Seamless Internet connectivity for ubiquitous communication

Ryo Yanagida
University of St Andrews
St Andrews, Fife, KY16 9SX, UK
ry6@st-andrews.ac.uk

Saleem N. Bhatti
University of St Andrews
St Andrews, Fife, KY16 9SX, UK
saleem@st-andrews.ac.uk

ABSTRACT

The direct and flexible use of any network connectivity that is available within an urban scenario is essential for the successful operation of ubiquitous systems. We demonstrate seamless communication across different networks without the use of middleware, proxies, tunnels, or address translation, with minimal (near-zero) packet loss to communication flows as handoff occurs between networks. Our solution does not require any new functions in existing networks, will work on existing infrastructure, and does not require applications to be re-designed or re-engineered. Our solution requires only modifications to the end-systems involved in communication, so can be deployed incrementally only for those end-systems that require the functionality. We describe our approach and its design, based on the use of the Identifier-Locator Network Protocol (ILNP), which can be realised directly on IPv6. We demonstrate the efficacy of our solution with testbed experiments based on modifications to the Linux kernel v4.9 LTS, operating directly over IPv6, and using unmodified binary applications utilising directly the standard socket(2) POSIX.1-2008 API, and standard C library calls. As our approach is 'end-to-end', we also describe how to maintain packet-level secrecy and identity privacy for the communication flow as part of our approach.

CCS CONCEPTS

• **Human-centered computing** → **Mobile devices**; • **Networks** → **Naming and addressing**; • **Security and privacy** → *Privacy-preserving protocols.*

PURBA 2019, September 9–13, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6869-8/19/09…$15.00
https://doi.org/10.1145/3341162.3349315

KEYWORDS

Identifier Locator Network Protocol (ILNP); Mobility; Multihoming

ACM Reference Format:
Ryo Yanagida and Saleem N. Bhatti. 2019. Seamless Internet connectivity for ubiquitous communication. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2019 International Symposium on Wearable Computers (UbiComp/ISWC '19 Adjunct), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3341162.3349315

1 INTRODUCTION

Weiser's original vision for Ubiquitous Computing [40] highlights many challenges for systems and technology. In the 25+ years since that article's publication, there has been significant progress in many of these challenges: the availability of device types (tablets / 'pads'); more and better wireless communication; more functionality in displays and device interaction (pens and surfaces); and increasing parallelism for better power efficiency, especially in mobile devices. However, there is one crucial challenge on which progress has not been made: a limitation of the way naming and addressing is used in the Internet Protocol (IP). From Weiser's seminal paper [40]:

"The Internet routing protocol, IP, has been in use for over 10 years. However, neither this protocol nor its OSI equivalent, CLNP, provides sufficient infrastructure for highly mobile devices. Both interpret fields in the network names of devices in order to route packets to the device. For instance, the "13" in IP name 13.2.0.45 is interpreted to mean net 13, and network routers anywhere in the world are expected to know how to get a packet to net 13, and all devices whose name starts with 13 are expected to be on that network. This assumption fails as soon as a user of a net 13 mobile device takes her device on a visit to net 36 (Stanford). Changing the device name dynamically depending on location is no solution: higher-level protocols such as TCP assume that underlying names will not change during the life of a connection, and a name change must be accompanied by informing the entire network of the change so that exsting services can

find the device."

To enable truly ubiquitous communication, an application should be able to (i) exploit any connectivity that is available; and (ii) move existing communication flows between different networks seamlessly, without disruption to the flows. This is especially important in an urban environment, where they may be rich provision of a wide range of connectivity for use by an application.

A fundamental problem of naming and addressing

Weiser noted that the way that naming is used in IP – the way IP addresses are bound to objects in the communication stack – means that, for example, a communication flow (a stream of packets) from a device uses a topologically significant value as part of its flow state. Effectively, a flow is bound to a physical point of attachment in the network. This limits mobility for the device, and makes the flow less agile in terms of migrating the flow from one physical interface to another. For example, today, migrating a flow from a 4G connection to a WiFi connection requires each application to implement its own solution. This was true for IPv4 [36], and is still true today for IPv6 [18]. Both IPv4 and IPv6 have, essentially, the same naming and addressing architecture: addresses are bound to network interfaces, and at the same time used as part of the state identifying a communication flow, such as a UDP flow or TCP flow.

IP networking without IP addresses

In this paper, we demonstrate the use of a network naming and addressing paradigm which deprecates the use of IP addresses. Instead, we use two new namespaces for addressing: the *Locator* and the *Node Identifier*. Judicious engineering allows this radically different approach to be applied within the existing network landscape, utilising directly the IPv6 packet format, without the need for proxies or middleboxes, tunnels, address translation functions, or large-scale updates to core infrastructure. Only the two end-system that wish to communicate need to be updated to understand the new mechanism. Essentially, this enables the vision of ubiquitous communication as put forward by Weiser.

Structure of this paper

In Section 2, we discuss the background to the problem of seamless connectivity for ubiquitous systems, and establish the position we have taken for proposing our solution. In Section 3, we describe the core aspects of our solution, from both an architectural and engineering viewpoint. In Section 4, we present a mobility scenario to demonstrate how a practical implementation of our approach can allow existing applications to benefit directly from our solution without any modifications to the application code. We conclude in Section 5, including key areas for future work.

2 ENABLING SEAMLESS INTERNET CONNECTIVITY

The idea of seamless connectivity is to allow the flexible use of any and all connectivity that is available. In an urban scenario, there may be many wireless networks present. For example, there may be different IEEE 802.11 / WiFi networks operating in a local(ised) geographical area, along with various coverage of 2G, 3G, 4G, and 5G networks offering overlapping coverage, as well as national and international coverage. For a device to be able to move seamlessly between these networks, there are a number of challenges:

- *address management*: end-system network addresses change as different networks are joined and then left by a device: this is fundamental to ensuring the connectivity and correct routing is maintained, to enable the smooth and unperturbed continuation of existing communication sessions;
- *quality of service (QoS)*: the differences in connectivity will also involve different properties of the network service in terms of throughput, delay, loss, and jitter;
- *access control and resource management*: commercial networks may restrict access and resource allocation for users based on some form of revenue model or user subscription, or through some relationship with an another service provider that the user may have a subscription to;
- *accounting*: there may be a need to monitor and log access and use of resources for performance, management, or billing purposes (related to access control and resource management);

The most *basic* provision of seamless connectivity at the network layer is the first point listed above: a fundamental issue in management of the addressing system, including the allocation and use of numerical address values. This impacts directly the operation of the user (data) plane. However, managing QoS and system / network resources, as well as gathering information for accounting are issues within the scope of the control plane and management plane, and do not directly impact the *basic* function of the data plane: to allow packet transmission and reception at end end-systems. So, the user (data) plane can be treated differently.

