WiMob 2015 - 11th IEEE Intl. Conf. Wireless and Mobile Computing, Networking and Communications. Abu Dhabi, UAE. Oct 2015

Mobility as a First Class Function

Ditchaphong Phoomikiattisak & Saleem N. Bhatti
School of Computer Science
University of St Andrews, UK
Email: {dp32,saleem}@st-andrews.ac.uk

***Abstract*—Seamless host mobility has been a desirable feature for a long time, but was not part of the original design of the Internet architecture or protocols. Current approaches to network-layer mobility typically require additional network-layer entities for mobility management, which add complexity to the current engineering landscape of the Internet. We present a host-based, end-to-end architecture for host mobility using the *Identifier-Locator Network Protocol (ILNP)*. ILNP provides mobility support as a *first class function*, since mobility management is controlled and managed by the end-systems, and does not require additional network-layer entities. We demonstrate an instance of ILNP that is a superset of IPv6 – called ILNPv6 – that is implemented by extending the current IPv6 code in the Linux kernel. We make a direct comparison of performance of ILNPv6 and Mobile IPv6, showing the improved performance of ILNPv6.**

I. Introduction

As the use of mobile devices and methods of wireless connectivity continue to increase, seamless mobility becomes more desirable and important. By *seamless mobility*, we mean that a device with packet flows in progress can continue its flows even if there are changes to: (a) its physical connectivity (e.g. moves from 3G to WLAN); or (b) its network layer (domain) connectivity (e.g. it changes from IP network A to IP network B, which might also occur when changing physical connectivity). While mobility using the same underlying technology – *horizontal handoff* (e.g. WLAN to WLAN) – is possible (usually through layer 1 or 2 handoff mechanisms), effective solutions for *vertical handoff* – across different wireless technologies and/or different networks – are still being developed. The use of vertical handoff holds many significant challenges. The key issue is managing the change in network-layer connectivity. Typically, different physical connectivity, e.g. changes in network interface, also involves changes to IP-layer connectivity – changes in the topological connectivity (*location*) of a device.

A. Contribution and structure of this paper

We describe and evaluate an implementation of a purely end-to-end approach to mobility, with management of mobile connectivity at the network layer as a first-class function of the end-system's operating system (OS), without requiring any middleboxes or changes to routing functions. Our mechanism is fully backwards compatible with existing APIs. Our mechanism uses ILNPv6, a superset of IPv6 that implements an Internet architecture described by the Identifier-Locator Network Protocol (ILNP). The challenge was to apply the radical new architectural approach of ILNP whilst making maximal use of the existing codebase, to allow incremental deployment on IPv6.

This is the first end-to-end, testbed-based performance evaluation of MIPv6 with ILNPv6, as a like-for-like comparison, with both protocols implemented in-kernel, and the use of application traffic flows from a widely-used testing application (iperf), via the standard C sockets API.

We address directly the key challenge stated above – managing the change in network-layer connectivity. In this paper, the term *handoff* always refers to a *vertical handoff*, unless specifically stated otherwise.

After outlining the ILNP architecture in Section II, some related work is described in Section III. Then, we describe our design and implementation of ILNPv6 in Section IV. We present our evaluation and results in Section V. After a discussion of some key issues in Section VI, we conclude in Section VII.

II. ILNP

The *Identifier-Locator Network Protocol (ILNP)* [1]–[8] is an IRTF Experimental protocol. It has a host-based, end-to-end architecture, is designed to support mobility (amongst other things), and can be implemented as a superset of IPv6 called *ILNPv6* [4]–[7]. ILNPv6 uses the current IPv6 packet header format, and so remains backwards compatible with the current deployed IPv6 routing infrastructure, but end-system stacks must be updated.

A. Dynamic namebindings for mobility

In support of recurring discussion within the community over several decades [9]–[14], ILNPv6 deprecates the use of IP addresses and uses two new *distinct namespaces*: a *Node Identifier (NID)* and a network *Locator (L64)* [1], along with *dynamic bindings* to implement various functionality, including mobility [2]. The use of the IP address in the network stack today has semantic overload which is considered harmful [14]. As shown in Table I, instead of using the IP address in various layers across the protocol stack with different semantics, ILNPv6 use *NID* and *L64* values. Transport-layer protocols bind only to a *NID* value, an identifier for a (logical, virtual or physical) *node*, that has no topological semantics. The network layer uses a *L64* value, which is *topologically significant*, for routing and forwarding. An $[I, L]$ pair together – an *Identifier Locator Vector (IL-V)* – is needed for communication.

In addition (not visible in Table I), there are one-to-many dynamic bindings between *NID* and *L64* values, as well as another set of dynamic bindings between physical interfaces and *L64* values. Hence, mobility in ILNP is implemented by adjusting these dynamic bindings between *NID* and *L64* values, and between *L64* values and interfaces. The *L64* values can

change as a mobile node moves without impacting end-to-end state invariance, as the *NID* value always remains stable. Mobile nodes can have multiple *NID* and *L64* values and use multiple interfaces simultaneously, by adjusting dynamic bindings between them as required.

TABLE I. USE OF NAMES IN IP AND ILNP.

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

* FQDN: Fully Qualified Domain Name

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

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

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

In tuple expression (2), we show the use of Node Identifier values, I, and Locator values, L, as for ILNP. TCP binds only to the I values, so changes to the interfaces or locator values require updates to the dynamic bindings, but does not impact the end-to-end state for TCP.

B. ILNPv6

While a clean-slate approach to building ILNP was possible, to aid deployment, we chose to implement ILNP as a superset of IPv6 – ILNPv6. In order to carry the *NID* and *L64* values between end-systems, the IPv6 packet header is used, with *NID* and *L64* values being encoded into the IPv6 header as shown in Figure 1. The *NID* value uses the same syntax as an IPv6 interface identifier, but the end-system code is updated to treat it as a node identifier. The *L64* has the same syntax and semantics as an IPv6 address / routing prefix, and so current IPv6 routing and forwarding can be used for ILNPv6 packets: routers see and process ILNPv6 packets as IPv6 packets.

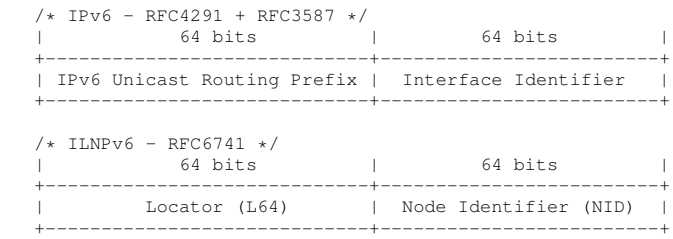


Fig. 1. IPv6 unicast address format and ILNPv6 unicast address format. The L64 value has the same syntax and semantics as the IPv6 routing prefix. The NID value has the same syntax as the IPv6 Interface Identifier, but has different semantics.

