IP without IP addresses

Saleem N. Bhatti
School of Computer Science
University of St Andrews
St Andrews, UK
saleem@st-andrews.ac.uk

Ditchaphong Phoomikiattisak
GISTDA
Bangkok, Thailand
ditchaphong@gistda.or.th

Bruce Simpson
Consultant
London, UK
bms@FreeBSD.org

ABSTRACT

We discuss a key engineering challenge in implementing the Identifier- Locator Network Protocol (ILNP), as described in IRTF Experimental RFCs 6740-6748: enabling legacy applications that use the C sockets API. We have built the first two OS kernel implementations of ILNPv6 (ILNP as a superset of IPv6), in both the Linux OS kernel and the FreeBSD OS kernel. Our evaluation is in comparison with IPv6, in the context of a topical and challenging scenario: host mobility implemented as a purely end-to-end function. Our experiments show that ILNPv6 has excellent potential for deployment using existing IPv6 infrastructure, whilst offering the new properties and functionality of ILNP.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Neworks**]: Network Architecture and Design; C.2.2 [**Computer Communication Neworks**]: Network Protocols

General Terms

Design, Performance

Keywords

ILNP; Identifier-Locator; Mobility; Internet architecture; IPv6

1. INTRODUCTION

An historic perspective on the (mis)use of IP addresses is given in [1], which describes how IP address values have been widely used for engineering convenience, whilst it is widely acknowledged that such usage is undesirable, architecturally. The key issue is one of semantic overload: IP addresses have been used to represent both the *identity* of an end-system, and also used for routing, acting as a (topological) *locator* value. Additionally, IP addresses are used in end-to-end protocol state (e.g. in TCP and UDP protocol control blocks), and are bound to physical interfaces.

Consequently, the end-to-end state for a TCP connection end-point, for example, is bound to a single, specific interface on a host. This makes some functionality, such as mobility and multihoming, cumbersome to implement directly in IP, requiring additional state information outside the end-host. For example, IP mobility requires the use of additional proxies (e.g. the Home Agent) [2], and multihoming requires the use of multiple entries in the routing infrastructure [3]. In such cases, the end-to-end semantics of the connection may be compromised.

1.1 Identifiers and Locators

The use of Identifier / Locator (*Id/Loc*) approaches in network architecture and protocol design result in a wide range of solutions. The IRTF Routing Research Group (RRG)[1] undertook a comprehensive, open review of 15 proposals that could be classified as Id/Loc mechanisms, with the goal of recommending a scalable routing and addressing architecture for the Internet. In its conclusion, reported as RFC6115, "Recommendations for a Routing Architecture", the RRG Chairs recommended ILNP [4, Section 17].

Whilst ILNP presents a clean architecture, there are some significant engineering challenges in implementing ILNP across the existing infrastructure. ILNP does not use IP addresses, and instead explicitly introduces new data-types: Identifier and Locator values. Current end-system stacks, some popular APIs, and many applications make strong assumptions about the use of IP addresses. Indeed, the C sockets API predates the existence of the Domain Name System (DNS), and so C programmers are exposed directly to IP address values, often using them explicitly, rather than using Fully-Qualified Domain Names (FQDNs) or application level

[1]https://irtf.org/concluded/rrg concluded 10 June 2014

AINTEC '16, Nov 30-Dec 02, 2016, Bangkok, Thailand

DOI: http://dx.doi.org/10.1145/3012695.3012701

names, as recommended in RFC 1958 [5, Section 4][2].

1.2 Contribution and outline

Based on the first two implementations of ILNPv6 in kernel space (Linux and FreeBSD), we discuss a key engineering challenge in deploying ILNPv6: supporting legacy applications, i.e. running IPv6 applications without modification over ILNPv6. Section 2 provides an introduction to ILNP and highlights the importance of this challenge. A solution for an unmodified IPv6 binary for *iperf* operating over ILNPv6 is given in Section 3, with a comparative performance evaluation against Mobile IPv6, plus a critique, in Section 4. We then present a wider discussion on applicability of ILNPv6 in Section 5, and conclude in Section 6 with a summary and items for future work.