In this paper, we focus on this first challenge. Address management can be seen, essentially, as a mobility problem: even if the end-system happens to be physically fixed geographically, e.g. a desktop computer or a server, changing connectivity from one provider or network to another also changes its topological connectivity to the network, so it has

'moved'. This has direct impact to the user (data) plane – the transmission and reception of data packets.

"IP addresses considered harmful"

We take the position that the IP address needs to be replaced by new data types that cleanly separate identity and location.

IP address ranges are allocated to specific network providers, through a hierarchical, administrative process. IP addresses have topological significance, and this may not reflect geographical location. So, when a device changes its connectivity from one provider or network to another, it will have to use an IP address from the new provider or network, meaning that existing state for communication flows must somehow be transitioned, or the flow state could become invalid.

Along with Weiser's observation, the overloaded semantics of an IP address – as both an *identifier* of the end-system, as well as a topological *locator* for the routing of traffic – has been documented over decades within the research community (e.g. [10, 13, 27]). One of the most recent expositions of the problem entitled, "IP Addresses Considered Harmful", [14] has an abstract that reads simply:

"This note describes how the Internet has got itself into deep trouble by over-reliance on IP addresses and discusses some possible ways forward."

A number of solutions have been proposed to support mobility for the Internet: for example, Mobile IPv6 (MIPv6) [32], and Proxy Mobile IPv6 (PMIPv6) [20]. Other approaches have proposed specifically a separation between the identifier and locator semantics of the IP address: for example, the Locator/ID Separation Protocol (LISP) [21], the Host Identity Protocol (HIP now at v2) [19], and Nimrod [16]. However, all such solutions require at least one of the following:

- a proxy or an agent: this offers indirection or redirection capability within the existing architecture (e.g. MIPv6, PMIPv6);
- tunnelling: an encapsulation mechanism that provides an overlay network, requiring additional network entities (such as proxies or agents) (e.g. LISP);
- address mapping: some sort of translation service, requiring additional mapping state, and also the modification of existing network entities, or the introduction of new network entities (e.g. LISP, Nimrod);
- application modifications: the normal API that is used by application developers (often socket(2) or based on socket(2)) may need to be extended, meaning that existing applications would need to be re-engineered or even re-designed (e.g. Nimrod, HIP).

That is, some sort of upgrade and/or support is needed in the network infrastructure, or some re-engineering of applications, each of which can be impractical or costly in terms of deployment and maintenance. Also, all these solutions still have the fundamental problem of using IP addresses as part of the functional architecture, which Weiser noted was the key issue.

A key enabler for ubiquitous systems

We take the position that ubiquitous applications should be offered seamless connectivity directly at the internetworking layer, to allow basic, packet-level communication across multiple network types. Application designers are then free to compose their own additional service components, to provide QoS, resource management, accounting capability, and other functionality as required.

Using middleware for providing support for ubiquitous systems potentially offers a way of addressing all four issues listed above (e.g. surveys for QoS-aware middleware [28], context-aware middleware [25], middleware for Internet of Things [31]). However, middleware must be deployed, and any support services must be available for operation of the application. Additionally, applications may need to be re-engineered, or even re-designed, to make use of the facilities and functionality of a specific middleware platform. Ultimately, the middleware itself makes strong assumptions about the underlying network connectivity: that the connectivity is present, and is easily accessible.

Our position is that whilst middleware does introduce excellent functionality, its inclusion into the application ecosystem for *some* applications, especially lightweight applications, may be an overhead, as often the application has specific needs that might not be met by general middleware platforms. Providing a common, seamless connectivity mechanism is the core enabler for ubiquitous applications, and using existing APIs reduces the additional overhead for existing and new applications. Indeed, having seamless connectivity would itself help to produce more robust middleware for ubiquitous systems, especially if a middleware platform is a strong requirement for a particular application.

Previous work on ILNP mobility

Mobility was seen as a challenging use case early on in the design and development of ILNP. So, many of the previous analyses and experiments for ILNP have been based on mobility scenarios: both for individual mobile nodes, e.g. [12, 33, 34], and for whole mobile networks, e.g. [8, 11, 37].

Our past work was a very basic proof of concept, based on Linux kernel v3.9, which showed that ILNP can support mobile nodes moving between two networks, with both UDP [33] and TCP [12, 35]. The work reported herein extends that previous mechanism: the *continuous movement scenario*

Table 1: Use of names in IP compared to ILNP.

Protocol Layer	IP (IPv4 and IPv6)	ILNP (ILNPv6)
Application	FQDN / IP address / Application	FQDN / Application
Transport	IP address	NID (dynamic binding to L64)
Network	IP address	L64 (dynamic binding to NID & interface)
(interface)	IP address	(dynamic binding to L64)

Application: Application-specific naming
FQDN: Fully Qualified Domain Name
L64: Locator (64 bits)
NID: Node Identifier (64 bits)

evaluated in Section 4 was not possible previously. Also, the rate of movement was much lower, and now the bottleneck is due only to the system-level effects in our testbed related to switching physical interfaces on and off for emulating movement. The latency for ILNP network-level handoff is a single round-trip-time (RTT) needed to complete the LU/LU-ack handshake, which will typically be the order of milliseconds (see Section 3).

3 OVERVIEW OF ILNP

The Identifier-Locator Network Protocol (ILNP) is defined as an architecture from the Internet Research Task Force (IRTF) [1–9]. The core concepts are presented in RFC6740 [4], with some general engineering considerations in RFC6741 [5]. It could be implemented as a 'clean-slate' protocol, but the main focus of the work so far has been to create a superset of IPv6 called 'ILNPv6', with additions to IPv6 as described in RFC6743 [3] and RFC6744 [7]. It is possible to create 'ILNPv4' as a set of extensions to IPv4 [1, 2, 6], though the engineering would be impractical, and it would be difficult to deploy. *We discuss in this paper only ILNPv6, and so henceforth just refer to it as 'ILNP'.* ILNP was subject to public discussion, review, and analysis as part of the IRTF Routing Research Group (RRG, now concluded), which assessed a number of proposals. A summary of that discussion and analyses can be found in RFC6115 [38], which states:

"We recommended ILNP because we find it to be a clean solution for the architecture. It separates location from identity in a clear, straightforward way that is consistent with the remainder of the Internet architecture and makes both first-class citizens. Unlike the many map-and-encap proposals, there are no complications due to tunnelling, indirection, or semantics that shift over the lifetime of a packet's delivery."