C. Mobile hosts with ILNP

For mobility, we consider two phases of communication:

- *Rendezvous:* For a correspondent node (CN) to *initiate* communication with a mobile node (MN), it must learn the current IL-V for the MN. With this information, the CN can then create a packet to send to the MN. If the MN functions only as a client system and the CN as a server, then this issue is moot.
- *Handoff:* When a communication session is in progress, the session must be transferred across an administrative or network connectivity boundary in order to maintain the communication session as the MN moves.

Rendezvous for ILNPv6 is similar to IPv6: the CN makes a Domain Name System (DNS) look-up using a Fully Qualified Domain Name (FQDN), which resolves to appropriate IL-V values in new DNS resource records for *NID* and *L64* values [5]. We return to the issue of name resolution later. For now, we consider handoff only.

Figure 2 shows a simple example of handoff in ILNPv6. An MN using NID value I_M moves from cell 1 using Locator L_1 to cell 2 and use of Locator L_2. If we assume that a transport flow is in progress with a CN which uses the IL-V $[I_C, L_C]$, then the ILNP transport-layer and network layer state at MN can be represented by the expression:

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

based on expression (2). When MN enters the region overlapping with cell 2, the value of L_2 would be available through IPv6 Router Advertisements – it is simply the IPv6 address prefix required for cell 2. MN receives L_2 and now informs CN of this new value using a *Locator Update (LU)* message, synonymous to an IPv6 Binding Update message. At this point, both MN and CN could just drop the use of L_1, but ILNPv6 permits a *NID* value to be bound to one or more *L64* values. So, it is possible for CN and MN to perform a *network-layer soft handoff* to minimise packet loss. This is advantageous when soft handoff is not supported by the sub-network technology across the handoff region, e.g. between different WLAN cells, or in vertical handoff between different technologies, such as from 3G to a WLAN cell.

Notice that in ILNPv6, mobility forms a duality with multihoming. In Figure 2, if the duration of the cell overlap permits, then MN and CN could continue to use both cells simultaneously: for example, if cell 1 was 3G and completely covered the same area as cell 2 which was a WLAN cell. This feature can be implemented by the MN to provide a multihoming resilience capability, either for existing transport protocols like TCP, or to enable new multipath transport protocols without the complex overhead of dealing with multiple IP addresses, as is currently the case with multipath TCP (MP-TCP) [15]. In the overlap region, the MN expression for our transport flow would now be:

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

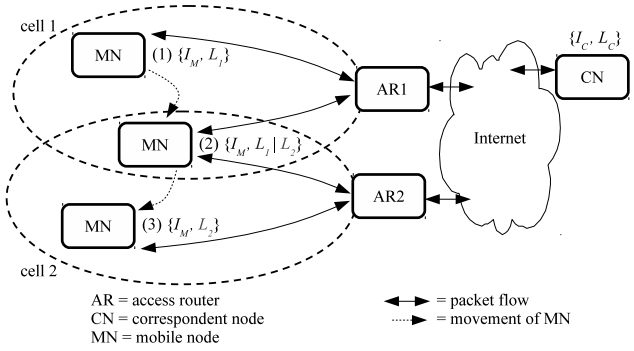


Fig. 2. An example of an ILNPv6 mobile node (MN) performing a network-layer soft handoff. (1) The MN starts in cell 1 using $[I_M, L_1]$ through access router 1 (AR1). A remote correspondent node (CN) uses $[I_C, L_C]$ for a packet flow that is in progress. (2) The MN enters the overlap between cell 1 and cell 2 and initiates handoff, e.g. because it sees IPv6 Router Advertisements from AR2 advertising prefix L_2. MN sends a *Locator Update (LU)* message to CN to inform CN that it is moving to Locator L_2. Until MN completely moves out of cell 1, it can continue to use L_1 also. (3) MN can drop the use of L_1 as it moves wholly into cell 2.

We see that the transport layer tuple is not effected during soft handoff or multihoming: end-to-end state is preserved.

ILNPv6 also provides another type of handoff called *hard handoff*. For this handoff model, an MN always has only *one L64* value. So, when entering a new network – (2) in Figure 2 – it obtains a new *L64* value, L_2, and discards the previous *L64* value, L_1. The hard handoff is simple because an MN always has binding to only one *L64* value and uses only one link. However, packet loss could occur for the in-flight packets sent from the CN using the stale *L64* value. In contrast, packet loss during soft handoff is minimised. This is because the MN maintains bindings with both *L64* values (L_1 and L_2) when it stays in the overlap region between the two networks, i.e. it is multihomed during handoff.

D. Rendezvous

With rendezvous, in ILNPv6, we are concerned with discovering the current Locator value(s) for the MN. If we wish to leverage existing deployments, then the Domain Name System (DNS) provides a suitable mechanism for rendezvous, as it does now. For ILNPv6, new DNS records have been defined for *NID* and *L64* values – NID and L64 resource records [5] – which are returned for a FQDN query.

An MN has two other requirements: (i) it needs to update the *L64* value held in its L64 DNS record; and (ii) as the MN could move at any time, the L64 record needs a low cache time. We discuss the second issue in Section VI-C, but the first issue is easily resolved: current Secure DNS Dynamic Update [16] is widely available in host system software today (e.g. in Microsoft Windows, and in Linux), as well as having support in server software, and experiments have shown that updates as frequent as once per second are easily possible with existing infrastructure [17].

E. Name and Address Resolution

When the DNS is used for storing *L64* values in L64 records (or if any other system is used for storing *L64* values), those values will need to have low cache times to prevent stale *L64* values at CNs. In the case of DNS, traditionally, it has been considered that cache times – time-to-live (TTL) – for DNS records should be kept relatively high, for example, several thousands of seconds. However, today, DNS entries for end systems can have very low values (e.g. a few seconds for DNS load balancing mechanisms), and our experiments on the operational DNS service for the School of Computer Science at St Andrews show that zero caching of DNS A records is entirely feasible without any disruption to network operations or any significant increase in load on DNS servers [18].

However, in the edge-site, this zero caching must also be applied to address resolution – mapping from an IL-V pair to a MAC/hardware address. For example, if an MN moves across a local subnet, its *L64* value will change also, and so address resolution tables must use zero caching. Again, in the local area, we take the position that this is a low and acceptable cost, though of course the actual impact is likely to be application-specific. However, for backwards compatibility with IPv6, there are no changes required to the IPv6 Neighbour Discovery (ND) protocol – the 128-bit IL-V for ILNPv6 is mapped to a MAC address, just as a 128-bit IPv6 address is mapped to a MAC address.

