COMPUTER SCIENCE REVIEW 17 (2015) 25–42

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cosrev

Survey

A survey of identifier–locator split addressing architectures

Miika Komu[a,b], *Mohit Sethi*[a,b,*], *Nicklas Beijar*[a]

[a] *NomadicLab, Ericsson Research, Finland*
[b] *Aalto University, Finland*

ARTICLE INFO

Article history:
Received 22 July 2013
Received in revised form
13 April 2015
Accepted 21 April 2015
Published online 18 May 2015

Keywords:
TCP/IP stack
Internet architecture
Namespace
Addressing models
Mobility
Multihoming
Renumbering
Internet transparency
identifier–locator split

ABSTRACT

The TCP/IP architecture of the Internet was originally designed around the contemporary restrictions of large computers that were difficult to move around. However, electronics followed Moore's law, resulting in cheaper and smaller electronics for consumers, and portable devices, such as laptops and cellular phones, became pervasive. Consequently, the original restriction on static hosts was no longer true even though is still present in the design of the TCP/IP networking stack. The TCP/IP stack remains still constrained by its original design, which was effectively a design compromise to make the addressing model simpler. As TCP connections are created based on the same addresses used by the underlying network layer, the connections break when the address changes or is removed. Thus, the TCP/IP architecture is challenged in the temporal dimension of addressing as it was designed to assume stable addresses. This is not only problematic from the viewpoint of initial connectivity but also critical in sustaining of active data flows. In this paper, we first outline the challenges related to the inflexible nature of the TCP/IP architecture resulting from the fact that the same namespace is shared between the transport and network layers. We then discuss existing solutions for these challenges that arise from the transient nature of addresses in the TCP/IP architecture. Finally, we perform a qualitative analysis of the solutions discussed in the paper.

Contents

* *Corresponding author at:* NomadicLab, Ericsson Research, Finland.
E-mail addresses: miika.komu@ericsson.com (M. Komu), mohit.sethi@aalto.fi (M. Sethi), nicklas.beijar@ericsson.com (N. Beijar).

http://dx.doi.org/10.1016/j.cosrev.2015.04.002

1. Introduction

The basis of the TCP/IP architecture and Sockets API is founded on the assumption of stable or persistent addresses because hosts were immobile in the original Internet. Paradoxically, addresses are nowadays non-persistent, especially due to the advancements in modern, mobile end-user equipment and dynamic network environments. Initially, IP addresses were supposed to only be used at the network layer, but then TCP just reused the addresses as its connection identifiers [1].

While the reuse of IP addresses at transport and network layers offers relief from address management issues, it is effectively a layer violation that results in undesired dependencies between the layers. An IP address is tied to the local network topology and effectively defines "where" the host is located, whereas a transport-layer identifier defines "who" the connection end-point is [2]. Consequently, the transport layer becomes dependent on the location of the end-host and its data flows are interrupted when the end-host changes its point of attachment to the network.

The problem is further aggravated by applications that should be using application-layer identifiers (defining "what"), such as FQDN-based identifiers, but instead reuse the underlying IP addresses.[1] The reasons are historical; the Sockets API, the de-facto low-level programming interface for network applications, was designed before DNS and is therefore heavily encumbered with the use of IP addresses [4]. To further aggravate the problem, applications have also few means of discovering when IP addresses are stale because the Sockets API does not attach any lifetime to the data structures associated with IP addresses [1].

As TCP/IP and the Sockets API are universally deployed and adopted, changing their fundamental nature is economically challenging. To fix the misalignment between applications excepting persistent addresses and networking stack offering ephemeral addresses, various "workarounds" to fix TCP/IP stack have emerged, with varying degrees of backward compatibility. However, many of the solutions tackle only a single problem emerging from non-persistent addressing, and it is not always guaranteed that such band-aid solutions interoperate with each other seamlessly and efficiently.

The rest of the paper is organized as follows: Section 2 gives an overview of the challenges associated with addressing in the current TCP/IP architecture. The addressing architectures that are proposed in response to these challenges are described in Section 3. Section 4 provides a qualitative analysis of the presented architectures. Lastly, Section 5 concludes the paper with a discussion.

2. Challenges

In this section, we look at the challenges related to the transient nature of addresses in the TCP/IP architecture from

[1] Due to the coupled role of addresses, Fully Qualified Domain Names (FQDNs) could be considered as the new "who", and Universal Resource Locators (URLs) as the new "where" [3] due to the pervasiveness of the web.

the standpoint of the application layer. While the lifetime of addresses, and perhaps the quality of service related to the use of the address, is directly visible to the application, we describe a more fine-grained taxonomy of the related challenges. Challenges related to long-term disconnectivity as tackled by Delay Tolerant Networks (DTN) are beyond the scope of this paper.

2.1. *Mobility*

Mobility refers to a situation where a single device or an entire network of devices changes its attachment to the network, which typically occurs due to physical movement of the device(s). As the address prefixes are bound to a topological and geographical location, mobility requires a device to change IP address. The problem of changing addresses due to mobility is dual-fold: when a host moves to different network, it can no longer be reached by other hosts by the previous address and its existing data flows are terminated. In other words, mobility management includes challenges related to service discovery and sustaining of existing data flows. As our focus is more on the application-layer challenges, we focus on host mobility [5] rather than network mobility [6] solutions.

2.2. *Multihoming*

The multihoming challenge results from the existence of multiple alternative paths to an end-host and can be considered dual-fold. Site multihoming occurs transparently from end-host applications (aside from the performance overhead on latency and throughput) but causes an increase of routing tables in the core network. In contrast, end-host multihoming is more explicit and visible to applications. The application gets a choice of several addresses on which to open and accept connections on. In practice, many developers ignore issues related to end-host multihoming, despite that even a single network interface can have multiple addresses [1].

End-host multihoming can be seen as a subset of mobility [7]. However, the multihoming challenge does not require the physical movement of the host, but the challenge rather stems from the availability of more than one address for a single host, either on the same or different network interfaces. It impacts not only the client side but also the server side. At the client side, many hand-held devices support multiple access technologies such as Wireless LAN (WLAN), 4G and Bluetooth. One of the multihoming problems emerges when the client inadvertently chooses to initiate communications from a "wrong" address, and this may result in a firewall on the path or the server blocking the traffic. At the server side, a misconception is that an application bound to a specific IP address will also send outgoing data from the same address [1].

Finally, besides initiation of data flows, the multihoming challenge manifests itself during communications at both the client and server side. When the connectivity between an active pair of addresses fails, it would be useful to automatically switch to a functional pair of addresses. Alternatively, two data paths could be simultaneously utilized to maximize throughput.