ILNP architecture

ILNP uses *Node Identifier (NID)* values and network *Locator (L64)* values as two distinct name types in the communication stack. The comparison with IP is shown in Table 1. The NID is a 64-bit value, is used only end-to-end (for example in transport protocol state), names a *node* (not an interface as in IP), and has no topological significance. The L64 is also a 64-bit value, names a *network*, and has topological significance – it is an IPv6 network prefix and can be used for routing. A binding between a NID and a L64 is an *Identifier-Locator Vector (I-LV)*, and that is what is used for addressing in ILNP. You can think of the NID 'residing' at the L64 for a packet which contains that I-LV.

NID and L64 values are discovered by use of fully-qualified domain names (FQDNs). So, just as for IP, a FQDN lookup can be used for ILNP. In the case of ILNP, however, it yields NID and L64 values, either those held in the local /etc/hosts file, or in the global Domain Name System (DNS). DNS resource records for ILNP are defined in RFC6742 [9], and are supported by commercial DNS server software.

NID values remain constant for the duration of a transport level session. A node may use one or more NID values simultaneously. The values may be administratively assigned and be semi-permanent; or they can be generated dynamically as ephemeral values in the same way (and leveraging the same code) that addresses can be generated dynamically for IPv6, e.g. for privacy as in RFC4941 [29]. In short, any mechanism that IPv6 uses to generate Interface ID (IID) values for IPv6 addresses can be used to generate NID values for ILNP.

L64 values are IPv6 routing prefixes. They can be learned from IPv6 Router Advertisements (RAs). L64 values do not form part of the end-to-end protocol state for transport protocols. L64 values can change during the lifetime of a transport session. Local L64 values could change as current prefixes expire and new prefixes are learned, for example due to mobility, dynamic re-homing, or failover of IP-level connectivity. Remote L64 values could change also, as a remote node's own L64 values change, and the remote node notifies such changes to a correspondent node using *Locator Update (LU)* messages [3].

Mobility in ILNP

As discussed above, providing a simple mobility model at the internetworking layer is key to enabling seamless connectivity across multiple network types. ILNP uses a NID value as the end-to-end invariant for UDP and TCP sessions,

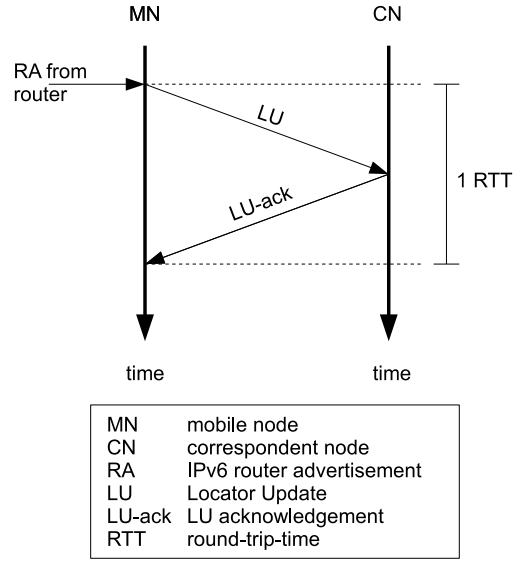

Figure 1: In ILNP mobility, the MN discovers a new L64 value from an IPv6 RA as it connects to a new network. The new L64 value is signalled to the CN in a LU message. When the CN's LU-ack is received at the MN, the new connectivity is complete, e.g. the MN has 'moved'. The MN can keep using any other L64 values (while they are available), as NID values can be bound to multiple L64 values (and so multiple interfaces) simultaneously.

instead of whole IP addresses, with dynamic bindings to L64 values. New L64 values are discovered as IPv6 routing prefixes. These are required to be advertised by any IPv6 router, either periodically or upon a solicitation message being issued by an end-system. Once the L64 value is known, the mobile node (MN) sends a Locator Update (LU) message to its correspondent node (CN), which confirms the new L64 value with a LU-ack message – please see Figure 1. At this point, both the MN and CN have synchronised on new NID/L64 bindings for the MN, and so the MN's new connectivity is known. The LU message can be protected by the ILNP nonce (see later), for defending against off-path attacks, but cryptographic techniques can be used if stronger protection is required.

ILNP transport protocol state is different to that of IP. The tuple expression (1) depicts flow-state for IP, and tuple expressions (2) for ILNP. The suffixes X and Y are, respectively, for the MN and CN in the communication session. A is an IP address, P is a port number, N is a NID value, and L is a L64 value. At the interface (if), an IP address is bound semi-permanently to the interface, but in ILNP, the binding is dynamic, and can expire (when IPv6 RAs for a given L64 value are no longer visible).

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

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

In expression (2), the MN uses source I-LV $\langle N_X, L_X \rangle$ and would transmit to a CN using destination I-LV $\langle N_Y, L_Y \rangle$. Note that a NID value is invariant for the duration of the communication session to maintain end-to-end integrity for the transport layer flows. The values for L_X and L_Y, and their respective dynamic bindings to N values and interfaces, can be changed as connectivity changes occur. When multiple L64 values are available to an end-system, a single NID can be bound simultaneously to any of those L64 values, and so the end-system can transmit and receive a single flow over multiple IP networks. For example, at the MN currently using L_X, if another locator, L_A becomes available, from expression (2), we have expressions (3) and (4)):

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

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

In expression (3), the MN is now using locator L_X and locator L_A. This means packets being sent to I-LV $\langle N_Y, L_Y \rangle$ could now be sent using as a source either $\langle N_X, L_X \rangle$, or $\langle N_X, L_A \rangle$. Effectively, the MN can be multihomed and have multipath connectivity: when used in transitioning communication across different networks, we refer to this a *network layer soft handoff*. So, during movement from one network to another, there should be no gratuitous packet loss due to the handoff process itself. This mechanism – a *network layer (layer 3)* soft handoff – is unique to ILNP. It enables seamless connectivity in terms of addressing at the end-systems, as well as for routing and forwarding of packets in the network: no special handling for mobility or multipath connectivity is required in the network, only standard unicast routing is used.

