Telecommun Syst (2009) 42: 273–291
DOI 10.1007/s11235-009-9186-5

ILNP: mobility, multi-homing, localised addressing and security through naming

Randall Atkinson · Saleem Bhatti · Stephen Hailes

Published online: 20 October 2009

Abstract Internet users seek solutions for mobility, multi-homing, support for localised address management (i.e. via NATs), and end-to-end security. Existing mobility approaches are not well integrated into the rest of the Internet architecture, instead primarily being separate extensions that at present are not widely deployed. Because the current approaches to these issues were developed separately, such approaches often are not harmonious when used together. Meanwhile, the Internet has a number of namespaces, for example the IP address or the Domain Name. In recent years, some have postulated that the Internet's namespaces are not sufficiently rich and that the current concept of an address is too limiting. One proposal, the concept of separating an address into an Identifier and a separate Locator, has been controversial in the Internet community for years. It has been considered within the IETF and IRTF several times, but always was rejected as unworkable. This paper takes the position that evolving the naming in the Internet by splitting the address into separate Identifier and Locator names can provide an elegant integrated solution to the key issues listed above, without changing the core routing architecture, while offering incremental deployability through backwards compatibility with IPv6.

Keywords Addressing · Mobility · Multi-homing · Security · Identifier · Locator · Internet protocol

R. Atkinson
Extreme Networks, RTP, NC, USA
e-mail: rja@extremenetworks.com

S. Bhatti (✉)
School of Computer Science, University of St Andrews, St Andrews, UK
e-mail: saleem@cs.st-andrews.ac.uk

S. Hailes
Department of Computer Science, University College London (UCL), London, UK
e-mail: s.hailes@cs.ucl.ac.uk

1 Introduction

We choose to take an historical perspective in introducing the problem space to show how usage of IP and functional requirements for IP have evolved. The distinction in naming and addressing for identification and topological location is not new, but is central to our proposal [7, 25, 27].

Mobility extensions to the Internet Protocol have been developed for IPv4 [24] and also for IPv6 [16]. However, neither of these mechanisms are widely deployed or commonly used today. This might be due partly to the complexity of the extensions. It is also due partly to the IP architecture that ties an IP address to an *interface* on a host. This also means that these mobility mechanisms do not interwork easily with other features that users would like to use today in real networks, namely multi-homing, NAT and security. Although engineering solutions have been proposed for all of these, they add to the complexity of Mobile IP as it exists today. This may be another reason that Mobile IP has not been widely deployed.

So, the IP address has two functions in the current architecture: as a node *identifier* providing (locally- or globally-scoped) uniqueness, and as a node *locator*, allowing the routers to forward packets in the correct direction towards the host. Because the IP address has topological significance, a mobile node needs to use another, topologically 'correct' IP address when it moves location.

Similarly, when the Internet was being designed, the concept of a campus or a single host being multi-homed to dif-

ferent networks having different administration was not central to the design. The ARPAnet had a single backbone network. In the late 1980s, there were still a small number of networks and multi-homing was still not yet common. With the advent of the Border Gateway Protocol (BGP), multi-homing became more common in the 1990s. Today, multi-homing is widely desirable, both because of the improved reachability it provides and because of the potentially improved network availability for a site network or a host. The original multi-homing solution developed for BGP remains in use today. This approach requires that each multi-homed network have a more-specific IP prefix advertised by each of its upstream providers. This de-aggregation of routing information has led to rapid growth in the size of the inter-domain (default-free-zone) routing table. While concerns about packet forwarding rates have largely been resolved through ASIC-based IP forwarding engines, concerns remain that the inter-domain routing system might have inherent scaling limits as the size of the routing table increases. One concern is simply the size and growth rate of the routing table. Another is that BGP convergence time might be significantly adversely affected. Network operators would prefer a solution to mobility and multi-homing that did not increase the size of the inter-domain routing table; ideally a solution would reduce both the size and the entropy of the inter-domain routing table.

Network Address Translation (NAT) [10] was widely deployed starting in the late 1990s, partly because of a concern about the perceived availability of IP addresses and partly for unrelated reasons, such as the perceived security advantages of deploying NAT. Unfortunately, NAT generally breaks any upper-layer protocol that embeds the IP address inside the protocol [13]. Because of its perceived security advantages, NAT is unlikely to disappear. In fact, one of the most commonly requested IPv6 features is NAT, even as some vendors and proponents market IPv6 as the way to eliminate NAT. If the IP address had not been misused as an identifier in both transport-layer and application-layer protocols, then NAT would not be a deployment barrier for new applications.

The networking application programming interfaces most commonly used today are based on the BSD UNIX paradigm of *Sockets*. *Sockets* is a relatively low-level interface. Unfortunately, the Internet's Domain Name System (DNS) did not exist at the time that the original *BSD Sockets* interface was implemented and deployed. Hence, the resolution of domain names to IP addresses usually occurs within an application. This has led to the unfortunate and widespread misuse of the IP address, intended for network layer use, as a host identifier. Such misuse creates significant issues for mobile nodes and sometimes also for multi-homed nodes. It also encourages application protocol designers to misuse the address as a host identifier, thereby including network-layer state in application-layer protocols (e.g. File Transfer Protocol uses IP addresses directly on the FTP Control channel, rather than using domain names or some other identifier[1]).

Aside from the networking APIs in common use, common transport protocols (e.g. TCP) also include network-layer state. For example, all the bits of both the source and destination IP address are used for transport protocol state (e.g. in the Transport Control Block), and the TCP pseudo-header checksum. Even the more recent Stream Control Transport Protocol (SCTP) includes knowledge of network-layer state (e.g. a list of valid remote IP addresses for each session). The presence of this state inside the transport protocols increases the complexity of solutions to mobility, localised addressing (NAT), and multi-homing. The current approaches to IP mobility are designed to ensure that the transport-protocols are unaware of the changes in the network location of a mobile node. However, unlike our proposal, this is achieved by using one IP address (e.g. Home Address) for all transport-layer sessions and using a different IP address (e.g. Mobile Address) for routing packets to the mobile node. Our proposal resists this unfortunate semantic overloading by introducing a cleaner structure to the address which includes two components, each with crisp semantics.

We moot that the misuse of the IP address in this way is often considered an acceptable engineering convenience, and so such usage continues. The advantage posed by such a convenient availability of a set of bits has now become a hindrance in the development and deployment of new network capabilities. So, we need to give application programmers a cleaner architectural naming system.

IPv6 and IPv4 share the same naming, addressing, and routing architecture. Moving to IPv6 does not eliminate the issues outlined above or their root causes. Instead, we propose that a different approach to naming and addressing is required if one wants to eliminate those issues and their root causes, and that it is possible to deploy our proposed approach incrementally.

The issues described above are all well known in the research community and our discussion so far does not offer any new insight: simply it summarises and highlights the issues. In Sect. 2, we recount the discussions on Identity and Location naming, and introduce our approach in the form of an architecture that makes a clean distinction between these two functions. Then, in Sect. 3, we consider a specific instance of our architecture. Here, although we describe a fictitious network protocol based on IPv6, our intent is to present sufficient engineering detail to show the viability of our approach. In Sect. 4 we consider selected engineering issues that may arise for the current Internet if deploying the approach of Sect. 3. In Sect. 5, we discuss further technical issues, with a summary in Sect. 6.

[1] FTP was deployed long before the DNS was invented.

2 Identity and location

Some issues in network architecture seem to recur over time. Recent postings to an Internet History mailing list highlight design issues that arose in the 1970s, recurred in the 1990s, and are reappearing even now on the IETF discussion mailing list. In our own case, we have considered the choices in naming and how those affect the overall capabilities of the network. We believe that separating the address into two distinct entities, an *Identifier*, I, used solely for end-to-end identity and a *Locator*, L, used only for routing and forwarding packets, enables significant improvements.

2.1 Historical efforts

Several proposals to separate identity and location have been presented to the Internet engineering community during the past decade. The first of these was Mike O'Dell's proposed "8 + 8" concept in 1996, which specified that the upper 8 bytes of an IPv6 address would be used only for routing and the lower 8 bytes of an IPv6 address would be used only for identification [23].[2] This proposal was very controversial and was not adopted by the IETF's IPv6 Working Group. Some present claimed that the proposal had fatal security flaws. Others claimed that it would not be able to support anonymity. Others felt that it was too late to change the IPv6 specifications.

Partly as a reaction to the rejection of O'Dell's proposal, the IRTF created the Name Space Research Group (NSRG) to study the question of whether the Internet had a sufficiently rich naming architecture. A clear majority of the NSRG believed the architecture was not sufficiently rich and that at least one additional namespace should be added to the architecture. A plurality felt that some form of identifier/locator split was needed, so that the routing and forwarding functions could be separated from the node identification functions. However, the NSRG operated under a rule that required unanimous agreement to recommend an idea to the broader Internet engineering community.[3]

Robert Moscowitz, who was a member of the NSRG, came up with an idea called *Host Identity Payload* that used a modified form of IP Security and created cryptographic identifiers. In this proposal, each node must have a public/private key pair and the node's identity is a hash of its public key. This provides strong cryptographic authentication. However, if the node's public key ever changes, the node loses its identity. Historically, many public keys are eventually lost or may become compromised, forcing a change in public key. So the Host Identity Payload's reliance on an Identifier derived from the public key seems undesirable. In the Host Identity Payload scheme, a compromised public key forces a concurrent loss of Identity. At present, the Host Identity Protocol (HIP) activity has two related groups working on HIP. The IRTF has a HIP Research Group and the IETF has a related HIP Working Group. HIP is being developed as an optional extension to IPv6, but remains controversial within the IETF. However, we are grateful to the HIP effort, as it has helped greatly in the formulation of our ideas for our own proposal, which is described below.

More recently, there has been widespread concern that the IPv6 routing architecture, which is identical to the IPv4 routing architecture, does not handle mobility or multi-homing in a scalable way. Instead, IPv6 suffers from the same routing issues and limitations as IPv4. So the IETF recently created the SHIM6 Working Group to try to address these issues. In essence, SHIM6 overloads the IPv6 address space with some IP addresses being used as locators and other IP addresses being used as identifiers, with each end node performing IPv6 NAT between the locator and the identifier within its IPv6 stack. Unfortunately, one cannot distinguish between an identifier and a locator, and it is easy for the networking protocol software to confuse one with the other. Also, this effectively means that 256 bits are needed for each active network interface, 128 bits for the IPv6 address used as locator and another 128 bits for the IPv6 address used as identifier. At the February 2006 meeting of the North American Network Operators Group (NANOG), it became very clear that many network operators do not consider the SHIM6 approach to be a workable solution to the problems with the IPv6 routing architecture.