2.3. *Site renumbering*

The third challenge is site renumbering. Renumbering issues surface for services that rely on hard-coded or manually configured IP addresses, or IP addresses stored between sessions (e.g. in P2P applications). On a host-level, many networks use dynamic addressing, based on e.g. Dynamic Host Configuration Protocol (DHCP). On a network-level, corporate mergers or a change in the ISP affects the IP address prefix of a site and requires the site to be renumbered. Due to human error, such stale addresses might still be forgotten in various places after the renumbering. For instance, hard-coded addresses might be remaining in access-control lists of a firewall and configuration files of web servers, as described in Request for Comments (RFC) 5887 [4]. The RFC further explains that the problems with cached or hard coded IP address literals may partially be attributed to the fact that the DNS look-up functions in the Sockets API do not pass the Time To Live (TTL) values to the application. As human error can result in downtime of services and economic loss, companies tend to avoid site renumbering.

2.4. *Routing scalability*

Besides address agility, the identity–locator split holds also the promise of improving the "routing scalability" of the "core" of the Internet, or Default-Free Zone to be more exact, which is currently facing some addressing related challenges. For example, many companies prefer Provider Independent (PI) addresses over Provider Allocated (PA) addresses to facilitate easier migration from one Internet Service Provider to another [2,8]. The drawback of PI addresses is that they do not aggregate as well as PA addresses; thus PI addresses create larger routing tables and challenge routing scalability [9]. However, the identity–locator split can be used to combine the benefits of PI and PA addresses. The identity namespace offers the same topology-independent functionality as PI addresses, and the locator namespace supports aggregation as it can be based on PA addresses. For applications, the unresolved scalability problems related to routing may cause degradation of Quality of Service in the future. The solutions (based on the identity–locator split) can limit the degradation and affect the way how applications identify other applications residing on remote hosts.

Another characteristic of the protocols is that many of them support at least site multihoming in addition to site renumbering. Many of the protocols are advocated as solutions to routing scalability. In this work, we treat problems with routing scalability as a symptom or side effect of the missing renumbering support rather than the root of the problem; the actual source of the problem is a non-scalable choice to either support site multihoming or site renumbering, such as is the case with PI addresses. Thus, "routing scalability" will not be discussed as its own separate problem but rather as a benign property of individual solutions to site renumbering.

2.5. *Internet transparency*

The fifth challenge is Internet transparency [10], which refers to withering of two desirable aspects of the original

Internet design: all hosts were universally addressable and intermediate hosts did not essentially modify packets. This resulted in end-to-end connectivity where all hosts were reachable by others. Unfortunately, this transparency was compromised by the evolution of the Internet by the advent of new types of middleboxes. To curb the depletion of IPv4 addresses, Network Address Translators (NATs) were deployed to allow a growth of hosts beyond the number of available public IP addresses. Unfortunately, the allocation of hosts in private realms hinders universal addressability.

Firewalls were invented to filter undesirable data flows on behalf of the application layer. However, firewalls and other type of middleboxes do not come without trade-offs. NATs complicate communications to servers and peer-to-peer (P2P) software, and firewalls practically enforce Hypertext Transfer Protocol (HTTP) to serve as the new narrow waist for the Internet [11]. Consequently, the Internet has been evolving into an end-and-middle architecture, favoring the client–server paradigm, and impeding the deployment of new transport and network-layer protocols.

RFC 6250 [1] describes two misconceptions about applications and reachability. First, reachability with NATs and firewalls is not always symmetric because clients can reach servers, but the reverse is not always true. The RFC mentions callbacks, which are present in the File Transfer Protocol (FTP) for example, as one source of the problem. For active data transfer in FTP, the client passes its IP address as a callback to the server, which then creates a new TCP connection to the client. If the client is behind a NAT, the creation of the TCP connection fails, resulting in a failure of the file transfer. A second issue described by the RFC is that reachability is not always transitive. For example, host A can reach B that can reach C, but this does not guarantee that A can reach C as the routes may be different between each node, with different firewalls or NATs between. This problem is also called a referral problem.

HTTP avoids the referral problem with its simple design of redirection. When a web server receives a request from a client that needs to be redirected to another server, the server instructs the client to connect directly to the other server instead of bluntly passing the client's address as a referral to the other server for connecting back, which would be problematic in the presence of NATs.

Architectures providing transparency allow future expansion of the Internet while providing transparent reachability to end host. Many of the solutions initially designed for IPv4 address depletion fall into this category, but NATs, for instance, fail to meet the latter criteria because they usually support uni-directional initiation of data flows.

3. Solutions

As discussed in the previous section, a root cause of the inflexibility of the TCP/IP architecture is that the transport and network layers share the same namespace, convoluting the semantics of the layers. Thus, it is only natural to decouple the namespaces, either in a strict way or loosely, in order to facilitate host mobility and multihoming. In general, this architectural approach is sometimes referred to as the identifier–locator split [1], and the Internet Architecture Board (IAB) has acknowledged that it is a viable way to modularize the TCP/IP architecture [2,12,8].

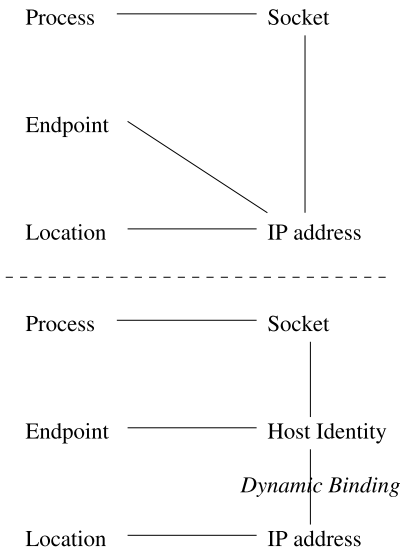

Fig. 1 – Vanilla TCP/IP vs. HIP socket bindings [16].

The paradigm of the identifier–locator split introduces one level of indirection in naming by introducing location-independent identities for the end-hosts that mask the details of the locators. In this section, we present addressing architectures that are based on this identifier–locator split paradigm. We exclude so called clean-slate solutions that are not based on IP based forwarding, such as Layered Naming Architecture [13] and Data Oriented Network Architecture [14].

We organize the solutions into three categories in the following sections: gateway-based solutions, core-edge elimination or separation based solutions.