III. Related Work

We present here a selection of mobility solutions, focussing on architectural concepts for those proposals that have been reviewed by the IETF or the IRTF, i.e. those that are considered to be deployable at scale. A more comprehensive list of mobility solutions can be found in RFC6301 [19].

A. Network-based Mobility Solutions

Mobility management in this type of solution is usually achieved by the use of proxies or middleboxes. These can often improve backwards compatibility because they 'hide' mobility from non-mobile (legacy) nodes. However, the addition of such network entities adds complexity to the current network landscape. Such entities have a set of general disadvantages: they offer sub-optimal packet routes for data transfer; they could become single points of failure; they could become performance bottlenecks; and they could offer an additional point for security attacks on the mobility mechanism. Sometimes, tunnelling is also used for communication between those entities, increasing packet overhead and potentially impacting the MTU size that is available to the application.

IETF Mobile IPv4 (MIPv4) [20] uses a Home Agent (HA) and a Foreign Agent (FA) to map between a Home Address (functionally, a node *identifier*) and Care-of-Address (functionally, a node *locator*). Mobile IPv6 (MIPv6) [21] eliminates the use of the FA and requires only the HA. MIPv4 uses tunnelling between HA and FA, while MIPv6 introduces a *Route Optimisation* mechanism to eliminate tunnelling. Nevertheless, MIPv4 and MIPv6 have the problem of high packet loss during handoff, and high handoff delay.

Hierarchical Mobile IPv6 (HMIPv6) [22] introduces a Mobility Anchor Point (MAP) to work with the HA in managing local mobility. Handoff delay could be reduced, but yet another entity is now part of the mobility solution, with additional signalling to/from the MAP.

Fast Handover for Mobile IPv6 (FMIPv6) [23] (based on MIPv6) uses a HA as a proxy. Additional tunnelling is

used between the Previous Access Router (PAR) and the New Access Router (NAR) to forward packets arriving at the previous location to minimise gratuitous packet loss during handoff [24]. There is additional complexity and overhead due to the extra signalling required to allow use of NAR and PAR.

Proxy Mobile IPv6 (PMIPv6) [25] is a completely network-based solution: it hides the mobility process from mobile nodes. Additional proxies are required: a Mobile Access Gateway (MAG) and Local Mobility Anchor (LMA). Traffic between MAG and LMA is also tunnelled.

The Locator Identifier Separation Protocol (LISP) [26] as well as its extensions for mobility support, LISP mobile node (LISP-MN) [27] and LISP-ROAM [28], use a proxy – *mapping system* – to map IP addresses into different schemas: Endpoint Identifier (EID) and Routing Locator (RLOC). Tunnelling is also used between LISP routing nodes. LISP-based solutions are 'map-encap' solutions, so end-to-end state integrity is not maintained, and the IP address carried in the packet has different semantics across the end-to-end path.

B. Host-based Mobility Solutions

Host-based solutions usually do not require any additional network entities. Mobility management is handled by end hosts. Therefore, the end-system protocol stack requires updates. However, we consider that deployment of such updates could be easily managed – similar to network-based software updates widely-used for operating systems today.

Level 3 Multihoming Shim Protocol for IPv6 (SHIM6) [29] adds a 'shim' layer between the network and the transport layer to separate identifier and locator from a single IP address. SHIM6 is designed for multihoming support. Mobility support is possible, but has the problem of high handoff latency [30].

The Host Identity Protocol (HIP) [31], [32], which was recently updated to HIPv2 [33], uses public and private key pairs to manage the identifier (i.e public key) and locator (i.e. IP address) of a mobile node. Although DNS may be used for session initiation, a *Rendezvous Server (RVS)* – an additional entity – is recommended to be deployed. Also, as public keys are required, a public key infrastructure is recommended for use with HIP.

C. Transport Layer Solutions

All solutions mentioned above operate at the *network layer*. There are also proposals for *transport layer* solutions. However, such solutions, of course, are limited to the choice of transport layer protocol, while, in principle, network layer solutions are applicable to any transport protocol. Moreover, most transport layer solutions typically manipulate IP addresses, which means the fundamental problem of semantic overload of the IP address remains. Additionally, some security issues remain unresolved.

The Stream Control Transmission Protocol (SCTP) [34] allows a host to set up multiple paths – with multiple source and destination address combinations – between a source and a destination host. SCTP is designed for multihoming support. Single host mobility (one-side mobile host) can be achieved by dynamically adding and deleting addresses of active SCTP associations [35]. To support two-side mobile hosts, a *Cooperation Server (SC)* – a new network entity – is required [36]. SCTP has some security issues, which are summarised in RFC5062 [37].

Multipath TCP (MP-TCP) [15], [38] extends TCP and allows a main TCP session to have bindings to different addresses i.e. multiple *sub-flows*. Like SCTP, MP-TCP is designed for multihoming support. Mobility using MP-TCP could be achieved by dynamically adding and removing sub-flows when a host enters and exits a network [39]. MP-TCP is backwards compatible with classic TCP. However, security issues remain [40] and a new a cross-path interference attack has been identified [41].

D. Previous work on ILNP

The initial ideas of host mobility using ILNP can be found in [2], [42], [43], and assessment of its feasibility using an overlay emulation can be found in [44]. The overlay emulation was used for initial evaluation of handoff performance under a range of emulated scenarios. The architectural description and protocol engineering considerations for ILNP are given in RFC6740 [3] and RFC6741 [4], respectively.

The first ILNP mobility prototype in Linux is described in [45], which presented a simplistic evaluation using a custom designed UDP application running on a *wired* network to demonstrate satisfactory overall protocol behaviour. No wireless experiments were conducted, and the test application was custom built.

In this paper, we use an updated codebase with a more complete implementation of ILNPv6, and present new results:

- we provide results from a Linux kernel codebase that allows use of ILNPv6 via the normal *socket(2)* API with *getaddrinfo(3)*: this is demonstrated by using an unmodified, existing UDP application, *iperf*, which is widely used for performance testing;
- our evaluation is based on a comprehensive set of testbed experiments with *wireless* connectivity (5GHz IEEE 802.11 WLAN), providing for a more realistic evaluation in a mobile / wireless domain;
- we provide a direct performance comparison, on the same testbed, with the Linux kernel implementation of Mobile IPv6 (MIPv6): experiments compare ILNPv6 hard handoff and soft handoff with MIPv6 and route optimisation enabled and disabled.

IV. ILNPv6 in Linux

ILNP is a radical departure from the current Internet architecture as it deprecates the use of IP addresses. However, judicious engineering allows an evolutionary approach to enable incremental deployment. An overview of our first ILNPv6 prototype can be found in [45]. Due to space limitations, we have not provided the full details of the kernel design and implementation of our prototype. We present here a brief description. Our implementation is based on Ubuntu 12.04 with Linux kernel version 3.9.0. We describe how (i) *NID* and *L64* values replaced IPv6 addresses; (ii) OS name resolution was changed; and (iii) the changes compared to our previous prototype, plus the new results specifically for this paper.