2.2 Terminology and definitions

For the purposes of this paper, we choose to use the definitions in Table 1 for our discussion.

Note that here we are using 'name' in the same sense as in [25]. However, we constrain our definitions by restricting our scope to the network level deliberately, in order to

Table 1 Terminology used in this paper

Term	DNS record	Definition
Address	AAAA, A	Name used both for locating and identifying a network entity
Locator	L	Name that locates, topologically, a sub-network
Identifier	I	Name that uniquely identifies a network entity, within the scope of a given locator

[2] O'Dell's draft mentions that the 8 + 8 monicker and a skeletal version of the proposal originally appeared as an e-mail from David Clark. The draft also acknowledges input from several others, including the first author here.

[3] The first author was a member of the IRTF NSRG.

help clarify our discussion for this paper. We recognise that broader and more sophisticated definitions are currently being discussed within the community for the labels 'address', 'identifier' and 'locator'.

Our goal is to confine the routing state within the network-layer, eliminating the current use of topology information (e.g. IPv4 address) by transport-layer and application-layer protocols. We do this by providing a *Locator, L*, at the network-layer. The bits that hold the value of L are not visible above the network layer.

As well as a network-layer locator, we also limit the *Identifier, I*, to a common, non-topological, end-to-end identifier used by transport-layer protocols. I is never used for routing, but considering the end-to-end arguments, we provide visibility of the value of our Identifier, I, at the network-layer, so that a common identifier can be used by all transport-layer protocols. If I instead were provided inside a specific transport protocol, then it would only be available for use by that transport protocol or applications that used that transport protocol. By binding the transport-layer state only to this new end-to-end Identifier, I, instead of binding it to a whole network-layer address, changes in the value of L do not impact any upper-layer protocols.

So, in this new model, a network layer address is, effectively, the concatenation of L and I, which we will denote $L : I$. For the sake of this discussion, we will name a new network protocol using this method of addressing (and the supplementary capability that we will describe) as the *Identifier-Locator Network Protocol* (ILNP).

2.3 Overview of ILNP

In our discussion, ILNP represents an abstract network protocol. We choose to take this approach in order that we can achieve separation between architecture and engineering. Indeed, it would be possible to build an instance of our protocol as a 'clean-slate' design. However, we chose, for pragmatic reasons, to think of an instance of ILNP which is derived from IPv6 and so we refer to this as *ILNPv6* in our discussion. To provide some practical perspective, we summarise the differences and similarities between IPv4 or IPv6 addresses and an ILNP address (i.e. $L : I$). We present a summary of use and properties of Identifiers and Locators below. We will expand upon this in the rest of the paper:

1. An ILNP Locator names a single IP sub-network, not a specific host interface.
2. An ILNP Identifier names a (virtual or physical) node and *is not* tied to a specific host interface or network location.
3. A host may have multiple Identifiers concurrently and may use multiple Identifiers simultaneously. However, any single transport-layer session must maintain the same value of I throughout its lifetime.
4. It is not *required* that an Identifier is globally unique, but it must be unique within the scope of any particular Locator with which it is used. The Identifier need not be cryptographically significant, though we do not preclude the use of cryptographic methods (e.g. hash of a public key) to generate an instance of an Identifier.
5. For any ILNP address, $L : I$, in which I is bound to an active transport-layer session, L can change as required. This use of L will be explained further when we consider how mobility and multi-homing are enabled in ILNP.
6. A host may have several Locators at the same time, for example if it is connected to multiple sub-networks or has multiple interfaces on different subnetworks.
7. The transport-layer state *is not* bound to an ILNP address. Only I is used in the transport-layer state, along with the transport layer port number.
8. The network layer *only* uses L for routing. This is similar to the use of an address prefix for routing in IPv4 and IPv6, and indeed L could be seen as an address prefix, locating an edge network.
9. Packet delivery on the final hop uses the whole of an ILNP address, as in IP. Hence, mechanisms such as ARP (IPv4) or Neighbour Discovery (IPv6) can be adapted for use easily.

We will show that ILNP enables significant improvements in mobility, while multi-homing is achievable using, essentially, the same mechanism. We also show that the use of Network Address Translation (NAT) does not impede the deployment of new services and protocols over ILNP. We also claim that end-to-end security using IP Security (IPsec) can work with mobility, multi-homing, and NATs if ILNP is deployed.

2.4 DNS and a new API

We first apply the traditional computer science concept of data hiding and define a new Networking API that omits the use of addresses or Locators, and instead is focused upon Domain Names.[4] That is, only the Fully Qualified Domain Name (FQDN) is used by normal applications; 'raw' IP address values are no longer visible (though it is clear they could still be used by those applications that absolutely needed to use them). This might seem to place an increased reliance on the DNS. However, the DNS has been in widespread use for two decades; most current Internet users cannot distinguish between a DNS fault and a general network fault. So while this initially might appear to make the network more brittle, we believe that there is little or no decrease in network availability as perceived by a typical user.

[4] At the time that the BSD Sockets networking API was originally defined, the Domain Name System did not exist.

Table 2 Use of names in ILNP and IP

Protocol layer	ILNP	IP
Application	FQDN	FQDN, IP address
Transport	Identifier, I	IP address
Network	Locator, L	IP address
Link	MAC address	MAC address

New DNS record types for Locators and Identifiers will be required and we expand on this later.

Our vision for ILNP is evolutionary, not revolutionary. We believe the proposed enhancements to naming enable significant near-term improvements in mobility, multi-homing, and NAT tolerance. A key feature in ILNP is that the end-system state is not tied to either topological information or to a particular interface. This, coupled with a new Networking API, simplifies creation of network-enabled applications. Use of the new API based on the use of domain names also re-positions applications to adapt more easily to more revolutionary network architectures that might appear in the future. We summarise in Table 2, a comparison of naming between ILNP and IPv4/IPv6, including specifically the use of the values I and L in the protocol stack.

L names a single subnetwork, rather than naming an individual node. Note that ILNP makes *visible* the value of I at the network layer in order for it to be usable by any transport-layer protocol (as explained earlier). At the network-layer, I is only used to identify the end system within the given subnetwork L. With respect to [25], we no longer have globally routable names for interfaces. The implications of this are discussed later.

2.5 Transport layer state

With ILNP, a transport layer name for a communication end-point in a given transport protocol would be given as the tuple of 4 values: the local and remote Identifier, I, and the local and remote port numbers, P, giving the tuple $\langle I_{\text{local}}, P_{\text{local}}, I_{\text{remote}}, P_{\text{remote}} \rangle$. When a transport layer packet is transmitted in an ILNP packet, the packet header also contains values of local and remote Locators, L_{local} and L_{remote}. Whilst it is likely that during the lifetime of a transport layer session, values of L could remain constant, it is not *required* and indeed the transport layer can exploit (i) changes in L throughout the lifetime of the session; and (ii) the use of multiple values of L for the same transport layer session. We expand on this later and show how it is used to enable mobility and multi-homing in an elegant manner.

3 The ILNP approach: ILNPv6

To demonstrate the ILNP approach, we chose to describe an instance of ILNP which we call *ILNPv6*, and we show how it could be introduced incrementally to an IPv6-based network. However, different engineering would allow the same concepts to be applied to IPv4. Our research is focused on the architectural considerations, but applying the architecture to an existing protocol helps ensure that engineering considerations are also identified and resolved.

Note that as well O'Dell's draft proposal from 1996, and the HIP work currently in progress, we are also grateful for the following work on network architectures within the research community (in no particular order): Nimrod [5], TurfNet [26], Layered Naming Architecture [3], 4 + 4 [29], Split Naming/Forwarding Architecture [17], FARA [6], Plutarch [8], i3 [28], IP Next Layer [12], Triad [4]. These works have helped greatly in our thinking to date.

3.1 Address and packet format

For the address format at the network layer, we can, initially, derive from O'Dell's original concepts. O'Dell attempted to explain his 8 + 8 approach as an architectural concept, without complete engineering detail. Our work seeks to provide both an architectural explanation and also provide sufficient engineering detail to help justify the claim that ILNP is practical to implement and deploy. For ILNPv6, as with O'Dell's proposal, the upper 8 bytes of the IPv6 address are used solely as the value of the Locator, L, and name a single subnetwork; while the lower 8 bytes of the IPv6 address are used solely as the Identifier, I, and name a single node. This proposal is an evolutionary next-step from the current Internet. ILNPv6 retains the central concepts of packet networking and provides improvements through the enhanced naming architecture.

For reference, the IPv6 header format is shown in Fig. 1. Our ILNPv6 header format is shown in Fig. 2. Note that the ILNPv6 header is the same size as an IPv6 header. The first 64 bits of the ILNPv6 header have the same syntax and semantics as for the IPv6 header. With ILNPv6, each of the 128-bit IPv6 address fields is, however, split into two 64-bit fields, a Locator and an Identifier. Existing approaches to header compression can be used with this new scheme to conserve capacity on low bandwidth links.

For unicast traffic, the Destination Locator replaces the destination routing prefix used with IPv6 and names a specific ILNPv6 sub-network. For multicast traffic, the Destination Locator specifies the location of a candidate multicast core router or a rendezvous point for that multicast group.[5] In both cases, the Source Locator names a subnetwork associated with the sending node. Anycasting is a subject for future study. ILNPv6 routing relies on longest prefix match,

[5] We are grateful for Mark Handley's help with aspects of this proposal, particularly with multicasting.

