Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 7464179, 29 pages
https://doi.org/10.1155/2019/7464179

Research Article

End-To-End Mobility for the Internet Using ILNP

Ditchaphong Phoomikiattisak[1] **and Saleem N. Bhatti**[2]

[1]*Geo-Informatics and Space Technology Development Agency, Bangkok, Thailand*
[2]*School of Computer Science, University of St Andrews, St Andrews, Fife KY16 9SX, UK*

Correspondence should be addressed to Saleem N. Bhatti; saleem@st-andrews.ac.uk

Received 7 May 2018; Revised 13 December 2018; Accepted 20 January 2019; Published 16 April 2019

Academic Editor: Yu Chen

As the use of mobile devices and methods of wireless connectivity continue to increase, seamless mobility becomes more desirable and important. The current IETF Mobile IP standard relies on additional network entities for mobility management, can have poor performance, and has seen little deployment in real networks. We present a host-based mobility solution with a true end-to-end architecture using the *Identifier-Locator Network Protocol (ILNP)*. We show how the TCP code in the Linux kernel can be extended allowing legacy TCP applications that use the standard C sockets API to operate over ILNP without requiring changes or recompilation. Our direct testbed performance comparison shows that ILNP provides better host mobility support than Mobile IPv6 in terms of session continuity, packet loss, and handoff delay for TCP.

1. Introduction

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 for mobile nodes (MNs) over the Internet. While the IETF *Mobile IPv4 (MIPv4)* [1] and *Mobile IPv6 (MIPv6)* [2] solutions have been defined for some time and implementation is available, they have seen little deployment due to their complexity and performance.

In this paper, we describe and evaluate an implementation of a true end-to-end approach to mobility. Our mechanism uses ILNPv6, a superset of IPv6 that implements an Internet architecture described by the *Identifier-Locator Network Protocol (ILNP)*. We present a testbed-based performance evaluation of ILNPv6 as an in-kernel modification to the Linux kernel (not a simulation). Using an unmodified TCP application, we show that our approach is fully backwards compatible with the existing sockets API and has better performance than MIPv6.

1.1. Contributions of This Paper. Our overall exposition in this paper is placed in the context of the first comprehensive performance evaluation of a Linux kernel implementation of TCP running over ILNPv6 in direct comparison with TCP over Mobile IPv6 (MIPv6). In so doing, we make the following scientific and engineering contributions:

(i) We present an architectural evaluation of an Identifier-Locator based solution to mobility (ILNP) against the current IETF architectural approach. *We show that a new Internet architecture, which has a radically different approach to addressing, can be used over current infrastructure to implement IP-level mobility.* This includes detailed, qualitative protocol-level analysis against MIPv6 as well as comparisons with the other IETF proposals.

(ii) To evaluate our new architectural approach, we provide a rigorous empirical performance comparison of TCP operation over ILNPv6 against MIPv6, both with and without MIPv6 Route Optimisation (RO) enabled. *We show that ILNPv6 outperforms MIPv6 in all cases we have tested. We provide the first detailed analysis of flow-level handoff dynamics for MIPv6, in comparison with ILNP.* Our evaluation includes operation over different end-to-end delays and examination of throughput, delay, loss, and retransmission attempts during the handoff period.

(iii) To assess application-level impact, we have compared in detail the performance of TCP CUBIC (the Linux default TCP variant) with TCP Hybla [3] and TCP Veno [4], the latter two TCP variants having designs that are optimised for wireless and heterogeneous networks. *Crucially, we show that the use of ILNP changes the design landscape for transport protocol operation in mobile IP scenarios.* Current transport protocols assume the presence of loss during handoff, and so their designs are focused on dealing with loss. *ILNP drastically reduces gratuitous loss during handoff (to near zero loss), so loss is no longer a major factor to consider in the transport protocol design for handoff. Immediately, this has benefits in performance for all existing TCP variants, as loss causes TCP to slow down transmission through its normal congestion control behaviour.*

(iv) *We have shown that it is possible for an architecturally radical approach, such as ILNP, which deprecates the use of IP addresses, to be implemented directly on current systems and work across current IPv6 infrastructure.* We extended our codebase from [5] allowing existing (legacy) IPv6 TCP applications to operate over ILNPv6 without modification, as initially demonstrated with some basic results in [6]. Our results in this paper are based on an in-kernel implementation and are fully backwards compatible with the existing C `sockets` (2) API. *This demonstrates a realistic and tractable path to enable both backwards compatibility and incremental deployment for ILNP.*

Overall, we show that it is possible to deploy on today's IPv6 infrastructure a new and truly end-to-end mobility architecture for IP. This new architecture does not require any of the additional mechanisms that are used in other IP mobility solutions: tunnelling; agents; proxies or other middleboxes; overloading of IP address semantics; or any additional entities, protocols, or state for routing.

1.2. Structure of This Paper. We start with an overview of ILNP and with a description of the handoff process for ILNP and mobile IP in Section 2. Then, the most relevant related work is presented in Section 3, with the focus on solutions that have been assessed by the IETF or IRTF for potential global deployment. In Section 4, we describe the ILNPv6 extensions to TCP, showing how ILNPv6 can be implemented as a superset of IPv6. Section 5 provides a description of the testbed and metrics used for our evaluation. Thorough comparative performance evaluation of TCP over ILNPv6 and MIPv6 on our testbed is presented in Section 6. After a discussion of some key issues in Section 7, we conclude with a summary in Section 8.

1.3. Abbreviations and Acronyms

API: Application Programming Interface;

AR: Access Router;

BU: Binding Update;

CN: Correspondent Node;

CoA: Care-of Address;

EID: Endpoint Identifier;

FA: foreign agent;

FMIP: Fast Handover for Mobile IP;

FQDN: Fully Qualified Domain Name;

HA: home agent;

HI: Host Identifier;

HIP: Host Identity Protocol;

HMIP: Hierarchical Mobile IP;

HoA: home address;

IETF: Internet Engineering Task Force;

IRTF: Internet Research Task Force;

ILNP: Identifier-Locator Network Protocol;

ILNPv6: ILNP as a superset of IPv6;

L64: 64-bit locator value for ILNP;

LAN: Local Area Network;

LISP: Locator-Identifier Separation Protocol;

LMA: Local Mobility Anchor;

LU: Locator Update;

MAG: Mobile Access Gateway;

MIP: Mobile IP;

MN: Mobile Node;

MP-TCP: Multipath Transmission Control Protocol;

MSS: Maximum Segment Size;

NAT: Network Address Translation;

NCoA: Next Care-of Address

NID: 64-bit node identifier for ILNP;

OTA: Over the Air;

PMIP: Proxy Mobile IP;

PoA: Previous Care-of Address;

RA: Router Advertisement;

RLOC: Routing Locator;

RO: routing optimisation;

RTT: round trip time;

RVS: Rendezvous Server;

SCTP: Stream Control Transport Protocol;

SHIM6: Level 3 Multihoming Shim Protocol for IPv6;

TCP: Transmission Control Protocol;

UDP: User Datagram Protocol;

WAN: wide-area network;

WLAN: wireless local area network.

Table 1: Use of names in IP and ILNP (modified from [7].).

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.

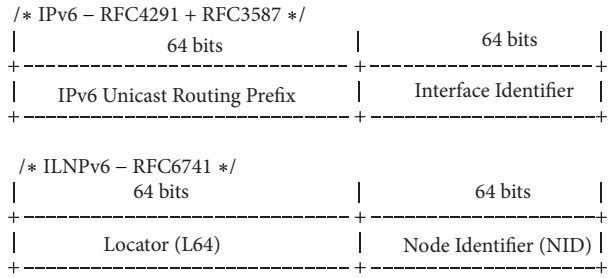

Figure 1: IPv6 unicast address format and ILNPv6 unicast address format. The L64 value has the same syntax and semantics as the IPv6 routing prefix. The NID value has the same syntax as the IPv6 Interface Identifier but has different semantics.

2. ILNP: A New Architecture Supporting Mobility

The *Identifier-Locator Network Protocol (ILNP)* [7–17] is an Internet Research Task Force (IRTF) Experimental protocol. It has a host-based, end-to-end architecture, is designed to support mobility (in harmony with other functions, such as multihoming), and can be implemented as a superset of IPv6 called *ILNPv6*.

2.1. Identifier-Locator Approaches to Networking. While the first ideas on separation of identity and location were discussed at the very early stages of thinking for the Internet [18], these ideas were not incorporated into the architectural design. Since then, the potential problems in the way IP addresses are used have been highlighted by the Internet community at intervals over several decades, e.g., in 2007 [19], in 1997 [20], and in 1977 [18]. Indeed, it is necessary to reconsider the use of addressing and the (mis)use of IP addresses in general [21].

So, the use of an Identifier-Locator approach to networking is now receiving considerable interest in the research community. There are many engineering solutions proposed, not just for mobility (e.g., [22, 23]), but also for improving routing scalability (e.g., [24–26]).

2.2. Architecture: Overview. A key architectural concept of ILNP [7] is the recognition that the overloaded use of IP addresses and static bindings between objects in the communication stack creates major problems in implementing functions (such as mobility) for the Internet [21].

So, ILNP deprecates the use of IP addresses, replacing them with two new distinct data types to explicitly acknowledge the presence of an *identity* and a *location* for an IP node. ILNPv6 defines both *node identifier (NID)* and network *locator (L64)* values, along with dynamic bindings to implement various functionalities, including host mobility. As shown in Table 1, instead of using the IP address in various layers across the protocol stack, ILNPv6 uses 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 state invariance for transport protocol session state. The NID represents an identity of a node which is tied to the whole node, not just a single network interface for that node. The network layer uses a L64 value, which is topologically significant, for routing and forwarding. The L64 value represents a single IPv6 subnetwork.

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 can change as a mobile node (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.

In Figure 1, the IPv6 address structure is compared with the use of NID and L64 values. Essentially, an L64 value has the same syntax and semantics as an IPv6 routing prefix, and the NID value has the same syntax as an IPv6 Interface Identifier. So, an ILNPv6 packet is treated the same as an IPv6 packet by IPv6 routers, which means that equipment such as switches and routers that can handle IPv6 packets can also handle ILNPv6 packets. However, when the ILNPv6 packet reaches the end-system, the NID value is interpreted differently, having different semantics to an IPv6 Interface Identifier. So, end-systems that wish to use IPv6 need to be upgraded as described in this paper. Note, however, that

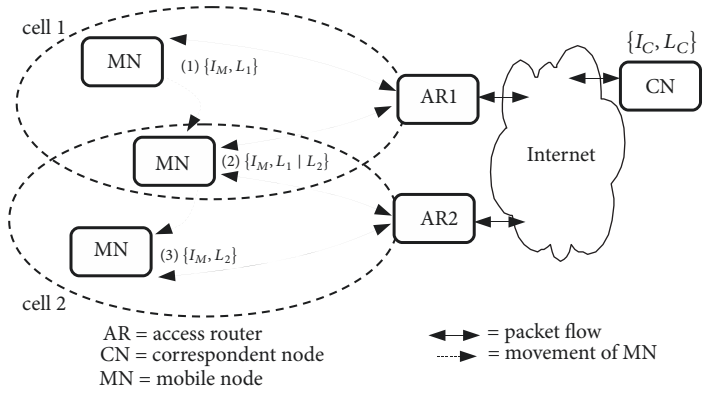

FIGURE 2: An example scenario of host mobility with ILNP using network layer (IP layer) soft-handoff. A mobile node (MN) at (1) uses identifier I_M and locator L_1 at access router AR1. As the MN moves from cell 1 to cell 2, in the overlap region (2), it can use both its current locator, L_1, and a new locator value, L_2, that is provided by AR2 in cell 2. (The new locator value is identical to an IPv6 routing prefix, as can be discovered from IPv6 Router Advertisement (RA) messages.) In the overlap region, MN is multihomed. When MN moves into cell 2 completely (3), it uses only L_2. As I_M does not change, end-to-end state invariance can be preserved for transport protocols, maintaining end-to-end integrity.

it is only those *end-systems* that need to use IPv6 that are upgraded; the *site* network devices (switches and routers) do not necessarily need to be upgraded, as long as they support IPv6 already.

2.3. A Simple Mobility Scenario. Figure 2 shows a simple example of handoff in ILNPv6. A MN, with NID value I_M, using locator L_1 in cell 1 moves to cell 2, where it will use locator L_2. When the MN enters the overlapping region between cell 1 and cell 2, the value of L_2 would be available to MN 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 correspondent node (CN) of this new value using a *Locator Update (LU)* message [12], synonymous to an IPv6 Binding Update (BU) message. The *Locator 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. The use of DNS for ILNP is described in [11, 27], but is only required for nodes that expect incoming connections, e.g., mobile servers. 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*). For hard-handoff, gratuitous 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 there is no soft-handoff supported by the subnetwork technology across the handoff region, e.g., between different administrative WLAN cells or between different technologies such as from a 3G or 4G cell to a WLAN cell in a vertical handoff scenario.

2.4. Handoff Process for MIP and ILNP. For our evaluation in Section 6, we compare directly MIPv6 with ILNPv6 in terms of their performance for application flows during handoff. So, in this section, we describe the differences in the handoff process between MIPv6 and ILNPv6.

In summary, the ILNPv6 handoff process has approximately the same overhead as that for MIPv6 without route optimisation (RO). However, MIPv6 without RO causes performance problems for the data flow during handoff, due to the well-known problem of *triangular routing* for MIP. So, it is recommended that MIPv6 is always used with RO. Our experiments in Section 6 consider MIPv6 with and without RO, for a rigorous evaluation. (Our results here are all for TCP, but a comparison of the handoff process with UDP data flows is presented in our previous work [5].)

2.4.1. MIPv6 Handoff. When a MN performs handoff in MIPv6, the new care-of address (CoA) of the MN must be updated at its home agent (HA). If RO is in use, the CN must also be made aware of the new CoA, so that data packets can be delivered to the correct location in the network. Figure 3 shows the signalling required for handoff for MIPv6 without RO. Two signals are required: (i) a Binding Update (BU) from the MN to the HA and (ii) a Binding Acknowledgement (BAck) in response, to inform the MN that the BU has been processed. In Figure 3, T_{HA} denotes the processing time at the HA.

If RO is enabled, additional signals are required as shown in Figure 4:

(i) *Home Test Init (HoTI)* and *Home Test (HoT)* are needed to test the reachability of the HoA from the CN. (T_{HoT} is the time taken to process the HoT at the CN.) Both HoTI and HoT must be sent via the HA.

(ii) *Care-of Test Init (CoTI)* and *Care-of Test (CoT)* are needed to test the reachability of the new CoA from the CN. (T_{CoT} is the processing time of CoT at the CN.)

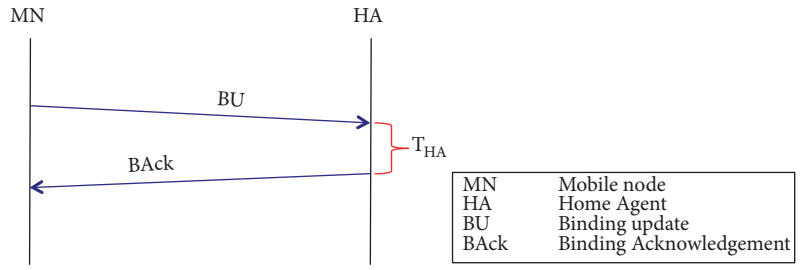

Figure 3: The handoff process for MIPv6 without RO.