The MN could continue to use L_A for as long as L_A is available. For our evaluation (see Section 4), L_A is signalled to the CN using a LU (see Figure 1), and once the MN receives the LU-ack from the CN (or the first packet from the CN using L_A instead of L_X), the MN drops the use of L_X, the soft handoff is complete, and its local state is now as shown in expression (4).

ILNP packet-level view

At the IPv6 level, ILNP packets carry an IL-V in place of an IPv6 address, as shown in Figure 2. All ILNP packets also carry an additional Nonce Destination Option (just 'nonce'

in the text, henceforth), as defined in RFC6744 [7]. This carries an ephemeral value of 4 bytes or 12 bytes, which provides lightweight protection against off-path packet spoofing, and also flags the packet as an ILNP packet. The L64 value occupies the same bits as the IPv6 unicast routing prefix. So, for the purposes of routing and forwarding in network elements such as switches and routers, an ILNP packet looks like an IPv6 packet, on the wire, as shown in Figure 3.

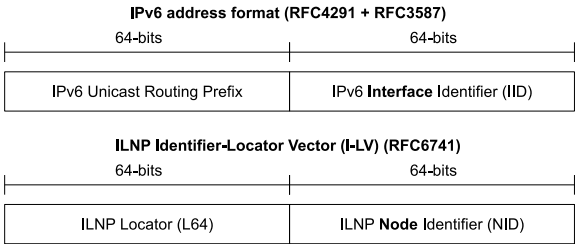

Figure 2: An IPv6 unicast address has a 128-bit format: a routing prefix (upper 64 bits), and an interface identifier (lower 64 bits). The ILNPv6 Identifier-Locator Vector (I-LV) format has the same structure as an IPv6 address. An IPv6 routing prefix is used as a L64 value, with the same syntax and semantics. The NID has the same syntax as an IPv6 Interface Identifier, but has different semantics.

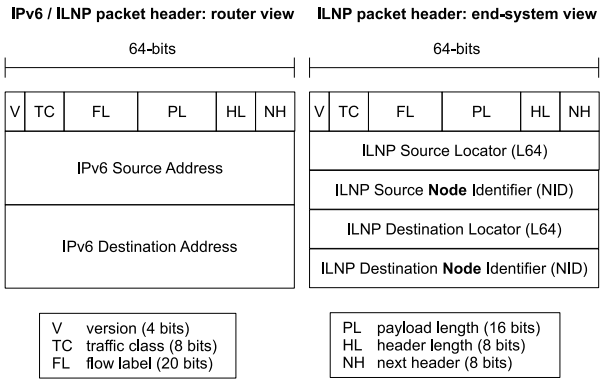

Figure 3: Following from Figure 2, as routers only inspect the upper 64 bits of an IPv6 address field for routing purposes, an ILNP packet looks like an IPv6 packet to a router. However, at an ILNP end-system, the 128-bit IPv6 address is resolved into a 64-bit L64 value, and a 64-bit NID value.

Enabling (legacy) non-ILNP applications to use ILNP

The modifications that are needed at the end-systems are summarised below. These allow 'well-behaved' legacy (non-ILNP) applications to use ILNP without modification. By 'well-behaved' we mean that the application does not use absolute values of IPv6 addresses (i.e. either whole or part of an IPv6 address) as part of its operation in the application layer state, or in its configuration. Indeed, this has always been the recommended behaviour for IP applications [15].

(1) Modifications to getaddrinfo(3) (libc) to allow NID and L64 values to be used from /etc/hosts[1].
(2) Modifications to the socket(2) POSIX.1-2008 implementation to allow existing IPv6 applications to work over ILNPv6 without modification[2].
(3) Modification of IPv6 and ICMPv6 packet processing paths within the Linux kernel. IPv6 code is modified to distinguish between L64 and NID values, in IPv6 packets. ICMPv6 code is modified to implement LU messages.
(4) Modifications to UDP and TCP processing of end-to-end communication within the Linux kernel. This is so that UDP and TCP end-system state binds only to the NID and not the L64, allowing the L64 to be dynamically rebound for new connectivity[3].

Supporting non-ILNP applications

To support non-ILNP applications, we maintain a consistent interface across the socket(2) API. The key to this is to allow name resolution – mapping from fully qualified domain names (FQDNs) to IP addresses – to return the same data structures and have the same behaviour an application would expect for IP. For local resolution, the /etc/hosts file syntax has been extended to allow I-LVs to be associated with names. Example entries for I-LVs in /etc/hosts are shown in Figure 4. This requires modifications to libc to modify the way getaddrinfo(3) reads /etc/hosts and returns I-LV values in struct addrinfo: essentially, the modification is to read the I-LV entries in /etc/hosts, and insert them into struct addrinfo to resemble IPv6 addresses.

```
# The format of an IL-V in /etc/hosts is:
#   LLLL-LLLL-LLLL-LLLL.NNNN+NNNN+NNNN+NNNN   hostname
# where:
#   LLLL are hex digits for 16-bits of a L64 value
#   NNNN are hex digits for 16-bits of a NID value

2001-2-aa-aa.0+0+0+a1      ilnp-host-a1
2001-2-aa-aa.0+0+0+a2      ilnp-host-a2
2001-2-bb-bb.0+0+0+b1      ilnp-host-b1
2001-2-bb-bb.0+0+0+b2      ilnp-host-b1
```

Figure 4: Example I-LV entries in /etc/hosts file.

[1]Modifications to getaddrinfo(3) to allow use of DNS records, as in RFC6742, also exist within our lab testbed, and will be integrated in the near future.
[2]Enhancing the socket(2) API itself, to exploit the presence of ILNPv6, and to allow 'ILNP-aware' applications, is an item for future work.
[3]As modifications are made to the checksum computation for UDP and TCP, for now, any offload processing of transport level checksums needs to be disabled for correct behaviour. This is, of course, only until the code in the appropriate devices has been updated.

For global name resolution, the ILNP DNS extensions defined in RFC6742 [9] are supported commercially in BIND[4], KnotDNS[5], and NSD/Unbound[6]. However, for ease of configuration of our evaluation testbed, DNS was not used during the experiments reported in Section 4.

Packet-level secrecy and integrity for ILNP

ILNP can support directly packet-level security and privacy. As for IPv6, *ephemeral* NID values can be generated as required using normal IPv6 mechanisms [17], to greatly improve identity privacy and perturb tracking and correlation of network activities over time.

