future internet

MDPI

Article

Mobility–Multihoming Duality

Ryo Yanagida [1,†] and Saleem Noel Bhatti [2,*,†]

[1] School of Computing Science, College of Science and Engineering, University of Glasgow, Glasgow G12 8RZ, UK; ryo.yanagida@glasgow.ac.uk
[2] School of Computer Science, University of St Andrews, St Andrews KY16 9SX, UK
* Correspondence: saleem@st-andrews.ac.uk
† These authors contributed equally to this work.

Abstract: In modern Internet-based communication, especially mobile systems, a mobile node (MN) will commonly have more than one possibility for Internet Protocol (IP) connectivity. For example, an MN such as a smartphone may be associated with an IEEE 802.11 network at a site while also connected to a cellular base station for 5G. In such a scenario, the smartphone might only be able to utilise the IEEE 802.11 network, not making use of the cellular connectivity simultaneously. Currently, IP does not allow applications and devices to easily utilise multiple IP connectivity opportunities—multihoming for the MN—without implementing special mechanisms to manage them. We demonstrate how the use of the Identifier Locator Network Protocol (ILNP), realised as an extension to IPv6, can enable mobility with multihoming using a duality mechanism that treats mobility and multihoming as the same logical concept. We present a network layer solution that does not require any modification to transport protocols, can be implemented using existing application programming interfaces (APIs), and can work for any application. We have evaluated our approach using an implementation in Linux and a testbed. The testbed consisted of commercial equipment to demonstrate that our approach can be used over existing network infrastructure requiring only normal unicast routing for IPv6.

Keywords: identifier locator network protocol (ILNP); internet protocol v6 (IPv6); multipath

Citation: Yanagida, R.; Bhatti, S.N. Mobility–Multihoming Duality. *Future Internet* **2024**, *16*, 358. https://doi.org/10.3390/fi16100358

Academic Editor: Ping Wang

Received: 20 June 2024
Revised: 16 September 2024
Accepted: 20 September 2024
Published: 1 October 2024

1. Introduction

A Future Internet would enable Ubiquitous Computing, allowing for flexible communication with any and all connectivity available, through multiple different network technologies. However, the addressing architecture of the current Internet Protocol (IP) does not enable this directly at the network layer [1].

Nevertheless, in the current Internet, mobile devices with multiple connectivity are becoming increasingly common for example smartphones and tablets. Multiple connectivity and mobility are enabled through retrospective enhancements to the IP architecture, but they do not integrate well. Currently, the user can suffer from disruption to communication on a host as the physical network connectivity changes. This can be to move between different network technologies (e.g., from 4G or 5G to IEEE 802.11/WiFi—also known as *vertical handover*). It can also be to enable and disable individual connectivity to different networks at the same time, e.g., the simultaneous use of 5G and IEEE 802.11/WiFi. This is because the IP address changes when the point of attachment (PoA) to a network changes, breaking the transport layer end-to-end state. Furthermore, while a device might be connected to multiple networks simultaneously, the use of IP addresses prevents a host from leveraging multiple network interfaces easily—true *host multihoming*—to improve its communication throughput or reliability. While solutions exist separately for mobility and multihoming, it is not possible to use them as a harmonised solution for a host.

Harmonised host mobility–multihoming is when *both* mobility and multihoming are occurring simultaneously. The host with harmonised mobility and multihoming capability

will be able to add or remove network connectivity dynamically, and they will be able to make use of more than one network actively for both sending and receiving. Such a mechanism can enable a much more flexible use of network connectivity, especially in the mobile context. For example, in situations where a mobile host is moving through an area with public IEEE 802.11 networks, it may be able to increase its available network capacity dynamically by making use of multiple connectivity points, as well as using existing cellular connectivity. If the cellular network happens to be congested, unlike the current single-homed mechanism, a host will not lose connectivity, and it will still be able to use the IEEE 802.11 connectivity in parallel. Once congestion has decreased, the host will be able to benefit from either the added capacity of the additional network or choose to connect only to a single network.

1.1. Benefits of Harmonised Mobility and Multihoming

There are proposals that enable mobility or multihoming individually, but it is not clear whether they can work together seamlessly in harmony. Often, such mechanisms are designed and implemented separately, and so trying to make them work together retrospectively could result in complex and fragile engineering solutions. For example, the standard Mobile IP solution today from the Internet Engineering Task Force (IETF) cannot work with multihomed systems, as, at the very least, they will use different control plane protocols. Additionally, Mobile IP and other IETF offerings require the use of agents/proxies and tunnelling, introducing additional complexity and overhead into the service provisioning.

Multihoming is often implemented by requiring a network service provider to implement special configuration or functions, for example, IP routing, with the use of very specific addresses and additional routing table entries. This results in a loss of flexibility and portability for a given solution for the end user.

While some transport protocols, such as Multipath TCP (MP-TCP) and QUIC, do offer *multipath* capability, they are transport layer (layer 4) protocols, and so they rely on IP layer (layer 3) mechanisms to provide the *multihoming* capability—they cannot provide it themselves. Additionally, if a new transport protocol is required, the use of multihoming and multipath capability will have to be implemented from scratch for that new protocol. Finally, even if a host is multihomed, existing protocols—TCP and UDP—cannot use the multipath capability: new protocols must be utilised.

Currently, there are no known general solutions to provide *harmonised mobility and multihoming* for an IP—that is, *the combined, seamless use of both mobility and multihoming together*. More specifically, there are no solutions at the IP layer (layer 3) to allow *any* transport protocol to function over *any* (different) layer 2 protocols. This would include using multiple network connectivity without any need for special configuration or functions from a network service provider.

Overall, our solution has the following benefits:

- A harmonised, combined, mobility and multihoming mechanism for hosts/end systems, with a single, simple, unified control plane protocol. No other solution currently exists for this.
- For mobility, there is no need for the use of agents/proxies/middleboxes, tunnelling, address translation, or special mechanisms in the network: only normal unicast routing is required.
- For multihihoming, there is no need for reliance on special routing or provisioning by a network service provider: again, only normal unicast routing is required.
- Our approach implements an IP layer (layer 3) mechanism that can be used by *any* transport layer (layer 4) protocol, including existing protocols such as TCP and UDP, which we demonstrate in our evaluation.

1.2. Contributions

The contributions of this paper are the following:

- An analysis of the addressing mechanisms used for mobility and multihoming today. We use a different addressing architecture that better matches today's dynamic, multinetwork connectivity.
- A realisation of an Identifier Locator-based addressing architecture. We implement the Identifier Locator Network Protocol (ILNP) as a superset of IPv6 to support *mobility–multihoming duality* at the network layer.
- A comprehensive testing and analysis of the performance our approach implemented in Linux. With a testbed using commercial network equipment, we demonstrate operation over IPv6 using standard TCP and UDP APIs via the `iperf2` traffic generation tool.
- The demonstration that an end-to-end architecture with a simple control plane can provide mobility–multihoming duality. Our approach works without proxies, middleboxes, tunnels, or address translation, nor modification to existing routing protocols. We use only normal unicast addressing and routing for both mobility and multihoming provision.

Overall, this presents an approach to harmonised mobility and multihoming for flexible connectivity at the network layer. Therefore, it is applicable to potentially any transport protocol, both current and future.

1.3. Paper Structure

We start in Section 2 with an examination of existing solutions to mobility and multihoming, focussing on those proposals that are being, or have been, actively considered as potential solutions for use on the Internet. We then provide an exposition of our radical approach in Section 3, examining both the architectural and engineering considerations with a focus on real deployment capability over IPv6. We describe our implementation of this approach in Linux in Section 4. In Section 5, we provide a description of our methodology for our evaluation, with details of the testbed and experiments. The results are presented in Section 6, highlighting our findings and showing the efficacy of our approach. We present a critical evaluation of our approach in Section 7, highlighting challenges for deployment on the Internet. We conclude in Section 8 by stating our key findings, as well as listing some items for future work.

2. Combined Mobility and Multihoming

The key point to note in considering related work is that at the time of writing, there are no combined mobility/multihoming solutions, so our proposed solution is unique. LISP has work in progress for providing mobility and multihoming together [2], but it is an extension to LISP multihomimg. For Shim6, a possible combined multihoming and mobility solution was described [3], but it was never implemented. More details for both LISP and Shim6 are shown below.

In this Section, we present a critical examination of related work before presenting our approach in the next Section. In our comparison of other possible solutions, we focus on those that have presence within the Internet Engineering Task Force (IETF). Such solutions are, arguably, the most likely to yield practical, deployable solutions at the time of writing.

Also, in our consideration of a Future Internet scenario (a near-future viewpoint) we focus on IPv6 as the network (layer 3) protocol of choice.

A key component in enabling any mobility and multihoming mechanism is the way that addressing and address bindings are used, and so this features in our discussion.

2.1. Host Mobility

Host mobility is a feature where a host with multiple connectivity moves from one point of attachment (PoA) to another PoA in a different network. This could make use of different sub-network technologies, i.e., across different layer 3 networks, or *vertical handover*. This is not the same as handovers in layer 2/layer 1 connectivity, which are typically handled within a specific network technology, e.g., radio-level handover in 5G networks [4].

Due to the way naming and addressing is used in the current IP architecture, moving across different IP networks and PoAs requires special handling. One or more of the

following mechanisms must be used, which we describe with their relative advantages and disadvantages:

- *Transport protocols.* Specific transport protocol mechanisms could enable mobility and multipath communication through multiple paths, e.g., using Multipath-TCP (MP-TCP) [5] or multipath QUIC [6].
  Strictly speaking, transport protocols do not provide multihoming, but they can use multiple paths—*multipath transport*—where multihoming is available at the site or directly to a host. Transport protocols offer a direct interface to an application, so they are important to consider. However, such an approach will, of course, require that applications use only those transport protocols that directly support multipath. This in turn might require the redesign and/or re-engineering of the application to make use of mobility and multipath connectivity, which is true in the case of QUIC, for example.
- *Middleboxes and proxies.* A middlebox or proxy entity can act as a relay or coordination point to aid a host that moves from one PoA to another. Such an entity typically will use additional control plane protocols with the host to provide such a capability.
  The use of middleboxes and proxies gives rise to a number of general problems, even though they are widely used. A middlebox or proxy approach entails the following:
  1. It could become a performance bottleneck;
  2. It is potentially a single point of failure (SPoF);
  3. It increases the attack surface for potential security and privacy threats, for example, a proxy is the ideal point from which to launch a Man-in-the-Middle (MITM) attack.
  4. It introduces additional operations, administration, and maintenance (OAM) overhead;
  5. It can become more complex in proportion to the scale of deployment, especially with respect to OAM.
- *Application-specific mechanism.* Application-specific mechanisms might use special control plane and user plane protocols for a host, or they might be enabled through the use of a proxy or middlebox. Such an approach has the advantage that it could provide tailored, efficient capability for a particular application's needs.
  Application-specific mechanisms might pose challenges in terms of deployment and uptake, however, as legacy applications could be left with limited connectivity. Application deployments might even be left as non-interoperable, especially if the particular approach prevents an older version from functioning with a newer version. So, overall, there is a deployment and OAM challenge.
- *Special in-network (vendor and provider) support.* Support from a specially designed or implemented mechanism from a vendor or network provider could provide a highly integrated, tailored, and efficient solution for a particular use case. Such an approach might rely on a proprietary protocol and network configuration.
  However, such a solution might be constrained to a single sub-network technology, could be more difficult to deploy at scale, or could tie a user into a specific vendor and/or network provider, limiting networks that the host can potentially 'move' to and from. This approach will narrow the scope of deployment when considering global connectivity. This could be especially so when there is a need to provide connectivity *across* different sub-network technologies, i.e., not any one of 4G, 5G, or IEEE 802.11 by itself offers complete mobile coverage globally.
- *Overlay approaches using tunnelling.* This is where one protocol is encapsulated—tunnelled—within another so that the underlying network infrastructure is unaware of the protocol internal to the tunnel. Tunnelling is typically used as an encapsulation mechanism, where the underlying network infrastructure will not support a protocol directly. A tunnel must be created between tunnel end points, which are new (logical or physical) entities in a network. This approach has the advantage that it can be deployed 'over' an existing network technology. Also, it can use a mixture of approaches as

suits a particular application, e.g., one or more of the approaches listed above could be incorporated, including the use of proxies, or special in-network support.
The main drawbacks here are the requirement that are typical for a new control plane protocol for the OAM functions of the tunnelling mechanism. This requires increased OAM overhead for deployment at scale. If proxies or special in-network support are used, then the drawbacks of those approaches also apply.

2.2. Host Multihoming

Host multihoming enables the use of multiple connectivity available to the host simultaneously and continuously, for example, through multiple network paths or multiple PoAs. This has several applications, such as failover, resilience to Denial-of-Service (DoS) traffic attacks, and load sharing. Specifically for mobile devices, host multihoming may improve throughput using load sharing, as mobile connectivity might be constrained.

Today, host multihoming is not well-supported with IPv4, and with IPv6, the support is present but limited. Similar to host mobility, it might need a middlebox approach, an application-specific mechanism, and/or special support from a network provider, with the constraints listed above.

2.3. Locator Identifier Separation Protocol (LISP)

The Locator Identifier Separation Protocol (LISP) [7] is a network-based solution to realise an identifier–locator split architecture using a routing approach, with direct support for multihoming. LISP defines an End-system Identifier (EID) as a name that acts as an identity for a host and a Routing Locator (RLOC), which acts as a network name for a host. In LISP, both EID values and RLOC values are each an IPv6 address (128 bits), i.e., the semantics of an IPv6 address are overloaded in the communication stack. There is work in progress to define a mobility function for LISP — *EID Mobility* [2]. This allows EID values to 'move' and establish different bindings—mappings—to RLOC values, i.e., effectively move from network to network. In principle, it should be possible for the harmonious operation of both mobility and multihoming for a network site and also possibly for an individual host.

However, the mobility management is complex. It requires a control plane involving a number of additional entities under the administration of a network provider [2]. These include a Map Server, which would be discovered from a Map Resolver, plus Ingress Tunnel Routers (ITRs) and Egress Tunnel Routers (ETRs) to enable the tunnelling mechanism, which is central to the operation of LISP.

Our approach requires no additional network entities, and it has a much simpler control plane.

2.4. End Host-Based SHIM Layer Approach

The Level 3 Multihoming Shim Protocol for IPv6 (Shim6) [8] is a host-based shim layer multihoming solution. Shim6 uses an Upper-Layer Identifier (ULID) and a Locator in its addressing scheme. So, as for LISP, there is overloading of the IP address semantics. While Shim6 was designed for multihoming, by dynamically adding and removing ULID-Locator bindings, host mobility can be realised [3]. Also proposed is a different approach called SEMO6: SEamless MObility using Shim6 [9]. However, the implementation and deployment capability of those proposals are not clear, and they were not progressed within the IETF. Further work on Shim6 did demonstrate how site mobility could be supported, but it did not combine mobility multihoming for individual hosts [10].

Dynamic Internet Mobility for End Systems (DIME) [11] is based on Shim6. While the current version only supports mobility, there is a proposal that a key component of the approach, the Stack-Trans, can be further developed to support multihoming. However, the Stack-Trans is an additional network entity, and it uses a network address translation (NAT) approach to map between a Host Identifier (HI) for a host and an IP address, which acts as a Locator. An implementation in Linux of the mobility part was documented,

but there are no full proposals or implementations available for supporting both multihoming and mobility at the time of writing.

Our approach does not require network address translation, is implemented in Linux, and has been used and tested with commercial equipment to show deployment capability.

2.5. Mobile IPv6 (MIPv6)

MIPv6 is the IPv6 variant of Mobile IP (MIP) [12] (the successor of Mobile IPv4 [13]). Basic MIPv6 support for mobility is via a solution that is part host and part network. It requires a mobility management server (with a proxy/forwarding function), the Home Agent (HA), to track a Mobile Node (MN)'s topological location, as well as forward packets to and from the Correspondent Node (CN). Meanwhile, the MN needs a user-land daemon to establish a tunnel to the mobility management server. Unlike MIPv4, a MIPv6 MN does not require a Foreign Agent (FA) (another proxy) to be present in the network it moves to. However, the requirement for the HA, and therefore the presence of a home network (HN), still remains, along with the use of two IP addresses to refer to the MN (as explained below).

MIPv6 overloads the IPv6 address space semantics, using two IPv6 addresses for the MN: a Home Address (HoA) and a Care-of Address (CoA). The CoA is, effectively, a locator, as it indicates the current topological location of the MN. The HoA is, effectively, an identifier, as it is used to refer to the host globally. Packets are delivered to the CoA initially via a proxy—through the MN's HA. After a Route Optimisation (RO) handshake between the MN and CN, the value of the MN's current CoA is made known to the CN, and direct communication can take place.

The main drawbacks with MIPv6 are the following:

- The network into which the HA moves, the Foreign Network (FN), must support MIPv6. The control plane messages required for operation of MIPv6 must be able to ingress and egress the FN, e.g., the FN firewall needs to be specifically configured.
- While the MN can move to potentially any network, additional infrastructure at the HN introduces deployment and OAM overhead.
- In addition, if the MN or CN cannot enable RO, sub-optimal 'triangular routing' will be used, with traffic tunnelled from the MN to the HA and then from the HA to the CN. This adds unnecessary traffic load to the HN in terms of both ingress and egress, potentially adding latency and becoming a throughput bottleneck between the MN and CN. Additionally, the more MNs an HN has to support, the more the HN and the HA suffer from the increasing load for tunnelling traffic.
- Overall, the HA might become a performance bottleneck, a SPoF, and a point of attack for disruption to the MIPv6 service.
- MIPv6 does not support multihoming.

2.6. Proxy Mobile IPv6 (PMIPv6)

PMIPv6 is a fully network-based variant of MIPv6 [14,15]. Instead of MIPv6's mobility management signalling coming from the host itself, PMIPv6 relies on network entities to carry out the required signalling, management, and tunnelling of packets. PMIPv6 introduces the Mobility Access Gateway (MAG) and Local Mobility Anchor (LMA). The LMA has a similar function to the HA; it allocates the HoA and manages traffic ingress and egress for a PMIPv6 network.