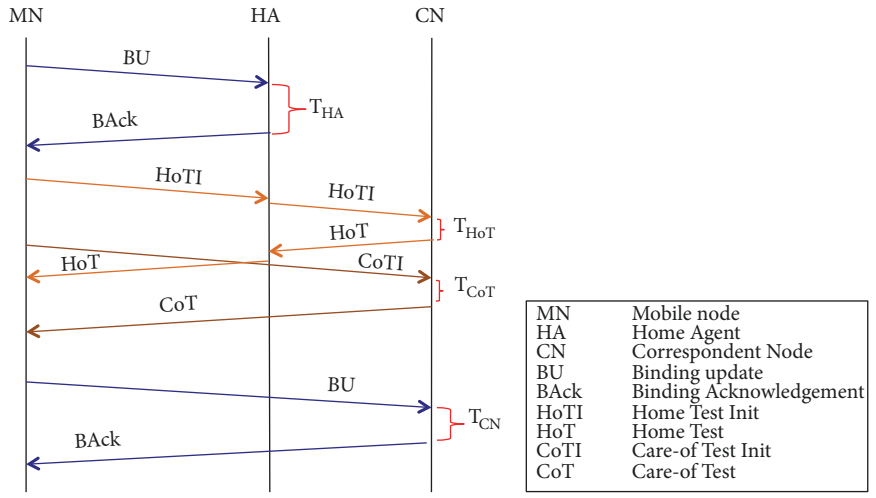

Figure 4: The handoff process for MIPv6 with RO.

(iii) BU and BAck between the MN and the CN to update the CoA. (T_{CN} is the time taken to process the BU at the CN.)

2.4.2. ILNPv6 Handoff. The handoff process in ILNPv6 is much simpler: all that is required is that the new *L64* value of the MN is signalled to the CN. As shown in Figure 5, ILNPv6 uses a simple handshake: LU and LU-ACK, directly between the MN and the CN. (T_{CN} is the time taken to process the LU at the CN.)

In some situations, the MN may also need to update the L64 value to a directory service, such as the DNS. However, this would only be required if the MN is a mobile server and expects incoming connections, e.g., if a WWW server is running. Even then, this may depend on the application: many applications have their own mechanisms for establishing presence—an application-specific *rendezvous* service—and do not rely on DNS. For example, Skype uses a peer-to-peer model with its own presence and resolver mechanism [28]. If specific application-level integration was required for ILNP, further studies would be needed. So, this process is not included in our studies because it is not always necessary for an MN but is suitable for future work in an application-specific context.

3. Related Work

We present here a selection of proposed solutions for host mobility support, focusing on those that have been reviewed by the IETF or the IRTF, i.e., those that are considered to be deployable at scale. A more comprehensive list of mobility solutions can be found in RFC6301 [29] and also previous surveys of mobility mechanisms [30, 31].

Table 2 compares these 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 necessarily require additional network entities* to achieve host mobility.

The key item to note from Table 2 is that all solutions apart from ILNP reuse IP addresses. ILNPv6 does use the IPv6 address fields in the IPv6 packet (for backwards compatibility and to ease deployment), but they are used to carry identifier and locator values and *not* an address.

3.1. Previous Work on ILNP. The architectural description and protocol engineering considerations for ILNP are given in RFC6740 [7] and RFC6741 [10]. The initial ideas of host mobility using ILNP can be found in [9, 32, 33]. Assessment of its feasibility using overlay emulation can be found in

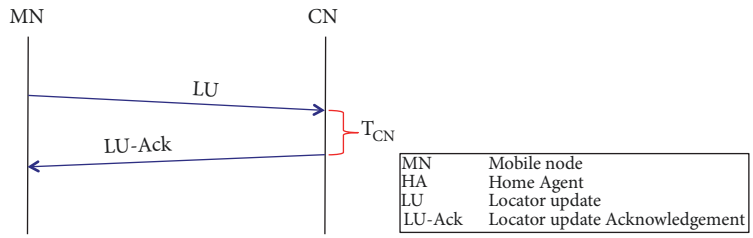

Figure 5: Control signals during an MN handoff for ILNPv6.

[34]. The emulation results confirm that host mobility using a network layer soft-handoff mechanism provided by ILNPv6 is feasible.

The first ILNP mobility prototype in Linux is described in [35], and the initial evaluation for packet flows shows that the results matched the feasibility study using an overlay. The extensions to a full UDP implementation in Linux and the performance evaluation of UDP using ILNPv6 are presented in [5], showing that UDP applications running over ILNPv6 using a wireless network have excellent handoff performance (better than MIPv6) in terms of throughput, packet loss, and handoff delay.

The first, basic, preliminary results for TCP operation and performance of ILNP as implemented in the Linux kernel are presented in [6]. That paper shows that IPv6 applications using the `sockets` (2) interface can operate over ILNPv6 *without being modified* and at the same time offer performance that is better than MIPv6.

In the control plane, ILNPv6 also performs better than MIPv6 [36]: (i) ILNPv6 has lower overhead than MIPv6 in most cases, both in terms of packet overhead and signalling traffic volume; and (ii) ILNPv6 has better reliability in handoff completion when the network is lossy.

3.2. Network-Based Solutions. Network-based solutions (such as MIPv6, PMIPv6, and LISP) require additional network entities. These can often have the advantage that they 'hide' mobility from nonmobile (legacy) nodes, improving backwards compatibility. However, the addition of new network entities adds complexity and costs to the current network landscape: new equipment may be required, increased capital expenditure, and there is an overhead for operations, administration overhead for network and systems management, and so an impact on operational expenditure. The addition of new network entities may require reconfiguration of existing network infrastructure, which may in turn destabilise existing functions, possibly impacting reliability. Additionally, new network entities, such as proxies, may introduce a single point of failure, become performance bottlenecks, and also introduce new points that need to be monitored and protected from security attacks. Indeed, the presence of new, additional entities may in turn lead to new, additional attack vectors, which might only be discovered after a successful security attack has occurred.

3.2.1. Mobile IP and Extensions. Currently, the IETF *Mobile IP (MIP)* standard is the most well-known solution for IP mobility. MIP uses both indirection and redirection to allow MNs to roam.

MIPv4 [1], based on IPv4, uses implicit *indirection* to support MNs. An MN has a permanent *home address (HoA)* at its home network, which is fixed and acts as an identifier. It also has a mutable *care-of address (CoA)*, which acts as a locator, and is assigned from the foreign network (FN) into which the MN roams. From an engineering viewpoint, proxies—a *home agent (HA)* and a *foreign agent (FA)*—connected with IP-in-IP tunnels (between HoA and CoA) are used to give the impression that a MN is topologically stable. This creates suboptimal routing—called *triangular routing*—and end-to-end integrity of the transport protocol session may be lost.

MIPv6 [2] initiates communication via the HA as for MIPv4. However, it can then use *redirection*, by sending *Binding Update (BU)* messages to signal a topologically correct address, i.e., its CoA, from its new location to remote hosts. This allows a MN to communicate directly with a CN without passing data packets through the HA; i.e., triangular routing is removed, after the communication session has started.

A major issue for both MIPv4 and MIPv6 is handoff performance: high gratuitous packet loss can occur when a MN moves between networks, as *hard-handoff* is used; the 'old' network connectivity is dropped and the 'new' network connectivity is initiated at both sender and receiver. The MN and CN act independently; there may be packets 'in flight' when handoff occurs, using old CoA values, and so gratuitous packet loss occurs. Various extensions to MIPv6 have been proposed to tackle this problem, such as Hierarchical Mobile IPv6 (HMIPv6) [37], and Fast Handover for Mobile IPv6 (FMIPv6) [38].

HMIPv6 introduces a *Mobility Anchor Point (MAP)* to manage mobility of MNs in its local region. So, when a MN moves within the region, the handoff latency can be reduced; hence gratuitous packet loss can also be reduced. FMIPv6 reduces gratuitous packet loss by allowing a MN to detect that it has moved to another network when it is still connected to its current network, a form of *soft-handoff*. The new CoA (NCoA) can also be determined ahead of the handoff, which can be used immediately after it moves to that subnet. The previous access router also creates a tunnel to provide continuity between use of the previous CoA (PCoA) and NCoA. Any delayed packets (packets in flight) arriving at the PCoA would be forwarded to the NCoA. Clearly, both HMIPv6 and FMIPv6 add complexity to the handoff process, additional signalling overhead, and new network

Table 2: Comparison of selected 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 addressing	HoA & CoA	HoA & CoA	HoA & CoA	EID & RLOC	HI & IP	ULID & L	NID & L64
Re-use of IP address	Yes	Yes	Yes	Yes	Yes	Yes	No
Supported legacy address space	IPv4	IPv6	IPv6	IPv4/IPv6	IPv4/IPv6	IPv6	IPv4^{+}/IPv6
Concurrent multipath transfer	No	No	No	No	No	No	Yes
Tunneling	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.
CoA: care-of address.
EID: endpoint identifier.
FA: foreign agent.
HA: home agent.
HI: Host Identifier.
HoA: home address.
L: locator.
L64: locator.
LMA: Local Mobility Anchor.
MAG: Mobile Access Gateway.
NID: node identifier.
RLOC: Routing Locator.
RVS: Rendezvous Server.
ULID: Upper Layer Identifier.

entities (such as the MAP for HMIPv6) need to be provided and managed.

There is another form of MIPv6, *Proxy Mobile IPv6 (PMIPv6)* [39]. This approach enhances MIPv6 to be a completely network-based solution. MNs do not get involved in the mobility management process. A MN still has a HoA and a CoA, but PMIPv6 introduces another entity, a *Mobile Access Gateway (MAG)*, to track movements of MNs on its link and signal a *Proxy Binding Update* message to the MN's *Local Mobility Anchor (LMA)*, similar to a HA in MIPv6. The traffic between MAG and LMA uses a bidirectional tunnel. To minimise the problem of gratuitous packet loss during handoff of PMIPv6, a Fast Handover mechanism is proposed [40], applying the concepts of FMIPv6 to improve PMIPv6 performance.

To minimise potential adverse issues with routing performance and single point of failure, a *Distributed Mobility Management (DMM)* mechanism has been introduced to extend the IETF standard protocols, i.e., the mobile IP family. RFC7333 [41] 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 suboptimal routes. Each anchor is ideally placed near the MNs for maximising performance. RFC7429 [42] 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 RFC7333. However, the DMM approach would still suffer the usual drawbacks associated with the use of middleboxes, multiple, distributed mobility anchors; additional signalling overhead; an increased security attack surface; single points of failure; and performance bottlenecks.

3.2.2. LISP. The *Locator-Identifier Separation Protocol (LISP)* [43] uses the 'map-and-encap' method for mapping IP addresses into a separate routing schema, using *Endpoint Identifier (EID)* and *Routing Locator (RLOC)* values. Additional management and control modules (a *mapping system*) are required to map between these two values and encapsulate IP packets sent between LISP routing nodes. However, the map-and-encap function increases both the per-packet protocol overhead and the routing complexity of the deployed network. As LISP was originally designed for multihoming purposes, there are now two extensions to LISP which have been proposed for mobility support: LISP mobile node (LISP-MN) [23] and LISP-ROAM [44].

3.3. Host-Based Solutions. Host-based solutions, such as HIP, ILNP, and SHIM6, usually do not require additional network entities, and so do not introduce additional complexity into the network. They have the potential disadvantage that they require updates to the end-system protocol stack. However, today's modern operating systems (OSs), for desktops, for servers, and for mobile devices, regularly use network-based (or 'over the air' (OTA)) software updates, so we take the position that deployment of code updates for end-systems could be managed easily, without special mechanisms, during the normal administrative processes that are common to the management of modern computing and communication systems. For example, the major desktop operating systems (Linux, Apple macOS, and Microsoft Windows) all provide regular, OTA upgrades and updates to the OS, as do the main mobile/handheld device (Google Android and Apple iOS).

3.3.1. HIP. The *Host Identity Protocol (HIPv2)* [45, 46] separates identity of a host from its IP address using public and private key pairs. The public key is used as a Host Identifier by higher layer protocols (such as TCP) to represent the host identity, while an IP address is used for routing. Hence, HIP requires the deployment and use of strong cryptography, even within protected enclaves. This could impair performance, both of application protocols and of packet processing, due to a higher computational burden in per-packet cryptography. Although DNS may be used as a rendezvous mechanism to initiate the connection, for improved performance, it is recommended that the HIP Rendezvous Server (RVS, a new network entity) should be deployed. HIP also requires a new API for applications [47], and hence does not work for legacy applications. The HIP-Aware Agent [48] could be used to allow legacy applications to operate over HIP but is yet another entity that would need to be deployed, managed, maintained, and protected within a deployment scenario.

3.3.2. SHIM6. The *Level 3 Multihoming Shim Protocol for IPv6 (SHIM6)* [49] is a host-based solution that implements Locator-Identifier separation. SHIM6 requires implementation of an extra 'shim' layer between the network and the transport protocol to perform mapping between identifier and locator values, with both identifier and locator values being IPv6 addresses. In addition, SHIM6 is not designed to enable mobility but is aimed at multihoming. Mobility support could be possible for SHIM6 [50], but there is a problem in high rehoming time (i.e., high handoff latency), while optimisation mechanisms are work-in-progress, e.g., [51]. Mobility support for a multihomed mobile node is also possible [52].

3.4. Other Possible Solutions. There are also potential solutions at the transport layer and the application layer. However, they usually have limitations in that they may be designed to support a specific transport protocol, or a specific application, and so might not support mobility for every type of service. Transport-layer solutions like the *Stream Control Transmission Protocol (SCTP)* [53] and *Multipath TCP (MP-TCP)* [54, 55] provide mobility support for only specific transport protocols, which means they do not support applications that use different transport protocols, such as UDP for real-time, interactive voice, and video. Application layer solutions like *Session Initiation Protocol (SIP)* [56] provide mobility support through signalling for session management, by integrating infrastructure with some services, e.g., VoIP. However, not all types of services and applications can be supported by SIP. Therefore, we take the position that, architecturally, the network layer is the most suitable place to tackle host mobility *generally* for IP based applications, but that solutions such as

SCTP and MP-TCP may offer suitable engineering solutions in certain, *specific* circumstances.

Another approach to a host-based solution is to use a host-oriented network address translation (NAT) function, as described in [57]. However, while easing deployment, the use of a NAT function on the host itself could create undesired interactions with network-deployed NAT functions, as well as raising the usual issues that are associated with the use of NAT functions, e.g., loss of true end-to-end connectivity.

3.4.1. Multipath TCP (MP-TCP). In general, transport-layer mobility solutions provide a platform on which specific application mobility solutions can be built [58], and so are complimentary to solutions such as ILNP, which aim to provide more general, IP layer mobility.

MP-TCP [54, 55] 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 was enabling multiple-path transport connections, i.e., for multihomed nodes. Mobility using MP-TCP could be achieved by dynamically adding and removing subflows when a host enters and exits a network [59]. 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 be aware of multiple paths transfer, may be beneficial in fully utilising MP-TCP [60].

Despite the potential capability of multihoming and mobility support, MP-TCP introduces new security threats, summarised in RFC6181 [61]. The security risks are mostly from the arbitrary adding of subflows to the ongoing connection state, which could cause, for example, denial-of-service and man-in-the-middle attacks. A new cross-path interference attack also has been identified [62]. This new attack allows people from one subflow to gain information (such as throughput, packet loss, and round trip time) of another subflow. At the time of writing, work is in progress to update MP-TCP to address the various security issues.

4. TCP with ILNPv6 in the Linux Kernel

In this section, we highlight that although ILNP uses a radically different *architecture* to IP, judicious *engineering* allows much of the existing IPv6 code to be reused. This allows a dual-stack—IPv6/ILNPv6—kernel, with ILNPv6 being realised as a superset of IPv6, to support TCP operation, including different TCP variants. Also, by design, ILNPv6 packets are forwarded by IPv6 routers as if they are IPv6 packets, so ILNPv6 can communicate across the existing, global IPv6 core.

Our relevant previous work is as follows:

(i) In [5], we describe how some basic IP layer and UDP layer (only) modifications in the Linux kernel were made to provide basic ILNPv6 functionality as a proof of concept. The key contribution of that paper was to show the potential for low loss during handoff.

(ii) In [6], initial, basic modifications of TCP code in Linux kernel v3.9.0 were made to enable operation of legacy TCP applications over ILNPv6. This showed possibility that TCP-based flows could operate over ILNP using the standard C `sockets` (2) API without any knowledge of ILNPv6.