A. Encoding NID and L64 values

As shown in Figure 1, *L64* and *NID* values are encoded into the IPv6 address space [4]. The top 64 bits, *L64*, have the same syntax and semantics as an IPv6 routing prefix. The value is obtained from an IPv6 Router Advertisement (RA). The lower 64 bits, *NID* , has the same syntax as the IPv6 Interface Identifier, but different semantics. The *NID* represents a whole node, not a specific interface of the node. The *NID* value can be constructed in exactly the same way for ILNPv6 as it is for IPv6. For convenience, the *NID* value in this implementation was derived from the MAC address of the first active interface, but other methods, such as cryptographically generated addresses (CGA) [46] or privacy extensions for IPv6 addresses [47] could be used also.

B. Name resolution

For session initiation, ILNP can use DNS [3], [5] just like IPv6 and MIPv6. However, in this implementation, our focus was to study the OS handoff performance, so we used a modified version of `/etc/hosts` and *getaddrinfo()* (in *libc*) for name resolution. An ILNPv6 hostname with *NID* and *L64* values has an entry in `/etc/hosts` with the format:

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

Then, `getaddrinfo()` extracts *NID* , *L64* and preference values (the latter is currently unused), and passes them to the kernel via the *Netlink Socket*, to be stored in the *IL-V cache*, a new data structure in the kernel. To maintain backwards compatibility with the *sockets* API, the `getaddrinfo()` caller will receive an IPv6 'lookalike' address which is built from *NID* and *L64* values (see Figure 1). So, well-behaved legacy applications (those that use the socket descriptor only, and do not use address bits for application state) will work with ILNPv6 as they would with IPv6.

C. Identifying ILNPv6 packets

Essentially, the main IPv6 code-path within the Linux kernel has been augmented to process ILNPv6 packets. So, the modified code-path can support both IPv6 and ILNPv6 in parallel, treating ILNPv6 as a superset of IPv6. This is beneficial for backwards compatibility with IPv6, as well as for allowing a realistic path for incremental deployment.

To detect ILNPv6 packets for outgoing flows, the destination IP address is checked against the IL-V cache. If the address is in the cache, ILNPv6 is used for communication instead of IPv6. The socket is marked as an ILNPv6 socket using a new flag in the socket data structure, (`struct sock`). To allow differentiation between IPv6 packets and ILNPv6 packets, and also to provide off-path protection for packets, a nonce value is added to every ILNPv6 packet before sending, using a *Nonce Destination Option (NDO)* (ICMPv6 type 139) [7]. For each incoming packet, an ILNPv6 packet is detected if it carries a nonce value. Please refer to [45] for further details of additional mechanism for sending and receiving ILNPv6 packets.

D. Transparent support for the socket(2) API

The previous ILNP prototype [45] had support for UDP applications using only `sendto(2)` and `recvfrom(2)`. In this updated codebase, UDP applications also can now use the `connect(2)`, `read(2)` and `write(2)` calls, allowing any existing IPv6 applications, such as *iperf* to operate over ILNPv6. We modified the Linux UDP lookup function to use only *NID* for the *udptable* lookup. The *udptable* lists all current UDP sessions of the host and is used by the kernel to track which port and application incoming packets should be forwarded to. So, any change of *L64* value does not affect the table lookup because only *NID* is now used. This means legacy (non-ILNPv6) application binaries can use ILNPv6 (no re-coding or recompilation required).

V. Evaluation

We examined handoff performance by comparing Mobile IPv6, both with route optimisation (RO) enabled and disabled, and with ILNP using hard handoff and soft handoff. We considered the impact on UDP flows.

Note that HIP (see Section III-B), is considered out of scope for this evaluation due to two important constraints:

1) *HIP uses public keys.* HIP uses public keys in order to create host identities. This means that a public key infrastructure should be in place to generate host identities, and so HIP is best suited to those applications that have stringent requirements for the use of cryptographically verifiable identities at the network layer – this is not a general requirement for all applications, and is not used in MIPv6 or ILNPv6.
2) *Applications need to be modified to use HIP.* The use of the host identity in HIP requires that applications that use the current standard *C sockets* API have to be modified to operate over HIP [48]. This is not the case for either MIPv6 or ILNPv6.

These two constraints prevent a straight-forward comparison of performance with HIP for typical applications.

A. Experiment Configuration

Our testbed was configured as in Figure 3. R1, R2, R3, CN and MN were each separate, physical hosts. The following connections were *wired* Ethernet 1Gbps connections: CN to R1, R1 to R2, and R2 to R3. The MN used 5GHz 802.11ac WLAN links. We used *netem* [1] to add extra delay of 100ms in each direction between R1 and R2 and between R1 and R3 for emulating WAN access. We created 4 MN handoff scenarios as follows:

1) LAN to LAN (*netem* disabled)
2) LAN to WAN (*netem* enabled between R1 and R3)
3) WAN to LAN (*netem* enabled between R1 and R2)
4) WAN to WAN (*netem* enabled both between R1 and R2 and between R1 and R3)

The routers R2 and R3 were separate hosts using an *unmodified* Linux OS running *hostapd* [2] and *radvd* [3] to act as IPv6 access points announcing IPv6 address prefixes for the site networks L_2 and L_3. To allow the mobile host to pick

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

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

[3] http://www.litech.org/radvd/

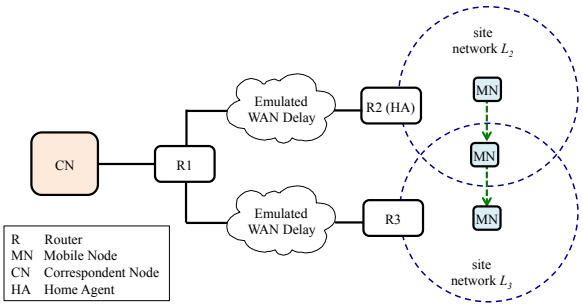


Fig. 3. The topology for the experiment. The CN connects to R1 via 1Gbps ethernet. The MN initially connects to R2(HA) using WLAN – the dashed / blue circles depict the radio cell scenario being emulated. The green / dashed arrows identify movements of MN to site network L_3 generating a handoff.

up new prefixes quickly once it enters a new network, we configured *radvd* to generate RAs every 1s-2s. The MN had two WLAN interfaces – one was configured to connect to R2, the other for connection to R3.

Note that only the end-hosts MN and CN used our modified Linux kernel supporting ILNPv6. Routers R1, R2 and R3 used standard Linux installations, with netem configured to implement the clouds marked 'Emulated WAN Delay' in Figure 3. Both CN and MN were also configured to support MIPv6 using *umip*[4], to allow a direct comparison between MIPv6 and ILNPv6 on the same infrastructure.