Within the network managed by a MAG, an MN receives an address that is mobile, the HoA. The address of the MAG is the CoA, which the LMA uses to forward packets to reach the MN. As the MN moves from one MAG to another, the MAGs notify the relevant LMA of an MN's change in PoA. The MAGs and the LMAs signal each other to confirm an MN's movement and set up tunnels and routes for correct traffic forwarding. The LMA will be the gateway to the rest of the Internet, while the MAG will provide the tunnel to the LMA.

Unlike MIPv6, PMIPv6 does not require host modifications. However, the infrastructure is a lot more complex, and it limits a MN's movement only to specific networks that have PMIPv6 infrastructures.

Limitations common across different proxy-based solutions still apply to PMIPv6, such as the performance bottleneck and the SPoF.

Also, PMIPv6 does not support multihoming.

2.7. Other MIPv6 Variants

Over time, variants of MIPv6 and PMIPv6 have been designed in order to deal with a number of shortcomings. An assessment of mobility approaches is provided in [16]. We provide here a summary within the context of our proposal, particularly in keeping with those proposals that are aimed at real deployment by being progressed via the IETF.

A key drawback of all of these approaches is that none of them are designed to support multihoming together with mobility, and they all introduce increased OAM overhead:

- *Fast handover for Mobile IPv6 (FMIPv6)* [17]. This seeks to improve handover performance by reducing latency at the network layer by using a proxy to aid the handover. FMIPv6 has a number of enhancements in the areas of address management, handover procedure, dealing with packet loss during change in a PoA, and high rates of movement. However, in order to provide such capability, it introduces additional complexity and OAM overhead. Extensions to the control plane with new messages types are defined, and a proxy with tunnelling is also used as part of the control plane architecture.
- *Hierarchical Mobile IPv6 (HMIPv6)* [18]. The main goals of HMIPv6 are to improve signalling overhead for the MN and to allow smoother and more efficient transition for a MN from one network to another. However, the approach introduces an additional network entity—the Mobility Anchor Point (MAP)—which the MN must communicate with. The MAP now takes responsibility for communication of the control plane interaction between the MN and its HA. The MAP is yet another proxy in the MIPv6 architecture, and so it has the general drawbacks of using a proxy, as already discussed.

In comparison, our approach does not use proxies or tunnels, nor does it introduce further complexity into the control plane—signalling is end-to-end directly between the MN and CN.

2.8. Host Identity Protocol (HIP)

HIP [19] is an end host-based network protocol that uses cryptographic keys as host identifiers. The HIP was not designed initially to enable mobility and multihoming: rather, the primary aim was to provide more secure communication between hosts. However, as it uses an identifier–locator split mechanism, later documents have proposed mobility [20] and multihoming [21] approaches that make use of HIP. These became part of its architectural standard documentation in July 2021 [22].

The HIP introduces the Host Identifier (HI), a cryptographic public key representing the identity of a host. It also introduces the Host Identifier Tag (HIT), an operational representation of the HI that is only 128 bits, being much smaller than current commonly used public key sizes, which can be 2048 bits or 4096 bits.

In HIP, the use of IP addresses still remains; the IP address is used as a locator, with the full 128 bits of the IPv6 address bound to each interface that is used. The HIP host initiating communication, the Initiator, starts a HIP Base Exchange (BEX) by sending a message to the peer, the Responder. This message initiates a handshake using a Diffie–Hellman key exchange for a session key. Once the handshake is complete, an HIP association, a shared state between two HIP hosts, is created. The binding of an HI to a Fully Qualified Domain Name (FQDN) is done using the Domain Name System (DNS). As the IP address no longer serves any part in host identification, the socket has to be re-written for HIP, requiring a new Application Programming Interface (API) for applications [23]. As it is a host-based solution, networking components such as routers will not be able to participate, thus site

multihoming and mobility cannot be achieved, nor is traffic engineering based on a HIT possible. To enable host mobility, a rendezvous mechanism is needed [24], which involves additional network infrastructure entities.

Overall, HIP mandates the use of secure identities, requires additional network entities, and also requires application changes—none of which are required with our approach.

2.9. Multipath TCP (MP-TCP)

MP-TCP [5] is a transport layer solution to allow multipath transport. An MP-TCP socket consists of sub-flows with different pairs of addresses/ports while presenting a single TCP socket to the application. While it presents a TCP socket API to the application, additional options can expose MP-TCP-specific information. MP-TCP-capable hosts will initiate the three-way handshake with an MP-TCP-specific option, MP-CAPABLE, for setting up an MP-TCP main flow with a single IP address. Then, sub-flows are added using MP-JOIN to add another address. As the name suggests, it is a solution specific to TCP, so is not usable for UDP applications, e.g., real-time applications.

Although its original goal was multipath transport, by adding and removing subflows during transmission, it can be used for mobility [25]. There are proposals for integrating mobility management to enable mobility for MP-TCP [26]. However, there are no formal standards or implementations present at the time of writing.

2.10. QUIC

QUIC [5] is a comprehensive UDP-based transport protocol. It offers direct support for much of the functionality present within MP-TCP (e.g., congestion control, flow control, sub-flows, etc.) but with improved communication session control and integrated security. Support for multipath in QUIC is currently a work in progress [6]. However, mobility support in QUIC does not yet exist, though it is conceivable that similar mechanisms to those of MP-TCP could be designed.

2.11. Other Approaches

As discussed in detail in Section 3, ILNP is an Identifier Locator (IL) approach to networking. Although there are other approaches to IL networking, they have varying architectures and, especially, approaches to addressing. Architecture and addressing are particularly important to consider, given that we have noted at the beginning of this section that we have a desire to focus on practical, deployable solutions.

A survey of IL approaches to networking is provided in [27]. That study made qualitative comparisons between many proposals that use an IL approach. The comparison included discussion on state management and control plane functionality, as well as considering different OAM approaches, e.g., edge-core separation, centralised vs de-centralised, etc. The study included ILNP, HIP, LISP, MIPv6, and Shim6 (all compared in Table 1), as well as related proposals.

Table 1. A summary of different mobility and multihoming solutions.

Criterion	LISP	MIPv6	Shim6	DIME	HIP	MP-TCP *	QUIC *	ILNP
FUNCTION								
Addressing	IP (EID, RLOC)	IP (HoA, CoA)	IP (ULID, Loc)	IP (HID, Loc)	HI, IP (Loc)	IP	IP	NID, L64
Layer	**3**	**3**	**3+** (shim)	**3+** (shim)	**3+** (shim)	**4**	**4**	**3**
Mobility	**+**	✓	**+**	✓	**+**	**+**	–	✓
Multihoming	✓	–	✓	**+**	**+**	–	**-**	✓
CHANGES								
Host	**N**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**Y**
Applications	**N**	**N**	**N**	**N**	**Y**	**Y**	**Y**	**N**
Network	**F,P**	–	**F**	**F**	–	–	–	–
New Entities	**3,T**	**3,T**	**N**	–	**A,3,T**	–	**A**	–

Table 1. *Cont.*

Criterion	LISP	MIPv6	Shim6	DIME	HIP	MP-TCP *	QUIC *	ILNP
FUNCTION								
Addressing	How addresses and addressing are used. 'IP' is 'IP address', 'Loc' is 'Locator'. Other definitions in main text above.							
Layer	Which layer is modified **3** network **4** transport							
Mobility	How mobility is supported ✓ : within core architectural design **+** : as additional/extension functionality − : not supported							
Multihoming	How multihoming is supported ✓ : within core architectural design. **+** : as additional/extension functionality. − : not supported.							
CHANGES								
Host	**Y**es or **N**o : host needs to be updated.							
Applications	**Y**es or **N**o : APIs and/or applications need to be updated.							
Network	Existing network **F**unctions or **P**rotocols need to be updated.							
New entities	New entities for **A**pplication layer, Layer **3** (network), or **T**unnelling.							
LISP	Overloads IP address usage, needs new network entities, control plane protocols, and tunnelling.							
MIPv6	Overloads IP address usage, needs to network entities.							
Shim6	Overloads IP address usage, needs Shim layer for ULID/Loc mapping.							
DIME	Overloads IP address usage, needs NAT-like function for HID/IP mapping.							
HIP	Needs public keys to generate secure HI values, needs new network entities, applications need to be modified.							
MP-TCP	Transport layer only.							
QUIC	Transport layer only, new API, needs applications to be modified.							
ILNP	Network layer, no tunnels, proxies, or address translation; no modification to APIs or applications. (See Section 3.)							
*	Strictly, multihoming happens at the network layer (layer 3). Transport (layer 4) protocols can offer *multipath* communication when multihoming exits.							

Another review and assessment of IL approaches is given in RFC6115 [28]. That study took a practical view and considered global deployability, scalability, and compatibility with current Internet architecture as part of its analysis. Again, ILNP was included with HIP, LISP, and Shim6, and others, so there is overlap with the list of approaches examined in the survey noted above [27]. The RFC6115 recommended the use of ILNP as the preferred solution.

Meanwhile, where general Internet-wide deployability is not a key consideration, there are possibilities for enterprise-oriented solutions for combined mobility and multihoming. These often leverage Virtual Private Network (VPN) technology for specific use cases, and two key examples are given below.

A mature solution is IKEv2 Mobility and Multihoming Protocol (MOBIKE), where IKE is the Internet Key Exchange Protocol [29]. The use of MOBIKE is defined in [30] to provide both mobility and multihoming, and it builds upon a VPN-based system initially defined in [31]. The VPN-based approach is focussed on enterprise users. This means that it has an OAM overhead for overall deployment, and it is not suited for general-purpose use by Internet users, e.g., domestic users. As ILNP only requires unicast connectivity and uses a simple, end-to-end signalling protocol, it does not have the same constraints.

Another enterprise-focussed approach, especially for providing interconnection between datacentres, is the Ethernet VPN (EVPN) [32]. As the name suggests, the main goal of this approach is to extend the *logical* viewpoint of a single Ethernet network beyond its normal *physical* constraints. EVPN is built upon the notion of a Virtual Private LAN Service (EVPLS) [33]. EVPN extends this capability to provide the logical viewpoint of the Ethernet beyond the boundary of a single network. EVPN uses extensions to the Border Gateway Protocol (BGP) along with Multiprotocol Label Switching (MPLS) [33–35] for state

distribution and management between network sites. The EVPN can provide multihoming and also provides mobility for nodes based on the management of Media Access Control (MAC) address values for those nodes between network sites. There is clearly a significant control plane overhead for managing the state across an EVPN. ILNP does not have such state management overhead, using only a simple, end-to-end control plane protocol.

2.12. Summary of Related Work

A summary of the salient points from the discussion above are given in Table 1. The final column of the table is for the Identifier Locator Network Protocol (ILNP) upon which our approach is built and which is described in the rest of this paper.

Note that, strictly, multihoming is made possible at the network layer (layer 3), and transport protocols (such as MP-TCP and QUIC) offer multipath communication when multihoming is already enabled. In contrast, ILNP enables multihoming directly at the network layer. However, it is important to include such transport protocols in this comparison, as that is the interface by which applications access multihoming connectivity.

In general, there are trade-offs to be made in considering the various solutions:

- As discussed, any proxy-based solution has the potential to become a performance bottleneck, an SPoF, and offer an additional attack surface in both the data (user) plane and the control plane. Any proxy systems also will have an OAM overhead. However, such solutions might reduce the burden on the hosts or the requirement to modify end-to-end protocols or applications.
- Any solution relying on new network entities also has similar advantages and disadvantages to proxy-based solutions. They also have a deployment cost, requiring a network service provider who is willing and able to support such a service. Indeed, this applies to all providers that a host might use in the case of mobility. Also, the use of additional network entities often requires the use of tunnelling.
- New APIs and changes to applications are generally undesirable, as they greatly increase the barrier to entry for usage of the system. It might be impractical to modify some applications.
- Changes to a host operating system (OS) might be disruptive to the operation of that host. However, the modern ecosystem for OS distribution and maintenance, using 'over-the-air' updates, means that such disruption can be minimised, and deployment can be managed incrementally.

Arguably, a host-based solution that maintains current APIs offers the least disruption. It has the greatest potential for incremental deployment, as well as the lowest barrier to entry for those wishing to use a harmonised mobility multihoming solution. This is our approach with ILNP.

Additionally, from a network perspective, there are already industry directions for what multihoming can look like at the network layer for the IPv6. This includes the goals [36] and the provision of direct multihoming to sites [37] so that multiple connectivity can be used by hosts, as well as also security issues to be considered [38]. These will be discussed further as we describe our solution in Section 3.

3. Identifier Locator Network Protocol (ILNP)

It is clear from our discussion in Section 2 and Table 1 that the use of addressing and a particular *binding* of an instance of an address value to an object are important in enabling a particular approach. We see in Table 1 that the IP address is overloaded in its semantics in order to enable some form of redirection (e.g., tunnelling in LISP) or indirection (e.g., mappings from the EID to the RLOC in LISP). An address value can play the role of a (global) Identifier or a (global) Locator (as in MIPv6). The Identifier has no topological semantics, but it is a name that uniquely labels a host. A Locator is a name that does have topological significance, and it is used for routing (and/or forwarding).

In our approach, we explicitly recognise this dichotomy of address usage semantics. With the Identifier Locator Network Protocol (ILNP), we change the addressing model so

that the *Node Identifier (NID)* values and *Locator (L64)* are clearly defined without overloading IP address semantics.

In our discussion, we use the term *name* as a general label referring to, "a set of bits that has a specific semantic and syntax at a given layer in the protocol architecture and is bound to a specific object in that protocol architecture". So for example, a "network name" is a routing prefix (the upper 64 bits of the IPv6 address) for IPv6—a Locator (L64) for ILNP—and the host part of the address (lower 64 bits) is a unique "node name" for a host—a Node Identifier (NID) in ILNP (see Section 3.2).

3.1. Mobility and Multihoming: A Naming Problem with a Naming Solution

In this subsection, we will discuss the abstract model for our approach first to establish the principle of operation. We can see both the mobility and multihoming functionality as naming problems, with a common, naming-based solution. Consider a node with name N_A that is at a network with name L_1. It has an *Identifier Locator Vector* (IL-V) $\langle N_A, L_1 \rangle$.

- For **mobility**,we allow dynamic bindings between a node name, N_A, and a network name, L_1. This allows a node to 'move' from one network to another by changing the binding between values. For example, a node with name N_A bound to a network with the name L_1 can 'move' to a network with name L_2 by changing its IL-V from $\langle N_A, L_1 \rangle$ to $\langle N_A, L_2 \rangle$
- For **multihoming**, we allow dynamic bindings with a 1:*many* relationship between a node name and a network name. This allows a node to have a PoA to multiple networks at the same time. A node with name N_A can be bound to four network names—L_1, L_2, L_3, and L_4—making it multihomed using multiple IL-Vs with the same node name, e.g., the set

$$\langle N_A, L_1 \rangle, \langle N_A, L_2 \rangle, \langle N_A, L_3 \rangle, \langle N_A, L_4 \rangle$$

A short-hand notation for this is

$$\langle N_A, L_1 | L_2 | L_3 | L_4 \rangle$$

In both cases, the logical requirement is to allow *dynamic name bindings*. In this way, bindings between node names and network names can be changed to show new connectivity, whether that is because of movement or because of multihoming. In practical terms, node names and network names are also bound to objects in the network stack to enable protocol operation. So, for ILNP, we hace the following:

- At the transport layer, a communication end point binds to an NID value, identifying a node with a name. This binding should remain fixed for the duration of a communication session to enable end-to-end integrity for a communication flow.
- The NID value has a dynamic binding with one or more L64 values, 'connecting' a node to one or more networks. The NID-L64 binding(s) (IL-Vs) can change at any time to reflect changing network connectivity.
- The L64 value will have a (dynamic or fixed) binding to a physical interface on a host so that the network name is bound to a real network for transmission.

The last point above might not be obvious from our discussion so far, but it should be clear when we discuss the practical implementation of ILNP in the next subsection.

If we compare these names and bindings to an IPv6, we have the following:

- At the transport layer, a communication end point binds to a full IP address.
- The full IP address is also bound to a physical interface.

Effectively, the IPv6 bindings tie a transport layer communication end point (and therefore a transport communication flow) to a physical interface and so to a single-network PoA.

This is a basic property of the IP addressing architecture. The various uses of multiple addresses and/or proxies and/or tunnelling and/or address mappings are mechanisms

used in the approaches other than ILNP to circumvent this fundamental constraint (as summarised in Table 1).

This is not the case for ILNP: the dynamicity of the NID-L64 bindings(s) allows a transport communication flow to 'move', or to be 'connected' to multiple networks.