As flow continuity is maintained, it is possible to use end-to-end mechanisms for protecting packets, such as encryption of packet contents, as well as message authentication code (MAC) mechanisms for protecting against modification and forgery. No third-party entities, such as proxies, need to be trusted, but, of course, can be incorporated at the application level, if required by the application.

Mechanisms in IPsec [22], providing packet-level security features in a standard way can be used, by binding security associations to the 64-bit ILNP NID, which is invariant across connectivity changes, rather than the IPv6 address. This can be used to protect both data packets as well as LU messages. A more detailed discussion of integration between ILNP and IPsec is an item for future work.

Packet-level privacy for ILNP

The 64-bit NID values for ILNP can be generated as required for individual communication sessions. ILNP exploits many features in IPv6 that are already available to allow this. When *ephemeral* NID values are created, they enable identity privacy, as those NID values can be used for a single communication session only, and new NID values generated for new communication sessions. This improves the privacy of a user when traffic monitoring is in progress, especially preventing correlation of activities over time. IPv6 has mechanisms to generate NID values, e.g. see [17] for a discussion, and to check that those NID values are not being used by other end-systems on the same network [30].

More sophisticated privacy mechanisms, such as location privacy, are also possible if other ILNP users or network administrators cooperate to provide *Locator Re-writing Relay (LRR)* nodes for forwarding ILNP packets. This would change (re-write) the value of a source L64 values in packets before transmission. Even the use of a single LRR at the border of a site network (for example, the border router of an enterprise network or a service provider) can provide improved location privacy. This possibility is described in more detail in [8, 11], and would require cooperation of network administrators, or other ILNP users (overall, this could provide an ILNP parallel to a Tor[7] network).

Also, as ILNP does not require the use of proxies, middleboxes, or agents, and only uses unicast routing, there is very little state information held on a node that is not one of the communicating end-systems. This increases an attackers effort in tracking or correlation of information about the end-systems and their users.

4 EVALUATION OF A MOBILITY SCENARIO

Our evaluation was based on the emulation of a 'continuous' mobility scenario. A MN 'moves' across three different networks, round-robin. The intention was to show that the ILNP connectivity is stable for standard UDP and TCP flows, using an existing IPv6 binary (not ILNP-aware) of a common performance tool, *iperf*, to generate packet flows. To demonstrate the benefits of using ILNP, we compared its performance with that of Mobile IPv6 (MIPv6) on the same testbed. We start this section with a brief description of MIPv6, which is the Internet standard for mobile systems.

Mobile IPv6

Mobile IPv6 (MIPv6) was originally defined in 2004 [24], but has been updated in 2011 [32]. A key difference between MIPv6 and ILNP is that MIPv6 maintains a Home Agent (HA) at its Home Network (HN), which knows of a permanent Home Address (HoA) for the MN. This is always the first point of contact for any CN and is, effectively, an identity for the CN. This means that a mixture of indirection and redirection, with tunnelling, proxies, and address changes need to be signalled between the MN, HA, and CN when the MN is away from its HN.

In Figure 5, we see the handoff mechanism to manage node movement for MIPv6. The MN enters a new network and detects an IPv6 RA. So, it generates a new Care of Address (CoA) – a valid address for that network – and uses a Binding Update (BU) to signal its Home Agent (HA) at its designated (configured) Home Network (HN). Effectively, the HoA is the locator for the MN. Any new communcation requests for the MN are always addressed to the HoA, and so come first via the HA, which acts as a proxy for the MN when it is away from its HN. Initially, data packets are tunnelled from the HA to the MN using the MN's CoA. The MN then uses a Home Test Init (HoTI) message and Care of Test Init (CoTI) message, to perform a *return routability* test (checking connectivity), after which it can then send a BU to the CN directly to update the CN with its CoA.

[4]https://www.isc.org/downloads/bind/
[5]https://www.knot-dns.cz
[6]https://www.nlnetlabs.nl/projects/unbound/
[7]https://www.torproject.org

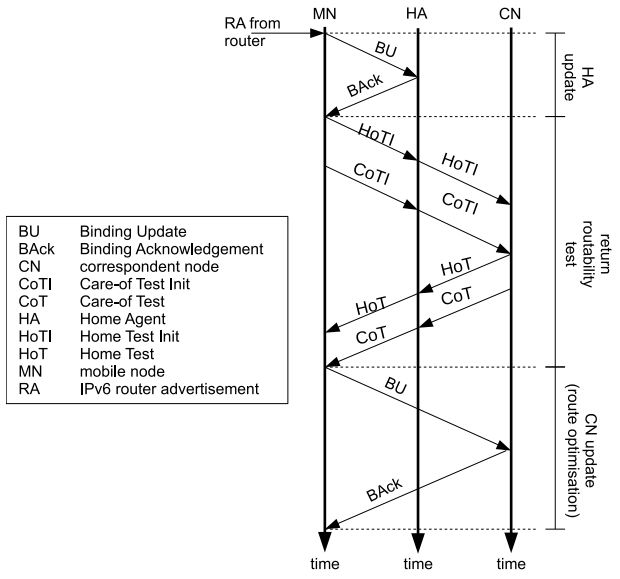

Figure 5: In our testbed, we trigger a MIPv6 handoff when a new IPv6 RA is detected, as for ILNP. The MN obtains a new Care of Address (CoA) and updates its Home Agent (HA) with its new CoA, using a Binding Update (BU). It then tests return routability (the 'Test' messages), and then update the CN with its new CoA using a BU, so that the CN is aware of the new, topologically-correct IP address for the MN.

This prolonged handshake has the potential to introduce unwanted latency, as data transfer for the new connectivity cannot begin until the handshake has completed. So, the Internet community have implemented various 'optimisations' to parallelise the three phases of the handshake [39]: those optimisations were enabled for MIPv6 during our evaluation.

Testbed configuration and evaluation

Our testbed configuration was as shown in Figure 6. The CN and MN only were ILNP nodes, running Debian 9.7, but with the v4.9 LTS Linux kernel modified to implement ILNP. We did not implement any performance optimisations for ILNP, as our focus has been on producing correct functionality. All the router nodes were commercially available, EdgeRouterX[8] units from Ubiquiti Networks, and were not modified in any way. Name resolution was performed at CN and MN via an extended /etc/hosts file, to maintain a simple experimental configuration, and to avoid latency due to DNS lookups. All connectivity was Gigabit Ethernet, with direct connections between nodes, and mobility was emulated by switching interfaces on and off. Handoff was triggered by the reception of IPv6 RAs for new prefixes. Traffic flows were generated via *iperf* v2.0.9, unmodified, i.e. the *iperf* binary was the standard IPv6-capable iperf binary installed from the Debian 9.7 repositories. The standard *tcpdump* installation was used to log packet transfers at the CN (i.e. a pcap – packet capture – file was recorded), and then post-processed for both UDP and TCP to produce the data reported herein.