2. ILNP

The Identifier-Locator Network Protocol (ILNP) is an Experimental protocol from the IRTF, described in RFCs 6740-6748 [6–14]. ILNP proposes a new naming architecture compared to IP: instead of using IP addresses, ILNP uses node *Identifier (I)* values to explicitly name hosts, and *Locator (L)* values to explicitly name networks. I and L values are bound dynamically, as required. L values are also *dynamically* bound to interfaces, compared to the *static* binding of IP addresses to interfaces.

From an engineering perspective, ILNPv6 [7, 9, 10] is the version of ILNP that is implemented as a superset of IPv6, and this is the focus of our work to date[3].

2.1 Encoding I and L values in ILNPv6

ILNPv6 uses a 64-bit *node identifier (NID)* value unique to the node, which can be generated using the same mechanisms as for an IPv6 interface identifier. However, the *NID* is not bound to a single interface as it is in IPv6. A *NID* is used in end-to-end protocol state, such as in TCP end-point tuples, in place of a full IPv6 address. To allow packets to be routed across the network, ILNPv6 uses a 64-bit *locator value (L64)* that is equivalent to an IPv6 routing prefix, and is learned from IPv6 Router Advertisements (RAs) [7]. The *NID* and *L64* together form an *I-L Vector (I-LV)*, which is inserted into IPv6 packets into the header fields that would normally be used by IPv6 addresses, as shown in Figure 1.

As the *L64* uses the same syntax and semantics as an IPv6 routing prefix, an ILNPv6 packet will be routed across an IPv6 network as if it was an IPv6 packet. The *L64* value does not form part of the end- system,

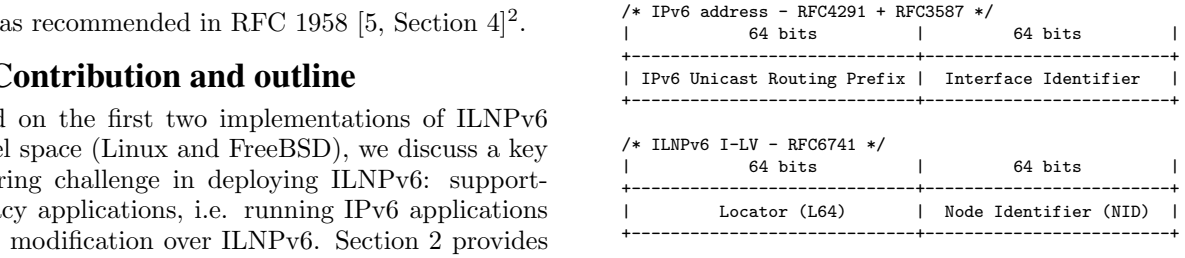


Figure 1: IPv6 unicast address format (top) and the use of I-LVs for the ILNPv6 unicast address format (bottom). The L64 value has the same syntax and semantics as an IPv6 routing prefix. The NID value has the same syntax as an IPv6 Interface Identifier, but has different end-to-end semantics.

upper-level protocol state. So, the *L64* value in an ILNPv6 packet is mutable, without impacting end-to-end integrity. The *NID* value is only used by the end-system, but as its syntax is the same as an IPv6 Interface Identifier, the I-LV can be used by local Neighbour Discovery (ND) as for IPv6.

So, ILNPv6 can be deployed by implementation in end-systems only, with updates required only to those end-systems requiring to use ILNPv6, and should not require code changes to switches or routers: ILNPv6 packets will be seen as IPv6 packets. We have implemented a mechanism to distinguish ILNPv6 flows and IPv6 flows for correct flow demultiplexing at the end-system (with minimal impact to performance compared to IPv6). This would also be required for supporting IPsec with ILNPv6 [6, Section 7] [7, Section 9].

2.2 Transport protocols

The end-system state for transport protocols would normally use full IPv6 addresses, so transport protocol code was modified in order to allow ILNPv6 to be used by upper layer protocols. In summary, the transport layer code was modified to use *NID* values in place of IPv6 addresses. However, this was completed without impacting the use of the API: IP addresses are passed to a C sockets interface from user code, as for IPv6 application code, and so ILNPv6 can support legacy binaries without modification.

