clearly not true for an IP address today.

The Internet Architecture Board (IAB) emphasised the necessity of separating *identifier* and *locator* semantics from an IP address [31]. They defined an *identifier* as *"a bit string which is used throughout the lifetime of a communication session between two hosts"*, and a *locator* as *"a bit string which is used to identify where a particular packet must be delivered"*. Ideally, an identifier should never change, but a locator changes as the topology changes. However, neither IPv4 nor IPv6 can fulfill those two properties. The IAB also recommended that, as the transition from IPv4 to IPv6, it would be ideal if a node could be provided a unique identifier for upper layer end-to-end protocols. Moreover, one of the key findings pointed out in the IAB workshop in 2006 [74] is a problem of overloading of the IP address with the semantics of both identifier and locator. So, clearly, a separation of identifier and locator of a node is highly desirable for the future Internet.

2.2. Addressing, Location and Identity

As mentioned previously, the IP address was not designed to provide mobility support over the Internet. The 'misuse' of an IP address to represent both location and identity of a host has been considered for over a decade. Many past and ongoing works have agreed that separation of an IP address into two namespaces, representing *Location* and *Identity*, is necessary to solve mobility and other related issues. It can be seen in Section 2.4 that every mobility solution relies on a separation of identifier and locator, either by introducing a new set of addresses or by 'splitting' the current IP address.

A very early proposed new addressing scheme is an *8+8* scheme by Mike O'Dell [87]. This idea aimed to solve problems in multihomed networks since mobility was not widely considered at that time. The key idea of *8+8* is the splitting of the 16 byte IPv6 address into two objects: the *End System Designator (ESD)*, as the node's identifier, and the *Routing Goop (RG)*, as the node's locator. The ESD designates every interface of a system (hosts, router and other network equipment) in the *8+8* Internet, and its value is globally unique. The ESD may use the value of the IEEE EUI-64 [56] (deriving from the MAC address), or other schemes. The TCP protocol would use only the ESD to perform pseudo-header operation and session association. The second value, RG, represents a topological location of a computer system. When a host rehomes in a new network attachment point, for instance, in switching of an active link in multihoming sites or changing of an Internet Service Provider in single-homed sites, only the RG value is changed. The following mechanism is used to route the data to the appropriate destination:

- For routing within the same site, the ESD value is enough.
- For routing across different sites, the RG value, which encodes the topology information, is required.

However, there are some issues about this *8+8* scheme. First, the RG introduces a new structure of addressing, which requires routers to be updated. Second, the Domain Name System (DNS) needs to be updated in order to support the new address mapping by introducing new records: *AA* (for the ESP value) and *RG* (for the RG value). The *8+8* proposal was revised to another Internet-draft called *GSE (Global, Site, and End-system address elements)* [88]. Unfortunately, both the Internet-drafts of *8+8* and *GSE* were not adopted by the IETF, and they eventually expired. However, these are the motivations for many subsequent mobility solutions including ILNP.

2.3. IEN 135 – A Very First Host Mobility Solution

One of the early concerns in mobility support in the Internet is discussed in IEN 135 [113]. The purpose of the IEN is to allow hosts to move between networks in the Internet without disrupting the upper layer protocols (e.g. TCP) based on the issue when a host has to renew the IP address once it moves to a new network.

The IEN proposes a solution to allow MNs to have a special address called a *Virtual Network Address (VNA)*. This address must be reserved and must not be used for other purpose (e.g. use as a common IP address). The VNA of every MN should be unique under the same community of interest. This VNA will be used by the upper layer protocol such as TCP and it will never change.

In order to route the packets to an MN, a special system called a "forwarder" is introduced. Every mobile network has a forwarder which contains the information of the mapping of the VNA and the actual local address of every MN in such network. When a host wants to send a packet to the MN, the sender uses two-part source routing. The first part contains the normal IP address of the forwarder and the second part is the VNA of the MN. Once the packet reaches the forwarder, it maps the VNA to the local IP address and forwards the packet to the MN.

To support this mechanism, a global database that maintains the location of every MN is required. The database contains the mapping of the VNA of the MN and the IP address of its currently associated forwarder. This information is to be queried when a host wants to send a packet to any MN. The mapping information is required to be updated once an MN changes its location, that is, it has to send the IP address of the new associated forwarder to the database.

This idea was designed to be a very first thought of mobility support in the Internet, and was intended to be an inspiration to researchers to develop more advanced solutions.

2.4. Current Host Mobility Solutions

This section presents a selection of recent proposed solutions of host mobility support, focussing on those have been reviewed by the IETF or the IRTF because they usually i) have sufficient engineering detail to show the concept is possible, ii) are deployable, iii) are scalable, and iv) are backward compatible with the current architecture, as they go through the process [29]. Table 2.2 compares selected mobility solutions under different attributes. The solutions are categorised into two types: *network-based* solutions and *host-based* solutions. Network-based solutions refer to ones that *require additional network entities* for mobility management, while host-based solutions *do not necessary require additional network entities* to achieve host mobility.

TABLE 2.2. Comparison of Mobility Management Solutions

Attribute	MIPv4	MIPv6	PMIPv6	LISP	HIP	SHIM6	ILNP
Mobility management	Network-based	Network-based	Network-based	Network-based	Host-based	Host-based	Host-based
Additional infrastructure	HA & FA	HA	LMA & MAG	Mapping System	RVS (optional)	-	-
MN modification	Yes	Yes	No	Yes	Yes	Yes	Yes
Operating layer	L3	L3	L3	L3	L3 & ‘HIP’	L3 & ‘shim’	L3
MN address	HoA & CoA	HoA & CoA	HoA & CoA	EID & RLOC	HI & IP	ULID & L	NID & L64
Supported legacy address space	IPv4	IPv6	IPv6	IPv4/IPv6	IPv4/IPv6	IPv6	IPv4$^+$/IPv6
Concurrent multipath transfer	No	No	No	No	No	No	Yes
Tunelling	Yes	Yes	Yes	Yes	No	No	No
Standardisation	IETF (PS)	IETF (PS)	IETF (PS)	IETF (E)	IETF (PS)	IETF (PS)	IRTF (E)

$^+$ Technically possible, deployability unclear.

HA	Home Agent	MAG	Mobile Access Gateway	ULID	Upper Layer Identifier
FA	Foreign Agent	EID	Endpoint Identifier	L	Locator
HoA	Home Address	RLOC	Routing Locator	NID	Node Identifier
CoA	Care-of-Address	RVS	Rendezvous Server	L64	Locator
LMA	Local Mobility Anchor	HI	Host Identifier		

2.4.1. Network-based Mobility Solutions.

Mobility management in this type of solution is usually achieved by the use of additional entities (such as proxies or middleboxes). Sometimes, tunnelling is also used for communication between those entities, and perhaps with the MN. The advantages and disadvantages of network-based solutions are listed below.

Advantages

- Deployable – there is no changes required to the current routing scheme. So, the solutions could be deployed in the current IPv4/IPv6 network.
- Backward compatibility – non-mobile nodes do not have to be upgraded, and they can operate over the new infrastructure. In addition, some solutions are purely network-based i.e. mobility management is done by only additional network entities. So, end hosts do not need to be upgraded for enabling mobility support.

Disadvantages

- Complexity – the addition of new network entities adds complexity to the current network landscape.
- Routing performance – the data packets usually need to go through the proxies resulting in sub-optimal packet routes.
- Security – the proxies could become single points of failure, and they could offer an additional point for security and privacy attacks on the mobility mechanism.
- Overhead – the use of tunnelling increases packet overhead and potentially impacts the Maximum Transmission Unit (MTU) size that is available to the application.
- Extra expense – a requirement of new equipment increases the *capital expenditure (CAPEX)* and there is an overhead for operations, i.e. administration overhead for network and systems management, and so an impact on *operational expenditure (OPEX).*

(a). *Mobile IP and Extensions.*

The most well-known solution for host mobility support is Mobile IP, which is an IETF standard to provide mobility support for IPv4 (MIPv4 [91]), and IPv6 (MIPv6 [92]). MIPv4 uses a Home Agent (HA) and a Foreign Agent (FA) to map between a *Home Address (HoA)*, which acts as an identifier for the MN, and *Care-of-Address (CoA)*, which acts as a locator for the MN. MIPv6 eliminates the use of the FA and requires only the HA. MIPv4 uses tunnelling between HA and FA, while MIPv6 introduces a *Route Optimisation (RO)* mechanism to eliminate tunnelling and avoid sub-optimal packet routes. Section 2.5 gives more detail about MIPv4 and MIPv6.

Hierarchical Mobile IPv6 (HMIPv6) [108] is an extension to MIPv6. The goal of HMIPv6 is to reduce the handoff delay – the duration that an MN requires to complete the handoff process. HMIPv6 introduces a new entity, called the Mobility Anchor Point (MAP), to manage mobility of MNs in its local region. Each MN now has two types of CoA: the Regional CoA (RCoA) and the Local CoA (LCoA). The MN would register its RCoA with the HA. The RCoA acts as an ordinary CoA in MIPv6. Packets from a CN would travel through the HA to the MAP using RCoA. By mapping the RCoA to a proper LCoA, the MAP forwards the packets to the MN using the LCoA. When an MN performs a handoff within the local region, only the LCoA is changed and not the RCoA. The MN would update its new LCoA to the MAP. Since the RCoA remains the same, the handoff is hidden from the HA and the CN. This mechanism provides a faster handoff, lower overhead and reduces the burden at the HA. However, the new entity, MAP, increases complexity and costs for deployment, management and maintenance, as well as introducing an additional potential point for security and privacy attack.

Fast Handover for Mobile IPv6 (FMIPv6) [65] extends MIPv6 in order to minimise packet loss during handoff. A handoff in MIPv6 happens when an MN is no longer reachable by the previous subnet, and the connection to the new subnet is subsequently established. Even though the MN enters the new subnet before it disconnects from the previous subnet, it does not use the new connection until the previous subnet is no longer reachable. This is a hard handoff model and then cause packet loss problems. FMIPv6 allows an MN to detect that it has moved to new subnet when it is still connected to the previous subnet. The new CoA can also be formulated, which can be used immediately after the handoff occur (when it disconnects from the previous subnet). This reduces the handoff delay time and could then reduce gratuitous packet loss. To minimise gratuitous packet loss to nearly zero, FMIPv6 creates a tunnel between the *Previous Access Router (PAR)* and the *New Access Router (NAR)*. Any traffic arriving at the PAR would be forwarded to the MN via the NAR. The tunnel exists until the Binding Update process is completed and all traffic is routed to the NAR. Although this extension improves MIPv6 in terms of the handoff performance (gratuitous packet loss could be minimised to close to zero [58]), it requires a lot more signalling overhead and extra tunnelling between PAR and NAR. As with HMIPv6 and the use of the MAP, the extra interaction between PAR and NAR has similar issues.

Proxy Mobile IPv6 (PMIPv6) [52] is another extension of MIPv6. PMIPv6 enhances MIPv6 to be a complete network-based solution – mobility support is managed purely by network entities. So, MNs do not need any changes or upgrades to enable host mobility. PMIPv6 uses two network entities, a *Local Mobility Anchor (LMA)*, which is similar to an HA in MIPv6 and a *Mobile Access Gateway (MAG)*, which is used for managing mobility of MNs. Though an MN does not get involved in the mobility management process, it still has two IP addresses: an HoA and a CoA. An MN always uses only the HoA and believes that it is always in the home network. The CoA is used to reach the proper MAG, which is responsible for forwarding data and tracking movements of MNs on its link. When an MN moves between subnets, an associated MAG would update the CoA of the MN to the LMA using a *Proxy Binding Update* message. The traffic between MAG and LMA uses a bidirectional tunnel. As PMIPv6 uses the *hard handoff* model, there is the problem of gratuitous packet loss during handoff. This problem is minimise by applying a concept of FMIPv6 to PMIPv6 [122]. Again, additional tunnelling and signalling overhead are required to achieve this, and there are the general problems of a network entity using as a proxy/middlebox, as mentioned previously.

Distributed Mobility Management (DMM) extends the IETF standard protocols, i.e. the Mobile IP family, to solve the problems of having a centralised mobility management entity (e.g. HA in MIPv6). RFC 7333 (I) [33] summarises basic concepts and requirements for DMM. The DMM concept proposes the use of distributed anchors instead of a single, centralised one, to avoid network traffic traversing a single proxy via sub-optimal routes. Each anchor is ideally placed near the MNs for maximising performance. RFC 7429 (I) [68] provides information on how DMM could be applied to the current IETF standard protocols such as MIPv6, HMIPv6 and PMIPv6. It also presents a gap analysis between the current practices and the requirement in RFC 7333 (I). However, the work has yet to reach a suitable level of maturity, and would still suffer the usual drawbacks associated with the use of middleboxes (the multiple, distributed mobility anchor).

(b). *Locator/Identifier Separation Protocol (LISP).*
LISP [45] maps an IP address into two schemas: *Endpoint Identifiers (EIDs)* and *Routing Locators (RLOCs)*. The value of an IP address can be interpreted differently (i.e. either EIDs or RLOCs) during a communication. The EIDs are used within a *LISP site*, where end hosts reside. For outgoing packets from a LISP site, the original IP packet uses the EID of the destination host as a destination IP address. An Ingress Tunnel Router (ITR) consults the *Mapping System* and performs a mapping between EID and RLOC. The packet would then be encapsulated by another IP header (tunelling) using the value of the RLOC as the destination IP address, which will be used for routing and forwarding to the destination LISP site. When the packets arrive at the destination LISP site, an Egress Tunnel Router (ETR) verifies that the RLOC is a correct value (i.e. representing its own site), then strips the outer IP header and forwards the packet to the destination host based on the EID value presented in the original packet. In the case of a host changing its location, only the value of RLOCs are changed not EIDs. The mapping information in the *Mapping System* is required to be updated, so that the ITR/ETR can use the correct values.

Originally, LISP was designed to provide multihoming support. To enable host mobility, there are two different mechanisms.

- LISP mobile node (LISP-MN) [99] – each MN is modified to act as a LISP site i.e. the mapping between EIDs and RLOCs is done by the MN itself. However, a mapping system is still required to store the mapping data. The problem of LISP-MN is relatively long handoff delay, costing 1.5 RTT.
- LISP-ROAM [49] – mobility management is done by network devices including ITR, ETR, and the mapping system. So, the MN does not need modification. However, LISP-ROAM is still at an early stage. It has a problem of high handoff delay (around 5 seconds, reported in their initial evaluation), which cause connection interruption during such period.

2.4.2. Host-based Mobility Solution.

Mobility support using host-based solution usually does not require additional network entities. End hosts are responsible for handling mobility management. The advantages and disadvantages of host-based solutions are listed below.

Advantages

- Incrementally deployable – there are no changes required to the current network infrastructure and routing scheme. So, the solutions could be deployed in the current IPv4/IPv6 network without requiring additional entities or any modifications.
- No tunnelling – communication between hosts uses the current routing and forwarding scheme without tunnelling. So, routing performance is no worse than current classic IPv4 and IPv6. There is also no extra overhead from packet tunnelling.

Disadvantages

- End hosts modification – the end hosts need to be upgraded in order to provide mobility support, which sometimes could be complicated and difficult to implement, especially for older/embedded systems.

Considering advantages and disadvantages of network-based and host-based solutions, this thesis takes the position that host-based solutions, like ILNP, are a better choice. The main disadvantage of end hosts modification can be ameliorated through the use of over-the-air software updates. as is common for most OSs today.

(a). *Identifier Locator Network Protocol (ILNP).*
ILNP [7–16, 18] is an IRTF Experimental protocol. It has a host-based, end-to-end architecture, and is designed to support mobility (amongst other things). For this work, ILNP is selected for investigation for the possibility of end-to-end host mobility support. ILNP is a purely host-based solution, which has advantages over network-based solutions as mentioned earlier. ILNP also has advantages over other host-based solutions (those will be presented later). More information about ILNP is presented in Section 2.6.

(b). *Host Identity Protocol (HIP).*
HIP [78,84], and an updated version: HIPv2 [77], enables host mobility by separating identity of an MN from the IP address using public and private key pairs. The transport protocol and IPSec protocol use the public key as a Host Identifier (HI) for binding. While the IP address is used for only packet routing and forwarding. For handoff, the IP address (which is the *locator*) of the MN would change. The new IP address would be updated to the CNs to allow subsequent packets to be routed to the correct subnet. On the other hand, the value of HI needs not to be re-generated. Hence, the on-going end-to-end sessions (e.g. TCP sessions) can continue. HIP requires the deployment of public key infrastructure despite in a secure site network. This consumes a high computation load and could degrade performance of applications and network interfaces.

For session initiation, although DNS may be used (with a new resource record for HI) [85], a new network entity called a *Rendezvous Server (RVS)* is recommended to be deployed for better performance of handling frequenlty moving MNs [66].

Due to the complexity of modifying an end host to support HIP, there is a proposal to enhance HIP to be a network-based solution [80]. *HIP Proxy*, a new network entity, is introduced to provide mobility support to MNs. End hosts do not need modification, but the network-based solution has several problems as discussed earlier.

HIP requires a new API for applications [63], and hence does not work for legacy applications. The HIP-Aware Agent [54] could be used to allow legacy applications to operate over HIP.

(c). *Level 3 Multihoming Shim Protocol for IPv6 (SHIM6).*
SHIM6 [86] decouples *Locator (L)* and *Upper Layer Identifier (ULID)* from an IP address of an MN. ULID is used for session binding in upper layer protocols, and remains static during the communication session. However, L may change as an MN changes point of attachment (e.g. for mobility and multihoming scenarios). An *Update Request* is sent by the MN to the CNs to inform a change. An *Update Acknowledgement* is then sent back from the CNs. SHIM6 requires implementation of an extra 'shim' layer between the network layer and the transport layer to map between ULIDs and Ls. Extra signalling is also required during the session establishment process.

In addition, SHIM6 is not designed to enable mobility, but is aimed at multihoming. Mobility support could be possible [40], but there is a problem in high handoff delay. There are some works in optimisation, e.g. [79]. Mobility support for a multihomed mobile node is also possible for SHIM6 [1].

2.4.3. Transport Layer Solutions.

All solutions mentioned above operate at the *network layer*. There are also proposals for *transport layer* solutions to mobility. Those solutions are usually *host-based*, so they share the same advantages (incrementally deployable and no use of tunnelling) and disadvantages (end host modification required). Additional disadvantages of transport layer solutions are listed below.

- Limited choices of transport layer protocol – the host must use the transport protocol that has been modified, while network layer solutions, permit any applications/transport protocols to be used.
- Security – new security issues are found according to additional features introduced in the solutions, more details below.

(a). *Stream Control Transmission Protocol (SCTP).*
SCTP [110] allows a host to set up multiple paths – with multiple source and destination address combinations – between a source and a destination host. SCTP is designed for multihoming support. An early idea of using SCTP in mobility scenario can be found in [120]. Single host mobility (one-side mobile host) using SCTP can be achieved by dynamically adding and deleting addresses of active SCTP associations [112]. Special treatments are required to support two-side mobile hosts, for example the use of a *Cooperation Server*, a new network entity to maintain the binding addresses of MNs [44]. SCTP has some security issues, especially on denial-of-service and man-in-the-middle attacks, which are summarised in RFC5062 (I) [111].

(b). *Multipath TCP (MP-TCP).*
MP-TCP [46, 47] extends TCP and allows a main TCP session to have bindings to different addresses i.e. has multiple *subflows*. After a main TCP connection is established, MP-TCP allows a host to set up a new path (i.e. subflow) by using a TCP handshake with the *MP_JOIN* TCP option for identifying the main flow to join. Note that both end hosts must be MP-TCP capable.

The original goal of MP-TCP is enabling multiple path transport connections, i.e. for multihomed nodes. Mobility using MP-TCP could be achieved by dynamically adding and removing sub-flows when a host enters and exits a network [95]. MP-TCP is backwards compatible with classic TCP as well as with current applications without needing changes at the socket API. However, an extension to the API, allowing applications to aware of multiple paths transfer, may be beneficial [103].

Despite a capability of multihoming and mobility support, MP-TCP introduces new security threats, summarised in RFC6181 (I) [20]. The security risks are mostly from the arbitrary adding of subflows to the ongoing connection, which could cause, for example, denial-of-service and man-in-the-middle attacks. A new cross-path interference attack also has been identified [105]. This new attack allows people from one subflow to gain proprietary information (such as throughput, packet loss, and round trip time) of another subflow.

2.4.4. Other Approaches for Host Mobility.

Apart from those mentioned mechanisms, there are other approaches proposed to provide mobililty support over multiple networks. Recent surveys [50, 124] have summarised the mobility solutions during the last few decades. Some of them have been covered in this chapter, the others, which support *host mobility* are summarised here.

- Columbia [57] – introduces a new network entity called Mobile Support Station to manage host mobility. Another IP address, derived from a special prefix, is assigned to each MN as its identifier. Each subnet has a Mobile Support Station to detect movement of MNs.
- Virtual Internet Protocol (VIP) [114] – has similar concepts as MIP. A mobile host has two IP addresses: Virtual IP address (acts as identifier) and regular IP address (acts as locator). Mapping between those values is managed by a home agent at the home network.
- Loose Source Routing (LSR) Protocol [23] – uses two network entities: Mobile Router (MR) and Mobile Access Station (MAS). MR is similar to MIP's home agent, which assigns a home IP address to an MN. The MN obtains another IP address from MAS, similar to MIP's foriegn agent.
- Mobility Support Using Multicast in IP (MSM-IP) [81] – uses IP multicast to provide mobility support. Each MN has a unique multicast IP address. When the node moves, it asks the multicast router of the new network to join its multicast group, so the packet can be delivered via the new access router. However, there are scalability problems for multicast routing.
- Cellular IP [116], Handoff-Aware Wireless Access Internet Infrastructure (HAWAII) [96], and Terminal Independent Mobile IP (TIMIP) [51] – are designed to manage mobility in the local domain that is running MIP. Each

of them has special agents to manage mobility in the local domain. Like HMIPv6, the global IP address remains stable if the MN moves within the local domain.

- End-to-End (E2E) Communication [107] – use DNS to manage mobility. An MN updates its current IP address to the DNS as it moves. A new TCP Migrate option is introduced to allow IP addresses to be modified as the flow continues.
- IKEv2 Mobility and Multihoming Protocol (MOBIKE) [43] – is an extension to provide mobility support for the Internet Key Exchange (IKEv2). When an MN moves, a new IKE Security Association (derived from the new IP address), is sent to all CNs.
- Global Home Agent to Home Agent (HAHA) [117] – is an extension to MIP aiming to mitigate the triangle routing problem. All home agents (HAs) join an IP anycast group. An MN has a primary HA, which can be changed as it moves. Packets from CNs are delivered to the nearest HA, and then forwarded to the MN via its primary HA.
- Back to My Mac (BTMM) [35] – provides mobility support for Mac OS hosts. Each host has an IPv6 Unique Local Address (ULA) as an identifier, while an ordinary IP address is used as a locator. Mapping between these two values is done by DNS.
- Global Locator, Local Locator, and Identifier Split (GLI-Split) [73] – is an Identifier/Locator split approach. A unique feature of GLI-Split is it has two locator values: global locator (GL) and local locator (LL). GL is used for routing in the Internet, while LL is used inside edge networks (GLI-domain). Like other approaches, a separate identifier (ID) is used for upper layer protocols. Similar to ILNP, ID and L (could be GL or LL) values are encoded in the form of an IPv6 address to enable backward compatibility. Hence, end hosts need an upgrade to support this. Two mapping systems, global and local, are required to map IDs to GL and LL.
- Mobility and Multihoming Supporting Identifier Locator Split Architecture (MILSA) [89] – is another Identifier/Locator split approach. MILSA divides

hosts into a group of a *Realm*. There is a *Realm Manager (RM)* that manages IDs and Ls for the nodes in its realm. The mapping of ID and L is done by an *Identifier Sublayer*, a new layer between the network layer and transport layer, which retrieves mapping information from a certain RM.

- Routing Architecture for the Next Generation Internet (RANGI) [121] – has a hierarchical architecture which separates the network into multiple *Administrative Domians (ADs)*. It has similar concepts to HIP, i.e. the use of a cryptographic host ID. However, RANGI's host ID also combines with the AD's ID. While the locators are IPv6 look-alike addresses. The first 96 bits are a *Locator Domain Identifier*, which is an IPv6 prefix assigned by a provider, and the last 32 bits are the IPv4 address of the node.
- Node Identity Internetworking Architecture (NIIA) [3,104] – uses public/private key pairs to decouple node identity from an IP address, similar to HIP. The network is separated into different *Locator Domains*. Each node in the domain registers its FQDN, NIDs and locators to the *NID routers (NRs)*, a new network entity. NRs are used for session initiation and routing between domains.

2.5. Mobile IP

Mobile IPv4 (MIPv4) [91] and *Mobile IPv6 (MIPv6)* [92] are based on similar concepts. Mobility management is handled by allowing an MN to have two IP addresses: an HoA and a CoA. HoA represents an *identity* of the MN. It is used by upper layer protocols (such as TCP), and it always remains stable even though the MN moves across different networks. So, end-to-end connectivity can be maintained. CoA, on the other hand, represents a *location* of the MN. It can be changed based on the current location of the MN, and is used for routing and forwarding by routers. As mentioned before, additional network entities, HA and FA (the latter is for MIPv4 only), are required to perform mappings between those two addresses. The HA is represented by the HoA, and acts as a 'base station' of the MN. Any traffic destined to the MN must be passed through the HA. The HA forwards arriving packets to the MN using the CoA. For MIPv4, the CoA represents the FA, which is responsible for forwarding the packets to the MN. For MIPv6, thanks to the IPv6 *Stateless Address Auto-configuration* and *Neighbour Discovery*, an MN is able to configure its CoA without requiring the FA. So, the packets from the HA can be tunnelled directly to the MN. MIPv6 also has another advantage over MIPv4: better routing performance. MIPv6 has an RO feature allowing an MN to update its current CoA value to the CN using the *Binding Update (BU)* message. Therefore, the CN can directly communicate with the MN without passing through the HA.

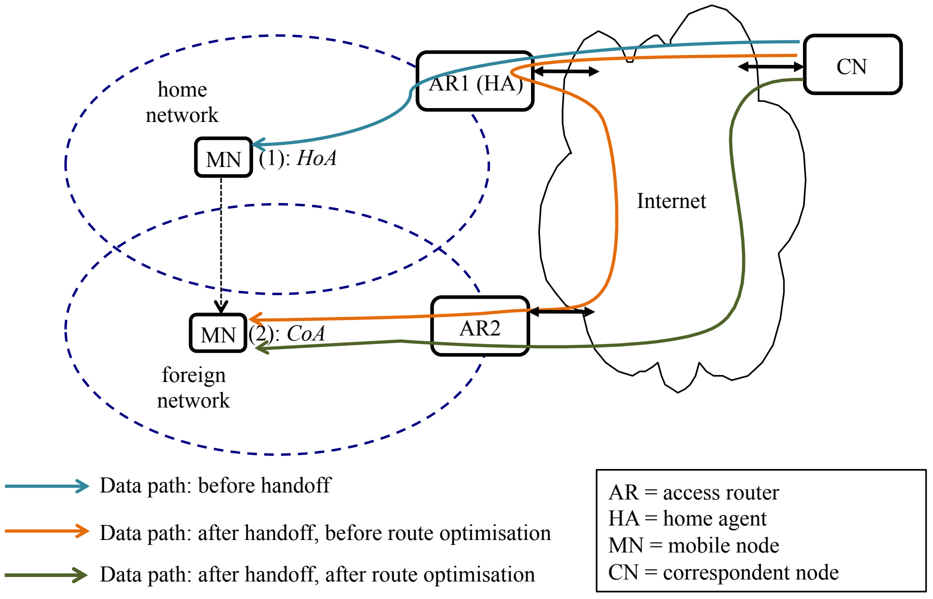


FIGURE 2.1. An example scenario of host mobility with MIPv6.

An example scenario of MIPv6 can be seen in Figure 2.1. Here MIPv6 is the focus rather than MIPv4 because, in this work, it is used as a direct performance comparison to ILNPv6.

(1) At the point (1) in Figure 2.1, the MN initially resides in the *Home Network (HN)*, which is managed by the HA, and uses only HoA.
(2) Any correspondent nodes (CNs) that wish to contact the MN would use the HoA as the destination address. Here, all packets could be delivered to the MN directly in the HN.

(3) The MN performs a *handoff*: it moves into a new site network, called the Foreign Network (FN), then moves out and disconnects from the HN (at position (2) in Figure 2.1).

(4) The MN obtains a CoA at the FN.

(5) The MN sends a BU to the HA to update its CoA.

(6) The HA registers the MN's CoA, and creates a tunnel between itself (HoA) and the MN (CoA). Then, a *Binding Acknowledgement (BAck)* is sent back to the MN.

(7) Now, any packets from CNs first travel to the HA because the CN still uses the HoA for communication. The HA would forward all packets to the MN via the tunnel.

(8) The packets from the MN to the CN can be sent directly from the MN to the CN – they do not need to pass through the HA.

(9) When RO is enabled, the MN would then update its CoA to the CN as well. There are two steps to do this: i) *Return Routability* procedure – the MN sends *Home Test Init (HoTI)* and *Care-of Init (CoTI)* to the CN for testing reachability of the HoA and CoA. The *Home Test (HoT)* and *Care-of Test (CoT)* would be sent back from the CN; ii) *Binding Update* – similar to an update to the HA, a BU is sent from the MN to the CN to inform them of the MN's new CoA, a BAck would be received back if the process is successful.

Although MIPv6 could enable host mobility, its major problem is the use of a *hard handoff* model – the 'old' network connectivity is dropped before the 'new' network connectivity is initiated. Therefore, gratuitous packet loss could be observed during the handoff because some packets are sent to the incorrect location. This problem can be clearly seen in the performance analysis in Chapter 4 and 5. Many of the extensions and updated for MIPv6 (e.g. FMIPv6 and HMIPv6) described in Section 2.4 are designed to deal with the handoff problem.

2.6. Identifier Locator Network Protocol (ILNP)

This section describes the *Identifier Locator Network Protocol (ILNP)*, the protocol that is selected for investigating the possibility of seamless host mobility support.

ILNP is a promising solution to enable seamless host mobility in the future, thanks to several key advantages.

(1) ILNP is a completely end-to-end solution. It does not require any new network entities or any upgrades for current network infrastructure. Only end systems' network stacks need to be modified.
(2) ILNP allow a *network layer soft handoff*, which, in theory, minimises gratuitous packet loss when an MN changes locations.
(3) ILNP is potentially scalable. It should have low signalling overhead, which would not impact overall performance of the network despite a high number of MNs.
(4) ILNP integrates features for solving various other issues, including network mobility [98], multihoming [106], traffic engineering [5] and virtual machine migration [25] (not evaluated in this thesis).

2.6.1. ILNP Architectural Concepts.

An architectural concept of ILNP [12] is deprecating the use of IP addresses and uses two new distinct namespaces: a *Node Identifier (NID)* and a network *Locator (L64)*, along with dynamic bindings to implement various functionalities, including host mobility. As shown in Table 2.3, instead of using the IP address in various layers across the protocol stack, ILNPv6 use NID and L64 values. Transport-layer protocols bind only to a NID value, an identifier for a (logical, virtual or physical) node, that has no topological semantics. This is to maintain end-to-end invariance for transport protocol session state. The NID represents an identity of a node which is tied to the whole node, not its interface. The network layer uses a L64 value, which is topologically significant, for routing and forwarding. The value of the L64 represents a single IP sub-network. An $[I, L]$ pair together – an *Identifier Locator Vector (IL-V)* – is needed for communication.

In addition, there are one-to-many dynamic bindings between NID and L64 values, as well as another set of dynamic bindings between physical interfaces and L64 values. Hence, mobility in ILNP is implemented by adjusting these dynamic bindings between NID and L64 values, and between L64 values and interfaces. The L64 values

Table 2.3. Use of names in IP and ILNP (extended version of Table 2.1).

Protocol layer	IPv4 and IPv6	ILNP (ILNPv6)
Application	FQDN*, IP address	FQDN* or app.-specific
Transport	IP address	Node Identifier (NID)
Network	IP address	Locator (L64)
(interface)	IP address	dynamic binding

* FQDN: Fully Qualified Domain Name

can change as an MN moves without impacting end-to-end state invariance, as the NID value always remains stable. MNs can have multiple NID and L64 values and use multiple interfaces simultaneously, by adjusting dynamic bindings between them as required.

2.6.2. ILNPv6 – An Engineering of ILNP.

While a clean-slate approach to building ILNP is possible, to aid deployment, ILNP could be implemented as a superset of IPv6, called ILNPv6 [13]. In order to carry the NID and L64 values between end-systems, the IPv6 packet header is used, with NID and L64 values being encoded into the IPv6 header as shown in Figure 2.2. The NID value uses the same syntax as an IPv6 interface identifier, but the end-system code is updated to treat it as a node identifier. The NID values need not be globally unique, but must be unique within the same sub-network. The value can be derived from IEEE 802 48-bit MAC address or IEEE EUI-64 identifier [56], which is similar to the ways that IPv6 Interface IDs are chosen [55]. Other methods, such as cryptographically generated addresses (CGA) [19] or privacy extensions for IPv6 addresses [82] could be used also. The L64 has the same syntax and semantics as an IPv6 prefix, and so current IPv6 routing and forwarding can be used for ILNPv6 packets: routers see and process ILNPv6 packets as IPv6 packets. Therefore, ILNPv6 could be deployed over current IPv6 network infrastructure without requiring changes.

2.6.3. Host Mobility with ILNP.

The initial ideas of host mobility using ILNP can be found in [6, 7, 17]. There are two cases of mobility: host mobility and site mobility; both can be supported by ILNP.

```
/* IPv6 - RFC4291 + RFC3587 */
|           64 bits           |         64 bits         |
+-----------------------------+-------------------------+
| IPv6 Unicast Routing Prefix |  Interface Identifier   |
+-----------------------------+-------------------------+

/* ILNPv6 - RFC6741 */
|           64 bits           |         64 bits         |
+-----------------------------+-------------------------+
|        Locator (L64)        |  Node Identifier (NID)  |
+-----------------------------+-------------------------+
```

FIGURE 2.2. IPv6 unicast address format and ILNPv6 unicast address format. The L64 value has the same syntax and semantics as the IPv6 routing prefix. The NID value has the same syntax as the IPv6 Interface Identifier, but has different semantics.

This work, however, considers only the host mobility case, i.e. when individual nodes are mobile. For mobility, there are two phases of communication:

- *Rendezvous:* For a CN to *initiate* communication with an MN, it must learn the current IL-V for the MN. With this information, the CN can then create a packet to send to the MN. If the MN functions only as a client system and the CN as a server, then there is no issue.
- *Handoff:* When a communication session is in progress, the session must be transferred across an administrative or network connectivity boundary in order to maintain the communication session as the MN moves.

(a). *Rendezvous.*

ILNPv6 could leverage the current DNS to provide a suitable mechanism for rendezvous. Enhancement of DNS is required to store NID and L64 values of MNs. New DNS records have been defined for NID and L64 values, called NID and L64 records [18]. Those values are returned for a FQDN query. These new records are already implemented in widely-used DNS software: ISC BIND/*named* from v9.9.3[1], and NSD/*Unbound* from v3.2.15[2], and *Knot DNS* from v1.3.0[3].

[1]https://kb.isc.org/article/AA-00970/81/BIND-9.9.3-P1-Extended-Support-Version-Release-Notes.html

[2]http://www.nlnetlabs.nl/svn/nsd/branches/NSD_3_2/doc/ChangeLog

[3]https://gitlab.labs.nic.cz/labs/knot/raw/v2.0.0/NEWS

An MN that expects incoming connection request has two other requirements: (i) it needs to securely update the L64 value held in its L64 DNS record; and (ii) as the MN could move at any time, the normal, the L64 record needs a low cache time, so that the stale values are not cached.

The first issue is easily resolved: current Secure DNS Dynamic Update [119] is widely available in host system software today (e.g. in Microsoft Windows, and in Liunx), as well as having support in server software, and experiments have shown that updates as frequently as once per second are easily possible with existing infrastructure [90]. Also, BIND, Unbound and Knot DNS support DNS security for secure dynamic DNS updates of L64 values.

For the second issue, using a low cache time – time-to-live (TTL) – for DNS might generate high amounts of traffic from frequent updates. However, a previous empirical evaluation shows that values of TTL as low as zero for IPv4 *A* records have no significant impact on DNS load [24], so use of low TTL for the L64 records for mobile ILNP hosts would have little impact on DNS load.

(b). *Handoff.*

Handoff in ILNPv6 is supported by manipulating dynamic bindings between NID and L64 values of an MN. There are two types of handoff that ILNPv6 provides: *hard handoff* and *soft handoff*. For the hard handoff model, an MN always has only *one* L64 value and uses only one link. So, when entering a new network, it obtains a new L64 value and discards the previous L64 value. Soft handoff, in contrast, allows an MN to have an association with more than one L64 value. This means that an MN can 'belong' to, say, two networks simultaneously. Therefore, *network layer soft handoff* is possible for ILNP, something that is not possible for Mobile IP today.

Figure 2.3 shows a simple example of handoff in ILNPv6. An MN, using NID value I_M, moves from cell 1 using Locator L_1 to cell 2 and use of Locator L_2. When the MN enters the region overlapping with cell 2, the value of L_2 would be available through IPv6 Router Advertisements (RAs) – it is simply the IPv6 address prefix required for cell 2. The MN receives L_2 and now informs the CN of this new value using a *Locator Update (LU)* message, synonymous to an IPv6 BU message. The *Locator*

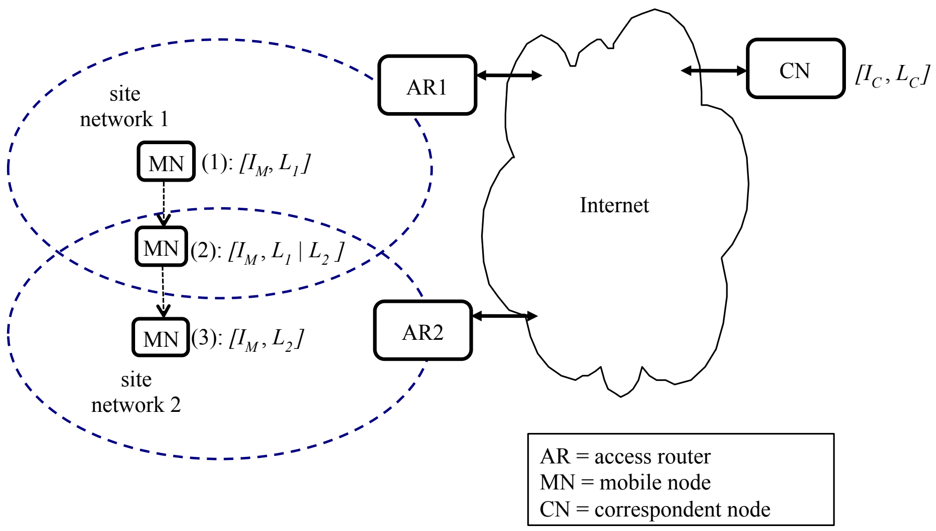
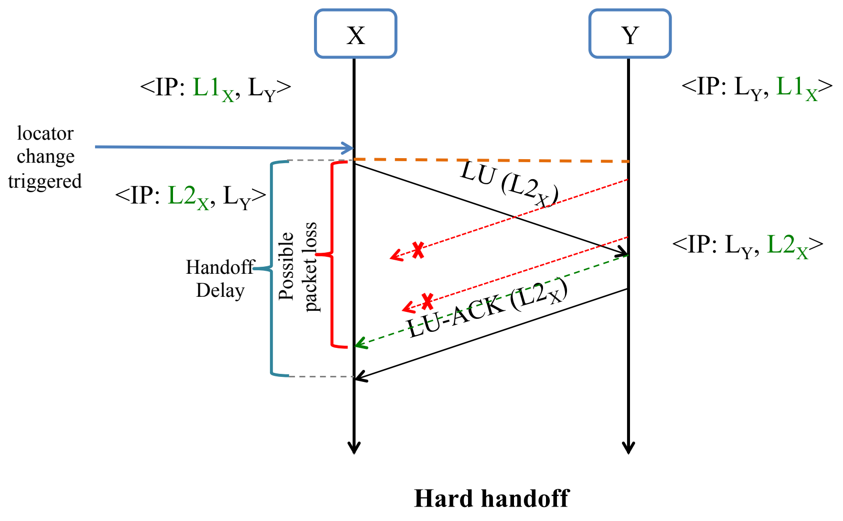


FIGURE 2.3. An example scenario of host mobility with ILNP (with soft handoff).