For the results in this paper, we have extended our implementation so that

(i) ILNPv6 is implemented as a true-superset of IPv6 in the Linux v3.9.0 kernel. This means that ILNPv6 can coexist with IPv6, supporting backwards compatibility and incremental deployment.

(ii) Packet processing paths for both the user-plane and control plane for IPv6 and ILNPv6 are integrated, including mobile IP.

(iii) Full TCP integration, with state management and segment processing support for operation over both IPv6 and ILNPv6, including mobile IP. This now works for any variant of TCP, not just the default TCP version (TCP CUBIC).

4.1. TCP State Management. For transport protocols, like TCP, the changes required were (i) to bind end-system state only to the node identity, NID, not the whole IP address (also modifications to protocol handling, such as pseudo-header checksum computation); (ii) to set up and maintain dynamic bindings between the NID and L64 value(s); and (iii) to set up and maintain dynamic bindings between L64 values and interfaces (this impacted interaction with other protocols, such as Neighbour Discovery).

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) cause the state to become invalid.

$$\langle tcp : P_X, P_Y, A_X, A_Y \rangle \langle ip : A_X, A_Y \rangle \langle if : A_X \rangle \quad (1)$$

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

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

According to our example of handoff in Figure 2, the MN using NID value I_M is in cell 1 using locator L_1 and moves to cell 2 and starts to use locator L_2. Assuming that a transport flow is in progress with the CN using $[I_C, L_C]$, then

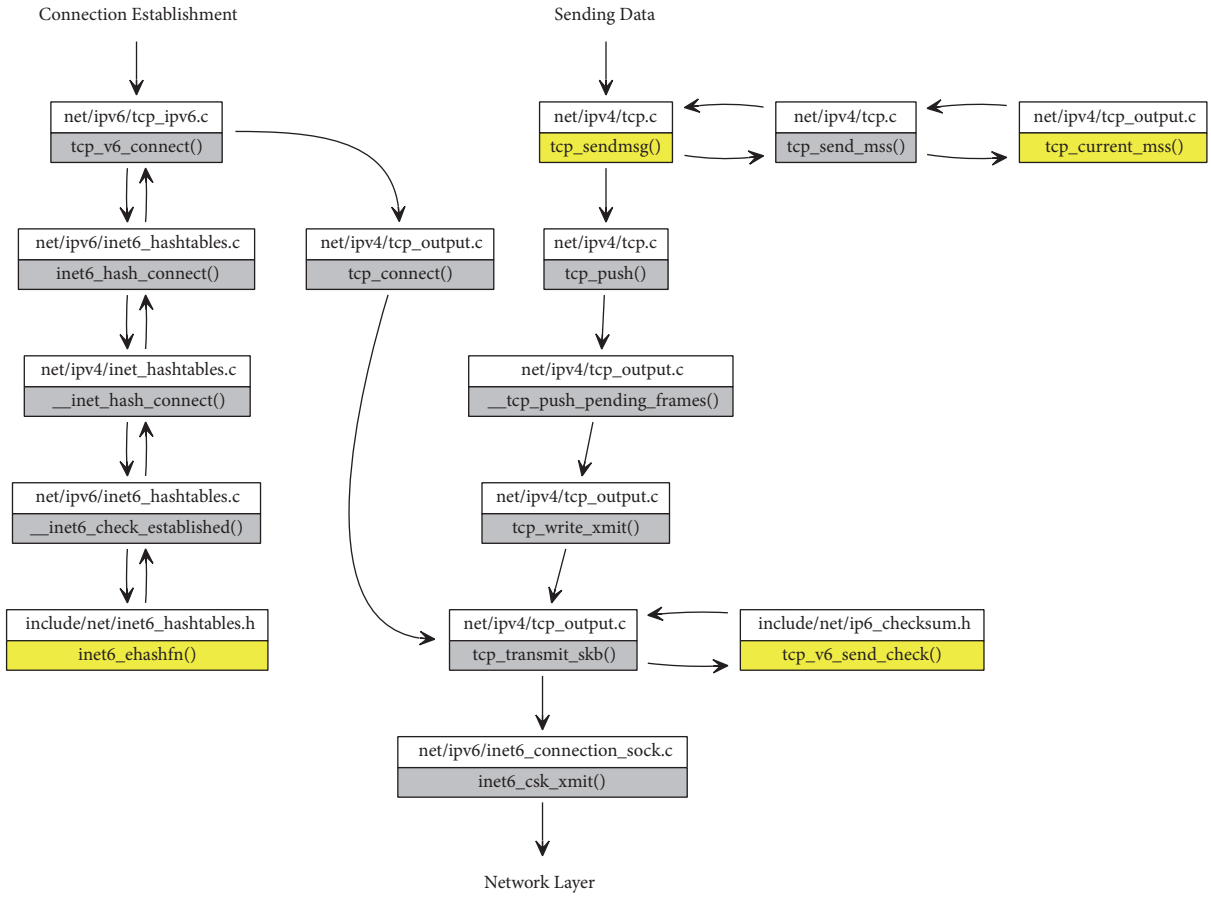

Figure 6: A function call graph in the Linux kernel for sending a packet in the TCP layer. The grey boxes are unmodified functions, and the functions in yellow are modified to handle ILNPv6 in a dual-stack operation with IPv6.

the network layer and transport-layer state for ILNP at MN can be represented by the tuple expression:

$$\langle tcp : P_M, P_C, I_M, I_C \rangle \langle ilnp : L_1, L_C \rangle \quad (3)$$

When the MN enters the overlapping region between cell 1 and cell 2, the value of L_2 would be available through IPv6 RAs. The MN receives L_2 and now informs the CN of this new value using a ILNPv6 *Locator Update (LU)* message [12]. At this point, the MN could just drop the use of L_1 (for hard-handoff), but ILNPv6 permits a NID value to be bound to one or more L64 values simultaneously, allowing network layer soft-handoff, which minimises gratuitous packet loss during handoff. In the overlap region, the MN expression for our transport flow would now be

$$\langle tcp : P_M, P_C, I_M, I_C \rangle \langle ilnp : L_1 \mid L_2, L_C \rangle \quad (4)$$

It can be seen that the transport-layer tuple is not affected during handoff in ILNP: end-to-end state is preserved.

4.2. Implementation. This section explains how the Linux kernel v3.9.0 TCP code can be modified to support ILNPv6.

Figure 6 shows the functions and call graph for sending a TCP packet. First, before sending data packets, TCP requires a connection establishment with another endpoint. The establishment process starts at `tcp_v6_connect()`. First, information about the session is added to the TCP hashtable (e.g., source and destination IP address and source and destination port number). The `inet6_ehash_fn()` function 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 in place of the full IP address.

Sending of data packets starts with a call to the `tcp_sendmsg()` function. The function was modified to mark the socket data structure as an ILNPv6 socket if ILNPv6 was used, using an additional flag in the socket data structure. The first step before sending a data packet is 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 [13] 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 call graph: `tcp_transmit_skb()`, which calls function `tcp_v6_send_check()` for TCP checksum calculation. The code here was

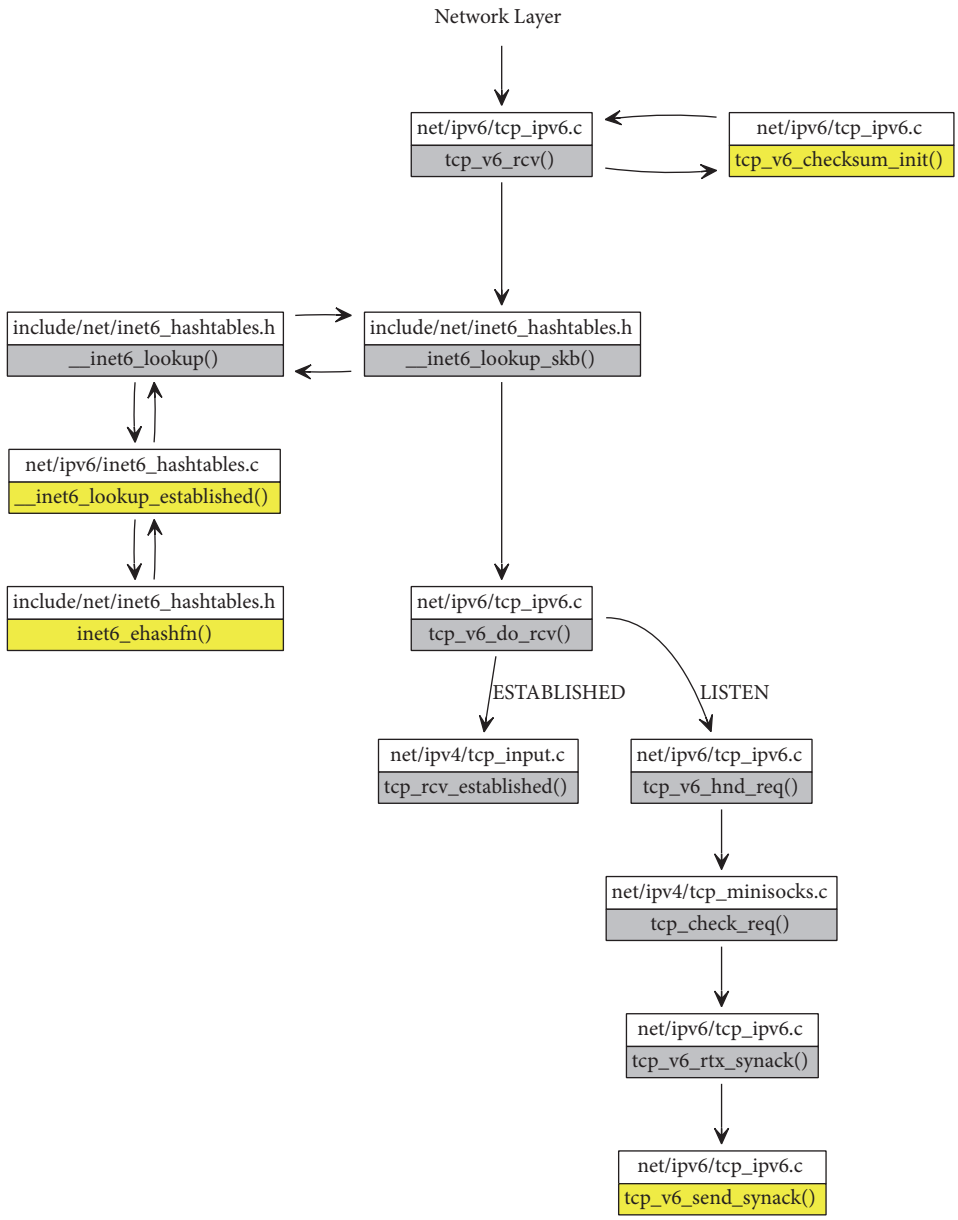

Figure 7: A function call graph in the Linux kernel for receiving a packet in the TCP layer. The grey boxes are unmodified functions, and the functions in yellow are modified to handle ILNPv6 in a dual-stack operation with IPv6.

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 operations, as explained in [5].

For the receiving of a flow, the processing path of a TCP/ILNPv6 packet is shown in Figure 7. Verified packets from the network layer are forwarded to TCP at `tcp_v6_rcv()`. The TCP checksum calculation is performed by a modified version of the function `tcp_v6_checksum_init()`, again, to allow for the use of only the NID for checksum calculation. Then, the TCP hashtable lookup is performed by the modified `inet6_ehash_fn()` as stated above. After the lookup is successful, the function `inet_lookup_established()` determines which socket (bound to a specific application) that the packet should be forwarded to, by comparing the provided source NID, destination NID, source port, and destination port values to ones stored in the TCP hash-table.

Once the correct socket structure is found, `tcp_v6_do_rcv()` invokes different functions depending on the TCP state, as for IPv6. 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 the NID values for checksum calculations instead of full IPv6 addresses.

4.2.1. Temporary, Practical Limitations. Since only the in-kernel, core TCP operation at the transport layer is modified to support ILNPv6 at the moment, the *Generic Segmentation Offload (GSO)* (http://www.linuxfoundation.org/collaborate/workgroups/networking/gso) function, including *TCP segmentation offload (TSO)*, and the TCP checksum offload, all of which operate at the network device level, should currently be disabled (using `ethtool(8)`) when using ILNPv6. These functions are normally used on high-performance server systems, and not normally used on desktop systems or mobile/handheld devices. Also, this is a temporary measure only: of course, those functions will in due course also be modified in the appropriate device drivers to provide support for ILNPv6 before being enabled, and that is left for future work.

5. Testbed: Configuration and Metrics

Our previous work [5, 35] shows that UDP applications operating over ILNPv6 have excellent handoff performance, and the results match the feasibility study (using an overlay) that was reported in [34]. For TCP, a naive expectation would be that we would observe similar results as we have for UDP: that ILNPv6, especially, with soft-handoff, should show better performance than MIPv6 in terms of session continuity, gratuitous packet loss, and handoff delay. However, such a naive expectation does not take into account TCP's congestion control algorithm(s), which modulate the flow transmission behaviour.

In this evaluation, MIPv6 was chosen for performance comparison against ILNPv6 for several reasons. Firstly, it is considered the standard for IP mobility. Secondly, it works with legacy applications without requiring any changes or extensions to the current `socket` (2) API. Thirdly, there is an existing Linux kernel implementation which can be used for direct, comparative evaluations.

For other standardised solutions listed in Section 3, there are some constraints preventing a straightforward and appropriate comparison of application-level performance with ILNPv6, as explained below:

(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` (2) API have to be modified to operate over HIP [47]. Legacy applications may be used, but special treatments are needed [48].

(2) SHIM6 was designed for multihoming support. There is still no standard mobility solution using SHIM6. Also, extensions to the C `sockets` (2) API are recommended [63] to allow maximal usage of the protocol.

(3) LISP was originally designed for multihoming support, but can also now support mobility. The mobility extensions, both LISP-MN [23] and LISP-ROAM [44], require additional network entities to be in place, so neither present an end-to-end solution, and would require support from network service providers.

(4) PMIPv6 is a purely network-based solution, hence it has a completely different model to ILNPv6. Also, the use of a proxy does not present a transparent, end-to-end solution.

(5) MP-TCP and SCTP both operate at the transport layer only, unlike ILNPv6 and MIPv6, both of which are network layer mechanisms.

So, by using MIPv6, it is possible to use exactly the same application binary, *iperf* (https://iperf.fr/) in our experiments, allowing a *direct* and *fair* comparative evaluation with ILNPv6. *iperf* has been widely used previously in performance comparisons of packet flows with TCP. Note that *iperf* itself can experience performance issues when used at extremely high data rates (e.g., many 100s of Mbps, or Gbps), but our use was limited to lower data rates, as explained below.

5.1. Experiment Configuration. The network topology of the experiment is shown in Figure 8. All systems were Gateway GR380 F1 servers with Intel Xeon 5500 series processors (note that such a relatively high specification is not necessary for running either ILNPv6 or MIPv6. This was the hardware available to us at the time of our evaluation and allowed us to execute all experiments and measurements comfortably, without the need to be concerned about the operational performance bounds of the testbed). The specifications of the machines are shown in Table 3.

The connection between the MN and the routers used IEEE 802.11ac 2x2 WLAN links. R2 and R3 used the WLAN interfaces to create 2 different wireless networks using *hostapd* (http://wireless.kernel.org/en/users/Documentation/hostapd); i.e., they acted as wireless access points. The MN had 2 wireless interfaces. The first interface was configured to connect to the network announced by R2, and the second one was for a connection to R3. The other nodes were connected by wired Ethernet 1Gbps links. All nodes ran Linux kernel version 3.9.0. *Note that R1, R2, and R3 ran a standard Linux codebase, with no code modifications for ILNPv6 capability*, to demonstrate that ILNPv6 packets can traverse unmodified IPv6 routers. The MN and CN also ran Linux kernel version 3.9.0 but with a modified kernel to support ILNPv6, and using also the standard, kernel MIPv6 code.