3.1. *Solutions based on core-edge elimination*

In this section, we describe end-to-end solutions, where changes to realize the architecture are needed both at the client and server side. The term we employ for this category is "core-edge elimination" [8] which was originally specific to IPv6 deployment, but here we extend it to any new addressing architecture.

3.1.1. *HIP*

Host Identity Protocol (HIP) [15] is an inter-networking architecture that allows end-hosts to authenticate each other and protect their data flows. HIP solves the current issues of IP-layer mobility, multihoming, multi-access, NAT traversal, IPv4–IPv6 interoperability and security in a unique integrated manner. HIP implements an identifier–locator split by introducing a new name space called as the Host Identity namespace and a related layer known as the Host Identity (HI) Layer between the Transport and Network Layers. In HIP, hosts are identified by Host Identifiers (HI) instead of IP addresses. The difference between the old and new socket bindings between identifiers and locators is depicted in Fig. 1.

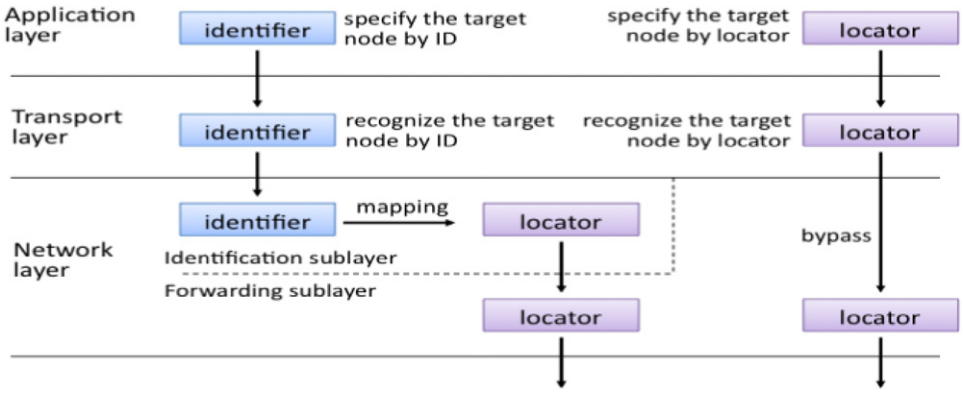

Fig. 2 – Concept of ID/locator split in LIN6 [18].

The host identifier is the public key component of a public–private key pair. A more compact representation of HI is called as Host Identity Tag (HIT) which is used in HIP control messages and API calls. A HIT is a 128-bit hash of the public key with a special IPv6 prefix. HIP also defines Local-Scope Identifiers (LSIs) which are assigned IPv4 addresses that the host locally maps to a HI.[2] Applications that do not support IPv6 can use LSIs instead of HITs. The use of these virtual "surrogate" addresses makes HIP backward compatible with legacy applications.

HIP defines several ways for discovering the mapping between a remote HI and the corresponding routable addresses [17]. A new DNS Resource Record (RR) type for HIP can be used to maintain the HI-to-IP address mappings of servers, and to detect when a server supports HIP. Alternatively, a distributed hash table can be used for storing the mappings. When an application sends data using an LSI or HIT, the HIP layer intercepts the packet and triggers a key exchange procedure between the two communication hosts. The key exchange and subsequent control messages are secured with public keys. During the key exchange, the two hosts also establish a symmetric key which is used for protecting the actual data. Typically, HIP implementations use IPsec for this purpose, albeit this can be negotiated during the key exchange. Since multiple interface addresses can be associated dynamically to the same HI, implementing multihoming becomes trivial.

When the IP address of a host changes (for instance, due to DHCP lease expiry, a switch from WLAN to 4G or a change in physical location), the host informs its connected peers about its new whereabouts. The upper-layer applications need not be aware of this and just continue to use HITs. This allows HIP-based mobility to function transparently from applications.

3.1.2. *LIN6*

Location Independent Addressing for IPv6 (LIN6) [18] is an IPv6 specific new protocol that intends to solve problems related to the mobility of end-hosts. LIN6 is based on the generic Location Independent Networking Architecture (LINA) which is designed to separate the identifier and locator in the Internet specifically for IPv6 addresses. The primary motivation behind such a split is mobility support. However, it provides additional benefits for end-host multihoming and security.

Conventional IPv4/IPv6 addressing does not support location-independent identifiers for a mobile node. To solve this problem, LINA modifies the network layer and introduces two separate entities for identifying and locating hosts. This approach of splitting the network layer is similar to the one taken by HIP.

LIN6 defines two new entities called the Interface Locator and Node Identifier. The Interface Locator is used for determining the current point of attachment to the network while the Node Identifier is used to identify the node itself in an interface-independent manner. Fig. 2 shows how the network layer is divided into two sub-layers in LIN6.

An application can choose to communicate with a remote application by either specifying the Interface Locator or the Node Identifier. When Node Identifiers are used, the application communicates with a peer application irrespective of the location of the host on which the application resides. However, when the application chooses to use the Interface Locators, it communicates with any node at that location without being concerned about the identity of the node.

What distinguishes LIN6 from HIP is that instead of dividing the network addresses into two separate namespaces as in HIP, it is based on an "embedded" addressing model. The designers of LIN6 argue that mobility protocols that introduce extension headers incur too much overhead, and should be avoided in the design of LIN6 to facilitate its deployment and adoption. Hence, LIN6 embeds the node identifier in the Interface Locator. An interface locator that contains the Node Identity embedded inside is called an ID-Embedded Locator.

The LIN6 address structure can be seen in Fig. 3. LIN6 assumes that each node has one or more globally unique 64-bit node identifiers assigned by some authority and this identifier is called as LIN6ID. In order to ensure that applications can use LIN6 identifiers just as they use IPv6 addresses, a "generalized LIN6 ID" is defined that contains the unique LIN6ID and a "LIN6 Prefix". The LIN6 prefix is a constant 64-bit value. This means that the generalized LIN6

[2] The unassigned IANA 1.0.0.0/24 prefix has been suggested for the use as LSIs.

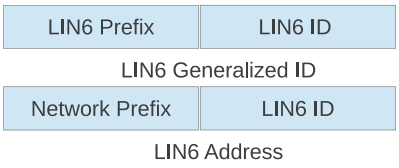

Fig. 3 – LIN6 addressing.

ID will remain the same for a particular host even if the point of attachment of the node to the network changes.

A "LIN6 address" acts as the current ID-Embedded locator of a LIN6 node. The lower 64-bits of the LIN6 address contain the LIN6ID and the upper 64-bits contain the network prefix of the subnet to which the interface is attached. A LIN6 address, therefore, changes when the location of the node changes.