Update Acknowledgement (LU-ACK) is sent back from the CN once the LU is processed. If required, the MN also securely updates its relevant DNS entries (e.g. the L64 record) to allow incoming sessions to be correctly established. At this point, the MN could just drop the use of L_1 (when using hard handoff), or permit a NID value to be bound to both L64 values (when using soft handoff). As shown in Figure 2.4, for hard handoff, packet loss could occur for the in-flight packets sent from the CN using the stale L64 value. In contrast, packet loss during soft handoff is minimised. This is because the MN maintains bindings with both L64 values (L_1 and L_2) when it stays in the overlap region between the two networks, i.e. it is multihomed during handoff. This is advantageous when soft handoff is not supported by the sub-network technology across the handoff region, e.g. between different WLAN cells, or different technologies such as from a 3G to a WLAN cell.

2.6.4. Security Considerations.

A major security issue introduced by all mobility management solutions is false binding when an attacker maliciously updates the location of another MN (e.g. via fake LU to CNs). This results in disruption of communication between MN and CN because the CN is deceived that the MN has moved to a new location, hence all packets



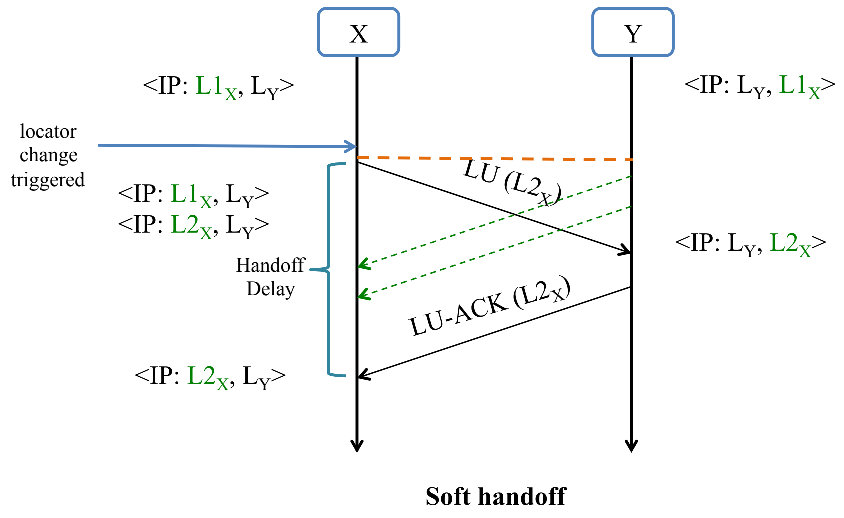


FIGURE 2.4. A Comparison of Hard Handoff and Soft Handoff Using ILNP.

will be destined to an incorrect location. This could result in a man-in-the-middle attack or a denial-of-service attack.

ILNP protects LU messages by using a *Nonce Destination Option* [15], which is exchanged between MN and CN at the beginning of the session. This can prevent forged LU packets only from 'off-path' attackers – those who are unable to monitor the path where such values are exchanged. To protect the BU/LU from 'on-path' attackers, additional cryptographic mechanisms, such as IPSec, are required. For ILNP, NID values are used as part of the IPsec Security Association, in place of IP addresses (see details below).

2.6.5. Interoperation with Other Features.

Apart from host mobility, ILNP also, theoretically, provides support for other features. However, investigating such features *is not* included in this thesis, but they should become possible if ILNP mobility is implemented.

(1) *Multihoming*

In ILNP, multihoming is treated as the same problem as mobility. Host multihoming is similar to host mobility, the site multihoming is the same as site mobility. When either an individual host or the whole site changes the connection, ILNP treats this the same as when an MN changes the location i.e. the L64 values are changed, LU messages are sent and DNS records are updated, if necessary [106].

Notice that in ILNPv6, mobility forms a duality with multihoming. In Figure 2.3, if the duration of the cell overlap permits, then MN and CN could continue to use both cells simultaneously: for example, if cell 1 was 3G and completely covered the same area as cell 2 which was a WLAN cell. This feature can be implemented by the MN to provide a multihoming resilience capability.

(2) *Concurrent Multipath Transfer*

From Table 2.2, the feature that only ILNP can provide is concurrent multipath transfer capability. This is another advantage of ILNP over other solutions. As mentioned above, ILNP allows an MN to have multiple L64 values with multiple active interfaces at the same time. This could enable multipath transfer using the existing transport protocols like UDP or TCP without the complex overhead of dealing with multiple IP addresses, as is currently the case with MP-TCP [47]. The transport layer tuple is not affected from multipath delivery from multiple interfaces: end-to-end state is preserved. Of course, some mechanisms, such as those that deal with packet re-ordering, may need to be tuned.

(3) *IPSec*

For the Internet today, IPSec would be used for securing a communication between two end nodes. Full IP addresses of both ends are used for IPSec Security Association (SA) bindings. These addresses are required to be fixed during the communication. When ILNPv6 is in use, IPSec SAs would bind to only the NID instead of a full IP address. Therefore, secured communication with IPSec is possible, even though a node changes point of attachment (e.g. handoff) because only the L64 value would change. The IPSec SA remains valid because it is bound to only the fixed NID value. Of course, the other features of IPsec – Authentication Header (AH) and Encapsulating Security Payload (ESP) – are also usable with ILNPv6.

(4) *Network Mobility*

Network mobility is where a whole site (i.e. one or more sub-networks) is mobile. This can be supported by two different mechanisms. First, the mobility is managed by every host in the site, i.e. every host acts as if it is an MN, though it may not actually be mobile. When the site network moves to a different location, every host in the site is responsible to perform a handoff, as discussed above. In this mechanism, the site border router (SBR) need not to be ILNP-capable because everything is handled by the end hosts. The second approach based on "Locator Rewriting" mechanism at SBR [16, 98], which require the SBR to be ILNP-capable. The hosts in the site network need not to be aware of site mobility. An example of network mobility with the "Locator Rewriting" mechanism is shown in Figure 2.5. When the site network moves from ISP 1 to ISP 2, the handoff from L_1 to L_2 is done by the SBR, i.e. the SBR sends LU messages to all CNs of the hosts in the site as well as securely update the DNS records. The hosts in the site still use the same L64 value, which is local. The SBR is responsible to map between the local L64 (L_L) and global L64 (L_1 or L_2). This mechanism is similar to NAT as used today for IP.

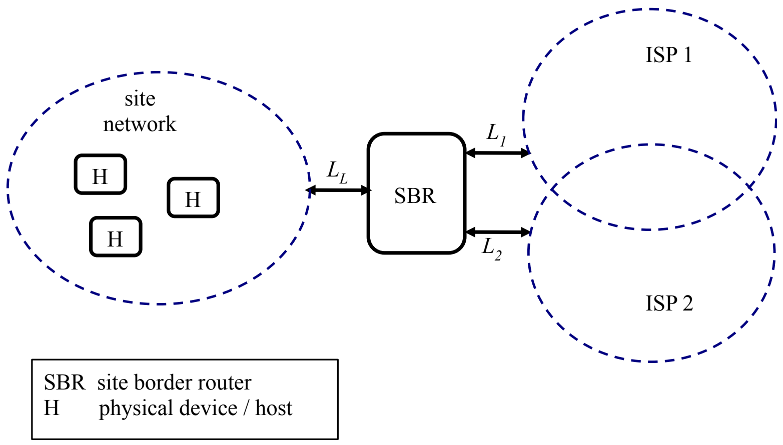


FIGURE 2.5. An example scenario of network mobility with ILNP using the "Locator Rewriting" mechanism.

(5) *Traffic Engineering*

The use of ILNP for traffic engineering options can be done by the "Locator Rewriting" mechanism [5, 16]. It allows the SBR to control outgoing traffic from different hosts to different outgoing interfaces by rewriting the local L64 value from different host to different global L64 values based on the policy. This enables, for example, a load balancing and a control of egress traffic paths.

2.6.6. An Alternative ILNPv6 Prototype.

The Internet Draft [27] discusses an alternative way to implement ILNPv6 in the Linux Kernel 3.2.12. However, the document explains only the overview of the implementation choice and lacks detail. There is no available source code and no performance evaluation for the prototype. Hence, it is not possible to do a direct comparison between that prototype and the implementation presented in this thesis. Table 2.4 compares key differences between the alternative implementation and the implementation choice in this thesis.

Firstly, for the alternative prototype, the core implementation of packet sending and packet receiving is done by using Netfilter Hook, which normally provides functionalities of the firewall. The modification of Netfilter has negative impact for deployment in the real systems and could affect other existing firewall rules. The

TABLE 2.4. Comparison of the alternative prototype and the prototype in this thesis.

Alternative Prototype [27]	This thesis prototype
1. The implementation is done by using Netfilter Hook (firewall).	1. The implementation is done in the core Linux kernel by extending the current IPv6 code.
2. A 'locator rewriting' mechanism is used to erase/refill L64 value to hide it from the transport layer.	2. Transport layer code is modified to use only NID for session binding.

implementation in this thesis is done in the Linux kernel, which provides a cleaner approach, potentially better performance, and could be simply deployed by upgrading the kernel of end systems.

Secondly, the alternative prototype uses the locator rewriting mechanisms to hide locator values from the Transport layer. This may provide an easier approach than modifying the code of transport layer protocols, but the socket calls also need to be modified to erase/refill the L64 values in order to communicate with the applications. In addition, this choice of implementation may not work in every scenario, for example if the node has more than one L64 value, it is not possible for the host to refill the correct L64 values because it does not know which L64 has been erased in different layers. Also, it is unlikely that the alternative implementation would be able to support soft handoff. In this thesis, transport layer protocols are modified to ignore the L64 part of the ILNPv6 address when doing session lookup and checksum calculation.

Nevertheless, the Internet Draft points out some practical problems of ILNPv6 encountered during the implementation. For example, after the first few packets that contain the nonce values, the receiver might not be able to distinguish packets from the sender if both ILNPv6 and IPv6 are in use with the same 128-bit values in the address fields of the packet. So, for the implementation in this thesis, the nonce values are inserted in every ILNPv6 packet. Though some may argue that this results in bigger packets, an additional nonce option is only 8 bytes long, and provides protection against off-path packet-based attacks on the end-host.

2.7. Summary

This chapter has reviewed the past and the current host mobility solutions. There are two types of host mobility solutions: host-based and network based, each has its own advantages and disadvantages. Again, this thesis takes a position that host-based solutions are a better choice.

For host mobility, there are two phases of communication: *rendezvous* and *handoff*. This work focusses on only the handoff – how a communication session is maintained when an MN changes point of attachment. The session initiation (rendezvous) is left for further detailed study, discussed in Section 7.2.1.

MIPv6 and ILNP are selected for a detailed study. MIPv6 has a known problem for handoff performance, especially gratuitous packet loss due to its *hard handoff* model. ILNP provides network layer *soft handoff* capability, which, in theory, should improve overall handoff performance.

Chapter 3

Changing the IP Layer in the Linux Kernel

This chapter demonstrates how ILNPv6 could be implemented in the Linux kernel to replace the use of addresses by identifiers and locators at the network layer, using a dual stack approach. The ILNPv6 functions do not have to be written from scratch, but can reuse the current IPv6. First, a feasibility study using ILNPv6 as an overlay network was conducted. This was to show that host mobility using ILNPv6 is possible. Then, the rest of the chapter shows how IPv6 in the Linux kernel was enhanced to enable ILNPv6 functionalities.

3.1. Feasibility Study of Host Mobility Using ILNP

The feasibility study[1] was conducted by using ILNPv6 as an overlay network running on top of classic UDP/IPv6. The study is motivated by the previous overlay prototype that was built on top of the UDP/IPv4 network [26], for example, the use of IP multicast to emulate sub-networks and emulate handoff. However, compared to the previous overlay prototype, this study focusses on the investigating of the *network layer soft handoff* capability of ILNP.

The overlay implementation was created as a proof-of-concept for mobility support in ILNPv6. The study focused on protocol dynamics rather than on specific engineering details. So, the use of an overlay was sufficient to demonstrate the operation of ILNPv6 as well as to evaluate the approach using new namespaces. The overlay

[1]The work in this section has been published in PM2HW2N 2013 and MILCOM 2014

system was written in C, using UDP/IPv6 on Linux with the standard sockets API. The emulated protocol layers are illustrated in Table 3.1. UDP/IPv6 multicast was used to emulate link layer collision domain, effectively, while the actual operation of packet forwarding, based on NID and L64 values, was handled by the ILNPv6 layer.

Table 3.1. Overlay protocol stack of the prototype.

Protocol layer	Protocol	Comment
Application	Packet transfer	Packets with a numeric ID
Transport	STP	a simple transport protocol
Network	ILNPv6	ILNPv6 Overlay
Link	UDP/IPv6	Unreliable link layer

The application protocol was a reliable packet stream. Each packet in each session had a unique numeric ID allowing the sender to determine if a packet was successfully sent (i.e. by receiving an acknowledgement for the packet). The simple transport protocol (STP) was an unreliable, connectionless protocol. It was a 'dummy' protocol that performs multiplexing of data from the ILNPv6 layer to appropriate applications. The ILNPv6 layer used a modified IPv6 header as shown in Figure 2.2 in Chapter 2. These ILNPv6 packets were carried in multicast UDP/IPv6 packets.

IPv6 multicasting was used for emulating the different logical networks on the same physical network. Each network had a unique NID value which simply mapped to an IPv6 multicast group. ILNPv6 hosts appeared on different networks (used different L64 values) by the overlay becoming a member of different multicast groups at the IPv6 layer, as shown in Figure 3.1. The experiment was run on a wired network instead of wireless network to investigate the actual behaviour of ILNPv6 without unpredictable noise and interference. However, several network scenarios e.g. high loss and/or high delay environments were emulated.

3.1.1. Experiment Configuration.

Based on the topology in Figure 3.1, the application emulated two kinds of traffic: Voice over IP (VoIP) traffic flows and streamed video flows (non real-time). These were sent from H1 to H2. The VoIP traffic was emulated Skype traffic using a packet size of 300 bytes to generate a 64 kbps flow, based on previous studies [28, 34]. The streamed video flow was generated by sending 1400 byte packets every 17 milliseconds.

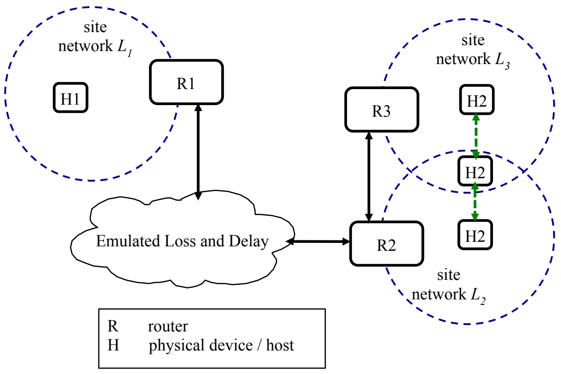


FIGURE 3.1. The topology for the overlay experiment. The hosts H1 and H2 reside in different networks, with Locator values of L_1 and L_2, respectively. The green / dashed arrows identify movements of H2 between site networks using L64 values L_2 and L_3, generating a handoff.

This is the maximum payload size that could be sent without fragmentation – after adding the 40 byte ILNPv6 header, 12 byte STP header (the overlay headers), 8 byte UDP header, and 40 byte IPv6 header, with a 1400 byte payload, it is the Ethernet MTU of 1500 bytes. This generated 658 kbps traffic which was slightly more than the average data rate of YouTube, 632 kbps [125]. Each flow was 65 seconds long – this allows the MN which handoff every 5 seconds to perform 13 handoffs in each flow.

Both VoIP traffic and video streaming traffic were run over emulations of different network conditions. First, three different delay scenarios: LAN, MAN and WAN. The LAN scenario was the usual network conditions in the test environment (a teaching lab in the school), while the other two were emulated by adding a delay of 10 ms and 100 ms respectively in each direction between H1 and H2. For each delay scenario, packet loss rates of 0%, 5% and 10% were also emulated for each direction of traffic. So, these produced round trip path loss of 0%, 10% and 20%, respectively. All network delay and loss conditions were emulated with the widely-used Linux network emulation software, *netem*[2].

The experiment was conducted 20 repetitions for each delay / loss combination listed above. In each run, H2 moved between site network L_2 and site network L_3 every 5 seconds as the flow was in progress. This was emulated by changing the IPv6 multicast group of H2. H1 was informed about the change of L64 value of H2 from

[2]`http://www.linuxfoundation.org/collaborate/workgroups/networking/netem/`

LU messages generated by H2. An LU-ACK is sent by H1 back to H2. If the sender did not receive the LU-ACK in a given timeframe, the LU would be retransmitted. The retransmission timeout for emulated LAN, MAN and WAN networks were set to 5 ms, 25 ms and 210 ms respectively, which was slightly higher than the round-trip time (RTT). Nevertheless, if an LU-ACK had not been received after 20 LU transmissions, the MN assumed the handoff had failed.

3.1.2. Results.

Both loss and delay were measured at the application layer of the sender side (H1, in Figure 3.1). The experienced loss was calculated from the number of sent packets and the number of acknowledged packets. The experienced delay was measured using half of the round-trip time (RTT/2) of each acknowledged packet, as the path was symmetric. The handoff delay was measured at the network level at the sender side from the duration between an LU message being sent and the associated LU-ACK being received. Since each flow was run for 65 seconds and a handoff occurred every 5 seconds, handoffs occurred a maximum of 13 times during a run. The analysis is based on the timings for 11 of these handoffs only, ignoring the first and the last handoffs, to avoid errors due to small variations in the start and end times of flows between the hosts.

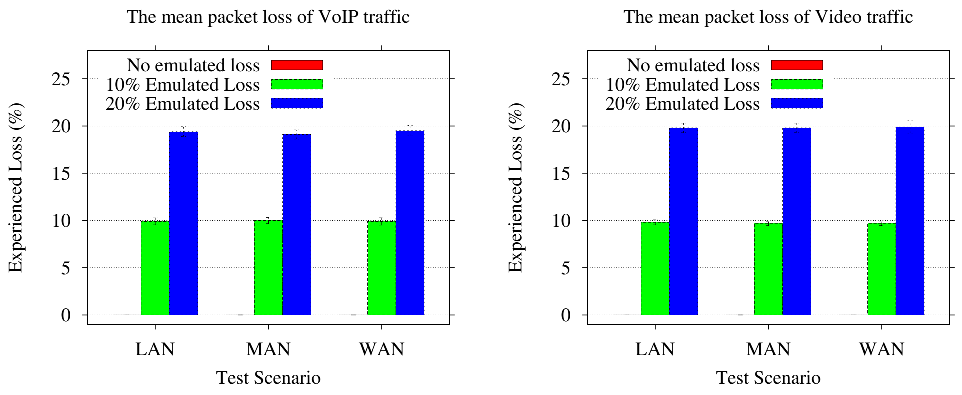


FIGURE 3.2. The mean packet loss. Error bars at 95% confidence.

The mean packet loss of each test scenario is shown in Figure 3.2. The numbers, in the same emulated loss environment, are stable across the different emulated delays, and are close to the emulated values. So, the handoff between networks does not introduce additional packet loss compared to the 'natural' value, even if some LU/LU-ACK messages are lost i.e. there is no gratuitous packet loss.

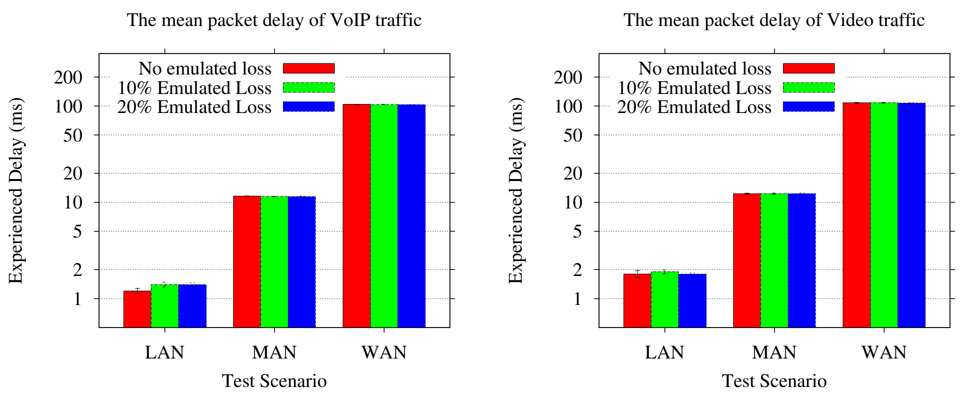


FIGURE 3.3. The mean packet delay. Error bars at 95% confidence.

The mean packet delay of each test scenario can be seen in Figure 3.3. The measured delay was unlikely to be impacted by packet loss because the measured values were close to the emulated values. The video streams produced slightly higher packet delays than the VoIP traffic due to their larger packet size.

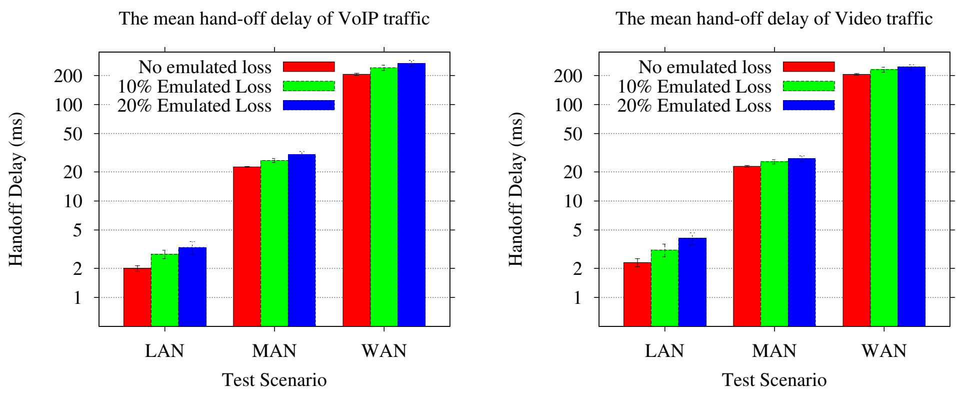


FIGURE 3.4. The mean handoff delay. Error bars at 95% confidence.

The mean handoff delay of each test scenario is presented in Figure 3.4. The handoff delay was directly proportional to the number of LUs sent, as displayed in Figure 3.5. These values were directly impacted by the emulated loss, which may have caused the LU/LU-ACK handshake to take longer. For 0% emulated loss, the number of sent LUs per handoff was very close to 1 since loss was unlikely to occur. The handoff delay was close to twice the emulated one-way delay (i.e. it was close to

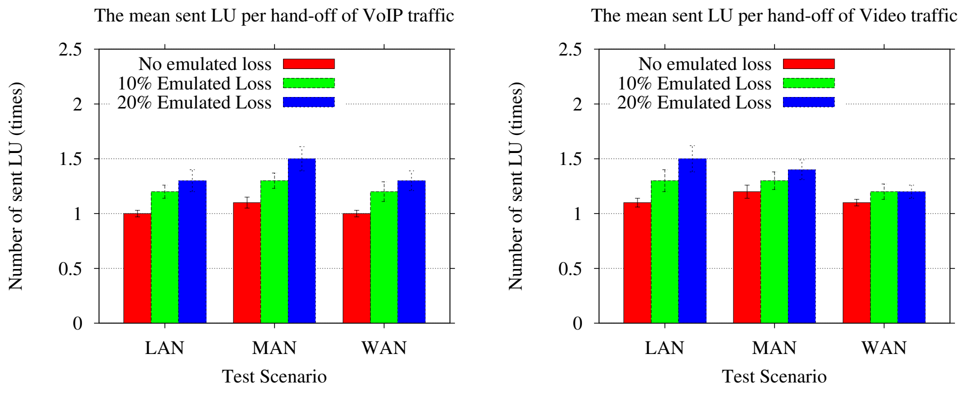


FIGURE 3.5. The mean sent LU per handoff. Error bars at 95% confidence.

the RTT), as expected. In the scenarios of 10% and 20% emulated loss, the handoff duration increased. However, the number of sent LUs per handoff did not change significantly. For 10% emulated loss, around 1-2 LUs from the total 11 handoffs were expected to be lost and require retransmission. Thus, in total, around 12-13 LUs were sent and this is around 1.1-1.2 LUs per handoff. Likewise, for 20% emulated loss, around 2-4 LUs were expected to be lost, resulting in around 13-15 LUs being sent in 11 handoffs, which is around 1.2-1.4 LUs per handoff.

In conclusion, there was no significant gratuitous packet loss when the mobile host moved between networks even when some of the LU/LU-ACK messages were lost. In the higher loss scenarios, there was only a slight observable impact on the handoff duration. The higher delay scenarios did not impact the observed loss rates. *So, the study showed that ILNPv6 could work well in a range of wireless network scenarios including those with a high loss and/or a high delay.*

3.2. ILNPv6 Detailed Design and Implementation in Linux

This section provides details of the design and implementation of ILNPv6 prototype in Ubuntu 12.04 with Linux kernel version 3.9.0. ILNP is a radical departure from the current Internet architecture as it deprecates the use of addresses. However, judicious engineering allows an evolutionary approach to enable incremental deployment. According to the previous work on implementation experiences of the Datagram Congestion Control Protocol (DCCP) in the Linux kernel [72], reusing the existing implementation of TCP as much as possible was preferable. Hence, this ILNPv6 implementation in the Linux Kernel reused existing IPv6 code where possible. This allowed a faster development compared to a 'clean slate' approach, as well as improved backward compatibility to the current systems.

3.2.1. Encoding NID and L64 values.

Figure 2.2 in Chapter 2 presents how L64 and NID values are encoded into the IPv6 address space [13]. The top 64 bits, L64, have the same syntax and semantics as an IPv6 routing prefix. The value is obtained from an IPv6 RA. The lower 64 bits, NID, has the same syntax as the IPv6 Interface Identifier, but different semantics. The NID represents a whole node, not a specific interface of the node. The NID value can be constructed in exactly the same way for ILNPv6 as it is for IPv6. For convenience, the NID value in this implementation is derived from the MAC address of the 'eth0' interface of the Linux host. This exploits the current IPv6 configuration code in the Linux kernel. Note that the MAC address is used simply as a source of bits with a high probability of being unique, to serve as a NID value.

3.2.2. Name Resolution.

For session initiation, ILNP can use DNS [12,18] just like IPv6 and MIPv6. However, this work focuses on studying the handoff performance, so, for convenience, modified versions of `/etc/hosts` and `getaddrinfo()` (in *libc*) are used for name resolution. An ILNPv6 hostname with NID and L64 values has an entry in `/etc/hosts` with the format:

```
L64|preference,NID hostname
```

For example:

```
2001:1111:0000:0000|10,021b:21ff:fe67:97a8 ilnp-linux1
```

The `getaddrinfo()` networking API is modified to interpret this new entry in the file. NID, L64 and preference values (the latter is currently unused), are passed to the kernel via the *Netlink Socket* to be stored in the *IL-V cache*, a new data structure in the kernel. To maintain backwards compatibility with the *sockets* API, the `getaddrinfo()` caller receives an IPv6 'lookalike' address which is built from NID and L64 values (2001:1111::21b:21ff:fe67:97a8, in this example). So, well-behaved legacy applications (those use the socket descriptor only, and do not use address bits for application state) work with ILNPv6 as they would with IPv6.

3.2.3. Identifying ILNPv6 Sockets.

To identify ILNPv6 socket for outgoing flows, once the `connect()` or `sendto()` is called, the IP address presented in that call is checked against the IL-V cache, which stores a list of ILNPv6 capable hosts. If the address is in the cache, the socket is marked as an ILNP socket using a new 'is_ilnp' flag in the socket data structure, (`struct sock`), and ILNPv6 is used for communication instead of IPv6. To allow differentiation between IPv6 packets and ILNPv6 packets, and also to provide off-path protection for packets, a nonce value is added to every ILNPv6 packet before sending, using a nonce destination option (ICMPv6 type 139) [15].

3.2.4. ILNP Communication Cache (ILCC).

The *ILNP Communication Cache (ILCC)* is a logical store of all active ILNPv6 communication sessions of the host. The ILCC is implemented in the kernel as a combination of a hash table and linked list. Each ILCC entry contains: i) current local and remote NID values; ii) a list of local L64 values and a list of remote L64 values; iii) session nonce values (which are bi-directional in this implementation); and iv) session timers (for clearing session state).

Important members of each ILCC element are shown in Table 3.2. Each element contains remote and local L64 values implemented as a linked list of `struct l64_info`, which has important members shown in Table 3.3. The nonce value represents a unique communication session. In this implementation, a *bidirectional* nonce value is used – both communication end-points use the same nonce value. The sender uses the nonce value presented in the ILCC or generate a new one, if the value is not set (i.e. a new session is initiated). For a new session, the receiver obtains the nonce

TABLE 3.2. Important members of an ILCC entry.

Name	Type	Description
Remote L64	`struct l64_info`	List of remote L64
Remote NID	64-bit int	Value of remote NID
Remote Nonce	32-bit int	Remote nonce of this session
Local L64	`struct l64_info`	List of local L64
Local NID	64-bit int	Value of local NID
Local Nonce	32-bit int	Local nonce of this session
Session Timer	`struct timer_list`	Timer to clear nonce values after session timeout

TABLE 3.3. Important members of `struct l64_info`.

Name	Type	Description
L64 value	64-bit int	Value of L64
flag	32-bit int	status of this L64 (active, valid, aged, or expired)
lifetime	32-bit int	Duration that this L64 stay valid
Timer	`struct timer_list`	Timer to set flag to 'expired' for stale L64

value from the Nonce Option in the first received packet. For each session, there is only one 'active' local L64 and one 'active' remote L64 at the same time. The 'active' local/remote L64 is changed when a handoff occurs (see Section 3.2.7 for more details).

3.2.5. Sending ILNPv6 Packets.

Figure 3.6 shows the function call graph of the sending flow for ILNPv6 traffic. Data packets from the transport layer arrives at the network layer on `ip6_push_pending_frames()` for UDP, and `ip6_xmit()` for TCP. Those functions are modified to invoke `ilnp_send()` if the packet is ILNPv6.

The core operation of ILNP is implemented in a new file, `net/ipv6/ilnp.c` which also deals with the dynamic bindings between NID and L64 values. The new function `ilnp_send()` is responsible for sending ILNP packets. Figure 3.7 summarises the flow of control in `ilnp_send()`. The source and destination IPv6 addresses provided by the transport layer protocol are checked against the ILCC. For the first packet in a flow, the ILCC does not have any information, so the local/remote NID and L64 values are extracted from the provided IPv6 address field and added to the ILCC. For

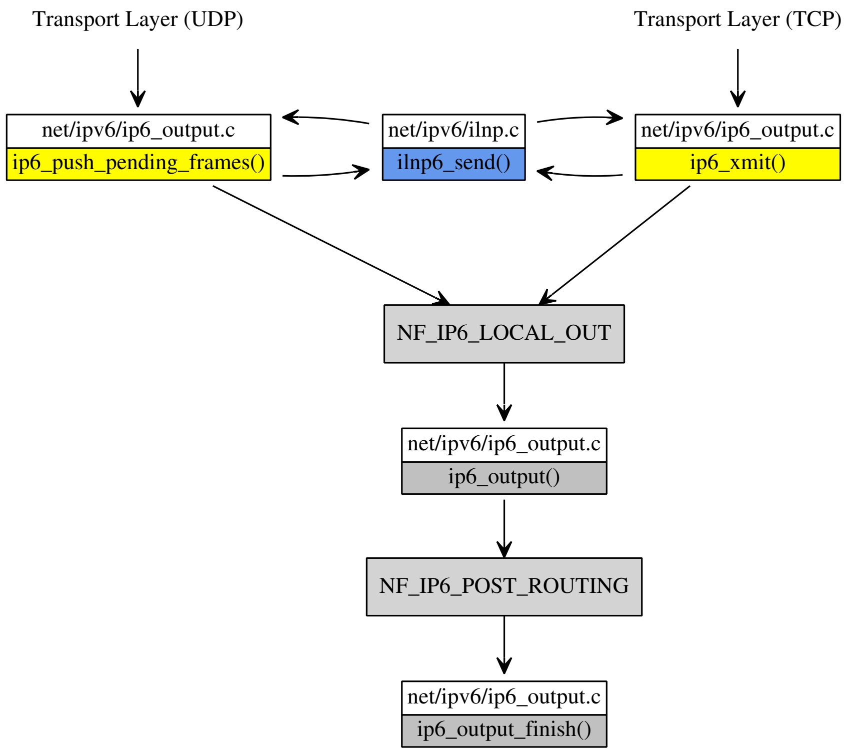


FIGURE 3.6. A function call graph in the Linux kernel for sending a packet in network layer . The grey boxes are unmodified functions. The functions in yellow boxes are modified, and a new function is shown in blue.

a new session, a new nonce value is generated. Then the nonce value for the session is inserted into the packet.

3.2.6. Receiving ILNPv6 Packets.

The function call graph for receiving an ILNPv6 packet is in Figure 3.8. The core operation of receiving ILNPv6 packets is implemented in the new function `ilnp_rcv()` in `net/ipv6/ilnp.c`. If the incoming packet is ILNPv6 (i.e. it carries a nonce value), `ip6_input_finish()` is modified to invoke `ilnp_rcv()`. A new handler for the nonce, `ilnp6_nonce()` is added. Validation of the nonce value against the ILCC is performed by the new function `ilnp6_check_nonce()`. If the nonce value is correct, the packet is passed to the transport layer.

Figure 3.9 summarises the flow of control in `ilnp_rcv()`. The source and destination NID/L64 are validated here against the information in the ILCC. The nonce

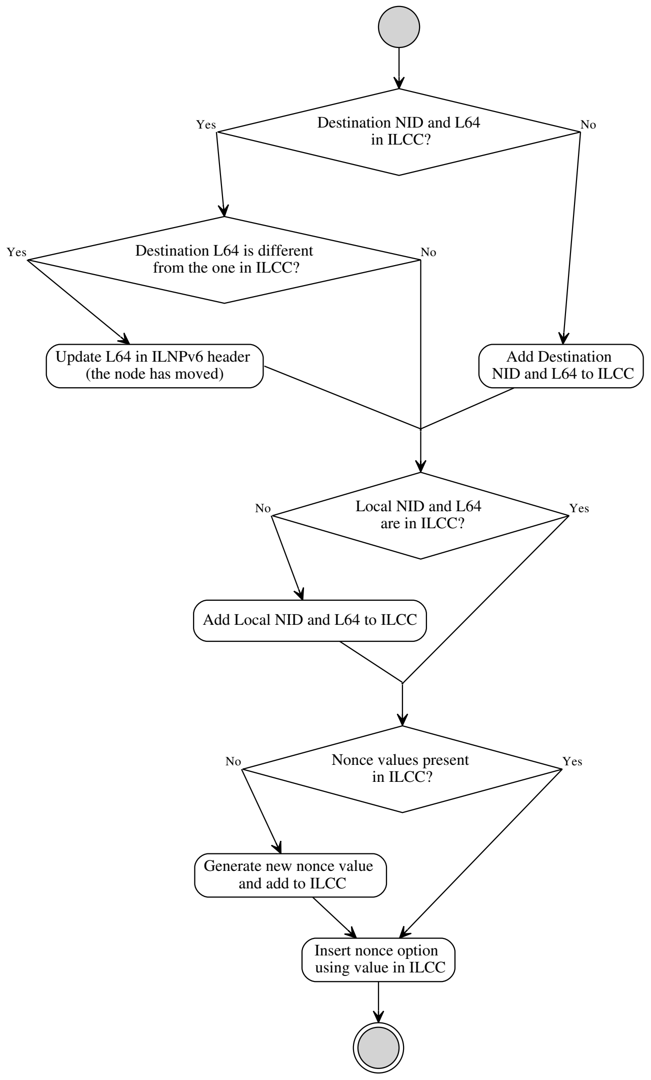

Destination NID and L64 in ILCC?
Yes
No
Destination L64 is different from the one in ILCC?
Yes
No
Update L64 in ILNPv6 header (the node has moved)
Add Destination NID and L64 to ILCC
Local NID and L64 are in ILCC?
No
Yes
Add Local NID and L64 to ILCC
Nonce values present in ILCC?
No
Yes
Generate new nonce value and add to ILCC
Insert nonce option using value in ILCC

FIGURE 3.7. A flowchart of `ilnp_send()` function.

value will be examined later on (see Figure 3.8). For the first packet in a flow, the host would create a new ILCC entry.

3.2.7. Handoff Management.

Active communication sessions must be maintained during location changes, i.e. handoff. In ILNP, handoff is managed by manipulating the dynamic bindings between NID and L64 values. A location change is detected when a new prefix is received from the IPv6 RA sent by a new access router.

Local state must be modified for the new location. Figure 3.10 shows how the RA is handled in the modified kernel. Once a new RA is received, a new ILNPv6 IL-V,

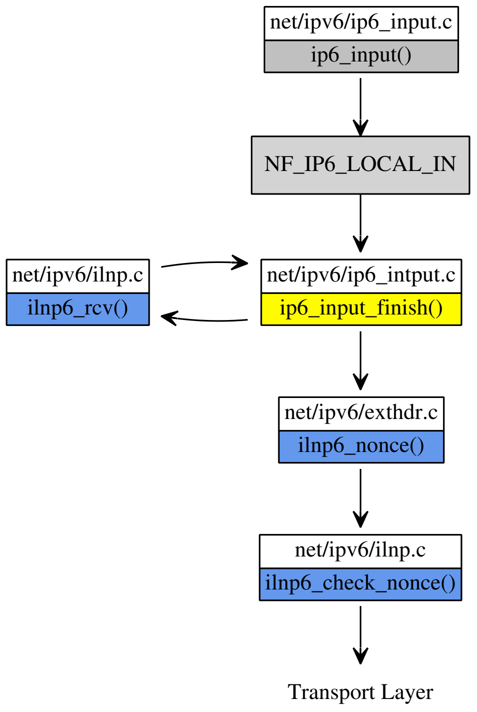


FIGURE 3.8. A function call graph in the Linux kernel for receiving a packet in network layer. The grey boxes are unmodified functions. The functions in yellow boxes are modified, and a new functions are shown in blue.

which looks syntactically like an IPv6 address, is configured. `addrconf_prefix_rcv()` is modified to use the newly received prefix as the L64 value, and the NID is derived from MAC address of the first active interface (see Section 3.2.1). `ndisc_router_discovery()` is modified to call the new function `ilnp6_rcv_ra()` every time an RA is received. `ilnpv6_rcv_ra()` then decides if a handoff should occur (e.g. a new prefix is received), and invokes the new function `ilnp6_ret()`.