5.2. Handoff in the Testbed. Movement at the IP level results in a different subnetwork point of attachment (SNPA) and/or the use of a new IP subnet. We emulated this movement in our testbed by turning interfaces on and off. The use of interfaces in this way allowed a controlled experimental environment for reproducibility of results, while allowing some of the

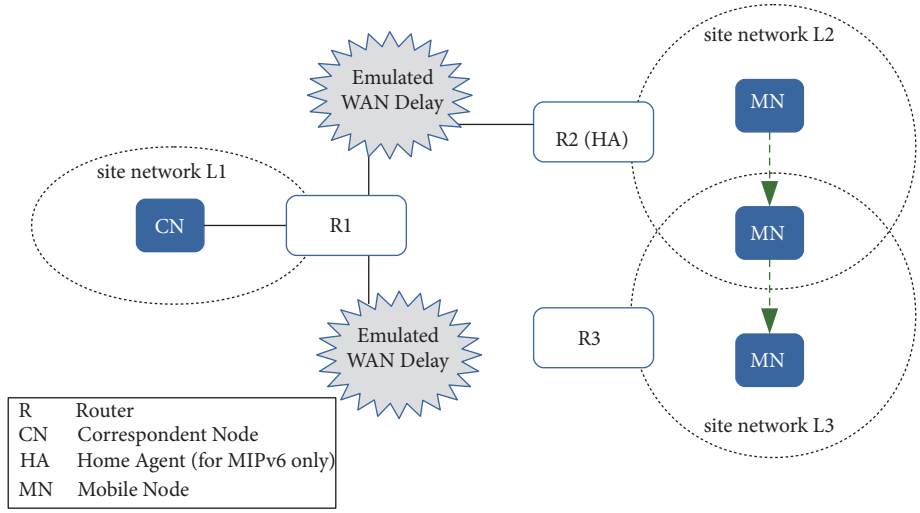

Figure 8: The topology for the experiment (also as in [5]). The CN connects to R1 via 1 Gbps Ethernet. The MN initially connects to R2(HA) using 2x2 IEEE 802.11ac; the dashed/blue circles depict the radio cell scenario being emulated. The green/dashed arrows identify movements of MN to site network L_3, the movements generating a handoff.

Table 3: Summary of the testbed hardware and software.

Model	Gateway GR380 F1
CPU	Intel Xeon E5520 @ 2.27GHz
Memory	12GB DDR3
OS	Either a *modified* Linux kernel version 3.9.0 (for MN and CN) or with *unmodified* Linux kernel version 3.9.0 (for R1, R2 and R3)
Ethernet driver	Intel igb v4.1.2-k
Wireless adapter	Edimax EW-7822UAC
Wireless driver	Realtek rtl8812au v4.2.2

dynamics of a widely used radio technology to be included in our evaluation.

In real wireless scenarios and application domains, the exact dynamics of operation and performance will depend on many factors for a radio-based interface, for example,

(i) the wireless communication systems in use, such as 3G, 4G, future 5G, IEEE 802.11 variant, and so on,

(ii) the performance and quality of service (QoS) issues resulting from vertical handoff, such as between 4G and WLAN, and so on (please see Sections 7.7 and 7.8),

(iii) environmental conditions impacting radio transmission, such as precipitation, foliage, building orientation, obstructions, and so on,

(iv) the dynamics and nature of movement, such as speed of movement, direction of movement, especially if considering 3-dimensional movement, and not just people/vehicles along the ground, and so on,

(v) network engineering considerations, such as radio cell-size, RF channel allocation and management strategy, mobility models of users, and so on.

To account for all of these factors would obfuscate the dynamics of the network (IP) level operation. Our work is focused on a new architecture, protocol, and behaviour at the network (IP) level, so our comparison and evaluation were also at the network level protocol (and above), and not on the RF network radio technologies. Indeed, IP, by its initial design and philosophy, operates independently of any subnetwork technology, and so the design and evaluation of any IP-level mechanisms should not be dependent on any lower-layer technologies.

Of course, we would expect that if ILNP were to be considered for a real scenario or application domain, it would be vital to undertake the usual engineering practices before deployment, as required. This would include extensive site-surveys and suitable tests to establish absolute performance capability and to enable an appropriate network configuration to be defined.

5.3. Handoff Scenarios. Different network conditions for handoff scenarios were emulated using *netem* (http://www.linuxfoundation.org/collaborate/workgroups/networking/netem), by adding extra delay of 100ms in each direction between R1 and R2 and between R1 and R3. Note that our

use of *netem* is convenient and appropriate for our particular experiment, as it was not our intention to carry-out high-performance/stress tests of TCP. 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) 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

TCP flows were generated from the CN to the MN using *iperf* version 2.0.5. All TCP flows used the default values in *iperf* except that the TCP window size was limited to 320 Kbytes. In our initial experiments, we found that without this constraint, the CN could generate a localised congestion effect: TCP packets were queued at the MN, and in some cases prevented RA packets from being delivered in a timely manner. *Note that this is not a constraint of either ILNPv6 or MIPv6, but was a pathological condition of our testbed only.* Eventually, a handoff, for both ILNPv6 and MIPv6, could be improperly triggered because the MN had not seen an RA from its current access router during a certain period of time, so the old L64 value (IPv6 prefix) was considered to be stale. For ILNPv6, the problem can 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 was out of scope of this work. Therefore, limiting the TCP window size to constrain the amount of TCP traffic for both MIPv6 and ILNPv6 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.

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: *tcpdump* was used to capture packets at the MN and at the CN for *post hoc* analyses.

In Figure 8, the MN started in site network L_2, which was the home network for MIPv6. The *iperf* TCP flows were transmitted from the CN to the MN. 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. The movement was emulated using controlled, scripted invocations of *ifconfig* to bring the WLAN interfaces up and down. The tests were executed for each of the 4 handoff scenarios mentioned above: LAN to LAN, LAN to WAN, WAN to LAN, and WAN to WAN.

5.4. Measurements and Metrics. We used four performance metrics, as listed below.

(1) TCP flow data rate: TCP flow data rate graphs of different handoff scenarios are presented. This is to visualise how handoff by ILNPv6 and MIPv6 impacts ened-to-end 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 the data that can be sent, the better the TCP performance. This was a long enough duration to cover the handoff period and to allow TCP congestion control algorithms to operate.

(3) Retransmissions: 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 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 a single round trip time (RTT) between MN and CN.

In our results, where we have error bars, they are small, so the results are statistically sound; and in some results we have used box-plots in order to show clearly the distribution and variation of the results.

6. Evaluation: Results and Analyses

This section presents TCP performance over ILNPv6 and MIPv6. In general, our findings are that ILNPv6 performs better than MIPv6 in nearly all cases. We compare directly for each scenario (as described in Section 5.3) and for each metric (as described in Section 5.4).

6.1. 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 as an example of the time-domain flow dynamics that are typically observed. The TCP throughput is limited by the TCP window size and is calculated as in [64]

$$\begin{aligned} &\textit{Maximum TCP throughput} \\ &= \frac{8 \times \textit{TCP window size}}{\textit{RTT}} \end{aligned} \tag{5}$$

So, with a 320 Kbyte window (see above), the maximum TCP throughput in the LAN environment, which has RTT=~ 5 ms, is $(8 \times 320)/5 = 512$ Mbps (64 Mbytes/s). However, this high rate cannot be achieved 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 the WAN environment, with RTT ~200 ms, the throughput has a ceiling of $(8 \times 320)/200 = 12.8$ Mbps (1.6 Mbytes/s).

The handoff in MIPv6 and ILNPv6 happens at different times. The MIPv6 handoff is triggered when the MN moves out of the previous network completely as specified in RFC6275 [2, Sec. 11.5]. For ILNPv6, a handoff is triggered after seeing a new RA from the new network. We can see in the flow graphs that the MN performed handoff around the times t=21s to t=23s, for MIPv6 and around t=13s to t=15s, for ILNPv6. The MN moved into the new network at t=5, but it needed to wait for the wireless link association and

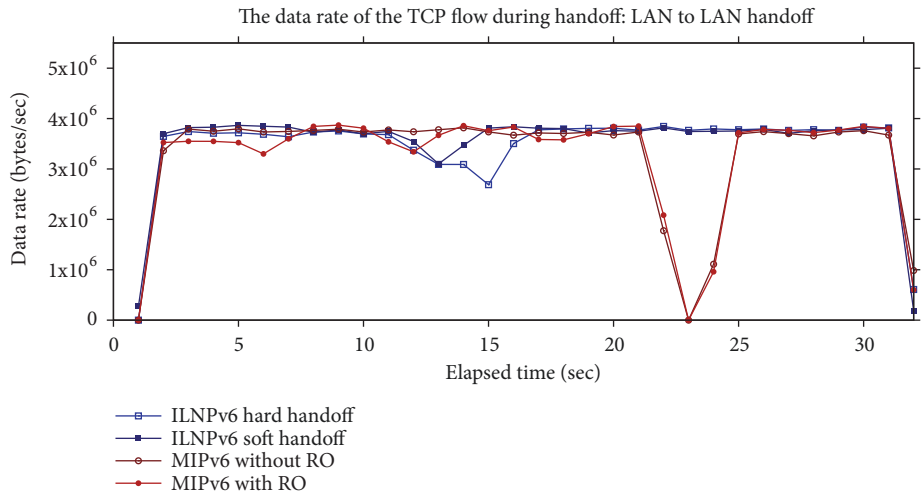

Figure 9: Typical profile for a TCP flow showing the data rate (bytes/sec) at the MN, with LAN to LAN handoff. There was an interruption during the MN handoff for MIPv6, while there was a small drop in data rate for ILNPv6.

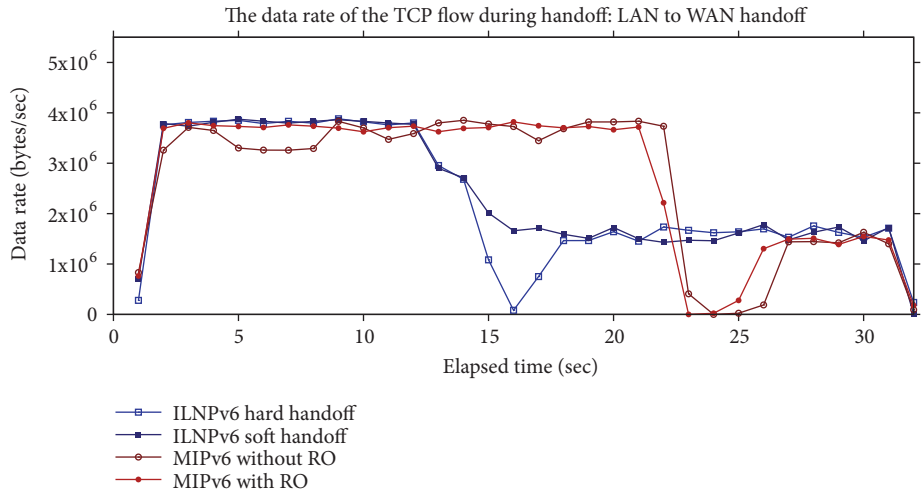

Figure 10: Typical profile for a TCP flow showing the data rate (bytes/sec) at the MN, with LAN to WAN handoff. There was an interruption during the MN handoff for MIPv6 and ILNPv6 hard-handoff, while there was no interruption for ILNPv6 soft-handoff. The data rate dropped after the handoff due to higher end-to-end delay.

the Duplicate Address Detection (DAD) before a handoff for ILNPv6 was triggered.

The key finding is that ILNPv6 with soft-handoff performed better than MIPv6 because the TCP flow was not interrupted when a handoff occurred, but MIPv6 suffered disruption to the flow, including large drops in throughput (sometimes down to zero throughput for several seconds). The detailed analysis of the flows in each handoff scenario is given below.

LAN to LAN Handoff. As shown in Figure 9, in the LAN environment, in the MIPv6 case, there was an interruption during the MN handoff, where the throughput dropped to zero. 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. There was a slightly greater drop in throughput using hard-handoff compared to soft-handoff. This result shows that TCP is more sensitive than UDP since, for UDP, we observed no loss and no drop in throughput during soft-handoff in our previous work [5]. The drop in throughput was caused by TCP retransmissions because of packet misordering (see Section 6.1.2 for more details).

LAN to WAN Handoff. From Figure 10, we observe that, during handing off from the LAN network to the WAN network, the TCP throughput decreased in every case. TCP sent less data because the RTT was much higher, and the TCP window size was limited (as explained in (5), Section 6.1). Nevertheless, ILNPv6 still performed better than MIPv6. MIPv6, again, suffered from a long interruption, when the MN could not receive any data, and the interruption was

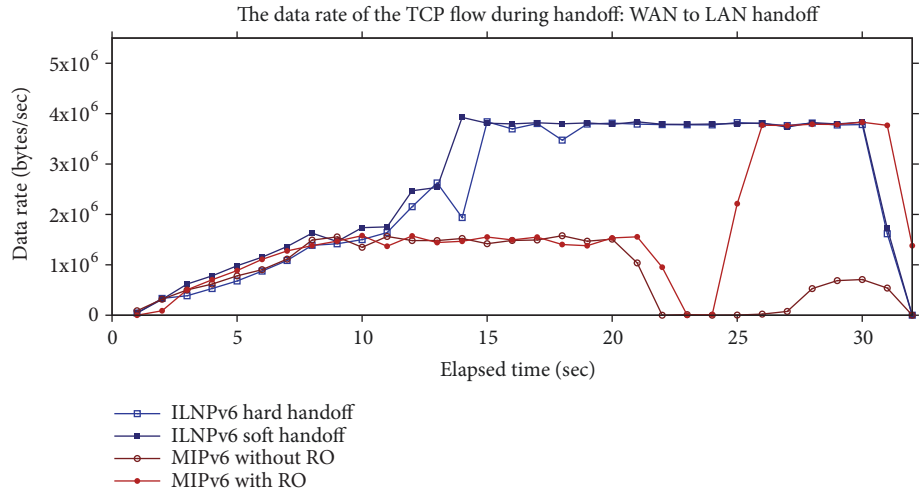

Figure 11: Typical profile for a TCP flow showing the data rate (bytes/sec) at the MN, with WAN to LAN handoff. There was an interruption during the MN handoff for MIPv6, while there was no interruption for ILNPv6. The data rate increased after the handoff due to a lower delay link, except MIPv6 without RO, where the traffic still traversed the high-delay link because of the tunnel to the HA.

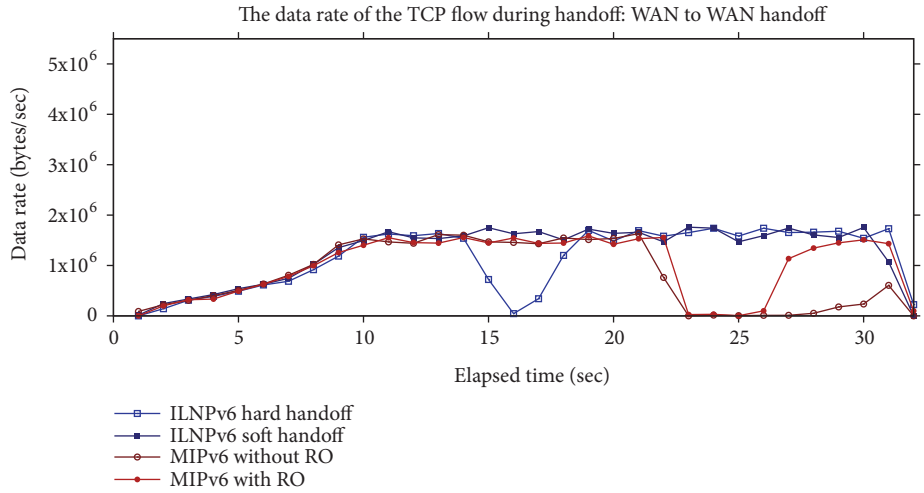

Figure 12: Typical profile for a TCP flow showing the data rate (bytes/sec) at the MN, with WAN to WAN handoff. There was an interruption during the MN handoff for MIPv6 and ILNPv6 hard-handoff, while there was no interruption for ILNPv6 soft-handoff.

worse than the LAN-LAN handoff. There was no significant difference observed for MIPv6 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 degraded gradually, consistent with normal TCP behaviour, due to the handoff to a link with a higher end-to-end delay.