2.3 Legacy (IPv6) binaries over ILNPv6

Support of legacy applications is vital for incremental deployment and backwards compatibility. ILNPv6 is implemented as a super-set of IPv6, at the protocol level, so there are already simple mechanisms defined in for an ILNPv6 node communicating to a non-ILNPv6 node to "drop down" to IPv6 [7, Section 10.6].

However, our concern in this paper is the practical issue of allowing existing code using C sockets to function over ILNPv6 without the need to re-engineer or re-compile existing programs, assuming that the communicating end-systems are ILNPv6-enabled. This is potentially a significant challenge, as C sockets requires the IPv6 address value to be fetched by the user code be-

[2]C sockets is still widely used, if not directly by applications programmers, then by lower-level systems, e.g. Java and Python run-time environments are written in C.

[3]It is also possible to realise ILNPv4 [11–13], i.e. ILNP implemented as a superset of IPv4.

fore creating a socket structure. While ILNPv6 already has DNS records defined [8][4], we focus specifically on how to signal across the API to indicate the use of an ILNPv6 session, instead of an IPv6 session.

To show the robustness of ILNPv6 operation, our evaluation (Section 3) used a scenario with a mobile host running a legacy application, the widely-used tool, *iperf*.

3. LEGACY APPLICATIONS

We used an *unmodified* legacy application binary, *iperf*, and compared operation between mobility implemented with both Mobile IPv6 [2] and ILNPv6. This is a challenging scenario, because as well as demonstrating the basic operation of a program running over C sockets, it also tests that the operation of ILNPv6 mobility is transparent, beneath the C sockets API, without impacting the operation of the application.

The testbed used for the evaluation is shown in Figure 2: this was used previously [15], so we keep the details of the testbed brief, but we present here new, unpublished results using TCP.

The testbed consisted of identical rackmount machines, which are labelled CN, R1, R2, R3 and MN in Figure 2. MN used a 802.11ac (ISM 5GHz) wireless LAN (WLAN) link to the testbed, while all other connections were 1Gbps Ethernet. The popular *netem* package was used to emulate additional delay on the R1-R2 and R1-R3 links, as part of the experiment. For Mobile IPv6, R2 was also the Home Agent (HA) for MN, the HA being a proxy/middlebox that is required for the operation of Mobile IPv6. In ILNPv6, mobility is a purely end-to-end function, so no proxy/middlebox is required. *Note that only CN and MN ran ILNPv6 OS kernels – all other hosts ran unmodified IPv6 OS kernels.*

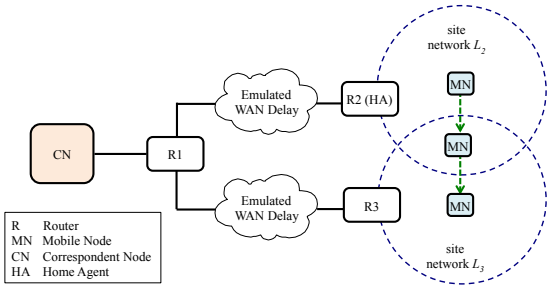


Figure 2: Testbed setup for mobility. The CN connected to R1 via 1Gbps ethernet. The MN initially connected to R2(HA) using WLAN,the dashed / blue circles depict the radio cell scenario emulated. The green / dashed arrows identify movements of the MN to site network L_3, which generated a handoff. CN and MN were ILNPv6-capable Linux hosts. R1, R2 and R3 were unmodified Linux hosts configured as routers. We used Ubuntu 12.04 LTS, with Linux kernel v3.9.0, and only CN and MN ran ILNPv6 modifications.

[4]Implemented in BIND9, KnotDNS, and NSD.

3.1 Signalling via name resolution

The signalling for indicating the use of an ILNPv6 flow exploits the typical usage of the C sockets API. The typical code pattern in a client system makes a call to the POSIX *getaddrinfo()* function with a fully-qualified domain name (FQDN), and then uses the information returned to open a socket. So, we modified the *getaddrinfo()* implementation in *libc* (*glibc*) in order to implicitly signal a name resolution resulting in values that indicate an ILNPv6-capable host as the remote target, as described below.