A handshake is used between the MN and the CN to allow remote state to be updated with the MN's new L64 value. This is based on an LU message from the sender, accompanied by an LU-ACK from the receiver. The LU is implemented as a new ICMPv6 type (type 156, code 0) [11]. `ilnp6_ret()` is a Linux timer task, which calls the new function `send_lu()` to send LUs to all active correspondent nodes listed in the ILCC. The timer task is activated evey 1 second to retransmit the LU, if the LU-ACK is not received. At the receiver side, `icmpv6_rcv()` is extended to call a

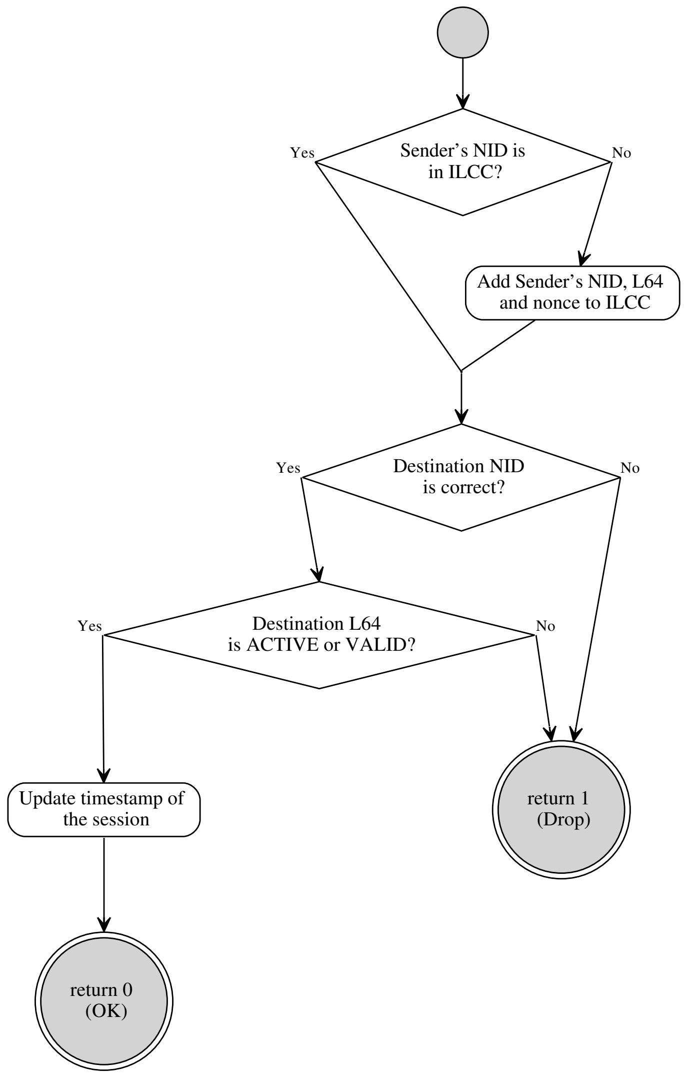

Sender's NID is in ILCC?
Yes
No
Add Sender's NID, L64 and nonce to ILCC
Destination NID is correct?
Yes
No
Destination L64 is ACTIVE or VALID?
Yes
No
Update timestamp of the session
return 1 (Drop)
return 0 (OK)

FIGURE 3.9. A flowchart of `ilnp_rcv()` function.

new function `rcv_lu()` when an LU is received (see Figure 3.11). The remote L64 value in the ILCC is then updated and an LU-ACK is sent to the sender.

TABLE 3.4. Status of L64 values in the ILCC [13].

Status	Meaning
Active	L64 is valid and currently in use.
Valid	L64 is valid (soft handoff in progress).
Aged	L64 cannot be used (hard handoff in progress).
Expired	L64 is no longer usable.

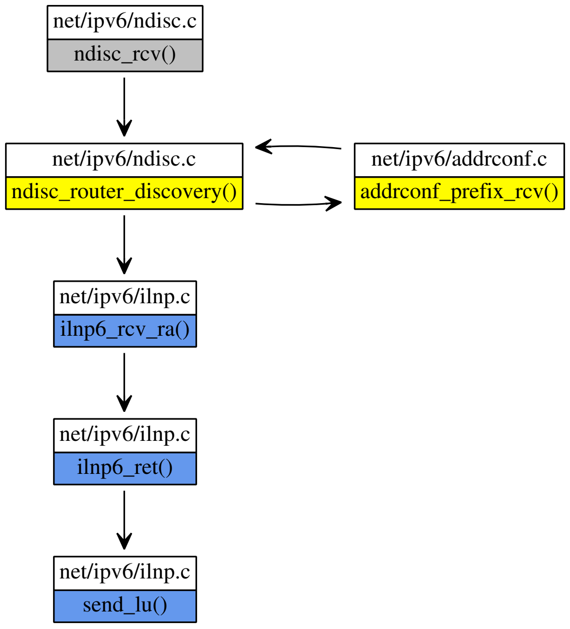


FIGURE 3.10. A function call graph of handling RA in the Linux kernel. The grey boxes are unmodified functions. The functions in yellow boxes are modified, and a new functions are shown in a blue.

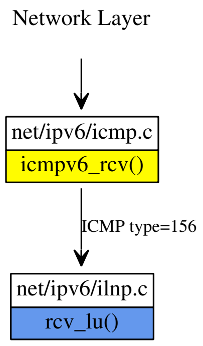


FIGURE 3.11. A function call graph of handling LU in the Linux kernel. The functions in yellow boxes are modified, and a new functions are shown in a blue.

Table 3.4 enumerates the status possibilities for NID values in the ILCC. For each communication session, there is only one 'active' local L64 and one 'active' remote L64. The 'active' local/remote L64 is changed on handoff.

ILNPv6 provides two handoff models: *hard handoff* and *soft handoff*. For hard handoff, the MN always uses only *one* L64 value at a time (a similar model to MIPv6). Soft handoff, however, allows the MN to use more than one L64 value (e.g. an 'active' one and a ‘valid’ one). This allows the MN to stay connected to two IP networks simultaneously.

A simple handoff process can be seen in Figure 3.12. Once the MN obtains a new L64 value (L_{MN2}), it updates the ‘active’ local L64 in ILCC and sets the previously active L64 (L_{MN1}) to ‘aged’ (if hard handoff is used), or to ‘valid’ (if soft handoff is used). Then, an LU is sent to the CN to notify the change of L64 value. Once an LU is received, the ‘active’ destination L64 in ILCC of the CN is updated to the new value and the LU-ACK is sent back to the MN.

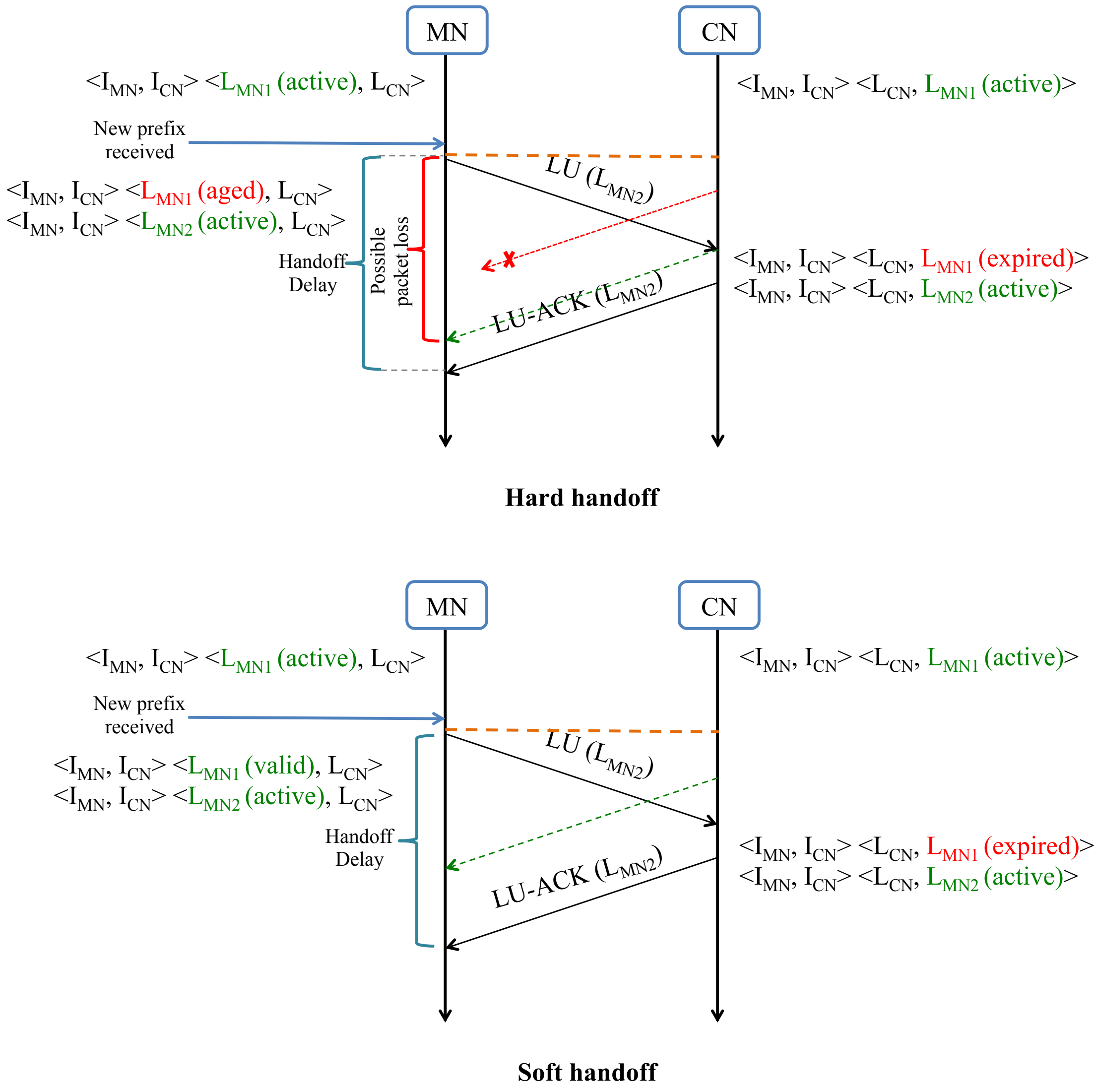

MN
CN
<I_MN, I_CN> <L_MN1 (active), L_CN>
<I_MN, I_CN> <L_CN, L_MN1 (active)>
New prefix received
LU (L_MN2)
<I_MN, I_CN> <L_MN1 (aged), L_CN>
<I_MN, I_CN> <L_MN2 (active), L_CN>
Handoff Delay
Possible packet loss
<I_MN, I_CN> <L_CN, L_MN1 (expired)>
<I_MN, I_CN> <L_CN, L_MN2 (active)>
LU-ACK (L_MN2)
Hard handoff
MN
CN
<I_MN, I_CN> <L_MN1 (active), L_CN>
<I_MN, I_CN> <L_CN, L_MN1 (active)>
New prefix received
LU (L_MN2)
<I_MN, I_CN> <L_MN1 (valid), L_CN>
<I_MN, I_CN> <L_MN2 (active), L_CN>
Handoff Delay
<I_MN, I_CN> <L_CN, L_MN1 (expired)>
<I_MN, I_CN> <L_CN, L_MN2 (active)>
LU-ACK (L_MN2)
Soft handoff

FIGURE 3.12. Implementation of changes of L64 status during hard handoff and soft handoff.

With hard handoff, packet loss occurs for the in-flight packets sent from the CN using the stale L64 value. In contrast, packet loss during soft handoff is minimised. This is because the MN maintains bindings with both L64 values (L_{MN1} and L_{MN2}) when it stays in the overlap region between the two networks, i.e. it is multihomed during handoff.

3.2.8. Extensions to Transport Layer Protocols.

For transport protocols, like UDP and TCP, the changes required are: (i) to bind end-system state only to the node identity, NID, not the whole IP address; and (ii) to set-up and maintain a dynamic binding between the NID and L64 value(s).

Consider a TCP connection at a node X with a correspondent node Y. With IP, the tuple expression (1) shows the use of the IP address (A) and port numbers (P) throughout the stack. For example, transport protocol state is bound to an interface by use of the IP address, A – the transport protocol state is tightly bound to the interface, so changes to the interface (vertical handoff) or IP address (movement across network domains) causes the state to become invalid.

$$\langle tcp : P_X, P_Y, A_X, A_Y\rangle\langle ip : A_X, A_Y\rangle\langle if : A_X\rangle \tag{1}$$

$$\langle tcp : P_X, P_Y, I_X, I_Y\rangle\langle ilnp : L_X, L_Y\rangle\langle if : (L_X)\rangle \tag{2}$$

Tuple expression (2) shows the use of NID values, I, and L64 values, L, as for ILNP. TCP must be modified to bind only to the I values, so changes to the interfaces or locator values require updates to the dynamic bindings, but does not impact the end-to-end state for TCP.

According to an example handoff in Figure 3.12, the MN using NID value I_{MN} moves from cell 1 using Locator L_{MN1} to cell 2 and use of Locator L_{MN2}. Assuming that a transport flow is in progress with the CN using the IL-V $[I_{CN}, L_{CN}]$, then the ILNP transport-layer and network layer state at MN can be represented by the expression:

$$\langle tcp : P_{MN1}, P_{CN}, I_{MN}, I_{CN}\rangle\langle ilnp : L_{MN1}, L_{CN}\rangle \tag{3}$$

based on expression (2). When the MN enters the region overlapping with cell 2, the value of L_{MN2} would be available through IPv6 RAs. The MN receives L_{MN2} and now informs the CN of this new value using an LU message. At this point, the MN could just drop the use of L_{MN1} (for hard handoff), but ILNPv6 permits a NID value to be bound to one or more L64 values allowing soft handoff to minimise packet loss.

In the overlap region, the MN expression for our transport flow would now be:

$$\langle tcp : P_{MN}, P_{CN}, I_{MN}, I_{CN}\rangle\langle ilnp : L_{MN1}|L_{MN2}, L_{CN}\rangle \tag{4}$$

It can be seen that the transport layer tuple is not effected during soft handoff: end-to-end state is preserved. This TCP tuple description is also applicable for the UDP sessions.

The extensions to UDP and TCP in the Linux kernel to enable this capability, along with the detailed evaluation of using UDP and TCP applications over ILNPv6 will be presented later, in Chapters 4 (for UDP) and 5 (for TCP).

3.3. Initial Evaluation

This initial evaluation[3] focussed on handoff behaviour of ILNPv6 in Linux at the Network layer. So, a simple custom-built ‘unconnected’ UDP application was used. ‘Unconnected’ UDP applications use only the `sendto()` and `recvfrom()` API, and there is no session binding. Hence, the only modification at the transport layer was changing the UDP checksum calculation for ILNPv6 packets to use only the NID value, both when sending and receiving UDP packets. So, changes to L64 value on handoff did not result in an invalid checksum value. The initial experiment examined if the implemented ILNPv6 worked as expected by studying the handoff performance

[3]The work in this section has been published in MobiArch 2014

comparing hard handoff and soft handoff, considering: (i) the impact on UDP application flows; and (ii) the handoff mechanism based on the LU/LU-ACK exchange.

3.3.1. Experiment Configuration.

The testbed was configured as in Figure 3.13. All connections were *wired* Ethernet 1 Gbps connections. A WiFi network and a 3G network behaviours are emulated in terms of loss and delay by using *netem*, a popular Linux network emulation tool. Table 3.5 summarises the characteristic of the WiFi network and the 3G network, and Figure 3.14 presents the cumulative frequency distribution of the one-way delay. These profiles were obtained from real *ping* measurements of the University of St Andrews WiFi network and a UK 3G network from an Apple iPhone 5, over a 40 minute period for each network.

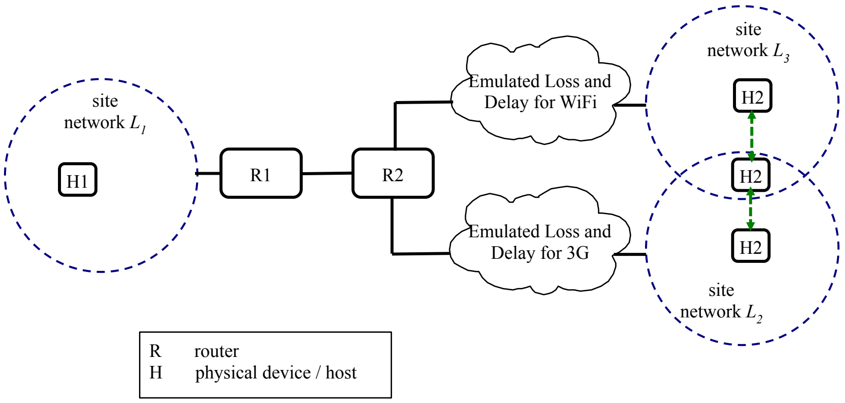


FIGURE 3.13. The topology for the initial experiment. The hosts H1 and H2 reside in different networks, with Locator values of L_1 and L_2, respectively. The green / dashed arrows identify movements of H2 between site networks generating a handoff.

TABLE 3.5. Network characteristics of WiFi and 3G.

Connection	Packet Loss [%]	One-way Delay [ms]			
		mean	stdev	95%	99%
WiFi	0.26	33	26	76	180
3G	10.10	112	50	140	276

The routers, R1 and R2, were unmodified Linux machines running *radvd*[4] to announce prefixes for the site networks L_1, L_2 and L_3. To allow the MN to pick up new

[4]`http://http://www.litech.org/radvd/`

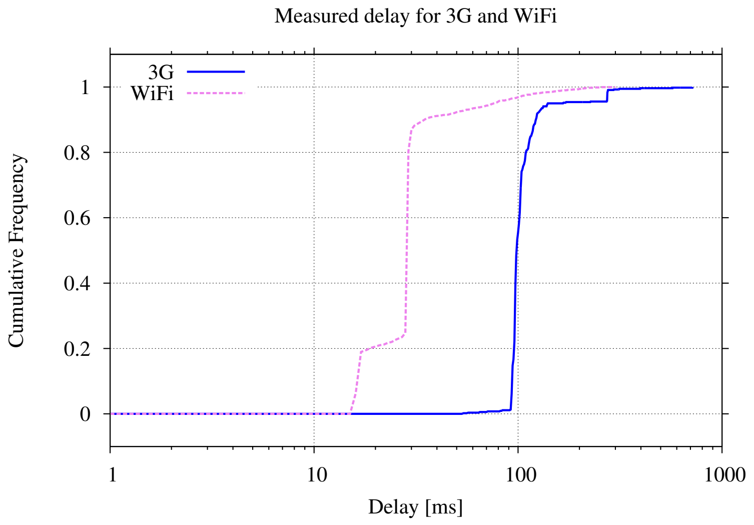


FIGURE 3.14. The cumulative frequency distribution of one-way delay of WiFi network (St Andrews) and 3G network (a UK provider).

prefixes quickly once it entered a new network, *radvd* was configured to generate RAs every 1-2 seconds.

A simple UDP/IPv6 application was used to generate two kinds of traffic between H1 and H2: Voice over IP (VoIP) flows and streamed video flows (similar to the overlay evaluation). The VoIP flows were generated based on characteristics of Skype traffic [28, 34], using a packet size of 300 bytes to generate a 64kbps flow. While 1 Kbyte packets were used to generate 250 kbps video flows, as an emulation of mobile YouTube traffic [97]. For both flows, an acknowledgement was sent for each packet, so the sender could evaluate loss.

The MN emulated movement between WiFi and 3G while each of the two flow types was in progress. Handoff was emulated by turning interfaces on and off between the emulated WiFi network and the emulated 3G network. The handoff was triggered when a new interface of H2 came up and received a new L64 value from an IPv6 RA. The handoff was completed by an LU/LU-ACK handshake as: (i) H2 updated new L64 value to H1 by sending a LU message, (ii) H1 responded with an LU-ACK to H2. If H1 did not receive the LU-ACK within 1 second, the LU would be retransmitted. The retransmissions stopped after 5 attempts. The handoffs were made every 9 seconds, with 5 seconds in the overlap area (i.e. when both interfaces of H2 were on).

Each flow was 75 seconds long (this allowed the MN to have 8 handoffs with 3 seconds spare at the end). Ten repetitions were performed for each of the scenarios above using hard handoff and soft handoff.

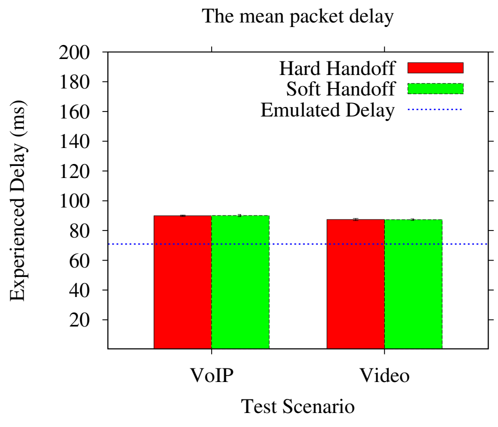
3.3.2. Results.

There are 4 performance metrics that were used in this initial evaluation.

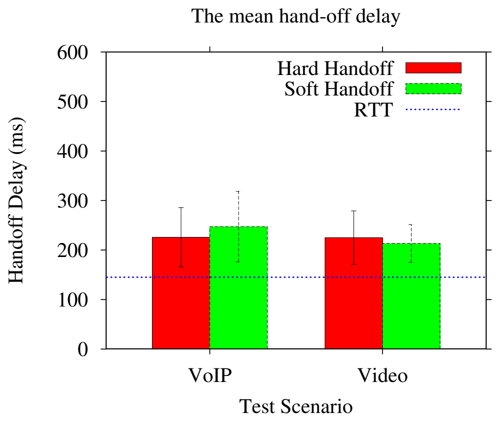
(1) *Packet delay*: the application level packet delay. This is the time taken to deliver a packet to another end host. The values were measured using half of the round-trip time (RTT/2) of each acknowledged packet, as the path was symmetric. The closer this is to the natural value, the better. (Figure 3.15a.)
(2) *Handoff delay*: the time taken for an MN to completely switch to use a new L64, i.e. the duration of the LU/LU-ACK handshake. The closer this value is to the RTT, the better. (Figure 3.15b.)
(3) *Overhead*: the number of LU / LU-ACK handshakes that are required in order to complete a handoff process. The closer the number is to 1, the better. (Figure 3.15c.)
(4) *Gratuitous packet loss*: the application level packet loss additional to the natural network loss, i.e. loss caused by the handoff mechanism. The values are the differences of overall loss (calculated from the number of sent packets and the number of acknowledged packets) and natural loss (see below). The closer this is to zero, the better. (Figure 3.15d.)

The main finding is that, soft handoff minimises gratuitous loss (Figure 3.15d), while having similar performance to hard handoff in terms of packet delay (Figure 3.15a), handoff delay (Figure 3.15c), and signalling overhead (Figure 3.15c).

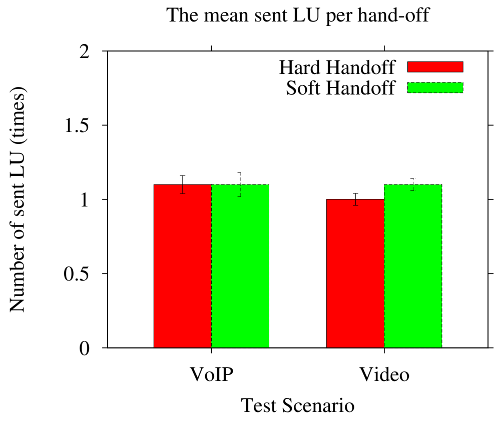
Figure 3.15d shows that gratuitous packet loss was minimised, and was almost zero. Hard handoff incurred gratuitous packet loss, as expected from the discussion in Section 3.2.7 and Figure 3.12, as some packets were transmitted with the aged L64 value while the LU/LU-ACK handshake was incomplete. As the MN started in the WiFi network and handed-off every 9 seconds, it spent 36 seconds (48%) in the 3G network and 39 seconds (52%) in the WiFi network. Hence, the 'natural' loss is



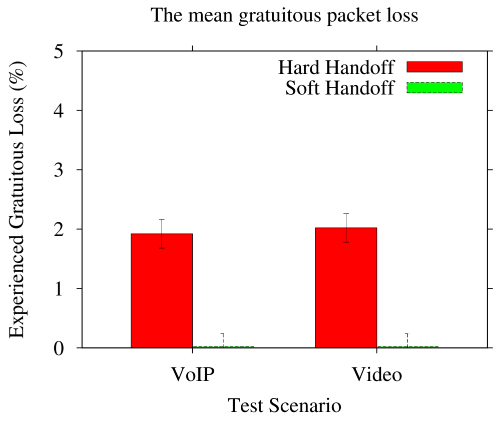
(A) Packet delay.



(B) Handoff delay.



(C) Overhead.



(D) Gratuitous packet loss (Overall loss – Natural loss).

FIGURE 3.15. Performance of hard and soft handoff. Error bars at 95% confidence from 10 runs. Horizontal/bue lines are 'natural' values.

calculated from the Eqn. (5), use $Loss_{WiFi} = 10.10$ and $Loss_{3G} = 0.26$ (from Table 3.5), giving a value of 4.98%.

$$Loss = (52Loss_{WiFi}/100) + (48Loss_{3G})/100) \tag{5}$$

The measured delay, for both hard handoff and soft handoff is slightly higher than the emulated values in every case by ~20ms (see Figure 3.15a) due to the cumulative processing delays in the end systems and *netem*. The 'natural' (emulated) delay can also be calculated from Eqn. (5): by replacing $Loss_{WiFi}$ with $Delay_{WiFi}$ (33ms) and $Loss_{3G}$ with $Delay_{3G}$ (112ms), the natural delay is 70.92ms.

The minimum value of the mean handoff delay (Figure 3.15b) would be the RTT – 66ms when handing off to WiFi (4 times) and 224ms when handing off to 3G (4 times). So, the mean RTT of every handoff during a flow was 145ms. The observed handoff delays were higher than the RTT because of some LU/LU-ACK loss, which meant handoffs may have taken more than one attempt to complete (see the larger error bars). However, the mean value is still low (less than 2 RTT), and it can be seen that the mean number of LUs sent (i.e. handoffs attempted) is low – close to 1 (Figure 3.15c).

3.4. Testbed Configuration

The testbed that was used for performance evaluation in Chapter 4 and 5 is shown in Figure 3.16. All systems (depicted as brown boxes) were Gateway GR380 F1 servers with Intel Xeon 5500 series processors, as shown in Figure 3.17. The specifications of the machines are shown in Table 3.6.

Table 3.6. Summary of the testbed hardware and software.

Model	Gateway GR380 F1
CPU	Intel Xeon E5520 @ 2.27GHz
Memory	12GB DDR3
OS	- VMware ESXi 5.1.0 (the headend) - Ubuntu 12.04 LTS with *modified* kernel version 3.9.0 (ilnp-linux1 and ilnp-linux2) or with *unmodified* kernel version 3.9.0 (ilnp-linux3 - ilnp-linux6)
Ethernet driver	Intel igb v4.1.2-k
Wireless adapter	Edimax EW-7822UAC
Wireless driver	Realtex rtl8812au v4.2.2

The host *dp32* acted as a headend connecting between the School of Computer Science, St Andrews (CS) network and local subnets of the testbed. The headend, which ran VMware ESXi 5.1.0, provided a point-to-point remote access to each testbed system using SSH via its ESXi shell. VMware was also used to create virtual machines to test the ILNPv6 kernel before installing to the real systems. The host *ilnp-linux1* - *ilnp-linux6* were Ubuntu 12.04 LTS with kernel version 3.9.0. The host *ilnp-linux1* and *ilnp-linux2* (the MN and the CN for the testbed) were installed with the modified kernel that provided ILNPv6 functionality and were configured for MIPv6 support. The other 4 systems were *unmodified* kernels.

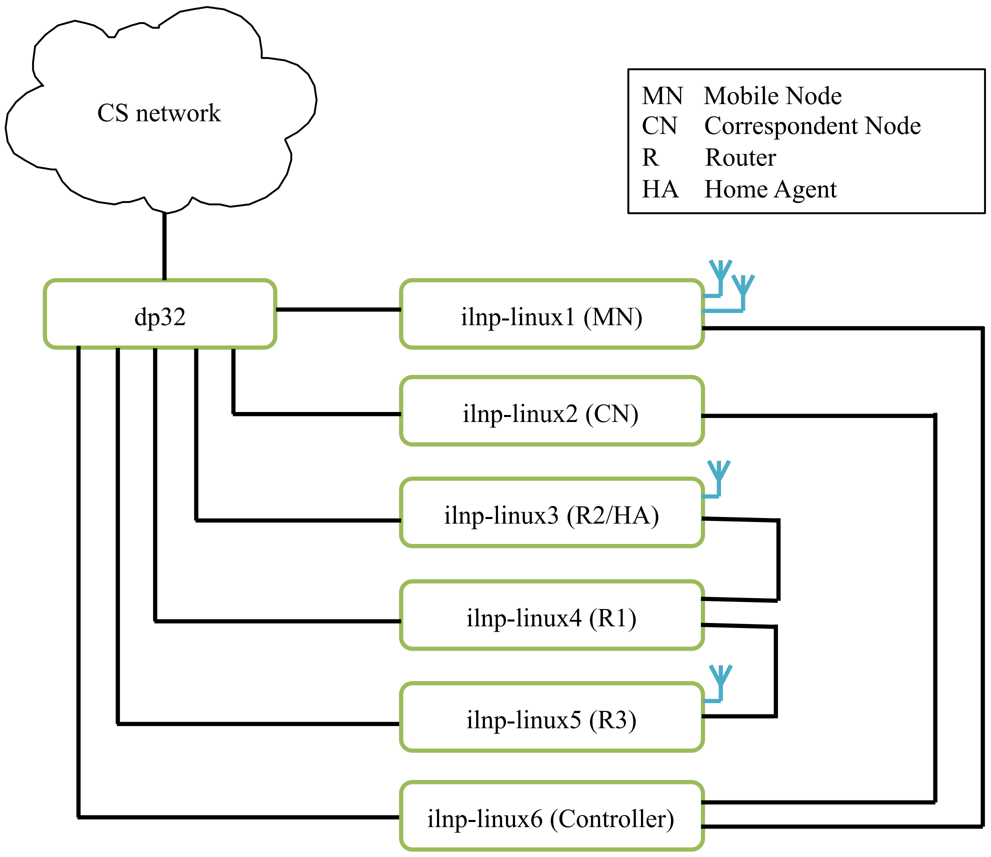


FIGURE 3.16. The diagram of the testbed for performance evaluation of UDP and TCP over ILNPv6 and MIPv6. The green boxes are machines connected by wired Ethernet 1Gbps, as shown in black lines. The blue radio antennas represent 802.11ac 2x2 WLAN links.

As in Figure 3.16, the 6 machines were connected using wired Ethernet 1 Gbps (the black lines) and 802.11ac 2x2 WLAN (the blue radio antennas), which created a topology as will be seen in Figure 4.3 (Chapter 4) and Figure 5.3 (Chapter 5). The WLAN link was the Edimax EW-7822UAC, a dual-band USB adapter as shown in Figure 3.18. The host *ilnp-linux3* and *ilnp-linux5* used the WiFi interfaces to create 2 different wireless networks i.e. they acted as wireless access points. The host *ilnp-linux1* had 2 wireless interfaces. The first interface was configured to connect to the network announced by *ilnp-linux3*, and the second one was for a connection to *ilnp-linux5*. The host *ilnp-linux6* was a controller running a shell script to remotely send commands to the MN and the CN during the evaluations.

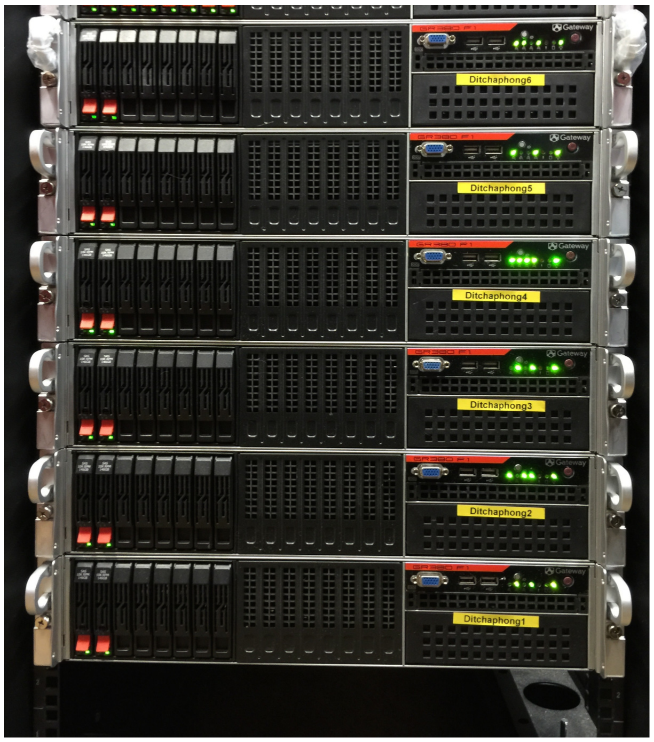


FIGURE 3.17. The machines used for the testbed showing six Ubuntu Linux servers.

FIGURE 3.18. The wireless dual-band USB adapter (Edimax EW-7822UAC) used to provide wireless conectivity for the testbed[5]

The testbed was also used in the initial evaluation in Section 3.3, but all connections were *wired* – the MN connected to R2 and R3 using wired Ethernet 1 Gbps not via the WLAN.

[5]Image from http://www.edimax.com/edimax/merchandise/merchandise_detail/data/edimax/global/wireless_adapters_ac1200_dual-band/ew-7822uac

3.4.1. Experiment Design and Limitations.

The fixed servers with WiFi interfaces were used for the evaluation in this thesis instead of using real mobile devices and real movement. The reason behind this is for the *reproducibility* of the work. Using real mobile devices may better represent the real-world scenario. However, it is not suitable for experimental purposes: it is difficult to produce the same set of control variable, e.g. the time to handoff and the duration of staying in the overlap area, for different runs of the experiment. Therefore, the use of 2 different wireless interfaces and performing a 'real switch' between those interfaces was chosen as a mechanism to 'trigger' a handoff for the experiments. The interface switching could represent the real vertical handoff in the network layer, i.e. it is similar to a mobile device switching from WiFi interface to 3G/LTE interface due to mobility.

There is also a limitation of using 2 WiFi interfaces instead of using a WiFi and a 3G/LTE interface. At the time of conducting the experiment, there was no 3G/LTE providers in the local area providing support of native IPv6 over the 3G/LTE network. Therefore, it was impossible to run ILNPv6 and MIPv6 over 3G/LTE, so using 2 WiFi interfaces was the most suitable choice.

Finally, the evaluation used *iperf* as a testing application instead of a real voice/video applications like Skype. This was because *iperf* was more suitable for testing purpose. Performance metrics such as loss and throughput could be simply measured. Also, this thesis concerns network level QoS and not application level QoE. In addition, Skype does not use `/etc/hosts` or DNS for name resolution, but has its own name resolver [21]. Therefore, it could not operate over ILNPv6 without re-engineering the resolving process to support the use of NID and L64 for ILNPv6.

3.5. Summary

This chapter has shown how ILNPv6 was implemented at the network layer in the Linux kernel by reusing the IPv6 code. The IPv6 address was replaced by NID and L64. This chapter also demonstrates how handoff in ILNPv6 is achieved by using the LU/LU-ACK exchange. Overall, the ILNPv6 prototype has a good handoff performance, and the results matched the feasibility study using an overlay: i) packet delay was close to the one-way link delay (RTT/2), ii) Handoff delay was only slightly higher than the RTT, iii) the number of LUs sent per handoff was close to 1, and iv) for soft handoff, gratuitous packet loss is nearly zero, i.e. the handoff performance at the packet level is near optimal.

Chapter 4

UDP Operation and Performance over ILNPv6

This chapter[1] presents a thorough performance evaluation of the ILNPv6 Linux kernel implementation against *umip*[2] MIPv6 using a UDP application. Based on the initial evaluation in Section 3.3, ILNPv6 is expected to perform better than MIPv6 in terms of throughput, packet loss and handoff delay. ILNPv6 with soft handoff should have the best performance and showing zero packet loss. This evaluation has three key points, it:

(1) is demonstrated by using an unmodified, existing UDP application, *iperf*, which is widely used for performance testing. This shows that the ILNPv6 codebase allows use of ILNP via the normal `socket(2)` API in Linux with `getaddrinfo(3)` name resolution;

(2) uses a set of experiments with *wireless* connectivity (IEEE 802.11 WLAN) for all experiments for a realistic evaluation in a mobile / wireless domain;

(3) has a direct performance comparison with the Linux kernel implementation of MIPv6 – both with RO enabled and disabled.

[1]The work in this chapter has been published in WiMob 2015

[2]`http://umip.org/`

4.1. Extension to the Linux Kernel for UDP Operation

Changes to the UDP code were made to allow legacy UDP applications to continue using the socket API (e.g. `connect()`, `sendto()` and `recvfrom()`) to operate over ILNPv6.

For the initial evaluation in Section 3.3, the UDP checksum calculation for ILNPv6 packets was modified to use only the NID value, both when sending and receiving UDP packets allowing 'unconnected' UDP sessions to operate. Figure 4.1 shows the function call graph of the sending flow for UDP traffic. The `udp_v6_push_pending_frames()` function was modified to use only NID for checksum calculation. The packet is then forwarded to the network layer, and continues the sending path as described in Section 3.2.5.

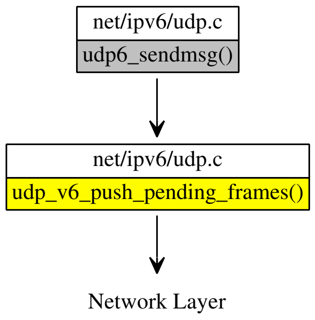


FIGURE 4.1. A function call graph in the Linux kernel for sending a packet in UDP layer. The grey boxes are unmodified functions, and the functions in yellow boxes are modified.

For the receiving flow, the function call graph for receiving a UDP/ILNPv6 packet is shown in Figure 4.2. After operations in the network layer and nonce verification, the packet is passed to `udpv6_rcv()`. `udp6_csum_init()` was modified to use only NID for the checksum calculation. However, only modifying the checksum calculation is not sufficient for 'connected' UDP sessions, i.e. those using `connect()` with `read()` and `write()` such as *iperf*. For that use case, `__udp6_lib_lookup()` was modified to use only the NID for the *udptable* lookup. The *udptable* logs all current connected UDP sessions of the host and is used by the kernel to track which port and application incoming packets should be forwarded to. Therefore, the change of L64 value (due to mobility) does not affect the table lookup because only the NID is now used – allowing packets to be delivered to the proper application.

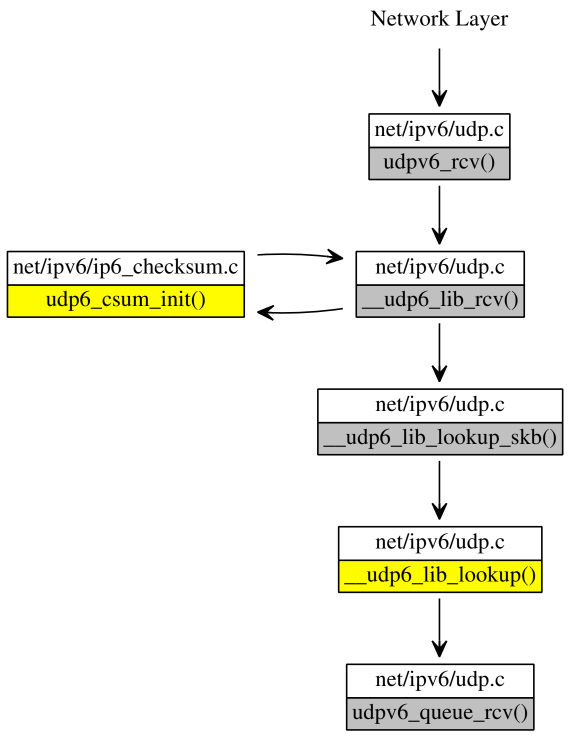


FIGURE 4.2. A function call graph in the Linux kernel for receiving a packet in UDP layer. The grey boxes are unmodified functions, and the functions in yellow boxes are modified.

4.2. Rationale for the Evaluation

In this evaluation, MIPv6 is chosen for performance comparison against ILNPv6 for several reasons. First, it is considered the standard for IP mobility. Second, it works with legacy applications without requiring any changes or extensions to the current `socket(2)` API. Third, there is an existing Linux kernel implementation, which is currently deployed (although not widely used).

For other standardised solutions listed in Section 2.4, there are some constraints preventing a straightforward and appropriate comparison of application-level performance with ILNPv6.