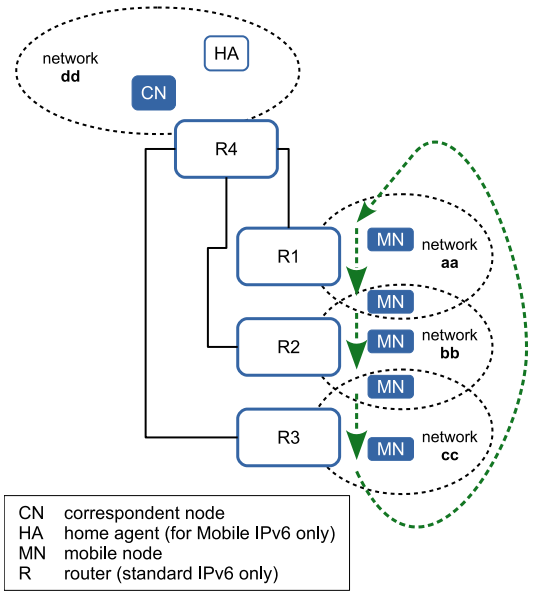

Figure 6: The ILNP testbed for mobility experiments. Only the CN and MN nodes were running the end-system changes as described in this paper, but within a standard Debian 9.7 Linux installation. The MN and CN used ASRock C3558D4I-4L Intel Atom C3558 (4-core) SoC mini-ITX motherboards, with 8GB of DDR4 DRAM, each with 4x Gigabit Ethernet. The green/dashed arrows show the emulated movement of the MN. The HA node was used only for MIPv6 measurements. All router nodes were commercial Ubiquiti Networks EdgeRouterX units.

A 120s flow was generated between CN and MN using *iperf* v2.0.9. As the flow was active, the MN was 'moved' across three different networks by enabling and disabling network interfaces on the MN, forcing changes in connectivity at the physical level, and waiting 10s during the soft handoff between two networks. The *iperf* throughput was deliberately limited to 10 Mbps using normal command line arguments to allow full packet capture with *tcpdump* for detailed analyses. This 120s flow measurement was repeated 20 times for MIPv6 and then again for ILNP, for each of TCP and UDP.

A note on MIPv6 performance

MIPv6 performance was generally poor on our testbed configuration. At the 10s movement pattern we have described

[8] https://www.ui.com/edgemax/edgerouter-x/

above, performance of the 120s flow with *iperf* stalled, until we reduced the transmission buffer to 1260 bytes (using command line option '-l 1260' for *iperf*). This configuration / tuning was found by trial and error to enable a MIPv6 flow to complete. No special configuration or tuning was required for ILNP, and flow measurements never stalled.

Results and observations

In Figure 7 is shown a typical observation at the CN of the throughput that is attained on each network ('aa', 'bb', and 'cc', as labelled in Figure 6), as well as the the aggregate throughput (network 'dd'). The facet-set of graphs shows firstly the typical throughout observed on each individual network ('aa', 'bb', 'cc'), and then the aggregate throughout observed at the CN. Results are shown for both MIPv6 (left column of Figure 7) and ILNP (right column of Figure 7). The MIPv6 flow is much 'burstier', with discontinuities (throughput at zero), due to the latency and buffering effects during handoff. The throughput peaks (which exceed the limit of the y-axis we have chosen for comparing results), is ~150 Mbps. We had chosen to control directly the *iperf* flow to 10 Mbps, but the 1 Gbps link allowed those bursts to be accommodated. However, on a resource-constrained wireless link, as might exist in an urban scenario, such bursts may lead to congestion and loss. Some of the burstiness is due to the behaviour of *iperf*, which tries to moderate the bursts to regulate output to 10 Mbps as we have configured. The TCP flow over ILNP suffers no such disruption, and the progression of the flow is consistent and continuous.

Summary statistics for TCP are given in Table 2. ILNP had more consistent throughput than MIPv6. ILNP also generated fewer retransmissions. It is likely, therefore, that ILNP would make better use of available connectivity and be more effective when connectivity changes are encountered. This is important for urban applications: less wasted network capacity, fewer retransmissions, and lower loss lead to better overall performance, especially where resources are constrained, e.g. wireless network capacity, CPU cycles, and battery life. Overall, our results show ~zero gratuitous loss, and very little misordering for ILNP during handoff.

Table 2: TCP summary statistics for *iperf* measurements.

	MIPv6	ILNP
Throughput (Mbps)*	1.6 / 9.1 / 10.1	10.5 / 10.5 / 10.5
Retransmissions	0 / 18.5 / 32	0 / 0 / 12

All values minimum / median / maximum, 1 d.p., where shown. All values as observed at the CN. Each flow was 10 Mbps configured rate in *iperf*, 120s duration, 20 repetitions.
*Reported values above 10 Mbps are due to rate control by *iperf*.

Discussion: latency, misordering, loss

We have used Ethernet links in our testbed to allow convenient configuration of our scenario, and for ease of reproducibility of results. We have previous experiments with an older, less robust kernel (v3.9) on IEEE 802.11ac (5GHz Wifi) [35], but only moving from one network to one other network. Use of multiple wireless connectivity is of particular importance for ubiquitous systems and applications. The handoff latency due to the various different wireless technologies will be very different, and it was not our aim to evaluate that diverse behaviour in this study. Overall, the latency in handoff due to ILNP is 1 RTT, as described in Section 3 and Figure 1, and the disruption to traffic is low, as discussed above.

The misordering occurs due to multipath effects during soft handoff. In our lab-based testbed, the buffer effect of the network path between MN and CN, b_p, is given by $d_e.RTT$, where d_e is the end-to-end datarate of the path. For our testbed, with d_e set to 10 Mbps with *iperf*, and a RTT of ~2 ms, we had a value for b_p of ~2500 bytes, or less than 2 packets (the packet size on Ethernet is ~1440 bytes, as constrained by the maximum transmission unit (MTU) size of 1500 bytes). This means that there were relatively few packets in flight, and so multipath effects were small. As the RTT along the path between the MN and CN increases, and also as the RTT *difference* between the network paths increases, the multipath effect would be more pronounced. For example: with the same 10 Mbps flow, but with a RTT of 20 ms, we would have ~16 packets in flight; and with a RTT of 200 ms, we would have ~167 packets in flight. We would still expect little gratuitous loss during soft handoff with ILNP, but there would be a greater possibility of misordering of packets.