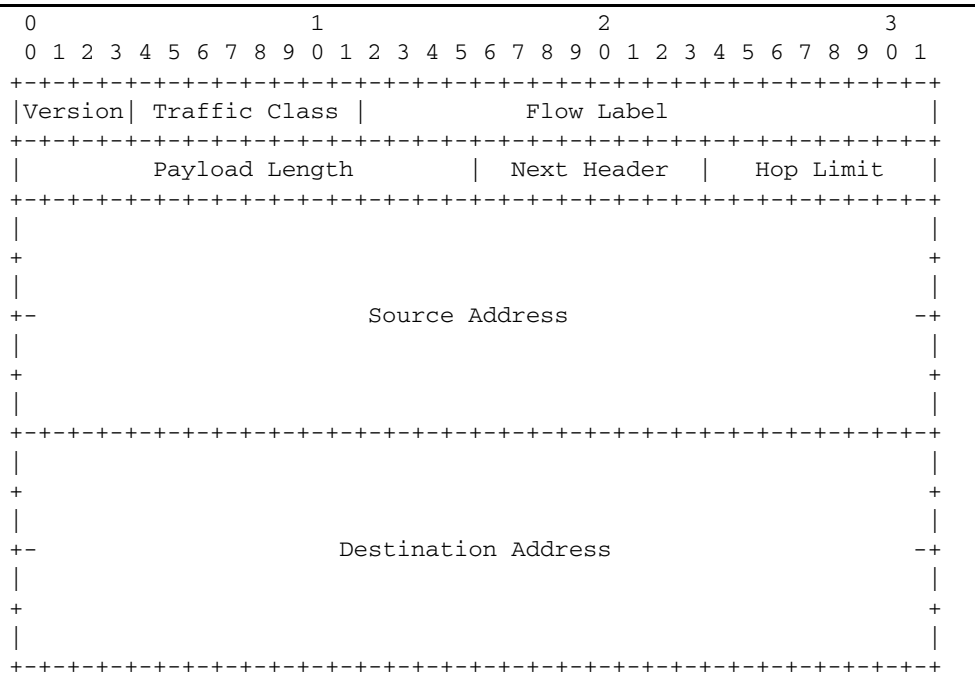

Fig. 1 IPv6 packet header

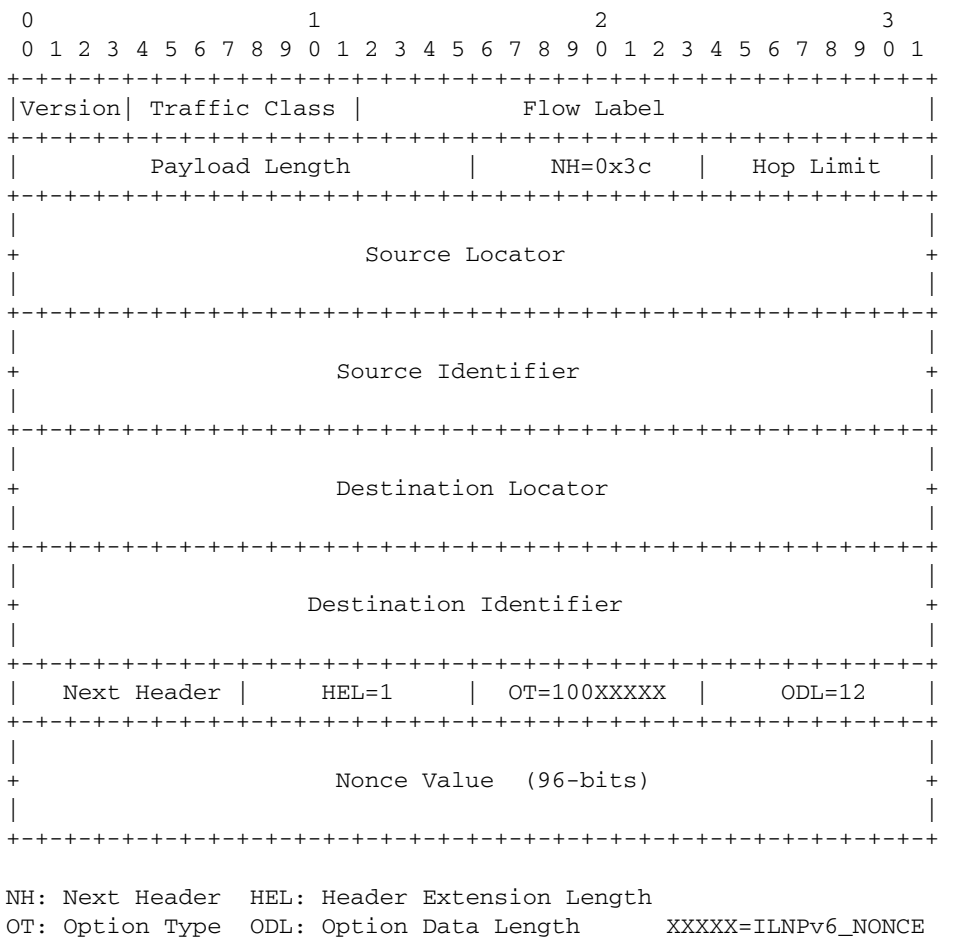

Fig. 2 ILNPv6 packet header with optional Nonce

just as IP does today. The split between Locator and Identifier is fixed, so one does not need a network mask to differentiate the Locator from the Identifier.

In this proposal, Identifiers are not *required* to be globally unique. In practice, we propose a method that will ensure Identifiers have a high probability of being globally unique,

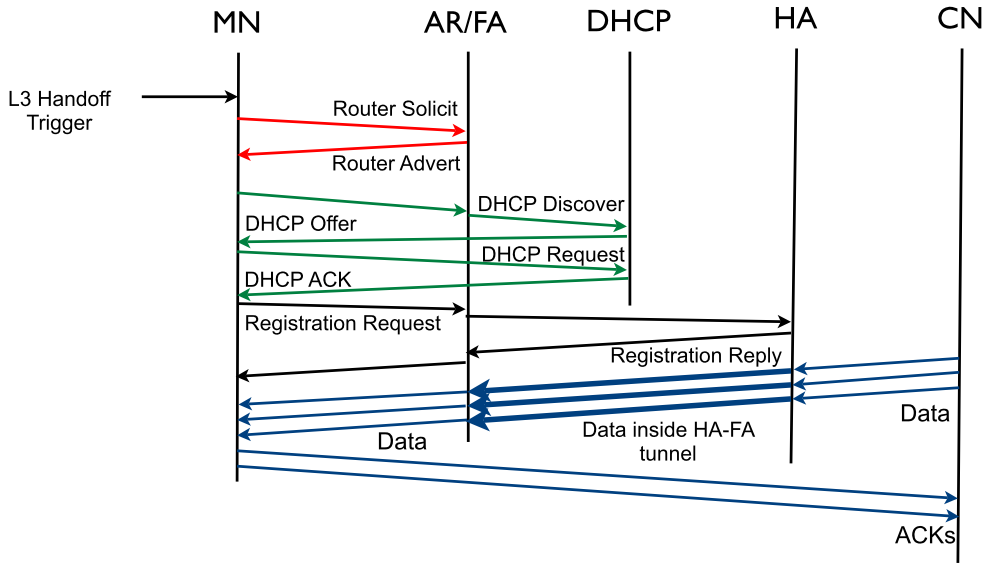

Fig. 3 Mobile IPv4 handoff time-sequence diagram

MN	Mobile Node
AR/FA	Router/Foreign Agent
DHCP	DHCP Server
HA	Home Agent
CN	Correspondent Node

which is more than sufficient. We propose that the format of I in ILNPv6 is the same as that for an IEEE EUI-64 identifier [15]. A host can then simply derive its Identifier(s) from the (set of) IEEE MAC addresses belonging to interfaces on that machine. Note that the use of a MAC address in this way is simply a convenient mechanism for deriving the correct number of bits with a high probability that they will be unique. There is no other significance to the use of a MAC address as a value of I. Unlike the host portion of an IPv6 address, a particular ILNPv6 Identifier value is not tied to any particular network interface. A multi-homed node can use the same Identifier on all interfaces simultaneously, if desired. Further, an anonymous Identifier, or a locally specified Identifier, or a cryptographically verifiable Identifier could can be formed by setting the *local scope bit* defined by IEEE as part of the EUI-64 specification. Finally, each multicast group has its own Identifier; such group Identifiers always have the IEEE EUI-64 *multicast bit set*. Also, by using bits derived from the MAC address in the Identifier, we obviate the need for IPv6 Duplicate Address Detection (DAD).

IPv6 Neighbor Discovery is used unchanged in ILNPv6, so last-hop routers need not change. Also, since the Locator is effectively the IPv6 routing prefix, it is clear that core routers and routing protocols need not change. A mobile node may discover its Locator value much the same way it might discover its IP routing prefix today, through Router Advertisements and Router Solicitations.

3.2 Mobility

In the earliest days of the Internet, a typical computer was too large to be very mobile. So support for mobile nodes was not a native property of the original IPv4 specification. However, portable computers have been around for some time now, so support for mobile nodes is important. The IETF created the *Mobile IP* standard during the 1990s. Later, Mobile IPv6 was developed as an extension to the IPv6 standards. However, neither standard is widely implemented or deployed outside the research community at present.

3.2.1 Mobile IPv4

Mobile IPv4 (see Fig. 3) uses a complex architecture involving at least three cooperating nodes. The *Mobile Node* is the system trying to communicate with other Internet nodes. Each mobile node has a *Home Agent* that provides forwarding of packets addressed to the Mobile Node whenever the Mobile Node is not connected to the Home Agent's subnetwork. The *Foreign Agent* is located on the same subnet as the Mobile Node and provides packet forwarding for the mobile node if the mobile node is not on the Home Agent's subnet.

The basic principle of Mobile IPv4 is that packets from *Correspondent Nodes* always travel to the Mobile Node's conceptual home address, H, located at the IP network that forms the node's Home Network (HN). Then, if the Mobile Node (MN) is not connected directly, a Home Agent (HA) located on that last-hop IP subnetwork will accept the packets addressed to the Mobile Node and forward them to the Mobile Node's current location, at its Care-of-Address (CoA), using IP-in-IP tunnelling of the original packet. However, packets from the Mobile Node to the Correspondent Node (CN) travel directly, using normal routing, (except when the CN is itself mobile, in which case the return packets travel back to the CN via the Home Agent acting for the CN). This packet forwarding path forms a triangle with vertices at the CN, the HA, and the MN. Each Mobile Node requires at least one trustworthy Home Agent to forward traffic on its behalf. The presence of this triangle routing may increase the latency for packet travel from the Correspondent Node to the Mobile Node. Additionally, the path asymmetry may perturb some protocol behaviour at higher layers, e.g. TCP's "ACK clocking" behaviour for rate control.

So, Mobile IPv4 uses "triangle routing" whereby packets from the correspondent travel first to the Home Agent and are then forwarded to the Mobile Node. Packets from the Mobile Node to the correspondent travel directly. As unicast Reverse Path Forwarding (uRPF) checks are now commonly deployed, the Mobile Node might need to tunnel its packets to the correspondent so that they are not mistaken for forgeries [11]. Tunnelling packets increases packet size, which in turn often causes packet fragmentation. The lack of a location-independent identifier makes key management for Mobile IP difficult. This in turn is a deployment impediment for Mobile IP.