(1) HIP – uses public keys in order to create host identities. This means that a public key infrastructure should be in place to generate host identities, and so HIP is best suited to those applications that have stringent requirements for the use of cryptographically verifiable identities at the network layer; this is not a general requirement for all applications. Moreover, the use of the host identity in HIP requires that applications that use the current standard C sockets API have to be modified to operate over HIP [63]. Legacy applications may be used, but special treatments are needed [54].
(2) SHIM6 – is designed for multihoming support. There is still no standard mobility solution using SHIM6. Also, extensions to sockets API are recommended [62] to allow maximal usage of the protocol.
(3) LISP – is originally designed for multihoming support. The mobility extensions, both LISP-MN [99] and LISP-ROAM [49], have not been standardised yet.
(4) PMIPv6 – is a purely network based solution, hence it has a completely different model to ILNPv6.
(5) MP-TCP and SCTP – both operate in transport layer, unlike ILNPv6 which works in the network layer.

So, by using MIPv6, it is possible to use exactly the same application binary, for example *iperf* [3], allowing a direct and fair comparative evaluation to ILNPv6.

4.3. Experiment Configuration

The testbed was configured as in Figure 4.3. The following connections were *wired* Ethernet 1Gbps connections: CN to R1, R1 to R2, and R1 to R3. The MN used 802.11ac 2x2 WLAN links. *netem* was used to add extra delay of 100 ms in each direction between R1 and R2 and between R1 and R3 for emulating WAN access. Four MN handoff scenarios were created as follows:

(1) LAN to LAN (*netem* disabled)
(2) LAN to WAN (*netem* enabled between R1 and R3)

[3]`https://iperf.fr/`

(3) WAN to LAN (*netem* enabled between R1 and R2)

(4) WAN to WAN (*netem* enabled both between R1 and R2 and between R1 and R3)

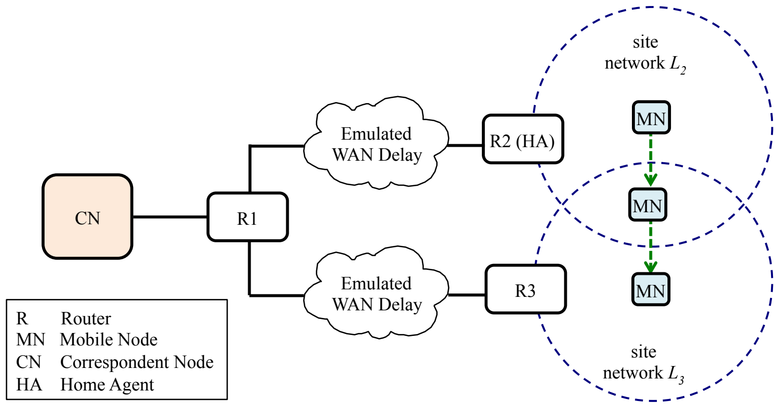


FIGURE 4.3. The topology for the experiment. The CN connects to R1 via 1Gbps ethernet. The MN initially connects to R2(HA) using WLAN – the dashed / blue circles depict the radio cell scenario being emulated. The green / dashed arrows identify movements of MN to site network L_3 generating a handoff.

The routers R1, R2 and R3 were machines using an *unmodified* Linux OS running *hostapd* [4] and *radvd* to act as access points announcing prefixes for the site networks L_2 and L_3. So, R1, R2 and R3 run a code base that is IPv6, but has no ILNPv6 capability. To allow the MN to pick up new prefixes quickly once it enters a new network, *radvd* was configured to generate RAs every 1 - 2 seconds. The MN had two WLAN interfaces – one was configured to connect to R2, the other for connection to R3.

Only the MN and CN used the modified Linux kernel supporting ILNPv6. Both machines were also configured to support MIPv6 using *umip*, to allow a direct comparison between MIPv6 and ILNPv6.

A performance testing software, *iperf*, was used to generate bi-directional UDP flows at two different mean bit rates as crude packet-level representations of the following application flows: i) 64 kbps (for Skype VoIP traffic based on [28]); and ii) 2350 kbps (for Netflix High Definition (HD) video traffic based on [2]). The packet sizes used were 300 bytes for VoIP traffic [34], and 1300 byte packets for HD video

[4]`http://wireless.kernel.org/en/users/Documentation/hostapd`

traffic. Each flow lasted 30 seconds and was repeated 10 times for each combination of the two flows, and for MIPv6 (with and without RO enabled), as well as for ILNPv6 hard handoff and ILNPv6 soft handoff. An unmodified *tcpdump* [5] was used to capture packets at the MN and CN for analysis.

In Figure 4.3, the MN started in the site network L_2, which was the Home Network for MIPv6. The bi-directional *iperf* flows were sent between the CN and the MN. As each flow was in progress, the MN started to enter the site network L_3 at t=5s into the flow, it moved out of the site network L_2 at t=20s i.e. the MN stayed in the overlap area for 15 seconds. The movement was emulated using *ifconfig* to bring the WLAN interfaces up and down. The test was repeated for 4 handoff scenarios mentioned above: LAN to LAN, LAN to WAN, WAN to LAN, and WAN to WAN.

4.4. Results

4.4.1. Handoff Performance.

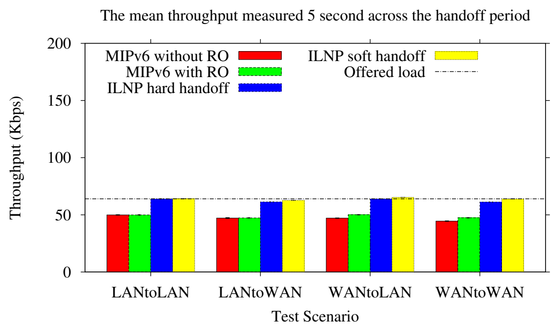
This evaluation focusses on the *handoff performance* of ILNPv6 and MIPv6, using the following metrics which show the behaviour of the MN during a *handoff period*.

(1) Throughput: The throughput during the handoff period is measured at the MN. Values close to the offered load are better.
(2) Packet Loss: The packet loss during the handoff period is measured at the MN. Lower values are better, zero is ideal.
(3) Handoff Delay: The time that the MN needs to complete the handoff process. Lower values are better, the minimum (ideal) time will be one round trip time (RTT) between MN and CN.

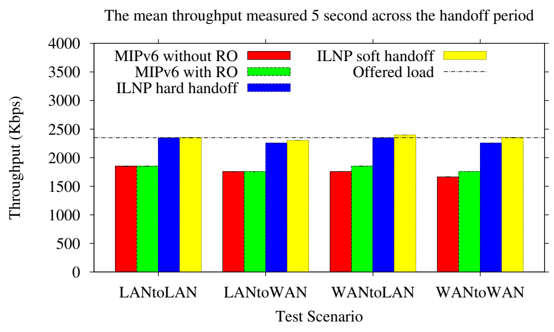
The throughput and packet loss were measured across the handoff period from time t=20s to t=25s of each flow for the MIPv6 case, and from the point t=11s to t=16s for ILNPv6 case. *These durations were selected because they covered the observed handoff period for every test case.* The handoff in MIPv6 and ILNPv6 happens at different times. In MIPv6, it is triggered when the MN moves out of the previous network completely [92, Sec. 11.5]. This is a *link layer* trigger e.g. the

[5]`http://www.tcpdump.org/`

previous link is down or is no longer reachable. For ILNPv6, it is a *network layer* trigger i.e. handoff after seeing an RA from the new network. In this experiment, however, the MN did not handoff immediately after seeing the new RA. There was a delay of 2 seconds allowing the Duplicate Address Detection (DAD) process to be completed, so the new address (new *L64*) can be used. Therefore, in this experiment the MN would handoff around the point of t=21s to t=23s, for MIPv6 and around t=12s to t=14s, for ILNP (it took around 5 seconds after the new link was up until the MN saw the first RA due to link configurations and wireless association).



(A) VoIP traffic.

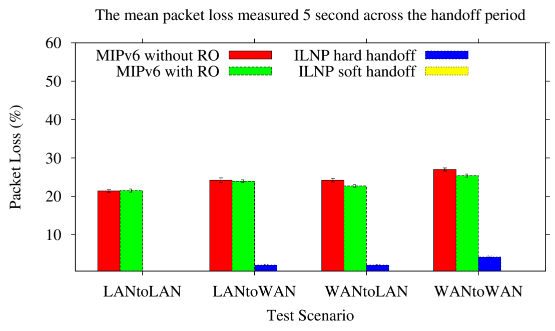


(B) HD Video traffic.

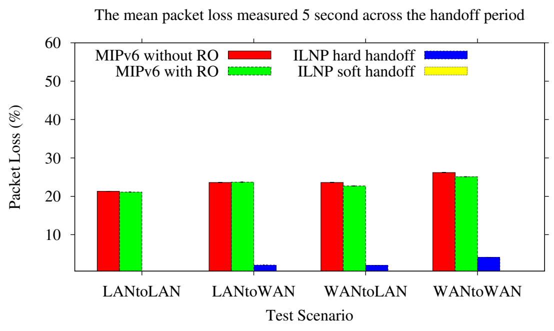
FIGURE 4.4. Throughput for a 5 second period across the handoff period for the UDP emulated traffic flows. Error bars at 95% confidence.

(a). *Throughput.*

The UDP throughput, measured for a 5 second span across the handoff period, is shown in Figure 4.4. A similar trend was found in both VoIP traffic and HD Video traffic. The results for ILNPv6, especially with soft handoff, showed close to the ideal throughput (equal to the offered load). For WAN to LAN handoff, the observed throughput slightly exceed the offered load due to bursty traffic caused by multipath delivery during the MN handoff (see Section 4.4.2 for more details). A small drop of throughput was found when hard handoff was used. For MIPv6, the throughput was substantially reduced because the MN could not receive any packets during handoff. There was small improvement of throughput when RO was enabled, when the home network was a WAN link (i.e. when handing off from a WAN): sending data directly from CN to MN was better than traversing the HA, which reside on a path with a longer delay.



(A) VoIP traffic.



(B) HD Video traffic.

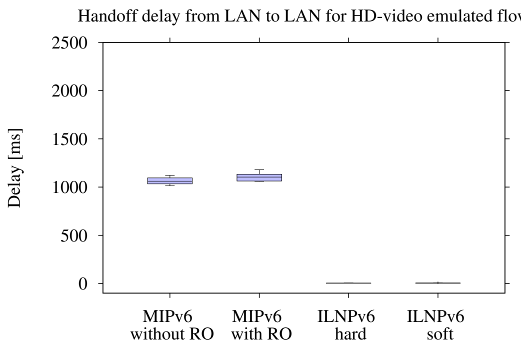
FIGURE 4.5. Packet loss for a 5 second period across the handoff period for the UDP emulated traffic flows. Error bars at 95% confidence. ILNP soft handoff has *zero* loss hence it does not visible in the bar charts.

(b). *Packet Loss.*

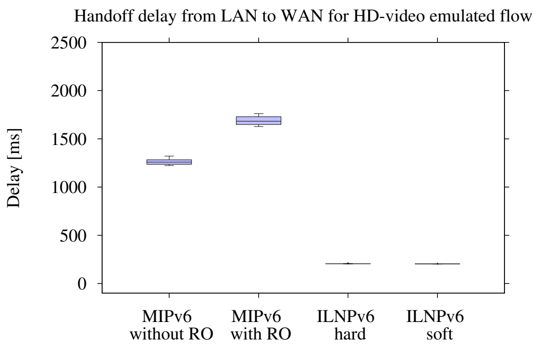
Figure 4.5 summarises packet loss during a 5 second period across the handoff period. The results were similar for both VoIP traffic and HD Video traffic. ILNPv6 with soft handoff again outperform MIPv6: *zero* packet loss was observed. A small loss was found when ILNP hard handoff was used. MIPv6 suffered from an amount of packet loss because, again, the MN could not receive any packets during handoff. When handing off from the WAN with RO enabled, the overall packet loss slightly reduced because more packets could be delivered if they did not traverse the HA.

(c). *Handoff Delay.*

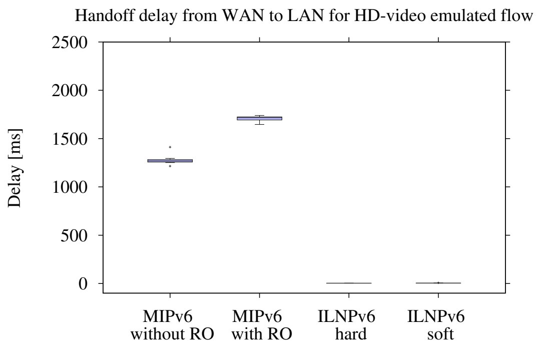
The observed handoff delay is shown in Figure 4.6 and Figure 4.7. This is only *network layer handoff delay*, which concerns the handoff signalling for MIPv6 (BU/BAck) and ILNPv6 (LU/LU-ACK). Lower layer delays such as wireless association, neighbour discovery, and DAD are not presented here. Again, ILNPv6 provided better performance than MIPv6 in terms of shorter handoff delay. The LU/LU-ACK handshake in ILNPv6 should take 1 RTT to the CN as it is purely end-to-end. Hence, the observed delay was a few milliseconds, when handing off to the LAN, and was ∼200 ms when handing off to the WAN, because ILNPv6 sends the LU and receives the LU-ACK via the new link. For MIPv6 without RO, the BU/BAck handshake took around 1 second plus 1 RTT to the HA. The extra 1 second was from the BU processing and tunnel creation at the HA before the BAck was sent back to the MN. When RO



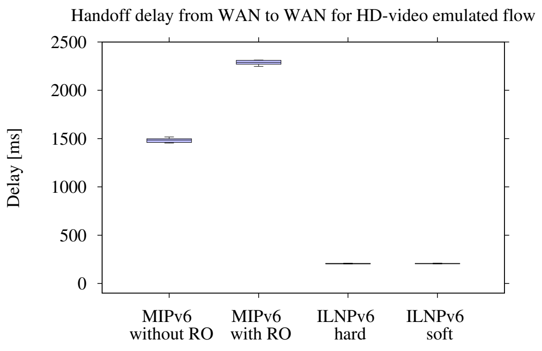
(A) LAN to LAN handoff.



(B) LAN to WAN handoff.



(C) WAN to LAN handoff.



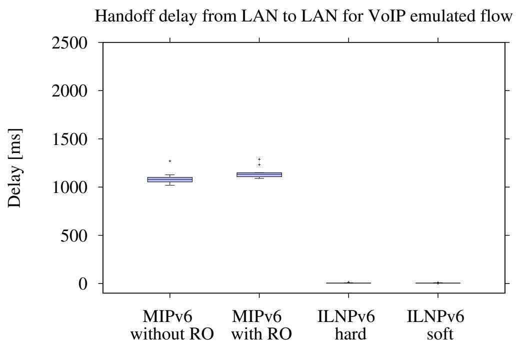
(D) WAN to WAN handoff.

FIGURE 4.6. Handoff delay for the HD Video emulated traffic flows. ILNP handoff was ∼1 RTT.

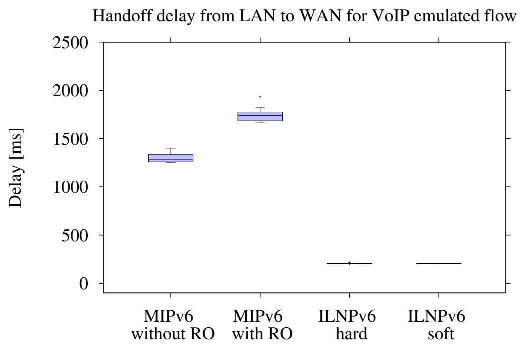
was enabled the handoff delay was higher because there were additional processes: the *return routability test* and the *binding update to the CN*. The process took longer over the WAN path. Handoff delay of MIPv6 also had higher variance than ILNPv6 because of extra scope for delay from the processing time at the HA and the RO process, compared to just the RTT for ILNPv6.

4.4.2. Analysis of Overall Flow.

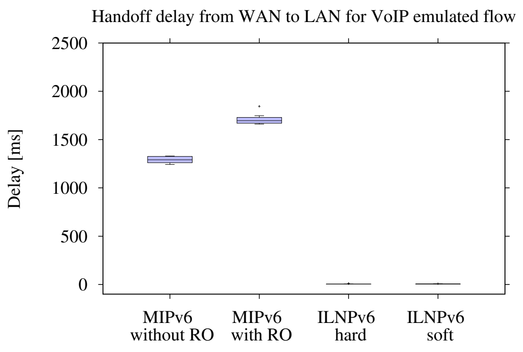
This section presents how the overall communication flow in each handoff scenario looks like. Since there was a similar trend for both VoIP traffic and HD video traffic, the analysis here shows only the HD video flow. For each handoff scenario, one of ten sets of result is selected to present here to show typical dynamics of a flow and handoff.



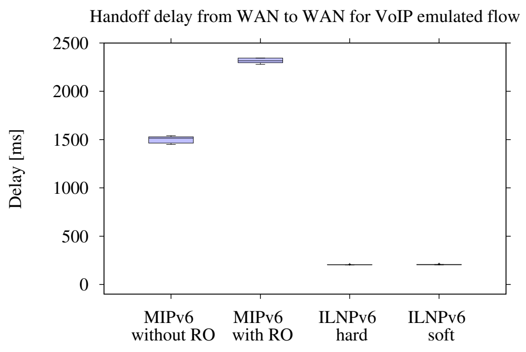
(A) LAN to LAN handoff.



(B) LAN to WAN handoff.



(C) WAN to LAN handoff.

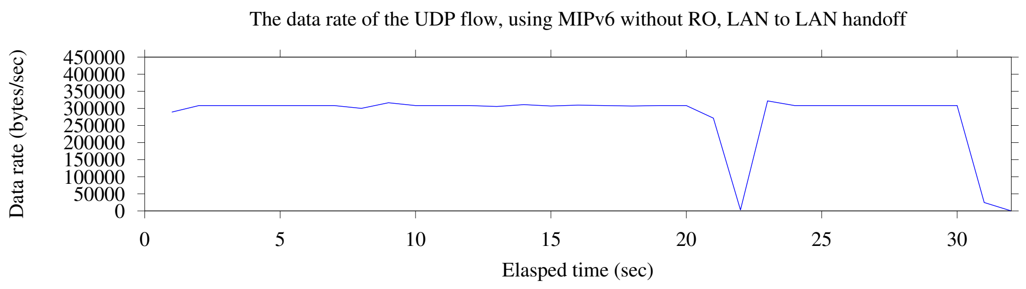


(D) WAN to WAN handoff.

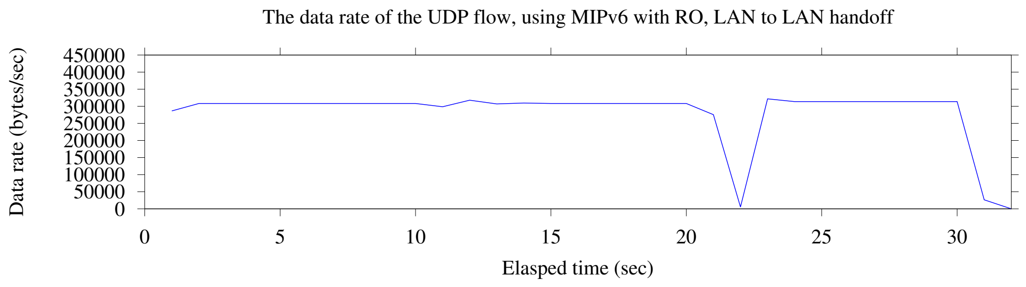
FIGURE 4.7. Handoff delay for the VoIP emulated traffic flows. ILNP handoff was ∼1 RTT.

LAN to LAN handoff

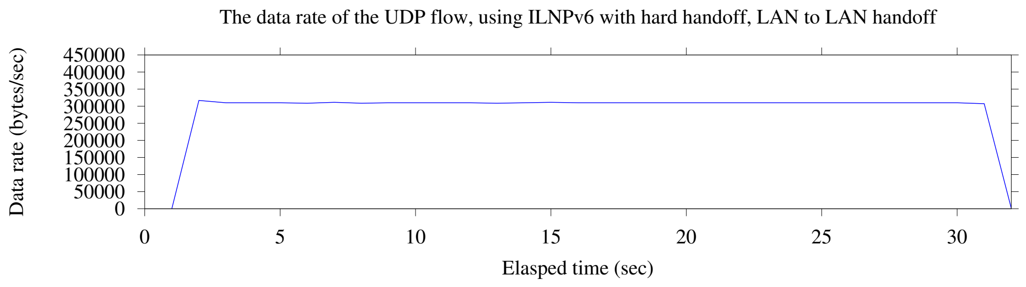
As shown in Figure 4.8 and Figure 4.9, in the LAN environment, ILNP performed better than MIPv6. For MIPv6, there was an interruption during the MN handoff – as seen where the throughput drops to zero (Figure 4.8), and where the sequence number discontinues (Figure 4.9). On the other hand, there was no interruption for ILNP. There was no difference between hard handoff and soft handoff because the CN can receive an update of the L64 value in a very short time (only a few milliseconds). So, very few packets were transmitted with the stale L64 value. Also, there was no difference between MIPv6 with and without RO enabled. The interruption time was similar – equal to the time that was used to update the HA – because when RO was enabled, the MN could still receive packets through the HA even though the RO process had not finished.



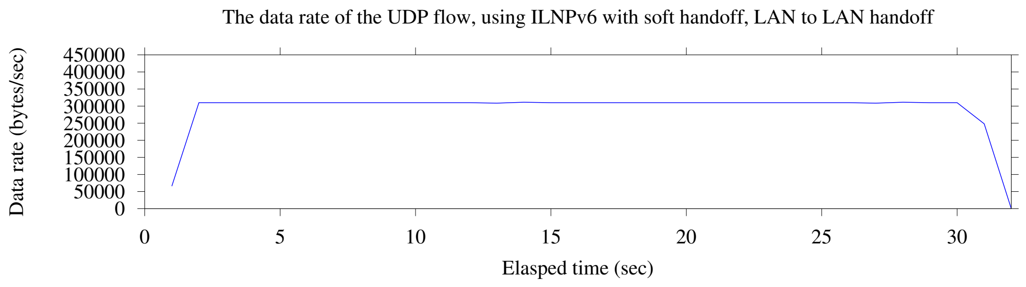
(A) MIP without RO.



(B) MIP with RO.

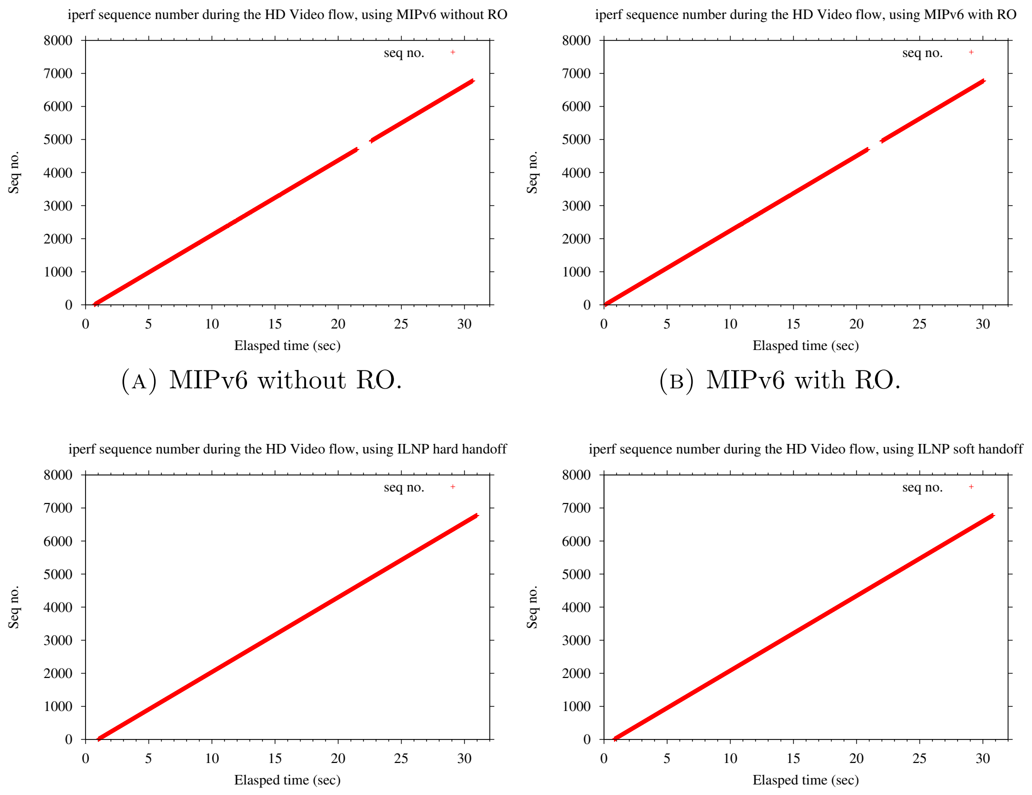


(C) ILNP hard handoff.



(D) ILNP soft handoff.

FIGURE 4.8. Communication flow for the UDP emulated HD Video traffic flows showing the data rate (bytes/sec) at the MN, LAN to LAN handoff. There was an interruption during the MN handoff for MIPv6, but no interruption for ILNPv6.



(a) MIPv6 without RO. (b) MIPv6 with RO.

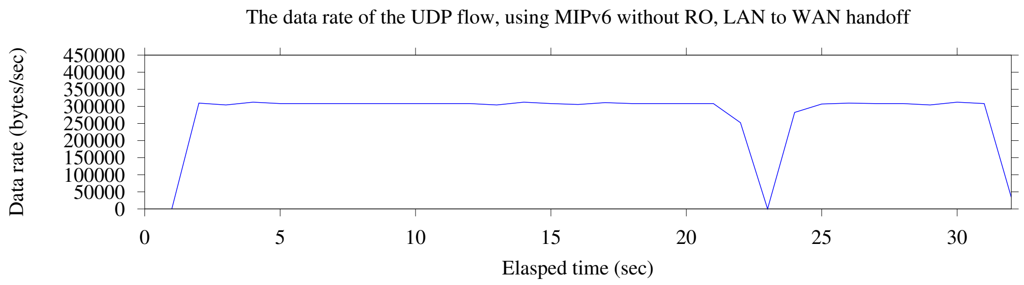
(c) ILNP hard handoff. (d) ILNP soft handoff.

Figure 4.9. Sequence number graph for the HD Video emulated traffic flows, LAN to LAN handoff. There was a gap of sequence number (implying packet loss) during the MN handoff for MIPv6, but not for ILNPv6.

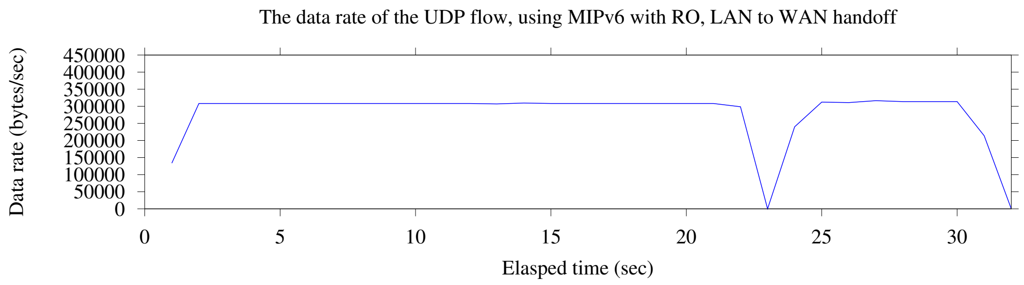
LAN to WAN handoff

ILNPv6 still performed better than MIPv6 during handoff from LAN to WAN, as seen in Figure 4.10 and Figure 4.11. MIPv6, again, suffered from a long interruption, when the MN could not receive any data. Similar to the LAN environment, there was no difference whether RO was enabled or not.

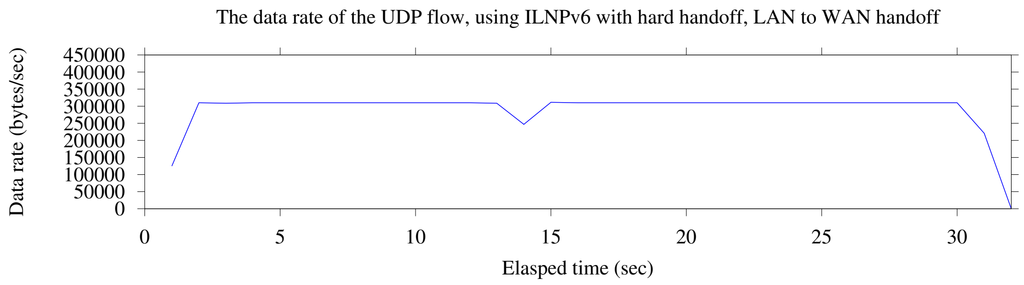
For ILNP soft handoff, there was no interruption and no packet loss, but there was a temporary degradation of the throughput, because the MN moved to a higher latency link. As shown in Figure 4.11d, the sequence number was continuous, but the flow got delayed when the MN performed handoff to a higher delay link. For ILNPv6 hard handoff, a slightly larger drop in throughput was observed, because some packets were lost when transmitted with the stale L64 value.



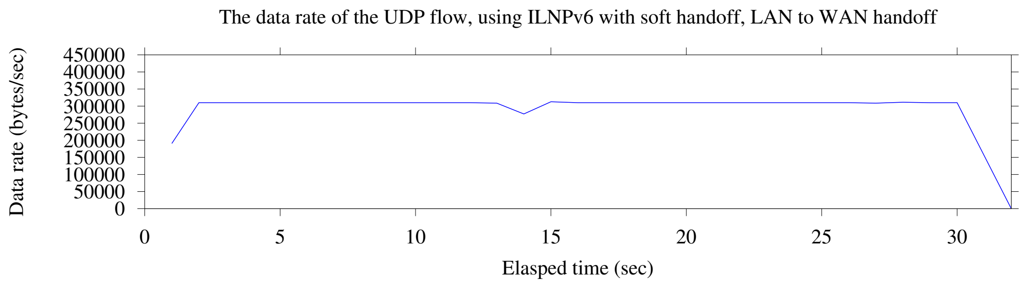
(A) MIP without RO.



(B) MIP with RO.

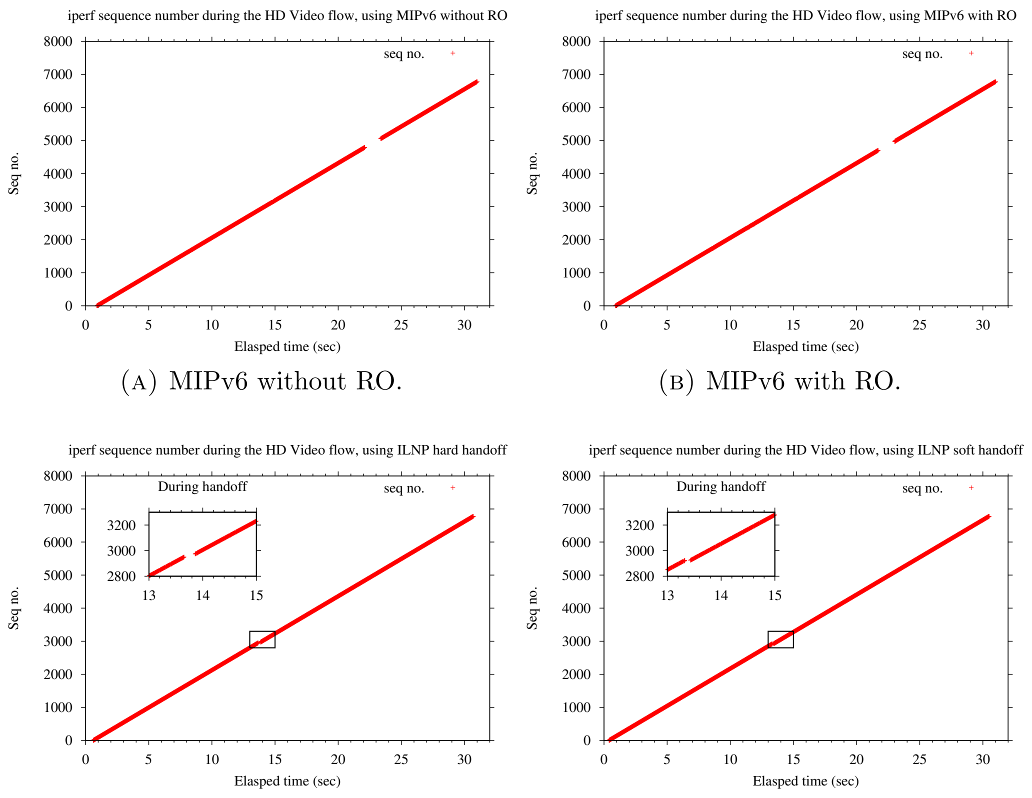


(C) ILNP hard handoff.



(D) ILNP soft handoff.

FIGURE 4.10. Communication flow for the UDP emulated HD Video traffic flows showing the data rate (bytes/sec) at the MN, LAN to WAN handoff. There was an interruption during the MN handoff for MIPv6, and a small drop of throughput for ILNPv6 hard handoff and soft handoff.



(A) MIPv6 without RO.

(B) MIPv6 with RO.

(C) ILNP hard handoff.

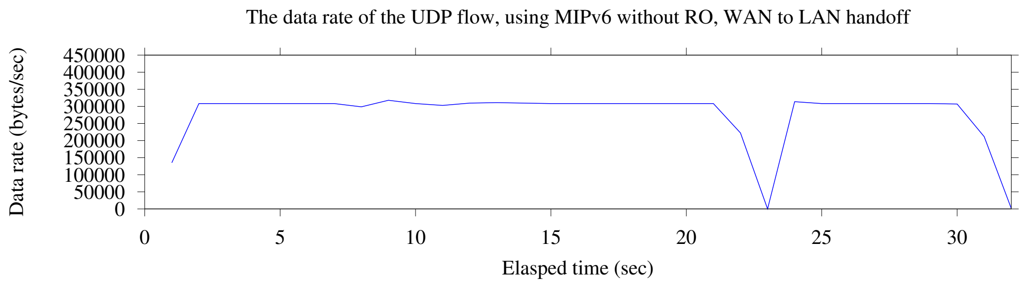
(D) ILNP soft handoff.

FIGURE 4.11. Sequence number graph for the HD Video emulated traffic flows, LAN to WAN handoff. There was a gap of sequence number (implying packet loss) during the MN handoff for MIPv6 and ILNPv6 hard handoff. For ILNPv6 soft handoff, there is a small delay before the sequence number continues.

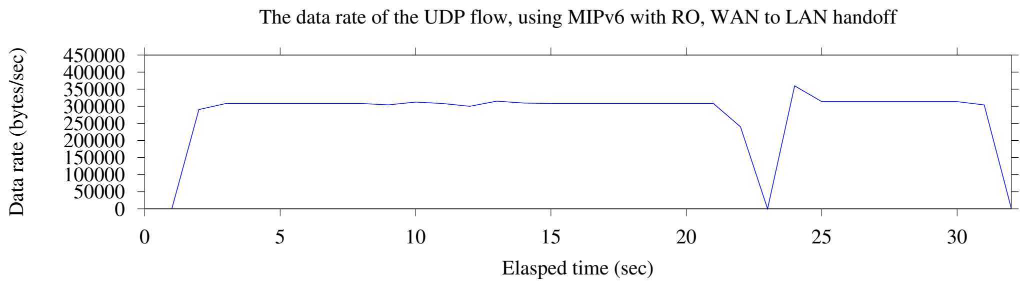
WAN to LAN handoff

Like the previous scenarios, Figure 4.12 and Figure 4.13 shows that there was an interruption of the flow when MIPv6 was used. When RO was enabled, there was a 'peak' of throughput after handoff (see Figure 4.12b), because the packets forwarded from the HA via a high delay link arrived at the MN at the same time as packets directly sent from CN via a lower delay link. This is also observed in Figure 4.13b, where a small rise of sequence number was observed after interruption. However, in MIPv6 without RO this did not occur, because the MN always communicated via the HA after handoff.

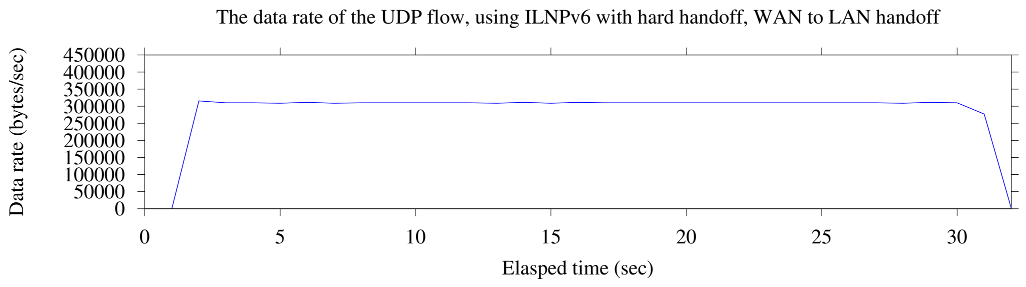
ILNP soft handoff also had a 'peak' of throughput after handoff (see Figure 4.12d) because packets from the previous, higher delay link arrived at the MN at the same



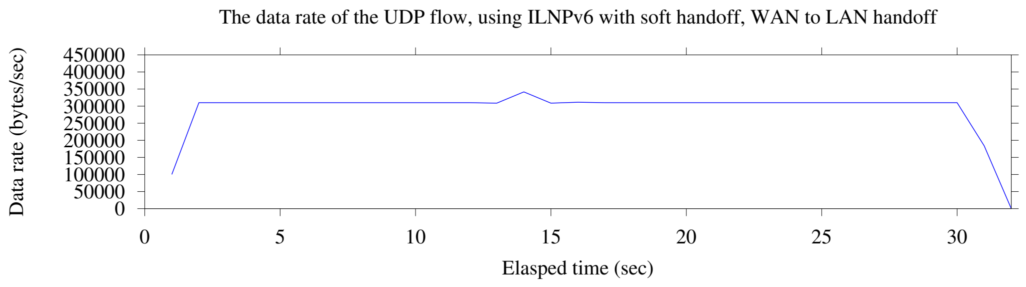
(A) MIP without RO.



(B) MIP with RO.

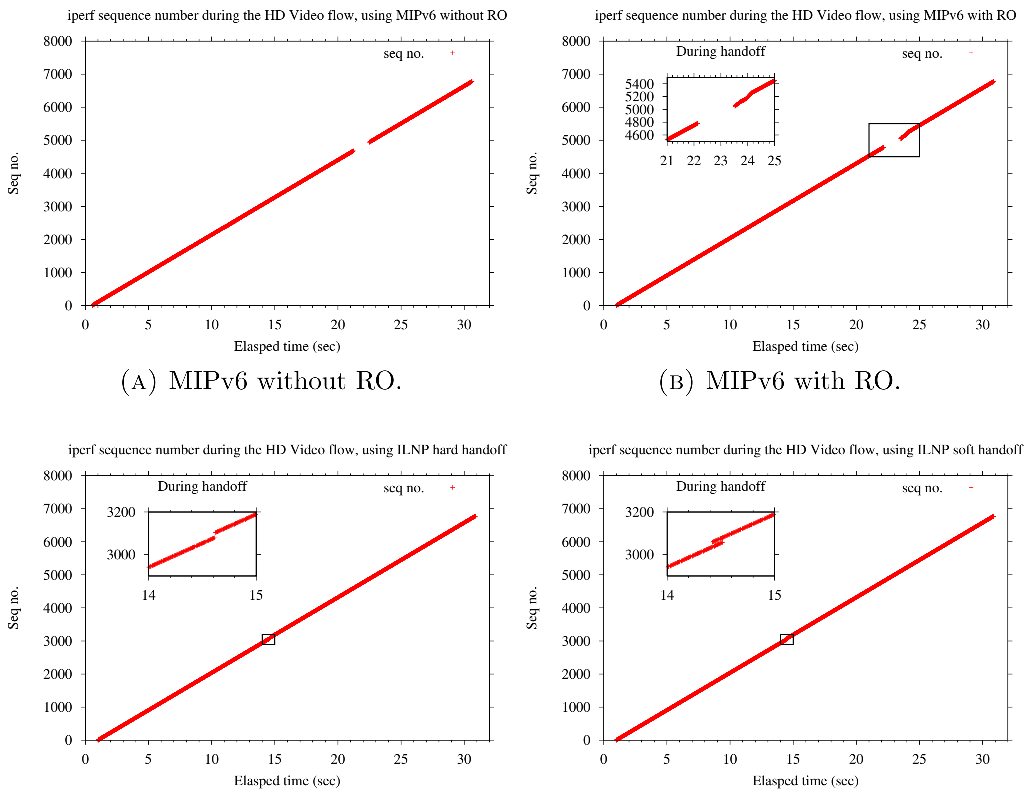


(C) ILNP hard handoff.