WAN to LAN Handoff. The TCP flows in this handoff scenario showed an opposite trend to the previous one. The TCP throughput increased after the MN handoff from the WAN network to the LAN network, as displayed in Figure 11. 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 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 [65]. Due to the high-delay link, the slow start process took some time before reaching 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 Figures 9 and 10 that TCP throughput climbed faster at the beginning of the flows, which was to be expected as normal TCP behaviour.

WAN to WAN Handoff. ILNPv6 with soft-handoff still outperformed the others in this environment. As shown in Figure 12, there was no interruption of the flow for ILNPv6 soft-handoff. There was a small interruption when ILNPv6 operated with hard-handoff. For MIPv6, a longer

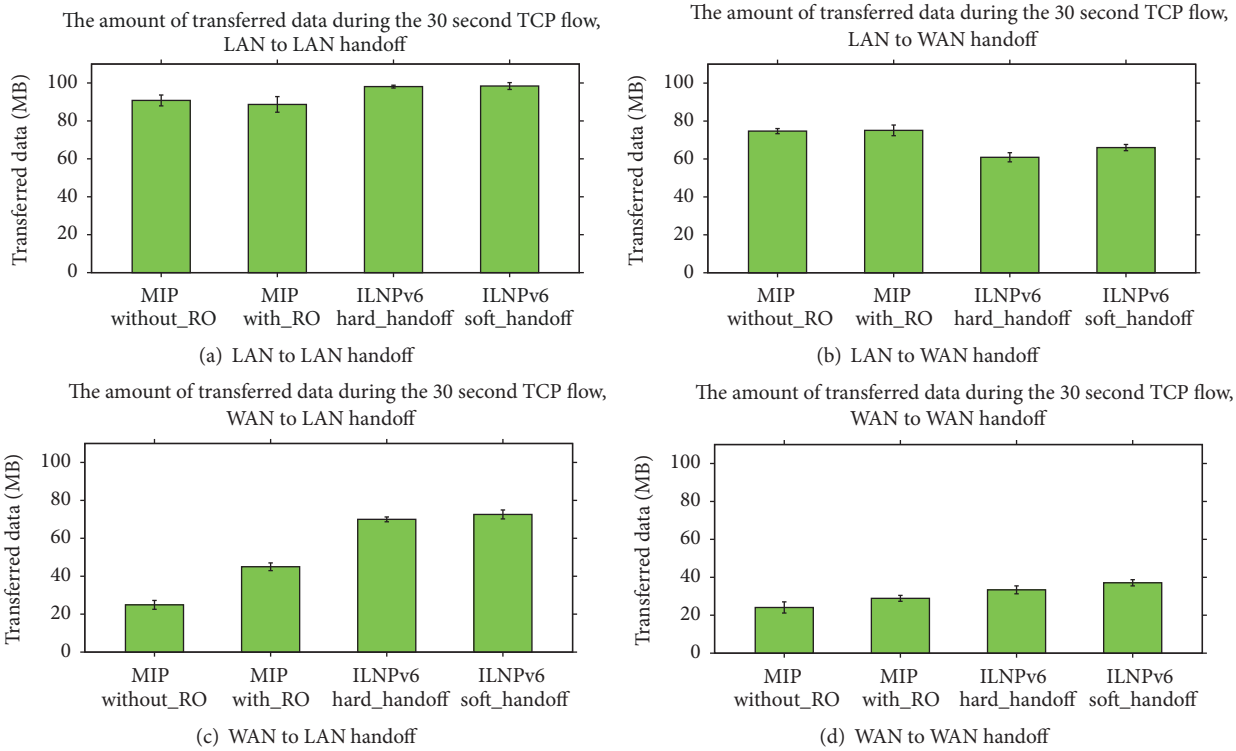

Figure 13: Total data transfer volumes during 30 second TCP traffic flows. With ILNPv6, TCP can transfer more data than with MIPv6 in almost every case, except when using handoff from LAN to WAN, because the MN uses the LAN link longer when MIPv6 is used.

interruption was observed. Moreover, the MN suffered the most when RO was disabled, as might be expected. 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 high, and TCP had unsatisfactory performance. Again, TCP slow start can be observed at the beginning of the flows.

6.1.1. Successfully Transferred Data. Figure 13 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 LAN-LAN environment, TCP sent the greatest amount of data because the RTT was low, and so the TCP throughput was high; TCP throughput and RTT are inversely proportional [64]. Therefore, in the WAN network (higher RTT), TCP sent less data.

In most cases, more data was sent when ILNPv6 was used, compared to MIPv6, because the data was sent without interruption (soft-handoff) or minimal interruption (hard-handoff); see Section 6.1. 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 area (t=5s, in this experiment). So, the MN spent a much longer time on the WAN network when using ILNPv6 than when using MIPv6, which hands off to the WAN network after it moves out of the LAN network (t=20s, in this experiment). Assuming that more data can be sent in the LAN network, it is understandable that MIPv6 allowed more data to be sent in this case. This is pathological to our experimental scenario, and MIPv6 has not been designed specifically to optimise handoff performance in such scenarios. Hence, to maximise TCP performance, an adjustment to the ILNPv6 or MIPv6 handoff decision algorithm may be required, e.g., let the MN stay on a 'better' link as long as possible before handoff. Of course, this may need information from the link layer, e.g., signal strength or other quality of service indicators. 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 in the 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). When RO is disabled, data packets must traverse the high-delay WAN link to the HA, causing an increase in RTT and hence lower data rate. This was avoided when RO was enabled, so the data rate improved.

6.1.2. Retransmissions Attempted. The number of TCP retransmissions in each handoff scenario is shown in Figure 14. This number was measured at the CN by counting sent packets having the same TCP sequence number.

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 far fewer retransmission attempts than MIPv6, as it has lower gratuitous packet loss during handoff. Especially, ILNPv6 with soft-handoff should, potentially, have

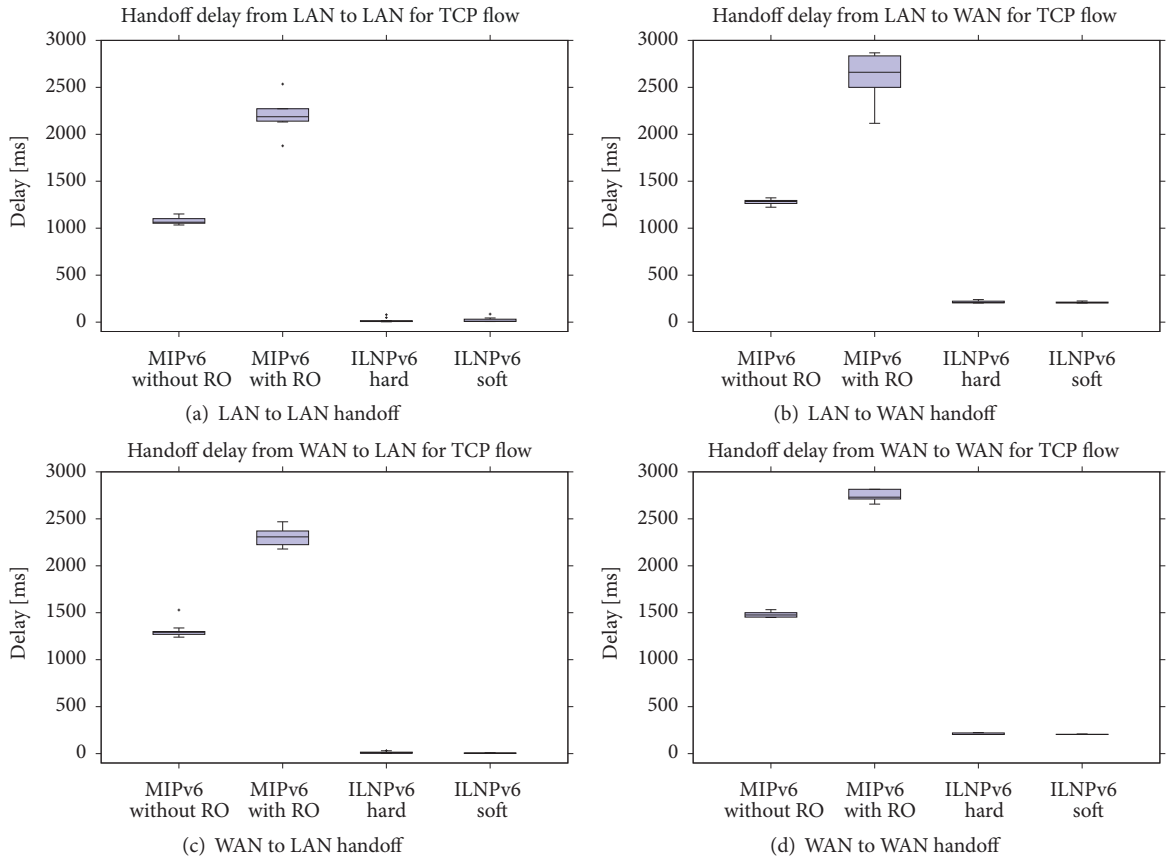

Figure 14: Number of retransmissions attempted at the CN. ILNPv6 with soft-handoff had the lowest number of retransmissions.

close to zero retransmissions because almost zero packet loss occurred. However, ILNPv6 had a larger number of retransmissions than expected. For hard-handoff, the numbers were close to the MIPv6 case. For soft-handoff, a lower number of retransmissions were observed than for MIPv6, 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 high (~30Mbps), so some packets were queued at the MN before getting processed. When a handoff occurred, for the ILNPv6 case, packets that came via the new link did not see any queue, and were thus 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 sequence number jump (packet reordering) as packet loss. Finally, the CN, which received many duplicate acknowledgements, had to retransmit packets which were in fact queued at the old link. Note that this response to sequence number jumps and duplicate acknowledgements is normal TCP behaviour, the fast retransmit/fast recovery mechanism [65, Section 3.2]. For hard-handoff, a similar situation 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 for 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 12.8Mbps, see the calculation above in Section 6.1)—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 the problem of queued packets was not observed. For ILNPv6 with hard-handoff, again, some retransmissions were observed due to packet loss during handoff.

Overall, ILNPv6 with soft-handoff had the lowest number of retransmissions for every case, which implied lowest gratuitous packet loss. However, some optimisations are needed in order to improve TCP performance with ILNPv6

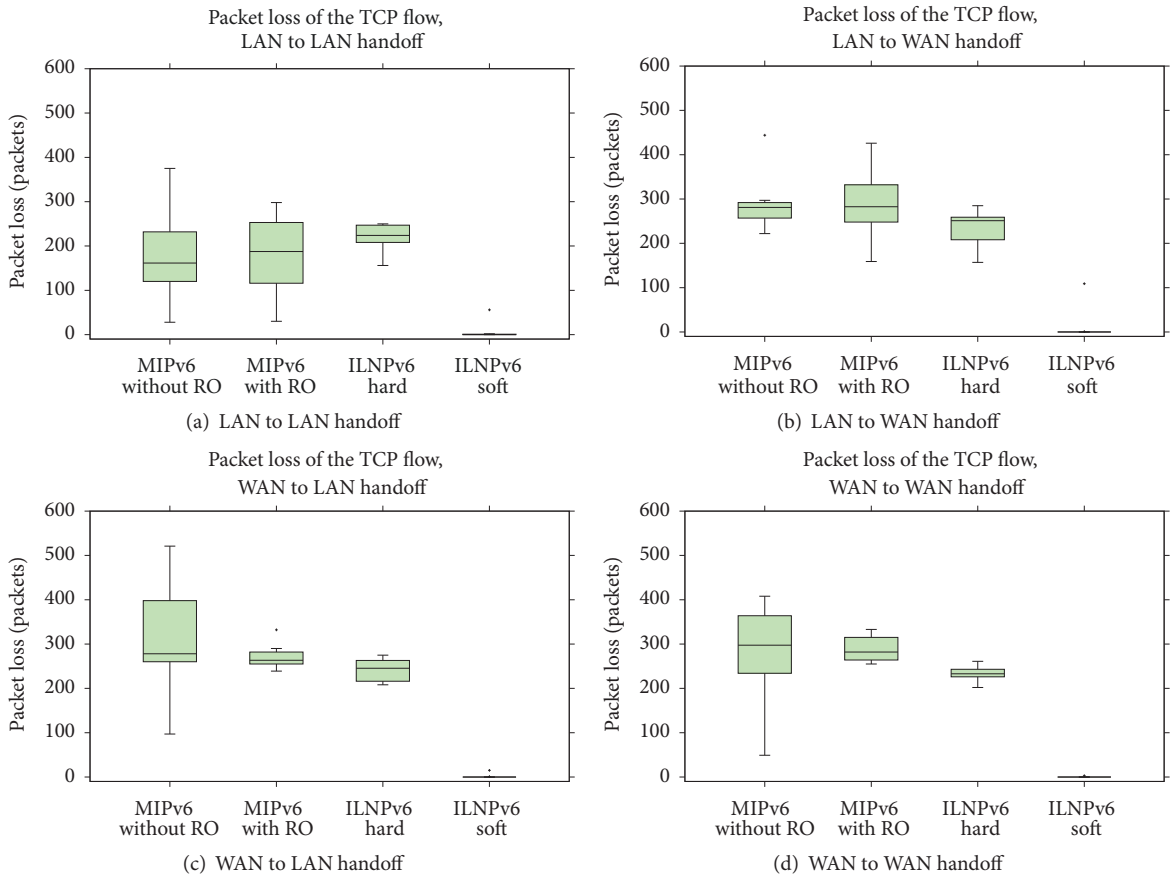

Figure 15: Number of lost packets in each flow. ILNPv6 with soft-handoff has the lowest number of packet loss and is close to zero.

since a number of retransmissions happen not because of loss, but due to 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.

6.1.3. Packet Loss. As previously discussed, the number of retransmissions was not always caused by packet loss. To measure the actual number of lost packets, 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 had 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 lost packets.

Figure 15 shows the number of lost packets in each handoff scenario with MIPv6 and ILNPv6. Apart from ILNPv6 soft-handoff, all results are similar to the retransmission observations (Figure 14). This means that retransmissions in MIPv6, both with and without RO, and ILNPv6 hard-handoff, were all 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 lost packets 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, ILNPv6 had much lower interruption times in the flow during handoff (see Section 6.1). 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 of 320 Kbytes, and each TCP packet consisting of a 1400 byte payload (according to the *tcpdump* log), the number of packets that can be sent without receiving an acknowledgement is $(320 \times 1024)/1400 \approx 234$ packets, which is close to the median values of packet loss for MIPv6 and ILNPv6 hard-handoff in Figure 15. 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.