We used *iperf*[5] to generate *bi-directional* UDP flows at two different mean bit rates as simple packet-level representations of the following application flows: i) 64 kbps (for Skype Voice over IP (VoIP) traffic based on [49]); and ii) 2350 kbps (for Netflix High Definition (HD) video traffic based on [50]). We used packet sizes of 300 bytes for VoIP traffic [51], and 1300 byte packets for HD video traffic. Each flow lasted 30 seconds and was repeated 10 times for each combination of the two flows, and for MIPv6 (with and without RO enabled), as well as for ILNPv6 hard handoff and ILNPv6 soft handoff. *tcpdump* was used to capture packets at the MN for analysis. In Figure 3, the MN started in the site network L_2, which was the Home Network for MIPv6. The bi-directional *iperf* flows were sent between the CN and the MN. As each flow was in progress, the MN started to enter the site network L_3 at t=5s into the flow, it moves out of the site network L_2 at t=20s i.e. the MN stayed in the overlap area for 15s. The movement was emulated using *ifconfig* to bring the WLAN interfaces up and down. We repeated the test for 4 handoff scenarios listed above: LAN to LAN, LAN to WAN, WAN to LAN, and WAN to WAN.

B. Results

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

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

The throughput and packet loss were measured from time t=20s to t=25s of each flow for the MIPv6 cases, and from t=11s to t=16s for the ILNPv6 cases. These durations are selected to cover the handoff period of every test scenario. The handoff in MIPv6 and ILNPv6 happens at different times. In MIPv6, it is triggered when the MN moves out of the previous network completely [21, Sec. 11.5]. This is a *link layer* trigger e.g. the previous link is down or is no longer reachable. For ILNPv6, it is a *network layer* trigger i.e. handoff after seeing a router advertisement (RA) from the new network. In this experiment, however, the MN did not handoff immediately after seeing the new RA. There is a delay of 2s because we need to wait for the Duplicate Address Detection (DAD) process to be completed, so the new address (new *L64*) can be used. Therefore, in this experiment the MN performed handoff around the t=21s to t=23s, for MIPv6 and around t=13s to t=15s, for ILNPv6.

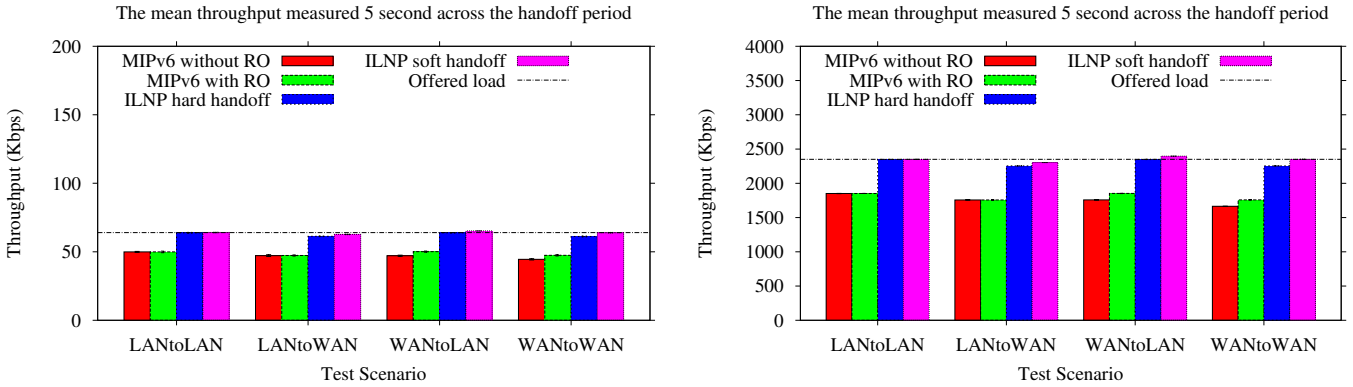
The throughput of the flow, measured for a 5 second span across the handoff period, is shown in Figure 4. A similar trend is found in both VoIP traffic and HD Video traffic. The results for ILNPv6, especially with soft handoff, showed close to ideal throughput (equal to the offered load). A small drop-off in throughput was observed when hard handoff was used. For MIPv6, the throughput was substantially reduced because the MN could not receive any packets during handoff – see Figure 6 for an example of the data flow progress during the handoff period. There was a small improvement of throughput for MIPv6 when RO was enabled, but only if the home network was a WAN link (i.e. when handing off from a WAN): sending data directly from CN to MN was better than traversing the HA, which resided on a path with a longer delay.

Figure 5 summarises packet loss during a 5 second span across the handoff period. The results were similar for both VoIP traffic and HD Video traffic. ILNPv6 with soft handoff again outperformed MIPv6: *zero* packet loss was observed for ILNPv6. A small loss was found when ILNPv6 hard handoff was used. MIPv6 suffers from greater packet loss because, again, the MN cannot receive any packets during the handoff. Also, again, when handing off from the WAN, use of RO reduced slightly the overall packet loss during handoff, as packets do not traverse the HA.

Figure 6 gives an example of the flow dynamics during the 5 second span across the handoff period. When MIPv6 was in use, there was an interruption of the flow, which was the duration of the Binding Update to the HA. When RO was used, we saw a peak of received data when the RO process was completed, because some packets directly sent from the CN arrive at the same time as the delayed packets that have been relayed by the HA. For ILNPv6 hard handoff, a small interruption was observed, which was the duration of the LU/LU-ACK handshake. However, there was no interruption for ILNPv6 with soft handoff.

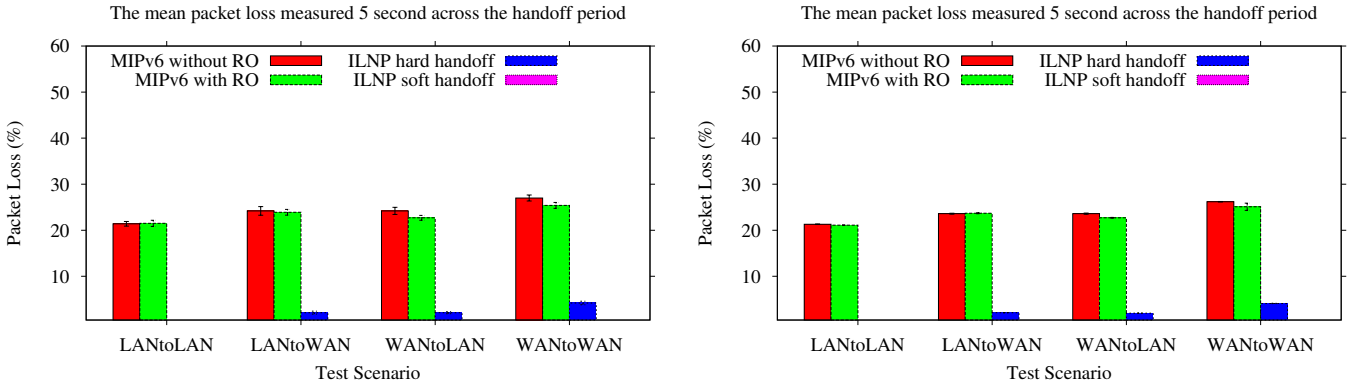
The observed handoff delay is shown in Figure 7 and Figure 8. This was observed *network layer handoff delay*, measured as the handoff signalling for MIPv6 (BU/BU- ACK)