During communication between two parties, the transport layer at the sending side specifies the destination with the generalized LIN6 ID. The identification sublayer shown in Fig. 2 extracts the LIN6ID from the generalized LIN6 ID and maps it to the subnet prefix. The sender obtains the mapping by first querying the DNS for the Mapping Agent (MA) of the destination node. Once the identification sublayer knows the MA responsible for the node, it sends a request to obtain the actual mapping. The identification sublayer finally uses this mapping to create the LIN6 address and passes it to the forwarding sublayer. The forwarding sublayer then forwards the packet towards the target node. A reverse procedure is applied at the receiving end to obtain the generalized LIN6 ID which is then passed to the transport layer. Therefore, from the point of view of the transport layers, communication occurs between LIN6 generalized IDs, which are location independent.

LIN6 nodes can communicate with traditional IPv6 nodes by using IPv6 addresses instead of generalized LIN6 IDs. Thus, LIN6 is backwards compatible with IPv6. Multihoming support is provided by allowing a single generalized LIN6 ID to be associated with several real IPv6 interface addresses. The authors of LIN6 have proposed to use IP level security such as IPSec for protecting communication between LIN6 nodes.

3.1.3. MAT

Mobile IP with Address Translation (MAT) [19] is an end-to-end architecture that supports seamless IP handoff. Similar to HIP, MAT modifies the end-hosts and makes no changes to the core network. Identical to some other mobility protocols, MAT uses two addresses for a mobile node. The "Home Address (HA)" is the permanent address that specifies a node's identity, and, the "Mobile Address (MA)" is a temporary address that represents the current location of the node. MAT is essentially an extension of Mobile IPv6 [20].

The network layer is divided into two sublayers—a "MAT sublayer" that performs address translation, and a "Delivery sublayer" that delivers packets according to the translated destination address. When the MAT sublayer receives a packet from the transport layer, MAT needs to determine whether the target node supports MAT or not. It also needs to find out the address of the "IP Address Mapping Server (IMS)" for the target node when it is a MAT node. The architecture proposes to use DNS for finding the address of the IMS responsible for a node. If the DNS response does not include any IMS address, it is safe to assume that the target node does not support MAT. This concept of storing the static IMS address for each node in DNS is similar to the one taken by LIN6 for storing Mapping Agents of every node in the DNS.

The MAT sublayer then uses this mapping server to find the Mobile Address (MA) for the given destination Home Address (HA). Now, depending on the type of the target node (MAT/non-MAT), and the mobility status of source and destination nodes, the MAT sublayer performs address translations.[3] After the address translation, the packet is passed to the delivery sublayer, which transmits the packet for normal routing. At the receiving end, a reverse process is applied to the packet before it is passed to the transport layer.

3.1.4. FARA

Forwarding directive, Association and Rendezvous Architecture (FARA) [21] presents a generic architecture that tries to solve the problem of overloading of IP address semantics by separating the identity from network layer routing. FARA defines a set of components and relationships among them but leaves out many design decisions to a particular implementation.

The three basic components of FARA are:

- Entity: an end point of communication. It is also the unit of mobility.
- Association: logical persistent communication links between the communicating parties.
- Communication Substrate: responsible for delivering data on behalf of the logical associations. It is assumed to provide connectionless packet delivery.

When two end-hosts A and B wish to communicate using FARA, they need to establish an association between them. This association is identified by an "Association Id (Aid)" which is unique among the communicating parties A and B. This association is comparable to IPSec security associations set up in HIP, although the data flows may not necessarily be encrypted. FARA leaves it up to the communicating parties to decide the type of authentication and encryption they wish to use. To set up the association, an initial handshake is necessary for which A needs the "Forward Directive (FD)" of B. The destination FD contains information for routing data to the intended recipient. For A to find the FD of B, FARA provides a directory service called "FARA directory service (fDS)" which resembles DNS. The initial handshake also includes "Rendezvous Information (RI)" which is used for setting up the association and assigning it a unique Aid.

Another factor that separates FARA from some of the other identifier–locator split architectures is that it does not introduce a new globally unique namespace. This means that in the fDS, there can be multiple entries for the same $\langle FD, RI \rangle$ pair and it is up to the rendezvous mechanism to disambiguate between them.

[3] If the participating nodes are MAT nodes, then, both the source and destination address translation is performed.

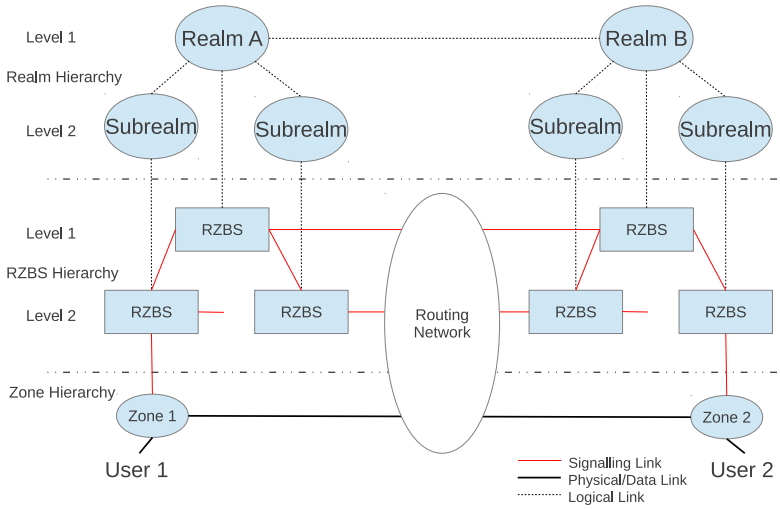

Fig. 4 – MILSA.

3.1.5. *MILSA*

Mobility and Multihoming Supporting Identifier–Locator Split Architecture (MILSA) [22] is a solution proposed to solve problems associated with naming, addressing and routing in the Internet. Its architecture is based on three fundamental principles:

- Separation of identifier and locator to isolate upper layers
- Separation of logical trust relations (explained below) from physical connectivity relations
- Separation of signaling and data forwarding infrastructure for better routing efficiency.

As seen in Fig. 4, MILSA has 3 hierarchies: Zone Hierarchy, Realm Hierarchy and Realm Zone Bridging Servers (RZBS) Hierarchy. The zone hierarchy refers to the physical network links between end-hosts. The realm hierarchy represents the trust relationships between different objects involved.[4] A realm is a logical concept with no physical links to the other two hierarchies. Lastly, the RZBS hierarchy consists of Realm Zone Bridging Servers which are responsible for mapping identifiers to locators. The Realm Zone Bridging Servers maintain signaling links among themselves and essentially form an overlay network.