3.2.2 Mobile IPv6

Mobile IPv6 (see Fig. 4) has a complex architecture similar to Mobile IPv4. In Mobile IPv6, the IP address is overloaded so that some addresses are used primarily for routing, while others are used primarily for identity (e.g. TCP pseudo-header). However, both kinds of address come from the same namespace. Each mobile node has a *permanent* IP address that is used for identification and is sometimes (i.e. only when actually at "home") used for routing packets. Additionally, each mobile node that is not at "home" has a second *temporary* IP address that is used for routing packets to its remote location. Correspondents normally send packets to the "home address" and an agent forwards them to the mobile node's current location. Replies to the correspondents travel directly, creating the triangle routing situation that also exists with IPv4. Additionally, IPv6 Neighbour Discovery requires that a node perform Duplicate Address Detection (DAD) when first coming up on a network link. DAD can significantly increase the delay when a mobile node changes network location.

Mobile IPv4 and Mobile IPv6 are based on the same underlying concepts, but the implementation details are somewhat different. First, the similarities with Mobile IPv4 will be discussed, and then the differences.

With Mobile IPv6, each Mobile Node (MN) has a permanent IPv6 address, which is called its *Home Address*. This is used as a stable identifier for the Mobile Node. For example, TCP session state is bound to the Mobile Node's Home Address. So, regardless of where the Mobile Node is connected to the network, transport-layer protocols (e.g. TCP, UDP) and application protocols name the node using its Home Address. As with Mobile IP, a Correspondent Node that wishes to communicate with the Mobile Node will send packets to the Mobile Node's Home Address. Then, a *Home Agent* (HA) located on the same IP subnetwork as that Home Address will forward traffic to the Mobile Node at its current location. The Mobile Node's current location is indicated by its *Care of Address*, which is used as a locator. Traffic forwarded between the Home Agent and the Care of Address is sent via an IP-in-IP tunnel. The Home Agent also responds to IPv6 Neighbor Discovery protocol messages, including Duplicate Address Detection (DAD), that are intended for the Mobile Node and are present on the Home Address's subnetwork whenever the MN is absent. Duplicate Address Detection significantly slows the network-layer handoff time, which has caused the IETF to explore ways to optimise DAD [19, 21, 31]. Mobile IPv6 introduces a new Mobility Header which is used to carry various mobility-related control messages between the Mobile Node and the Home Agent. These control messages permit the Mobile Node to inform the Home Agent of any changes to its current location, including when the Mobile Node comes home to its Home Address.

Unlike Mobile IPv4, packets from the Mobile Node are tunnelled back to the Home Agent, decapsulated from the tunnel by the Home Agent, and then are forwarded along to the ultimate destination. This IPv6 tunnelling incurs a fixed 40 byte overhead per packet tunnelled. So the "triangle routing" issue of Mobile IPv4 does not exist in the same form with Mobile IPv6. This difference helps ensure that traffic from the Mobile Node will not be dropped due to ingress IP address filtering [11]. Unfortunately, this tunnelling is computationally expensive, increases latency, and causes packet fragmentation.

In order to eliminate some of this tunnelling and also to generally reduce packet latency, Mobile IPv6 has an optional mechanism to provide *Route Optimisation*. With this mechanism, the Mobile Node informs the Correspondent Node of its actual location within the network by exchanging *binding update* (BU) messages. This optimisation reduces the chance that packets will need to be fragmented, and generally reduces the round-trip time, but the additional overhead

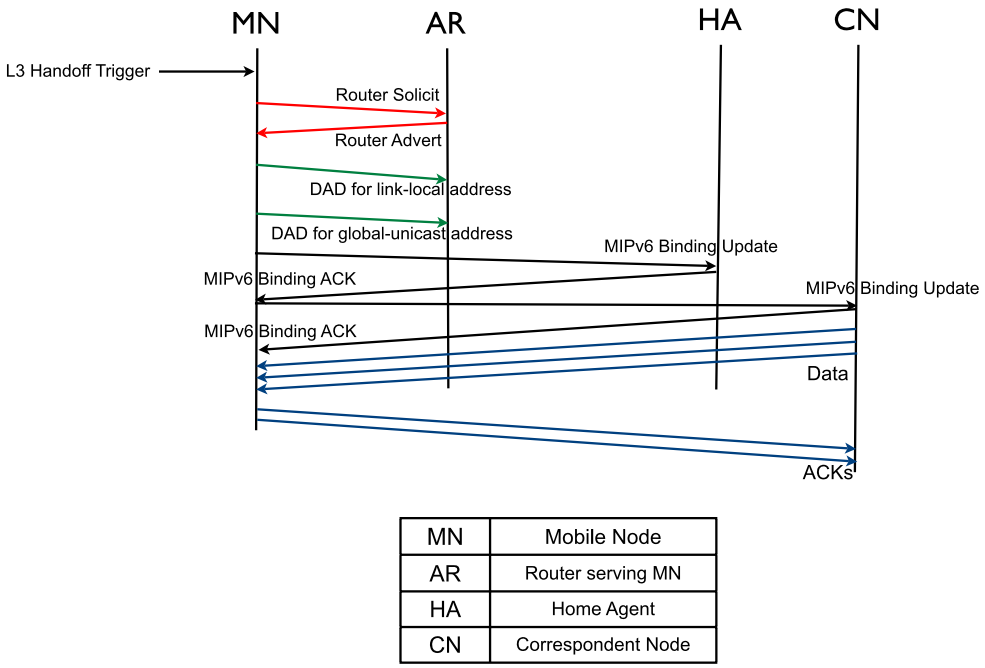

Fig. 4 Mobile IPv6 handoff time-sequence diagram

of the Home Address Option or Routing Header means that some packets will still need to be fragmented prior to transmission and reassembled upon receipt.

Figure 4 shows the packet time-sequence diagram for a network-layer handoff using Mobile IPv6. Since IPv6 includes stateless auto-configuration, DHCP is omitted. After movement is detected, there are 3 round-trips plus Duplicate Address Detection (DAD) delays for 2 different addresses (link-local and global unicast) required before data can flow from the correspondent node to the mobile node.

3.2.3 Optimisations of Mobile IP

With Mobile IPv4 and Mobile IPv6, a wide range of optimisations, for example methods for eliminating triangle routing, have been proposed. The IETF is working to optimise Mobile IPv6 so that DAD is not always needed. Regrettably, this appears likely to make Mobile IPv6 even more complex. At the time of writing, a number of other changes to Mobile IPv4 and Mobile IPv6 are being discussed within the IETF. Space limitation prevents a full discussion of the possible optimisations that could be deployed. These optimisations often add more complexity to the already complex mobility protocols. We believe the current operational need for engineering optimisations is partly indicative of architectural limitations with current mobility approaches.

3.2.4 Host Mobility with ILNP

With ILNPv6, the separation of Locator and Identifier greatly simplifies mobility. The Locator is used only for routing packets from the sender to the recipient's subnetwork, while the Identifier is used in upper-layer protocols (e.g. TCP pseudo-header). Whenever a node moves from one subnetwork to another, the node first securely updates its Locator record in the DNS. This enables new sessions to be established directly to its current location, obviating the Home Agent. Second, as a performance optimisation, the node sends out a newly-defined, authenticated (ICMP) *Locator Update* messages to all current correspondent nodes. The recipients of those Locator Update messages authenticate the message and then update their local Identifier/Locator cache if the authentication succeeds. In this scheme, both nodes in a session can move concurrently. If a node does not respond, for example because some Locator Update messages were lost in transit, then the node's correspondents can make a DNS forward lookup on that node's domain name to learn its current set of Locators. Similarly, the Foreign Agent is obviated because all packets travel directly from sender to receiver. This also eliminates the need to tunnel packets. If the MAC address is used to form the Identifier, Duplicate Address Detection is never required; link layer protocols would fail if two nodes tried to use the same MAC (i.e. link) address.

With ILNP, mobility support is a native property of the network protocol, rather than an add-on protocol. In fact, with ILNP, mobility and multi-homing are supported by a common set of mechanisms. When a mobile node changes its location, its Locator will change. At that point, the mobile node sends ICMP control messages—Locator Update (LU) messages—to all existing correspondents informing of the node's new Locator(s). These LU messages are au-

thenticated to prevent forgery attacks, either using a lightweight non-cryptographic method that prevents off-path attacks or using more comprehensive cryptographic authentication. Additionally, the mobile node updates the set of L records in its DNS entry by using Secure Dynamic DNS Update [30]. If the direct ICMP messages are not delivered to an existing correspondent for any reason, then that correspondent can learn the updated Locator(s) by making a DNS query. New correspondents will discover the current Locator(s) through the DNS as part of the normal session initiation process. So with ILNP there is no routing table impact due to mobility and we eliminate the protocol complexity of the current Mobile IP techniques.

Neither a Home Agent nor a Foreign Agent is needed for ILNP, unlike both Mobile IPv4 and Mobile IPv6, since ILNP nodes support IETF Secure Dynamic DNS Update. Secure Dynamic DNS Update does require a packet exchange, but this packet exchange need not be initiated or completed before the Mobile Node updates its existing correspondents with the Mobile Node's new location using a Locator Update (analogous to the IPv6 binding update). Microsoft Windows XP clients and servers support Secure Dynamic DNS Update, so it is widely available [18]. Further, Secure Dynamic DNS Update is useful in the current Internet and already is deployed in some places. Another potential issue with ILNP is its reliance on DNS Security to authenticate domain-name, Identifier, and Locator mappings. However, ILNP uses the existing DNS Security mechanism designed for the IP Internet. The deployed IP Internet has significant vulnerabilities if DNS Security is not in use, so we need DNS Security for the existing IP Internet, i.e. independent of the question of use within ILNP.

Duplicate Address Detection (DAD) is not required for ILNP because the Identifier is formed from an IEEE identifier already present within the node, unlike Mobile IPv6. The IEEE EUI-64 value is very probably globally unique. However, link-layer communications will fail first should more than one node on the same link try to use the same MAC address. So ILNP does not need to consider that case. The absence of DAD reduces the network-layer handoff latency.

ILNP never requires packet tunneling and always uses optimal routing through the normal routing mechanisms, unlike both Mobile IPv4 and Mobile IPv6. The elimination of tunnelling might significantly improve performance, both due to the optimised routing of packets and the absence of packet fragmentation/reassembly due to tunnelling overhead.

Figure 5 shows the packet time-sequence diagram for a network-layer handoff using ILNPv6. Since ILNPv6 supports stateless auto-configuration, DHCP is omitted. After movement is detected, only 1 RTT and 1 Locator Update are required before data can flow from the correspondent node to the mobile node. ILNP can provide much lower network-layer handoff latency than either version of Mobile IP.