(D) ILNP soft handoff.

FIGURE 4.12. Communication flow for the UDP emulated HD Video traffic flows showing the data rate (bytes/sec) at the MN, WAN to LAN handoff. There was an interruption during the MN handoff for MIPv6, while no interruption for ILNPv6. There was a peak of throughput after handoff for MIPv6 with RO and ILNPv6 soft handoff.



(A) MIPv6 without RO. (B) MIPv6 with RO.

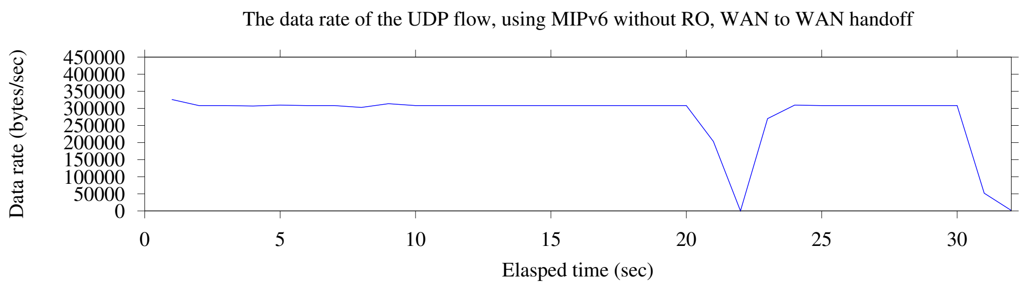
(C) ILNP hard handoff. (D) ILNP soft handoff.

FIGURE 4.13. Sequence number graph for the HD Video emulated traffic flows, WAN to LAN handoff. There was a gap of sequence number (implying packet loss) during the MN handoff for MIPv6. A small rise of sequence number after handoff is found when RO was enabled. There is a small jump in sequence number (indicating packet loss) for ILNPv6 hard handoff. For ILNPv6 soft handoff, there is an overlap of sequence number showing multipath delivery via multihoming during handoff.

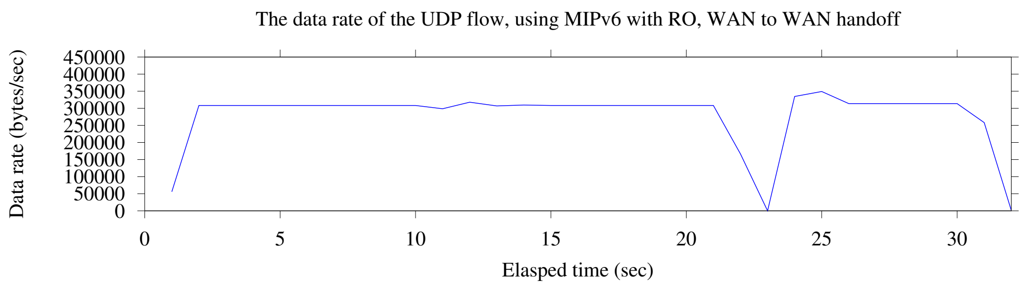
time that the MN received packets from new, lower delay link. It can also be seen in Figure 4.13d that the MN received packets from different links at the same time as the sequence numbers overlap. In ILNP hard handoff, from Figure 4.12c, it looks like that there was no interruption of the flow. However, there was a little packet loss because the MN could not receive delayed packets at the old link (because the stale L64 was used), observed as a jump of sequence number in Figure 4.13c.

WAN to WAN handoff

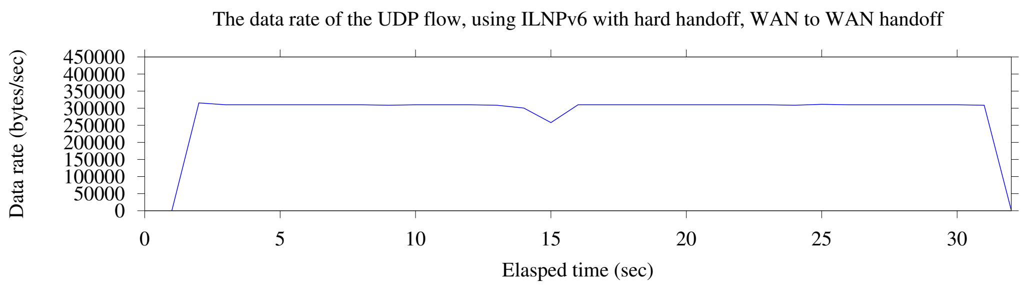
According to Figure 4.14 and Figure 4.15, for the pure WAN environment, MIPv6 (both with and without RO), again, had an interruption of the flows. For ILNPv6



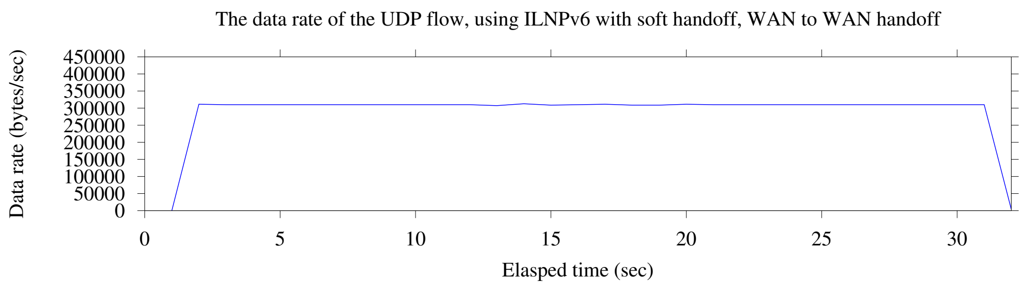
(A) MIP without RO.



(B) MIP with RO.

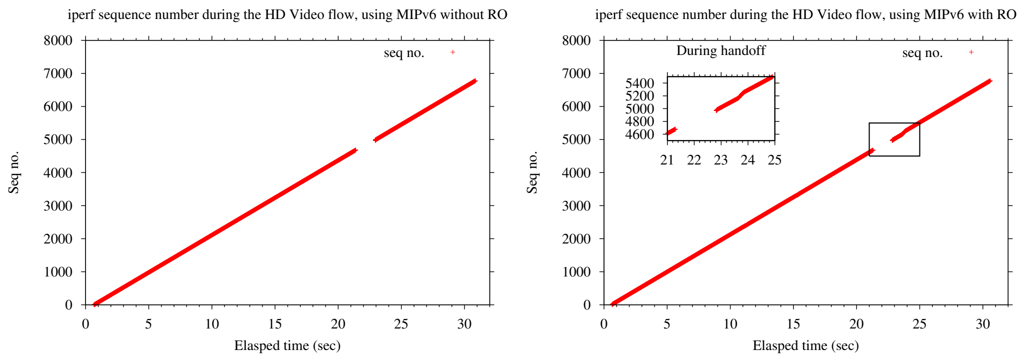


(C) ILNP hard handoff.

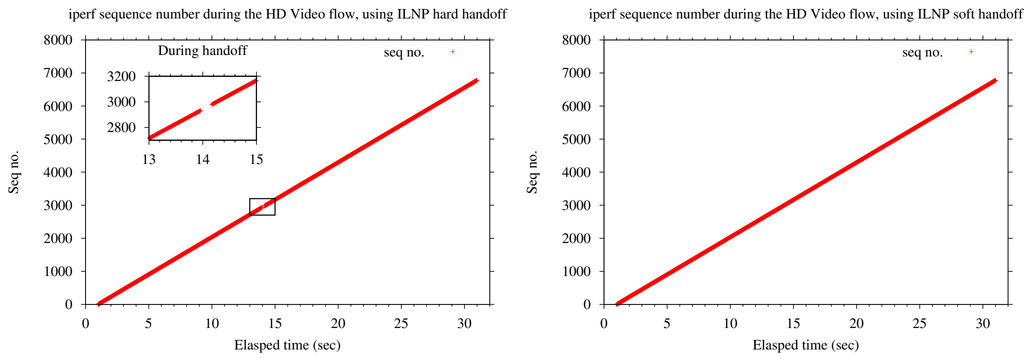


(D) ILNP soft handoff.

FIGURE 4.14. Communication flow for the UDP emulated HD Video traffic flows showing the data rate (bytes/sec) at the MN, WAN to WAN handoff. There was an interruption during the MN handoff for MIPv6, a drop of data rate for ILNPv6 hard handoff, and no interruption for ILNPv6 soft handoff.



(A) MIPv6 without RO. (B) MIPv6 with RO.



(C) ILNP hard handoff. (D) ILNP soft handoff.

FIGURE 4.15. Sequence number graph for the HD Video emulated traffic flows, WAN to WAN handoff. There was a gap of sequence number (implying packet loss) during the MN handoff for MIPv6 and ILNPv6 hard handoff. A small rise of sequence number was found when RO was enabled. For ILNPv6 soft handoff, there was no gap of sequence numbers.

hard handoff, the UDP throughput slightly dropped because of packet loss from the stale L64. Only when ILNPv6 with soft handoff was used, the data rate remained smooth during the flow. Again, for MIPv6, a 'peak' of throughput after handoff was observed when RO was enabled, see Figure 4.14b. This was, again, because packets had to traverse through the HA, and they arrived at the MN at the same time as packets directly sent from the CN. There was, however, no peak of throughput for ILNP with soft handoff because the old link and the new link had the same delay.

4.5. Summary

This chapter has shown how UDP code in the Linux kernel was modified to operate over ILNPv6. Legacy 'connected' UDP applications using the standard socket API can be used unmodified.

The testbed evaluation with 4 handoff scenarios showed that for UDP applications, ILNPv6 outperformed MIPv6 in every case, especially when soft handoff was used: gratuitous packet loss could be as low as zero, and there was no interruption of the communication flow. These results confirmed the expectation discussed earlier in this chapter. Handoff delay for ILNPv6 was also lower than for MIPv6. From this result, ILNPv6 with soft handoff could be very beneficial for media streaming and real time applications that use UDP. The communication flows should be able to run continuously during handoff, with minimal packet loss and interruption in mobile scenarios.

Chapter 5

TCP Operation and Performance over ILNPv6

The previous chapter provided a detailed evaluation of UDP over ILNPv6. For this chapter, TCP is evaluated. Similar to the UDP evaluation, MIPv6 was used for a performance comparison against ILNPv6, as explained in Section 4.2, and the testbed environment was identical. Therefore, the TCP evaluation in this chapter is expected to have similar results to the UDP results in Chapter 4. ILNPv6, especially, with soft handoff, should show better mobility support than MIPv6 in terms of session continuity, packet loss and handoff delay.

5.1. Extension to the Linux Kernel for TCP Operation

Modifications of TCP code in the kernel were made to allow legacy TCP applications, those using the standard socket API, to operate over ILNPv6.

Figure 5.1 shows the functions involved in sending a TCP packet. First, before sending data packets, TCP requires an establishment with another endpoint. The establishment process starts at `tcp_v6_connect()`. It first adds information about the sessions (e.g. source and destination IP address, and source and destination port number) to the TCP hashtable. The `inet6_ehash_fn()` was modified to use only the NID value instead of the whole IPv6 address (along with other information) for the hash calculation. This allowed received TCP/ILNPv6 packets to be deliverable to appropriate applications by using only the NID for hashtable lookup.

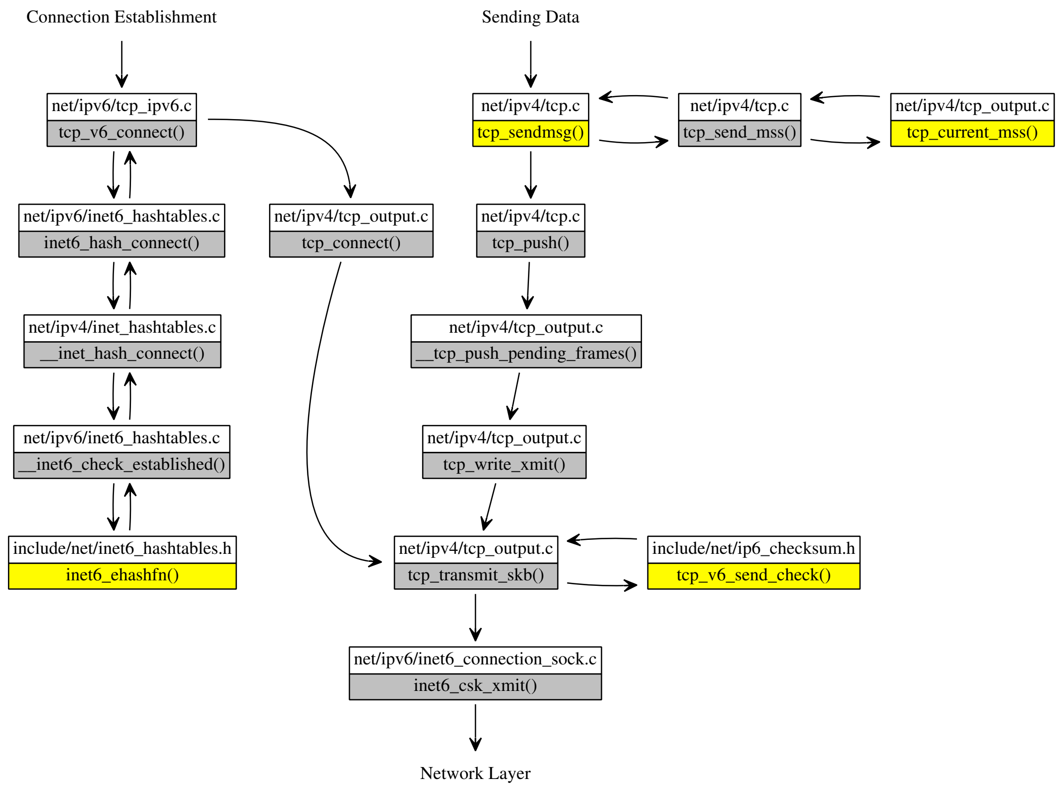


FIGURE 5.1. A function call graph in the Linux kernel for sending a packet in TCP layer. The grey boxes are unmodified functions, and the functions in yellow are modified.

Sending of data packets starts with a call to the `tcp_sendmsg()` function. The function was modified to mark the socket as an ILNPv6 socket (using the 'is_ilnp' flag, as described in Section 3.2.3) if ILNPv6 was used. The first step before sending a data packet was discovering the *Maximum Segment Size (MSS)* available for each packet. For ILNPv6 packets, the MSS must be reduced by 8 bytes, allowing the nonce destination option to be inserted into each packet. This is done by modifying `tcp_current_mss()`.

Both connection establishment packets and data packets are transmitted via a similar function: `tcp_transmit_skb()`, which calls function `tcp_v6_send_check()` for TCP checksum calculation. Again, the code here was modified to use only the NID value, instead of the whole IPv6 address, for checksum calculation. The packet would then pass to the network layer for further operation, as described in Section 3.2.5.

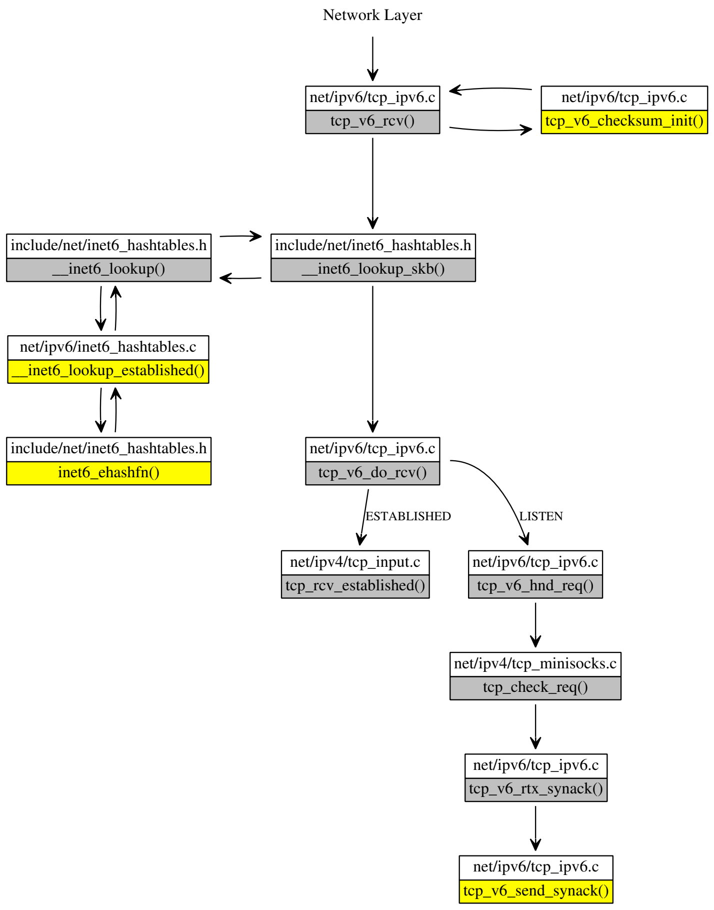


FIGURE 5.2. A function call graph in the Linux kernel for receiving a packet in TCP layer. The grey boxes are unmodified functions, and the functions in yellow are modified.

For the receiving of a flow, the processing path of a TCP/ILNPv6 packet is shown in Figure 5.2. Verified packets from the network layer are forwarded to TCP at `tcp_v6_rcv()`. The TCP checksum calculation is done by `tcp_v6_checksum_init()`, which was, again, modified to use only the NID for checksum calculation. Then, the TCP hashtable lookup is performed. The hash calculation is done by the modified `inet6_ehash_fn()` as mentioned above. After the lookup is successful, `inet_lookup_established()` determines the socket (which belong to a specific application) to be forwarded to by comparing the provided source NID, destination NID, source port and destination port to ones presented in the TCP hashtable.

Once the socket is found, `tcp_v6_do_rcv()` invokes different functions depending on the TCP state. If the connection is already established, `tcp_rcv_established()`

is responsible for processing the data. If the connection has not been established, `tcp_v6_hnd_req()` is called – usually this happens when the host receives a SYN packet. Eventually, a SYN-ACK packet is sent back by `tcp_v6_send_synack()`, which was, again, modified to use only NID for checksum calculation.

5.1.1. Limitations.

Since only the core TCP operation at the Transport layer is modified to support ILNPv6, the *Generic Segmentation Offload (GSO)* [1] including TCP segmentation offload (TSO) and TCP checksum offload, which operate at the network device level, must be disabled using `ethtool(8)`. Of course, those functions must be modified in the network device driver to provide support for ILNPv6.

5.2. Experiment Configuration

The TCP experiment was conducted in the same environment as the UDP experiment described in Section 4.3. The network topology of the experiment is shown in Figure 5.3, which is similar to the topology of the UDP test (Figure 4.3, Section 4.3). The connection between the MN and the routers used 802.11ac 2x2 WLAN links. The other nodes were connected by wired Ethernet 1Gbps. Handoff scenarios of LAN to LAN, LAN to WAN, WAN to LAN and WAN to WAN were also emulated using *netem*, as for UDP (see Section 4.3).

TCP flows were generated from CN to MN using *iperf*. All TCP settings used the default value except that the TCP window size was limited to 320 Kbytes. Without this constraint, the CN could generate too many TCP packets causing those packets to be queued at the MN, and prevented RA packets from being delivered in time. Eventually, a handoff, for both ILNPv6 and MIPv6, could be improperly triggered because the MN had not seen RA during a certain period of time. For ILNPv6, the problem may be fixed by increasing the lifetime of an L64 value, so that the delayed RA does not trigger a handoff. However, in MIPv6 the handoff model is more complicated, and tuning MIPv6 to be able to operate under this specific circumstance

[1]`http://www.linuxfoundation.org/collaborate/workgroups/networking/gso`

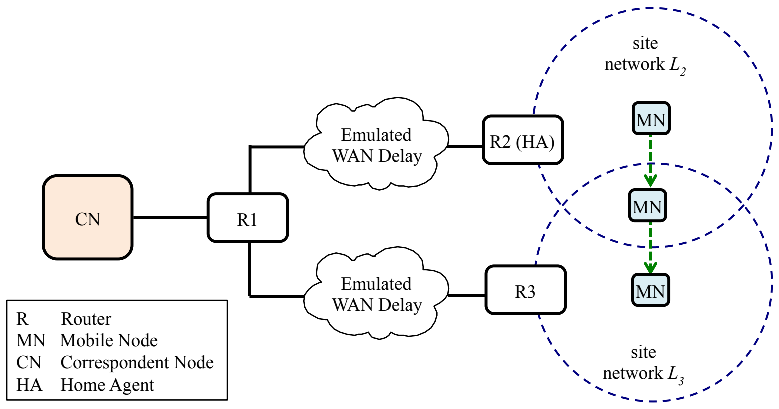


FIGURE 5.3. The topology for the experiment. The CN connects to R1 via 1Gbps ethernet. The MN initially connects to R2(HA) using WLAN – the dashed / blue circles depict the radio cell scenario being emulated. The green / dashed arrows identify movements of MN to site network L_3 generating a handoff.

was out of scope of this work. Therefore, limiting the TCP window size to constrain the amount of TCP traffic was the most appropriate approach to allow a direct performance comparison of MIPv6 and ILNPv6 under the same conditions, with default OS end-system configuration.

Like the UDP tests, each TCP flow lasted 30 seconds and was repeated 10 times for MIPv6 (with and without RO enabled), as well as for ILNPv6 hard handoff and soft handoff, and *tcpdump* was used to capture packets at the MN and the CN for analyses.

In Figure 5.3, the MN started in the site network L_2, which was the Home Network for MIPv6. The *iperf* TCP flows were sent from the CN to the MN. The movement of the MN was similar to the UDP experiment: the MN started to enter the site network L_3 at t=5s, and moved out of the site network L_2 at t=20s. Again, the movement was emulated using *ifconfig* to bring the WLAN interfaces up and down. The test was repeated for the same 4 handoff scenarios mentioned above: LAN to LAN, LAN to WAN, WAN to LAN, and WAN to WAN.

5.3. Results

This section presents TCP performance over ILNPv6 and MIPv6. There are four performance metrics, listed below.

(1) TCP flow behaviour: TCP flow data rate graphs of different handoff scenarios are presented. This is to visualise how handoff by ILNPv6 and MIPv6 impacts TCP performance. A handoff without an interruption in the flow is ideal.

(2) Successfully transferred data: volume of bytes of data that can be sent during the 30 second flow. The more data that can be sent, the better TCP performance.

(3) Retransmission: Number of TCP retransmissions attempted by the CN in each flow. The retransmissions are usually caused by packet loss, but could also be caused by packet errors and misordering. The lower the number of retransmissions, the better of the TCP performance.

(4) Packet loss: Number of lost packets in the flows. Lower values are better, zero is ideal.

(5) Handoff Delay: the time that the MN needs to complete the handoff process. Lower values are better, the minimum (ideal) time will be one round trip time (RTT) between MN and CN.

5.3.1. TCP Flow Data Rate Behaviour.

This section presents how the overall TCP flow data rate behaves in each handoff scenario. For each handoff scenario, one of ten repetitions is selected to present here, as an example of the flow dynamics. The TCP throughput is limited by the TCP window size, and is calculated by [36]:

$$Max\ TCP\ throughput = \frac{TCP\ window\ size \cdot 8}{RTT} \tag{1}$$

So, with a 320 Kbyte window, the maximum TCP throughput in the LAN environment, which RTT is ∼5 ms, is $(320 \cdot 8)/5 = 512$ Mbps. However, this high rate cannot be achieve due to the limited link speed of the WLAN links. The observed throughput in the LAN is around 30 Mbps (3.75 Mbytes/s). For WAN environment, with RTT ∼200 ms, the throughtput has a ceiling of $(320 \cdot 8)/200 = 12.8$ Mbps or 1.6 Mbytes/s.

The key finding is ILNPv6 always performed better than MIPv6 because the TCP flow was not interrupted (if soft handoff was used) when a handoff occurred. The detailed analysis of the flows in each handoff scenario can be found below.

LAN to LAN handoff

As shown in Figure 5.4, in the LAN environment, there was an interruption during the MN handoff – as seen where the throughput drop to zero – for MIPv6 case. Like the UDP results, there was no difference between MIPv6 with and without RO enabled: the interruption time was similar – equal to the time that was used to update the HA. On the other hand, there was no interruption for ILNPv6, but there was a small drop in throughput during handoff. Hard handoff had a slightly greater drop than soft handoff.

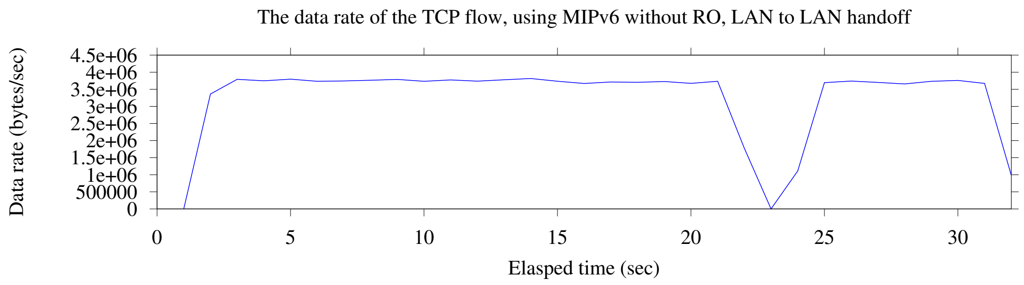
LAN to WAN handoff

According to Figure 5.5, during handing off from the LAN network to the WAN network, the TCP throughput decreased in every case. TCP sent less data because the round-trip-time (RTT) was much higher and the TCP window size was limited (as explained in the Eqn. 1, Section 5.3.1). Nevertheless, ILNPv6 still performed better than MIPv6. MIPv6, again, suffered from a long interruption, when the MN could not receive any data, and it was worse than the pure LAN environment. There was no significant difference whether RO was enabled or not.

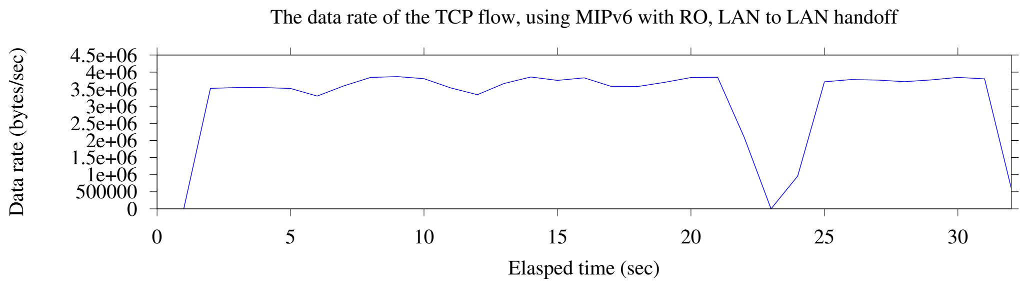
ILNPv6 hard handoff had a small interruption in the flow, but a much shorter period than MIPv6. For ILNP soft handoff, there was no interruption. The throughput just gradually dropped due to a handoff to a higher delay link.

WAN to LAN handoff

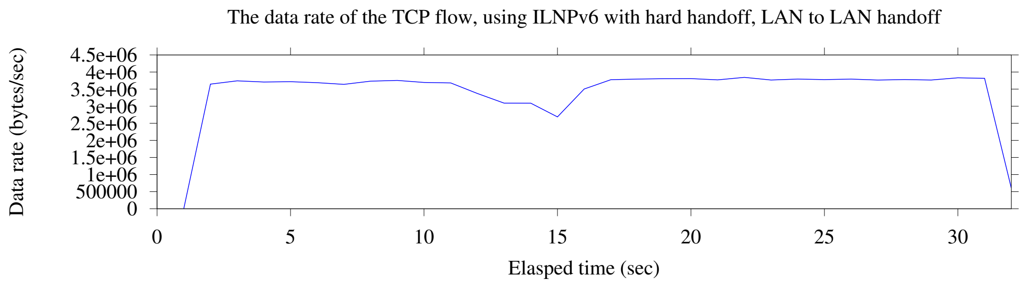
The TCP flows in this handoff scenario were in the opposite direction to the previous one. The TCP throughput increased after the MN handoff from the WAN network to the LAN network, as displayed in Figure 5.6. However, this was not true for MIPv6 without RO, because the traffic still had to traverse the HA, which resided across the WAN link. ILNPv6 with soft handoff still showed the best behaviour: the



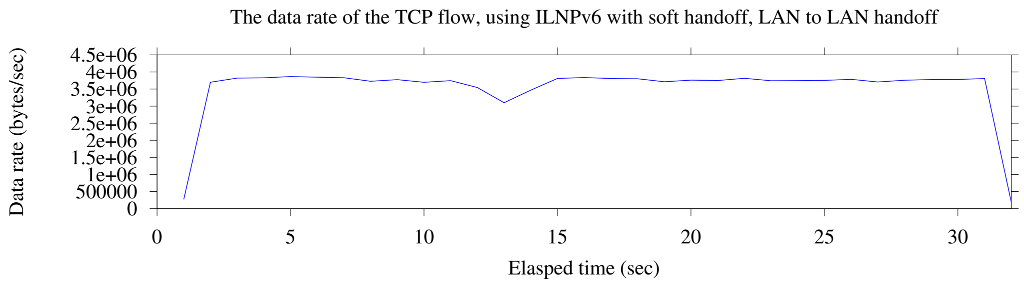
(A) MIP without RO.



(B) MIP with RO.

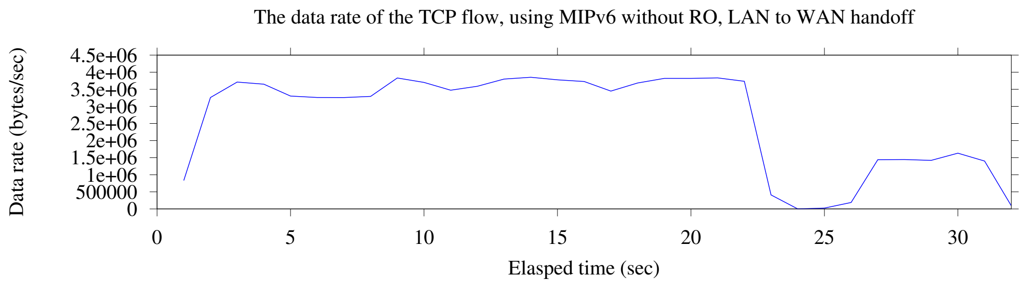


(C) ILNP hard handoff.

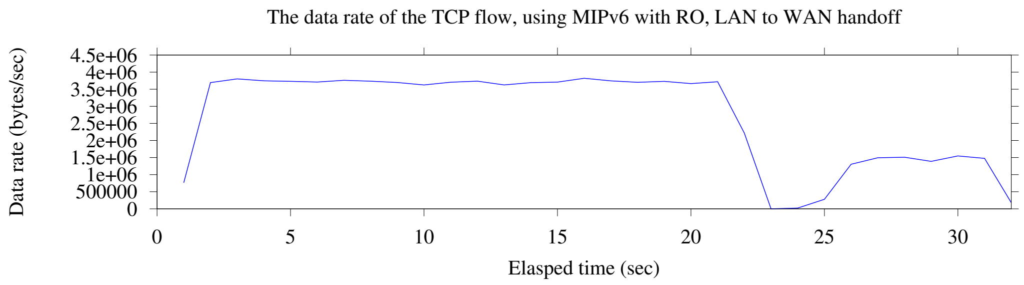


(D) ILNP soft handoff.

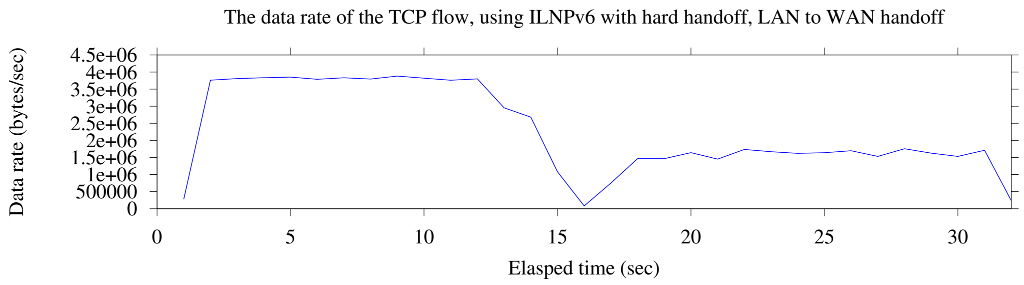
FIGURE 5.4. Communication flow for the TCP traffic showing the data rate (bytes/sec) at the MN, LAN to LAN handoff. There was an interruption during the MN handoff for MIPv6, while there was a small drop in data rate for ILNPv6.



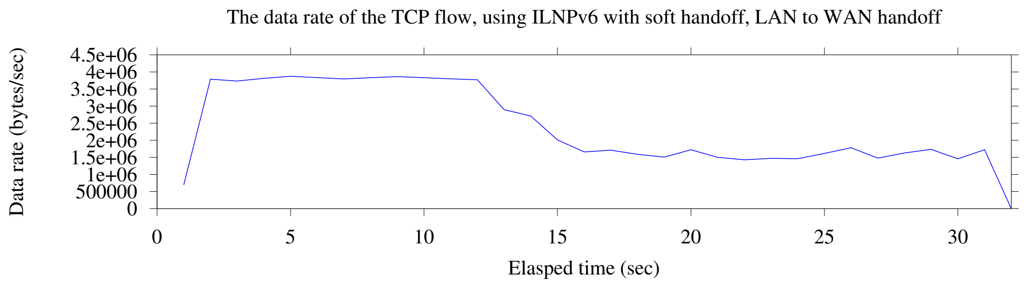
(A) MIP without RO.



(B) MIP with RO.

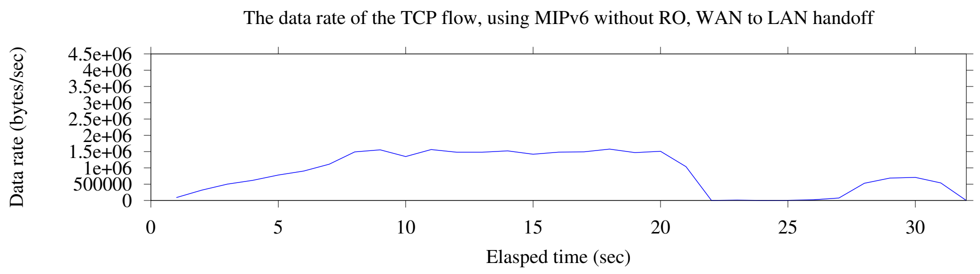


(C) ILNP hard handoff.

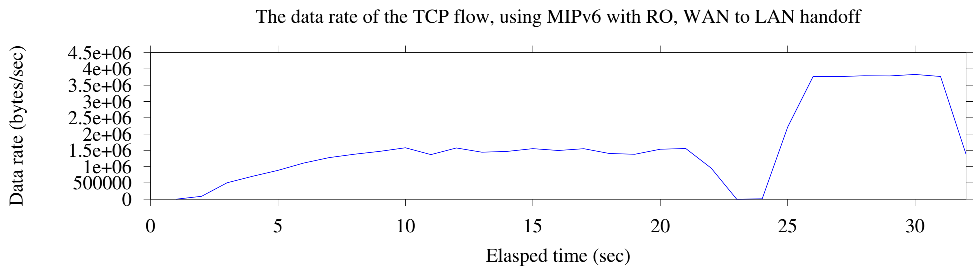


(D) ILNP soft handoff.

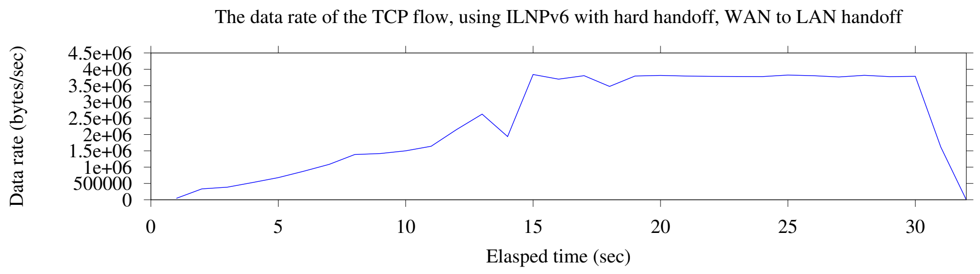
FIGURE 5.5. Communication flow for the TCP traffic showing the data rate (bytes/sec) at the MN, LAN to WAN handoff. There was an interruption during the MN handoff for MIPv6 and ILNPv6 hard handoff, while there was no interruption for ILNPv6 soft handoff. The data rate dropped after the handoff due to higher delay.



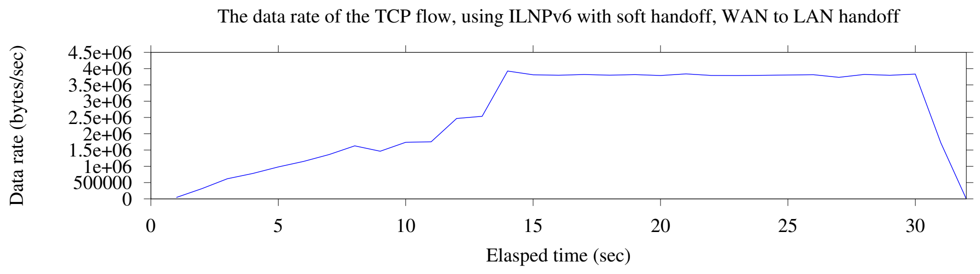
(A) MIP without RO.



(B) MIP with RO.



(C) ILNP hard handoff.



(D) ILNP soft handoff.

FIGURE 5.6. Communication flow for the TCP traffic showing the data rate (bytes/sec) at the MN, WAN to LAN handoff. There was an interruption during the MN handoff for MIPv6, while there was no interruption for ILNPv6. The data rate increased after the handoff due to a lower delay link, except MIPv6 without RO, where the traffic still traversed the high delay link.

flow gradually increased without any interruption, as the TCP algorithm adjusted to the lower RTT.

For both ILNPv6 and MIPv6, the throughput at the beginning of the flow was quite low, and it gradually increased. This was caused by the TCP slow start mechanism [4]. Due to the high delay link, the slow start process took some time before reaching the threshold, when the maximum data rate could be achieved. The slow start process was much faster in the LAN environment: it can be seen in Figure 5.4 and 5.5 that TCP throughput climbed faster at the beginning of the flows, and this was normal TCP behaviour.

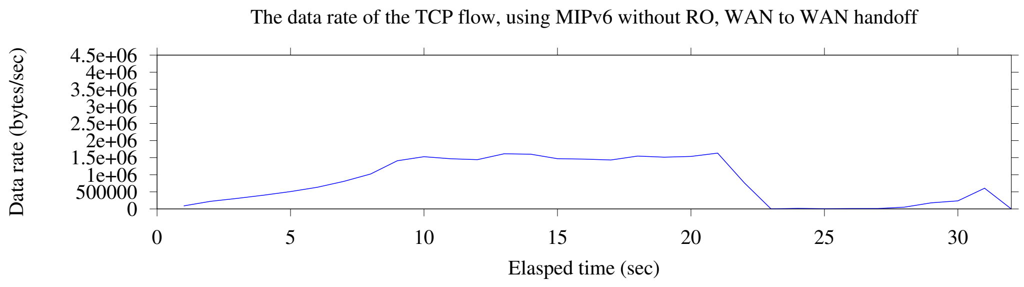
WAN to WAN handoff

ILNPv6 with soft handoff still outperformed the others in this environment. As shown in Figure 5.7, there was no interruption of the flow for ILNPv6 soft handoff. There was a small interruption when ILNPv6 operate with hard handoff. For MIPv6, a longer interruption was found. Moreover, the MN suffered the most when RO was disabled. After the handoff, the throughput was very poor because the traffic traversed the HA over a WAN link and was then forwarded to the MN via another WAN link. So, the RTT was very high and TCP had unsatisfactory performance. Again, TCP slow start can be observed at the beginning of the flows.