The FQDN could be resolved by a DNS lookup or by querying the */etc/hosts* file. If the host is ILNPv6-capable, new DNS records for *NID* and *L64* values will be returned, or the lookup in */etc/hosts* will return an I-LV from a */etc/hosts* file that has extended syntax as shown in Figure 3.

```
# /etc/hosts file extended syntax for ILNPv6
#
# L64       64-bit Locator value (in IPv6 address format)
# lprec     the Locator's precedence value (currently not used)
# NID       64-bit Node Identifier value (in canonical EUI64 format)
# hostname  a valid hostname value
#
# An entry -- an I-LV record -- has the structure:
#
#   L64|lprec,NID     hostname

2001:0db8:d00d:0000|10,02-1f-5b-ff-fe-ff-13-74   foo.glob.com
2001:0db8:cafe:0000|20,2a-37-37-ff-fe-1c-cf-fe   bar.blob.com
```

Figure 3: The extended syntax for I-LV information in `/etc/hosts` for ILNPv6.

When an I-LV is resolved, the FQDN and the I-LV is stored in a new 'look-aside cache' in the OS kernel. When the socket is created, if the address information passed in the *struct sockaddr* matches an entry in the look-aside cache, then an ILNPv6 session is initiated, rather than a IPv6 session. The look-aside cache entry would be purged after (i) the DNS entry's Time-To-Live (TTL) value has expired; or (ii) after 1s if the I-LV was read from */etc/hosts*.

3.2 Experiment

In Figure 2, the movement of an ILNPv6-capable Mobile Node (MN) was emulated across the boundary of two different network sites, labelled L_2 and L_3. The MN used two IEEE 802.11ac WLAN links only.

The Mobile IPv6 (MIPv6) model for mobility uses two IP addresses for a Mobile Node (MN). A MN's Home Address (HoA) acts as an identifier: it is returned from a DNS lookup for incoming connections to the MN regardless of where the MN is. When the MN moves from its Home Network (HN) to a Foreign Network (FN), a proxy at the HN, a Home Agent (HA), is informed by the MN of its new address, a Care-of Address (CoA), which it derives based on the network prefix learned from the FN (from IPv6 RAs). The HA then monitors packets at the HN that are using the

MN's HoA as a destination address, and forwards those packets to the CoA using a tunnel.

This end-to-end discontinuity through the HA can cause sub-optimal routing; the HA can also be a performance bottleneck, a single point of failure, and offers an additional attack vector; and there is a tunnelling overhead. So, MIPv6 allows the use of a *Route Optimisation (RO)* procedure: a handshake – a Binding Update (BU) from the MN and a Binding Acknowledgement (BAck) from the Correspondent Node (CN) – allowing the CoA to be signalled during handoff. With RO, TCP protocol state at the CN has to be reconfigured to use the (new) CoA, so the BU also contains TCP sequence number information. However, in our experiments, the MIPv6 implementation uses the 'Home Address Option' header [2, Section 6.3], which carries the HoA of the MN to the CN. The CoA in the IPv6 header is changed to the HoA value before the packet is passed to the transport layer, allowing a TCP session bound to the HoA to remain valid.

ILNPv6 also supports two forms of handoff. With *hard handoff*, a 'break before make' model is used: the MN's connectivity to the 'old' cell, using the 'old' *L64* value, is dropped, and a link in the 'new' cell is established, with the 'new' *L64* value. (As for IPv6, *L64* values come from IPv6 RAs.) With *soft handoff*, ILNP's dynamic binding between *NID* and *L64* values allows a MN to use the 'old' and 'new' *L64* values simultaneously, in analogy to soft handoff in radio systems using two radio channels. In both cases, a Locator Update (LU) / Locator Update Acknowledgement (LUAck) allows the MN to signal the new *L64* value to the CN.

On the testbed of Figure 2, to emulate a WAN link, *netem*[5] was used to generate an extra 100ms of delay between R1 and either R2 or R3 as required, otherwise the link was considered a LAN link. The performance of a 30s TCP flow between CN and MN was measured using the legacy tool *iperf v2.0.5*. This measurement was repeated 10 times (after tuning with some initial trials), for each of MIPv6 without RO, MIPv6 with RO, ILNPv6 hard handoff, ILNP6 soft handoff, with WAN emulation and with no WAN emulation. A single handoff was initiated at t=5s, and completed at t=20s. The handoff was emulated with scripted commands to *ifconfig* to bring WLAN interfaces up and down. The ILNPv6-capable CN was connected using a 1Gbps ethernet link.