3.3 Network mobility with ILNPv6

With ILNPv6, multi-homing, node mobility and network mobility are essentially handled by the same mechanism: through the change in the value of the Locator, L. When a mobile node moves to another IP sub-network, it will change its value of L, discovering a suitable value locally from Router Advertisements. An ILNP node may hold and use more than one value of L concurrently if it is multi-homed, whether through a single router that happens to be multi-homed, or through multiple routers, each offering a different value for L. With ILNP, a mobile network can be seen as a special case of multi-homing: values of L can be changed as site connectivity changes.

3.4 Multi-homing

There are two kinds of multi-homing, site multi-homing and host multi-homing. Separately, mobile networks appears to be a special case of site-multihoming.

3.4.1 Site multi-homing today

RFC4984 [20] on page 4 states, *The clear, highest-priority takeaway from the workshop is the need to devise a scalable routing and addressing system, one that is scalable in the face of multihoming ....*

Today, site multi-homing is handled by advertising the more specific IP routing prefix for the site via each of the site's upstream service providers. This means that if a site has 3 upstream providers, the global routing table would contain 3 separate advertisements of the site's more specific prefix. If a link between that site and one of its upstream providers goes down (e.g. due to a fibre cut), then the adversely affected upstream provider will withdraw the more specific IP routing prefix advertisement. In turn, this will cause traffic to that site to travel via one of the remaining operational links. Unfortunately, this current approach significantly increases the entropy of routing tables within the default free zone. Concerns about BGP convergence times and routing table size arise from the currently high growth rate in inter- domain routing table entropy. Both IPv4 and IPv6 use this same approach to site multi-homing. Host multi-homing is not well supported by the current Internet architecture.

In response to service provider concerns about routing table entropy due to growth in site multi-homing, the IETF has created the SHIM6 working group. They hope to have an alternative strategy. The current SHIM6 proposal overloads IPv6 addresses so that some IPv6 addresses are used as a Locator to route packets while other IPv6 addresses are

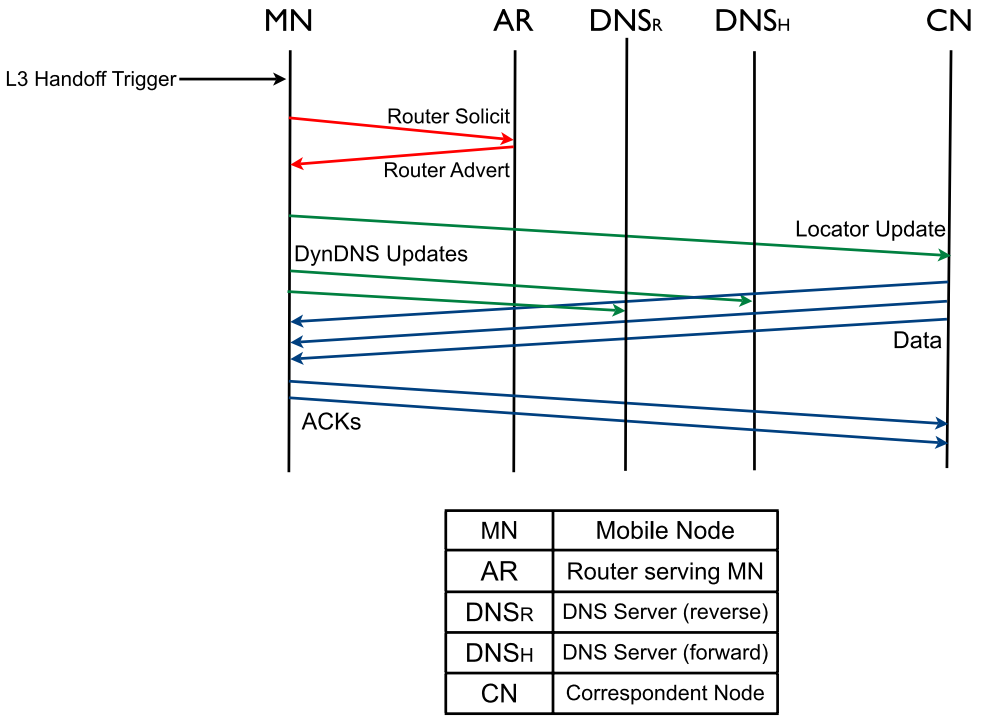

Fig. 5 ILNPv6 handoff time-sequence diagram

MN	Mobile Node
AR	Router serving MN
DNSR	DNS Server (reverse)
DNSH	DNS Server (forward)
CN	Correspondent Node

used as an Identifier in upper-layer session state (e.g. TCP pseudo-header).

Note that in multi-homing today, the multi-homing capability is considered a routing function: the same address is advertised in multiple locations within the network. Clearly, if a topologically significant address is advertised in two different parts of the network, address aggregation cannot be used and so additional routes have to be advertised. This problem is compounded by the increasingly popular practice of using provider independent addresses (PIAs) for commercial reasons. Overall, this exacerbates the scalability problems for routers in the Default Free Zone (DFZ) of the Internet.

3.4.2 Site and host multi-homing with ILNP

From an application (and user) point of view, multi-homing with IPv4 and IPv6 is transparent. However, this comes at an increased cost, and an application or user may gain advantage from having location made visible. Indeed, ILNP takes the approach that multiple Locators, and so multiple locations, can be made visible via the DNS.

With ILNPv6, rather than overloading the IPv6 address with two different semantics, the address is broken into separate Locator and Identifier elements. In the ILNPv6 approach, one need not introduce any more-specific prefixes into the routing table to support multi-homing. Instead, ILNPv6 uses the same mechanisms for multi-homing that it uses for mobility: allowing the use of multiple Locators for individual subnetworks. Further, this approach can support multi-homing for sites, sets of nodes, or individual nodes.

With site-multihoming, there is typically one routing-prefix for each service provider upstream of the site. Each node within the multi-homed site will have at least two Locators, with one Locator for each upstream service provider. These will be present in the DNS *L* records for each node within that site. If a backhoe were to cut a fibre link and thereby make one service provider unreachable, this would be discovered by the site border router, communicated to the other routers within the site, and the edge routers would cease to advertise the routing-prefix associated with the now unreachable service provider. In turn, hosts would learn of this change from the ICMP Router Advertisement messages. Then each host would update its L records in the DNS using Secure Dynamic DNS Update. Each host also would send ICMP Locator Update messages to existing correspondents as a performance optimisation.

Network mobility appears to be a special case of site-multihoming. For example, a ship at sea or airplane in flight might have one or several networks internally and one or more external uplinks. Each node within the mobile network would have at least one prefix for each external uplink. As the network moved, the set of currently valid uplinks would change, just as a fibre cut or installation of a new fibre might change the set of service provider uplinks from a multi-homed site. With ILNP, the same mechanisms used for site-multihoming can be used for network mobility. We believe that MANETs can also leverage the new naming architecture, an item for future work.

With host multi-homing, each multi-homed host has at least two active uplinks, with a distinct Locator for each uplink. If the host is part of a site that has site-multihoming enabled, those two Locators might be accessible via different physical interfaces or on the same physical interface. When both Locators are valid, traffic may use either Locator to reach the multi-homed host. If one Locator ceases to work, perhaps because of a cut fibre link, then the multi-homed host will use Secure Dynamic DNS Update to remove the now invalid Locator from the host's L record set and then will send ICMP Locator Update control messages to each current correspondent. If the ICMP messages are lost, the correspondent will eventually realise the node ceased to be reachable and then will perform a DNS resolution to determine the currently valid L records for that node. If the ICMP messages are received and prove authentic, then the correspondent will discover the change more quickly. These are precisely the same mechanisms that are used by mobile hosts.

With these examples, we can see that by having the right naming architecture, including having crisp semantics for the Locator and for the Identifier, it becomes clear that mobility and (both kinds of) multi-homing are actually the same problem and can be solved using the same set of mechanisms. This is a significant enhancement as compared with the current Internet Architecture.

3.5 Network address translation

Some applications protocols (e.g. FTP) and some lower-layer protocols (e.g. IP Security) do not work well through a NAT. Generally speaking, protocols that do not work well through a NAT are using IP addresses as Identifiers for nodes. No doubt this is due to the absence of non-topological Identifiers in the current Internet Architecture.

With ILNP, the NAT function only changes the value of L. This is invisible to the transport layer and to other end-to-end mechanisms that bind with I rather than with the complete network-layer address $(L : I)$.

While performing NAT on Locators will not break ILNPv6, ILNPv6 does not require that NAT occur anywhere. NAT is not required in routers and NAT is not required in end-systems with ILNPv6. We expect that some sites will want to use NAT with ILNPv6 for one reason or another. For example, ISPs might choose to selectively modify Locators in packets for traffic engineering purposes.

3.6 End-to-end security

For end-to-end security, there is a requirement to bind a security association to some form of identity, at least for some agreed finite duration of the security association. The HIP WG has taken the view that the network-level identity itself should also be cryptographically verifiable [22]. Whilst cryptographically verifiable identity does give extremely strong assurance of the identity, we believe that it is sufficient to have a name with end-to-end significance that can be bound to and that other mechanisms, such as security management protocols, may be used to establish other properties related to that name, including criteria for such an assurance function. We argue that not all upper layer protocols or applications will need this level of assurance, so it may be an unwelcome overhead, or they may wish to use their own, application-specific namespace to achieve this level of assurance. Additionally, differences in security policy at different network sites may also make such low-level identity verification redundant.

The ILNP Identifier provides an end-to-end namespace to which security associations can be bound. Note that ILNP does not preclude the use of a cryptographically verifiable value for the identifier (via use of the *local scope bit* as described above), but we do not require it. As the Identifier has only end-to-end significance, and is not used by the network layer, the security association bound to the Identifier is independent of network location. So, with ILNPv6, it should be possible for IP Security to work easily in conjunction with mobility, multi-homing, and with NATs, by having the IP Security Association bind to the nodes' Identifiers, rather than to their addresses or Locators.

4 Engineering issues for ILNPv6

In this section, some of the engineering that enables the new architecture to be implemented and deployed is outlined. Historical proposals for Identifier/Locator architectures have lacked sufficient engineering detail to persuade many that they were viable approaches. Further, several historic proposals for Identifier/Locator architectures have been rejected by some who believed that it would be either impossible or impractical to use such an architecture without significant negative impacts on security. So we have paid particular attention to security considerations in our architecture and engineering, with further discussion of security in the next section.