[4]http://umip.org/
[5]https://iperf.fr/



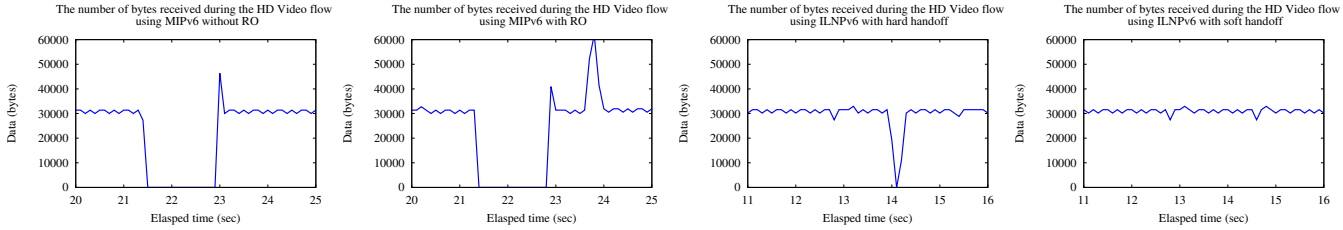
(a) VoIP traffic. (b) HD Video traffic.

Fig. 4. Throughput for 5 second across the handoff period for our UDP emulated traffic flows. (Higher values are better, up to the offered load. Error-bars are small so might not be visible.)



(a) VoIP traffic. (b) HD Video traffic.

Fig. 5. Packet loss for 5 second across the handoff period for our UDP emulated traffic flows. ILNP soft handoff has *zero* loss hence it not visible in the bar charts. (Lower values are better: ideal is *zero* loss. Error-bars are small so might not be visible.)



(a) MIPv6 withput RO. (b) MIPv6 with RO. (c) ILNP hard handoff. (d) ILNP soft handoff.

Fig. 6. Example graphs showing number of bytes received at the MN during the 5 second across the handoff period for HD Video traffic, WAN to WAN handoff. (A flat, stable line at the value 30000, without discontinuities, is ideal.)

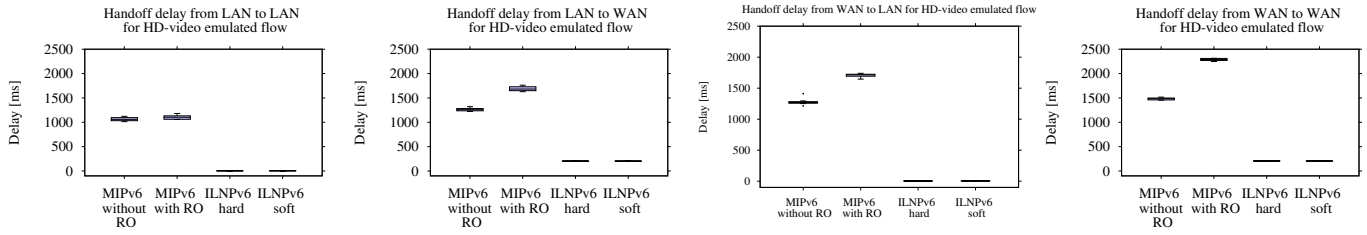
and ILNPv6 (LU/LU-ACK). Other signalling delays such as wireless association, Neighbour Discovery, and Duplicate Address Detection (DAD) were excluded for clarity. Again, ILNPv6 provided better performance than MIPv6 in terms of shorter handoff delay. The LU/LU-ACK handshake in ILNPv6 usually takes ∼1 RTT between MN and CN, as it is purely end-to-end. Hence, delay is a few ms, when handing off to the LAN, and is ∼200ms when handing off to the WAN, because ILNPv6 sends the LU and receives the LU-ACK via the new link. For MIPv6 without RO, the BU/BU-ACK handshake takes around 1s plus 1 RTT to the HA. The extra 1s is due to BU processing and tunnel creation at the HA before the BU-ACK is sent to the MN. When RO is enabled the handoff delay is higher because there are additional processes: the *return routability test* and the *binding update to the CN*. These processes take longer over the WAN path.

VI. Discussion

We make comparisons with Mobile IP, where appropriate. For Sections VI-D and VI-E below, full security and privacy implications are a complex issue: a detailed discussion and analyses is left for further study.

A. Multi-homed hosts

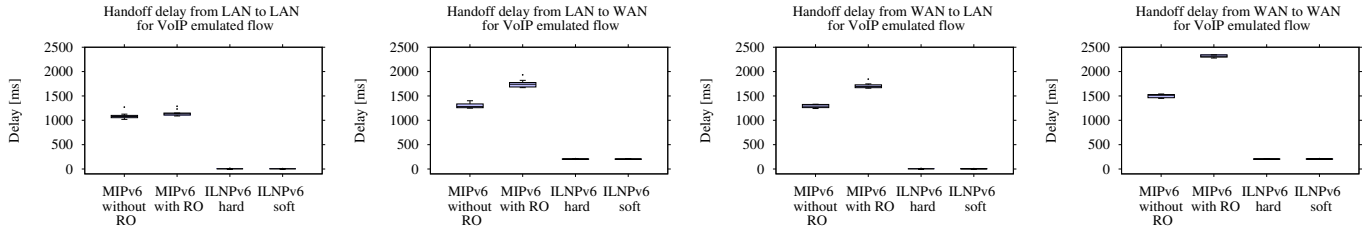
Multihoming is closely related to mobility: both functions require special treatment of location with respect to an end-



(a) LAN to LAN handoff. (b) LAN to WAN handoff. (c) WAN to LAN handoff. (d) WAN to WAN handoff.

Fig. 7. Handoff delay for our HD Video emulated traffic flows. ILNP handoff is ∼1 RTT. (Lower values are better: 1 RTT is ideal. Error-bars are small so not be visible.)

s



(a) LAN to LAN handoff. (b) LAN to WAN handoff. (c) WAN to LAN handoff. (d) WAN to WAN handoff.