4. RESULTS AND CRITIQUE

The key issue here is the continuity of the TCP flows, as handoff events can cause gratuitous packet loss. So, we present two metrics: the total number of lost packets across all the measurements is shown in Figure 4; and the total number of retransmission events across all measurements is shown in Figure 5.

[5]http://man7.org/linux/man-pages/man8/tc-netem.8.html

The *iperf* binary used was identical for MIPv6 and ILNPv6 experiments, and it was oblivious to the control-plane operation (handoff mechanism) for both MIPv6 and ILNPv6. ILNPv6 with soft handoff had the lowest packet loss (close to zero), and the lowest number of retransmission events.

In terms of operation of the *iperf* binary, the unmodified IPv6 binary was able to operate over both IPv6 and ILNPv6 without changes. The operation of the ILNPv6 mobility mechanism did not perturb the operation of the *iperf* and was not visible above the C sockets level.

4.1 Flow performance

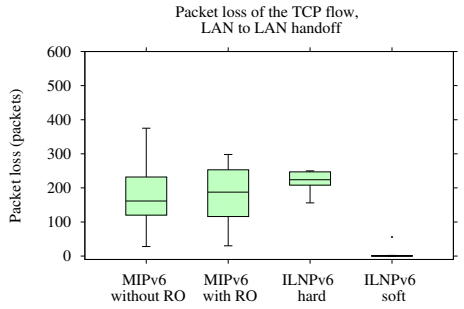
In terms of performance, both loss and retransmission information is needed to gain a better understanding of flow dynamics during handoff. While packet loss could occur during hand-off and trigger TCP retransmissions, as packets will traverse two different paths during handoffs, retransmission could also be caused by delayed packets triggering the TCP retransmission timeout (RTO). Indeed, comparing Figure 4 and Figure 5, we can see clearly that although ILNPv6 with soft handoff has close to zero loss, retransmissions still occur due to the RTOs being triggered as packets are delayed because of the path changes during handoff. However, ILNPv6 with soft-handoff also resulted in the lowest number of retransmissions, as well as the lowest packet loss.

MIPv6 without RO mainly suffers from loss during handoff, as only one path between MN and CN is in use, but may also suffer from delayed packets due to routing via the HA for all data packets. MIPv6 with RO has some loss at handoff while the Binding Update process synchronises the new CoA between the CN, MN and HA. ILNPv6 with hard handoff has similar performance to MIPv6 with RO, but shows a little improvement in loss and retransmissions as, unlike MIPv6, no HA is involved – the signalling is end to end. Finally, ILNPv6 with soft-handoff can use both transmission paths during handoff, so gratuitous loss is eliminated (only the "natural" loss of the actual path is measured, no additional loss due to the handoff process). For MIPv6 with RO and ILNPv6 soft-handoff, there will be RTOs triggered due to packet delays, as consequence of multipath effects, and misordering of transmitted packets.

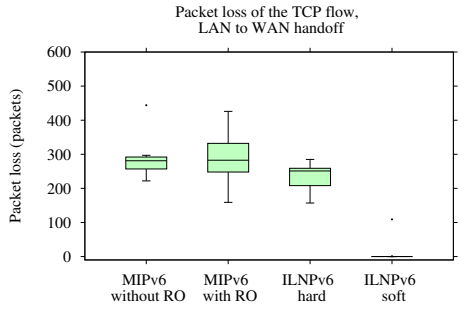
4.2 API usage for legacy applications

In this practical examination, two key assumptions were made: (i) the user code always makes a *getaddrinfo()* call before making a TCP connection, and does not use a previously retrieved result, so that the look-aside cache will have the correct entry: (ii) the user code does not make direct use of the address bits from the *getaddirinfo()* call (and does not use IP address values in configuration state).