4.1 Domain name system

The Domain Name System is enhanced to add 4 new resource records that supplement the *A*, *AAAA* and *PTR* records. The *L* record holds the Locator(s) associated with a domain name. The *I* record holds the Identifier(s) associated with a domain name. The *PTRL* record is used to name the authoritative DNS server for the named subnetwork. The *PTRI* record is used to find the domain name for a given Identifier in the context of a specific subnetwork. One uses the result of the PTRL request to determine where

to send the PTRI request. These records permit DNS to provide scalable reverse lookups for ILNPv6. Having a separate PTRL record facilitates DNS performance and scalability. For example, for a fixed host at a site requiring some level of anonymity, the PTRL record value is unlikely to change frequently, and can be assigned a different caching lifetime than the PTRI record, whose lifetime might be very short as it is generated dynamically as required for a new communication session. For a mobile host, where a host does not have such an anonymity requirement, the PTRL record may change frequently and so have a short caching lifetime, but the PTRI record could have a relatively long caching lifetime.

ILNP adds an additional DNS resource record to enable simple and scalable mobile networks. We introduce the use of a fully qualified domain name (FQDN) to name a mobile network, introducing an extra level of indirection in naming a node. A node that is connected to a mobile network may use a *LP* (Locator Pointer) record in the place of an *L* record. Where a *L* record would provide a 64-bit Locator associated with the node, a *LP* record provides the FQDN of the mobile network that the node is connected to. So if one does an *L* record lookup on a node's domain-name, the *I* records, *L* records (if any), and *LP* records are all returned. The correspondent then performs an *L* record lookup in the FQDN found in the *LP* record to learn the actual numeric Locator value. The *LP* record is a performance optimisation; one could use individual *L* records, at the cost of numerous DNS updates being required when a mobile network moves.

Of course, whilst we have described the *LP* record for use in mobile networks, it is also useful for fixed networks using ILNP. In all cases, it acts to reduce the volume of the data in a DNS server for a site and also to improve manageability of the DNS data.

Security is provided by the existing DNS Security specifications [2]. Dynamic DNS Updates can be provided by the existing Secure Dynamic DNS Update specifications [30]. The DNS can also be used to store public key certificates of single nodes, if desired.

Note that the set of deployed DNS servers does not need to be updated wholesale before ILNPv6 can be used. Only those DNS servers that will provide services for ILNPv6 nodes need to be updated with the new record types. This facilitates incremental deployment of the new network architecture.

4.2 Network layer

In an ILNPv6 implementation, an additional session cache is maintained inside the network-layer code of the end host. This table maintains the current mapping between Locators and Identifiers for each current session. The ILNPv6 network-layer packet is responsible for providing only the Identifier information to the upper layer protocols (e.g. TCP, UDP, SCTP) for received packets. Similarly, for transmitted packets, the network-layer receives only Identifier information from the upper-layer protocols and uses this session state table and the provided destination Identifier to determine the appropriate destination Locator to use for the outbound packet.

Of course, the ILNP implementation also processes ICMP Locator update messages, sending them to current correspondents when its own set of Locators changes, validating, authenticating and then processing them when received from a current correspondent. Further, the ILNPv6 implementation may use the DNS to validate the current set of Locators and Identifiers for a given correspondent (e.g. upon receipt of an ICMP Locator Update message) or to trigger a dynamic update to its own DNS records (e.g. when the node moves network location). Because of the interactions between ILNPv6 and the DNS, implementers might consider moving the DNS resolver and the DNS Dynamic Update function inside the kernel, to avoid kernel to user-space up calls. A kernel-based implementation of such functions may indeed provide other performance and security benefits.

4.3 Transport layer

Transport-layer protocols are modified very slightly. At present, the entire IP address is included in the transport-layer session state (e.g. TCP pseudo-header calculation). This creates difficulties for the current approaches to Mobile IP, because changes in the network-layer location adversely impact upper-layer protocols. For example, if a node changes its network-layer location, it will use a new IP address; this IP address change will break the existing transport session, absent some mechanism to update the remote node's transport session state with the new IP address. Similarly, this can create problems for multi-homed nodes. If a server initially has two IP addresses and later a fault severs connectivity to one of them, sessions associated with the faulty network interface cannot easily migrate to the remaining operational network interface.

With ILNP, only node Identifiers are included in transport-layer pseudo-header calculations, Protocol Control Blocks (PCBs), or other transport-layer state. Locators are omitted from upper-layer protocols. This enables TCP, for example, to maintain a session even if one or both communicating nodes change network location during a TCP session. This capability appears to obviate the need for SCTP's multiple endpoint support, though it does not interfere with that existing SCTP mechanism. The Datagram Congestion Control Protocol (DCCP) would simply use *I* values also. The use of transport protocol port numbers is unchanged.

Checksum algorithms will also need to be modified to use only the Identifier in the pseudo-header in place of the full IP address.

4.4 End-to-end security

The existing IP Security mechanisms, ESP and AH, continue to work with ILNPv6. However, ESP and AH now bind their Security Associations to each node's Identifier values, I, instead of to each node's IP addresses. This change permits ESP and AH to work well even if a Locator changes during a cryptographically protected session. For example, this means that ESP and AH will now work natively through a Network Address Translation (NAT) device or similar middlebox, without requiring the complex protocol mechanisms for NAT traversal currently required [1, 14].

Separately, a lightweight nonce may be used for authentication when the threat environment does not require cryptographic authentication. This nonce must not be predictable [9]. This network-layer nonce would be carried as a Destination Option in ILNPv6. The option only protects against off-path attacks, but enables deployments in low threat environments to avoid using IP Security on all packets. The nonce option provides security equivalent to what ordinary IP provides when IP Security is not in use for a session. The option of a lightweight security mechanism is a significant difference from HIP, which requires computationally expensive cryptographic authentication in all cases.

For the special case of an ICMP Locator Update, DNS Security also may be used to cryptographically validate the information received. So the potential security issues that previously made some uncomfortable with a split Identifier/Locator architecture have been resolved in our ILNPv6 proposal.

4.5 Re-use of existing IPv6 mechanisms

Where possible, ILNP reuses existing IPv6 mechanisms. Specifically, we can reuse most of IPv6 Neighbour Discovery, although omitting Duplicate Address Detection (DAD), which is no longer required when MAC addresses are used to derive Identifier values. The existing IPv6 router discovery, routing protocols, and router packet forwarding procedures can be reused without change.

This technology reuse means that it should be possible to deploy ILNPv6 over existing IPv6 backbone networks without having to change the backbone network itself. Minor changes would be desirable in the edge routers. For those hosts using ILNPv6, networking software in end-systems would need modification to add the ILNPv6 enhancements. It appears possible to implement ILNPv6 concurrently with IPv6 in a given host. This technology re-use facilitates both incremental deployment of ILNPv6 and experimentation. With incremental deployment, the first step is to upgrade the networking software in selected nodes, to upgrade their authoritative DNS servers to support the additional ILNP record types, and to configure the new DNS records for the upgraded nodes.

4.6 Mobile and network realms

There is growing interest in using IP for providing communication mechanisms between non-IP edge networks. Whilst tunnelling is always possible, and may be the most appropriate and desirable mechanism in some applications, a more lightweight and general approach for communication between different network realms may be desirable, especially for some mobile applications.

We note from our discussion above that while the Locator has a well-defined semantic for the IP network layer, the Identifier is opaque and its value is not important. Similarly, the Transport layer state is not tied to the Locator, only to the Identifier and does not make use of the Locator value. So, it should be possible, in principle, to run TCP and UDP across non-IP protocols at the edges of the network and still allow end-to-end communication across an IP core, given a suitable network layer gateway at the boundary between the IP and non-IP network realms.

For example, there is much interest in use of wireless and mobile sensor/actuator networks and such edge networks that may not use IP. However, an Identifier value could be used in sensor/actuator devices, especially one formed in the IEEE EUI-64 syntax, in order to allow state to be maintained across network realms.

Also, MANETs may benefit for inter-MANET communication, or communication with non-MANET nodes through the use of a naming approach based on ILNP.

4.7 Incremental deployment

We believe that ILNPv6 can be incrementally deployed. As we have explained above, as the most-significant 64-bits of the ILNP address, the Locator, coincide with the IPv6 routing prefix, the core routers and routing protocols do not have to change.

Of course, end-system networking software will need to change. The network layer operation will be modified to recognise the $I{:}L$ split in the IP address, and also to keep state for current $I{:}L$ bindings. Neighbor Discovery should not have to change, however. It should be possible to have mixed concurrent operation of ILNPv6 and IPv6, on a per-session basis, with ILNPv6-enabled nodes.

We believe that a version of ILNP based on IPv4 is also possible. Engineering would not be as elegant as for ILNPv6, using the current IPv4 address as the value for L, and requiring an IP option header to carry the I value. However, given the discussion above on Network Realms, at this point, we believe protocols above ILNP should be able to interoperate through gateways.

The most disruptive change to existing infrastructure is likely to be the increased reliance on DNS. Although we believe that for small scale deployment and testing, a modified hosts file could be used, for operational deployment, the

DNS has to be upgraded. Even then, no wholesale upgrade to all DNS servers and libraries is required: only those serving ILNP nodes need the new records, (*I*, *L*, *PTRI*, *PTRL*, *LP*), and then only those serving mobile nodes need the DynDNS and DNSsec support.

4.8 Applications programming

The main complaint about Network Address Translation (NAT) is that when NAT is deployed, then some networked applications cease working. If the applications were designed and coded for more abstract networking APIs, then the applications would not include any network-layer state, and would therefore continue working properly even in the presence of NAT. Additionally, the lack of higher-level name spaces that are not bound to network-level names hinders other functions such as mobility and multi-homing.

So, we also propose a new networking API for C/C++ programming. This new API has more appropriate abstractions than the current BSD Sockets API. We believe that networked applications ought to be able to use only domain names and service names to open new sessions. For example, the new API does not require the application software to perform domain name to IP address translation (e.g. *gethostbyname*()). Instead, the new API accepts domain names as the end-point names for the session, handling the details of domain name to Identifier/Locator translation internally. Further, the new API uses service names directly, eliminating the need for application protocols to have hard-coded protocols and port numbers or to perform service name to port translation (e.g. *getservicebyname*()) within the application. Because the new networking API uses data hiding and more appropriate abstractions, the same API should work equally well whether the underlying networking stack is based on IPv4, IPv6, or ILNP. In fact, a thoughtful implementation of the API would determine which network-layer protocol to use based on the DNS records that exist for the remote end of the session and the local networking capabilities. Initial prototyping of this API might be undertaken in the form of a user-space library, but ultimately it would be best to implement this inside the kernel.

We hope that such a new, simpler, more abstract, networking API also will make it easier for application authors to develop networked applications. By using this new API, we eliminate some of the causes for the misuse of the IP address as an Identifier. Finally, we hope that applications which use this new API will be able to transition more easily to any revolutionary network architectures that might follow. We note that Java already includes both a more abstract networking API, *URLConnection*, in addition to a traditional *Socket* API. We believe that the availability of the simpler Java networking API has been one contributor to the ease of writing new distributed applications in Java.