MILSA defines a Hierarchical URI-like Identifier (HUI) naming system. An example of such an identifier can be "User-1.Subrealm-2.Realm-A". The leftmost part of the identifier, "User-1", is a flat name and should be unique within a subrealm. The remainder of the identifier is obtained from the logical position of the object in the realm hierarchy. In this example, User-1 belongs to Subrealm-2, which is a part of Realm-A.

In the MILSA architecture, trust relationships are set up and maintained between the hierarchical realms. This means that a RZBS belonging to one subrealm may trust another RZBS in a different subrealm, either directly, or through a RZBS at a higher hierarchy level as shown in Fig. 4. Realms higher in the hierarchy have higher responsibility and are in charge of their subrealms.

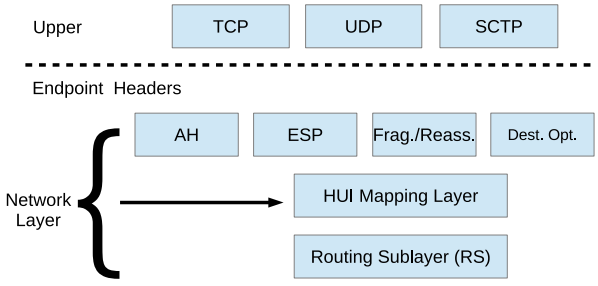

Fig. 5 – HUI Mapping sublayer [22].

A new HUI Mapping Sublayer (HMS) is introduced at the network layer as shown in Fig. 5. HMS is located between the Endpoint Headers (which include Authentication Header (AH) or Encapsulating Security Payload (ESP) headers) and the Routing Sublayer (RS). The upper layers use HUI for communication and are not aware of the current destination locator. The lower layers use locators for routing and are not aware of the identifiers. HMS performs the mapping between the identifiers and locators. If multihoming is enabled, HMS is also responsible for maintaining HUI-to-locator mappings and monitoring the reachability of all the links.

Objects register their mapping with the RZBS responsible for the subrealm to which they belong. When an object needs to communicate with an object located in another realm, it sends a mapping request to its RZBS which then follows the hierarchy until a mapping is found. Since objects update their mapping when a change occurs, the RZBS system always returns the most up-to-date locator for the destination. RZBS can also provide backup and proxy service. Since the mapping request/response and other control messages are signaled between the RZBSs separately from the data messages exchanged between the end-hosts, a separation between the control and data planes is achieved. This separation enhances the routing efficiency of the architecture.

[4] An object can be an end-host, a router, a RZBS server or any other such entity.

3.1.6. SHIM6

Level 3 Multihoming Shim Protocol for IPv6 (SHIM6) [23] is a protocol proposed for providing multihoming support for the IPv6 address space. It also provides failover and load sharing capabilities without compromising on the scalability of the global routing system. Similar to the other solutions presented, SHIM6 also decouples the identification role from the location role of IP addresses. However, it does not introduce a new name space for performing this decoupling.

The solution creates a new sublayer within the IP layer called the SHIM6 layer. This new sublayer, along with two new protocols, Shim6 and Reachability Protocol (REAP) [24], form the core of the solution. SHIM6 creates a context between two communicating parties with a 4-way handshake which contains information that uniquely identifies the session and a set of available addresses for the nodes. SHIM6 arbitrarily chooses one of the available addresses as the Upper Layer Identifier (ULID) which is used for session identification. All the other available addresses are used as locators. A combination of Hash Based Address (HBA) [25] and the Cryptographically Generated Address (CGA) [26] are used to bind a set of addresses to a particular node and verify the authenticity of an address claim by a node.

SHIM6 uses a failure detection and recovery mechanism called as the REAchability Protocol (REAP) [24]. Failures can be detected either by the absence of keep-alive messages or information from upper layer protocols. The recovery mechanism finds a pair of working addresses with exploration.

3.1.7. Six/One

Six/One [27] is an IPv6 specific solution for routing and addressing that aims to reduce the routing table size and updates in the Default Free Zone (DFZ). The primary reason behind the increased number of entries and the number of updates to the routing tables is the use of provider-independent addresses by edge network operators. Such provider-independent addresses can provide several benefits to edge network operators, including better traffic-engineering options, rehoming to a new provider without reconfiguration costs and flexible provider selection during multihoming. However, these addresses require indirection which leads to slow distribution of translation data (provider address to edge network address) or aggressive signaling [28].

The proposed Six/One architecture solves this problem by maintaining a single IPv6 addressing space for both edge and provider domains. In Six/One, edge networks use the address space given by each of their providers. Hosts can use all the address spaces that the different providers have assigned to the edge network operators (in case of multihomed edge networks) interchangeably without breaking an active connection. This concept is similar to the one used by SHIM6. However, what makes it different from SHIM6, is the fact that all the host addresses differ only in their 64 lower order bits. This enables the edge or provider network to change addresses on the fly depending on the provider through which the packet is routed. The address that a host uses to contact another host acts only as a suggestion and the edge network can choose to heed to this suggestion or overwrite the address and choose another provider for the packet. Hosts then modify the subsequent packets according to the rewrites of the edge network.

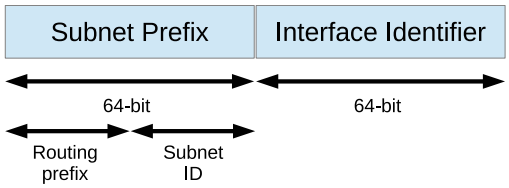

Fig. 6 – Six/One address structure.

In Six/One, a host still uses a 128-bit IPv6 address where the upper 64-bit form the “subnet prefix”, and the lower 64-bit form the “interface identifier”. The subnet prefix is advertised by the access router and the host appends it with the interface identifier to form a complete IPv6 address. For the edge network, a subnet prefix can be seen as a composition of “routing prefix”, which indicates the provider from which this address was obtained, and a “subnet ID”, which is used for internal routing. The length of the routing prefix determines the size of the address space which depends on the contract between the edge network operator and the provider. The addressing format is shown in Fig. 6.

In this architecture, hosts are configured to have sets of addresses, called “address bunches”, which are different only in subnet prefixes and can be used interchangeably without breaking active sessions. This translation between addresses is transparent to upper layer protocols. A host can obtain an address bunch via stateless/stateful auto-configuration or manual pre-configuration. A host must know its own address bunch as well as the address bunch of the host that it is communicating with. The hosts maintain this information in a per communication-session “context”. The local and remote address used by protocols above Six/One layer are called as “primary addresses” and do not change during a session. The addresses which are used by Six/One after translation are called as “active addresses”.