When using UDP, the application would need to detect and deal with misordering. When using TCP, the misordering could generate spurious TCP retransmissions due to the normal behaviour of the TCP congestion control algorithm, interpreting misordering as loss. The impact of this for TCP could be reduced (but not completely alleviated) by using TCP's selective acknowledgement (SACK) mechanism [26][9].

Other protocols, such as MIPv6 and LISP, would also suffer the same misordering, as it is a function of the network paths, as described above. However, MIPv6 and LISP would also suffer packet loss during handoff [23, 35], which could be more disruptive for an application, whilst gratuitous loss is ~zero for ILNP in our experiments.

[9]TCP SACK has its own security issues, which would need to be considered before use. At the time of writing, it is recommended that TCP SACK should be disabled on Linux – see CVE-2019-11477 https://nvd.nist.gov/vuln/detail/CVE-2019-11477

5 CONCLUSION

We have shown how a radical change to addressing in IP networks can be implemented, and used to provide seamless connectivity that would be beneficial to ubiquitous communication. This would be particularly useful in urban environments, where a rich set of Internet connectivity could then be exploited by applications and devices.

Our solution uses the Identifer-Locator Network Protocol (ILNP), implemented as a superset of IPv6. ILNP changes the addressing model to use Node Identifier (NID) and Locator (L64) values to enable explicit naming of end-systems and networks, respectively. ILNP also allows dynamic bindings between NID and L64 values, and between L64 values and physical interfaces. ILNP can exploit the new naming architecture to eliminate gratuitous loss during a change in connectivity.

We have evaluated our mechanism using a testbed experiment emulating 'continuous mobility'. That is, a mobile node (MN) had its connectivity changed across three networks multiple times, moving across the networks in 'round-robin' fashion, whilst TCP flows were in progress to a correspondent node (CN). The testbed used our own modifications to the Linux v4.9 LTS kernel for the CN and MN, but used commercial routers and standard unicast routing for the connectivity between CN and MN. The packet flows where generated using a standard, IPv6 (non-ILNP aware) binary for *iperf* v2.0.9, and measurements were made for ILNP and Mobile IPv6 (MIPv6).

We analysed the packet traces received at the CN, and found that ILNP outperformed MIPv6. ILNP had ~zero gratuitous loss during handoff, and very little misordering, as well as very consistent packet transfer and throughput. MIPv6 packet flows were disrupted due to its handoff mechanism, with inconsistent and discontinuous packet flows.

As the NID and L64 values can be dynamically assigned and bound, there is great potential for ILNP to offer improved identity privacy and location privacy of users. This would make it harder for an observer or attacker to analyse traffic patterns, especially correlating traffic over time for an individual user.

Future work

On the horizon for development of this work is the inclusion of two specific capabilities of particular utility for urban ubiquitous applications:

- *security and privacy*: the mechanisms for packet-level security, as well as packet-level identity privacy and location privacy will be highly beneficial for individual users.
- *mobility-multihoming duality*: full integration and control of both the multihoming and mobility aspects of ILNP together will allow extremely flexible use of any and all Internet connectivity that is available, including simultaneously supporting multiple different service providers. This will also allow resilience of the connectivity in cases of episodic connectivity, failure in connectivity by an individual service provider, or if a traffic-based denial of service attack impacts a particular network.

Overall, we believe that ILNP can offer the network connectivity described in Weiser's vision for ubiquitous computing.

REFERENCES

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

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

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

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

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

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

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

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

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

[10] C.J. Bennett, S.W. Edge, and A.J. Hinchley. 1977. *Issues in the Interconnection of Datagram Networks.* Internet Experiment Note (IEN) 1. ARPA Network Working Group.

[11] S. N. Bhatti, R. Atkinson, and J. Klemets. 2011. Integrating Challenged Networks. In *MILCOM 2011 - 30th IEEE Military Communications Conf.* 1926–1933. https://doi.org/10.1109/MILCOM.2011.6127596

[12] S. N. Bhatti, D. Phoomikiattisak, and B. Simpson. 2016. IP without IP addresses. In *AINTEC 2016 - 12th Asian Internet Engineering Conf.* 41–48. https://doi.org/10.1145/3012695.3012701

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

[14] Brian E. Carpenter. 2014. IP Addresses Considered Harmful. *SIGCOMM Comput. Commun. Rev.* 44, 2 (April 2014), 65–69. https://doi.org/10.1145/2602204.2602215

[15] B. Carpenter (Ed). 1996. *Architectural Principles of the Internet.* RFC 1958 (I). IAB.

[16] I. Castineyra, N. Chiappa, and M. Streenstrup. 1996. *The Nimrod Routing Architecture.* RFC 1992 (I). IETF.

[17] A. Cooper, F. Gont, and D. Thaler. 2016. *Security and Privacy Considerations for IPv6 Address Generation Mechanisms.* RFC 7721 (I). IETF.

[18] S. Deering and R. Hinden. 2017. *Internet Protocol, Version 6 (IPv6) Specification.* RFC 8200 (STD 86). IETF.