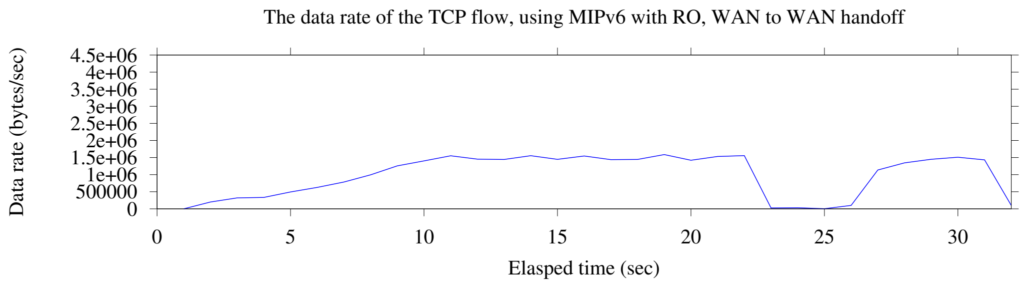
5.3.2. Successfully Transferred Data.

Figure 5.8 shows the amount of data (in MB) that was transferred during the 30 second TCP flow using ILNPv6 and MIPv6 under different handoff scenarios. In a pure LAN environment, TCP sent the highest amount of data because the RTT was low and so the TCP throughput was high – TCP throughput and RTT are inversely proportional [36]. Therefore, in the WAN network (higher RTT), TCP sent less data.

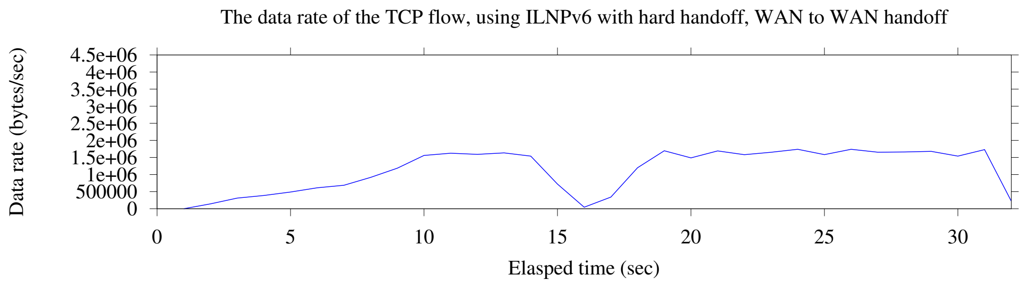
In most cases, more data was sent when ILNPv6 was used instead of MIPv6, because the data was sent without interruption (soft handoff) or minimal interruption (hard handoff) – see Section 5.3.1. However, this was not true for the LAN to WAN handoff: more data was sent when MIPv6 was used. As mentioned previously, for ILNPv6, the handoff to the WAN network occurs when the MN enters the overlap



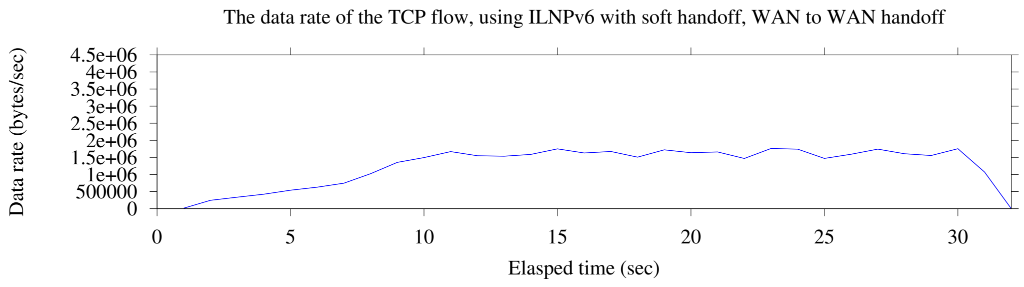
(A) MIP without RO.



(B) MIP with RO.

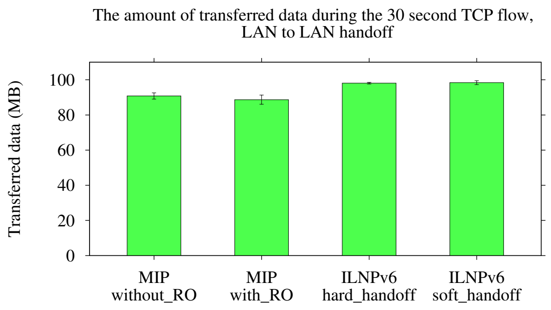


(C) ILNP hard handoff.

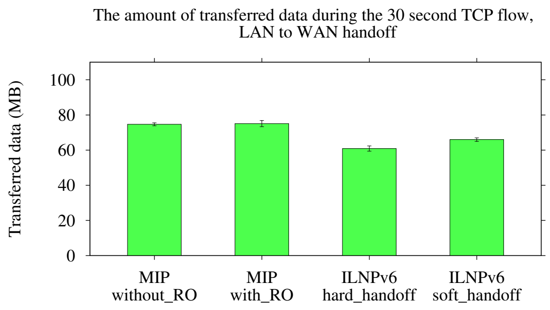


(D) ILNP soft handoff.

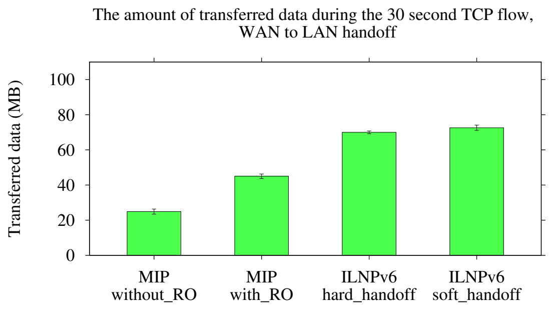
FIGURE 5.7. Communication flow for the TCP traffic showing the data rate (bytes/sec) at the MN, WAN to WAN handoff. There was an interruption during the MN handoff for MIPv6 and ILNPv6 hard handoff, while there was no interruption for ILNPv6 soft handoff.



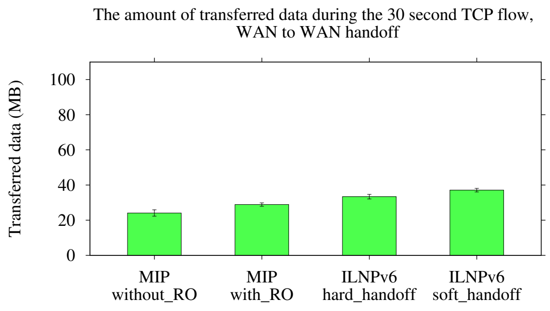
(A) LAN to LAN handoff.



(B) LAN to WAN handoff.



(C) WAN to LAN handoff.



(D) WAN to WAN handoff.

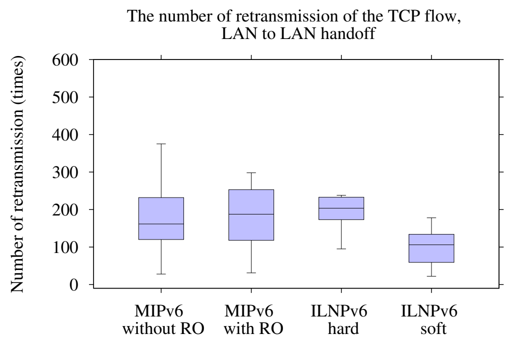
FIGURE 5.8. Total data transfer volumes during 30 second TCP traffic flows. Error bars at 95% confidence. With ILNPv6, TCP can transfer more data than with MIPv6 in almost every case, except when handoff from LAN to WAN because the MN stays in the LAN link longer when MIPv6 is in used.

area (t=5s, in this experiment). So, it spent a much longer time on the WAN network than when using MIPv6, which hands off to the WAN network after it moves out of the LAN network (t=20s, in this experiment). Provided that more data can be sent in the LAN network, it is understandable that MIPv6 allowed more data to be sent in this case. Hence, to maximise TCP performance, an adjustment to the ILNPv6 handoff decision algorithm may be required, e.g. let the MN stay in a 'better' link as long as possible before handoff. Of course, this may need information from the link layer if the MN should handoff, e.g. when the signal strength of the better link is too weak. Similarly, in the WAN to LAN handoff case, a lot more data was sent for ILNPv6, partly because the MN stayed in the LAN network longer than MIPv6 case.

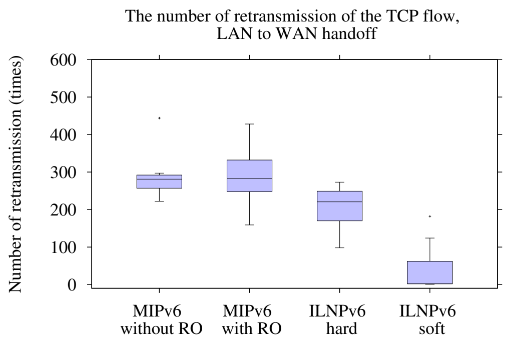
For MIPv6, RO improved the amount of data that was sent if the HA resided in the WAN network (i.e. the WAN to LAN handoff and WAN to WAN handoff cases), since data packets did not have to traverse to the high delay link (causing an increase in RTT).

5.3.3. Retransmission Attempted.

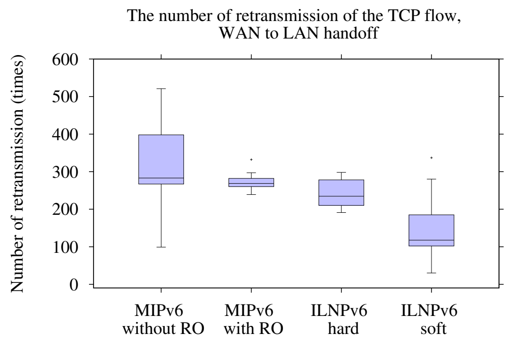
The number of TCP retransmissions in each handoff scenario is shown in Figure 5.9. This number was measured at the CN by counting sent packets having the same TCP sequence number.



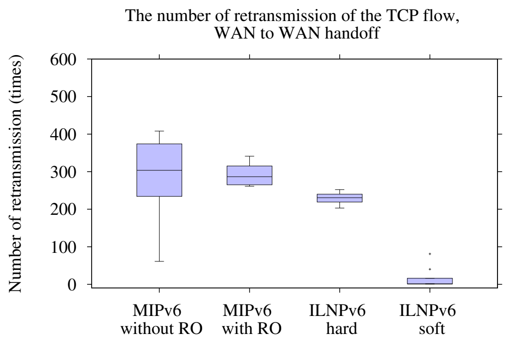
(A) LAN to LAN handoff.



(B) LAN to WAN handoff.



(C) WAN to LAN handoff.



(D) WAN to WAN handoff.

FIGURE 5.9. Number of retransmission attempted at the CN. ILNP with soft handoff had the lowest number of retransmissions.

MIPv6 had a high number of retransmissions in every case. This was mostly caused by packet loss during handoff when the MN could not receive any data. Theoretically, ILNPv6 should have much fewer retransmission attempts than MIPv6. Especially, ILNPv6 with soft handoff should have nearly zero retransmissions because nearly zero packet loss occurred. However, ILNPv6 surprisingly had some amount of retransmissions. For hard handoff, the numbers were close to MIPv6 case. For soft handoff, lower retransmissions were observed, and the numbers were quite low during the MN handoff to the WAN network. These retransmissions were caused by packet queuing and misordering from intense TCP traffic – see below for further information in each scenario.

In the LAN to LAN handoff scenario, the TCP traffic rate was very high ($\sim$30 Mbps), so some packets were queued at the MN before getting processed. When a handoff occurred, for the ILNP case, packets that came via the new link did not have any queue, and thus got processed before the queued packets at the old link. Therefore, TCP received packets with a 'jump' in sequence number, and duplicate acknowledgements were sent back to the CN because the MN interpreted this as that the queued packets in the old link as being lost. Finally, the CN, which received many duplicate acknowledgements, had to retransmit the queued packets. Note that this is normal TCP behaviour. For hard handoff, similar things occurred, but all queued packets at the MN would eventually be dropped at the network layer due to their stale L64 value.

For the WAN to LAN handoff scenario, similar behaviour was also observed because of a similar reason: after handoff, packets arrived at the new link before packets from the old link got processed. However, this was not because of a queue at the old link – since the TCP throughput in the WAN link is not as high as in the LAN (less than 12.8 Mbps, see the calculation above) – but because the new link has much lower delay.

For the LAN to WAN case, ILNPv6 with soft handoff had a lower number of retransmissions. This is because, after handoff, the new link was much slower than the old link, so quite a small number of packets arrived at the new link while the packets from the old link were queued, and fewer duplicate acknowledgements were sent to the CN. ILNPv6 with hard handoff still had some retransmissions owing to packet loss during handoff.

For the WAN to WAN handoff case, ILNPv6 with soft handoff, again, had a very low number of retransmissions, almost zero. As mentioned before, the TCP traffic in the WAN is of a lower rate than in the LAN, so no problem of queued packets.

For ILNPv6 with hard handoff, again, some retransmissions were still observed due to packet loss during handoff.

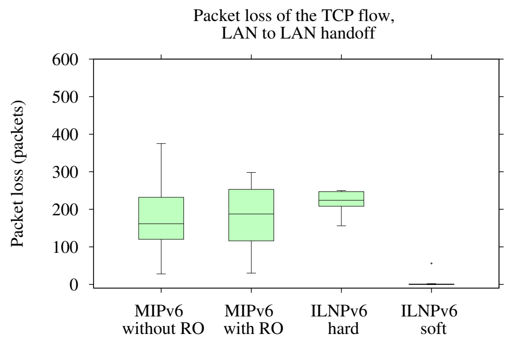
Overall, ILNPv6 with soft handoff had the lowest number of retransmissions for every case, which implied lowest packet loss. However, some optimisations are needed in order to improve TCP performance with ILNPv6 since a number of retransmissions happen not because of loss, but due to the packet misordering. This is a multipath effect – traffic for a single flow traverses two paths, each with different characteristics – something which “standard” TCP is not designed to deal with.

5.3.4. Packet Loss.

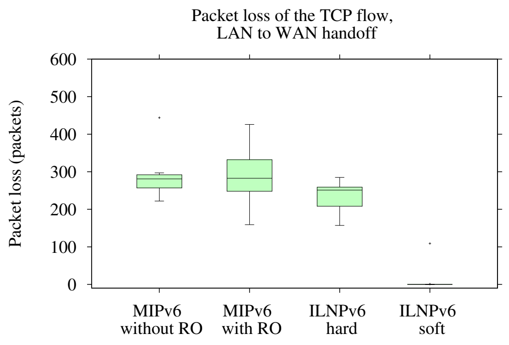
As previously discussed, the number of retransmissions was not always caused by packet loss. To measure the actual number of packets lost, the retransmitted packets at the CN were checked against the received packets at the MN by matching the TCP sequence numbers. So, if a packet was retransmitted at the CN (i.e. it has duplicate TCP sequence number at the CN), but was not received at the MN (i.e. no duplicate TCP sequence number for such a packet), that meant the packet was actually lost. However, if the duplicate packet was also received at the MN, the packet was not lost, and the retransmission was for other reasons, such as misordering. For ILNPv6 hard handoff, some packets were received at the MN, but were dropped at the network layer due to stale L64 values – these were also counted as packet loss.

Figure 5.10 shows the number of packet lost in each handoff scenario with MIPv6 and ILNPv6. Apart from ILNPv6 soft handoff, all results are similar to the retransmission graph (Figure 5.9). This means that retransmissions in MIPv6, both with and without RO, and ILNPv6 hard handoff were caused by packet loss. However, retransmissions in ILNPv6 with soft handoff were likely to be caused by other reasons (as discussed earlier) because the number of packets lost is much lower than the number of retransmissions, and is close to zero.

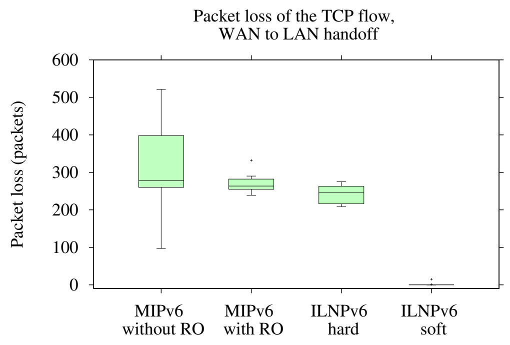
ILNPv6 hard handoff had similar levels of packet loss as MIPv6, however, much less interruption time in the flow during handoff (see section 5.3.1). MIPv6 did not have higher packet loss because there were no packets sent from the CN during the interruption time after the TCP window was full. Considering the TCP window size



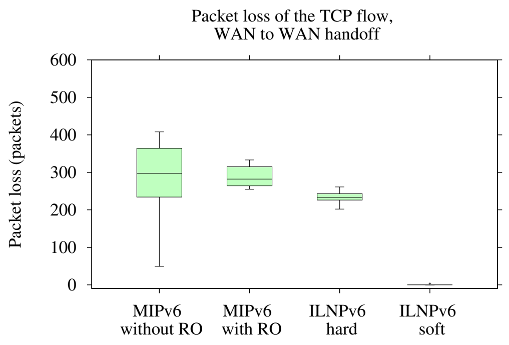
(A) LAN to LAN handoff.



(B) LAN to WAN handoff.



(C) WAN to LAN handoff.



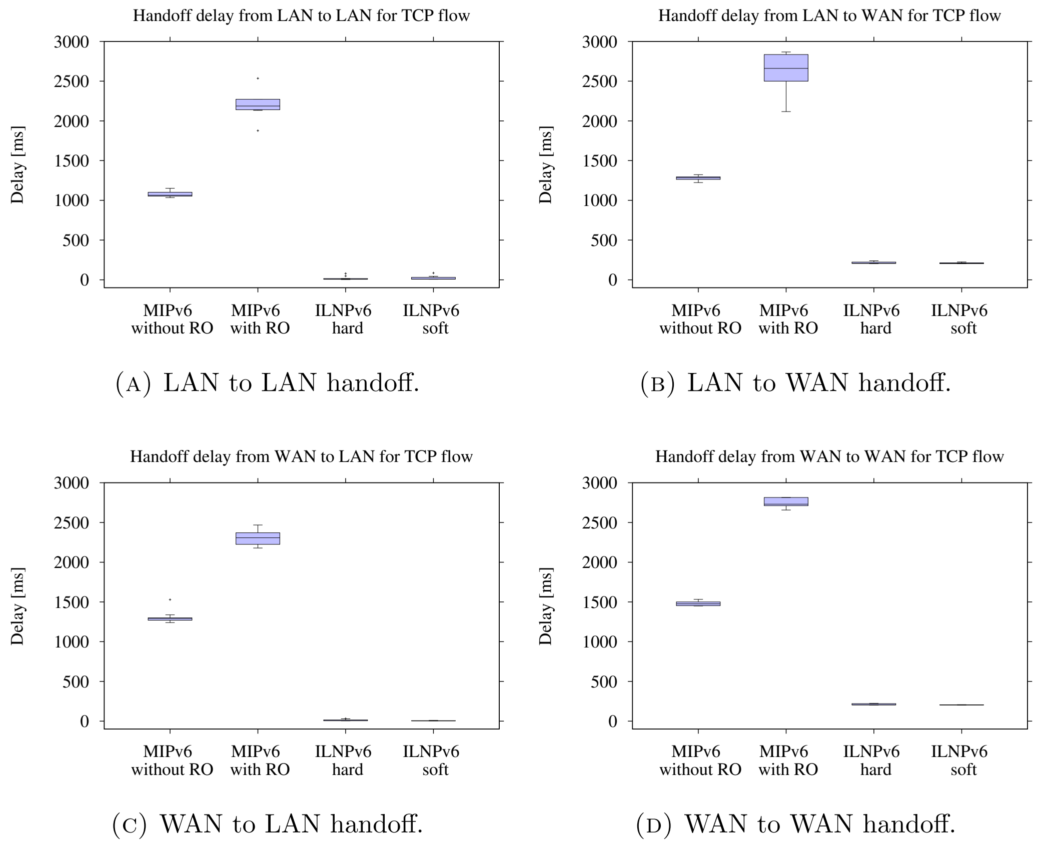
(D) WAN to WAN handoff.

FIGURE 5.10. Number of lost packets in each flow. ILNP with soft handoff has the lowest number of packet loss, and is near to zero.

of 320 Kbytes, and each TCP packet of 1400 byte payload (according to the *tcpdump* log), the number of packets that can be sent without receiving an acknowledgement is $(320 \cdot 1024)/1400 \approx 234$ packets, which is close to the median values of packet loss for MIPv6 and ILNPv6 hard handoff in Figure 5.10. The number could rise or fall if i) packets are sent of smaller size; or ii) some packets are retransmitted more than once. ILNPv6 hard handoff had a smaller variation for loss than MIPv6.

5.3.5. Handoff Delay.

Figure 5.11 presents handoff delay for TCP flows. Again, similar to the UDP tests, this shows only *network layer handoff delay*, related to the handoff signalling for MIPv6 (BU/BAck) and ILNPv6 (LU/LU-ACK). It does not include lower layer delays.



(A) LAN to LAN handoff.

(B) LAN to WAN handoff.

(C) WAN to LAN handoff.

(D) WAN to WAN handoff.

FIGURE 5.11. Handoff delay for TCP traffic flows. ILNP handoff was ∼1 RTT.

As expected, handoff delay for TCP had a similar trend to UDP – see Figure 4.6 and Figure 4.7 in Section 4.4.1. ILNPv6 had a much shorter handoff delay than MIPv6 in every case, and close to the RTT. MIPv6 had an additional delay of ∼1 second, and longer delay was found if RO was enabled.

Comparing to UDP, the handoff delay for TCP was slightly higher. This was because the TCP traffic was at a higher rate than the UDP tests, so signalling packets might be slightly delayed in sending, receiving and processing. Like the UDP results, handoff delay of MIPv6 had higher variance than ILNPv6 due to more scope of delay: computation at the HA and the RO process.

5.4. Discussion and Future Works

Comparing to the UDP results in Chapter 4, TCP performance for host mobility still has room to improve. Nevertheless, with ILNPv6, TCP handoff still performed better than with MIPv6: i) With ILNPv6 soft handoff, there was no interruption of the flows for every scenario (Section 5.3.1); ii) More data can be transferred under the same time frame, in most cases (Section 5.3.2); iii) ILNPv6 soft handoff had the lowest number of retransmission attempts (Section 5.3.3) and nearly zero packet loss (Section 5.3.4); and iv) Handoff delay was much shorter (Section 5.3.5).

To maximise TCP performance, additional tuning is required. The key finding here is TCP must be able to deal with packet misordering that is caused by the multipath effect during ILNPv6 handoff, as shown in Section 5.3.3. Of course, this enables an alternative mechanism to achieve multipath transfer using TCP in multihomimg scenarios, equivalent to MP-TCP [46, 47]. The initial work on ILNPv6 mutihoming shows satisfy performance in the network layer [106]. However, investigation of using TCP in ILNPv6 multihoming is a subject for further study, and is out of scope of this work.

There are also other possible optimisations to TCP. First, the queuing algorithm at routers and the MN may have to change to prioritise some ICMP packets such as RA and handoff signals, so that such packets do not get blocked or delayed by the intense TCP traffic (see Section 5.2). Second, the ILNPv6 handoff decision algorithm may need improvement, allowing an MN to utilise a better quality link as much as possible (see Section 5.3.2). Finally, there are many variants of TCP congestion control mechanisms, and the evaluation here used the Linux default setting (i.e. TCP CUBIC [53]). Further studies of using ILNPv6 with other TCP variants may be beneficial, especially TCP Hybla [30] and TCP Veno [48]; the former is designed for heterogeneous networks, and the latter is optimised for working in wireless networks. With those variants, a better performance (e.g. fewer retransmissions) may be found for both ILNPv6 and MIPv6. However, most TCP enhancements for mobile and wireless environments has been designed for hard handoff solutions – to deal with loss, errors and disconnections [42]. A new enhancement to TCP congestion control to work with soft handoff (to deal with multipath effect) is desirable.

5.5. Summary

This chapter has presented how the TCP data path and the TCP control block (e.g. session binding and checksum calculation) in the Linux kernel was extended to operate over ILNPv6. Legacy TCP applications using socket API can be used unmodified.

Most TCP results confirmed the expectation mentioned earlier in this chapter. The performance evaluation showed that ILNPv6 provided better mobility support for TCP than MIPv6. However, there were some surprise results. First, although TCP performed the best when ILNPv6 with soft handoff was used, there is still engineering optimisation needed, unlike for UDP, where soft handoff provided ideal performance. A new TCP congestion control algorithm that deals with the multipath effects of soft handoff is desirable. Second, ILNPv6 with soft handoff clearly had better performance than the hard handoff mode for TCP. For UDP there was just a small drop of throughput during the MN handoff, but for TCP, the hard handoff mode caused an interruption in the flow and caused a higher amount of packet loss. Third, for MIPv6, TCP performed much worse than UDP, especially when involving WAN links because the RTT impacted TCP performance, due to the use of RTT in operation of the congestion control algorithm.

Overall TCP is much more sensitive to packet loss and RTT than UDP, and hence causing the handoff performance, while using both MIPv6 and ILNPv6, to be less than optimal.

Chapter 6

Control Plane Analyses

Chapter 4 and 5 provide detailed performance evaluation in the *data plane*. This chapter presents comprehensive analyses of the *control plane*. The control signals of both ILNPv6 and MIPv6 are independent to the data plane, so they are similar for any transport protocol in any communication sessions, i.e. same control signals are used for mobility support in both UDP and TCP sessions. Therefore, these analyses are applicable for UDP, TCP and other transport protocols as well.

6.1. Overview of Signalling Used

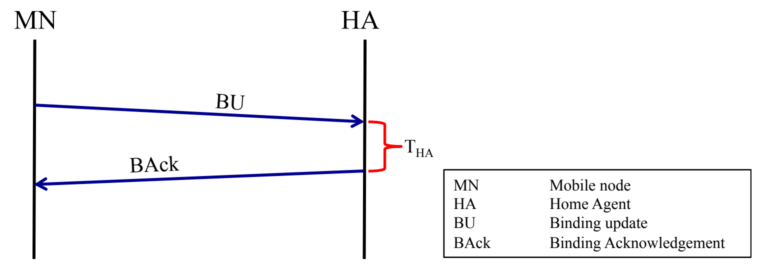


FIGURE 6.1. Control signals used during an MN handoff using MIPv6 without RO.

First, Figure 6.1 shows the control signals used for handoff management for MIPv6, when RO is disabled. There are only two control signals required: i) BU from the MN to the HA to inform a change of the CoA, and ii) BAck sent back from the HA to the MN after the BU has been processed. Here, T_{HA} denotes as the processing time at the HA.

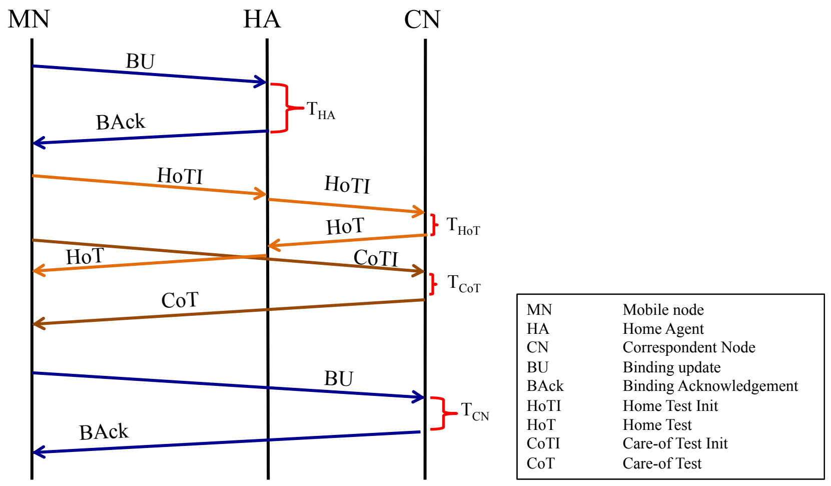


FIGURE 6.2. Control signals used during an MN handoff using MIPv6 with RO.

If RO is enabled, there are additional control signals as shown in Figure 6.2, including:

- *Home Test Init (HoTI)* and *Home Test (HoT)* for testing the reachability of the HoA from the CN. T_{HoT} is the processing time of HoT at the CN, both HoTI and HoT must go through the HA.
- *Care-of Test Init (CoTI)* and *Care-of Test (CoT)* for testing that the new CoA of the MN is reachable from the CN. T_{CoT} is a processing time of CoT at the CN.
- BU and BAck between the MN and the CN to update the CoA. T_{CN} is the processing time of a BU at the CN.

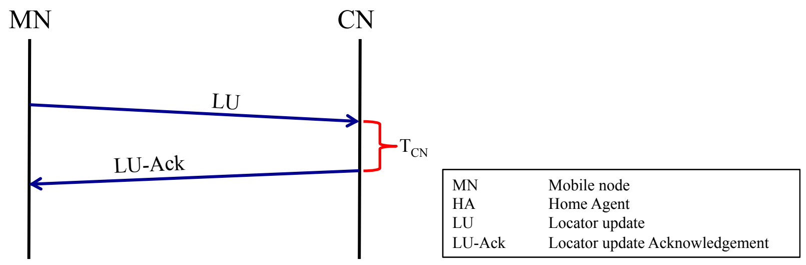


FIGURE 6.3. Control signals used during an MN handoff using ILNPv6.

As shown in Figure 6.3, ILNPv6 uses only 2 packets for handoff signalling: LU and LU-ACK. The signalling packets go directly between the MN and the CN to update the value of *L64* of the MN. T_{CN} is the processing time of LU at the CN.

In a real network, the MN may also update the L64 value to the DNS. However, this process is not included in the analyses in this chapter because it is not always necessary for an MN. A DNS update is required only if the MN expects incoming sessions (i.e. it is a mobile server), which is uncommon today. Moreover, some applications such as Skype do not rely on DNS, owing to its peer-to-peer model. Skype has its own resolver mechanism [21]. Of course, further studies are required for integrating ILNPv6 with *rendezvous* services (DNS and other specific services) as well as analyses of the control signals used.

Overall, ILNPv6 handoff signals are of the same order as MIPv6 without RO. However, as widely known, MIPv6 has a serious performance problem, *triangular routing*, if RO is disabled. The results in Chapter 5 clearly show that TCP performance of MIPv6 without RO is very poor, especially when the home network is accessed via a high latency (WAN) link. MIPv6 with RO produces a lot more signalling packets, which means more overhead for the network.

The following three sections are analyses of ILNPv6 and MIPv6 handoff signalling under different circumstances to answer the following questions:

(1) How do lossy networks impact handoff signalling? What is the success rate of a complete handoff under those scenarios?
(2) What are the impacts of a loss of a signalling packet? How does it affect handoff duration and interruption time?
(3) How much overhead, in terms of number of packets and number of bytes would ILNPv6 and MIPv6 generate if the protocols are used at scale?

6.2. Impacts of Lossy Environment

This section shows how a lossy network, which is a common problem in the mobile and wireless networks, due to their high bit error rates, impacts handoff success rate. The same network topology, as in the performance evaluation in Chapter 4 and 5, is used for this analysis, as shown in Figure 6.4. Combinations of packet loss rates are introduced (1) between R1 and R2 (HA), the old link before the MN handoff, and (2)

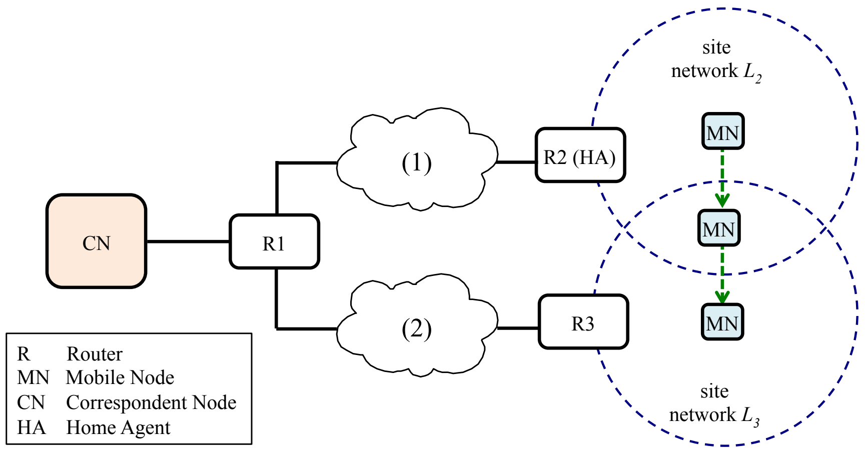


FIGURE 6.4. Topology for signalling analysis. Combinations of packet loss rates are introduced at (1) and (2).

between R1 and R3, the new link after the MN handoff. There is no additional loss introduced between MN and R2 (HA), between MN and R3, and between R1 and CN. This is just an example case to study the impact of *indicative loss* at certain points in the network. There are more scenarios in the real Internet, for example, a shorter path to the HA, or lossy CN–R1 link. Further investigation in an *operational scenario* would be required to provide an absolute result under different circumstances.

6.2.1. Handoff Success Rate Calculation.

According to Figure 6.1 - 6.3, the handoff is completed when all signalling packets sucessfully reach the destination. P_X is denoted as a probability of the success of X. So, for MIPv6 without RO, the probability that a handoff is successful ($P_{MIPv6noRO}$) is the probability that a BU to HA is successful ($P_{BU_{HA}}$) and probability that a BAck from HA is successful ($P_{BAck_{HA}}$). Note that $P_{BU_{HA}}$ and $P_{BAck_{HA}}$ are independent.

(1) $$P_{MIPv6noRO} = P_{BU_{HA}} \cdot P_{BAck_{HA}}$$

Likewise, the probability that a handoff is successful for MIPv6 with RO ($P_{MIPv6wRO}$) is the probability that all control signals are completed, including BU to HA ($P_{BU_{HA}}$), BAck from HA ($P_{BAck_{HA}}$), HoTI (P_{HoTI}), HoT (P_{HoT}), CoTI (P_{CoTI}), CoT (P_{CoT}), BU to CN ($P_{BU_{CN}}$), and BAck from CN ($P_{BAck_{CN}}$). Again, those signalling packets are all independent.

$$\begin{aligned} P_{MIPv6wRO} = P_{BU_{HA}} \cdot P_{BAck_{HA}} \cdot P_{HoTI} \cdot P_{HoT} \\ \cdot P_{CoTI} \cdot P_{CoT} \cdot P_{BU_{CN}} \cdot P_{BAck_{CN}} \end{aligned} \tag{2}$$

Finally, the probability that a handoff is successful for ILNPv6 (P_{ILNPv6}) is the probability that LU is completed (P_{LU}) and the probability that LU-ACK is completed (P_{LU-ACK}). P_{LU} and P_{LU-ACK} are also independent.

$$P_{ILNPv6} = P_{LU} \cdot P_{LU-ACK} \tag{3}$$

Deriving from the topology in Figure 6.4, Figure 6.5 illustrates the path of each control signal used in a handoff process for both MIPv6 and ILNPv6. The probability that each control signal is successfully delivered is the probability that the signalling packet can travel from the sender to the receiver. For example, the probability that a BU is delivered is the probability that BU can travel from MN to R3 (P_{MN-R3}), from R3 to R1 (P_{R3-R1}), and finally from R1 to HA (P_{R1-HA}).

$$P_{BU_{HA}} = P_{MN-R3} \cdot P_{R3-R1} \cdot P_{R1-HA} \tag{4}$$

Therefore, with similar calculations, the probability that other control signals are successfully delivered are:

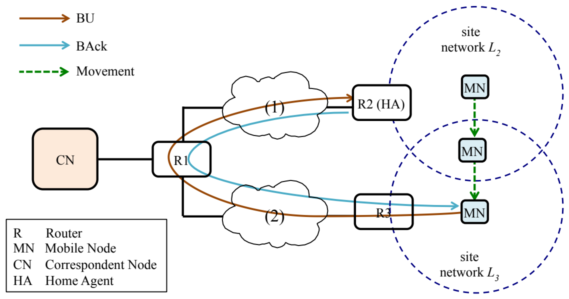
$$P_{BAck_{HA}} = P_{HA-R1} \cdot P_{R1-R3} \cdot P_{R3-MN} \tag{5}$$

$$P_{HoTI} = P_{MN-R3} \cdot P_{R3-R1} \cdot P_{R1-HA} \cdot P_{HA-R1} \cdot P_{R1-CN} \tag{6}$$

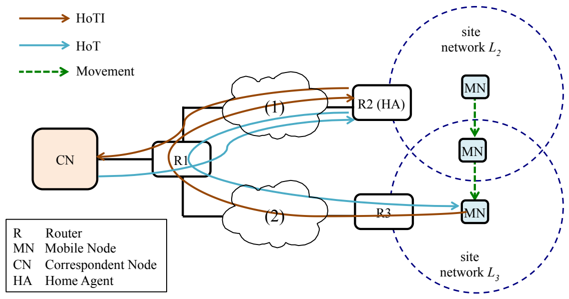
$$P_{HoT} = P_{CN-R1} \cdot P_{R1-HA} \cdot P_{HA-R1} \cdot P_{R1-R3} \cdot P_{R3-MN} \tag{7}$$

$$P_{CoTI} = P_{BU_{CN}} = P_{LU} = P_{MN-R3} \cdot P_{R3-R1} \cdot P_{R1-CN} \tag{8}$$

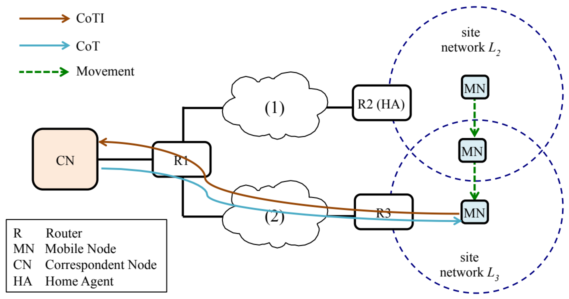
$$P_{CoT} = P_{BAck_{CN}} = P_{LU-ACK} = P_{CN-R1} \cdot P_{R1-R3} \cdot P_{R3-MN} \tag{9}$$



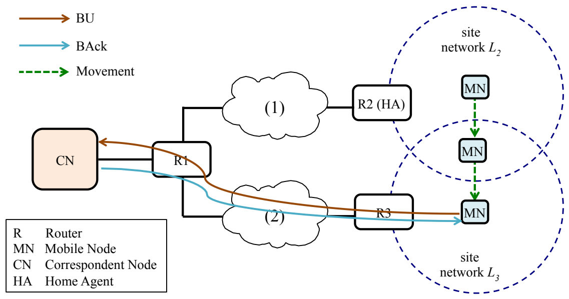
(A) BU/BAck to/from HA.



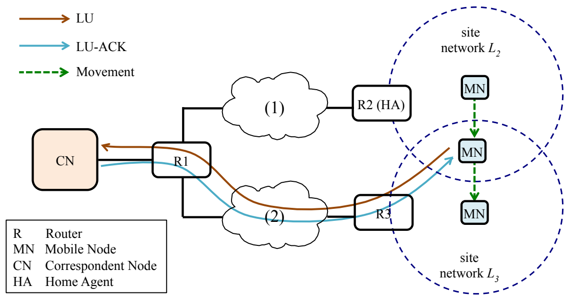
(B) HoTI/HoT.



(C) CoTI/CoT.



(D) BU/BAck to/from CN.