3.2. ILNP and IPv6

Whilst the change in addressing architecture for ILNP is radical, we have implemented ILNP as a superset of IPv6. The IPv6 packet carries an IL-V in its address fields. This allows a L64 value to be realised as an IPv6 routing prefix, which constitutes the most significant 64 bits of an IPv6 address. The ILNP NID value is then inserted as the least significant bits of the IPv6 address. This is shown in Figure 1. The L64 has the same syntax and semantics as a routing prefix—a network name—and is the only part of the address bits used by the IPv6 core network components for the routing packets. So, ILNP packets can be carried across an IPv6 network. The NID has the same syntax as the Interface ID (IID) for an IPv6 address, but it has different semantics. However, as these semantics are only local to the host, updating the host is all that is required to deploy ILNP.

```
/* IPv6 - RFC4291(DS) */
|            64 bits           |            64 bits            |
+------------------------------+------------------------------+
| IPv6 Unicast Routing Prefix |  Interface Identifier (IID) |
+------------------------------+------------------------------+

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

Figure 1. Comparison of the IPv6 unicast address format with the ILNP unicast addressing 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 it has different semantics. The NID-L64 pairing is an Identifier Locator Vector (IL-V), which can be used the same way as an IPv6 address.

For local forwarding, as for IPv6, ILNP uses the full 128 bits of the IL-V as part of IPv6 Neighbor Discovery (ND) [39]. So, ND mechanisms and implementations do not need to be modified—link layer (layer 2) protocols can forward ILNP packets as they would IPv6 packets, e.g., forwarding in layer 2 relays, such as switches.

At the transport layer (layer 4), there are also differences in bindings with respect to the state held in the communication end points. The state bindings in IP, the tuple expression (1), show the state information for a communication end point at host X for TCP communication between hosts X and Y. A values are IP addresses, P values are port numbers, and if designates an interface on the host.

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

For ILNP, the state bindings are as in the tuple expression (2):

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

The port numbers are unchanged. However, the TCP state is bound to the NID values (node names), N, and the IP layer is now only concerned with L64 values (network names), L. The binding notation $(\ldots)$ denotes a dynamic binding rather than the fixed binding in an IP.

Then, tuple expression (3) shows what happens when the ILNP node moves from the network with name L_X to the network with name L_A. The tuple expression (4) shows what

happens when the ILNP node becomes multihomed on networks with names L_X and L_A. L values are ILNP L64 values that are the same as the IPv6 (routing) prefix values.

$$\langle tcp : P_X, N_X, P_Y, N_Y \rangle \langle ilnp : (L_A), (L_Y) \rangle \langle if : (L_A) \rangle \quad (3)$$

$$\langle tcp : P_X, N_X, P_Y, N_Y \rangle \langle ilnp : (L_X|L_A), (L_Y) \rangle \langle if : (L_X|L_A) \rangle \quad (4)$$

The ILNP-specific state information is held in a data structure called the *ILNP Communication Cache (ILCC)*, which is a logical entity within the host that we use in our description of ILNP.

3.3. ILNP Control Plane

When node X locally changes the state of its ILCC, relevant information needs to be communicated to the remote correspondent node (CN), Y. This is so that Y becomes aware of the new connectivity of X and can update its own ILCC. For a transport flow that is in progress, the NID stays fixed for the duration of that flow. However, the L64 values can be changed. When X has new L64 values, it notifies these to Y with a simple control plane protocol using a *Locator Update (LU)* message [40]. An example of such a handshake for a Mobile Node (MN) and a Correspondent Node (CN) is shown in Figure 2. This is a simple, efficient messaging system with low latency—a single round trip time (RTT) for an update.

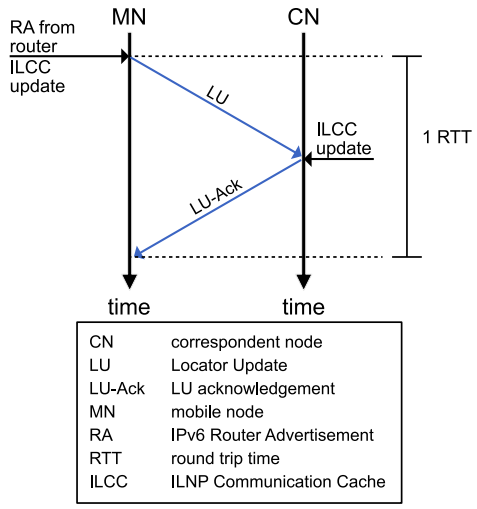


Figure 2. An example of a Locator Update (LU) handshake for a Mobile Node (MN). The MN discovers a new L64 value via an IPv6 Router Advertisement (RA) message. It updates its local ILNP Communication Cache (ILCC) and sends an LU message to the Correspondent Node (CN). The CN updates its own ILCC and sends a LU-ack (acknowledgement) to the MN.

A similar exchange would occur for multihomed nodes: the handshake allows multiple L64 values to be communicated in a single LU message.

Additionally, the dynamic name bindings in ILNP allow for NID-L64 bindings to be independently managed. So, there is, effectively, a network layer soft handover as a node moves or changes connectivity. This means that packet loss should be minimised, as packets 'in-flight' have the opportunity to be delivered (and acknowledged) before NID-L64 bindings are removed.

3.4. Security and Privacy

As a basic enhancement to security compared to IPv6, all ILNP packets carry a pseudo-random Nonce value [41] in each new transport flow. This provides protection against off-path attackers, including remote traffic based Denial-of-Service (DoS) attacks and spoofed/forged packets. If the threat model is greater, then an IPsec [42] can be used with an ILNP by using NID values in place of IP addresses for the IPsec security association (SA) [43].

For NID values, ILNP can use any of the privacy mechanisms used to generate IID values for IPv6 [44,45]. However, ILNP has the additional benefit in that it can generate and use ephemeral NID values [46]. A new NID value can be used for each transport layer flow, further improving security by making spoof/forging attacks harder. This also enhances privacy by preventing flow correlation attacks [47]. Furthermore, by using network layer multhoming and multipath communication, it is possible to perturb traffic monitoring and analysis attacks, even for encrypted traffic [48].

4. ILNP Implementation in Linux

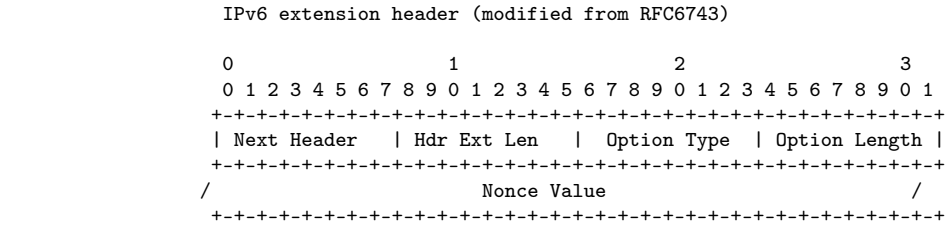
ILNP is an Experimental status protocol from the IETF, with the relevant RFCs for this proposal being RFC6740/1/3/4 [40,41,43,49]. We have implemented ILNP in Linux [50] as a superset of IPv6. The changes required within the IPv6 implementation are, in summary, the following:

1. Modification of IPv6 packets for ILNP packets with the addition of the ILNP IPv6 extension header defined in [41].
2. Additional ICMPv6 messages for the LU handshake, as defined in [40].
3. L64 state management and IPv6 RA handling compatible with ILNP.
4. A simple load-sharing mechanism to distribute packets across multiple interfaces.
5. Modifying the route selection mechanism and associated functions for IPv6.
6. An additional `sysctl` interface call to enable/disable L64 values. This is for management plane functions, and it does not require applications to be modified.
7. An additional `sysctl` interface call to change host-wide operation to ILNP. This, again, is for management plane functions, and it does not require applications to be modified.

The last two of the items listed above are local management mechanisms. The first five of these items modify the processing of the transmission and reception path for IPv6 packets, and so they are described in separate sections below. The state information related to those first five items is managed within an in-kernel data structure that represents the ILCC.

4.1. ILNP in IPv6

ILNP packets are IPv6 packets that carry an end-to-end extension header, as defined in RFC6744 [41] and shown in Figure 3. The extension header is an IPv6 Destination Option header. So, it does not cause additional load to routing or forwarding processes; it is generated and processed only by end systems. The extension header contains a Nonce value of either 32 bits or 96 bits, which is used by the ILNP end systems. It serves two purposes: it marks a packet as being an ILNP packet, and the (pseudo-random) Nonce value contained acts to perturb off-path packet forging attacks.



Next Header: As defined for IPv6.

Hdr Ext Len: Length of this header extension in bytes; value 8 or 16 depending on length of Nonce value.

Option Type: 139

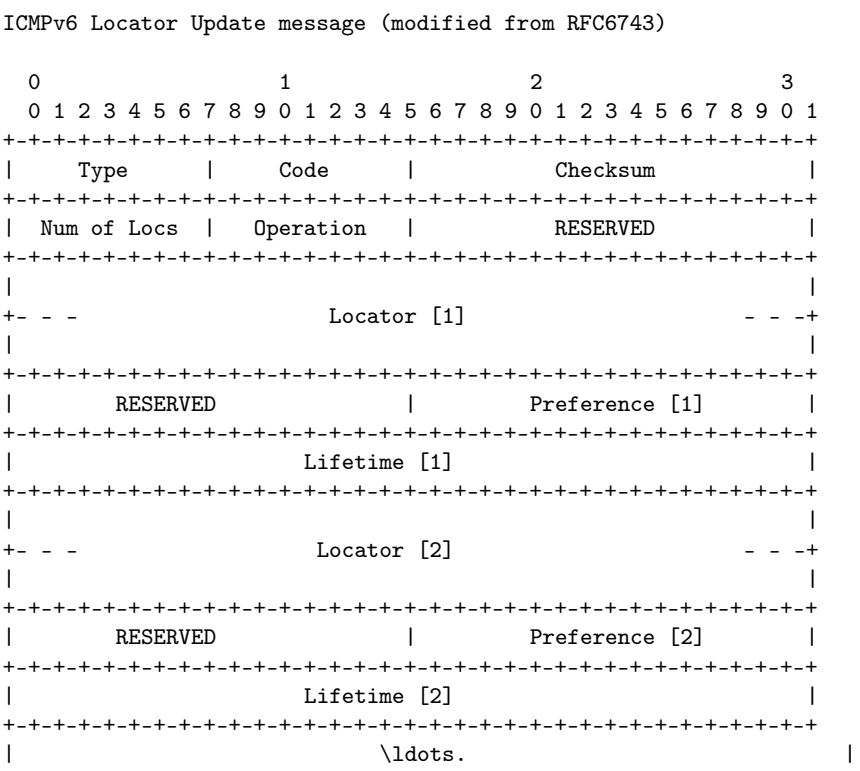
Option Length: Length in bytes of the Nonce value field. Value is 4 (when the Nonce value is 32 bits) or 12 (when the Nonce value is 96 bits).

Nonce Value: An unpredictable cryptographically random value used to prevent off-path attacks on an ILNP session.

Figure 3. The ILNP IPv6 extension header as in RFC6744.

4.2. Locator Update (LU) Messages

In multihoming scenarios, multiple L64 values must be active, and thus the LU message must be able to carry multiple L64 values. The LU message implementation is an extended version of RFC6743 [40], as shown in Figure 4.



Type: 156

Code: 0 or `ILNP_LU_REPLACE_SOFT`.

Checksum: The 16-bit one's complement of one's complement sum of the ICMP message, starting with the ICMP type. For computing the Checksum, the Checksum field is set to 0.

Num of Locs: The number of 64-bit Locator values that are advertised in this message. This field MUST NOT BE zero.

Operation: The value in this field indicated, whether this is a Locator Update Advertisement (0x01) or a Locator Update Acknowledgement (0x02).

Reserved: A field reserved for possible future use. At present, the sender MUST initialise this field to zero. Receivers should ignore this field at present. This field may be used for some protocol function in future.

Locator[i]: The 64-bit Locator values currently valid for the sending ILNP node.

Preference[i]: The preferability of each Locator[i] relative to other valid Locator[i] values. The Preference numbers here are identical, both in syntax and semantics, to the Preference values for L64 records, as specified in RFC6742.

Lifetime[i]: The maximum number of seconds that this particular Locator may be considered valid in its unsigned 32-bit value. Normally, this is identical to the DNS lifetime of the corresponding L64 record if one exists.

Figure 4. The ILNP LU message structure based on the message format from RFC6743.

Compared to RFC6743, we have the following:

- The `Lifetime` field is extended to 32 bits to align with the equivalent field for IPv6 prefixes in IPv6 RAs, which are also 32 bits [39].
- A new LU `Code` value, `ILNP_LU_REPLACE_SOFT`, is defined. This allows for a "soft" replacement at the CN of the current set of L64 values with the new set of L64 values contained within the LU message (see Section 4.3).

4.3. L64 State Management

An L64 value is an IPv6 routing prefix. L64 values for an end system are stored in the ILCC. When a new prefix is discovered from IPv6 RA messages, e.g., when a physical interface is enabled, the prefix can be used as an L64 value. This can be used for mobility and multihoming: the local ILCC is updated, and this L64 needs to be signalled to the correspondent node (CN) via LU messages. The replacement of one set of L64 values with a new set of L64 values can be either *soft* or *hard*.

With *soft replacement* or *soft handover*, the changes are applied in stages; that is, the sender of the LU waits for an LU acknowledgement from the CN before applying the change in L64 values locally. This is in contrast to a *hard replacement* or *hard handover*, where the sender would apply the change straight away, without waiting for an LU acknowledgement from the CN. We only use the soft approach.

When conducting soft handovers in mobility–multihoming scenarios, the communication session must first transition to the *'new path(s)'* before the *'old path(s)'* is (are) decommissioned. This reduces the disconnection time between the CN and MN, and it minimises packet loss during transmission. In the context of mobility–multihoming, one or more interfaces (PoAs) or L64 values are changed within the ongoing communication. Such a state change must be handled by the ILNP mobility–multihoming ILCC mechanism itself, and it must be coordinated between the two hosts participating in the communication session.

The implementation of ILNP includes a 'disabled interface list' to facilitate the successful transfer of communication away from unused interfaces before they become disabled. Specifically, a network interface could be added to such a list, which triggers the LU message to be sent to the CN to notify it that one or more L64 values for the session are being changed.

For the purposes of the evaluation, the interface was not turned off completely. Instead, it was only recorded in the disabled interface list. This allowed the evaluation to remain simpler in terms of the procedure, and it eliminated the time required for lower layer (layer 1/2) initialisation beyond the beginning of the communication session. This is because the lower layer initialisation is a variable that can change depending on the lower layer technology, and so it is not a factor in considering the operation of ILNP. In our testbed, we found that such initialisation took around $\sim$7–10 s with the equipment we used.

4.4. Load Sharing across Interfaces (Paths)

Load sharing is an application of host multihoming where the capacity of more than one interface available to the host is aggregated, and the traffic is shared amongst those interfaces.

For our experiments, Deficit Round-Robin (DRR) scheduling was used [51] as a general mechanism for distributing packets across interfaces. However, the load-sharing scheme can be changed to whatever better suits a particular application.

The state management mechanism for each L64 entry holds the DRR score derived from the bytes sent from each packet's length. Each time an additional interface is added, the DRR score for each L64 entry in the ILCC is reset. For both the source L64 and the destination L64, the L64/interface with the lowest score at any given time is selected for transmission. As each L64 value is selected for outgoing packets, their respective DRR scores are increased by the size of the packet transmitted, whilst the scores on each of the other L64 values are decreased by the same amount.

4.5. Route Selection Mechanism and Transport Layer Packet Processing

In order for ILNP to select a source L64 for multipath transport, the outgoing interface must correspond to the L64 selected by ILNP on a per-packet basis. However, the two distinct transport layer packet processing mechanisms for TCP and UDP select the outgoing interface at the early phase of their packet processing in the Linux kernel.

The packet processing path involves route lookup when the UDP and TCP packet headers are being constructed *before* they are passed onto the network layer packet processing mechanisms. This is to ensure that the pseudo-header source address (and Checksum calculation) matches with the actual source address and the outgoing interface. In the context of a conventional IPv6, the actual source address and the respective outgoing interface do not change; therefore, they are cached after the initial lookup. However, that is not the case with ILNP, especially in the context of multihoming, where every packet may leave from a different interface to the previous packet for the same flow.

To allow ILNP to select the correct outgoing interface, the ILNP code needs to modify the logic for the route selection mechanism for the IPv6. So, both the UDP and TCP packet processing mechanisms were modified to correctly set the outgoing route and the respective interface. Specifically, by preventing the caching of routes and instead triggering route lookup for every packet, ILNP allows for selection of the outgoing L64/interface for each packet. This in turn allows for maximum flexibility for packet forwarding across multiple interfaces. Figure 5 shows a simplified diagram illustrating how the UDP and TCP packet processing were modified to enable source L64 selection and respective outgoing interface selection.

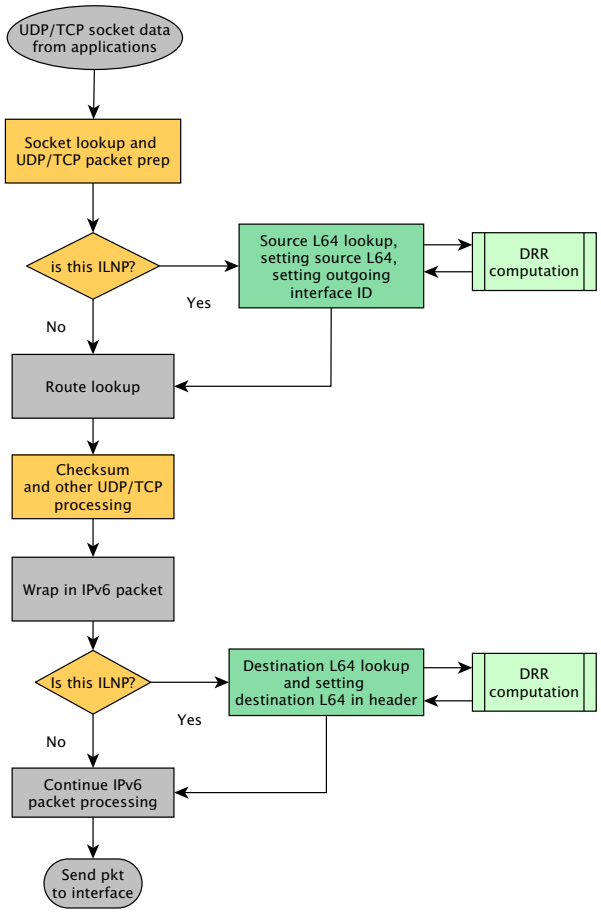


Figure 5. A flowchart describing UDP/TCP packet processing with ILNP mobility–multihoming duality mechanism with Deficit Round-Robin (DRR) load sharing. Overall, the existing IPv6 packet processing code path has been re-used and modified effectively. Grey boxes indicate unmodified processes with respect to IPv6 packet processing. Orange boxes indicate modifications of existing IPv6 packet processing logic. Green boxes are the additional logic and processing for ILNP.

4.6. Linux System Configuration

The IPv6 system configuration that was used served two purposes:

- Ensured that the possible processing of packets on the network interface card (NIC) was disabled. This was so that the ILNP packet processing could function correctly (TCP/IP and UDP/IP processing on the NIC are not ILNP-aware).
- Configured the IPv6 Router Advertisement (RA) and Router Solicitation (RS) messages to provide the IPv6 routing prefix information to correctly signal that the new IPv6 prefixes (paths) were available. Note that this is not ILNP-specific but is an IPv6 mechanism.

For the first item, `ethtool(8)` was used with options, as shown in Table 2, for each interface.

Table 2. Interface configuration options for `ethtool(8)` to avoid incorrect ILNP processing in NIC hardware. These were executed for each interface/NIC.

Option	Explanation
gso off	turn off generic segmentation offload
gso off	turn off generic segmentation receive
tso off	turn off TCP segmentation offload
rx off	turn off TCP receive check sum
tx off	turn off TCP transmission check sum

For the second item, the RA and RS operation was configured using `sysctl(8)` so that the information regarding available IPv6 prefixes was made available in a timely manner. Again, this needed to be configured for each interface/NIC. The system parameters configured are shown in Table 3.

Table 3. The parameters configured using `sysctl(8)`, to be applied for each interface, using the `net.ipv6.conf.`*`interface`* prefix to enable timely generation of IPv6 RA and RS messages. This was executed for each *interface*/NIC. DAD is Duplicate Address Detection, part of IPv6 address management, and not specific to ILNP.

Parameter	Explanation
router_solicitation_delay = 0	delay before generation of RS when an interface is enabled
router_solicitation_interval = 1	delay interval between periodic RS messages
optimistic_dad = 1	enable use of optimistic DAD, i.e., 'optimistic' addresses
use_optimistic = 1	always use (do not deprecate) optimistic addresses

5. Evaluation of Mobility Multihoming Duality

In this section, we describe the evaluation scenario for our approach. It is an empirical, testbed-based evaluation using unmodified, commercial IPv6 equipment for network connectivity. Commercial workstations were used for hosts, which ran the publicly available Linux implementation of ILNP [50] with the modifications executed as described in Section 4 and summarised below.

5.1. Testbed Hardware, Network Topology, and Configuration

The testbed consisted of the following:

- A total of 4× commercial IPv6 routers, R1–R4, that were unmodified with respect to ILNP.
- A total of 2× host nodes, for an MN and CN, consisting of a low-end server motherboard, where this server motherboard was chosen because it had 4× ethernet interfaces onboard for convenience.
- The ILNP kernel was modified from [50] with some additional control features:
  1. A system parameter, `net.ilnp6.handoff_mode`, which can be set using an `sysctl` call to invoke the ILNP behaviour for the host.
  2. A system parameter, `net.ilnp6.disabled_interface` which holds a list of interfaces that is disabled (not in use).

3. When multiple L64 values, and therefore multiple networks, are in use, a Deficit Round-Robin (DRR) mechanism [51] was used to share packets between the networks.

- For a node, the route selection mechanism for the IPv6 was updated, removing the assumption of the use of a single network. This is described in Section 4.5 and discussed further in Section 7.1.

The details of the system are given in Table 4.

The connectivity of the testbed was as shown in Figure 6. There were four IPv6 routers, R1–R4, providing access to four IPv6 networks that are listed in Table 5, which we will for convenience call networks aa–dd.

R1, R2, and R3 provided access networks for the MN. The configuration of the IPv6 RAs for these routers is as given in Table 6.

The glibc C library was modified to hold IL-V values in the `/etc/hosts` file to avoid DNS latency and for experimental convenience. Specifically, the glibc was modified to parse the custom IL-V syntax as shown below:

```
n-n-n-n.l+l+l+l    hostname
```

The `n-n-n-n` value is the NID, which is of a similar format to the unicast IPv6 IID, where ":" was replaced with "-". The `l+l+l+l` value is the L64 value, the unicast IPv6 routing prefix, where ":" was replaced with "+". This provided a fixed IL-V used to initialise ILNP communication, after which the movement and multihoming occurred. Below is an example of an IL-V entry:

```
d250-99ff-fed1-5a0c.2001+2+dd+dd       mn-ddd-ilnp
```

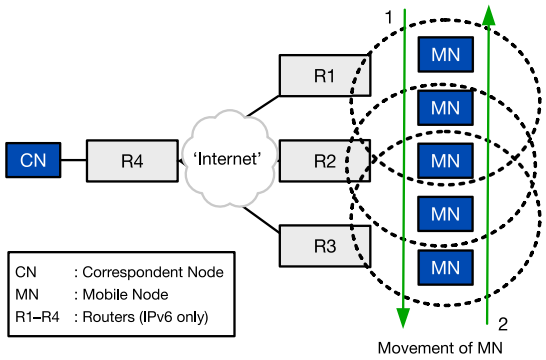


Figure 6. A scenario diagram describing host movement for the mobility–multihoming duality evaluation. There are four IPv6 networks, aa–dd, connected via the 4 routers, R1–R4, scenario. Network dd is used connect R1, R2, and R3 to R4, and it represents connectivity on over the Internet between MN and CN. The arrow labelled 1 is the first movement the MN carries out: moving from network aa on R1 to network cc on R3. The arrow labelled 2 shows the second set of movement, where the MN returns from network cc back to network aa.

Table 4. Testbed equipment information.

	Equipment/Software	Additional Information
R1, R2, R3	Ubiquiti Network ER-X [1]	Software version 1.10.9
R4	Ubiquiti Network ER-6P [2]	Software version 1.10.7
MN/CN	Motherboard ASRock C3558D4I-4L [3]	Intel Atom C3558 (4-core) SoC 4× Gigabit Ethernet 8 GB DDR4 RAM, 223 GB SSD
	Operating System	Debian 9.7 with Linux kernel v4.9 ILNP
	Traffic generation	`iperf2` v2.0.9
	Packet Capture	`tcpdump` v4.9.3 with `libpcap` v1.8.1

[1] https://dl.ubnt.com/datasheets/edgemax/EdgeRouter_X_DS.pdf, accessed on 1 September 2024. [2] https://dl.ubnt.com/datasheets/edgemax/EdgeRouter_DS.pdf, accessed on 1 September 2024. [3] https://www.asrockrack.com/general/productdetail.asp?Model=C3558D4I-4L, accessed on 1 September 2024.

Table 5. IPv6/ILNP configuration for evaluation testbed. The ILNP L64 is an IPv6 prefix.

Network	Router	ILNP L64 Value
aa	R1	`2001:2:aa:aa::/64`
bb	R2	`2001:2:bb:bb::/64`
cc	R3	`2001:2:cc:cc::/64`
dd	R4	`2001:2:dd:dd::/64`

Table 6. Lifetime and interval settings of IPv6 Router Advertisements (RAs) on access networks provided by R1, R2, and R3.

Parameter	Value [s]
max-interval	5
min-interval	3
valid-lifetime	15
preferred-lifetime	9

5.2. Experimental Methodology

New L64 values detected via IPv6 RAs can be VALID or ACTIVE states within the ILCC. A VALID state for an L64 value means that it is available for use, but it is not currently in use. The ACTIVE state means that the L64 is in use. It is possible for more than one L64 to be ACTIVE—an "overlap" period for the use of the networks to which the L64 values refer.

When multiple L64 values are ACTIVE, multiple networks are in use. Our implementation uses a host-wide load balancing, with a Deficit Round-Robin (DRR) [51] mechanism on both the source and destination L64 selection. This allows both send and receive to happen on the ACTIVE L64s present on the MN. In addition, this includes a mechanism to disable network interfaces from the communication on the host by listing them in a list of disabled interfaces. This mechanism gracefully removes the L64s associated with those interfaces included on the disabled interface list. This is done by notifying the CN of the new set of L64s using LU messages, allowing those interfaces to be disconnected without the loss of packets or disruption to the communication.

The following describes how each stage is performed:

1. **Adding an L64**
   In an ongoing communication between two hosts, the ILCC holds one NID and one L64, starting on the network aa. Upon adding a new L64 on the MN, the following steps take place:
   (a) The layer 2 'link-up' occurs at the new interface on the MN.
   (b) An IPv6 RA is received at the MN.
   (c) ILNP adds the new prefix as an L64 in the local ILCC at the MN, and it sets the new L64 as VALID.
   (d) On the L64 currently in use, the MN sends to the CN an LU containing two L64s: the new L64 and the one that is currently in use.
   (e) The MN sets the new L64 as ACTIVE, and it begins sending from it.
   (f) The CN receives the LU on the known L64, and it updates its ILCC entry for the MN with the additional L64.
   (g) The CN sends a LU-ack to the MN's newer L64, and it begins sending to the MN's new L64.
   (h) The MN receives the LU-ack from the CN.
2. **Removing a L64**
   With an ongoing communication utilising multiple L64 values between two hosts, the following steps take place to gracefully remove an L64.
   (a) On the MN, using the ILNP `sysctl` interface entry, add an interface to be removed from the communication to the disabled interface list.
   (b) The MN identifies the L64 belonging to the interface and sets its state to VALID.

(c) The MN sends an LU with all ACTIVE L64s: that is, without any L64 values that belong to an interface in the list of disabled interfaces in step 2a.
(d) The CN receives the LU with the new L64 values, and it sets the state of the removed L64 as VALID in the L64 entry for the MN.
(e) The CN sends an LU-ack to the MN's L64 that is included in the LU.
(f) The MN receives the LU-ack, and it can now turn off the interface.

5.3. Mobility–Multihoming Duality Performance Measurements

By combining the two sets of actions listed above, ILNP hosts can dynamically add and remove a network without disruption to the communication flow. This allows the host to move completely away from the initial network and then back (if required). This in turn realises completely agile and flexible network connectivity between the two hosts—*mobility–multihoming duality*. Figure 7 gives a timeline breakdown of this mobility–multihoming duality scenario.

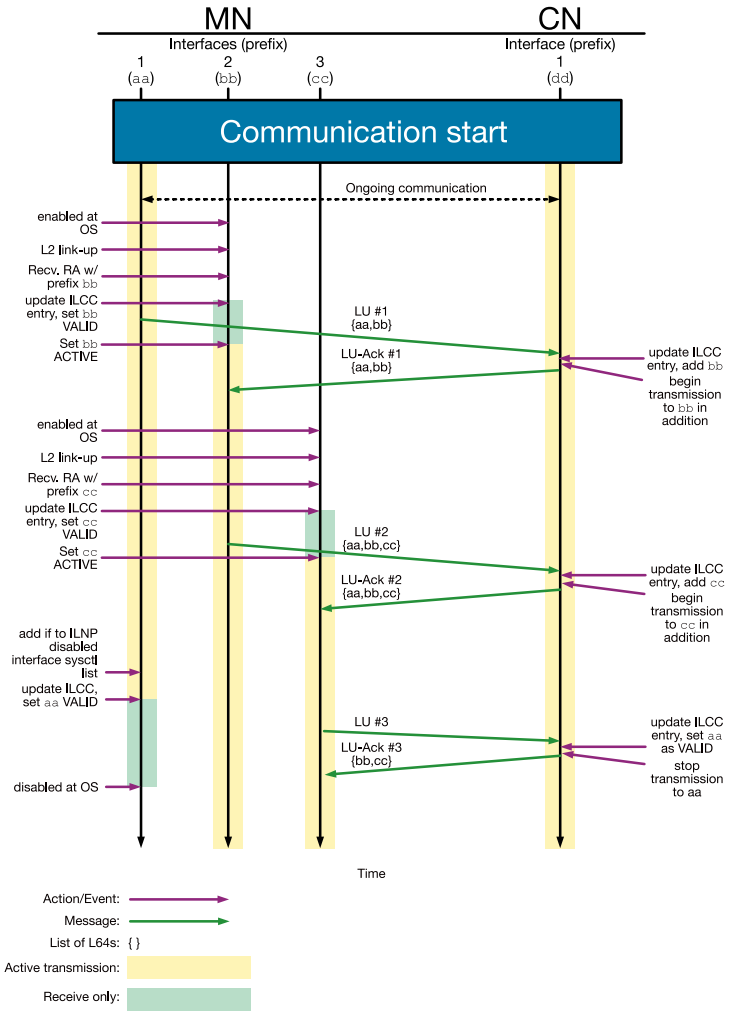


Figure 7. A timeline diagram showing an example of a mobility–multihoming duality scenario. The MN has three interfaces, and the CN has one interface. The MN starts communication only on network aa. The MN and the CN begin a communication session using a single interface on both sides. The MN activates interface 2, receives L64 (IPv6 prefix) `bb`, sends an LU, and sets the new L64 as ACTIVE. The CN responds with an LU-Ack, acknowledging the new set of L64 values that the MN now has. The MN continues to activate another interface, which is also signalled to the CN. The MN then lists the first interface in the `net.ilnp6.disabled_interface sysctl` list, triggering another LU and state changes in the ILCC. After the CN acknowledges the removal of the first interface, the MN removes the interface.

The following describes the procedure of the mobility–multihomed scenario that was executed, and it is also summarised in Figure 8:

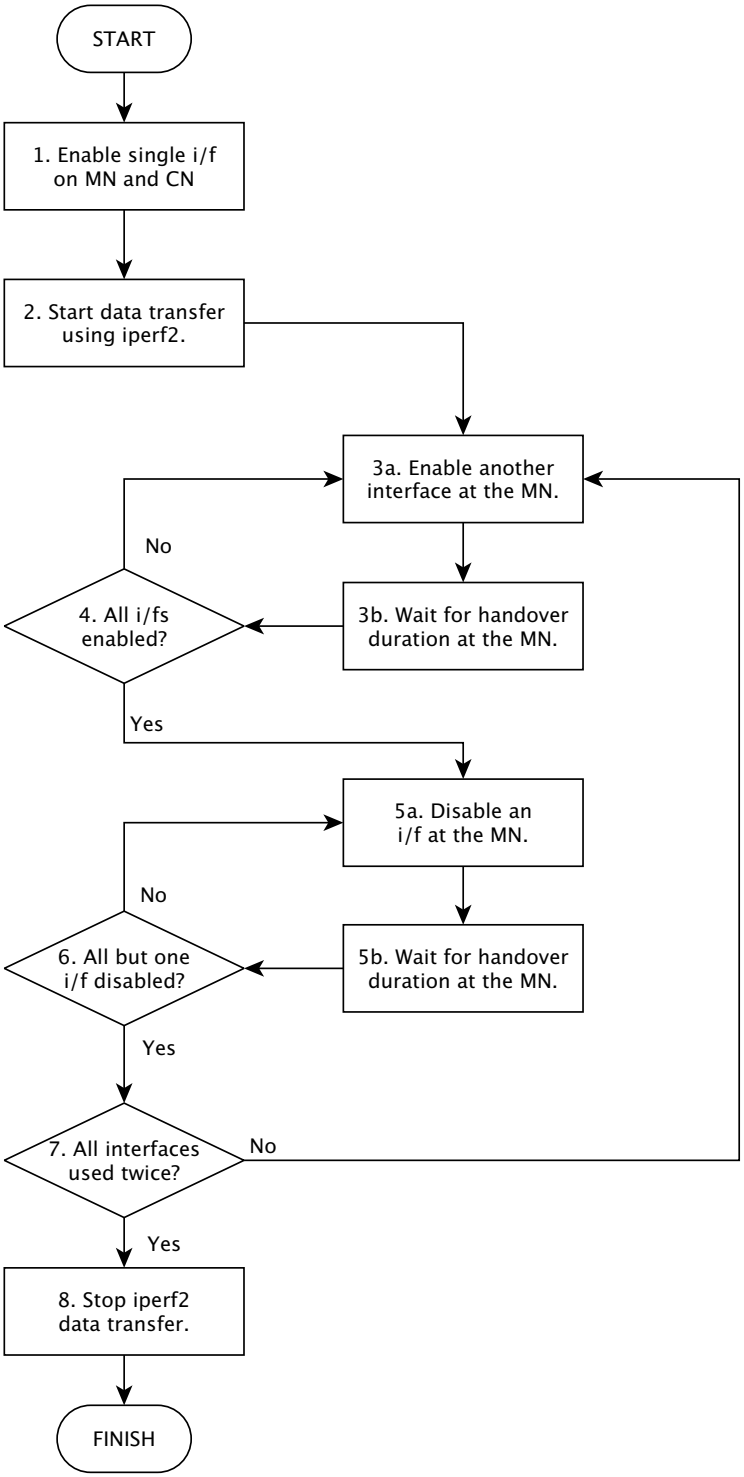


Figure 8. The procedure for a data collection run. Both CN and MN initially have only a single interface (i/f) enabled. `iperf2` is started with bi-directional data transfer. Additional interfaces are enabled at the MN until all interfaces are enabled. Then, interfaces are disabled at the MN until only a single interface remains enabled. This is repeated so that all additional interfaces have been enabled/disabled twice.

1. Enable one interface on both the CN and the MN
   Specifically, the MN will enable interface 1 initially.
2. Begin bi-directional data transfer between two hosts using `iperf2`. Note that the `iperf2` binary was not modified: it was the standard binary operating as if it was using the IPv6.
3. Trigger the MN's movement scenario, which enables additional interfaces.
   (a) Enable another interface.
   (b) Wait for handover duration.
4. Repeat step 3 until all interfaces are up and enabled.
   At this point, the MN has three interfaces enabled for the first time.
5. Trigger the MN's movement scenario disabling interfaces.
   (a) Disable another interface.
   (b) Wait for handover duration.
6. Repeat step 5 until all but one interface is up.
7. Repeat steps 3–5 once such that there are two occasions where all interfaces are up, and then to return to the single-interface state.
8. Terminate the data transfer.

In this experiment, TCP and UDP flows generated using `iperf2` were used. In addition, three artificial delays were induced in the configuration to emulate different Internet scenarios, as described in Table 7. The National and Intercontinental RTT values were chosen based on measurements conducted by the authors from their home networks in the UK. The order of magnitude difference between delay values also presents a good scope for a performance comparison.

Table 7. The three delay scenarios for evaluation—LAN, National, Intercontinental.

Delay [ms]	Description
0	*LAN* equivalent (actual delay ∼1 ms or less)
20	*National* delay equivalent.
200	*Intercontinental* delay equivalent

5.4. Metrics

To evaluate the performance and the capability of the approach, the following metrics were used in experiments across the three sets of evaluations:

- Throughput.
- Packet loss.
- Packet misordering.
- Sequence number analysis.

To measure packet-level metrics, packets were captured using `tcpdump`, saved as a `pcap` file—both at the sender and the receiver—and then post-processed.

A throughput measurement for a run was taken using the sum of the individual relevant IPv6 packet payload size, p, for the run divided by the duration of the run.

$$\text{throughput} = \frac{\sum p}{t_{\text{flow finish}} - t_{\text{flow start}}} \tag{5}$$

The IPv6 payload size was calculated from the packet length field in the IPv6 packet header. The duration of the run was measured between the first and the last packet of the flow. A flow was determined by packets filtered using port numbers, a transport layer protocol, and a source/destination address (IL-V) to correctly identify the flow that was received at the host. The throughput measurement was what was received at the CN or MN to observe successfully transmitted packets.

Analysis of sequence number observations was used to determine the loss, misordering, and continuity of packet flows.

With UDP-based traffic, misordering was identified using sequence numbers in the payload part of the captured UDP packets. `iperf2` included a sequence number in every packet in the first four bytes of the `iperf2` UDP payload.

TCP traffic has sequence numbers in its header that can be observed at the sender and receiver. So, duplicate packets can be detected. Duplicate packets might be transmitted as the flow is perturbed when connectivity changes either due to movement or due to network connectivity being added or removed. Overall, we were concerned about the proportion of duplicate packets that were transmitted, as this is clearly an overhead—whether that duplicate is a data packet or an acknowledgement.

$$\text{duplicate ratio} = \frac{n_{\text{duplicate packets sent}}}{n_{\text{total packets sent}}} \tag{6}$$

With UDP-based traffic, loss was derived from the difference between the total sent packets and the total received packet count. Since UDP-based traffic does not have retransmissions, observing the packet counts at the sender and the receiver was sufficient to detect loss.

Misordering detection is similar with both UDP and TCP. Misordering was detected by observing when the difference between the sequence number of a packet had decremented rather than incremented, and if it had decremented, this meant that the sequence number had not wrapped.

Both loss and misordering are presented as ratios. The loss ratio is the proportion of the number of lost packets to the total number of packets sent.

$$\text{loss ratio} = \frac{n_{\text{packets lost}}}{n_{\text{total packets sent}}} \tag{7}$$