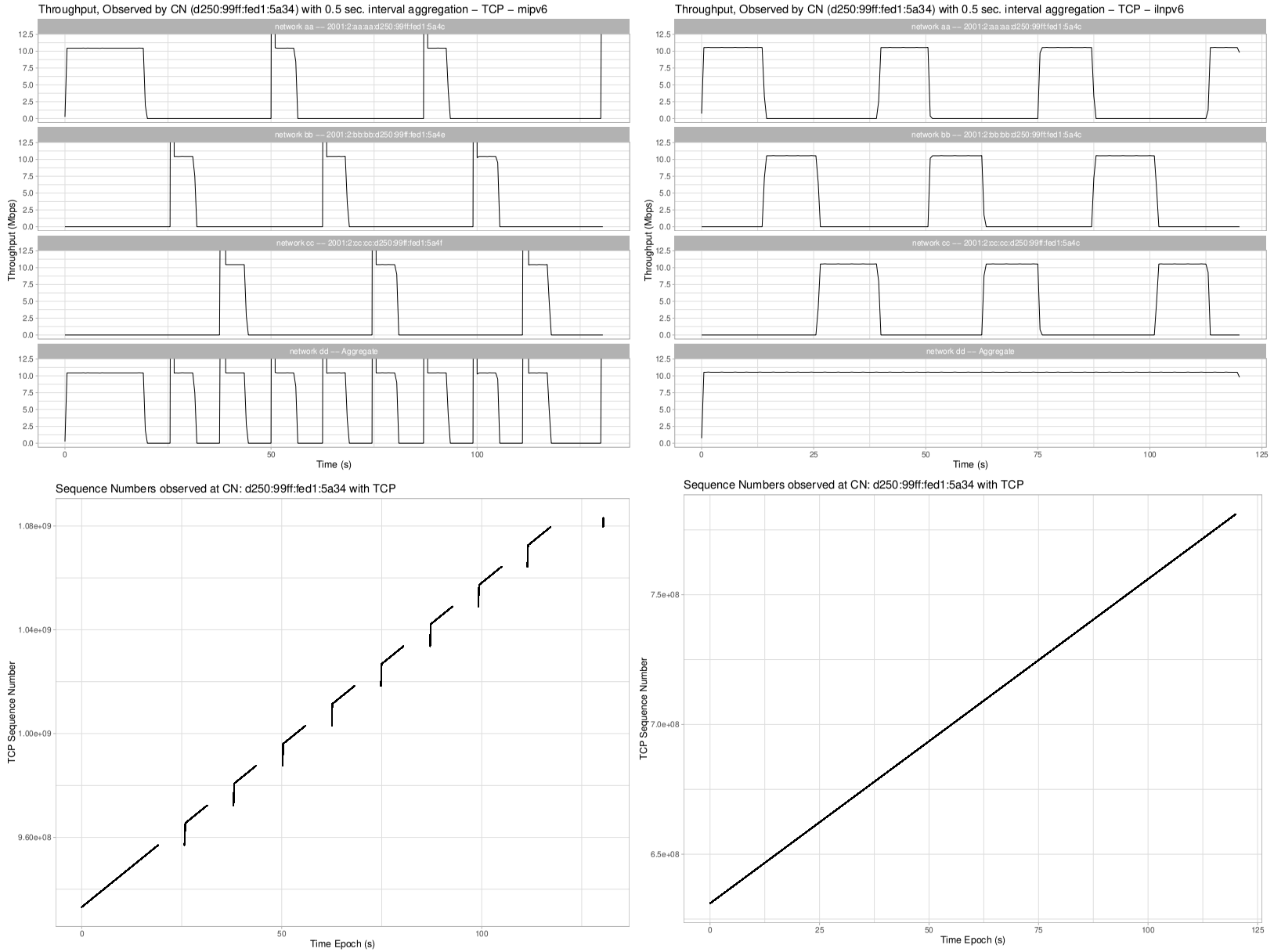

Figure 7: A view of a typical flow observed on our testbed at the CN. The top row shows typical throughput on each network. Ideally, there should be a smooth transition from one network to another, and the aggregate throughput should be consistent and continuous. The bottom row shows the corresponding sequence numbers for the TCP sediments that are transmitted. Ideally, the progression of sequence numbers should be a straight line, and so a smooth transfer of data. The MIP flow (left column) is heavily disrupted, due to handoff and buffering effects. The peaks that extend beyond the y-axis were ~150 Mbps. The ILNP flow (right column) has smooth transitioning of the flow as the MN moves across the networks, and does not suffer any significant perturbation to the flow due to handoffs.

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

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

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

[22] S. Frankel and S. Krishan. 2011. *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap.* RFC 6071 (I). IETF.

[23] Musab Isah, Steven Simpson, and Chris Edwards. 2018. An improved LISP mobile node architecture. *Journal of Network and Computer Applications* 118 (Sep 2018), 29–43. https://doi.org/10.1016/j.jnca.2018.03.018

[24] D. Johnson, C. Perkins, and J. Arkko. 2004. *Mobility Support in IPv6.* RFC 3775 (PS Obsoleted by RFC6275). IETF.

[25] Xin Li, Martina Eckert, José-Fernán Martinez, and Gregorio Rubio. 2015. Context Aware Middleware Architectures: Survey and Challenges. *Sensors* 15, 8 (2015), 20570–20607. https://doi.org/10.3390/s150820570

[26] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. 1996. *TCP Selective Acknowledgment Options.* RFC 2018 (I). IETF.

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

[28] K. Nahrstedt, Dongyan Xu, D. Wichadakul, and Baochun Li. 2001. QoS-aware middleware for ubiquitous and heterogeneous environments. *IEEE Communications Magazine* 39, 11 (Nov 2001), 140–148. https://doi.org/10.1109/35.965372

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

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

[31] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng. 2017. IoT Middleware: A Survey on Issues and Enabling Technologies. *IEEE Internet of Things Journal* 4, 1 (Feb 2017), 1–20. https://doi.org/10.1109/JIOT.2016.2615180

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

[33] D. Phoomikiattisak and S. N. Bhatti. 2015. Mobility as a First Class Function. In *WiMob 2015 - IEEE Intl. Conf. Wireless and Mobile Computing, Networking and Comms.* 858–867.

[34] D. Phoomikiattisak and S. N. Bhatti. 2016. Control Plane Handoff Analysis for IP Mobility. In *WMNC2016 - 9th IFIP Wireless and Mobile Networking Conference.* 65–72.

[35] D. Phoomikiattisak and S. N. Bhatti. 2019. End-To-End Mobility for the Internet Using ILNP. *Wireless Communications and Mobile Computing* 2019, Article ID 7464179 (Apr 2019), 29. https://doi.org/10.1155/2019/7464179

[36] J. Postel (Ed). 1981. *Name, addresses, ports, and routes.* RFC 791 (S). DARPA Internet Program.

[37] D. Rehunathan, R. Atkinson, and S. Bhatti. 2009. Enabling Mobile Networks Through Secure Naming. In *Proc. IEEE MILCOM 2009.* 8.

[38] T. Li (Ed). 2011. *Recommendation for a Routing Architecture.* RFC 6115 (I). IRTF.

[39] C. Vogt and J. Arkko. 2007. *A Taxonomy and Analysis of Enhancements to Mobile IPv6 Route Optimization.* RFC 4651 (I). IETF.

[40] Mark Weiser. 1993. Some Computer Science Issues in Ubiquitous Computing. *Commun. ACM* 36, 7 (Jul 1993), 75–84. https://doi.org/10.1145/159544.159617