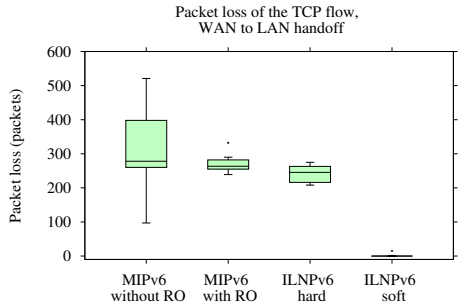
Arguably, the second assumption is a reasonable expectation, if we consider the recommendation of RFC1958 [5] to use FQDNs or application-level names.



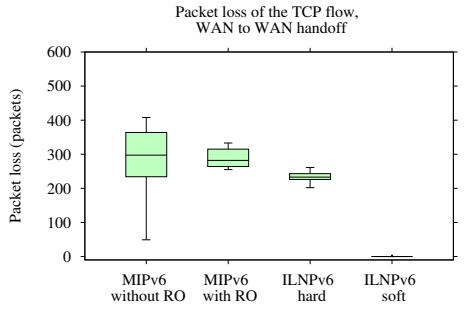
(a) LAN to LAN handoff.



(b) LAN to WAN handoff.

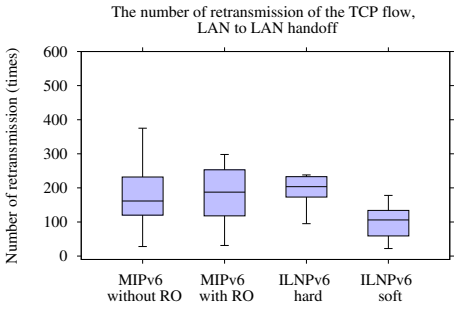


(c) WAN to LAN handoff.

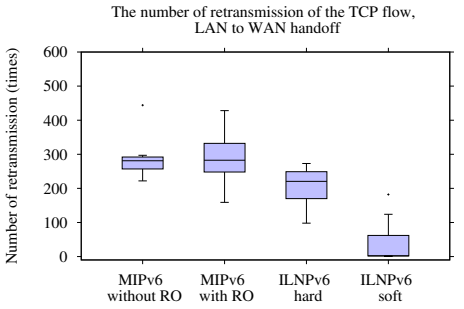


(d) WAN to WAN handoff.

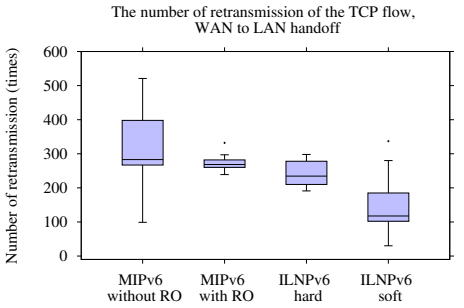
Figure 4: Lost packets during the handoff process. ILNPv6 with soft handoff had the lowest packet loss (close to zero).



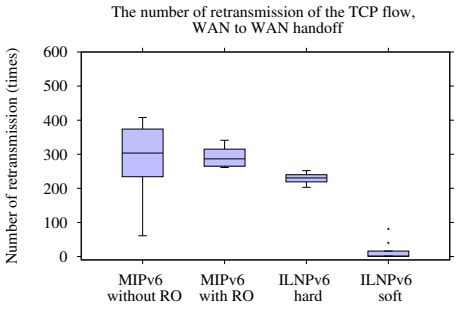
(a) LAN to LAN handoff.



(b) LAN to WAN handoff.



(c) WAN to LAN handoff.



(d) WAN to WAN handoff.

Figure 5: Total retransmission events triggered during handoff. ILNPv6 with soft handoff had the lowest number of retransmission events.

The first assumption may not hold: applications may cache DNS results. However, DNS resource record (RR) time-to-live (TTL) values are typically not returned to user code via the standard sockets API. So, we take the position that a careful programmer should *always* make a *getaddrinfo()* call as described, which is commonly used for the code pattern described in Section 3.1. The implied assumption here, also, is that the *getaddrinfo()* implementation does honour DNS RR TTL values and does not use other caching. This is not stated in the POSIX.1-2008 specification[6], so behaviour could vary across implementations. Also, some Linux distributions might use the Name Service Cache Daemon (*nscd(8)*), which was disabled in our testbed.