Of course, a few specialised applications (e.g. management applications such as *traceroute* and *ping*) might require the direct use of *L* and *I* values. Hence, we do not require that all applications use the new Networking API. We expect that newly written applications normally would use the new API, because it is easier and faster to use.

We are careful to note here that the objects that such an API identifies remain *communication end-points*. Future APIs may also consider naming of objects that represent entities that are more specific to certain application domains, and this is beyond our scope.

5 Discussion

We now present some points of critical discussion for ILNP. We concentrate on practical issues that are of current interest within the research community. In particular, we are concerned with the use of ILNP across existing network infrastructure.

5.1 No interface name

As a direct consequence of a Locator naming a sub-network and an Identifier providing a location-independent name for a (logical, virtual, or physical) node, interfaces no longer have globally routable names. This might affect specialised applications that rely on the use of names for interfaces, for example network management applications. There are two issues. Firstly, a suitable namespace might be desirable for naming interfaces. Secondly, those applications that need to use interface names must be re-written in order to use this new namespace. The use of Locator/Identifier naming might force an application to adopt an application-specific namespace. The topic of application-specific namespaces is beyond the scope of this paper.

5.2 Retro-fitting IPv4: ILNPv4

ILNP could also be implemented as a set of modifications for IPv4, giving ILNPv4. Here, the IPv4 addresses would become the *Locators* and separate *Identifiers* would be carried in a new IP option. As with the previously described IPv6-centric approach, all of the transport-layer state would be bound to the Identifiers and a new session mapping table would be added to the IP layer in host implementations. Similarly, one could optionally carry a nonce in an IPv4 option to provide light-weight protection against off-path attacks. Mobility and multi-homing would work as described previously and would bring the same benefits. This would provide many of the same architectural benefits as the IPv6-oriented approach in ILNPv6. A possible realisation of a ILNPv4 header is shown in Fig. 6.

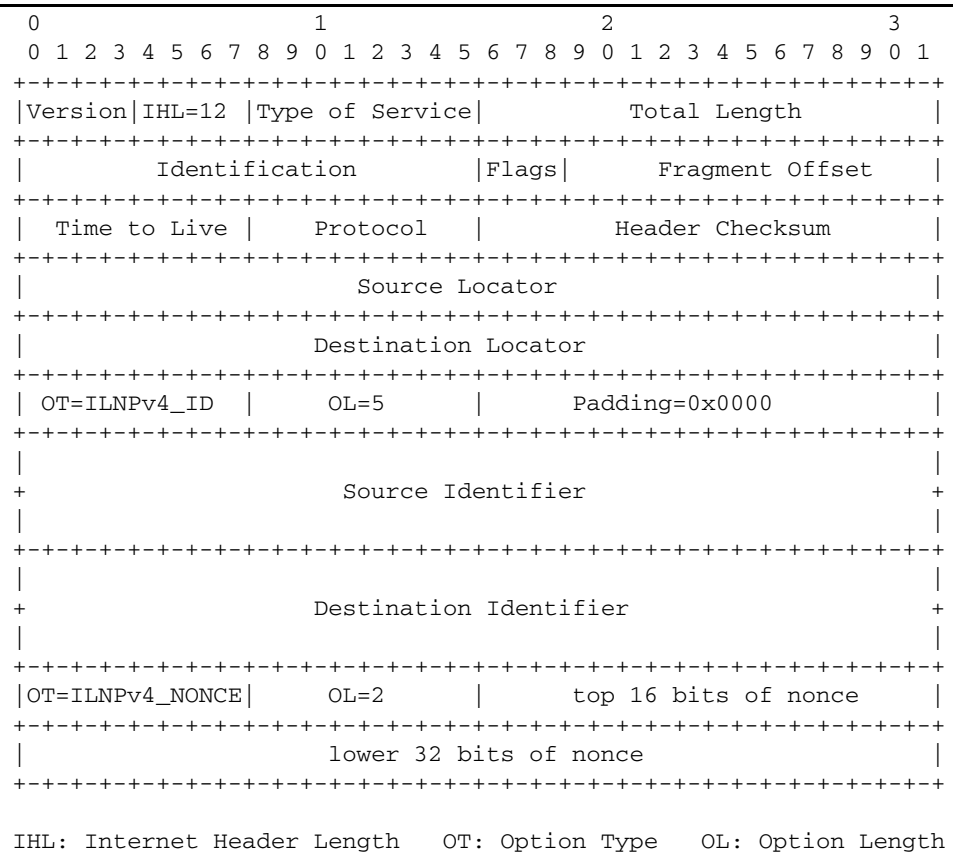

Fig. 6 ILNPv4 packet header with optional Nonce

If the IPv4 address field in the IPv4 header were reused for ILNPv4, the current IPv4 address prefix would be used as the Locator, and the host part of the IPv4 address would be ignored. Again, this would mean that there is virtually no impact on routing ILNPv4 packets through an existing IPv4 core.

For ILNPv4, ARP would need to be modified to use the combination of the ILNPv4 Locator and the Identifier. So, the edge router at the final hop, as well as dual-stack IPv4/ILNPv4 hosts in the subnetwork, would need to know when to send a normal IPv4 ARP and when to send a modified ILNPv4 ARP. If the full 32-bits of the IPv4 address were used for ILNPv4 Locator values, then the lifetime of the IPv4 address space potentially could be prolonged. Of course, there is the possibility of confusion here, with ambiguities between a 32-bit value being a ILNPv4 locator or a normal IPv4 address for a node interface.

Practical considerations (e.g. limited IPv4 option space, routers that forward IPv4 packets with options via the slow-path) reduce the value proposition of ILNPv4, compared with ILNPv6. However, we feel that a proof of concept implementation of ILNPv4 should be possible with approximately the same effort as ILNPv6.

5.3 Generating identifier values

For ILNPv6, we have proposed above, a simple and pragmatic approach to the generation of values of the Identifier, I. We make mandatory the use of the IEEE EUI-64 syntax. Normally, an internal IEEE MAC address is used to form an Identifier in EUI-64 format. Since IEEE provides a large number space, this approach yields an Identifier with a very high probability of being unique, at least within the scope of a given Locator. This could easily be used in bootstrapping systems and in auto-configuration protocols, including ZeroConf.[6] The Locator for ILNPv6 is equivalent to an IPv6 address prefix. Hence it can be discovered easily using existing mechanisms (e.g. IP Router Discovery).

For most nodes, for example a desktop workstation with a single interface, an ILNPv6 address is likely to have fixed values of L and I. So, the basic, most common use case for ILNPv6 is very simple. Further, for the normal case where the EUI-64 value is formed from an IEEE MAC address, link layer communications will fail if more than one node tries to use the same MAC address on a given link.

However, by setting the local *scope bit* in the Identifier, and assuming another bit is used to indicate a multicast Identifier, any other value could be used for the remaining 62 bits

[6] http://www.zeroconf.org/.

of the ILNPv6 Identifier. For example, the Identifier values might be derived from a public key, e.g. 62-bits taken from the hash of a public key, as in the HIP architecture. Indeed, conceptually any local policy could be used for generating and allocating values for I. However, if the I value is not the default EUI-64 value, then Duplicate Address Detection (DAD) may be needed (depending on the algorithm used) to protect against Identifier collisions within the link. Also, the authoritative DNS server for a given link can only hold reverse information for one user of that Identifier on that link, so DNS necessarily will discover any attempts by more than one node to use the same Identifier value on a given link even if DAD were not in use.

5.4 Security issues

Potentially, there are new security concerns introduced by ILNPv6. Although the role of DNS is already a key factor in Internet operation, ILNPv6 relies on DNSSec and DynDNS being present in order to support mobility. These DNS functions have yet to be widely deployed. However, they are only needed for those hosts that wish to use mobility as proposed by ILNPv6. Additionally, mobility support requires a new ICMP message, Locator Update. This is synonymous to the Binding Update of mobile IPv6, and the security issues are similar: the message needs to be authenticated to prevent possible disruption due to malicious intent.

Should a DNS server be compromised, or DNSSec be subverted, the main risk is a DoS attack where a bad host, W, illegitimately claims an Identifier, I_V, that belongs really to victim V. If host W falsely claimed identifier I_V by putting that value into its own DNS I entries and then W initiated a long-lived session with a node X, V would not be able to communicate with legitimate host X (that legitimately uses identifier I_X) for the lifetime of W's session with X (plus some short cache timeout period). This attack can be prevented by including the FQDN of the remote node for each session inside the ID/Locator cache of the stack.[7] Further, if W and V are present on the same subnetwork, this conflict can be detected by any node on that subnetwork, including the first-hop router.

Overall, if a DNS server, or DNSSec/DynDNS is subverted, there is greater potential for disruption to a number of mobile nodes, specifically the potential for DoS and possibly man-in-the-middle attacks. However, if a DNS server is subverted within the currently deployed Internet, there are a wide range of (largely equivalent) security issues.

In other respects, at the network layer, our current thinking is that ILNPv6 will be at least as secure as (i.e. no less secure than) IPv4 or IPv6.

[7] It also helps to have thoughtful validation within the ILNP portion of the network layer implementation.

5.5 Network realms

Considering the increasing heterogeneity of networks, especially edge networks (such as sensor networks) it is becoming increasingly common to consider networks with non-IP (or perhaps non-standard use of IP) interconnecting across an IP network. In such cases, various mechanisms could be used for enabling end-to-end connectivity. Many mechanisms may not be transparent, or they may be transparent but require middleboxes, proxies or application level gateways that need application specific knowledge and maintain mappings of session state. We may think of these edge networks as being separate *network realms*, each perhaps with its own addressing, routing, and naming.

A well-known example of edge networks that break the end-to-end state are networks that are accessed through NATs. These have been discussed earlier and we have proposed how ILNP can deal gracefully with NATs, whilst still maintaining exact end-to-end state for a session. We moot that the use of ILNP can ease the interworking between network realms, even when IP is not the carrier in each network. In such a case, typically some sort of middlebox, proxy or application-level gateway will be required to map session state as well as perform a protocol translation if needed. ILNP has the advantage that the Identifier can be used as a network independent identifier, allowing easier mappings of session state and identification of end-systems across network realms. For example, if the default EUI-64 flavour of Identifier is used, this represents a (highly probably) globally unique identifier. So, session state maintenance that requires protocol mappings would have some shared state through a name that is common across the network realms (and likely to be unique globally).

Of course, other more complex namespace translations or resolutions may be required across network realms, and such a discussion is beyond the scope of this paper.