The misordering ratio is the proportion of the number of misordered packets to the total number of packets received carrying the transport layer payload.

$$\text{misordering ratio} = \frac{n_{\text{packets misordered}}}{n_{\text{total packets received}}} \tag{8}$$

Note that the loss ratio was only used for UDP, as TCP retransmits lost packets, so the duplicate ratio was used for TCP instead, as noted above.

6. Results of Evaluation

ILNP successfully provided smooth handovers transitioning from single-homed to a multihomed load-shared state and back to single-homed as it moved across the different network PoAs.

Summary results are shown in Figure 9 on page 26 and Figure 10 on page 27. These allow comparison between TCP and UDP data as general comaprison of packet flow behaviour.

Typical runs from the results for the lab scenario are also presented. These show, respectively, the individual runs of measurements for TCP and UDP. These provide a more intuitive view of the operation of mobility and multihoming duality with load-sharing functions. They show how the throughput varied over time, how the share of throughput was distributed across the multiple paths/PoAs used, and how the change in connectivity was initiated by the use of LU messages. The example runs are from the scenario with no additional delay, i.e., the LAN scenario:

- TCP at the CN in Figure 11 and MN in Figure 12 on page 28;
- UDP at the CN in Figure 13 and MN in Figure 14 on page 30.

For TCP in Figure 11 (for the CN) and Figure 12 (for the MN) on page 28, we note the following:

- In the top graph of each Figure, there was a smooth transition of traffic as additional paths/PoAs were added and removed (top-3 facets) at both the CN and MN, as well as the consistent aggregate throughput (lower facet).
- In the bottom graph, the progression of sequence numbers for TCP showed little interruption (e.g., loss) to the flow of traffic at both the CN and MN.
- The similarity of the figures between the CN (Figure 11) and MN (Figure 12) shows the consistency in operation for TCP over ILNP.

For UDP, we see a similar pattern in the results for Figure 13 (for the CN), and Figure 14 (for the MN) on page 30: a smooth transition of traffic and no loss.

We also present aggregated results in the subsections below from 20 runs for each of TCP and UDP, and for each of the three delay configurations (as in Table 7—0 ms, 20 ms, and 200 ms). The flows were bi-directional, and both were used in the aggregated results, which were labelled as a CN or a MN according to where the observations were made.

Note that we present the results for the TCP and UDP together to show the consistent behaviour from ILNP at the network layer only. The results for TCP and UDP are not directly comparable, as TCP has complex characteristics and behaviour compared to UDP. This difference is also why we used different metrics, as discussed in Section 5.4.

6.1. TCP Misordering

Figure 9a on page 26 shows misordered packet counts for the TCP scenario. Across the different delay configurations and in both directions, TCP over ILNP performed consistently, with negligible misordering observed.

6.2. TCP Duplicate Packets

Figure 9b on page 26 shows the TCP duplicate packets that were observed. As for the misordering ratio across the different delay configurations and directions of flow, duplicate packets were negligible.

6.3. TCP Throughput

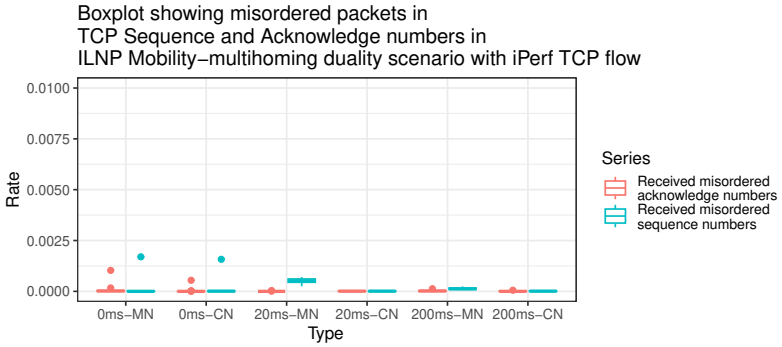
Figure 10a on page 27 shows the throughput observed. With the target throughput of 10 Mbps, the observed throughput remained consistent. While the 200 ms scenario experienced a slightly wider variation in observed throughput, the spread was, at most, 0.04 Mbps (0.4%). Some of this variation was due to the packet pacing used by `iperf2` to maintain the target throughput.

6.4. UDP Packet Delivery Statistics

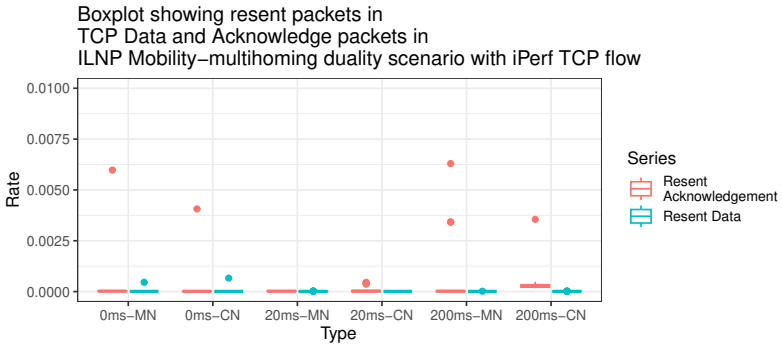
Figure 9c on page 26 shows the UDP packet delivery statistics. Both the misordering and loss observed were less than 1% across all delay configurations and in both directions.

6.5. UDP Throughput

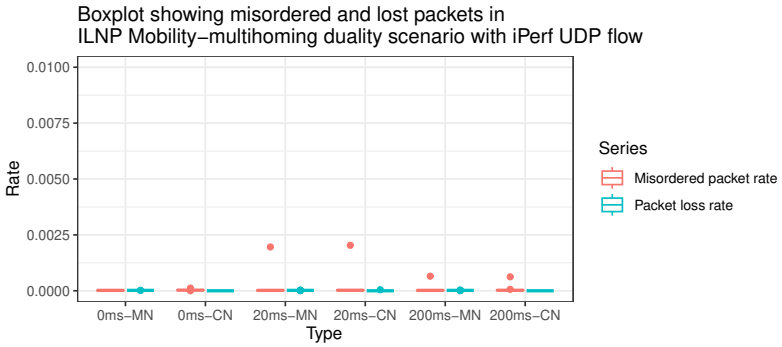
Figure 10b on page 27 shows the throughput observed for the UDP mobility—multihoming scenarios. The throughput observed remained consistent, with outliers within $\pm$0.02 Mbps ($\pm$0.2%) of the target throughput of 10 Mbps. Again, some of this variation was due to the packet pacing used by `iperf2` to maintain the target throughput.



(**a**)

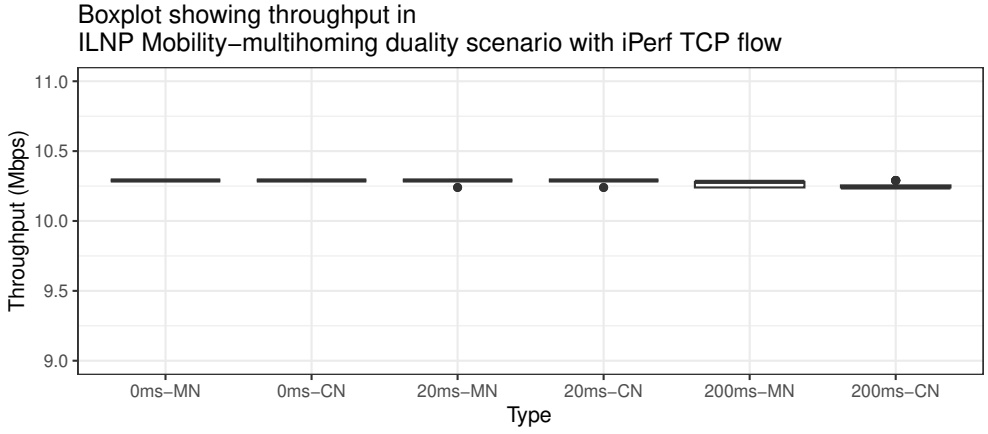


(**b**)

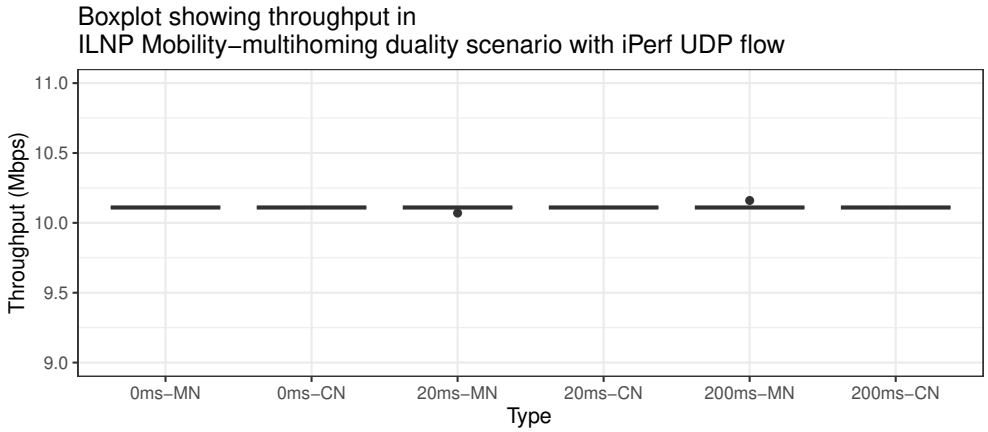


(**c**)

Figure 9. Plots showing packet delivery statistics of TCP and UDP over ILNP. Note that the y axis is in the range of 0.00–0.01, i.e., 0.00–1.00%. The box around the median value is invisible, as the results were consistent across the runs, and the rate remained near zero. In all cases, both the misorder and loss were very low and very consistent across multiple runs. Note that the different transmission characteristics and behaviours for TCP and UDP mean these metrics are not directly comparable. (**a**) TCP misordering ratio based on sequence numbers (data packets) and acknowledgement numbers received. Negligible misordering was observed. (**b**) TCP duplicate ratio based on sequence numbers (data packets) and acknowledgement numbers sent and received.No significant numbers of duplicate packets were observed in the sequence numbers or the acknowledgement numbers. (**c**) UDP packet statistics observed in mobility–multihoming duality scenarios with `iperf2` UDP over ILNP. With all scenarios, both misordering and loss ratio remained low or nil. The small size of the box indicates that there was little to no variation across different runs.

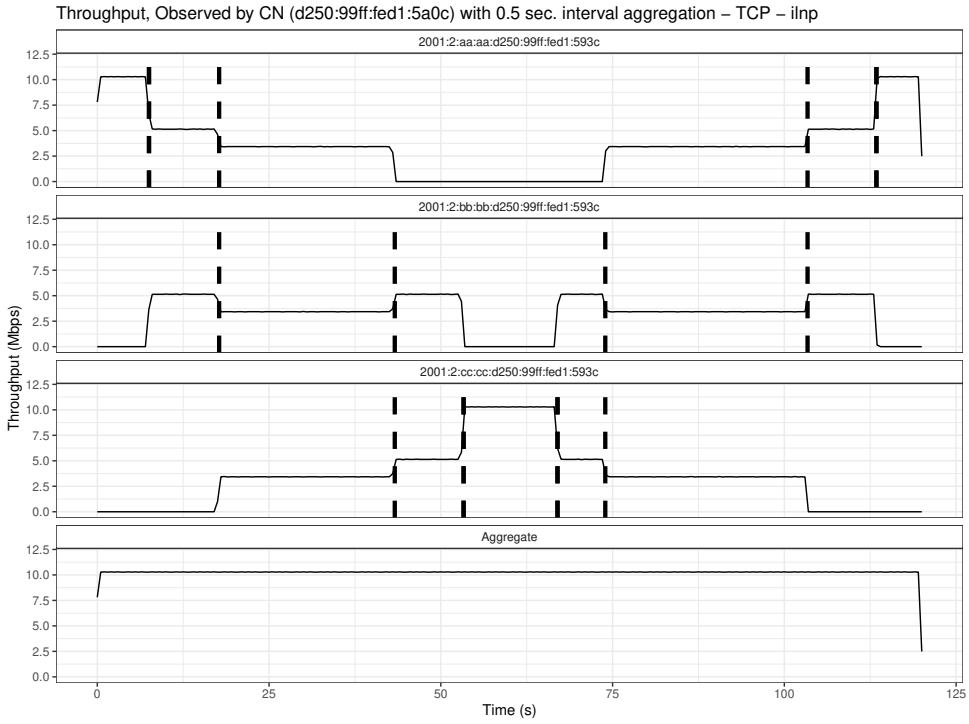


(**a**)

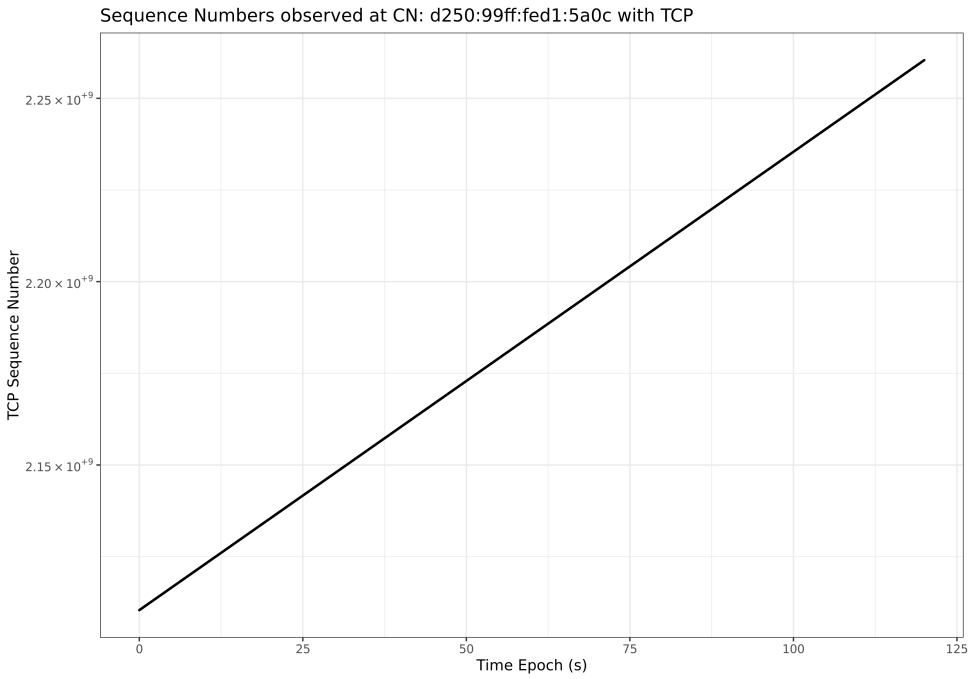


(**b**)

Figure 10. Plots showing the throughput for TCP and UDP over ILNP. Note that, due to the different protocols and their characteristics, these are not directly comparable to each other. (**a**) Throughput observed in the mobility–multihoming duality scenarios with `iperf2` TCP flows over ILNP. Across all delay scenarios, the throughput remained near the 10 Mbps target with little variation, as shown by barely visible 25th and 75th percentile line of the box plot. Note that the *y* axis is in the range of 9.0–11.0 Mbps. The box around the median value is invisible, as the results were consistent across the runs, and the value remained near 10.2 Mbps. (**b**) Throughput observed in mobility–multihoming duality scenarios with `iperf2` UDP flows over ILNP. The throughput remained consistent at around the 10 Mbps target with very few exceptions. Note that the *y* axis is in the range of 9.0–11.0 Mbps. The box around the median value is invisible, as the results were consistent across the runs, and the value remained near 10.1 Mbps.

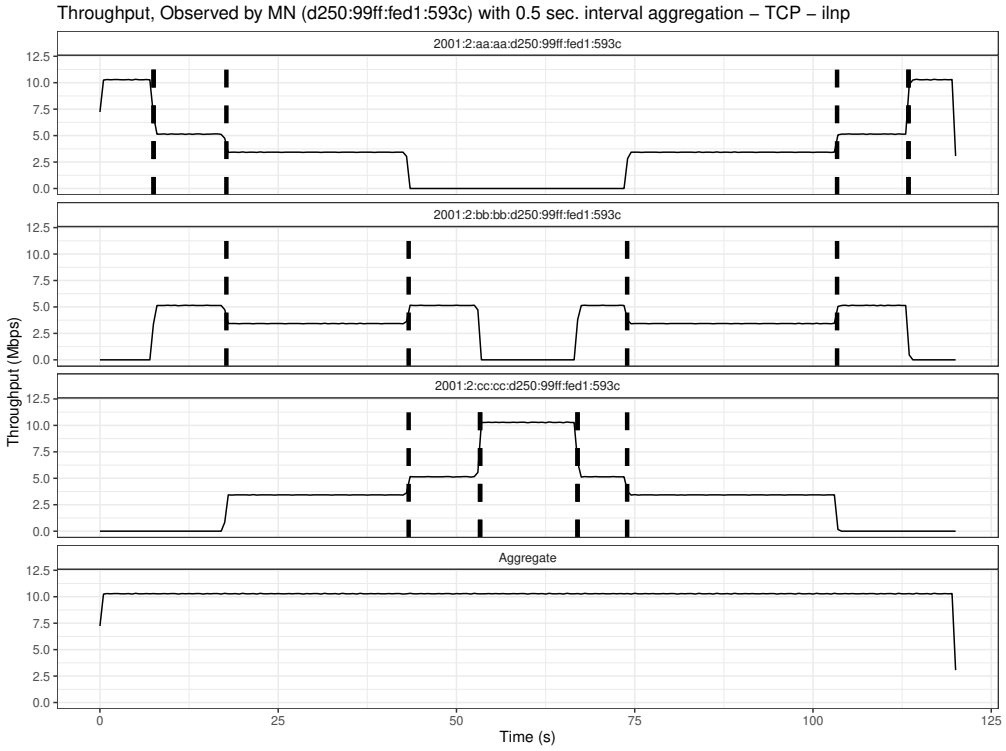


(**a**) Throughput plot.

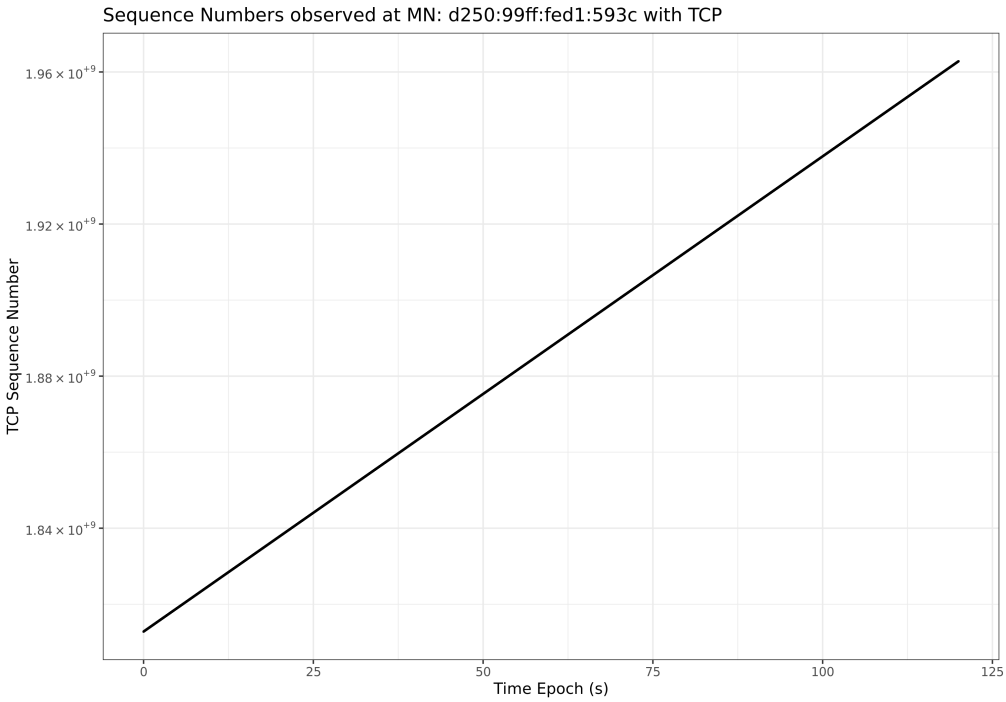


(**b**) Sequence number plot

Figure 11. Plots showing the throughput and sequence numbers observed in typical mobility–multihoming duality `iperf2` TCP scenario evaluations received at the CN. In each column, the top graph is throughput (faceted top to bottom as network aa, bb, cc, and aggregate). There was consistent aggregate throughput (bottom facet), with the expected throughput observed at the respective source/destination IL-V on each network (aa, bb, cc) as expected. The vertical dashed line shows the Locator Update (LU) message event. In each column, the lower graph is the TCP sequence number progression. This also showed consistent increase, indicating a consistent flow and delivery of packets.

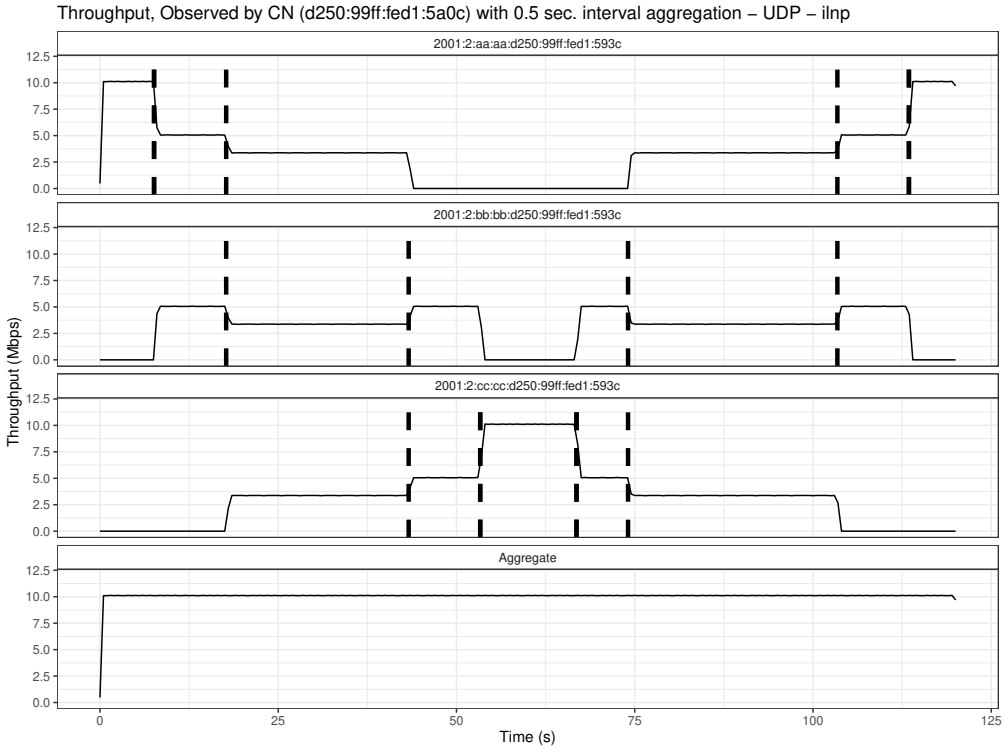


(**a**) Throughput plot.

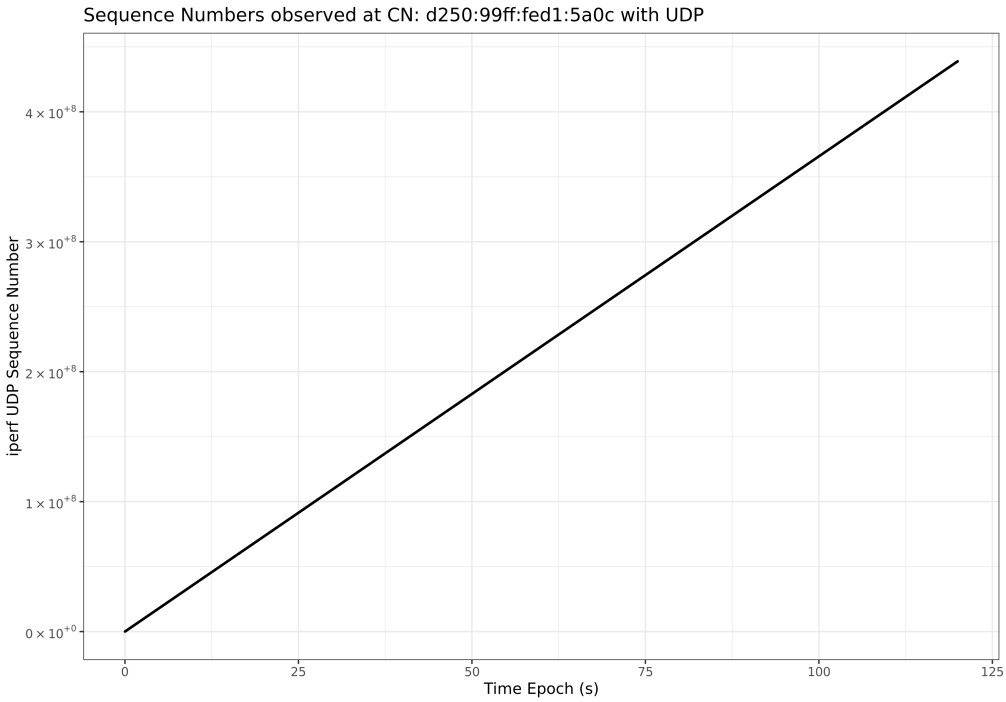


(**b**) Sequence number plot.

Figure 12. Plots showing the throughput and sequence numbers observed in typical mobility–multihoming duality `iperf2` TCP scenario evaluations received at the MN. In each column, the top graph is throughput (faceted top to bottom as network aa, bb, cc, and aggregate). There was consistent aggregate throughput (bottom facet), with the expected throughput observed at the respective source/destination IL-V on each network (aa, bb, cc) as expected. The vertical dashed line shows the Locator Update (LU) message event. In each column, the lower graph is the TCP sequence number progression. This also showed consistent increase, indicating a consistent flow and delivery of packets.

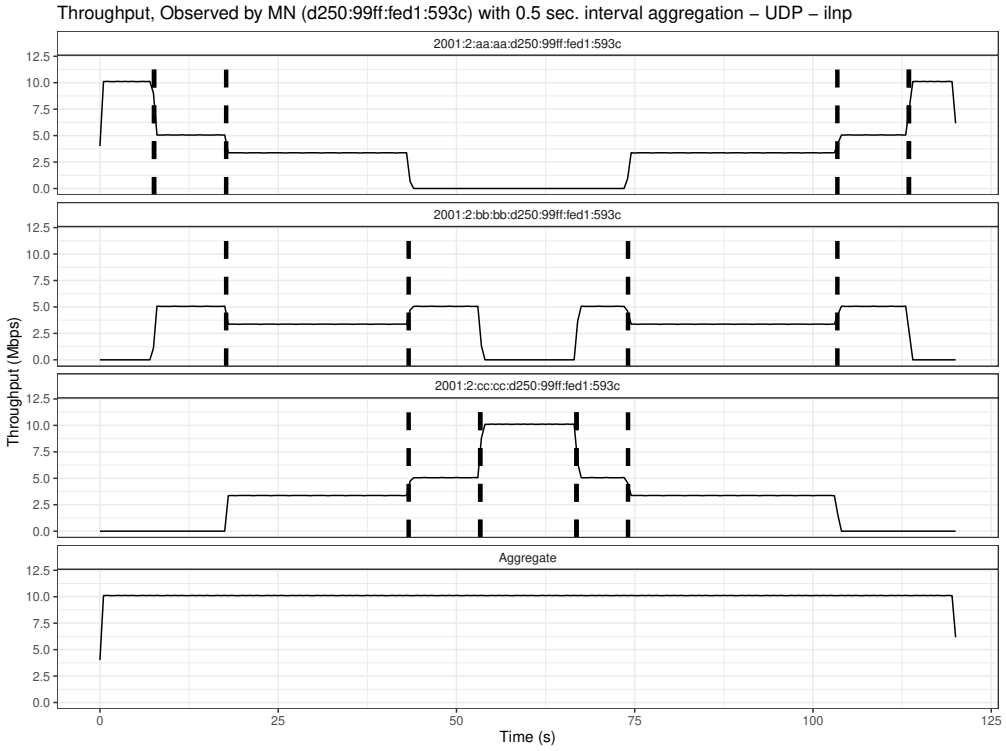


(**a**) Throughput plot.

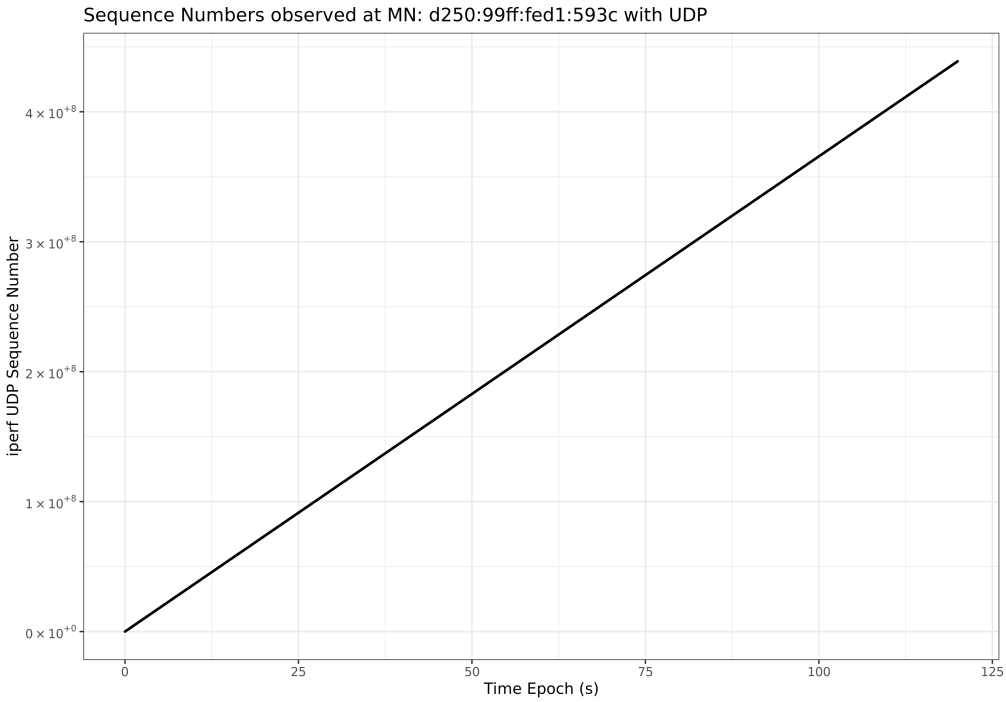


(**b**) Sequence number plot.

Figure 13. Plots showing the throughput and sequence numbers observed in typical mobility–multihoming duality `iperf2` UDP scenario evaluations received at the CN. In each column, the top graph is throughput (faceted top to bottom as network aa, bb, cc, and aggregate). There was consistent aggregate throughput (bottom facet), with the expected throughput observed at the respective source/destination IL-V on each network (aa, bb, cc) as expected. The vertical dashed line shows the Locator Update (LU) message event. In each column, the lower graph is the `iperf2` sequence number progression. This also showed consistent increase, indicating a consistent flow and delivery of packets.



(**a**) Throughput plot.



(**b**) Sequence number plot.

Figure 14. Plots showing the throughput and sequence numbers observed in typical mobility–multihoming duality `iperf2` UDP scenario evaluations received at the MN. In each column, the top graph is throughput (faceted top to bottom as network aa, bb, cc, and aggregate). There was consistent aggregate throughput (bottom facet), with the expected throughput observed at the respective source/destination IL-V on each network (aa, bb, cc) as expected. The vertical dashed line shows the Locator Update (LU) message event. In each column, the lower graph is the `iperf2` sequence number progression. This also showed consistent increase, indicating a consistent flow and delivery of packets.

6.6. Transport Layer Multipath Connectivity

As noted in Section 2, multipath transport does not provide multihomed connectivity, but this is the way applications access multihoming when it is available. So, it is important to understand the action and behaviour of transport protocols. For our experiments, we see that a "normal" TCP (i.e., in this case a TCP CUBIC [52], as implemented in the Linux kernel) can be used successfully over an ILNP without the need for it to be mobile-aware or multihoming-aware.

However, a multipath TCP (MP-TCP) is mature and is deployed, so we present here a brief comparison of our approach with MP-TCP. It was not possible to conduct a complete comparison for the following reasons:

1. MP-TCP is a TCP only solution—comparison can only be made with TCP scenarios with ILNP.
2. MP-TCP is a transport layer solution that assumes that multihoming is in place, while ILNP operates at the network layer to provide multihoming.
3. MP-TCP does not implement mobility–multihoming duality—mobility is not the focus of the design, though it is possible to have mobile capability with MP-TCP [25].
4. For our particular empirical evaluation (see below), the MP-TCP implementation available did not allow for the scheduler to change. MP-TCP's default scheduler was used, while ILNP used our DRR scheduler.

The scenario and testbed was the same as for our ILNP evaluation with the configuration below, including some changes that were needed for the MP-TCP implementation:

- Operating System: Debian Buster (Debian 10) was used, as its default kernel is version 4.19, to match the requirement for the v0.95.2 MP-TCP kernel.
- MP-TCP kernel: v0.95.2 (available from github) https://github.com/multipath-tcp/mptcp/releases/tag/v0.95.2 (accessed on 01 September 2024).
- Path-manager: Set to 'fullmesh'.
- Scheduler: Set to 'default'.
  This is the default scheduler setting recommended. This selects paths based on an RTT and congestion-window.
- Routing: As required, a routing table was created for each interface.
  Existing scripts available on https://multipath-tcp.org (accessed on 01 September 2024) did not support IPv6, so a new script was written to achieve the same routing table configuration.
- `iproute2`: A custom `iproute2` was installed to disable an interface from being used by MP-TCP.
  The script was updated to make use of this in a similar manner to how ILNP has a `sysctl` configuration path to disable an interface.
- `iperf2`: The same version of the `iperf2` program was used for the ILNP evaluation.

As for the ILNP evaluation, the MN moved from single connectivity to three-way connectivity gradually, then returned to single connectivity, and data from 20 runs were collected. This is in direct analogy to the procedure described for our measurements for TCP and UDP in Section 5.3.

Both of the median throughputs recorded were 10.4 Mbps, as shown in Figure 15, which is as might be expected.

If we consider a more detailed examination, due to the MP-TCP scheduler and its behaviour, the throughput distribution across the paths was very different to that of TCP with ILNP. Consider how Figure 16a,b show the throughput values of a typical 120-second run as received at the MN and the CN for the MP-TCP. Clearly, the behaviour of the MP-TCP scheduler led to a 'bursty' traffic pattern across the network connections. Although the aggregate throughput is consistent, the 'bursty' behaviour of the transmission in MP-TCP could cause issues in the traffic management of the networks that the respective interfaces are connected to. Compare this to the smooth transitions with ILNP in Figures 11 and 12. Note how evenly the traffic is distributed for TCP with ILNP compared to MP-TCP.

The issue of the scheduling behaviour for the MP-TCP transmission is a long-standing point of discussion and research. For example, different scheduling has been proposed to improve MP-TCP behaviour for web traffic [53], and yet there is another approach to make the scheduling of MP-TCP transmissions more suitable for mobile networks [54]. A further discussion is presented in Section 7.2.

One of the other issues with MP-TCP is potential security threats. As described in [55], MP-TCP has several security threats, such as session hijacking. Therefore, the current application of MP-TCP has been limited to known good servers with known good clients, such as Apple's mobile platforms' notification delivery [56].

Overall, MP-TCP was designed with different objectives from ILNP. A primary focus of MP-TCP is *resource pooling* at the transport layer to make the best use of capacity across multiple paths and specifically for applications that use TCP. However, ILNP works at the network layer to enable mobility–multihoming duality for *any* transport layer protocol, not just TCP.

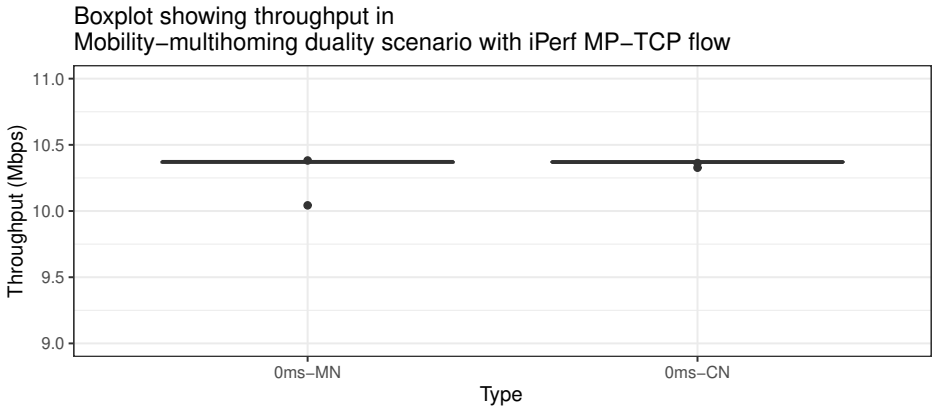
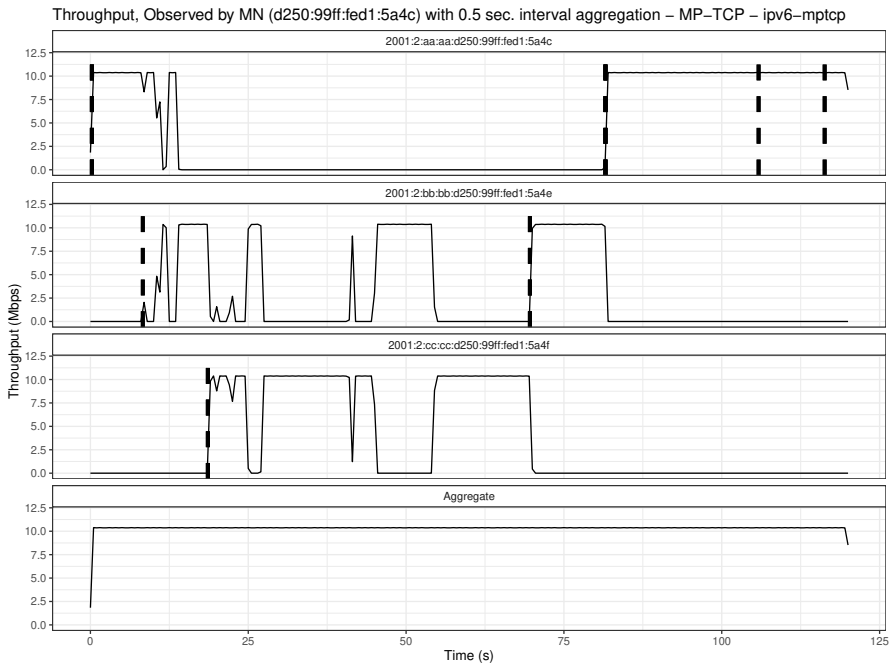
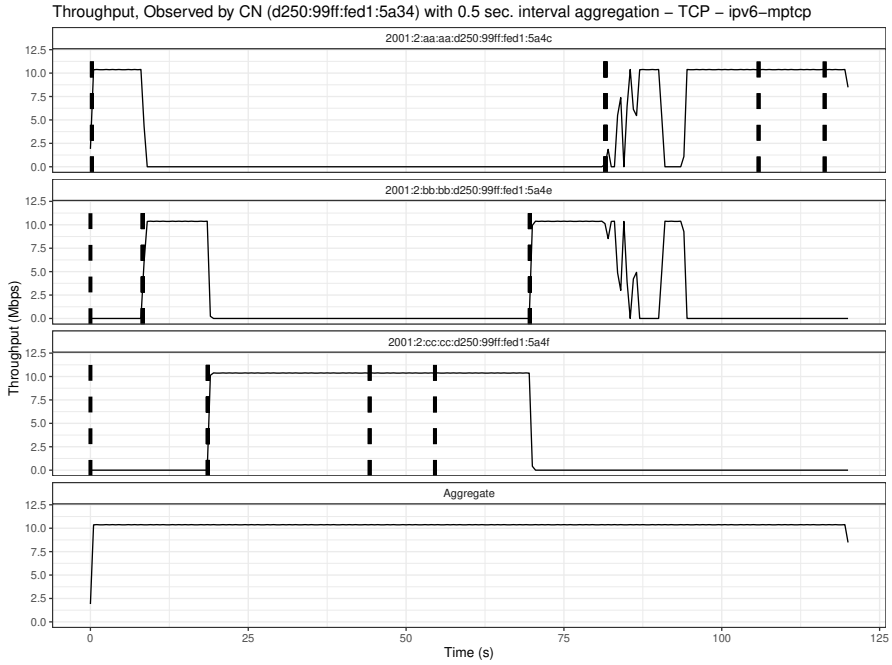


Figure 15. Box plot showing MP-TCP flow for 20 runs with no added delay on path. While the individual interfaces may exhibit 'bursty' behaviour due to the way multipath congestion control algorithm distributes traffic, it satisfies the target load requirement of 10 Mbps.



(**a**)

Figure 16. *Cont.*



(**b**)

Figure 16. MP-TCP typical behaviour on the same testbed as for the ILNP evaluation. The distribution of the throughput is uneven, and changes to throughput on the individual interfaces are 'bursty'. The vertical line shows the protocol level signalling (MP-TCP-specific multipath control plane protocol) to add or remove connectivity received at the respective IPv6 addresses. (**a**) Throughput facet plot of MP-TCP flow received at the MN. The top three plots show the throughput received at the addresses of the respective three interfaces at the MN, and the bottom plot shows the aggregate throughput. (**b**) Throughput facet plot of MP-TCP flow received at the CN. The top three plots show the throughputs received from the addresses of the respective three interfaces at the MN, and the bottom plot shows the aggregate throughput.

7. Discussion

The issue of naming and address semantic overloading is a long-standing problem within the Internet community [57,58]. Indeed, over time, the over-use of IP addresses for various different purposes has caused confusion with address usage. There is now additional complexity in the IP addressing architecture when considering functionality such as mobility and multihoming [59].

ILNP follows the end-to-end principle [60] in considering the naming and addressing issue. The clean, crisp separation of semantics for addressing in ILNP, with clear distinction in the usage between NID values (relevant only at the host for identity and not topologically significant) and L64 values (used for routing, as they are topologically significant). With the use of dynamic bindings for NID and L64 values within the network stack, this allows ILNP to avoid much of the complexity introduced by other proposals, as summarised in Section 2 and Table 1.

However, it is instructive to have a critical discussion of the issues that arise from using the Identifier Locator approach proposed. This is especially instructive in light of the established assumptions for the use of IP addressing and naming that exists today.

7.1. Route Selection at the Host

The IP route metric-based mechanism normally provides immediate change to the route. This is without requiring modifications to the flow lookup process in the outgoing packet processing mechanisms within the packet processing path of the OS kernel. The flow lookup process is executed from distinct processing code paths for UDP and TCP. However,

such a mechanism does not allow multipath transport for a multihomed host to select paths on a per-packet basis.

First, the route metric-based mechanism is less fine-grained and inflexible—as soon as a lower-metric route is added, the entire host's outgoing route handling changes. However, it is not possible to control the precise timing of this change, and it has no relation to the individual packets in the packet processing path.

In order for ILNP to select a source L64 for multipath transport, the outgoing interface must correspond to the L64 selected by ILNP on a per-packet basis. However, the two distinct transport layer packet processing mechanisms select the outgoing interface at the early phase of their packet processing. This phase of packet processing had to be modified in order for ILNP to gain the ability to select the outgoing interface and the route such that the selected source L64 value would be reflected accordingly. The packet processing path involves a route lookup when the UDP and TCP packet headers are being constructed before they are passed onto the IP layer packet processing mechanisms. This is to ensure that the pseudo-header source address (and Checksum calculations) match the actual source address and the outgoing interface. In the context of a conventional IP, the actual source address and the respective outgoing interface do not change; therefore, they are cached after the initial lookup. However, that is not the case with ILNP, especially in the context of multihoming, where the packets in a flow may leave from different interfaces depending on the binding between local L64 values and interfaces.

To enable multipath transport, specifically to allow ILNP to select the correct outgoing interface for each packet, the ILNP code needs full control of the route selection mechanism. In order to realise this, both the UDP and TCP packet processing mechanisms were modified to correctly set the outgoing route and the respective interface. Specifically, by preventing the caching of routes and triggering route lookup for every packet, ILNP gained the ability to control each packet accordingly.

Note that this is all because of the inherent "single-homed" assumptions that are in IP implementations, even though IPv6 should support multihoming.

7.2. Transport Layer Congestion Control

As discussed in Section 2.12 and noted in Table 1, transport layer protocols rely on the multihoming provided at the network layer for multipath transport. However, there is an important function that needs to be considered for packet transmission: congestion control.

Congestion control is considered to be an end-to-end issue and so is typically implemented in transport layer protocols, such as TCP and QUIC. So, transport protocols control packet transmission, yet some level of transmission control is also required at the network level when multihoming is used.

Our results show that multihoming at the network layer with ILNP can be compatible with TCP congestion control. Note that in our experiments, the Linux default TCP variant was used—TCP CUBIC [52]. Figures 10a and 15 show that the overall end-to-end throughput for TCP over ILNP and MP-TCP are equivalent for the same scenario. However, if we consider Figures 12a and 16a, we see that the transmission characteristics are very different. This is due to MP-TCP using transmission control and scheduling at the transport layer and our experiments using DRR at the network layer. In our experiments, we found a smoother distribution of traffic across paths using TCP over ILNP compared to MP-TCP.

Packet transmission scheduling for MP-TCP has been a key design consideration from its inception, and it remains a research issue. This original design study [61] considered that network layer transmission control using an Equal Cost Multipath (ECMP) [62] was compatible and usable with MP-TCP. A comprehensive study of approaches shows that round-robin scheduling at the network layer can also be beneficial for MP-TCP under some circumstances [63]. However, overall, at the time of writing, there is no single packet scheduling mechanism that is considered as a general approach for use with congestion control for multipath traffic.

Overall, the use of mutihoming and multipath transmission adds a new dimension when considering the design of congestion control algorithms. There is already interaction between the network layer and the transport layer in order to obtain the benefits of Explicit Congestion Notification (ECN) in QUIC and TCP [64–66]. There are also numerous TCP congestion control algorithms with different characteristics for different scenarios [67]. So, we can in the future expect the definition of a number of new packet scheduling and congestion control algorithms for multihomed systems using multipath transport protocols.

7.3. Mobile Servers

If a server is mobile and changes its connectivity (which could also be due to dynamic multihoming), the remote hosts that try to access it must be able to resolve at least one of its current IL-Vs in order to connect to it. ILNP has defined DNS records [68] that are supported in commercial software such as BIND. However, when an L64 value for a server changes, it needs to be updated within the DNS entry for that server. The DNS already has a mechanism defined for a Secure Dynamic Update [69] for performing such updates, and it is widely implemented. However, this would need to be enabled, configured, and maintained for those sites that wanted to have mobile servers.