6.1.4. Handoff Delay. Figure 16 presents handoff delay for TCP flows. 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

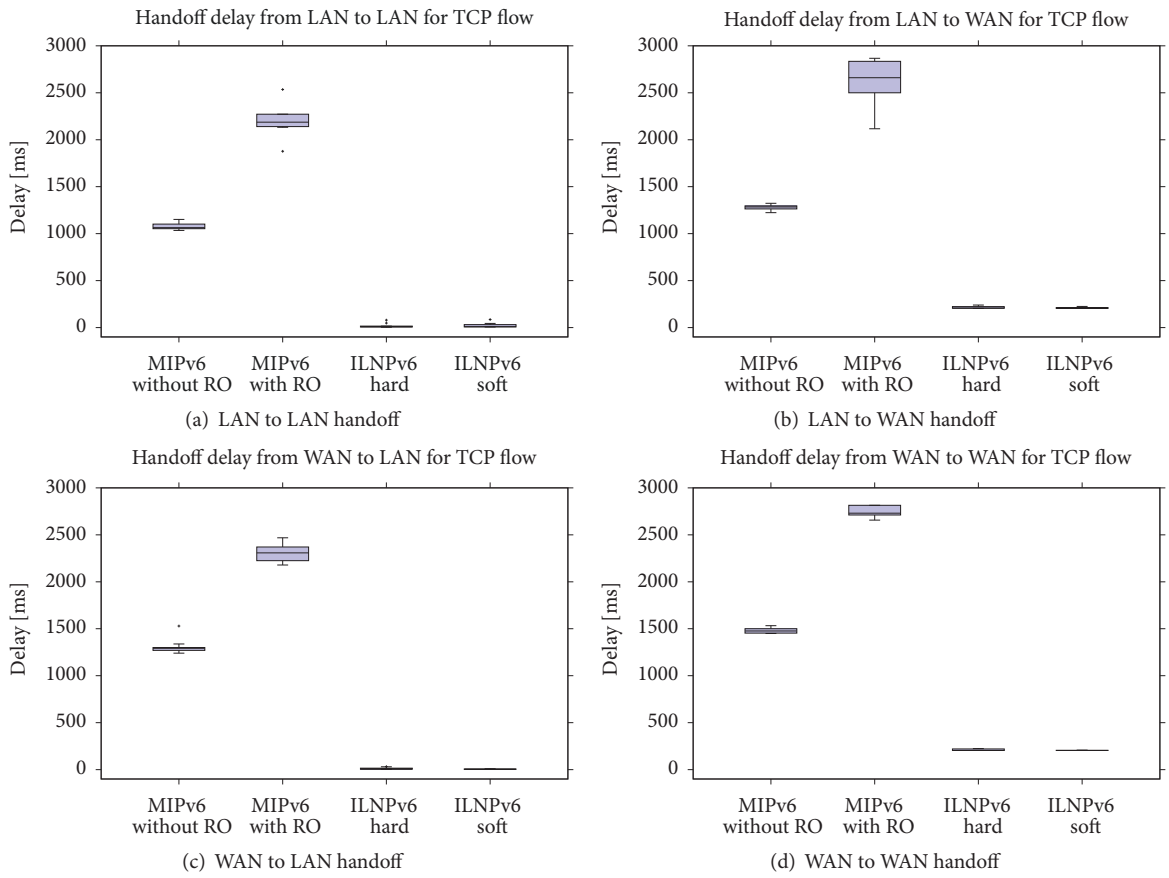

Figure 16: Handoff delay for TCP traffic flows. ILNPv6 handoff was ~1 RTT.

delays such as wireless association, Neighbour Discovery, and Duplicate Address Detection (DAD).

As expected, ILNPv6 had a much shorter handoff delay than MIPv6 in every case, and the value was close to a single RTT, i.e., close to ideal. MIPv6 had an additional delay of ~1s, which came from the BU processing and tunnel creation at the HA before the BAck was sent back to the MN. Longer delay was found if RO was enabled due to additional processing: the return routability test and the Binding Update handshake with the CN. The process took longer over the WAN path due to the higher end-to-end delay. Handoff delay of MIPv6 also had higher variance than ILNPv6 because of extra scope for delay variation from the processing time at the HA and the RO process, compared to ILNPv6 which required just the LU/LU-Ack handshake of a single RTT.

6.2. TCP Variants. As previously shown, TCP performance over ILNPv6 still has room to improve. The key finding above is that TCP must be able to deal with packet misordering that is caused by the multipath effect during ILNPv6 soft-handoff, as shown in Section 6.1.2.

We hence repeat the experiments using the same environment and configuration as described in Section 5.1, but instead of using the default setting of TCP congestion control algorithm in Linux (i.e., TCP CUBIC [66]), we examine the use of different TCP variants, i.e., TCP Hybla [3] and TCP Veno [4]. TCP Hybla is designed for heterogeneous networks, and TCP Veno is optimised for working in wireless networks. Our rationale is that our testbed set-up contains heterogeneous paths and paths with a wireless link. With TCP Hybla and TCP Veno variants, we might, naively, expect to see a better performance (e.g., fewer retransmissions) for both ILNPv6 and MIPv6, compared to TCP CUBIC.

6.2.1. Successfully Transferred Data. Figure 17 shows the total data transfer volumes (in MB) during the 30 second TCP flow using ILNPv6 and MIPv6 under different handoff scenarios using different TCP variants. Overall, there were no significant differences for different TCP variants, especially for LAN to LAN handoff, where the delay was low. The results had a similar trend as explained in Section 6.1.1. TCP Hybla gave slightly better results (higher data volumes can be transferred) in most cases, especially for WAN to LAN, and WAN to WAN handoff. This is because Hybla is designed to deal with high latency networks, such as satellite links, in heterogeneous networks. TCP Veno, surprisingly, gave the poorest performance in almost every case, observing the lowest amount of data transfer. TCP Veno adjusts to send less data where loss or error is detected to avoid unnecessary retransmissions. So, fewer retransmission attempts are observed, but less data is transmitted overall (see Section 6.2.2 for more details).

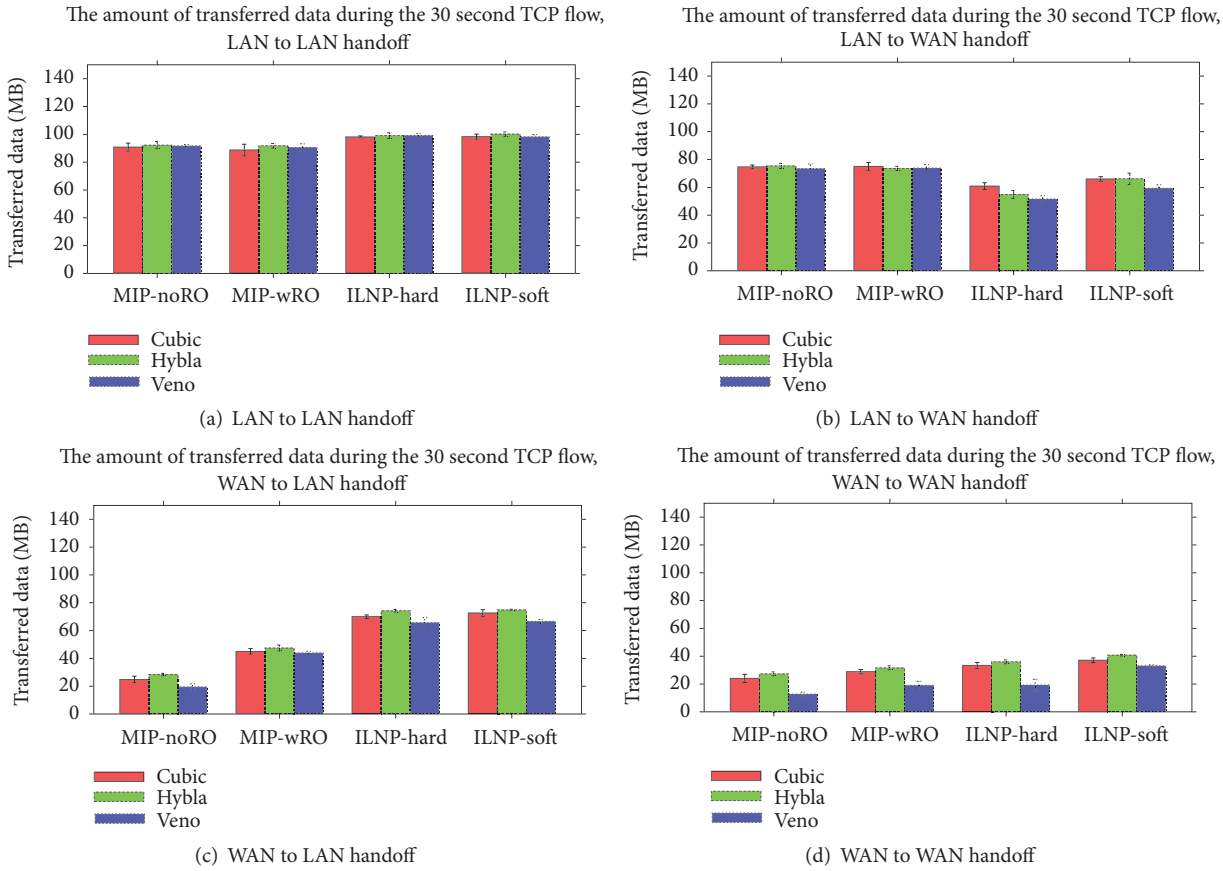

Figure 17: Total data transfer volumes during 30-second TCP traffic flows using different TCP variants.

6.2.2. Retransmissions Attempted. The number of TCP retransmissions in each handoff scenario using different TCP variants is shown in Figure 18. Again, this number was measured by counting the packets that were sent having the same TCP sequence number as those received at the CN.

TCP Hybla had similar results as the default TCP CUBIC in almost every case. A slightly lower number of retransmissions can be seen for ILNPv6 hard-handoff in LAN to LAN, and LAN to WAN scenarios. MIPv6 without RO produces the highest variance.

TCP Veno has a lower number of retransmission attempts in most cases, especially for MIPv6 with RO and ILNPv6 with hard-handoff. As mentioned above, the TCP Veno back-off algorithm is more sensitive to loss and error than the default TCP CUBIC, so fewer data retransmission attempts were observed, at the cost of reduced overall data transmission.

6.2.3. Packet Loss. The number of lost packets in each handoff scenario using different TCP variants is shown in Figure 19. In line with the default TCP CUBIC results in Section 6.1.3, apart from ILNPv6 with soft-handoff, all results are similar to the retransmission results (Figure 18). TCP Hybla had approximately the same level of loss as TCP CUBIC, while TCP Veno had slightly lower loss observed. For ILNP soft-handoff, packet loss was almost zero in every case for every TCP variant.

6.2.4. Summary of ILNPv6 Behaviour with TCP Variants. In summary, although some TCP congestion control algorithms, like TCP Hybla and TCP Veno, are customised for wireless and heterogeneous networks, we have observed that they have relatively small benefits in our testbed scenarios, especially for ILNPv6 with soft-handoff. TCP Hybla is useful for high-delay environments since more data can be sent (see Section 6.2.1). TCP Veno could help reduce packet loss and unnecessary TCP retransmissions (see Sections 6.2.2 and 6.2.3), but not by a significant amount. The Hybla and Veno variants were designed to consider reduced throughput and packet loss (e.g., due to TCP back-off behaviour or transmission errors). However, the use of ILNP soft-handoff changes the operating environment for mobility and the assumptions made by such variants, as it virtually removes gratuitous loss. While reducing packet loss, which is very beneficial to TCP, ILNPv6 soft-handoff introduces multipath effects (such as misordering and high variance in delay), which TCP congestion control protocols were not designed to deal with.

7. Discussion

This paper only investigates TCP performance over ILNPv6 to enable mobility support over the Internet in a testbed environment. This section presents some discussion and critical

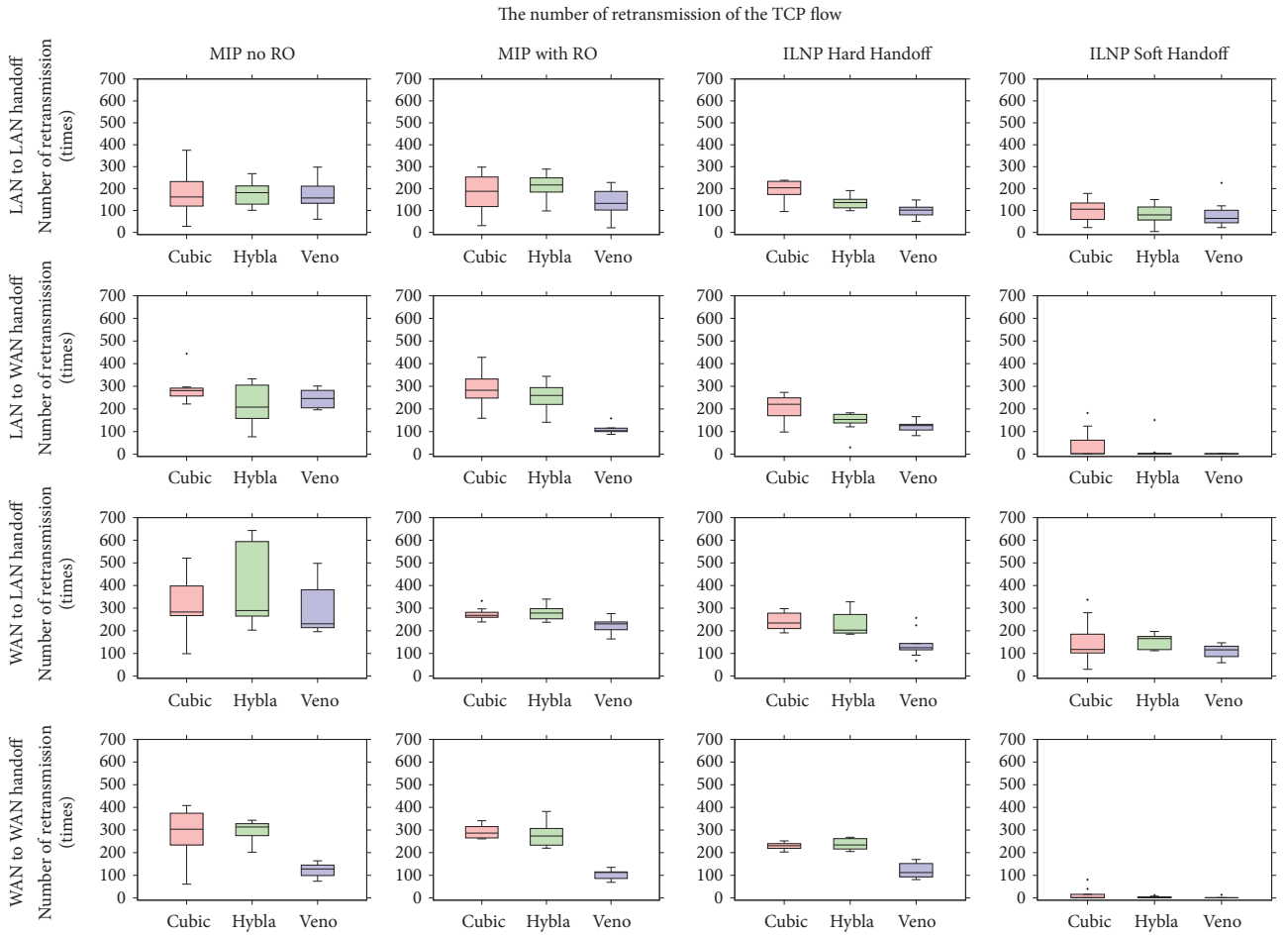

Figure 18: Number of retransmission attempts at the CN, using different TCP variants. ILNPv6 with soft-handoff had the lowest number of retransmissions in every case.

analyses relating to use of ILNPv6 in terms of operation and deployment; comparisons with IPv6 and/or Mobile IPv6 are made where appropriate.

7.1. TCP Optimisation. Compared to our UDP results from previous work [5], ILNPv6 provides improved performance for mobility compared to MIPv6. However, TCP performance for host mobility still has room for improvement. The use of ILNPv6 with soft-handoff changes the design space for TCP congestion control: packet loss is greatly reduced, but multipath effects must be considered. Nevertheless, in our experiments with ILNPv6, TCP handoff still performed better than with MIPv6: (i) with ILNPv6 soft-handoff, there was no interruption of the flows for any scenario (Section 6.1); (ii) more data was transferred in the same timeframe, in most cases (Section 6.1.1); (iii) ILNPv6 soft-handoff had the lowest number of retransmission attempts (Section 6.1.2) with almost zero packet loss (Section 6.1.3); and (iv) handoff delay was much shorter with ILNPv6 (Section 6.1.4).

To maximise TCP performance, additional tuning is required. The key finding here is that TCP must be able to deal with packet misordering that is caused by the multipath effect during ILNPv6 soft-handoff, as shown in Section 6.1.2. Of course, this would also be the foundation of an alternative mechanism to achieve multipath transfer using TCP in multihoming scenarios, for example, an ILNPv6 alternative to MP-TCP [54, 55]. The initial work on ILNPv6 multihoming shows satisfactory performance in the network layer [67]. However, investigations of using TCP with ILNPv6 multihoming and within a mobile scenario are a subject for further study.