However, for ILNPv6, the expectation is that for 'well-behaved' applications [6, Section 2.1] [14, Section 10.5], operation across ILNPv6 will work as if for IPv6, and this can be seen to be true for our experimental configuration.

5. DISCUSSION

We present some discussion on: (i) the wider aims of ILNP, with reference to the rationale of the IRTF Routing Research Group (RRG) work; (ii) briefly consider other Id/Loc solutions that have been implemented; and (iii) make some suggestions as to other practical challenges and functionality that could be achieved with ILNP.

5.1 Future Internet architecture

The recommendations of the RRG in RFC 6155 [4] were based on a charter (initiated in 2007) to recommend a new routing architecture for the Internet. The many Id/Loc proposal submitted came from a long-known (as early as 1977 [16, Section 3]) and recurrent recognition in the Internet community that the overloading of the IP address usage was a cause for serious concern [17–19]. So, a key aim of the proposals was to address the concerns of scalability for routing and addressing in the Internet architecture, and it was one of the initial drivers for early design and development in ILNP [20, 21]. Other Id/Loc proposals are discussed below, but a good discussion of all the proposals considered by the RRG is given in RFC 6115 [4].

5.2 Other Id/Loc approaches

The *Host Identity Protocol (HIP)* (HIPv2 [22]) requires the use of strong encryption as it relies on a public key system for generating host Identifier values that are used by higher layer protocols (such as TCP). IP addresses are still used, but only for routing, i.e. as Locators. However, HIP requires a new API for applications [23], and hence does not work for legacy applications. It may be possible to enable legacy applications via a HIP-Aware Agent [24], but that would introduce another entity that would incur management and maintenance overhead, as well as being a point of failure and potentially offering an attack vector.

The *Level 3 Multihoming Shim Protocol for IPv6 (SHIM6)* [25] is aimed specifically at multihoming and re-homing, but other functions (such as mobility) are possible. SHIM6 requires implementation of an extra 'shim' layer between the network and the transport protocol to perform mapping between Identifier and Locator values. SHIM6 uses IP address values for both Identifiers and Locators, so presents further IP address overloading.

The *Locator Identifier Separation Protocol (LISP)* [26] uses 'map-and-encap', and is a network-based Id/Loc solution. This has the advantage that legacy applications can be supported easily. It uses IP addresses for both Endpoint Identifier (EID) and Routing Locator (RLOC) values, further overloading the use of IP addresses. End-systems use only EID values. However, the LISP approach increases both the per-packet protocol overhead and the complexity of deployed systems, as additional entities are required for supporting the map-and-encap mechanism.

One of the earliest Id/Loc architecture proposals for the Internet was Nimrod [27, 28]. This defined two new datatypes: Endpoint Identifier (EID) values for hosts, and Locators for routing. However, an implementation was never completed to the satisfaction of the Nimrod team.

5.3 Other challenges

You will note that during soft-handoff, the ILNPv6 node is multihomed, albeit temporarily. Indeed, host-multhoming functionality is implemented in our FreeBSD work, which we have described previously [29]. So, ILNPv6 could offer at the IP-level multipath capability for upper-layer protocols in a general way, rather than the protocol-specific solutions that exist in protocols such as MP-TCP [30]. Of course, other mechanisms, e.g. congestion control for MP-TCP, would still be required to be implemented with ILNP. However, ILNP can offer multihomed IP connectivity as first class functionality so does not suffer the same issues, e.g. security issues, that are currently of concern for MP-TCP [31].

Localised addressing, such as the use of network address translation (NAT), is very popular for managing site networks. One of the major arguments against the use of localised addressing is that, in IP, end-to-end integrity of communication is lost, as IP address values, which form part of the end-to-end state, are changed by the network. However, ILNPv6 does not use addresses for end-system state, only *NID* values. As demonstrated in Section 3, even though *L64* values change, the *NID* remains constant during a session, so end-to-end integrity of the session and flows is maintained. In ILNP, localised addressing is a property of the architecture, rather than retro-fitted engineering, as it is in IP. So, localised addressing is easily possible in ILNP using a Locator Re-writing Relay (LRR) function at the site

[6]http://pubs.opengroup.org/onlinepubs/9699919799/

border router (SBR) [14, Section 2].