(E) LU/LU-ACK.

FIGURE 6.5. Control signal paths during an MN handoff.

In this analysis, as mentioned above, there is no loss between MN and R3, and between CN and R1, therefore

$$P_{MN-R3} = P_{R3-MN} = P_{CN-R1} = P_{R1-CN} = 1 \tag{10}$$

Hence, those individual probabilities can be removed from the equations (4) – (9), and could be rewritten as follows.

$$P_{BU_{HA}} = P_{R3-R1} \cdot P_{R1-HA} \tag{11}$$

$$P_{BAck_{HA}} = P_{HA-R1} \cdot P_{R1-R3} \tag{12}$$

$$P_{HoTI} = P_{R3-R1} \cdot P_{R1-HA} \cdot P_{HA-R1} \tag{13}$$

$$P_{HoT} = P_{R1-HA} \cdot P_{HA-R1} \cdot P_{R1-R3} \tag{14}$$

$$P_{CoTI} = P_{BU_{CN}} = P_{LU} = P_{R3-R1} \tag{15}$$

$$P_{CoT} = P_{BAck_{CN}} = P_{LU-ACK} = P_{R1-R3} \tag{16}$$

Finally, by replacing (11) – (16) in (1) – (3), the probability of a successful handoff using MIPv6, with and without RO, and for ILNPv6 can be calculated as follows.

$$P_{MIPv6noRO} = (P_{R3-R1} \cdot P_{R1-HA}) \cdot (P_{HA-R1} \cdot P_{R1-R3}) \tag{17}$$

$$\begin{aligned} P_{MIPv6wRO} = {} & (P_{R3-R1} \cdot P_{R1-HA}) \cdot (P_{HA-R1} \cdot P_{R1-R3}) \\ & \cdot (P_{R3-R1} \cdot P_{R1-HA} \cdot P_{HA-R1}) \cdot (P_{R1-HA} \cdot P_{HA-R1} \cdot P_{R1-R3}) \\ & \cdot (P_{R3-R1} \cdot P_{R1-R3})^2 \end{aligned} \tag{18}$$

$$P_{ILNPv6} = P_{R3-R1} \cdot P_{R1-R3} \tag{19}$$

6.2.2. Loss Scenarios.

Referring to Figure 6.4, combinations of loss are introduced at the old link – between HA and R1, position (1) – as well as the new link – between R3 and R1, position (2). Combinations of packet loss rates of 1%, 2%, 5% and 10% are used for this analysis. Table 6.1 presents values of each individual probability used in equation (17) – (19) under each scenario.

6.2.3. Results.

By replacing the number from Table 6.1 in equation (17) – (19), Table 6.2 presents

Table 6.1. Individual probabilities under different packet loss rates.

% loss, new link	% loss, old link	$P_{R1-R3} = P_{R3-R1}$	$P_{R1-HA} = P_{HA-R1}$
1	1	0.99	0.99
1	2	0.99	0.98
1	5	0.99	0.95
1	10	0.99	0.90
2	1	0.98	0.99
2	2	0.98	0.98
2	5	0.98	0.95
2	10	0.98	0.90
5	1	0.95	0.99
5	2	0.95	0.98
5	5	0.95	0.95
5	10	0.95	0.90
10	1	0.90	0.99
10	2	0.90	0.98
10	5	0.90	0.95
10	10	0.90	0.90

the handoff success rates for MIPv6, both with and without RO, and ILNPv6 in each scenario.

As expected, the handoff success rate decreases when the packet loss rate increases. MIPv6 is affected by loss in both the old link and the new link, while only loss in the new link impacts ILNPv6. This is because ILNPv6 uses only the new link for sending the signalling packets (LU and LU-ACK). So, an increase in the packet loss rate of the old link does not affect the handoff success rate for ILNPv6.

Under a low loss environment, i.e. 1% and 2% loss, there are only slight differences for the handoff success rate of MIPv6 without RO and ILNPv6. MIPv6 clearly has lower success rate than ILNPv6 when more loss (5% and 10%) is induced. When RO is enabled, MIPv6 is more sensitive to loss because there are a lot more signalling packets involved. The handoff success rate is as low as 75% when the old link and the new link have only 2% packet loss rate. Moreover, the number drops to below 50% when more than 5% packet loss rate is induced in both links.

Overall, ILNPv6 is more tolerant to the lossy environment. The handoff success rate is more than 90% if the network has less than 5% loss. Even in a very poor network, e.g. 10% loss rate, handoff using ILNPv6 still has more than 80% success rate.

TABLE 6.2. Probability of handoff success rate, under different loss conditions

Old link loss \ New link loss	1%	2%	5%	10%
1%	0.96	0.94	0.88	0.79
2%	0.94	0.92	0.87	0.78
5%	0.88	0.87	0.81	0.73
10%	0.79	0.78	0.73	0.66

(A) MIPv6 without RO

Old link loss \ New link loss	1%	2%	5%	10%
1%	0.86	0.80	0.62	0.40
2%	0.82	0.75	0.59	0.38
5%	0.68	0.63	0.49	0.32
10%	0.49	0.45	0.35	0.23

(B) MIPv6 with RO

Old link loss \ New link loss	1%	2%	5%	10%
1%	0.98	0.96	0.90	0.81
2%	0.98	0.96	0.90	0.81
5%	0.98	0.96	0.90	0.81
10%	0.98	0.96	0.90	0.81

(C) ILNPv6

6.3. Impacts of Signalling Packets Loss

Under lossy conditions, there are chances that handoff signals get lost. As previously shown above, the handoff success rate goes down when the network has higher loss rates. This section describes impacts, when a signalling packet is lost, to the handoff process in terms of handoff delay and interruption time.

- Handoff delay – is the duration that an MN used to complete a handoff process.
- Interruption time – is the duration that data packets could not be delivered to the MN.

This analysis concerns simple scenarios when only *one* of the control signal get lost for each handoff.

6.3.1. No Signalling Packet Loss.

According to Figure 6.1 - Figure 6.3, when there is no signalling packet loss, handoff delay and interruption time could be calculated as follows.

(a). *MIPv6 without RO.*

- Handoff delay – is calculated from the time that an MN sends out BU and the time that BAck is received at the MN.

$$Delay = RTT_{MN-HA} + T_{HA} \tag{20}$$

- Interruption time – is calculated from the time that a BU packet needs for tranmission from an MN to an HA as well as the time that the HA needs to process the BU.

$$T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA} \tag{21}$$

(Assuming that the path is symmetric.)

(b). *MIPv6 with RO.*

- Handoff delay – is calculated from the total time that is needed for an update to the HA, a return routability procedure (the HoT and CoT can happen simultaneously), and an update to the CN.

$$\begin{aligned} Delay &= BU_{HA} + max(HoT, CoT) + BU_{CN} \\ &= RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} \\ &\quad + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})] \end{aligned} \tag{22}$$

- Interruption time – is similar to MIPv6 without RO, because data packets can go through the HA, without interruption, when RO is in progress.

$$T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA} \tag{23}$$

(Assuming that the path is symmetric.)

(c). *ILNPv6.*

- Handoff delay – is calculated from the time that an MN sends out LU and the time that LU-ACK is received at the MN.

$$Delay = RTT_{MN-CN} + T_{CN} \tag{24}$$

- Interruption time – for hard handoff, this is calculated from the time that an LU travels from the MN to the CN, and the time that the CN processes the LU. For soft handoff, there is no interruption time if the MN stays in the overlap area of the old network and the new network.

$$T_{interrupt-hard} = (RTT_{MN-CN}/2) + T_{CN} \tag{25}$$

$$T_{interrupt-soft} = 0 \tag{26}$$

(Assuming that the path is symmetric.)

6.3.2. With Signalling Packet Loss.

There are many different control signals used for MIPv6 and ILNPv6. This analysis considers a scenario when each control signal gets lost. So, for MIPv6 without RO, BU or BAck could be lost. When RO is enabled, additional signalling packets including HoTI, HoT, CoTI, CoT, BU to CN and BAck from CN could be lost. For ILNPv6, LU or LU-ACK could be lost. The following is a detailed analysis of what would happen if those packets get lost.

(a). *MIPv6 without RO.*

1. BU is lost – as shown in Figure 6.6, if a BU is lost, a retransmission of BU is triggered when the MN does not receive a BAck after sending the BU within the time Δ_M. Hence, the handoff delay would increase by Δ_M, as would the interruption time.

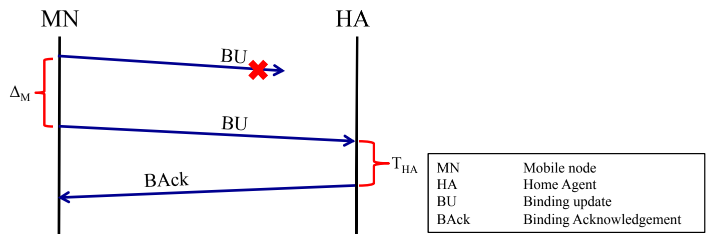


FIGURE 6.6. Signalling packets used in MIPv6 without RO in the case that a BU is lost

$$Delay = RTT_{MN-HA} + T_{HA} + \Delta_M \tag{27}$$

$$T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA} + \Delta_M \tag{28}$$

2. BAck is lost – similar to the case when BU is lost, another BU is retransmitted when the MN does not receive BAck after time Δ_M, as shown in Figure 6.7. The handoff delay, again, increases by Δ_M. However, the interruption time does not increase from the normal scenario because the HA has already learned the updated CoA of the MN from the first BU.

$$Delay = RTT_{MN-HA} + T_{HA} + \Delta_M \tag{29}$$

$$T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA} \tag{30}$$

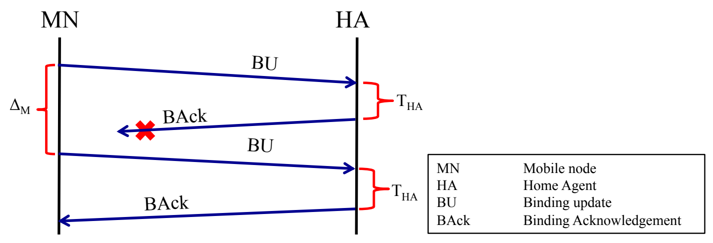


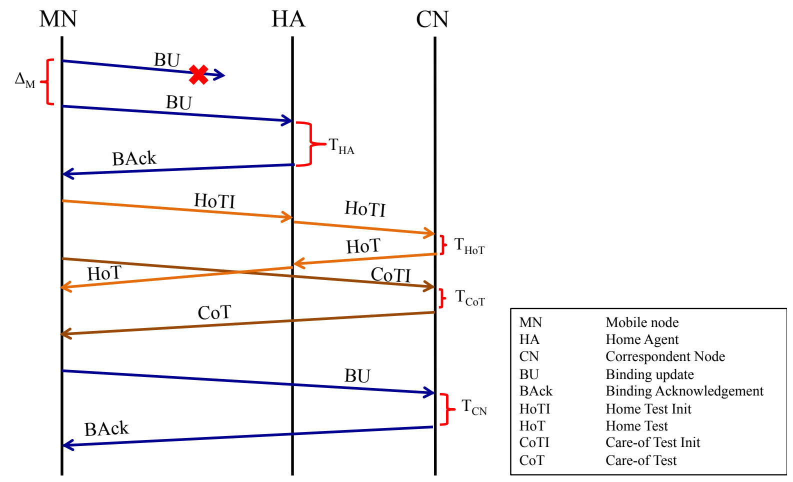
FIGURE 6.7. Signalling packets used in MIPv6 without RO in the case that a BAck is lost

(b). *MIPv6 with RO.*

1. BU is lost – similar to MIPv6 without RO, the BU is retransmitted after the time Δ_M, as shown in Figure 6.8. The handoff delay and interruption time, likewise, increase by Δ_M.

$$\begin{aligned} Delay = {} & RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + \Delta_M \\ & + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})] \end{aligned} \tag{31}$$

$$T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA} + \Delta_M \tag{32}$$



MN	Mobile node
HA	Home Agent
CN	Correspondent Node
BU	Binding update
BAck	Binding Acknowledgement
HoTI	Home Test Init
HoT	Home Test
CoTI	Care-of Test Init
CoT	Care-of Test

FIGURE 6.8. Signalling packets used in MIPv6 with RO in the case that a BU is lost

2. BAck is lost – again, this is similar to MIPv6 without RO, see Figure 6.9. Therefore, handoff delay and interruption time are calculated in the same way.

$$\begin{aligned} Delay = {} & RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + \Delta_M \\ & + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})] \end{aligned} \tag{33}$$

$$T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA} \tag{34}$$

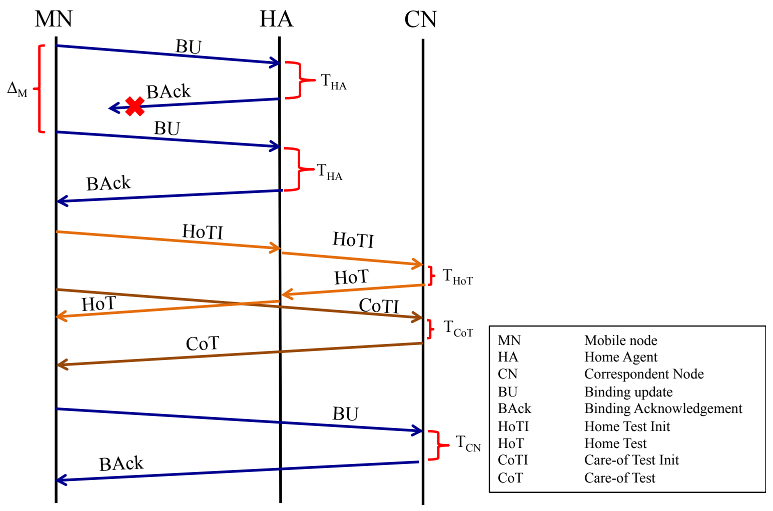


FIGURE 6.9. Signalling packets used in MIPv6 with RO in the case that a BAck is lost

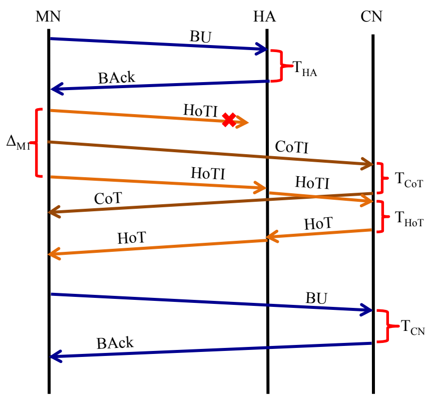
3. One of the RO signals is lost – Figure 6.10 shows scenarios where each of the RO packets, HoTI, HoT, CoTI, CoT, BU to CN and BAck from CN, is lost. There is another timer, Δ_{M1}, which would trigger retransmission of any lost RO packets. Handoff delay would increase by Δ_{M1}. However, Δ_{M1} could overlap with T_{HoT} or T_{CoT}, so the handoff delay presented here is the possible *maximum* value. Loss of RO packets does not affect the interruption time because update to the HA is complete and traffic can travel through the HA.

$$\begin{aligned}(35)\quad Delay = {} & RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + \Delta_{M1} \\ & + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})]\end{aligned}$$

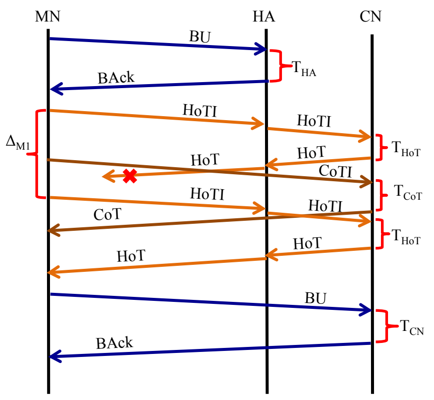
$$(36)\quad T_{interrupt} = (RTT_{MN-HA}/2) + T_{HA}$$

(c). *ILNPv6.*

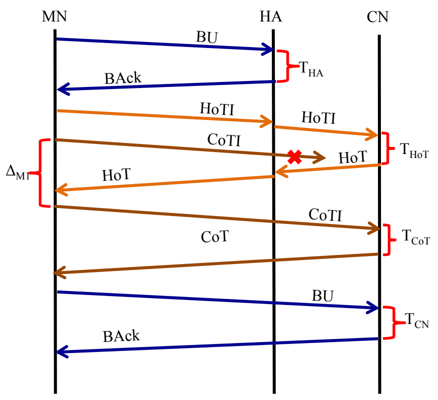
1. LU is lost – ILNPv6 also has similar behaviour as MIPv6 without RO. The LU would be retransmitted if the MN does not receive LU-ACK within the time Δ_I



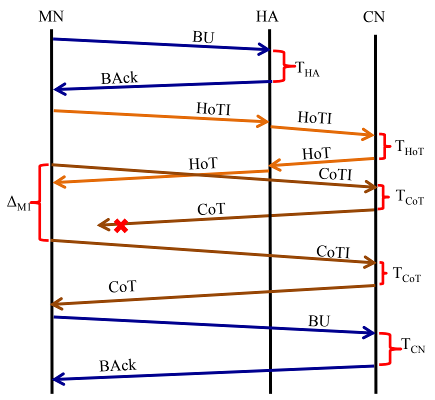
(A) HoTI is lost.



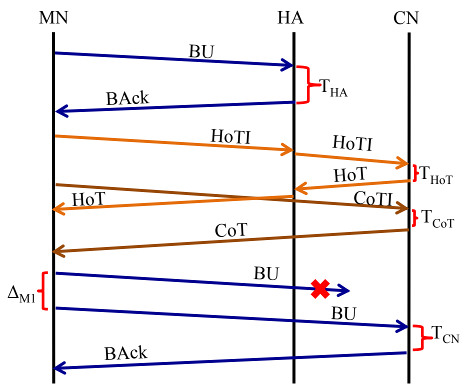
(B) HoT is lost.



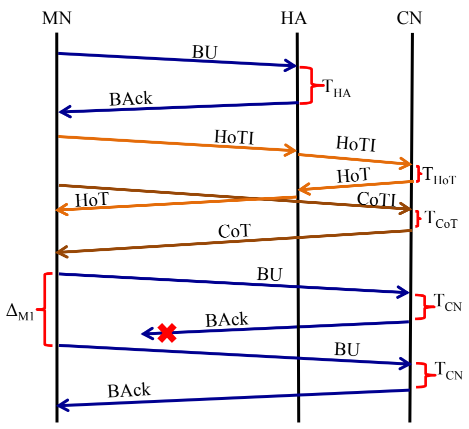
(C) CoTI is lost.



(D) CoT is lost.



(E) BU to CN is lost.



(F) BAck from CN is lost.

MN	Mobile Node	BU	Binding Update	HoT	Home Test
HA	Home Agent	BAck	Binding Acknowledgement	CoTI	Care-of Test Init
CN	Correspondent Node	HoTI	Home Test Init	CoT	Care-of Test

FIGURE 6.10. Signalling packets used in MIPv6 with RO, in the case that one of RO packets is lost.

(as in Figure 6.11). The handoff delay and the interruption time also increase by this amount of time (Δ_I).

$$Delay = RTT_{MN-CN} + T_{CN} + \Delta_I \tag{37}$$

$$T_{interrupt} = (RTT_{MN-CN}/2) + T_{CN} + \Delta_I \tag{38}$$

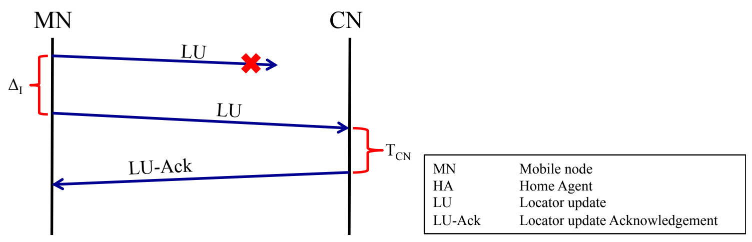


FIGURE 6.11. Signalling packets used in ILNPv6 in the case that an LU is lost

2. LU-ACK is lost – again, similar behaviour is observed. As in Figure 6.12, retransmission of LU happens after the time Δ_I. This affects the handoff delay but not the interruption time, since the CN is already informed about the new *L64* value.

$$Delay = RTT_{MN-CN} + T_{CN} + \Delta_I \tag{39}$$

$$T_{interrupt} = (RTT_{MN-CN}/2) + T_{CN} \tag{40}$$

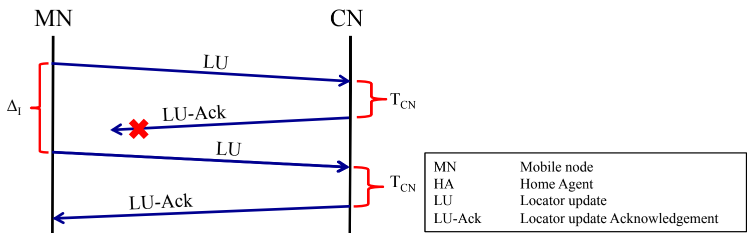


FIGURE 6.12. Signalling packets used in ILNPv6 in the case that an LU-ACK is lost

Table 6.3. Summary of expressions for handoff Delay

Case	MIPv6 no RO	MIPv6 with RO	ILNP hard	ILNP soft
No loss	$RTT_{MN-HA} + T_{HA}$	$RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})]$	$RTT_{MN-CN} + T_{CN}$	$RTT_{MN-CN} + T_{CN}$
LU or BU loss	$RTT_{MN-HA} + T_{HA} + \Delta_M$	$RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + \Delta_M + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})]$	$RTT_{MN-CN} + T_{CN} + \Delta_I$	$RTT_{MN-CN} + T_{CN} + \Delta_I$
LU-Ack or BAck loss	$RTT_{MN-HA} + T_{HA} + \Delta_M$	$RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + \Delta_M + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})]$	$RTT_{MN-CN} + T_{CN} + \Delta_I$	$RTT_{MN-CN} + T_{CN} + \Delta_I$
Any of RO^+ packets loss	N/A	$RTT_{MN-HA} + T_{HA} + RTT_{MN-CN} + T_{CN} + \Delta_{M1} + max[(RTT_{MN-HA} + RTT_{HA-CN} + T_{HoT}), (RTT_{MN-CN} + T_{CoT})]$	N/A	N/A

$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

Table 6.4. Summary of expressions for interruption time

Case	MIPv6 no RO	MIPv6 with RO	ILNP hard	ILNP soft
No loss	$(RTT_{MN-HA}/2) + T_{HA}$	$(RTT_{MN-HA}/2) + T_{HA}$	$(RTT_{MN-CN}/2) + T_{CN}$	0^*
LU or BU loss	$(RTT_{MN-HA}/2) + T_{HA} + \Delta_M$	$(RTT_{MN-HA}/2) + T_{HA} + \Delta_M$	$(RTT_{MN-CN}/2) + T_{CN} + \Delta_I$	0^*
LU-Ack or BAck loss	$(RTT_{MN-HA}/2) + T_{HA}$	$(RTT_{MN-HA}/2) + T_{HA}$	$(RTT_{MN-CN}/2) + T_{CN}$	0^*
Any of RO^+ packets loss	N/A	$(RTT_{MN-HA}/2) + T_{HA}$	N/A	N/A

*If MN is in overlap area.
$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

6.3.3. Results.

Calculation of handoff delay and interruption time in each scenario are summarised in Table 6.3 and Table 6.4 respectively. The value of variables in the equation could be obtained, partly from the experiment done in Chapter 4 and 5, and from the MIPv6 specification (RFC6275 (PS) [92]).

From the experiments:

- $T_{HA} \approx 1$ second
- $T_{CN} \approx T_{HoT} \approx T_{CoT} \approx 2$ ms

From RFC6275 (PS), Section 11.8:

- $\Delta_M = 1.5$ second, and exponentially increase after a failure
- $\Delta_{M1} = 1$ second, and exponentially increase after a failure

For Δ_I, the value is undefined, but could be any values greater than RTT_{MN-CN}. This kernel implementation uses $\Delta_I = 1$ second.

There are four network scenarios, similar to the experiment in Chapter 4 and 5: LAN to LAN handoff, LAN to WAN handoff, WAN to LAN handoff and WAN to WAN handoff. According to the results in Chapter 4 and 5, the values of RTT are listed below.

LAN to LAN handoff

- $RTT_{MN-HA} \approx RTT_{MN-CN} \approx RTT_{HA-CN} \approx 5$ ms

LAN to WAN handoff

- $RTT_{MN-HA} \approx RTT_{MN-CN} \approx 200$ ms
- $RTT_{HA-CN} \approx 5$ ms

WAN to LAN handoff

- $RTT_{MN-HA} \approx RTT_{HA-CN} \approx 200$ ms
- $RTT_{MN-CN} \approx 5$ ms

WAN to WAN handoff

- $RTT_{MN-HA} \approx 400$ ms
- $RTT_{MN-CN} \approx RTT_{HA-CN} \approx 200$ ms

By replacing those numbers in the equations in Table 6.3 and Table 6.4, the handoff delay and interruption time of each scenario using MIPv6, with and without RO, and ILNPv6 is shown from Table 6.5 to Table 6.12.

TABLE 6.5. Handoff Delay (ms) LAN to LAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1005	1024	7	7
LU or BU loss	2505	2524	1007	1007
LU-Ack or BAck loss	2505	2524	1007	1007
Any of RO^+ packets loss	N/A	2024	N/A	N/A

$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

TABLE 6.6. Interruption time (ms) LAN to LAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1002.5	1002.5	4.5	0*
LU or BU loss	2502.5	2502.5	1004.5	0*
LU-Ack or BAck loss	1002.5	1002.5	4.5	0*
Any of RO^+ packets loss	N/A	1002.5	N/A	N/A

*If MN is in overlap area.
$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

TABLE 6.7. Handoff Delay (ms) for LAN to WAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1200	1609	202	202
LU or BU loss	2700	3109	1202	1202
LU-Ack or BAck loss	2700	3109	1202	1202
Any of RO^+ packets loss	N/A	2609	N/A	N/A

$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

TABLE 6.8. Interruption time (ms) for LAN to WAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1100	1100	102	0*
LU or BU loss	2600	2600	1102	0*
LU-Ack or BAck loss	1100	1100	102	0*
Any of RO^+ packets loss	N/A	1100	N/A	N/A

*If MN is in overlap area.
$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

For the no-loss scenario, in the LAN environment, there is not much difference between MIPv6 with and without RO: the handoff delay and the interruption time

Table 6.9. Handoff Delay (ms) for WAN to LAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1200	1609	7	7
LU or BU loss	2700	3109	1007	1007
LU-Ack or BAck loss	2700	3109	1007	1007
Any of RO^+ packets loss	N/A	2609	N/A	N/A

$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

Table 6.10. Interruption time (ms) for WAN to LAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1100	1100	4.5	0*
LU or BU loss	2600	2600	1004.5	0*
LU-Ack or BAck loss	1100	1100	4.5	0*
Any of RO^+ packets loss	N/A	1100	N/A	N/A

*If MN is in overlap area.
$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

Table 6.11. Handoff Delay (ms) for WAN to WAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1400	2204	202	202
LU or BU loss	2900	3704	1202	1202
LU-Ack or BAck loss	2900	3704	1202	1202
Any of RO^+ packets loss	N/A	3204	N/A	N/A

$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

Table 6.12. Interruption time (ms) for WAN to WAN handoff scenario

Case	MIP no RO	MIP with RO	ILNP hard	ILNP soft
No loss	1200	1200	102	0*
LU or BU loss	2700	2700	1102	0*
LU-Ack or BAck loss	1200	1200	102	0*
Any of RO^+ packets loss	N/A	1200	N/A	N/A

*If MN is in overlap area.
$^+$RO packets include HoTI/HoT, CoTI/CoT and BU/BAck to CN.

are around 1 second. However, when the WAN network is involved, the handoff delay of MIPv6 with RO is significantly higher than MIPv6 without RO. The interruption, however, remains the same. Meanwhile, when handing off to LAN, the handoff delay in ILNP is just 7 ms and the interruption time is only 4.5 ms (for hard handoff), and could be as low as 0 for soft handoff. The number rises to around 200 ms delay and 100 ms interruption time (still 0 for soft handoff) for an MN handing off to the WAN network. The calculated handoff delays are in accordance with the experimental results in Chapter 4 and 5. Of course, for other scenarios the numeric values would be different, but the same equations could be used.

For every scenario, when a BU to HA or BAck from HA is lost, the handoff delay for MIPv6, both with and without RO, increases by 1.5 second. While handoff delay for ILNPv6 increases by only 1 second if LU or LU-ACK is lost. An increase in interruption time is also similar when BU or LU is lost. However, interruption time does not increase if BAck or LU-ACK is lost because the HA or the CN has been notified about the update already.

For MIPv6 with RO, when any of those RO packets is lost, the handoff delay increases by 1 second. Again, this does not affect the interruption time.

6.4. Scalability

This section provides a set of analyses that present the impact of using MIPv6 (with and without RO) and ILNPv6 for a large number of nodes. The analyses shows how large is the signalling overhead, in terms of number of packets and number of bytes, introduced into the network when the number of MNs and CNs increases for a *single* handoff. The number of packets is considered because it affects the processing and forwarding decision at the routers as well as the packet processing at the end systems. The number of bytes is considered due to the impact to the network capacity and processing power at the end systems.

6.4.1. Overhead Calculation.

(a). *MIPv6 without RO.*

From Figure 6.1 in the beginning of this chapter, MIPv6 without RO has only BU to the HA and BAck from the HA as overhead.

$$Overhead = BU_{HA} + BAck_{HA} \tag{41}$$

When the number of MN increases, that overhead is multiplied. However, an increase in the number of CNs does not affect the overhead since there is no control signals that go to or from the CN.

$$Overhead = MN \cdot (BU_{HA} + BAck_{HA}) \tag{42}$$

Therefore the *number of packets* as overhead can be calculated by:

$$Overhead_{packet} = 2 \cdot MN \tag{43}$$

The packet size of BU and BAck is obtained from the *tcpdump* log in the experiments in Chapter 4 and 5: $BU_{HA} = 110$ bytes, $BAck_{HA} = 94$ bytes. So, the *number of bytes* as overhead is calculated by:

$$Overhead_{bytes} = MN \cdot (110 + 94) = 204 \cdot MN \tag{44}$$

(b). *MIPv6 with RO.*
From Figure 6.2 in the beginning of this chapter, when RO is used, MIPv6 has additional overhead signals as well as BU to the HA and BAck from the HA.

$$\begin{aligned} Overhead = BU_{HA} + BAck_{HA} + HoTI + HoT + \\ CoTI + CoT + BU_{CN} + BAck_{CN} \end{aligned} \tag{45}$$

Like MIPv6 without RO, that overhead is multiplied when the number of MNs increases. An increase in the number of CNs also affects the overhead signals for the RO process (i.e. HoTI, HoT, CoTI, CoT, BU to CN, and BAck from CN).

$$\begin{aligned} Overhead = MN \cdot [BU_{HA} + BAck_{HA} + \\ CN \cdot (HoTI + HoT + CoTI + CoT + BU_{CN} + BAck_{CN})] \end{aligned} \tag{46}$$

So, the *number of packets* as overhead can be calculated by:

$$Overhead_{packet} = MN \cdot (2 + 6 \cdot CN) \tag{47}$$

Again, from the *tcpdump* log in the experiments in Chapter 4 and 5, the packet size of all signalling packets is obtained: $BU_{HA}, HoTI, BU_{CN}, BAck_{CN} = 110$ bytes, $BAck_{HA} = 94$ bytes, $HoT = 118$ bytes, $CoTI = 70$ bytes, $CoT = 78$ bytes. So, the *number of bytes* as overhead is calculated by:

$$Overhead_{bytes} = MN \cdot (204 + 596 \cdot CN) \tag{48}$$

(c). *ILNPv6.*
The overhead signals used by ILNPv6 are only LU and LU-ACK, as shown in Figure 6.3 in the beginning of this chapter.

$$Overhead = LU + LU\text{-}ACK \tag{49}$$

Both increases in the number of MNs and the number of CNs affect the signalling overhead because both are involved in exchanging the handoff signals.

$$Overhead = MN \cdot CN \cdot (LU + LU\text{-}ACK) \tag{50}$$

So, the *number of packets* as overhead is calculated by:

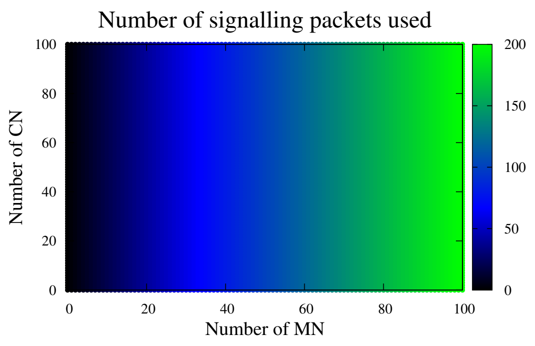
$$Overhead_{packet} = 2 \cdot MN \cdot CN \tag{51}$$

The sizes of LU and LU-ACK packets, again, are retrieved from the *tcpdump* log in the experiments in Chapter 4 and 5: $LU = LU\text{-}ACK = 86$ bytes, so:

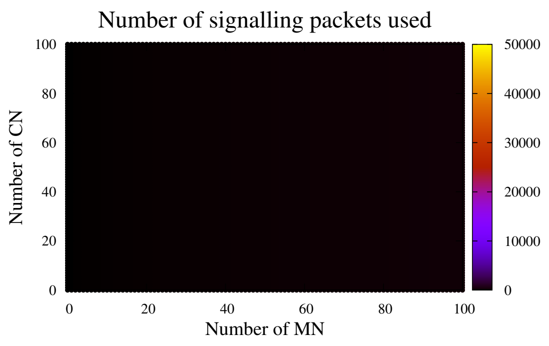
$$Overhead_{bytes} = 172 \cdot MN \cdot CN \tag{52}$$

6.4.2. Results.

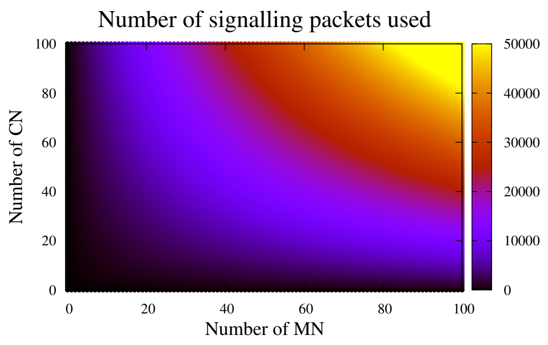
In this analysis, the considered number of MN and the number of CN are from 1 to 100. By replacing the numbers in the equations listed above, the number of signalling packets and the overhead in bytes are calculated.



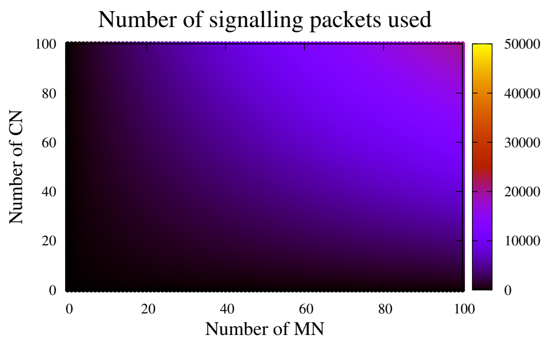
(A) MIPv6 without RO (detailed scale).



(B) MIPv6 without RO.



(C) MIPv6 with RO.

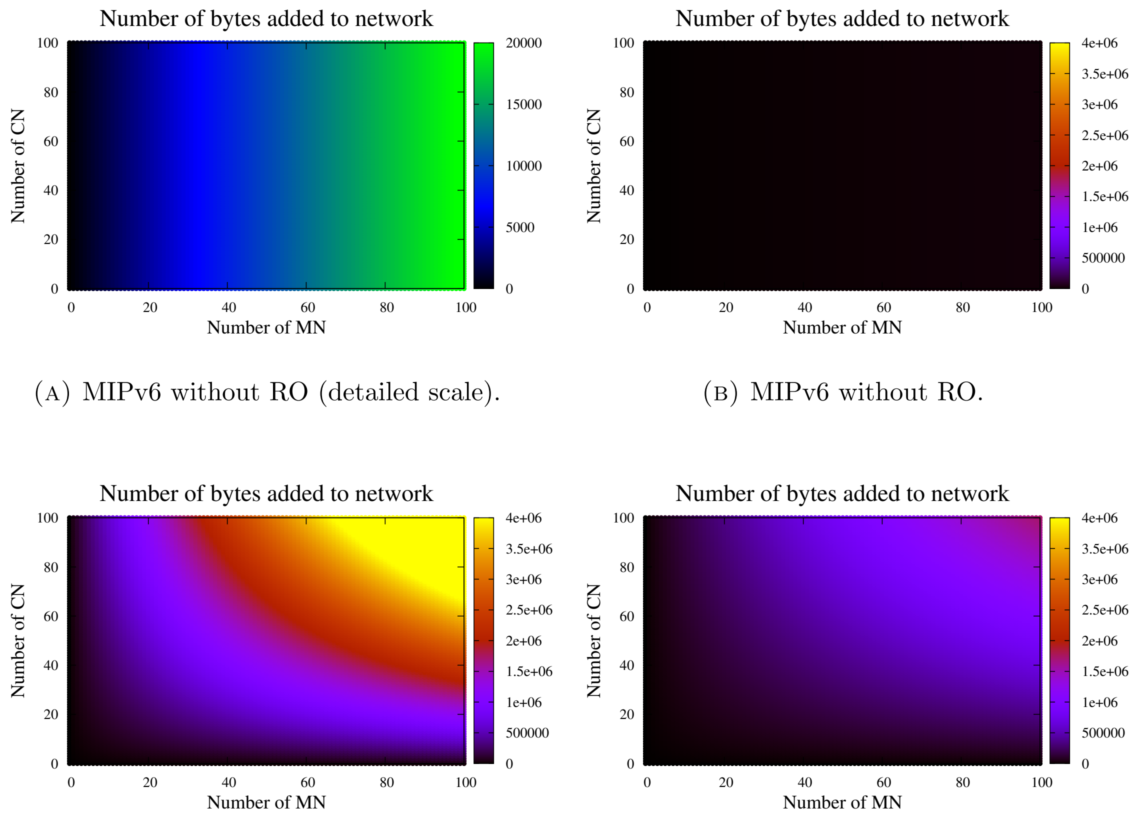


(D) ILNPv6.

FIGURE 6.13. Overhead packets. ILNPv6 has a much lower number of overhead packets compared to MIPv6 with RO. MIPv6 without RO yeilds the smallest packet overhead, and increases only when the number of MNs increases.

(a). *Number of overhead packets.*

Figure 6.13 shows the overhead signalling packets used for MIPv6, with and without RO, and ILNPv6. MIPv6 without RO has a very low overhead compared to the other two because an increase in the number of CN does not affect the overhead, only the number of MN does, as shown in Figure 6.13a. Only around 200 overhead packets are introduced into the network for 100 MNs. Nevertheless, ILNPv6 still has relatively low packet overhead compared to MIPv6 with RO. For 100 MNs each of which has 100 associated CNs, ILNPv6 produces around 20,000 extra packets, which is less than half of overhead packets of MIPv6 with RO (50,000 packets).



(A) MIPv6 without RO (detailed scale). (B) MIPv6 without RO.

(C) MIPv6 with RO. (D) ILNPv6.

FIGURE 6.14. Overhead bytes. ILNPv6 has a much lower number of overhead in bytes compared to MIPv6 with RO. MIPv6 without RO yeilds the smallest byte overhead, and increases only when the number of MNs increases.

(b). *Number of overhead bytes.*

Figure 6.14 shows the number of overhead bytes used for MIPv6, with and without RO, and ILNPv6. MIPv6 without RO, again, has a very low overhead compared to the other two because, as shown in Figure 6.14a, it is impacted by only an increase in the number of MN, and not the CN. With 100 MNs, only around 20 Kbytes of overhead are generated. Similar to the packet overhead, ILNPv6 has low byte overhead compared to MIPv6 with RO. For 100 MNs (each of which has 100 CNs), ILNPv6 generates around 1.5 Mbytes of overhead into the network compared to 4 Mbytes overhead for MIPv6 with RO.