5.6 Development and deployment of ILNPv6

At present, we are examining what modifications to a BSD kernel would be necessary to implement ILNPv6. Of course, our intention is to make the implementation available after we feel it is sufficiently mature.

Naturally, ILNPv6 would first be trialled on research networks. This would allow us to discover any unforeseen issues that might occur within the backbone, as well as allowing us to look at how existing applications would behave. Any applications that use the IP address within the application are likely to break if a node changes its location: FTP comes immediately to mind, as well as those WWW services that use cookies based on IP addresses.

Our test infrastructure will be a combination of lab test-beds and the UK's Joint Academic Network (SuperJANET5). After initial testing on lab test-beds, our aim is

to be able to route traffic across the production IPv6 UK backbone, without having any modifications to that backbone. If this is successful, it will show that ILNPv6 packets can be carried transparently across a IPv6 core network. The next stage will be to look at the transport protocol code and porting of applications, which naturally requires the development of the API we discussed earlier. We expect the most disruptive and delicate activity to be DNS upgrades. In the initial stages, we are likely to run completely separate servers for ILNPv6 capable DNS service. If this is successful, we will then look into integrating ILNPv6 DNS upgrades into a normal production DNS server.

6 Summary and future work

While Mobile IP has been a worthwhile effort, it represents a design compromise where naming practices are unchanged and mobility is an optional extension that has not been widely implemented. Both Mobile IPv4 and Mobile IPv6 have been difficult to deploy and are not well integrated with separate solutions to various other problems, such as multi-homing, network-address translation, and end-to-end IP-layer security. We have taken a different approach by proposing and evolution of the naming in the Internet Architecture to address all of these issues in an integrated manner.

> RFC4984 on page 7 states, ... *workshop participants concluded that the so-called "locator/identifier overload" of the IP address semantics is one of the causes of the routing scalability problem as we see today. Thus, a "split" seems necessary to scale the routing system, although how to actually architect and implement such a split was not explored in detail.*

Our proposed naming architecture is presented within an abstract protocol, the Identifier Locator Network Protocol (ILNP). ILNP enables fully integrated support for those several functions, so that deploying combinations of those capabilities is easier than at present. We presented an instance of this architecture, a new network-layer protocol derived from IPv6, which we name ILNPv6.

A feature of the new protocol is that it does not require significant changes to already deployed IPv6 backbone routers. So, one can use existing IPv6-enabled research networks for initial testing. Further, ILNPv6 is backwards compatible with IPv6 and can be deployed incrementally, thereby avoiding the need for a flag-day transition. Our proposal is evolutionary, but the new networking API we propose should help enable more revolutionary networking approaches in the future.

We recognise there is still work to be done on this proposal, particularly in the areas of operational scalability, implementation considerations, and performance optimisation. We believe that experimentation with a prototype will help in all of those areas. To demonstrate the efficacy of this proposal, we plan to undertake a proof of concept implementation as one of our next steps. We anticipate testing viability of that initial demonstration implementation using the UK's Joint Academic Network (JANET) between St Andrews, Scotland, and London, England.

Note that multiple L, I and LP records may be visible to DNS users. In order to provide hints on usage on multiple records of the same type, DNS entries could also include preference values as hints to the user and application. However, the visibility of multiple locations to the application, the use of dynamic naming, coupled with appropriately defined application-specific name spaces opens up excellent opportunity for investigating various functionality which we have yet to explore fully, including:

- Soft-handoffs in mobile networks through the use of multiple locators while in transit between two networks.
- Enabling ubiquitous communication capability for mobile devices, including vertical hand-off.
- Transferring communication sessions between end-system devices.
- Transport protocols and applications flows that can use multi-path, multicast, and anycast capability.
- Traffic engineering control enabled through naming and the use of preference values for path selection.
- Supporting communication with edge networks of non-IP devices, e.g. sensor networks.
- Efficient data-oriented routing in peer-to-peer networks and sensor networks.

In each of the cases listed above, the use of an appropriate application-specific naming scheme may enable powerful, flexible and easily controlled functionality through the definition, advertisement, discovery and manipulation of names. Such a heavy reliance on the use of names and name resolution may worry some in the community who feel that it should be possible to operate the network without complete reliance on external services such as DNS. Indeed, with ILNP, knowledge of the correct $I : L$ combination is all that is required, much as knowledge of the correct IPv4 or IPv6 address is required today. Whilst this requirement may still be considered vital in some application areas (perhaps military, for example), for most users of the Internet, reliance on naming and DNS is already a reality: a typical user cannot tell the difference between (and for practical purposes treats as identical) the situations of "network down" and "DNS server unavailable."

Although in this paper, we have discussed how mobility is unified with other network functions, it is clear that the appropriate use of naming and naming services could have far-reaching impact for the future Internet architecture.

References

1. Aboba, B., & Dixon, W. J. (2004). *IPsec-Network address translation (NAT) compatibility requirements*. RFC 3715, IETF.
2. Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005). *DNS security introduction and requirements*. RFC 4033, IETF.
3. Balakrishnan, H., Lakshminarayanan, K., Ratnasamy, S., Shenker, S., Stoica, I., & Walfish, M. (2004). A layered naming architecture for the Internet. *ACM Computer Communications Review*, *34*(4), 343–352.
4. Cheriton, D., & Gritter, M. (2000). *TRIAD: A new next generation Internet architecture*. Technical report, Stanford University, Stanford, CA, USA.
5. Chiappa, N. (1994). *IPng technical requirements of the nimrod routing and addressing architecture*. RFC 1753, IETF.
6. Clark, D., Braden, R., Falk, A., & Pingali, V. (2003). FARA: Reorganizing the addressing architecture. *ACM Computer Communications Review*, *33*(4), 313–321.
7. Cohen, D. (1978). *On names, addresses, and routings. Internet experiment note (IEN)* 23, ARPA Network Working Group.
8. Crowcroft, J., Hand, S., Mortier, R., Roscoe, T., & Warfield, A. (2003). Plutarch: An argument for network pluralism. In *FDNA 2003: Proceedings of the ACM SIGCOMM workshop on future directions in network architecture* (pp. 258–266). New York: ACM Press.
9. Eastlake, D., Schiller, J., & Crocker, S. (2005). *Randomness requirements for security*. RFC 4086, IETF.
10. Egevang, K., & Francis, P. (1994). *The IP network address translator (NAT)*. RFC 1631, IETF.
11. Ferguson, P., & Senie, D. (1998). *Network Ingress filtering: defeating denial of service attacks which employ IP source address spoofing*. RFC 2267, IETF.
12. Francis, P., & Gummadi, R. (2001). IPNL: A NAT-extended Internet architecture. In *Proceedings of ACM SIGCOMM 2001* (pp. 69–80). New York: ACM Press.
13. Holdrege, M., & Srisuresh, P. (2001). *Protocol complications with the IP network address translator*. RFC 3027, IETF.
14. Huttunen, A., Swander, B., Volpe, V., DiBurro, L., & Stenberg, M. (2005). *UDP encapsulation of IPsec ESP packets*. RFC 3948, IETF.
15. IEEE. (2007). *Guidelines for 64-bit global identifier (EUI-64)*. Tutorial, IEEE.
16. Johnson, D. B., Perkins, C. E., & Arkko, J. (2004). *Mobility support in IPv6*. RFC 3775, IETF.
17. Jonsson, A., Folke, M., & Ahlgren, B. (2003). The split naming/forwarding network architecture. In *Proceedings of 1st Swedish national computer networking workshop*, Arlandastad, Sweden, September 2003.
18. Liu, C., & Albitz, P. (2006). *DNS and BIND* (5th ed.). Sebastopol: O'Reilly and Associates.
19. Macker, J. (2003). *Interoperable networks for secure communications, task 6, phase 1*. Final report INSC-TASK6, North Atlantic Treaty Organisation (NATO).
20. Meyer, D., Zhang, L., & Fall, K. (2007). *Report from the IAB workshop on routing and addressing*. RFC 4984, IAB.
21. Moore, N. (2006). *Optimistic duplicate address detection for IPv6*. RFC 4429, IETF.
22. Moskowitz, R., & Nikander, P. (2006). *Host identity protocol architecture*. RFC 4423, IETF.
23. O'Dell, M. (1996). 8+8—*An alternate addressing architecture for IPv6*. Internet-draft draft-odell-8+8-00.txt, IETF.
24. Perkins, C. E. (Ed.) (1996). *IP mobility support*. RFC 2002, IETF.
25. Saltzer, J. H. (1993). *On the naming and binding of network destinations*. RFC 1498, IETF.
26. Schmid, S., Eggert, L., Brunner, M., & Quittek, J. (2005). TurfNet: An architecture for dynamically composable networks. In *Lecture Notes in Computer Science* (Vol. 3457, pp. 94–114). Springer, Berlin
27. Shoch, J. (1978). *Inter-network naming, addressing, and routing*. Internet experiment note 19, ARPA Network Working Group.
28. Stoica, I., Adkins, D., Zhuang, S., Shenker, S., & Surana, S. (2002). Internet indirection infrastructure. *ACM Computer Communications Review*, *32*(4), 73–86.
29. Turanyi, Z., Valko, A., & Campbell, A. T. (2003). 4 + 4: An architecture for evolving the Internet address space back toward transparency. *ACM Computer Communications Review*, *33*(5), 43–54.
30. Wellington, B. (2000). *Secure domain name system (DNS) dynamic update*. RFC 3007, IETF.
31. Weston, J. (2006). *Interoperable networks for secure communications, task 3*. Final report INSC-TASK3, North Atlantic Treaty Organisation (NATO).

Randall Atkinson is Chief Scientist at Extreme Networks, where he works on Ethernet and Internet technology. He has been active in Internet standards for over a decade and served two terms on the Internet Architecture Board (IAB). He has undergraduate and graduate degrees in Electrical Engineering and Computer Science from the University of Virginia.

Saleem Bhatti is a Professor at the School of Computer Science, University of St Andrews, UK. His research interests are in the general area of networked and distributed systems, and especially in network architecture, the control and management plane of the architectural landscape. His current focus is specifically on hybrid and ad hoc architectures, edge networks, and security related topics. He holds a B.Eng. (Hons), M.Sc. (Distinction), and Ph.D. degrees all from University College London (UCL), UK.

Stephen Hailes is a Professor in the Department of Computer Science at University College London (UCL), UK. His research interests are in the general area of wireless and mobile systems, distributed systems and security. His current focus includes wireless sensor networks, ad hoc networks, and applications of these technologies to control of autonomous vehicles, biological and veterinary sciences. He holds a B.A. and Ph.D. degrees from the Computer Laboratory, University of Cambridge, UK.