The address bunch used by hosts, need to be protected from impersonation attacks by malicious nodes, as a malicious node may register its own address as the active address in the address bunch. Six/One therefore creates a cryptographic binding between the various subnet prefixes and the common node identifier. Similar to SHIM6, Six/One uses Cryptographically Generated Addresses (CGA) [26] and Hash-Based Addresses (HBA) [25] for generating an address bunch and verifying the addresses.

3.1.8. ILNP

Identifier–Locator Network protocol (ILNP) [29] separates between the “Node Identifier (NID)”, which is a fixed non-topological unique name of a node, and the Locator, which is topologically tied and used for routing and packet forwarding. A host may have several NIDs and several Locators. NIDs are based on a modified EUI-64 format and they can have a global or local scope. Routing is entirely based on Locators, which have the syntax of IP unicast routing prefixes, allowing current routers to forward traffic without modification. Applications use FQDNs instead of using the Identifier directly. The mapping from a FQDN to a set of

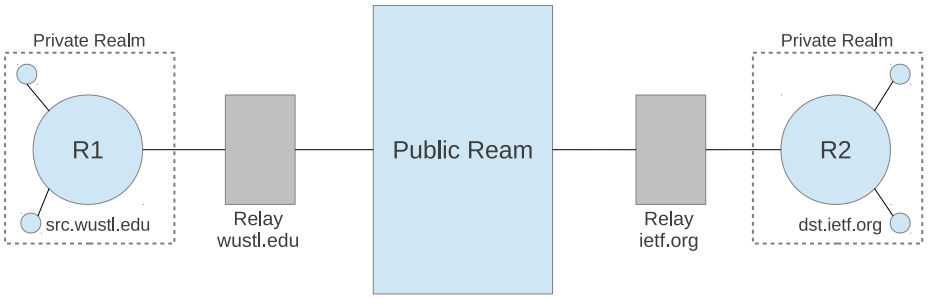

Fig. 7 – TRIAD Internet topology [32].

Identifiers and Locators is for each host stored in new resource records in the DNS.

Packets contain the source and destination in the form of a pair of an Identifier and a Locator. Thus, a network-layer endpoint is identified by the pair of a Locator and an Identifier. These pairs are encoded in IPv4 and IPv6 packet headers in different ways. In IPv6, the Identifier and Locator are each encoded with 64 bits of the 128 bit address. In IPv4, the Locators are in the address field of the IP header, while the Identifiers are transported in option headers. At the transport-layer, endpoints are identified with the Identifiers and port numbers.

ILNP uses dynamic bindings between Identifiers and Locators, and between Identifier–Locator pairs and the node interface. Locators can be updated due to connectivity changes. When a Locator is changed, the host informs its corresponding nodes with a Locator Update message and updates the locator value in DNS.

ILNP supports multi-homing by dynamically binding the provider-independent NID to a provider-aggregatable Locator, which is topologically bound. This allows a multi-homed site to have transport-layer and network-layer session resilience without globally visible provider-independent prefixes.

ILNP can be deployed in IPv6 and IPv4 networks without changes in applications, but end hosts must be modified to support ILNP. ILNP can be incrementally deployed without upgrading all nodes at once. While existing routers do not need to be modified, the first hop router facing the hosts needs to be changed and support the modified Address Resolution Protocol (ARP) protocol.

3.2. *Gateway-based solutions*

In this section, we describe "hybrid" solutions that require changes both at the end-hosts and middleboxes. Typically, such solutions require a modified client and a new gateway or a proxy that terminates the traffic, so that the server side remains unmodified.

3.2.1. *TRIAD*

The Translating Relaying Internet Architecture (TRIAD) [30] was proposed to solve the problem of IPv4 address depletion in the NATted Internet without the painful transition to IPv6. In TRIAD, Internet is viewed as a hierarchical set of interconnected realms, as shown in Fig. 7. At the leaf level, the *address realm* is an individual local network (e.g. LAN or wireless network). At higher levels, the address realm refers to local and global ISPs. The firewalls and border routers in the realms are extended to work as TRIAD "Relay Agents (RA)" that assist the communication between different realms.

TRIAD advocates the use of DNS names to identify each end host uniquely from different contexts. Within a realm, the routing, naming and addressing mechanisms work as they do in the current IPv4 architecture. Therefore, a realm requires no host or router changes. The relay agents (RAs) interconnect realms, provide naming and routing functionalities along with Wide Area Relay Addressing protocol (WRAP) based forwarding. WRAP is a shim protocol that carries the transport layer headers and data as its payload, and is similar to other IP encapsulation protocols. The WRAP header contains a pair of Internet Relay Tokens (IRTs): a forward token and a reverse token. An IRT is an variable length field that provides additional addressing space to what IPv4 provides.

Routing between two realms is based on FQDNs, similar to the initial stage of IPNL and is performed by relay agents. The relay agent responsible of a realm maintains a directory service for name lookups within that realm, as well as determines the next hop for names that refer to external realms. During the initial route discovery, each relay agent looks up its directory service to select the egress node for the packet and appends it to the forward token in the WRAP header. The tokens form a list of IP addresses of the egress nodes for the RAs on the path. Such a list provides transparent relaying for the end-hosts. In order to provide relaying for end-hosts, the RAs write a random nonce in the token, instead of writing the IP address of an egress node. The RAs also maintain state information that maps the nonce to an egress node so that packets can be routed based on the nonce values. Therefore, the actual packet forwarding remains opaque to the communicating parties. The end-to-end semantics is maintained by introducing the concept of a pseudo transport layer (TRIAD-TCP) which is based on TCP options. This layer is bound to names rather than the addresses.

3.2.2. *IPNL*

Similarly as TRIAD, Internet Protocol Next Layer [31] is an architecture motivated by the issue of IPv4 address depletion. It is a NAT based architecture which modifies the end-hosts and NATs. IPNL introduces a new layer between the network and transport layers that performs end-to-end routing while leaving the point-to-point routing for IP addresses.