Fig. 8. Handoff delay for our VoIP emulated traffic flows. ILNP handoff is ∼1 RTT. (Lower values are better: 1 RTT is ideal. Error-bars are small so not be visible.)

host (or a site) that is multihomed. MIP requires additional mechanisms to achieve multihoming [52], [53]. In ILNP, multihoming and mobility form a duality [1], [2], [54], so mobile hosts can also be multihomed.

Indeed, if we consider the soft handoff scenario in Figure 2, we see that the MN is multihomed during handoff, as the *NID* can be bound to more than one *L64* simultaneously. Hence, it is possible that a mobile host connects to different wireless technology, say, both WiFi and 3G simultaneously. This could enable functions such as WiFi offloading, for example, using 3G for signalling and small data packets, and using WiFi for bandwidth-consuming data. It could also help to manage QoS differences in the transition between networks when vertical handoff is used. However, a multihomed mobile host with ILNPv6 is a subject for further study.

B. Backwards compatibility with IPv6

An IPv6 router will treat ILNPv6 packets as if they were IPv6 packets. Our evaluation shows that unmodified Linux IPv6 routers can be used for routing and forwarding ILNPv6 packets from/to ILNPv6 hosts. Therefore, we believe that ILNPv6 could be deployed in the current IPv6 backbone without requiring any changes or upgrades to IPv6 routers.

To enable backwards compatibility with IPv6 hosts, e.g. when an ILNPv6 host communicates with an IPv6 host, the IL-V cache is used to verify if the destination supports ILNPv6 or not (see Section IV-C). However, a falsely detected ILNP host could be flagged with the use of the nonce value [7]. When a non-ILNP host receives a packet with a nonce value, a 'Parameter Problem' ICMP packet is returned to the sender as part of normal IPv6 host operation. Hence, the sender is notified if the destination host does not support ILNPv6. A new communication session would then be re-established with classic IPv6 [4].

C. Name resolution using DNS

In this implementation, we use `/etc/hosts` for name resolution. In a real deployment, DNS would be used. New DNS Resource Records for *NID* and *L64* values have been defined [5] and are implemented in widely-used DNS software: ISC BIND/*named* from v9.9.3[6], and NSD/*Unbound* from v3.2.15[7].

Since the *L64* value of the MN would be updated in the DNS when the node moves, DNS records that hold *L64* values need to have a very low DNS time-to-live (TTL), or else cached *L64* values would become stale. Our previous empirical evaluation shows that values of TTL as low as zero for IPv4 *A* records have no significant impact on DNS load [18], so use of low DNS TTL for the *L64* records on mobile ILNP hosts would have little impact on DNS load. Both BIND and Unbound also support DNS security for secure dynamic DNS updates of *L64* resource records.

D. Security considerations

DNS security is used for secure update of *L64* value to the DNS, as mentioned before. For MIPv6, IPsec is required for a secure Binding Update to the HA [21, Sec. 15.3]. For other general operations, ILNPv6 is as secure as IPv6.

SSL/TLS and other mechanisms that work over TCP should work, unless they make use of the network layer address bits, i.e. if higher level protocols use FQDNs, they should work without requiring any changes.

For ILNPv6, for providing lightweight protection against modification and forged packets, we introduce the use of a *Nonce Destination Option (NDO)*, a new IPv6 end-to-end

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

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

extension header [7]. This carries an unpredictable, cryptographically random value to initiate a session. It is then used subsequently as required. The nonce value is sufficient to protect against off-path attacks: any attacker without an on-path monitoring capability will not be able to see and use the Nonce value. The NDO should be used, as default, at the very least on packets that carry control messages, for example to provide off-path protection of Locator Update messages.

Where the threat regime is such that the NDO is not considered sufficient protection, ILNPv6 is compatible with IPsec [55] with one change: IPsec Security Associations (SAs) are bound to *NID* values and not 128-bit addresses as they are in IPv6 [2]. Note that while this change is architecturally significant, it is a relatively small engineering change. So, as long as the *NID* values do not change, the IPsec SA remains valid. Of course, the other features of IPsec – Authentication Header (AH) and Encapsulating Security Payload (ESP) – are also usable with ILNPv6.

Some applications may require assurances of identity of the source at the packet level, for example, to provide per-packet authentication. IPsec can be used for this purpose. However, if required, *NID* values for ILNPv6 nodes can also be generated as used for Cryptographically Generated Addresses (CGAs) for IPv6 [46].

ILNP also has the capability to protect whole network sites: various scenarios are possible [8], including site resilience [56], [57], traffic engineering [58], mobile site networks [59], heterogeneous edge networks [60], and virtual machine mobility across local and wide-area network paths [57].

E. Privacy considerations

The *NID* value is essential as the end-to-end invariant for transport protocol session state, and might also be used by application protocols, though we encourage use of an FQDN or application-specific namespace with ILNPv6. ILNP allows multiple *NID* values to be used by a node simultaneously, as long as any transport session uses the same *NID* value during its lifetime, to maintain end-to-end session state invariance. So, in support of identity privacy, *NID* values could be ephemeral values, generated as required [4, Sec. 2 & 11] [8, Sec. 8]. For example, a client system could generate 'random' *NID* values for use for different transport layer sessions (using IPv6 Duplicate Address Detection to check for collisions in the *NID* */L64* that is created).

In support of privacy, ILNPv6 can leverage another existing IPv6 recommendation for generating anonymous, ephemeral *NID* values as required [47]. A node can generate a new *NID* value shortly before initiating communication. Normally, such requirements may be for client systems accessing services, and one use may be to prevent tracking of users through the default *NID* value that is derived as for IPv6. However, if a mobile node expects incoming connections, then the distribution of the new identity would be by application-specific means. If DNS was used, and re-writing values of *L64* records was not appropriate, the TXT record [61] could be used, again in an application-specific manner, in conjunction with Secure DNS Dynamic Update.

VII. Conclusion and Future Work

We have shown that handoff management for IP mobile nodes can be implemented as a purely end-to-end function in the Linux OS kernel. Our implementation – ILNPv6 – is implemented as a superset of IPv6, so although a radically different naming architecture is used – based on *identifiers* and *locators* – it is still possible to operate over existing IPv6 infrastructure. ILNPv6 could be integrated into existing OS bases and deployed incrementally. We have shown that an ILNPv6-modified Linux kernel can support IPv6 binaries.

Our performance evaluation shows that ILNPv6, especially with soft handoff, provides excellent handoff performance in terms of throughput, packet loss and handoff delay with respect to UDP flows. Unlike MIPv6, there is minimal disruption during handoff (zero packet loss was observed for ILNPv6 soft handoff in our experiments). ILNPv6's end-to-end architecture means that additional entities, such as home agents, are not required, neither are tunnels.