7.4. Firewalls and Middleboxes

ILNP packets look like IPv6 packets "on the wire". So, for anywhere that an IPv6 is accepted by firewalls, then ILNP should also be accepted. However, there are differences in the way ILNP works that could cause difficulties with firewalls for ILNP unless those firewalls are configured to understand these differences. This might especially be the case for enterprise networks. These will typically have more restrictive firewalling compared to domestic (homed) networks, for example. The key differences in ILNP "on-the-wire" compared to an IPv6 are the following:

- The ICMPv6 messages for ILNP Locator Update (LU) signalling [40] may need to be explicitly enabled for firewalls that have "default deny" policies for those packets types that are not required for operational use.
- The ILNP Nonce [41] implemented as Destination Option extension header for IPv6 needs to be sent from host to host for an ILNP packet flow. Some firewalls might disallow such extension headers.
- The TCP checksum computation changes when TCP is used over ILNP to exclude the L64 value, which is mutable. This is not a problem for an ILNP-capable host, but a firewall or middlebox that checks the TCP Checksum and is not ILNP-aware will see the Checksum computation as incorrect. This is a layer 4 (transport layer) packet check and so will be used where there are stricter site security policies for firewalls.

The exact behaviour of firewalls and middleboxes varies, so it might not be possible to predict ahead of deployment where ILNP will and will not work. However, our experiments in this paper show that at the network level at least, commercial equipment can support the transmission of ILNP packets as if they are IPv6 packets.

7.5. Incremental Deployment

As ILNP is an end-to-end protocol, deployment will be via updates to hosts. As we have discussed in Section 2, this model for deployment is, arguably, the least disruptive and most scalable in today's environment of over-the-air updates. ILNP is designed to fall back to IPv6 operation when ILNP operation is not possible—specific procedures for this are defined [43,49]. Additionally, the use and operation of ILNP for individual installations can be controlled via `sysctl` calls. This means that users, from individual users at home to system administrators for managed network installations, can control the installation and use of ILNP.

7.6. Combatting Ossification at the Network Layer and Transport Layer

Over time, the increasing use of the Internet has led to a resistance to change. Change causes disruption, and disruption is unwelcome for those who rely on the services provided by the Internet.

At least part of this resistance to change is because of a reliance on in-network functions implemented in middleboxes and proxy systems. These various middleboxes and proxy systems assume protocols operated in a fixed way, which in turn has lead to concerns over the ossification of protocol design and behaviour [70]. This is true for both the network layer and the transport layer [71]. Indeed, this is part of the motivation for the development of QUIC over UDP. There are also related concerns over ossification being linked to the centralisation of service provision and network functions, as well as its overall impact on society [72].

However, ILNP takes a decentralised, end-to-end approach, and it strives to be in keeping with the spirit of the original design philosophy of the Internet [73]. Note that in our testbed, we uncovered the following:

- We observed that the commercial routers used were not ILNP-aware but treated ILNP packets as if they were IPv6 packets.
- The commercial routers only provided unicast routing and forwarding, with no special routing or addressing provision needed for mobility or multihoming to work.
- As only normal unicast routing and addressing were used, the provision and deployment of ILNP, including mobility and multhoming, could use provider-aggregated addressing, so this will have no detrimental impact on upstream or core routing state.
- As ILNP is a network layer architecture, which is backwards compatible with the IPv6, it will work for any transport protocol, including existing and new transport protocols. It should be possible for it to work in complement to QUIC (which works over UDP), and it could be possible even to adapt MP-TCP to operate over ILNP.
- As ILNP is deployable via host updates, it is incrementally deployable and does not require major infrastructure changes.

Overall, we believe that the ILNP approach as demonstrated could yield at least some reprieve from the concerns within the Internet community around the ossification and centralisation of the Internet infrastructure.

8. Conclusions

We have demonstrated a design and implementation for harmonised mobility and multihoming for the Internet Protocol based on an approach implemented as a superset of IPv6. We have considered the key addressing and naming issues for the architecture, as well as the dynamic bindings of layer-specific names to objects in a network architecture. We have shown that it is possible to implement a *mobility–multihoming duality* by managing those names and object bindings.

Specifically, we have shown the following:

- That current approaches to mobility and multihoming are restricted in their architectural approaches by relying on the re-use of IP addresses within their design.
- The use of crisp, well-defined names for nodes (hosts) and networks. Along with well-defined dynamic bindings for those names within the network stack, we have a harmonised approach. This realises a model of network connectivity that is flexible enough to provide both mobility and multihoming together.
- The use of ILNP with a simple signalling protocol and re-use of the IPv6 address space in IPv6 packets can be used to implement such a model of dynamic bindings for addressing. We have shown in our testbed experiments that such a model will operate over existing IPv6 networks. We have also shown that it can be used with existing TCP and UDP protocols, with good performance across end-to-end delays from $\sim$0 ms to 200 ms.

- With a mechanism implemented at the network layer, applications using TCP and UDP can benefit from the use of ILNP without needing to be redesigned or re-engineered. Our experiments used traffic flows generated by `iperf2` v2.0.9, the binary for which was not recompiled or modified in any way.

Overall, our experiments suggest that existing applications could make use of harmonised mobility and multihoming capability with ILNP over an IPv6 network.

Future Work

In our discussion of Section 7 we have highlighted some potential challenges in real deployment scenarios for ILNP. The greatest of these from a practical point of view is that of deployment in scenarios with firewalls and middleboxes. These might use more detailed inspection of traffic, especially in an enterprise environment. We can imagine a user-space tool that writes to and reads from the network wire images of ILNP packets directly. Such a tool would allow for the transmission and reception of ILNP packets so that testing can be performed without having to deploy the ILNP kernel. Software libraries and APIs that allow low-level access to network interfaces are already available that would permit such a tool to be constructed.

For practical purposes, it would be useful to measure the performance of some common applications running over ILNP. Our measurements in this paper show that TCP and UDP can operate over ILNP via the existing `socket(3)` API. That API forms the basis of many other software libraries and APIs implemented in popular high-level languages, such as Python, Rust, and Go.

From a scientific perspective, we have ongoing work related to improvements in security and privacy for IP-based applications. These also exploit the clean naming and dynamic binding mechanisms provided by the ILNP architecture [46,48]. Security and privacy are important areas for further improvement, as the Internet plays an increasingly prevalent role in service and information provision for many aspects of society and industry.

Author Contributions: Conceptualization, R.Y. and S.N.B; methodology, R.Y. and S.N.B.; software, R.Y.; validation, R.Y. and S,N.B; formal analysis, R.Y. and S.N.B.; investigation, R.Y.; resources, S.N.B.; data curation, R.Y.; writing—original draft preparation, R.Y. and S.N.B.; writing—review and editing, R.Y. and S.N.B.; visualization, R.Y. and S.N.B.; supervision, S.N.B.; project administration, S.N.B.; funding acquisition, S.N.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Summary data are contained within the article. The research data underpinning this publication can be accessed at [74].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
BGP	Border Gateway Protocol
CN	Correspondent Node
CoA	Care-of Address
DNS	Domain Name System
DoS	Denial of Service
DRR	Deficit Round-Robin
ECMP	Equal Cost Multipath
ECN	Explicit Congestion Notification

EID	End system Identifier
EVPN	Ethernet VPN
FA	Foreign Agent
FQDN	Fully Qualified Domain Name
HA	Home Agent
HoA	Home Address
HIP	Host Identity Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineering
IETF	Internet Engineering Task Force
IID	Interface Identifier
ILCC	ILNP Communication Cache
ILNP	Identifier Locator Network Protocol
IL-V	Identifier Locator Vector
IP	Internet Protocol
IPv6	Internet Protocol version 6
L64	Locator 64 bits (for ILNP)
Loc	Locator
LISP	Locator Identifier Separation Protocol
LISP-MN	LISP Mobile Node
LMA	Local Mobility Anchor
MAC	Media Access Control
MAG	Mobility Access Gateway
MIPv6	Mobile IPv6
MN	Mobile Node
MPLS	Multiprotocol Label Switching
MP-TCP	Multipath Transmission Control Protocol
ND	Neighbor Discovery
NIC	Network Interface Card
NID	Node Identifier
OAM	Operation, Administration, and Management
PoA	Point of Attachment
QUIC	now a name (but previously Quick UDP Internet Connections)
RA	Router Advertisement (for IPv6)
RO	Route Optimisation (for MIPv6)
RS	Router Solicitation (for IPv6)
RTT	Round Trip Time
SPoF	Single Point of Failure
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ULID	Upper-Layer Identifier
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network

References

1. Weiser, M. Some Computer Science Issues in Ubiquitous Computing. *Commun. ACM* **1993**, *36*, 75–84. https://doi.org/10.1145/159544.159617.
2. Portoles, M.; Ashtaputre, V.; Maino, F.; Moreno, V.; Farinacci, D. LISP L2/L3 EID Mobility Using a Unified Control Plane. In *Internet Draft (Work-in-Progress)*; IETF: Fremont, CA, USA, 2024.
3. Dhraief, A.; Montavont, N. Toward Mobility and Multihoming Unification—The SHIM6 Protocol: A Case Study. In Proceedings of the WCNC 2008—2008 IEEE Wireless Communications and Networking Conference, Las Vegas, NV, USA, 31 March–3 April 2008; pp. 2840–2845. https://doi.org/10.1109/WCNC.2008.497.
4. ETSI; LTE. Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2. In *Technical Specification*; TS 136 300 V17.6.0, 3GPP TS 36.300 version 17.6.0 Release 17; ETSI: Sophia Antipolis, France, 2024.
5. Ford, A.; Raiciu, C.; Handley, M.; Bonaventure, O.; Paasch, C. *TCP Extensions for Multipath Operation with Multiple Addresses*; RFC 8684(PS); IETF: Fremont, CA, USA, 2020.