There are also other possible optimisations to support TCP operating in mobile environments over ILNPv6 or MIPv6. Firstly, the queuing algorithm at access routers and the MN could be optimised to prioritise some ICMP packets such as RA and handoff signals (BU/BAck for MIPv6, LU/LU-ACK for ILNPv6), so that such packets do not get blocked or delayed by intense TCP traffic (see Section 5.1). Secondly, the ILNPv6 handoff decision algorithm could be improved, allowing an MN to give preference to a better quality link during handoff (see Section 6.1.1). These two mechanisms could also benefit MIPv6.

Finally, a new TCP congestion control algorithm could be investigated that could better exploit the modified end-to-end path characteristics that are observed with ILNPv6. The current TCP variants designed for wireless and heterogeneous networks, e.g., TCP Hybla and TCP Veno, do

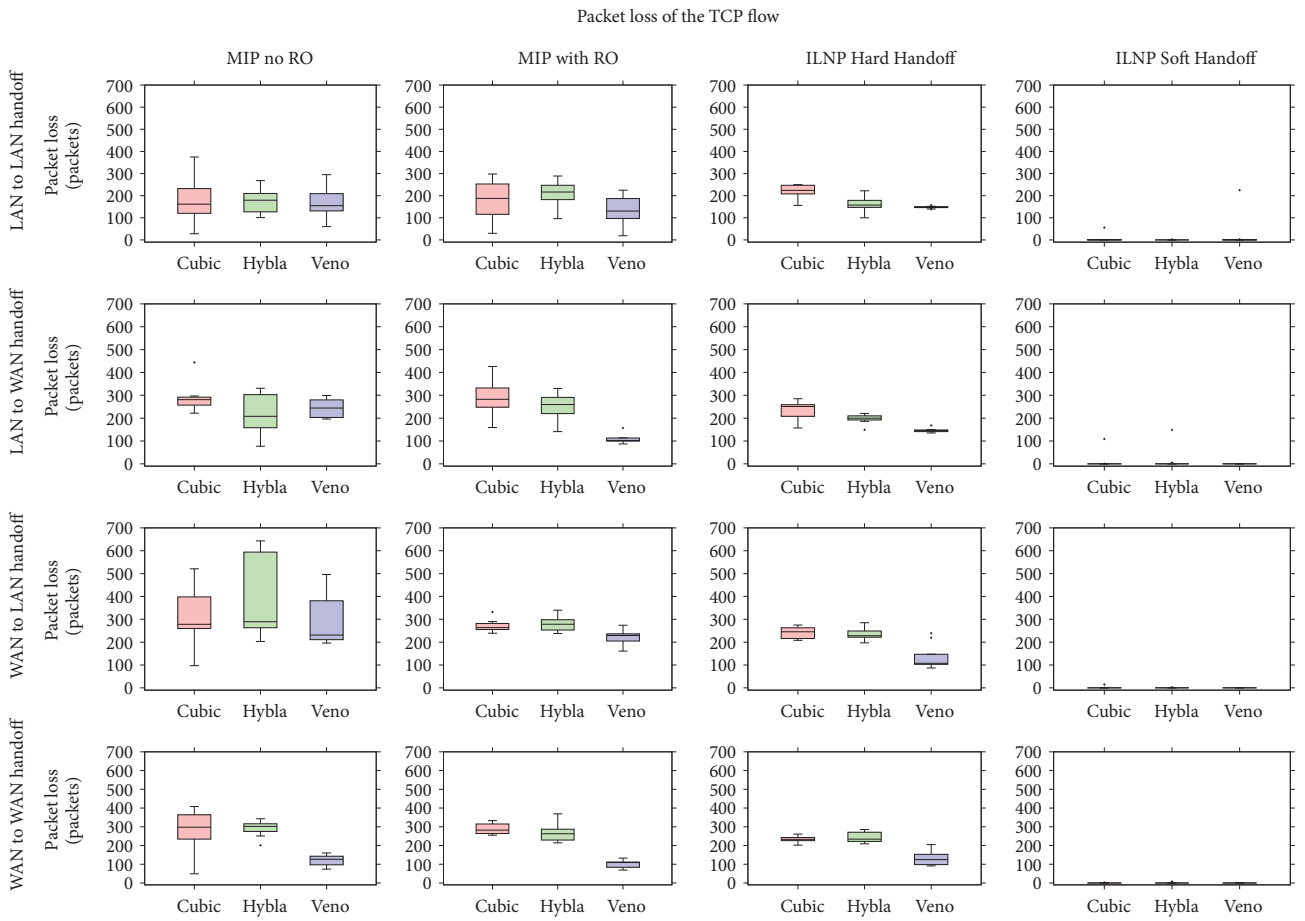

Figure 19: Number of lost packets in each flow, using different TCP variants. ILNPv6 with soft-handoff had the lowest level of packet loss and was close to zero.

not provide optimal TCP performance when using ILNPv6 with soft-handoff (see Section 6.2). Most TCP enhancements for mobile and wireless environments have been designed for hard-handoff solutions, to deal with loss, errors, and disconnections [68]. Our results here change the landscape of the problem space: for TCP congestion control in the context of mobility, TCP needs to deal with misordering due to multipath effects as well as potential errors and loss, but the latter two may have a lower impact when used with ILNPv6 soft-handoff compared to MIPv6.

7.2. Name Resolution Using DNS. This paper focuses on investigating handoff performance of ILNPv6. For name resolution; i.e., when a CN initiates a communication with an MN, the CN must learn the current NID and L64 values of the MN. ILNPv6 could leverage the current DNS to provide name resolution. New DNS Resource Records for NID and L64 values have been defined [11] and are implemented in widely used DNS software: ISC BIND/*named* from v9.9.3 (https://kb.isc.org/article/AA-00970/81/BIND-9.9.3-P1-Extended-Support-Version-Release-Notes.html), NSD/*Unbound* from v3.2.15 (http://www.nlnetlabs.nl/svn/nsd/branches/NSD_3_2/doc/ChangeLog), and Knot-DNS from v1.3.0 (https://gitlab.labs.nic.cz/labs/knot/raw/v2.0.0/NEWS).

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 time-to-live (TTL), or else cached L64 values would become stale. A previous emulation study [69] reports that using a low TTL value of hundreds of seconds should not impact significantly on DNS. Also, a previous empirical evaluation [27] 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 ILNPv6 hosts should 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 ILNPv6 and is widely deployed also. As DNS is already deployed worldwide, ILNPv6 does not incur any additional overhead in this regard, 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. In this respect, mobile IP should have no impact on DNS at all.

For ILNPv6, DNS is not essential in all contexts. For example, if ILNPv6 was used for a future wide-area mobile network, such as 5G or beyond, then a specific address

resolution service for mapping International Mobile Subscription Identifiers (IMSI)/International Mobile Equipment Identifiers (IMEI)/E.164 addresses to NID/L64 values might be used. This need not be DNS. Also, some applications already have their own, application-specific presence and/or rendezvous mechanisms.

7.3. Backwards Compatibility with IPv6 and Incremental Deployment. 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. Our performance evaluation shows that ILNPv6 works well over unmodified IPv6 Linux routers and unmodified Ethernet switches. Also, it can support applications that use the existing C socket APIs. However, `getaddrinfo` (3) `/libc` needs to be updated to interpret NID and L64 values [5, 6, 35]. In addition, some applications that use IP addresses at the application level will need reengineering to use the FQDN or only the NID, or application-specific names for configuration.

As there are no changes required to the current network infrastructure and routing scheme, the solutions could be deployed in the current IPv6 network without requiring additional entities or any modifications to the core network infrastructure. The distribution of kernel updates can be implemented as part of the normal system software update cycle that is common for desktop, server, and mobile operating systems.

When compared to MIPv6, ILNPv6 supports mobility directly at the end host. MIPv6 requires at least the home agent (HA) entity for normal operation.

In terms of practical deployment, as we have shown implementation of ILNPv6 is possible in Linux, then this, potentially, offers an easy path for implementation within the Android kernel (https://www.android.com), which is based on Linux. In Section 4, we presented a detailed description of the modifications needed in the packet processing paths, which offers a pattern for modifying other operating systems. Also, as our implementation of ILNPv6 is realised as a superset of IPv6, then it should be possible to modify any other IPv6 code to support ILNPv6.

7.4. Network Address Translation (NAT) Equivalent for ILNPv6. Even though IPv6 offers a much more plentiful supply of addresses, the use of network address translation (NAT) with IPv6 remains in wide use. One popular use of NAT is for the protection of the 'Local Network', but even here, the use of NAT is strongly discouraged [70].

The *Localised Numbering* function within ILNPv6 [17, Section 2] provides an alternative method to NAT using a *locator rewriting relay (LRR)* mechanism. A LRR function only makes changes to L64 values instead of the whole IP address, as is the case for IP. Therefore, performing the equivalent of NAT on L64 will not break ongoing communications, since the NID values remain stable. A site border router (SBR) which is ILNP-capable could map and rewrite L64 values to provide NAT-like functionality. This mechanism also enables an alternative method for traffic engineering, and provides location privacy protection (see [17] for more details).

Most importantly for our discussion, LRR works in complete harmony with mobility for ILNPv6, as the LRR function is implemented at the SBR, e.g., the access router for a radio cell. MIPv6 needs special handling for operation with NATs [71, Sections 3 and 4].

7.5. Firewalls. We have shown from our testbed that existing routers should see ILNPv6 packets as IPv6 packets, so too should firewalls. Hence, firewalls should not have difficulties in ILNPv6 operations, unlike for MIPv6 where tunnels and address changes in packets may cause problems. Specific codepoints for the new ICMPv6 Locator Update message [12] and the Nonce Destination Option [13] for ILNPv6 have been allocated by IANA (http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml and http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml). This helps in the creation and management of firewall rules without treating ILNPv6 as a special case.

However, MIPv6 requires special handling, with additional administrative considerations and configuration for operation with firewalls [72].

7.6. Simultaneous Mobility. Although this paper mainly focuses on a one-side mobile host scenario, ILNPv6 could also be used for supporting two communicating MNs. When two mobile hosts communicate, it is possible that both hosts hand off simultaneously, sometimes called the *Double Jump* problem.

For MIPv6, both hosts could still communicate via the HAs of both MNs since the HAs never move. However, there are problems in the RO process because the Return Routability Procedure and the Binding Update to another endpoint might never reach the destination [73]. So, the RO process would fail, and MNs would need to communicate through the HA, which would adversely impact performance. Some solutions have been proposed, e.g., [74].

For ILNP, simultaneous mobility can be handled as depicted in Figure 20. ILNP can leverage the soft-handoff mechanism to overcome the Double Jump problem. Although both MNs 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 (because, say, there is no overlap area between networks/cells), the LU could be sent to the previous location, never reaching the MNs, and the handoff could fail. Nevertheless, the communication sessions can be reestablished by consulting the DNS, after the L64 records have been updated by the moving MNs, or by the use of an application-specific rendezvous mechanism.

7.7. Applicability to Vertical Handoff. Vertical handoff is a scenario when a 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 features: (i) enabling architecture; (ii) decision

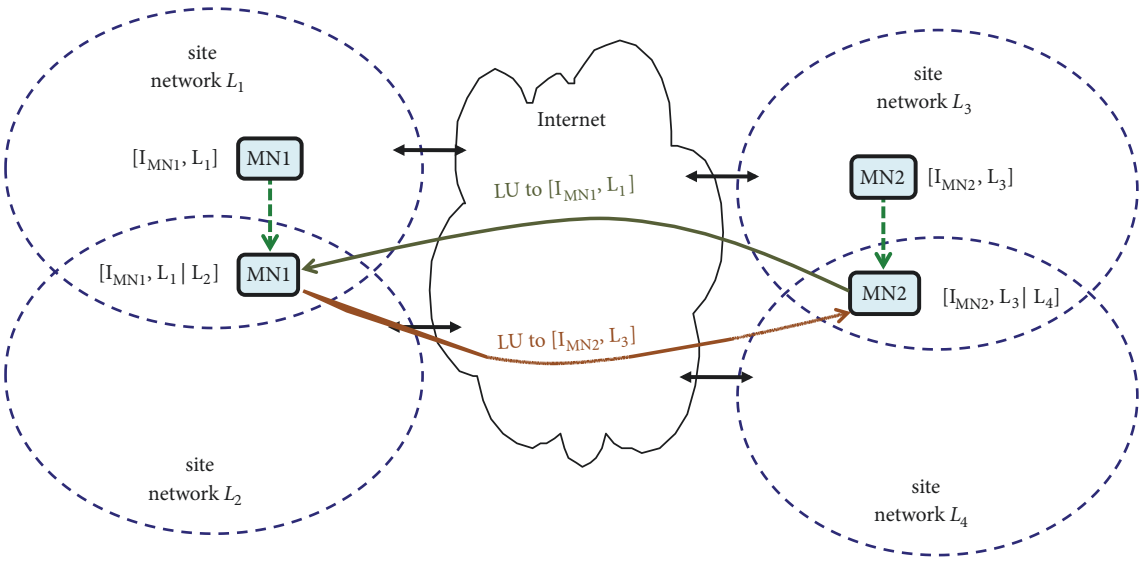

Figure 20: Simultaneous mobility using ILNP soft-handoff. The LU signal can reach both end hosts because they maintain bindings to the previous L64 value.

metrics and policy design; and (iii) radio link transfer design [75]. Based on the results of this work, we propose that ILNP is an example of an architecture that could enable seamless mobility in the IP layer, i.e., it presents a solution for the first feature, that of an enabling architecture. The other two key features are out of scope for ILNP, as well as for mobile IP, as they are concerned with the signals, triggers, and characteristics of/to/from the lower-level (sub)network technologies.

The decision metric determines if the MN needs to handoff and which network should be chosen. The metric may derive information from different factors, such as service type, network conditions, and user preferences. After the destination network is chosen, a handoff policy is used to determine when the handoff occurs. The decision algorithm could be *function-based* considering signal strength, capacity and cost of the overlapping networks. It could also be a *user centric decision* algorithm, which allows a user to specify his/her preference [30].

The radio link transfer is concerned with performance of link switching, e.g., minimising delay and retaining QoS of the ongoing flows, since different wireless technologies provide different network conditions, e.g., different bandwidth. More discussion on QoS is provided in Section 7.8.

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 ILNPv6 and any lower-layer handoff mechanisms. Investigations of using ILNPv6 in various mobile devices to observe effects of specific physical-layer and link-layer handoff will be required before deployment. However, as ILNPv6 enables efficient network layer handoff, the network layer is no longer a bottleneck in the mobility process.

7.8. Managing Changes in QoS on Handoff. While ILNPv6 deals with handoff, with network layer soft-handoff helping to reduce packet loss, there are still other problems to solve at the transport layer or application layer. For example, neither ILNP nor mobile IP guarantees the *quality of service (QoS)* or *quality of experience (QoE)* for 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. We have presented an example of the impact of such handoff for TCP, e.g., data rates can drop by an order of magnitude when moving from a low delay network to a high-delay network, because of the way TCP works. 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.

Nevertheless, our results above show that ILNPv6 with soft-handoff can significantly reduce flow disruption, with packet loss close to zero, and no significant drop in end-to-end throughput. At the time of writing, the most recent results that were available for LISP-MN showed that even with improvements to the basic LISP-MN architecture, TCP flows can suffer from throughput reduced to zero for several seconds during handoff [76].

7.9. Multihoming and Concurrent Multipath Transfer. In ILNP, multihoming and mobility form a duality: host multihoming is similar to host mobility, and site multihoming is the same as site mobility. When either an individual host or the whole site changes its point of attachment(s) to the network(s), ILNP treats this the same as when an MN changes

its location, i.e., the L64 value(s) is(are) changed, LU messages are sent, and DNS records are updated, if necessary [67].