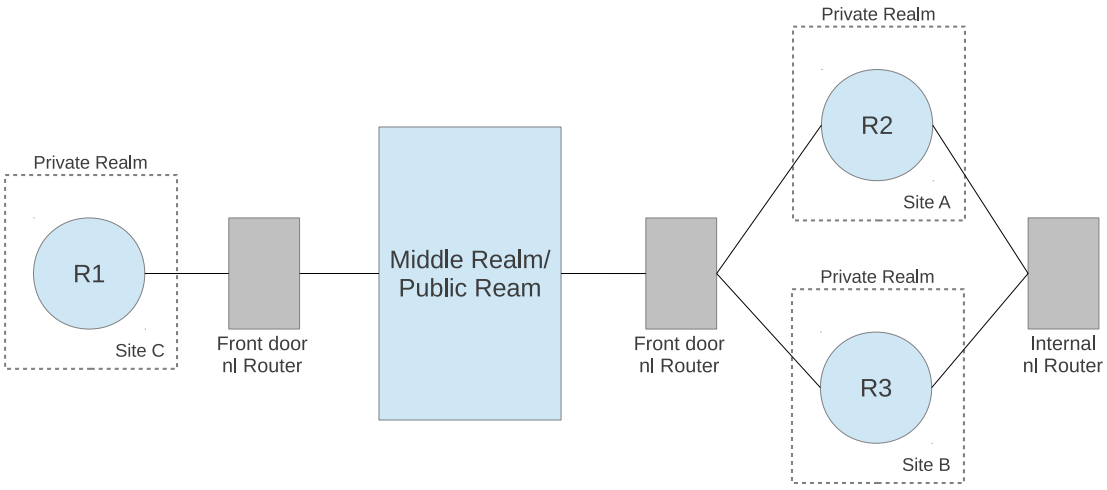

Fig. 8 – IPNL Internet topology [32].

IPNL considers the Internet topology to consist of potentially recursive private address realms connected to the public Internet. IPNL refers to NAT boxes as "nl-routers" and the public Internet as "middle realm". This topology is presented in Fig. 8. A nl-router that connects a private and public realm is called as "frontdoor nl-router", while a nl-router connecting two private realms is referred to as a "internal nl-route". IPNL introduces new packet headers that can have two kinds of routable addresses: FQDNs and IPNL addresses. Packets are addressed by a FQDN and/or an IPNL address. Therefore, IPNL has two explicit namespaces, a FQDN namespace and a namespace of IPNL addresses. Every realm in IPNL has DNS zones and each zone is associated with one realm called "home realm". A host can be attached to a realm different than the home realm. Such a host is called "visiting host".

For a packet to be forwarded on the basis of FQDN based routing, nl-routers dynamically maintain routing information for all zones behind the same frontdoor. The routers can have explicit forwarding rules for all zones behind the same frontdoor or use default forwarding to the frontdoor. At the minimum, the frontdoor should have forwarding entries for all the zones behind it. In case a packet is destined to a zone not belonging to the local realm, the frontdoor uses conventional IP routing to route packets across the middle realm. For routing in the middle realm, the frontdoor performs a DNS lookup to find the address of the neighboring frontdoor.

Although FQDNs are static and fully routable, they are also of variable length and consequently, it is costly to route packets with FQDN address. On the other hand, IPNL addresses have a short fixed length (10-bytes) and have the advantage of being efficiently routable. IPNL addresses are formed with the concatenation of Middle Realm Global IP (MRIP), Realm Byte Number (RN) and End-host IP (EHIP). IPNL calculates RNs for realms and associates zones to realms. The path discovery involves learning of various parts of the IPNL address such as source and destination MRIP, source and destination RN and the destination EHIP. During the process of path discovery, FQDN addresses are used. After path discovery, subsequent packets just carry the IPNL addresses which provide better efficiency.

Every node in IPNL requires a FQDN, which requires the scaling up of the DNS system. A distributed mechanism has been suggested for implementing this scale up. In the proposed method, each of the nl-routers is supposed to know about each host in each realm that it is connected to. It assumes that each realm contains only a few hosts.

In summary, IPNL is based on FQDN based forwarding. IPNL provides site isolation, and changes in the MRIP of the frontdoor never show up in internal routing. This has two major benefits:

- Renumbering of hosts is not required when the site's prefix changes.
- Connection to a single host persists even when the frontdoor address changes.

3.2.3. i3

Internet Indirection Infrastructure (i3) [33] was conceptualized to solve the problems faced in implementing services such as anycast, multicast and mobility using the current point-to-point communication abstraction of the Internet. The primary motivation for the architecture was the observation that communication services such solving mobility and multihoming issues have some similarities, but are disjoint and cannot not be reused by other services. i3 intends to provide a more unified architecture which can also be reused to provide a variety of other services.

The i3 model decouples the act of sending messages from the act of receiving messages. Nodes in the network send and receive messages with logical addresses rather than physical addresses. When a sender A has some data to send, it sends out a pair of ("id", "data"), where "id" is the logical address of the destination and "data" is the actual payload it wishes to send. Correspondingly, receivers send out so called triggers when they wish to receive some data. A trigger is in the form of a pair ("id", "address"), where the receiver wishes to receive the data addressed for "id" at its physical address specified as "address".

The Chord lookup protocol [34] is used for creating an overlay of i3 servers which are responsible for handling the

rendezvous and forwarding between senders and receivers. When the sender has a packet to send, it routes it towards the i3 server handling packets for that particular logical address. The server looks for registered triggers and forwards to any interested recipients over IP. To increase the routing efficiency, the protocol suggests the use of a variety of caching mechanisms.

Hi3 [35] is another proposal which intends to combine the benefits of i3 and HIP together. While HIP provides end-to-end opportunistic IPSec security, i3, with its rendezvous abstraction, protects against a number of denial of service attacks. HI3 combines the two protocols to provide an architecture that is more secure than i3 and at the same time, more flexible and DoS resistant than HIP.

3.2.4. 4 + 4

4+4 is an addressing architecture [36] that aims to extend the IPv4 address space while retaining the original end-to-end semantics of the Internet. A 4+4 extended address is formed by concatenating a 32-bit public address with a 32-bit private address. The public address is used to select the address realm in the Internet, while the private address is used to select the node inside the address realm. The address of the NAT that connects a realm to the Internet is used as the public address. When a node is directly connected to the Internet, it uses the existing public address as the public part and the 0.0.0.0 as the private part. The public and private parts of the address are also called "level 1" and "level 2" addresses.

A 4+4 packet has two extra 32-bit fields for the "inner" source and destination addresses. The outer address of the packet is always the one that is understood by routers in the current realm. Therefore, when the packet is traversing the private realm, the private address is part of the outer header, and, when the packet is in the public Internet, the public address is part of the outer header. This ensures that existing routers can forward 4+4 packets without understanding their semantics.