6. Bonaventure, O.; Huitema, C. Multipath Extension for QUIC. In *Internet Draft (Work in Progress)*; Liu, Y., De Coninck, Q., Kuehlewind, M., Eds.; IETF: Fremont, CA, USA, 2024.
7. Farinacci, D.; Maino, F.; Fuller, V. *The Locator/ID Separation Protocol (LISP)*; RFC 9300(PS); Cabellos, A., Ed.; IETF: Fremont, CA, USA, 2022.
8. Nordmark, E.; Bagnulo, M. *Shim6: Level 3 Multihoming Shim Protocol for IPv6*; RFC 5533(PS); IETF: Fremont, CA, USA, 2009.
9. Rahman, M.S.; Atiquzzaman, M. SEMO6—A multihoming-based seamless mobility management framework. In Proceedings of MILCOM 2008—2008 IEEE Military Communications Conference, San Diego, CA, USA, 17–19 November 2008; pp. 1–7. https://doi.org/10.1109/MILCOM.2008.4753382.
10. Abley, J.; Bagnulo, M.; Garcia-Martinez, A. *Considerations on the Application of the Level 3 Multihoming Shim Protocol for IPv6 (Shim6)*; RFC 6629(I); IETF: Fremont, CA, USA, 2012.
11. Garcia-Luna-Aceves, J.; Sevilla, S. A Simple Solution to Scale-Free Internet Host Mobility. In Proceedings of ICCCN 2017—26th International Conference on Computer Communication and Networks, Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–9. https://doi.org/10.1109/ICCCN.2017.8038446.
12. Johnson, D.; Arkko, J. *Mobility Support in IPv6*; RFC 6275(PS); Perkins, C., Ed.; IETF: Fremont, CA, USA, 2011.
13. Perkins, C. (Ed.) *IP Mobility Support for IPv4, Revised*; RFC 5944(PS); IETF: Fremont, CA, USA, 2010.
14. Leung, K.; Devarapalli, V.; Chowdhury, K.; Patil, B. *Proxy Mobile IPv6*; RFC 5213(PS); Gundavelli, S., Ed.; IETF: Fremont, CA, USA, 2008.
15. Bernardos, C.J. (Ed.) *Proxy Mobile IPv6 Extensions to Support Flow Mobility*; RFC 7864(PS); IETF: Fremont, CA, USA, 2016.
16. Zhu, R.Z.; Wakikawa, L.Z. *A Survey of Mobility Support in the Internet*; RFC 6301(I); IETF: Fremont, CA, USA, 2011.
17. Koodli, R. (Ed.) *Mobile IPv6 Fast Handovers*; RFC 5568(PS); IETF: Fremont, CA, USA, 2009.
18. Soliman, H.; Castelluccia, C.; ElMalki, K.; Bellier, L. *Hierarchical Mobile IPv6 (HMIPv6) Mobility Management*; RFC 5380(PS); IETF: Fremont, CA, USA, 2008.
19. Heer, T.; Jokela, P.; Henderson, T. *Host Identity Protocol Version 2 (HIPv2)*; RFC 7401(PS); Moskowitz, R., Ed.; IETF: Fremont, CA, USA, 2009.
20. Vogt, C.; Arkko, J. *Host Mobility with the Host Identity Protocol*; RFC 8046(PS); Henderson, T., Ed.; IETF: Fremont, CA, USA, 2017.
21. Vogt, C.; Arkko, J. *Host Multihoming with the Host Identity Protocol*; RFC 8047(PS); Henderson, T., Ed.; IETF: Fremont, CA, USA, 2017.
22. Komu, M. *Host Identity Protocol Architecture*; RFC 9063(PS); Moskowitz, R., Ed.; IETF: Fremont, CA, USA, 2021.
23. Henderson, T.; Nikander, P.; Komu, M. *Using the Host Identity Protocol with Legacy Applications*; RFC 5338(E); IETF: Fremont, CA, USA, 2008.
24. Laganier, J.; Eggert, L. *Host Identity Protocol (HIP) Rendezvous Extension*; RFC 8004(PS); IETF: Fremont, CA, USA, 2016.
25. Raiciu, C.; Niculescu, D.; Bagnulo, M.; Handley, M.J. Opportunistic mobility with multipath TCP. In Proceedings of the Mobiarch 2011: 6th International Workshop on Mobility in the Evolving Internet Architecture, New York, NY, USA, 28 June 2011. https://doi.org/10.1145/1999916.1999919.
26. Sun, Y.; Cui, Y.; Wang, W.; Ma, T.; Ismailov, Y.; Zheng, X. Mobility support in Multi-Path TCP. In Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 27–29 May 2011; pp. 195–199. https://doi.org/10.1109/ICCSN.2011.6014033.
27. Komu, M.; Sethi, M.; Beijar, N. A survey of identifier-Locator split addressing architectures. *Comput. Sci. Rev.* **2015**, *17*, 25–42. https://doi.org/10.1016/j.cosrev.2015.04.002.
28. Li, T. (Ed.) *Recommendation for a Routing Architecture*; RFC 6115(I); IRTF: Landwirt, Forsch, 2011.
29. Kaufman, C.; Hoffman, P.; Nir, Y.; Eronen, P. *Internet Key Exchange Protocol Version 2 (IKEv2)*; RFC 5996(PS); IETF: Fremont, CA, USA, 2010.
30. Devarapalli, V.; Eronen, P. *Secure Connectivity and Mobility Using Mobile IPv4 and IKEv2 Mobility and Multihoming (MOBIKE)*; RFC 5266(BCP)/BCP136; IETF: Fremont, CA, USA, 2008.
31. Eronen, P. (Ed.) *IKEv2 Mobility and Multihoming Protocol (MOBIKE)*; RFC 4555(PS); IETF: Fremont, CA, USA, 2006.
32. Sajassi, A.; Aggarwal, R.; Uttaro, J.; Bitar, N.; Henderickx, W.; Isaac, A. *Requirements for Ethernet VPN (EVPN)*; RFC 7209(I); IETF: Fremont, CA, USA, 2014.
33. Komplella, K.; Rekhter, Y. (Eds.) *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*; RFC 4671(PS); IETF: Fremont, CA, USA, 2007.
34. Lasserre, M.; Komplella, K. (Eds.) *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*; RFC 4672(PS); IETF: Fremont, CA, USA, 2007.
35. Aggarwal, R.; Bitar, N.; Isaac, A.; Uttaro, J.; Drake, J.; Henderickx, W. *BGP MPLS-Based Ethernet VPN*; RFC 7432(PS); Sajassi, A., Ed.; IETF: Fremont, CA, USA, 2015.
36. Abley, J.; Black, B.; Gill, V. *Goals for IPv6 Site-Multihoming Architectures*; RFC 3582(I); IETF: Fremont, CA, USA, 2003.
37. Miles, D.; Matsushima, S.; Okimoto, T.; Wing, D. *IPv6 Multihoming without Network Address Translation*; RFC 7157(I); Troan, O., Ed.; IETF: Fremont, CA, USA, 2014.
38. Nordmark, E.; Li, T. *Threats Relating to IPv6 Multihoming Solutions*; RFC 4218(I); IETF: Fremont, CA, USA, 2005.
39. Narten, T.; Nordmark, E.; Simpson, W.; Soliman, H. *Neighbor Discovery for IP Version 6 (IPv6)*; RFC 4861(DS); IETF: Fremont, CA, USA, 2007.

40. Atkinson, R.; Bhatti, S.N. *ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)*; RFC 6743(E); IRTF: Fremont, CA, USA, 2012.
41. Atkinson, R.; Bhatti, S.N. *IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)*; RFC 6744(E); IRTF: Fremont, CA, USA, 2012.
42. Kent, S. *IP Encapsulating Security Payload (ESP)*; RFC 4303(PS); IETF: Fremont, CA, USA, 2005.
43. Atkinson, R.; Bhatti, S.N. *Identifier-Locator Network Protocol (ILNP) Architectural Description*; RFC 6740 (E); IRTF: Fremont, CA, USA, 2012.
44. Cooper, A.; Gont, F.; Thaler, D. *Security and Privacy Considerations for IPv6 Address Generation Mechanisms*; RFC 7721(I); IETF: Fremont, CA, USA, 2016.
45. Gont, F.; Krishnan, S.; Narten, T.; Draves, R. *Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6*; RFC 8981(PS); IETF: Fremont, CA, USA, 2021.
46. Bhatti, S.N.; Haywood, G.; Yanagida, R. End-to-End Privacy for Identity & Location with IP. In Proceedings of NIPAA-21—2nd Workshop on New Internetworking Protocols, Architecture and Algorithms, Virtual Event, 1–5 November 2021; p. 6. https://doi.org/10.1109/ICNP52444.2021.9651909.
47. Kent, S. *Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement*; RFC 7624(I); IETF: Fremont, CA, USA, 2015.
48. Haywood, G.T.; Bhatti, S.N. Defence against side-channel attacks for encrypted network communication using multiple paths. *Cryptography* **2024**, *8*, 22. https://doi.org/10.3390/cryptography8020022.
49. Atkinson, R.; Bhatti, S.N. *Identifier-Locator Network Protocol (ILNP) Engineering Considerations*; RFC 6741(E); IRTF: Fremont, CA, USA, 2012.
50. Yanigida, R. ILNP software version: ilnp-public-1, September 2019. Prototype implementation of ILNP in Linux, kernel version 4.9 LTS. Available at https://ilnp.github.io/ilnp-public-1/, accessed 01 September 2024.
51. Shreedhar, M.; Varghese, G. Efficient Fair Queueing Using Deficit Round Robin. *ACM SIGCOMM Comp. Comm. Rev.* **1995**, *25*, 231–242. https://doi.org/10.1145/217391.217453.
52. Xu, L.; Ha, S.; Rhee, I.; Goel, V. *CUBIC for Fast Long-Distance Networks*; RFC 9438(PS); Eggert, L., Ed.; IETF: Fremont, CA, USA, 2023.
53. Noda, K.; Ito, Y. Proposal of Multi-path TCP Packet Scheduler to Adjust Trade-off between QoS Fluctuation and throughput for WebQoE Improvement. In Proceedings of ICCCS 2019—IEEE 4th International Conference on Computer and Communication Systems, Singapore, 23–25 February 2019; pp. 493–496. https://doi.org/10.1109/CCOMS.2019.8821657.
54. Xing, Y.; Xue, K.; Zhang, Y.; Han, J.; Li, J.; WeiMember, D.S.L. An Online Learning Assisted Packet Scheduler for MPTCP in Mobile Networks. *IEEE/ACM Trans. Netw.* **2023**, *31*, 2297–2312. https://doi.org/10.1109/TNET.2023.3246168.
55. Popat, K.; Kapadia, V.V. Multipath TCP Security Issues, Challenges and Solutions. In Proceedings of the Information, Communication and Computing Technology, New Delhi, India, 8 May 2021; Communications in Computer and Information Science; Bhattacharya, M., Kharb, L., Chahal, D., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 18–32. https://doi.org/10.1007/978-3-030-88378-2_2.
56. Paasch, C.; Bonaventure, O. Multipath TCP: Decoupled from IP, TCP is at last able to support multihomed hosts. *Queue* **2014**, *12*, 40–51. https://doi.org/10.1145/2578508.2591369.
57. Bennett, C.J.; Edge, S.W.; Hinchley, A. Issues in the Interconnection of Datagram Networks. In *Internet Experiment Note (IEN) 1*; ARPA Network Working Group: Ivrea TO, Italy, 1977.
58. Carpenter, B.; Crowcroft, J.; Rekhter, Y. *IPv4 Address Behaviour Today*; RFC 2101(I); IAB: Boulder, CO, USA, 1997.
59. Carpenter, B. IP Addresses Considered Harmful. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 65–69. https://doi.org/10.1145/2602204.2602215.
60. Saltzer, J.H.; Reed, D.P.; Clark, D.D. End-to-end arguments in system design. *ACM Trans. Comput. Syst.* **1984**, *2*, 277–288. https://doi.org/10.1145/357401.357402.
61. Wischik, D.; Raiciu, C.; Greenhalgh, A.; Handley, M. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11), Boston, MA, USA, 30 March–1 April 2011.
62. Thaler, D.; Hopps, C. *Multipath Issues in Unicast and Multicast Next-Hop Selection*; RFC 2991(I); IETF: Fremont, CA, USA, 2000.
63. Kimura, B.Y.L.; Lima, D.C.S.F.; Loureiro, A.A.F. Packet Scheduling in Multipath TCP: Fundamentals, Lessons, and Opportunities. *IEEE Syst. J.* **2021**, *15*, 1445–1457. https://doi.org/10.1109/JSYST.2020.2965471.
64. Fairhurst, G.; Welzl, M. *The Benefits of Using Explicit Congestion Notification (ECN)*; RFC 8087(I); IETF: Fremont, CA, USA, 2017.
65. Eddy , W. (Ed.) *Transmission Control Protocol (TCP)*; RFC 9293(S); IETF: Fremont, CA, USA, 2022.
66. Iyengar, J.; Thomson, M.T. (EdS.) *QUIC: A UDP-Based Multiplexed and Secure Transport*; RFC 9000(PS); IETF: Fremont, CA, USA, 2021.
67. Al-Saadi, R.; Armitage, G.; But, J.; Branch, P. A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3609–3638. https://doi.org/10.1109/COMST.2019.2904994.
68. Atkinson, R.; Bhatti, S.N.; Rose, S. *DNS Resource Records for the Identifier-Locator Network Protocol (ILNP)*; RFC 6742(E); IRTF: Fremont, CA, USA, 2012.
69. Wellington, B. *Secure Domain Name System (DNS) Dynamic Update*; RFC 3007(PS); IETF: Fremont, CA, USA, 2000.

70. Ammar, M. ex uno pluria: The Service-Infrastructure Cycle, Ossification, and the Fragmentation of the Internet. *SIGCOMM Comput. Commun. Rev.* **2018**, *48*, 56–63. https://doi.org/10.1145/3211852.3211861.
71. Papastergiou, G.; Fairhurst, G.; Ros, D.; Brunstrom, A.; Grinnemo, K.J.; Hurtig, P.; Khademi, N.; Tüxen, M.; Welzl, M.; Damjanovic, D.; et al. De-Ossifying the Internet Transport Layer: A Survey and Future Perspectives. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 619–639. https://doi.org/10.1109/COMST.2016.2626780.
72. ten Oever, N.; Beraldo, D. Routes to rights: Internet architecture and values in times of ossification and commercialization. *XRDS* **2018**, *24*, 28–31. https://doi.org/10.1145/3220561.
73. Clark, D. The design philosophy of the DARPA Internet protocols. *SIGCOMM Comput. Commun. Rev.* **1988**, *18*, 106–114. https://doi.org/10.1145/52325.52336.
74. Yanagida, R.; Bhatti, S.N. Mobility-Multihoming Duality (dataset), data for the paper, ". Available online: https://doi.org/10.17630/663a1ff8-5e96-47c1-8793-dc4610bf2579 (accessed on 30 September 2024).