For the future, we plan to examine the performance of TCP flows over ILNPv6, as well as explore multihomed mobile scenarios. We also wish to explore the possibilities of exploiting DNS, especially for mobility of whole networks, for which ILNP also provides support.

References

[1] R. Atkinson, S. Bhatti, and S. Hailes, "Evolving the Internet Architecture Through Naming," *IEEE JSAC*, vol. 28, no. 8, pp. 1319–1325, Oct 2010.

[2] ——, "ILNP: Mobility, Multi-homing, Localised Addressing and Security Through Naming," *Telecomm. Systems*, vol. 42, no. 3, pp. 273–291, Dec 2009.

[3] R. Atkinson and S. N. Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description," IRTF, RFC 6740 (E), Nov 2012. [Online]. Available: http://tools.ietf.org/html/rfc6740

[4] ——, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations," IRTF, RFC 6741 (E), Nov 2012. [Online]. Available: http://tools.ietf.org/html/rfc6741

[5] R. Atkinson, S. N. Bhatti, and S. Rose, "DNS Resource Records for the Identifier-Locator Network Protocol (ILNP)," IRTF, RFC 6742 (E), Nov 2012. [Online]. Available: http://tools.ietf.org/html/rfc6742

[6] R. Atkinson and S. N. Bhatti, "ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)," IRTF, RFC 6743 (E), Nov 2012. [Online]. Available: http://tools.ietf.org/html/rfc6743

[7] ——, "IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)," IRTF, RFC 6744 (E), Nov 2012. [Online]. Available: http://tools.ietf.org/html/rfc6744

[8] ——, "Optional Advanced Deployment Scenarios for the Identifier-Locator Network Protocol (ILNP)," IRTF, RFC 6748 (E), Nov 2012. [Online]. Available: http://tools.ietf.org/html/rfc6748

[9] C. Bennett, S. Edge, and A. Hinchley, "Issues in the Interconnection of Datagram Networks," Internet Experiment Note (IEN) 1, Jul 1977.

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

[11] B. Carpenter, "Architectural Principles of the Internet," Internet Architecture Board, RFC 1958 (I), Jun 1996.

[12] B. Carpenter, J. Crowcroft, and Y. Rekther, "IPv4 Address Behaviour Today," Internet Architecture Board, RFC 2101 (I), Feb 1997.

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

[14] B. E. Carpenter, "IP Addresses Considered Harmful," *SIGCOMM CCR*, vol. 44, no. 2, pp. 65–69, Apr 2014. [Online]. Available: http://doi.acm.org/10.1145/2602204.2602215

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

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

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

[18] S. N. Bhatti and R. Atkinson, "Reducing DNS Caching," in *GI2011 – 14th IEEE Global Internet Symp.*, Apr 2011.

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

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

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

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

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

[24] E. Ivov and T. Noel, "An experimental performance evaluation of the IETF FMIPv6 protocol over IEEE 802.11 WLANs," in *IEEE WCNC 2006*, vol. 1, Apr 2006, pp. 568–574.

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

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

[27] A. Rodriguez Natal, L. Jakab, M. Portoles, V. Ermagan, P. Natarajan, F. Maino, D. Meyer, and A. Cabellos Aparicio, "LISP-MN: Mobile Networking Through LISP," *Wireless Personal Communications*, vol. 70, no. 1, pp. 253–266, 2013.

[28] A. Galvani, A. Rodriguez-Natal, A. Cabellos-Aparicio, and F. Risso, "LISP-ROAM: Network-based Host Mobility with LISP," in *MobiArch 2014*, 2014, pp. 19–24. [Online]. Available: http://doi.acm.org/10.1145/2645892.2645898

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

[30] A. Dhraief and N. Montavont, "Toward Mobility and Multihoming Unification - The SHIM6 Protocol: A Case Study," in *IEEE WCNC 2008 - Wireless Comms. and Networking Conf.*, March 2008, pp. 2840–2845.

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

[32] P. Nikander, T. Henderson (Ed), C. Vogt, and J. Arkko, "End-host mobility and multihoming with the host identity protocol," IETF, RFC 5206 (E), Apr 2008.

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

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

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

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

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

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

[39] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley, "Opportunistic Mobility with Multipath TCP," in *ACM MobiArch 2011*. ACM, 2011, pp. 7–12.

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

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

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

[43] R. Atkinson, S. N. Bhatti, and S. Hailes, "Mobility as an Integrated Service Through the Use of Naming," in *MobiArch 2007 - 2nd ACM/IEEE Intl. Workshop on Mobility in the Evolving Internet Architecture*, Aug 2007, pp. 1:1–1:6, http://goo.gl/lkhok. [Online]. Available: http://saleem.host.cs.st-andrews.ac.uk/publications/2007/mobiarch2007/mobiarch2007-abh2007.pdf

[44] D. Phoomikiattisak and S. N. Bhatti, "Network Layer Soft Handoff for IP Mobility," in *PM2WH2N 2013 - 8th ACM Wrkshp. Perf. Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, Nov 2013, pp. 13–20.

[45] ——, "IP-layer Soft Handoff Implementation in ILNP," in *MobiArch 2014*, 2014, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2645892.2645895

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

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

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

[49] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype traffic: when randomness plays with you," in *SIGCOMM 2007*, 2007, pp. 37–48.

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

[51] K. Chen, C. Huang, and C. Huang, P.and Lei, "Quantifying Skype user satisfaction," in *SIGCOMM 2006*, 2006, pp. 399–410.

[52] K. Nagami, S. Uda, N. Ogashiwa, H. Esaki, R. Wakikawa, and H. Ohnishi, "Multihoming for Small-Scale Fixed Networks Using Mobile IP and Network Mobility (NEMO)," IETF, RFC 4908 (E), Jun 2007.

[53] C. Ng, T. Ernst, E. Paik, and M. Bagnulo, "Analysis of Multihoming in Network Mobility Support," IETF, RFC 4980 (I), Oct 2007.

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

[55] S. Kent and S. Keo, "Security Architecture for the Internet Protocol," IETF, RFC 4301 (PS), Dec 2005.

[56] R. Atkinson, S. Bhatti, and S. Hailes, "Harmonised Resilience, Security and Mobility Capability for IP," in *IEEE MILCOM 2008*, Nov 2008.

[57] S. Bhatti and R. Atkinson, "Secure & Agile Wide Area Virtual Machine Mobility," in *IEEE MILCOM 2011*, Oct 2012.

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

[59] D. Rehunathan, R. Atkinson, and S. Bhatti, "Enabling Mobile Networks Through Secure Naming," in *IEEE MILCOM 2009*, Oct 2009.

[60] S. Bhatti, R. Atkinson, and J. Klemets, "Integrating Challenged Networks," in *IEEE MILCOM 2011*, Nov 2011.

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