In Figure 2, 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, for example, a multihoming resilience capability, or a "WiFi-offloading" facility.

From Table 2, the feature that only ILNP can provide as a mobility solution is transparent, concurrent, multipath transfer capability at the network layer. This is another advantage of ILNP over other solutions. As mentioned above, ILNPv6 allows an MN to have multiple L64 values with multiple active interfaces at the same time. This could enable multipath transfer using 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 [55]. The transport-layer tuple is not affected by multipath delivery over multiple interfaces: end-to-end state is preserved. Of course, some existing mechanisms, such as those that deal with packet reordering and congestion control, would need to be tuned, as discussed above.

7.10. Privacy and Security. Although security in Internet protocols has been a key consideration for some time [77], the recommendation for the need to consider privacy issues came a decade later [78]. Nevertheless, ILNP has strong support for both packet-level security and flexible mechanisms for privacy, from its initial design.

For mobile systems, privacy is a key concern, as a mobile user can be potentially tracked by a number of important data values that are bound to the mobile device, e.g., an IP address value, or a NID value.

ILNPv6 supports packet-level security by use of an extension header that carries the Nonce Destination Option [13], and the also allows the use of IPsec as for IPv6 [79]. The ILNPv6 header can carry a 96-bit nonce value which protects the packet flow from 'off-path' attacks, such as address-spoofing attacks. The nonce value is set up when an ILNPv6 session is initiated, and then checked in subsequent received packets. Where the threat regime requires protection against 'on-path' attacks (e.g., by packet snooping, or man-in-the-middle attacks), then ILNP can use a modified version of IPsec, including full packet encryption if required. The main update required to IPsec for use with ILNPv6 is that the IPsec Security Association (SA) should use an ILNPv6 NID value, rather than an 128-bit IPv6 address [10, Section 9]. ILNPv6 can then allow more advanced security functions, such as secure failover in site connectivity [80], and secure, wide-area virtual machine mobility [81].

IPv6 also supports identity privacy [7, Section 10] and location privacy [17, Section 7], outside of IPsec. Identity privacy is supported by allowing the use of ephemeral NID values, which can be created as for IPv6 identifier values, e.g., using the privacy extensions for stateless address autoconfiguration (SLAAC) [82], or the more recent recommended updates for generating IPv6 ID values [83].

For location privacy, ILNPv6 can allow the L64 value in a packet to be changed, *en route*, without violating end-to-end consistency, as the end-state for transport protocols is bound only to the NID value. A *locator rewriting* function, for example, a *locator rewriting relay (LRR)* [17, Section 7], can change the L64 value in the packet, and so prevent the receiver of the packet from knowing the real network location of the sender of the packet [84]. As the mobility mechanism for ILNPv6 also requires changes to L64 values, the locator rewriting function acts as a L64 mapping mechanism, and would need to monitor and act upon Locator Update messages from a MN in order to keep L64 mappings up to date. An LRR function could reside at the mobile node, or at a router, e.g., an access router. Also, multiple LRR functions along a path can be used [84], again without impacting end-to-end integrity of the flow.

8. Conclusion and Future Work

We have shown that mobility management for IP mobile nodes can be implemented as a true end-to-end function in the Linux OS kernel by using ILNPv6, our superset of IPv6, using a new naming architecture. Existing kernel TCP code and IP code can be enhanced to allow legacy IPv6 applications to operate without modification over ILNPv6. Additionally, ILNPv6 packets can be transported across existing IPv6-capable infrastructure (switches and routers).

Our performance evaluation on a lab testbed showed that ILNPv6 outperformed IPv6 in every case in terms of flow continuity, packet loss, and handoff delay, especially when using network layer *soft-handoff* in ILNPv6. However, there were some surprising results. Overall, TCP is much more sensitive to packet loss and round-trip-time variations than UDP, causing handoff performance to be less than optimal for both MIPv6 and ILNPv6. Although TCP performed the best when ILNPv6 with soft-handoff was used, some engineering improvements would be needed to TCP in the future for optimal performance over ILNPv6. A new TCP congestion control algorithm that deals with the multipath effects of soft-handoff is desirable. However, even without such an enhancement, existing TCP variants always performed better over ILNPv6 compared to IPv6 in our experiments.

For the future, we aim to investigate: congestion control for ILNPv6; the impact on various kinds of application (e.g., real-time video) and network scenarios to examine users' quality of experience; look at combined functionality such as mobility and multihoming; examine the use of ILNPv6 for mobility scenarios that use satellite systems; and also examine the control plane issues for future ILNPv6 application scenarios such as 5G systems and beyond.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

There are no conflicts of interest to declare.

Acknowledgments

The contents of this paper are based on work conducted while Dr. Phoomikiattisak was a Ph.D. student at the University of St Andrews, under the supervision of Professor Bhatti. This work was partially funded by the Government of Thailand through a Ph.D. scholarship for Dr. Phoomikiattisak. Some of the background and results are documented in his Ph.D. thesis, which is available from the University of St Andrews [85].

References

[1] C. Perkins, "IP Mobility Support for IPv4," RFC 5944 (PS), IETF, 2010.

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

[3] C. Caini and R. Firrincieli, "TCP Hybla: A TCP enhancement for heterogeneous networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, 2004.

[4] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, 2003.

[5] D. Phoomikiattisak and S. N. Bhatti, "Mobility as a first class function," in *Proceedings of the 11th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2015*, pp. 850–859, UAE, October 2015.

[6] S. N. Bhatti, D. Phoomikiattisak, and B. Simpson, "IP without IP addresses," in *Proceedings of the the 12th Asian Internet Engineering Conference*, pp. 41–48, Bangkok, Thailand, November 2016.

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

[8] R. Atkinson, S. Bhatti, and S. Hailes, "Evolving the internet architecture through naming," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1319–1325, 2010.

[9] R. Atkinson, S. Bhatti, and S. Hailes, "ILNP: Mobility, multihoming, localised addressing and security through naming," *Telecommunication Systems*, vol. 42, no. 3-4, pp. 273–291, 2009.

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

[11] R. Atkinson, S. N. Bhatti, and S. Rose, "DNS Resource Records for the Identifier-Locator Network Protocol 935 (ILNP)," RFC 6742 (E), IRTF, 2012.

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

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

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

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

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

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

[18] C. J. Bennett, S. W. Edge, and A. J. Hinchley, "Issues in the interconnection of datagram networks," in *IEN*, vol. 1, University College London (UCL), Jul 1977.

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

[20] B. Carpenter, J. Crowcroft, and Y. Rekhter, "IPv4 Address Behaviour Today," RFC 2101 (I), IAB, 1997.

[21] B. E. Carpenter, "IP addresses considered harmful," *Computer Communication Review*, vol. 44, no. 2, pp. 65–69, 2014.

[22] J. Pan, R. Jain, S. Paul, and C. So-In, "MILSA: A new evolutionary architecture for scalability, mobility, and multihoming in the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1344–1362, 2010.

[23] A. Rodríguez Natal, L. Jakab, M. Portolés et al., "LISP-MN: mobile networking through LISP," *Wireless Personal Communications*, vol. 70, no. 1, pp. 253–266, 2013.

[24] M. Menth, M. Hartmann, and D. Klein, "Global locator, local locator, and identifier split (GLI-Split)," *Future Internet*, vol. 5, no. 1, pp. 67–94, 2013.

[25] S. Schütz, H. Abrahamsson, B. Ahlgren, and M. Brunner, "Design and implementation of the Node Identity Internetworking Architecture," *Computer Networks*, vol. 54, no. 7, pp. 1142–1154, 2010.

[26] T. Li, "Recommendation for a Routing Architecture," RFC 6115 (I), IETF, 2011.

[27] S. N. Bhatti and R. Atkinson, "Reducing DNS caching," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*, pp. 792–797, China, April 2011.

[28] S. A. Baset and H. G. Schulzrinne, "An analysis of the Skype peer-to-peer internet telephony protocol," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–11, Barcelona, Spain, April 2006.

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

[30] M. Zekri, B. Jouaber, and D. Zeghlache, "A review on mobility management and vertical handover solutions over heterogeneous wireless networks," *Computer Communications*, vol. 35, no. 17, pp. 2055–2068, 2012.

[31] A. Gladisch, R. Daher, and D. Tavangarian, "Survey on mobility and multihoming in future internet," *Wireless Personal Communications*, vol. 74, no. 1, pp. 45–51, 2014.

[32] R. Atkinson, S. Bhatti, and S. Hailes, "A proposal for unifying mobility with multi-homing, NAT and security," in *Proceedings of the 5th ACM International Workshop on Mobility Management and Wireless Access, MobiWac'07*, pp. 74–83, Greece, October 2007.

[33] R. Atkinson, S. Bhatti, and S. Hailes, "Mobility as an integrated service through the use of naming," in *Proceedings of the 2nd ACM/IEEE Intl. Workshop on Mobility in the Evolving Internet Architecture*, pp. 1:1–1:6, Kyoto, Japan, August 2007.

[34] D. Phoomikiattisak and S. N. Bhatti, "Network layer soft handoff for IP mobility," in *Proceedings of the 8th ACM Wrkshp. Perf. Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, pp. 13–20, Barcelona, Spain, November 2013.

[35] D. Phoomikiattisak and S. N. Bhatti, "IP-layer soft handoff implementation in ILNP," in *Proceedings of the 9th ACM MobiCom Workshop on Mobility in the Evolving Internet Architecture, MobiArch 2014*, pp. 1–6, USA, September 2014.

[36] D. Phoomikiattisak and S. N. Bhatti, "Control plane handoff analysis for IP mobility," in *Proceedings of the 9th IFIP Wireless and Mobile Networking Conference, WMNC 2016*, pp. 65–72, France, July 2016.

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

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

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

[40] H. Yokota, K. Chowdhury, R. Koodli, B. Patil, and F. Xia, "Fast Handovers for Proxy Mobile IPv6," RFC 5949 995 (PS), IETF, 2010.

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

[42] D. Liu, J. C. Zuniga, P. Seite et al., "Distributed Mobility Management: Current Practices and Gap Analysis," RFC 7429 (I), IETF, 2015.

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

[44] A. Galvani, A. Rodriguez-Natal, A. Cabellos-Aparicio, and F. Risso, "LISP-ROAM: Networkbased Host Mobility with LISP," in *Proceedings of the 9th ACM MobiCom Workshop on Mobility in the Evolving Internet Architecture, MobiArch 2014*, pp. 19–24, USA, September 2014.

[45] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)," RFC 7401 (PS), 1005, IETF, 2015.

[46] T. Henderson, C. Vogt, and J. Arkko, "Host Mobility with the Host Identity Protocol," RFC 8046 (PS), IETF, 2017.

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

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

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

[50] A. Dhraief and N. Montavont, "Toward mobility and multihoming unification - the SHIM6 protocol: a case study," in *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC 2008*, pp. 2840–2845, USA, April 2008.

[51] M. M. Feroz and A. K. Kiani, "SHIM6 Assisted Mobility Scheme, an intelligent approach," in *Proceedings of the 2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013*, pp. 725–728, USA, January 2013.

[52] A. Achour, B. Kervella, and G. Pujolle, "SHIM6-based mobility management for multi-homed terminals in heterogeneous environment," in *Proceedings of the 2011 Eighth International Conference on Wireless and Optical Communications Networks - (WOCN)*, pp. 1–5, Paris, France, May 2011.

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

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

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

[56] J. Rosenberg, H. Schulzrinne, G. Camarillo et al., "SIP: Session Initiation Protocol," RFC 3261 (PS), IETF, 2002.

[57] S. Sevilla and J. J. Garcia-Luna-Aceves, "A deployable identifier-locator split architecture," in *Proceedings of the 2017 IFIP Networking Conference and Workshops, IFIP Networking 2017*, pp. 1–9, Sweden, June 2017.

[58] A. Dutta, F. Vakil, J.-C. Chen et al., "Application layer mobility management scheme for wireless Internet," in *Proceedings of the Proceedings - 2001 International Conference on Third Generation Wireless and Beyond*, pp. 379–385, USA, July 2001.

[59] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley, "Opportunistic mobility with multipath TCP," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys'11 and Co-located Workshops - 6th MobiArch Workshop, MobiArch'11*, pp. 7–12, USA, July 2011.

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

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

[62] M. Z. Shafiq, F. Le, M. Srivatsa, and A. X. Liu, "Cross-path inference attacks on multipath 1040 TCP," in *Proceedings of the 12th ACM Workshop on Hot Topics in Networks, HotNets 2013*, pp. 15:7–15:1, ACM, November 2013.

[63] M. Komu, M. Bagnulo, K. Slavov, and S. Sugimoto, "Sockets Application Program Interface (API) forMultihoming Shim," RFC 6316 (I), IETF, 2011.

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

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

[66] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly High-speed TCP variant," *SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[67] B. Simpson and S. N. Bhatti, "An identifier-locator approach to host multihoming," in *Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications, IEEE AINA 2014*, pp. 139–147, Canada, May 2014.

[68] H. Elaarag, "Improving TCP performance over mobile networks," *ACM Computing Surveys*, vol. 34, no. 3, pp. 357–374, 2002.

[69] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 589–603, 2002.

[70] G. Van, T. Hain, R. Droms, B. Carpenter, and E. Klein, "Local Network Protection for IPv6," RFC 4864 1055 (I), IETF, 2007.

[71] H. Soliman, "Mobile IPv6 Support for Dual Stack Hosts and Routers," RFC 5555 (PS), IETF, 2009.

[72] F. Le, S. Faccin, B. Patil, and H. Tschofenig, "Mobile IPv6 and Firewalls: Problem Statement," RFC 4487 (I), IETF, 2006.

[73] K. D. Wong, A. Dutta, H. Schulzrinne, and K. Young, "Simultaneous mobility: Analytical framework, theorems and solutions," *Wireless Communications and Mobile Computing*, vol. 7, no. 5, pp. 623–642, 2007.

[74] L. Qiang, L. Shaomei, H. Hongyong, and W. Binqiang, "A multi-binding solution for simultaneous mobility of MIPv6," in *Proceedings of the 2nd IEEE International Symposium on Service-Oriented System Engineering, SOSE 2006*, pp. 143–146, China, October 2006.

[75] J. McNair and F. Zhu, "Vertical handoffs in fourth-generation multinetwork environments," *IEEE Wireless Communications Magazine*, vol. 11, no. 3, pp. 8–15, 2004.

[76] M. Isah, S. Simpson, and C. Edwards, "An improved LISP mobile node architecture," *Journal of Network and Computer Applications*, vol. 118, pp. 29–43, 2018.

[77] E. Rescorla and B. Korver, "Guidelines for Writing RFC Text on Security Considerations," RFC 3552 (BCP 72), IAB, 2003.

[78] A. Cooper, H. Tschofenig, B. Aboba et al., "Privacy Consideration1070 s for Internet Protocols," RFC 6973 (I), IETF, 2013.

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

[80] S. Bhatti, D. Phoomikiatissak, and R. Atkinson, "Secure & agile wide area virtual machine mobility," in *Proceedings of the 2012 IEEE Military Communications Conference*, Oct 2014.

[81] S. N. Bhatti and R. Atkinson, "Secure & agile wide area virtual machine mobility," in *Proceedings of the 2012 IEEE Military Communications Conference, MILCOM 2012*, USA, November 2012.

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

[83] F. Gont, A. Cooper, D. Thaler, and W. Liu, "Recommendation on stable IPv6 interface identifiers," RFC 8064 1080 (PS), IETF, 2017.

[84] S. N. Bhatti, R. Atkinson, and J. Klemets, "Integrating challenged networks," in *Proceedings of the 2011 IEEE Military Communications Conference, MILCOM 2011*, pp. 1926–1933, USA, November 2011.

[85] D. Phoomikiatissak, *Mobility as first class functionality : ILNPv6 in the Linux kernel [Ph.D. thesis]*, School of Computer Science, University of St Andrews, St, Andrewss, UK, 2016, http://hdl.handle.net/10023/7915.1085.