6.4.3. Discussion.

As mentioned earlier, these results are based on a *single* handoff of MNs. Of course, the overhead would increase if MNs handoff more frequently. However, from the results, it can be seen that ILNPv6 should be more scalable than MIPv6 with RO.

Also, this analysis does not include overhead from the DNS update for ILNPv6 because it is not always required (see Section 6.1). However, even when the DNS update is required, the overhead should still remain low since only the number of MNs affects the number of signal packets/bytes – like MIPv6 without RO, the number of CNs is not relevant for DNS updates.

Another important factor that affects overhead and scalability is caching. For MIPv6 with RO, after the RO process, the CN would have a cache to hold the CoA value of the MN. If the entry expires, the CN would send a *Binding Refresh Request* message [92, Sec. 9.5.5] to the MN. The MN would perform the return routability procedure as well as BU/BAck handshake with the CN, if it still uses such CoA. This allows subsequent connections to be established directly between the MN and the CN without passing through the HA. However, this also causes extra overhead to the network. With ILNPv6, on the other hand, if the L64 value at the CN expires, a new connection session can be established using DNS, without requiring extra overhead apart from the DNS query. More information about the role of DNS for ILNPv6 is discussed in Section 7.2.1.

6.5. Summary

This chapter has provided analyses of the control plane of MIPv6 and ILNPv6. Overall, ILNPv6 control signalling is more efficient than MIPv6 with RO in terms of loss tolerance, overhead and scalability. MIPv6 without RO control signalling is about the same level as ILNPv6 in terms of loss tolerance, and better than ILNPv6 in terms of overhead. However, with the known problem in handoff performance (as observed in Chapter 5), MIPv6 without RO is unlikely to be used in a real world scenario. The three questions stated in Section 6.1 are answered as follows.

(1) Lossy networks reduce the handoff success rate of both MIPv6 (with and without RO) and ILNPv6. MIPv6 with RO is the least tolerant to lossy networks, while MIPv6 without RO and ILNPv6 have a relatively similar level of tolerance (ILNPv6 is slightly better).

(2) A loss of handoff signalling packets sent from an MN (i.e. BU for MIPv6 and LU for ILNPv6) causes the handoff duration and the interruption time to be longer for both MIPv6 (with and without RO) and ILNPv6, while a loss of signalling packets sent from a CN (i.e. BAck and LU-ACK) affects only the handoff duration. ILNPv6 has a smaller increase in the handoff duration and the interruption time compared to MIPv6.

(3) For a high number of MNs and CNs, more overhead (packets and bytes) is introduced. ILNPv6 generates less than a half the overhead in terms of number of packets and number of bytes compared to MIPv6 with RO. MIPv6 without RO has a very small overhead, but has handoff performance issues (as discussed above).

Chapter 7

Conclusion and Future Works

7.1. Conclusion

Host mobility is highly desired for Internet users (see Section 1.2), but there is still no effective solution that is widely deployed. IETF Mobile IP is widely recognised as having severe performance issues. In this work, the main contribution is to examine the use of ILNP to enable purely end-to-end host mobility support. ILNP has a firm architectural concept, but there was no proof that it can be engineered and provide host mobility support, with satisfactory performance, in a real network with existing OSs and APIs.

The thesis statement, as in Section 1.5 was:

IP layer mobility can be supported as 'first class functionality' by using end hosts only, without changing current network infrastructure

This thesis has shown that the statement is true: it is possible to enable mobility in IP layer as *first class functionality* using ILNP. In regard to the three research questions stated in Section 1.5, this work has fulfilled the gap between ILNP architecture and engineering. It shows that:

(1) ILNP mobility can be built to an end-to-end system's network stack using a dual-stack approach by extending the existing code base. Additionally,

ILNP can be engineered to use the existing sockets API and support existing applications.

(2) ILNP can provide mobility support to UDP and TCP applications, with better performance than MIPv6.

(3) ILNP control signal is smaller, more robust to lossy environment, and more scalable comparing to MIPv6.

Chapter 3 deals with an *implementation* issue. It confirms that ILNP mobility can be built in a system network stack (Linux, in this case) by enhancing the current IPv6 code. A basic evaluation at the end of the chapter shows that ILNP mobility of the implemented prototype could work to its architectural design.

Issues of *performance* are covered by Chapter 4 and 5. A real legacy (non-ILNPv6) application, *iperf*, was used for performance evaluation of ILNPv6 over a real wireless network infrastructure. These two chapters show that ILNPv6, especially with soft handoff, provides a seamless handoff in the network layer, with minimum packet loss and minimum interruption of the flows, for both UDP and TCP, in both LAN and WAN environments. In comparison to MIPv6, ILNPv6 outperforms it in every case in terms of flow continuity, packet loss and handoff delay.

Chapter 6 analyses the *control plane* of ILNPv6 and MIPv6. There are three findings. First, ILNPv6 is more robust than MIPv6 in lossy environments. A handoff is more likely to succeed. Second, in the case of a signalling packet loss, there is less impact to handoff performance for ILNPv6 than MIPv6. MIPv6 suffers from longer handoff delay and longer interruption time than ILNPv6. Third, at scale, ILNPv6 generate less overhead, in terms of number of packets and number of bytes, into the network than MIPv6 (when RO is enabled).

To sum up, this work has shown that handoff management for IP mobile nodes can be implemented as a purely end-to-end function in the Linux OS kernel. ILNPv6 was implemented as a superset of IPv6, using a radically different naming architecture – based on identifiers and locators. Nevertheless, it is still possible to operate over existing IPv6 infrastructure (the testbed used IPv6 only routers). ILNPv6 could be integrated into existing OS bases and deployed incrementally. ILNPv6's end-to-end architecture means that additional entities, such as home agents, are not required, neither are tunnels. The performance evaluation showed that ILNPv6, especially with soft handoff, provides excellent handoff performance in terms of flow continuity, throughput, packet loss and handoff delay with respect to UDP and TCP flows. ILNPv6 handoff signalling is robust to lossy environments, and generates minimum overhead to the network.

7.2. Discussion

ILNP is a huge area of study, and this work only investigates the ILNP handoff performace to enable mobility support over the Internet. There are still many concerns before allowing ILNP to be globally deployed. This section presents some discussion and critical analyses relating to those concerns; comparisons with Mobile IP are made where appropriate.

7.2.1. Name Resolution Using DNS.

Since this work focusses on investigating handoff performance of ILNPv6, `/etc/hosts` is used for name resolution. In a real deployment, DNS would be used instead. New DNS Resource Records for NID and L64 values have been defined [18] and are implemented in widely-used DNS software: ISC BIND/*named* from v9.9.3, NSD/*Unbound* from v3.2.15 and *Knot DNS* from v1.3.0.

As discussed in Section 2.6.3, since the L64 value of the MN would be updated in the DNS when the node moves, DNS records that hold L64 values need to have a very low DNS TTL, or else cached L64 values would become stale. A previous emulation study [60] reports that using a low TTL value of hundreds of seconds should not impact significantly on DNS. Also, a previous empirical evaluation [24] shows that values of TTL as low as zero for IPv4 *A* records have no significant impact on DNS load, so use of low DNS TTL for the L64 records for mobile ILNP hosts would have little impact on DNS load. BIND, Unbound and Knot DNS also support DNS security for secure dynamic DNS updates of L64 resource records. DNS security is being implemented independently of ILNP and is widely deployed also. As DNS is already deployed worldwide, ILNP does not incur any additional overhead, e.g. managing proxies and tunnels as for MIPv6 and its extensions.

For Mobile IP, DNS is also used for name resolution, but the DNS lookup always resolves to the HoA at the home network of the MN.

7.2.2. Simultaneous Mobility.

Although this thesis mainly focusses on one-side mobile host, ILNPv6 could also be used for supporting two communicating MNs. When two mobile hosts communicate, it is possible that both hosts handoff simultaneously (sometimes called the *Double Jump* problem). For MIPv6, both hosts could still communicate via HAs of both sides since the HAs never move. However, there are problems in the RO process because the return routability procedure and the binding update to another end point might never reach the destination [38]. So, the RO process fails, and MNs must communicate through the HA which affects routing performance. Some solutions have been proposed, e.g. [69].

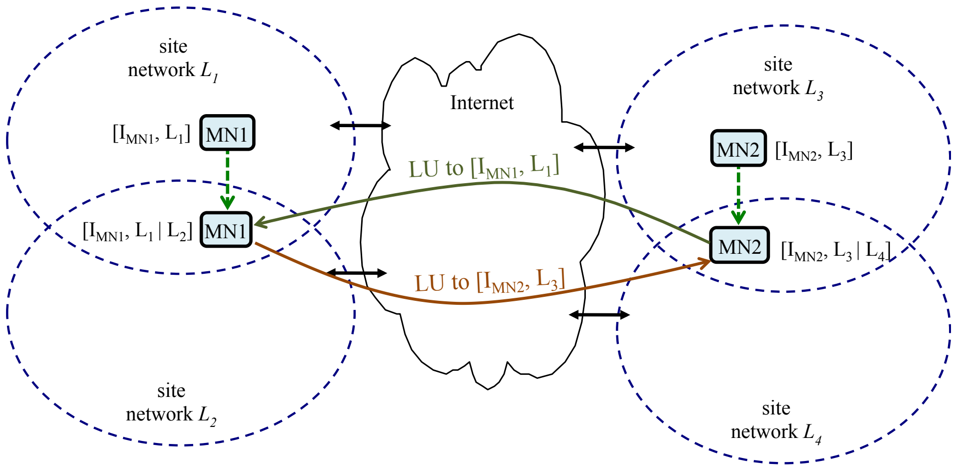


FIGURE 7.1. Simultaneous mobility using ILNP soft handoff. The LU signal can reach both end hosts because they maintain bindings to the previous L64 value.

For ILNP, simultaneous mobility can be handled as depicted in Figure 7.1. ILNP can leverage the soft handoff mechanism to overcome this problem. Although both end hosts perform handoff simultaneously, they both still maintain their old L64 values, and so the LU handshake can be completed even though an LU is sent to the previous L64. However, if hard handoff is used or when soft handoff is not possible, say, no overlap area between networks, the LU could be sent to the previous locations and never reach the MNs, and the handoff could fail. Nevertheless, the communication sessions can be re-established by consulting the DNS after the L64 records have been updated.

7.2.3. Applicablility to Vertical Handoff.

Vertical handoff is a scenario when an MN moves across different wireless technologies (e.g. WiFi, 3G, LTE and satellite) or administrative boundaries. To enable seamless vertical handoff, there are three key points: enabling architecture; decision metrics and policy design; and radio link transfer design [71]. Based on the results of this work, ILNP is an example of an architecture that could enable seamless mobility in the IP layer i.e. enabling architecture. The other two key points are out of scope for ILNP, as well as Mobile IP, as they are really concerned with the signals, triggers and characteristics of the lower-level network technologies.

The decision metric determines if the MN needs to handoff and which network should be chosen. The metric may derive from different factors such as service type, network conditions and user preferences. After the destination network is chosen, the handoff policy is used to determine when the handoff occurs. The decision algorithm could be *function based* considerring signal strength, capacity and cost of the overlapping networks. It could also be *user centric decision* algorithm, which allow a user to specify his/her preference [123].

The radio link transfer concerns with performance of link switching (e.g. minimising delay) and retaining QoS of the on-going flows – since different wireless technologies provide different network conditions e.g. different bandwidth. More details on QoS are provided in Section 7.2.4.

In real-world scenarios, (vertical) handoff also involves the physical and link layers. This work does not consider the interactions between the IP layer handoff using ILNP and any lower layer handoff mechanisms. Investigations of using ILNP in various mobile devices to observe effects of specific physical and link layer handoff may be required before deploying ILNP. However, as ILNP enables efficient network layer handoff, the network layer is no longer a bottleneck in the mobility process.

7.2.4. Managing changes in QoS on Handoff.

While ILNPv6 deals with handoff, and network layer soft handoff can help to reduce packet loss, there are still other problems to solve at the transport layer or application layer, for example ILNP does not guarantee the *Quality of Experience (QoE)* of applications. As mentioned above, moving between cells could result in changes in the end-to-end path QoS, e.g. changes in end-to-end throughput, loss and delay. These would have to be dealt with by high-layer protocols or middleware as they are today. Chapter 5 already presents an example of the impact of such handoff to TCP, e.g. data rates drop by an order of magnitude when moving from a low delay network to a high delay network. Therefore, the transport layer may need to have adaptive congestion control mechanisms that are responsive to changes in end-to-end path characteristics. Applications may need to adapt, and additional signalling and buffering may also be required to maintain or adapt flows in order to provide suitable QoE. Optimisation of ILNPv6 engineering under different types of applications and different network scenarios is a subject for further study.

7.2.5. Backwards Compatibility with IPv6.

ILNPv6 is an end-host enhancement of IPv6, so it could be deployed in the current IPv6 backbone without requiring any changes or upgrades to IPv6 routers. The performance evaluation conducted in Chapter 4 and 5 shows that ILNPv6 works well over unmodified IPv6 network infrastructure. To enable backwards compatibility with IPv6, i.e. when an ILNPv6 host communicates with an IPv6 host, an *ILNPv6 Nonce Option* [15] is leveraged. A receiver of a packet containing the ILNPv6 Nonce Option would know ILNPv6 is supported by the sender if it was an ILNPv6 enabled host itself. If the receiver did not support ILNPv6, normal IPv6 behaviour would result in an ICMP packet (i.e. 'Parameter Problem') being returned to the sender, causing the initiator to fallback to using IPv6 [13]. So, it is possible for an ILNPv6-capable host to communicate with IPv6 hosts without requiring modification of the IPv6 host.

One of the limitations of ILNPv6 is for those sites where DHCPv6 [41] is used. Since the NID of a MN has to remain constant when it moves, a new IPv6 address (128 bits) arbitrarily allocated by DHCPv6 at the new site cannot be used. The use of DHCPv6 and ILNPv6 requires further exploration: as the notion of an address changes with ILNP (as with other Identifier/Locator mechanisms), the notion of address management also must be revisited. Currently, ILNP assumes the use of IPv6 prefixes as Locator values from IPv6 RAs from IPv6 Neighbour Discovery (ND) [83], and the use of the Stateless Address Autoconfiguration (SLAAC) [115] mechanism to assign Identifier values. Note that both RAs/ND and SLAAC are mandatory parts of the core IPv6 draft standard, whilst DHCPv6 is not mandatory to support IPv6 deployment. In MIPv6, DHCPv6 can still be used to assign HoA/CoA to an MN. With specific DHCPv6 options [59], an MN can learn its HA's IP address and FQDN as well as its home network prefix.

7.2.6. Security Considerations.

This section provides a brief discussion of security considerations when using ILNP. Full security implications are a complex issue: a detailed discussion and analyses is left for further study.

As stated in Section 2.6.4, a *false binding* is considered as a main security issue of mobility management solutions. ILNPv6 prevents this attack from 'off-path' atackers by using the nonce destination option [15], and from 'on-path' attackers by using IPSec.

For MIPv6, protection is required for 1) BU from MN to HA (equivalent to a DNS update in ILNPv6), and 2) for BU from MN to CN (equivalent to LU from MN to CN in ILNPv6) [92]. The security of BU between MN and HA relies on IPSec. This is equivalent to the use of secure dynamic update for DNS in ILNPv6, for which a secret key of both ends has been exchanged in the beginning of the session and then used to encrypt the BU and DNS update packets. The security between MN and CN in MIPv6 is enabled by the *return routability* procedure. The primary purpose of the return routability is to ensure that both HoA and CoA of the MN can be reached by CN. In addition, during the procedure, the MN and CN could also exchange a token that would be used later on in the BU message. The use of a token is similar to the use of the nonce option to protect LU messages in ILNPv6. Again, the use of a token or nonce value can prevent forged BU/LU packets only from 'off-path' attackers. To protect the BU/LU from ‘on-path’ attackers, additional cryptographic mechanisms such as IPSec are required.

7.2.7. Privacy Considerations.

RFC 6973 (I) [37] summarises general privacy considerations for network protocols. For ILNP, the use of fixed NID value may violate identity privacy of a MN and cause a threat in *correlation*, where the activities of the MN can be tracked and combined over time [37, Sec. 5.2.1]. The NID value is essential as the end-to-end invariant for transport protocol session state, and might also be used by application protocols, though it is encouraged to use of an FQDN or application-specific namespace with ILNPv6. In theory, ILNP allows multiple NID values to be used by a node simultaneously, as long as any transport session uses the same NID value during its lifetime, to maintain end-to-end session state invariance. So, in support of identity privacy, NID values could be generated as required [13, Sec. 2 & 11] [16, Sec. 8]. For example, a client system could generate ‘random’ (ephemeral) NID values for use for different transport layer sessions (using IPv6 DAD to check for collisions in the NID/L64 that is created). This is an example of the *data minimisation* technique to mitigate the correlation problem [37, Sec. 6.1].

ILNPv6 can also leverage existing IPv6 privacy extensions for generating anonymous NID values as required [82]. A node can generate a new ephemeral NID value shortly before initiating communication. Normally, such requirements may be for client systems accessing services, and one use may be to prevent tracking of users through the default NID value that is derived as for IPv6. However, if an MN expects incoming connections, then the distribution of the new identity would be by application-specific means. If DNS was used, the TXT record [100] could be used, again in an application-specific manner, in conjunction with Secure DNS Dynamic Update.

The IPv6 privacy extensions are also applicable for MIPv6. However, MIPv6 also has another issue of privacy, where information of the HoA and CoA could be eavesdropped after RO (because the packets contain both values) [64]. A possible solution to protect this information is *Reverse Tunnelling* [94]. The MN and CN generate a privacy key based on the token exchanged during the return routability process. This key is used to encrypt the HoA in a payload packet instead of a plain HoA. However, this increases per-packet overhead since encryption and decryption of HoA must be processed for every packet sent between MN and CN.

7.2.8. Energy Usage.

Energy is an important aspect for mobile devices. Due to their limited battery life, energy efficient applications and protocols are preferable. For ILNP, the general operations are not different from 'classic IPv6', and thus should not have a huge difference in battery consumption. Of course, additional processing for the nonce option could cause slightly more energy consumption. However, energy usage for mobility using ILNPv6 should be more efficient than MIPv6. First, ILNPv6 soft handoff has a lower number of TCP retransmissions than MIPv6 (see Section 5.3.3), which means less computation power required and less energy wasted in transmissions. Second, ILNPv6 signalling overhead is less than MIPv6 with RO (see Section 6.4), so lower energy is used to create, send and process such signalling packets.

In addition, it should also be possible to leverage multipath effects from soft handoff in ILNPv6 to provide energy efficient path selection like work on MP-TCP [93]. As energy cost for each type of wireless interfaces (e.g. WiFi and 3G) varies for different kinds of load [75], choosing a path via a suitable interface could help the energy management. However, this is an item for future study.

7.3. Future Works

Although this thesis has shown that ILNPv6 could provide mobility support, there is certainly more work to be considered before a deployment to the Internet. The following are possible future works that could extend this work to improve host mobility capability using ILNPv6.

(1) Integration with DNS – as mentioned in Section 7.2.1, the current ILNPv6 prototype uses `/etc/host` for name resolution. So, extensions to the ILNPv6

prototype and performance evaluation of using DNS with ILNPv6 need to be explored.

(2) TCP optimisation – as discussed in Section 5.4, tuning of TCP to work better with ILNPv6 is a subject for further studies. These include an evaluation of ILNPv6 with different TCP variants.

(3) Impact on applications – it is necessary to investigate and tune ILNPv6 with various kinds of application (e.g. real-time video) to maximise users' QoE. This may concern adjustment to the handoff decision algorithm and leveraging of multi-path transfer.

(4) Multihomed mobile node – during soft handoff, an MN is multihomed, as the NID can be bound to more than one L64 simultaneously. Hence, it is possible that an MN connects to different wireless technology, say, both WiFi and 3G simultaneously. This could enable functions such as WiFi offloading (for example, using 3G for signalling and small data packets, and using WiFi for bandwidth consuming data), and path selection for energy efficiency (see Section 7.2.8).

(5) Security – a detailed study of security aspect of ILNPv6 is essential. This includes extending IPSec to work with ILNPv6 and exploring other possible security issues and solutions when using ILNPv6 in the Internet (see section 7.2.6).

(6) Privacy – to improve ILNPv6 privacy, further studies on using randomly generated NID as well as IPv6 privacy extensions with ILNP may be useful, regarding to Section 7.2.7 and with respect to RFC6973 (I) [37].

References

[1] A. Achour, B. Kervella, and G. Pujolle. SHIM6-based mobility management for multi-homed terminals in heterogeneous environment. In *Proc. WOCN 2011 - 8th Intl. Conf. Wireless and Optical Communications Networks*, pages 1–5, May 2011.

[2] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang. Unreeling NetFlix: Understanding and improving multi-CDN movie delivery. In *IEEE INFOCOM 2012*, pages 1620–1628, March 2012.

[3] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme. A Node Identity Internetworking Architecture. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6, April 2006.

[4] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. RFC 5681 (DS), IETF, Sep 2009.

[5] R. Atkinson and S. Bhatti. Site-Controlled Secure Multi-homing and Traffic Engineering for IP. In *Proc. IEEE MILCOM 2009*, Oct 2009.

[6] R. Atkinson, S. Bhatti, and S. Hailes. A Proposal for Unifying Mobility with Multi-Homing, NAT, and Security. In *MobiWAC'07 – 5th ACM Intl. Wkshp. on Mobility Mgmt. and W'less Access*, Oct 2007.

[7] R. Atkinson, S. Bhatti, and S. Hailes. ILNP: mobility, multi-homing, localised addressing and security through naming. *Telecommunication Systems*, 42:273–291, 2009.

[8] R. Atkinson, S. Bhatti, and S. Hailes. Evolving the Internet Architecture Through Naming. *Selected Areas in Communications, IEEE Journal on*, 28(8):1319 –1325, october 2010.

[9] R. Atkinson and S. N. Bhatti. Address Resolution Protocol (ARP) for the Identifier-Locator Network Protocol for IPv4 (ILNPv4). RFC 6747 (E), IRTF, Nov 2012.

[10] R. Atkinson and S. N. Bhatti. ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv4 (ILNPv4). RFC 6745 (E), IRTF, Nov 2012.

[11] R. Atkinson and S. N. Bhatti. ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6). RFC 6743 (E), IRTF, Nov 2012.

[12] R. Atkinson and S. N. Bhatti. Identifier-Locator Network Protocol (ILNP) Architectural Description. RFC 6740 (E), IRTF, Nov 2012.

[13] R. Atkinson and S. N. Bhatti. Identifier-Locator Network Protocol (ILNP) Engineering Considerations. RFC 6741 (E), IRTF, Nov 2012.

[14] R. Atkinson and S. N. Bhatti. IPv4 Options for the Identifier-Locator Network Protocol (ILNP). RFC 6746 (E), IRTF, Nov 2012.

[15] R. Atkinson and S. N. Bhatti. IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6). RFC 6744 (E), IRTF, Nov 2012.

[16] R. Atkinson and S. N. Bhatti. Optional Advanced Deployment Scenarios for the Identifier-Locator Network Protocol (ILNP). RFC 6748 (E), IRTF, Nov 2012.

[17] R. Atkinson, S. N. Bhatti, and S. Hailes. Mobility as an Integrated Service Through the Use of Naming. In *MobiArch 2007 - 2nd ACM/IEEE Intl. Workshop on Mobility in the Evolving Internet Architecture*, pages 1:1–1:6, Aug 2007.

[18] R. Atkinson, S. N. Bhatti, and S. Rose. DNS Resource Records for the Identifier-Locator Network Protocol (ILNP). RFC 6742 (E), IRTF, Nov 2012.

[19] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972 (PS), IETF, Mar 2005.

[20] M. Bagnulo. Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6181 (I), IETF, Mar 2011.

[21] S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, April 2006.

[22] C. J. Bennett, S. W. Edge, and A. J. Hinchley. Issues in the interconnection of datagram networks. IEN 1, July 1977.

[23] P. Bhagwat and C. E. Perkins. A mobile networking system based on Internet Protocol(IP). In *Mobile & Location-Independent Computing Symposium on Mobile & Location-Independent Computing Symposium*, MLCS, pages 7–7, Berkeley, CA, USA, 1993. USENIX Association.

[24] S. Bhatti and R. Atkinson. Reducing DNS caching. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 792 –797, april 2011.

[25] S. Bhatti and R. Atkinson. Secure & Agile Wide Area Virtual Machine Mobility. In *Proc. IEEE MILCOM 2011*, Oct 2012.

[26] S. Bhatti, R. Atkinson, and J. Klemets. Integrating challenged networks. In *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, pages 1926 –1933, nov. 2011.

[27] J. Bi, Y. Wang, and K. Gao. Implementation Experience of Identifier-Locator Network Protocol for IPv6 (ILNPv6). Internet Draft draft-bi-rrg-ilnpv6-implementation-experience-04, Tsinghua University, May 2015. URL: https://tools.ietf.org/html/draft-bi-rrg-ilnpv6-implementation-experience-04.

[28] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing Skype traffic: when randomness plays with you. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 37–48, New York, NY, USA, 2007. ACM.

[29] S. Bradner. The Internet Standards Process – Revision 3. RFC 2026 (BCP), IETF, Oct 1996.

[30] C. Caini and R. Firrincieli. TCP Hybla: a TCP enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking*, 22, 2004.

[31] B. Carpenter, J. Crowcroft, and Y. Rekhter. IPv4 Address Behaviour Today. RFC 2101 (I), IETF, Feb 1997.

[32] B. E. Carpenter. IP Addresses Considered Harmful. *SIGCOMM Comput. Commun. Rev.*, 44(2):65–69, Apr. 2014.

[33] H. Chan, D. Liu, P. Seite, H. Yokota, and J. Korhonen. Requirements for Distributed Mobility Management. RFC 7333 (I), IETF, August 2014.

[34] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei. Quantifying Skype user satisfaction. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 399–410, New York, NY, USA, 2006. ACM.

[35] S. Cheshire, Z. Zhu, R. Wakikawa, and L. Zhang. Understanding Apple's Back to My Mac (BTMM) Service. RFC 6281 (I), IETF, june 2011.

[36] B. Constantine, G. Forget, R. Geib, and R. Schrage. Framework for TCP Throughput Testing. RFC 6349 (I), IETF, Aug 2011.

[37] A. Cooper, H. Tschofenig, B. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith. Privacy Considerations for Internet Protocols. RFC 6973 (I), IETF, July 2013.

[38] K. Daniel Wong, A. Dutta, H. Schulzrinne, and K. Young. Simultaneous mobility: analytical framework, theorems and solutions. *Wireless Communications and Mobile Computing*, 7(5):623–642, 2007.

[39] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963 (PS), IETF, Jan 2005.

[40] A. Dhraief and N. Montavont. Toward Mobility and Multihoming Unification - The SHIM6 Protocol: A Case Study. In *IEEE WCNC 2008 - Wireless Communications and Networking Conf.*, pages 2840–2845, March 2008.

[41] R. Droms (Ed), J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (PS), IETF, Jul 2003.

[42] H. Elaarag. Improving TCP Performance over Mobile Networks. *ACM Computing Surveys*, 34(3):357–374, Sept. 2002.

[43] P. Eronen. IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555 (PS), IETF, june 2006.

[44] A. Ezzouhairi, A. Quintero, and S. Pierre. A New SCTP mobility scheme supporting vertical handover. In *WiMob 2006 - IEEE Intl. Conf. Wireless and Mobile Computing, Networking and Comms.*, pages 205–211, June 2006.

[45] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. The Locator/ID Separation Protocol (LISP). RFC 6830 (E), IETF, Jan 2013.

[46] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural Guidelines for Multipath TCP Development. RFC 6182 (I), IETF, Mar 2011.

[47] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (E), IETF, Jan 2013.

[48] C. P. Fu and S. Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. *Selected Areas in Communications, IEEE Journal on*, 21(2):216–228, Feb 2003.

[49] A. Galvani, A. Rodriguez-Natal, A. Cabellos-Aparicio, and F. Risso. LISP-ROAM: Network-based Host Mobility with LISP. In *MobiArch 2014*, pages 19–24, 2014.

[50] A. Gladisch, R. Daher, and D. Tavangarian. Survey on Mobility and Multihoming in Future Internet. *Wireless Personal Communications*, 74(1):45–81, 2014.

[51] A. Grilo, P. Estrela, and M. Nunes. Terminal independent mobility for IP (TIMIP). *Communications Magazine, IEEE*, 39(12):34–41, 2001.

[52] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. RFC 5213 (PS), IETF, Aug 2008.

[53] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Operating Systems Review*, 42(5):64–74, July 2008.

[54] T. Henderson, P. Nikander, and M. Komu. Using the Host Identity Protocol with Legacy Applications. RFC 5338 (E), IETF, Sep 2008.

[55] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (DS), IETF, Feb 2006.

[56] IEEE. Guidelines for 64-bit Global Identifier (EUI-64). Technical report. URL: http://standards.ieee.org/develop/regauth/tut/eui64.pdf.

[57] J. Ioannidis, D. Duchamp, and G. Q. Maguire, Jr. IP-based protocols for mobile internetworking. In *Proceedings of the conference on Communications architecture & protocols*, SIGCOMM '91, pages 235–245, New York, NY, USA, 1991. ACM.

[58] E. Ivov and T. Noel. An experimental performance evaluation of the IETF FMIPv6 protocol over IEEE 802.11 WLANs. In *Proc IEEE WCNC 2006*, volume 1, pages 568–574, April 2006.

[59] H. Jang, A. Yegin, K. Chowdhury, J. Choi, and T. Lemon. DHCP Options for Home Information Discovery in Mobile IPv6 (MIPv6). RFC 6610 (PS), IETF, May 2012.

[60] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. *IEEE/ACM Trans. Netw.*, 10(5):589–603, Oct. 2002.

[61] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (PS), IETF, Dec 2005.

[62] M. Komu, M. Bagnulo, K. Slavov, and S. Sugimoto. Sockets Application Program Interface (API) for Multihoming Shim. RFC 6316 (I), IETF, Jul 2011.

[63] M. Komu and T. Henderson. Basic Socket Interface Extensions for the Host Identity Protocol (HIP). RFC 6317 (E), IETF, Jul 2011.

[64] R. Koodli. IP Address Location Privacy and Mobile IPv6: Problem Statement. RFC 4882 (I), IETF, May 2007.

[65] R. Koodli. Mobile IPv6 Fast Handovers. RFC 5568 (PS), IETF, July 2009.

[66] J. Laganier and L. Eggert. Host Identity Protocol (HIP) Rendezvous Extension. RFC 5204 (E), IETF, Apr 2008.

[67] D. Lee, J. Kim, and P. Sinha. Handoff-aware adaptive media streaming in mobile IP networks. In *International Conference on Information Networking*, 2006.

[68] D. Liu, J. Zuniga, P. Seite, H. Chan, and C. Bernardos. Distributed Mobility Management: Current Practices and Gap Analysis. RFC 7429 (I), IETF, January 2015.

[69] Q. Liu, S. Li, H. He, and B. Wang. A Multi-binding Solution for Simultaneous Mobility of MIPv6. In *Service-Oriented System Engineering, 2006. SOSE '06. Second IEEE International Workshop*, pages 143–146, Oct 2006.

[70] J. Manner and M. Kojo. Mobility Related Terminology. RFC 3753 (I), IETF, June 2004.

[71] J. McNair and F. Zhu. Vertical handoffs in fourth-generation multinetwork environments. *Wireless Communications, IEEE*, 11(3):8–15, 2004.

[72] A. Melo. DCCP on Linux. In *Ottawa Linux Symposium 2005*, pages 305 – 311, july 2005.

[73] M. Menth, M. Hartmann, and D. Klein. Global Locator, Local Locator, and Identifier Split (GLI-Split). *Future Internet*, 5(1):67–94, 3 2013.

[74] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984 (I), IETF, Sep 2007.

[75] A. P. Miettinen and J. K. Nurminen. Energy Efficiency of Mobile Clients in Cloud Computing. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 4–4, Berkeley, CA, USA, 2010. USENIX Association.

[76] K. Mitsuya, M. Isomura, K. Uehara, and J. Murai. Adaptive Application for Mobile Network Environment. In *5th International Conference on ITS Telecommunications (ITST)*, pages 211–214, 2005.

[77] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson. Host Identity Protocol Version 2 (HIPv2). RFC 7401 (PS), IETF, April 2015.

[78] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (E), IETF, Apr 2008.

[79] M. Mudassir Feroz and A. Kiani. SHIM6 Assisted Mobility Scheme, an intelligent approach. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 725–728, Jan 2013.

[80] M. Muslam, H. Chan, L. Magagula, and N. Ventura. Network-based mobility and Host Identity Protocol. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 2395–2400, April 2012.

[81] J. Mysore and V. Bharghavan. A new multicasting-based architecture for internet host mobility. In *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '97, pages 161–172, New York, NY, USA, 1997. ACM.

[82] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (DS), IETF, Sep 2007.

[83] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (DS), IETF, Sep 2007.

[84] P. Nikander, T. Henderson, C. Vogt, and J. Arkko. End-Host Mobility and Multihoming with the Host Identity Protocol. RFC 5206 (E), IETF, Apr 2008.

[85] P. Nikander and J. Laganier. Host Identity Protocol (HIP) Domain Name System (DNS) Extension. RFC 5204 (E), IETF, Apr 2008.

[86] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533 (PS), IETF, Jun 2009.

[87] M. O'Dell. An Alternate Addressing Architecture for IPv6. Internet-Draft draft-odell-8+8-00.txt, Internet Engineering Task Force, 1996. Expired draft. URL: http://www.potaroo.net/ietf/all-ids/draft-odell-8+8-00.txt.

[88] M. O'Dell. GSE - An Alternate Addressing Architecture for IPv6. Internet-Draft draft-ietf-ipngwg-gseaddr-00.txt, Internet Engineering Task Force, 1997. Expired draft. URL: https://tools.ietf.org/html/draft-ietf-ipngwg-gseaddr-00.

[89] J. Pan, R. Jain, S. Paul, and C. So-in. MILSA: A New Evolutionary Architecture for Scalability, Mobility, and Multihoming in the Future Internet. *Selected Areas in Communications, IEEE Journal on*, 28(8):1344–1362, October 2010.

[90] A. Pappas, S. Hailes, and R. Giaffreda. Mobile Host Location Tracking Through DNS. In *LCS2002: 2002 IEEE London Communications Symposium*, Sep 2007.

[91] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (PS), IETF, Nov 2010.

[92] C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. RFC 6275 (PS), IETF, Jul 2011.

[93] C. Pluntke, L. Eggert, and N. Kiukkonen. Saving Mobile Device Energy with Multipath TCP. In *Proceedings of the Sixth International Workshop on MobiArch*, MobiArch '11, pages 1–6, New York, NY, USA, 2011. ACM.

[94] Y. Qiu, F. Zhao, and R. Koodli. Mobile IPv6 Location Privacy Solutions. RFC 5726 (E), IETF, Feb 2010.

[95] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley. Opportunistic Mobility with Multipath TCP. In *ACM MobiArch 2011*, pages 7–12. ACM, 2011.

[96] R. Ramjee, K. Varadhan, L. Salgarelli, S. R. Thuel, S.-Y. Wang, and T. La Porta. HAWAII: a domain-based approach for supporting mobility in wide-area wireless networks. *IEEE/ACM Trans. Netw.*, 10(3):396–410, June 2002.

[97] J. Ramos-munoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, and J. Lopez-soler. Characteristics of mobile youtube traffic. *Wireless Communications, IEEE*, 21(1):18–25, February 2014.

[98] D. Rehunathan, R. Atkinson, and S. Bhatti. Enabling Mobile Networks Through Secure Naming. In *Proc. IEEE MILCOM 2009*, Oct 2009.

[99] A. Rodriguez Natal, L. Jakab, M. Portoles, V. Ermagan, P. Natarajan, F. Maino, D. Meyer, and A. Cabellos Aparicio. LISP-MN: Mobile Networking Through LISP. *Wireless Personal Communications*, 70(1):253–266, 2013.

[100] R. Rosenbaum. Using the Domain Name System To Store Arbitrary String Attributes. RFC 1464 (E), IETF, May 1993.

[101] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (PS), IETF, June 2002.

[102] J. Saltzer. On the Naming and Binding of Network Destinations. RFC 1498 (I), IETF, Aug 1993.

[103] M. Scharf and A. Ford. Multipath TCP (MPTCP) Application Interface Considerations. RFC 6897 (I), IETF, Mar 2013.

[104] S. Schütz, H. Abrahamsson, B. Ahlgren, and M. Brunner. Design and Implementation of the Node Identity Internetworking Architecture. *Computer Network*, 54(7):1142–1154, May 2010.

[105] M. Z. Shafiq, F. Le, M. Srivatsa, and A. X. Liu. Cross-path Inference Attacks on Multipath TCP. In *HotNets XII - 12th ACM Wkshp. Hot Topics in Networks*, pages 15:1–15:7. ACM, 2013.

[106] B. Simpson and S. N. Bhatti. An Identifier-Locator Approach to Host Multihoming. In *AINA 2014 - IEEE 28th Intl. Conf. Advanced Information Networking and Applications*, May 2014.

[107] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 155–166, New York, NY, USA, 2000. ACM.

[108] H. Soliman, C. Castelluccia, K. ElMalki, and L. Bellier. Hierarchical Mobile IPv6 (HMIPv6) Mobility Management. RFC 5380 (PS), IETF, Oct 2008.

[109] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (I), IETF, Jan 2001.

[110] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (PS), IETF, Sep 2007.

[111] R. Stewart, M. Tuexen, and G. Camarillo. Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures. RFC 5062 (I), IETF, Sep 2007.

[112] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061 (PS), IETF, Sep 2007.

[113] C. Sunshine and J. Postel. Addressing mobile hosts in the ARPA Internet environment. IEN 135, Mar. 1980.

[114] F. Teraoka, Y. Yokore, and M. Tokoro. A network architecture providing host migration transparency. In *Proceedings of the conference on Communications architecture & protocols*, SIGCOMM '91, pages 209–220, New York, NY, USA, 1991. ACM.

[115] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (DS), IETF, Sep 2007.

[116] A. G. Valkó. Cellular IP: a new approach to Internet host mobility. *SIGCOMM Comput. Commun. Rev.*, 29(1):50–65, Jan. 1999.

[117] R. Wakikawa, G. Valadon, and J. Murai. Migrating home agents towards internet-scale mobility deployments. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, pages 10:1–10:10, New York, NY, USA, 2006. ACM.

[118] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[119] B. Wellington. Secure Domain Name System (DNS) Dynamic Update. RFC 3007 (PS), IETF, Nov 2000.

[120] W. Xing, H. Karl, A. Wolisz, and H. Mller. M-SCTP: Design And Prototypical Implementation Of An End-To-End Mobility Concept. In *In Proc. 5th Intl. Workshop The Internet Challenge: Technology and Applications*, 2002.

[121] X. Xu. Routing Architecture for the Next Generation Internet (RANGI) . Internet-Draft draft-xu-rangi-04.txt, Internet Engineering Task Force, 2010. Expired draft. URL: https://tools.ietf.org/html/draft-xu-rangi-04.

[122] H. Yokota, K. Chowdhury, R. Koodli, B. Patil, and F. Xia. Fast Handovers for Proxy Mobile IPv6. RFC 5949 (PS), IETF, Sep 2010.

[123] M. Zekri, B. Jouaber, and D. Zeghlache. A review on mobility management and vertical handover solutions over heterogeneous wireless networks. *Computer Communications*, 35(17):2055 – 2068, 2012.

[124] Z. Zhu, R. Wakikawa, and L. Zhang. A Survey of Mobility Support in the Internet. RFC 6301 (I), IETF, july 2011.

[125] M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of YouTube network traffic at a campus network - Measurements, models, and implications. *Comput. Netw.*, 53(4):501–514, Mar. 2009.