Mobility and multihoming form a duality in ILNP. The soft handoff mechanism uses (temporary, dynamic) mutihoming when the MN is in the overlap area between cells. There are other possible benefits from ILNPv6 which can be enabled with the combination of multihoming, mobility, and localised addressing with the LRR function at the SBR, including: mobility of whole network sites [32]; wide-area virtual machine image migration [33]; secure fail-over for network resilience [34]; and various site security features [35] including location privacy [6, Section 7]. These could be enabled without loss of end-to-end integrity for flows.

6. CONCLUSION

We have designed and evaluated the first two kernel implementations of ILNPv6, on Linux and FreeBSD. The kernels of both OSs were modified by extending the IPv6 code, implementing ILNPv6 as a superset of IPv6.

Our evaluation of mobility in the Linux kernel shows that for a class of well-behaved applications, especially those that conform to the guidelines on use of names in [5], ILNPv6 can support legacy applications directly, without the need for re-engineering or recompiling programs. This can be achieved with the existing C sockets interface. This requires changes to the *getaddrinfo()* function in *libc* (or *glibc*) and the */etc/hosts* file. However, it means that IPv6 binaries can operate directly over an ILNPv6 stack.

6.1 Future work

Future work will be focussed on completing the level of implementation in both the Linux and FreeBSD kernels, which will include moving the ILNPv6 code into the production kernels for both public OS sources. In the meantime, a tar ball of the existing code will be made available from the authors, on request.

7. ACKNOWLEDGEMENTS

D. Phoomikiattisak was funded by the Thai Government. B. Simpson was funded by Cisco Systems under a University Research Programme (URP) grant award.

8. REFERENCES

[1] B. E. Carpenter, "IP Addresses Considered Harmful," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 65–69, Apr 2014.

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

[3] G. Huston, "Architectural Approaches to Multi-homing for IPv6," IETF, RFC 4177 (I), Sep 2005.

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

[5] B. Carpenter (Ed), "Architectural Principles of the Internet," IAB, RFC 1958 (I), Jun 1996.

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

[7] ——, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations," IRTF, RFC 6741 (E), Nov 2012.

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

[9] 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.

[10] ——, "IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)," IRTF, RFC 6744 (E), Nov 2012.

[11] ——, "ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv4 (ILNPv4)," IRTF, RFC 6745 (E), Nov 2012.

[12] ——, "IPv4 Options for the Identifier-Locator Network Protocol (ILNP)," IRTF, RFC 6746 (E), Nov 2012.

[13] ——, "Address Resolution Protocol (ARP) for the Identifier-Locator Network Protocol for IPv4 (ILNPv4)," IRTF, RFC 6747 (E), Nov 2012.

[14] ——, "Optional Advanced Deployment Scenarios for the Identifier-Locator Network Protocol (ILNP)," IRTF, RFC 6748 (E), Nov 2012.

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

[16] C. J. Bennett, S. W. Edge, and A. J. Hinchley, "Issues in the Interconnection of Datagram Networks," University College London (UCL), IEN 1, Jul 1977, http://www.rfc-editor.org/ien/ien1.pdf.

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

[18] R. Atkinson (Ed), S. Floyd (Ed), "IAB Concerns and Recommendations Regarding Internet Research and Evolution," IAB, RFC 3869 (I), Aug 2004.

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

[20] R. Atkinson and S. N. Bhatti, "An Introduction to the Identifier Locator Network Protocol (ILNP)," in *LCS 2006 - London Communications Symp.*, Sep 2006, http://goo.gl/lI1a6.

[21] R. Atkinson, S. Bhatti, and S. Hailes, "Evolving the Internet Architecture Through Naming,"

IEEE JSAC, vol. 28, no. 8, pp. 1319–1325, Oct 2010.

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

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

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

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

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

[27] N. Chiappa, "IPng Technical Requirements Of the Nimrod Routing and Addressing Architecture," IETF, RFC 1753 (I), Dec 1994.

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

[29] 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, pp. 139–147.

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

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

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

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

[34] S. N. Bhatti, D. Phoomikiatissak, and R. Atkinson, "Fast, Secure Failover for IP," in *Proc. IEEE MILCOM 2014*, Oct 2014.

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