Upgraded NATs called "Realm Gateways" form an integral part of the 4+4 architecture. The realm gateways not only provide address translation functionality, but also 4+4 operations of IPv4 routing and so called address swapping (as explained later). 4+4 suggests extension of the current DNS system by providing unique domain names for each host in the network. There are two addresses for each host, a level 1 address which is the IP address of the realm gateway, and a level 2 address which is the IP address of the node. In order for a local host to find out the 4+4 address of a remote host, the local host needs to perform two DNS lookups.

For two 4+4-aware hosts to communicate, host A performs a DNS lookup for the 4+4 address of host B.[5] A then checks the DNS response to verify if any of the level 1 addresses returned matches to one of its own level 1 addresses. If there is a match, then both of them are located in the same realm and A sends an IPv4 packet towards B via internal routers. However, if there is no match, then B belongs to another realm and level 1 addresses have to be used. Let us assume that X and Y are the level 1 addresses (realm addresses) of A and B respectively. Then, a host A creates a 4+4 packet which has outer source/destination fields as A and Y and the inner source/destination fields as X and B. Thus, initially routing appears to be from A to Y. The realm gateway of A swaps the source address from A to X. Therefore, to the public Internet, routing appears to be from X to Y. At the realm router of B the destination address is swapped from Y to B. Thus, the source address inside the realm appears to be X, and the address swapping supports backward compatibility with existing networks.

However, this approach has its own drawbacks which include failure of end-to-end security mechanisms, a significant overhead involved with assigning a Fully Qualified Domain Name (FQDN) to each host and having two entries in the DNS system, thus, requiring two lookups.

3.2.5. *NodeID*

Node Identity Internetworking Architecture (NodeID) [37] builds on HIP to provide identity based overlay routing for hosts to discover their mutual, routable IP addresses. Thus, NodeID avoids the use of a special DNS Resource Record (RR) or a HIP proxy for determining the address corresponding to a Host Identity. It simplifies the current Internet topology by assuming a "core" at the top level which contains all the static domains (these domains have stable connectivity and do not change their location or point of attachment frequently). Non-core networks, such as Personal Area Networks (PANs) are considered dynamic and attach to the edge of this core or other dynamic domains via NodeID Routers (NRs). Every node has a Node Identifier (NID) which, similarly to HIP, is the public key component of a public–private key pair. The NRs have their identifier–locator mappings stored in a Distributed Hash Table (DHT) across the core.

At the edges of the core, a static routing policy is used and the routes that cannot be resolved are forwarded to the NR responsible for that domain. The NR then looks up the destination NR from the DHT. Such a routing approach avoids the insertion of core NIDs in the DHT which would negatively effect the scalability and efficiency of the architecture.

The fact that NodeID uses a DHT in the core, is similar to the DHT based EID-to-RLOC mapping suggested in LISP-DHT [38]. Therefore, NodeID can be seen as combination of HIP-like identifiers on end-hosts and LISP-like routing in the core.

Unmanaged Internet Architecture [39] is similar to the NodeID architecture and uses HIP-like identifiers along with DHT-based routing. For brevity, we do not present it separately.

3.2.6. *NUTSS*

NUTSS [40] provides a clear separation between the control and data planes, with the middleboxes divided into two types: policy-boxes (P-boxes), which handle policy decisions (firewall-like entities), and forwarding middleboxes (M-boxes), such as NATs, which forward traffic and perform address/port translation. Signaling follows different paths for the control and data planes. The P-boxes form a hierarchical

[5] We assume A and B also represent the level 2 IP addresses of the respective nodes.

overlay used for signaling and routing based on name-based identifiers. For this, the authors do not specify the protocol but consider Session Initiation Protocol (SIP) as one option. Each network has at least one P-box, which is connected to a P-box at a higher hierarchical level. Multihomed networks consequently connect to several parent P-boxes.

Endpoint names consist of the user, a DNS domain and a service, with wildcards allowed. Thus, it is possible to address a given service or user on a host. Each P-box maintains routes to the names below it. To create these name-based routes, a host registers itself through a hierarchy of P-boxes, and an intermediate domain may aggregate individual registrations into a joint registration. In a similar way, filters can be installed in order to deny traffic to given destinations.

A host must explicitly signal the creation of a flow via the P-boxes. To admit a flow, the P-box generates a security token and selects through which of the associated M-boxes the flow is directed. Separate address-routed signaling is required to set up the data path. For accepting a flow, an M-box requires a token assigned by a P-box.

To mitigate deployment of NUTSS, a three phase deployment plan is presented. First, public P-boxes are deployed and a few end host applications employ NUTSS with the NAT traversal capability as the driving killer application. Then, individual networks deploy their own p-boxes, and end-hosts learn about their existence via DHCP extensions. Finally, legacy middleboxes are replaced with M-boxes that also act as proxies for communications for the remaining legacy end-hosts.

3.2.7. *HRA*

Hierarchical Routing Architecture (HRA) [41] utilizes hierarchical routing to support routing across multiple independent address spaces. There are two different namespaces defined in HRA: the Host Identifier namespace and the Host Locator namespace. Similarly as in HIP, HRA has two levels of mappings: from the host name to a host identifier, and from the host identifier to a host locator.

The host identifier namespace consists of 128-bit host identifiers similar to the Host Identity Tags (HITs) in HIP. However, the host identifiers are composed of two parts: the first part is the Administrative Domain (AD) ID, and the second part is a hash of the AD ID concatenated with the public key of the host. As shown in Fig. 9, the AD ID is a globally unique hierarchical label containing organization affiliation. This includes the carrier, country and region identifiers. Such a hierarchical host identifier namespace eases the management of a global identifier namespace and also improves the lookup performance of the identity-to-locator mapping system.

The host locator namespace is divided into multiple independent address spaces which are referred to as Locator Domains (LDs). Each locator domain may deploy an independent address space such as IPv4, IPv6, global and private address spaces. Additionally, different LDs may deploy overlapped address spaces. Thus, HRA does not require the IP addresses to be globally unique. Each LD is identified by a hierarchical globally unique ID as seen in Fig. 10.

Within HRA, the mapping of host name to the HI is stored in DNS, while the mapping of HI to the LD ID and Locators

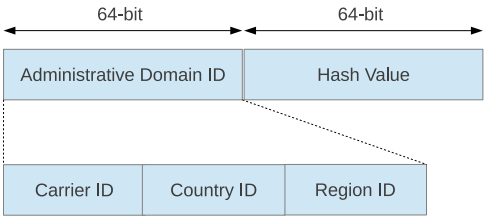

Fig. 9 – Hierarchical host ID format.

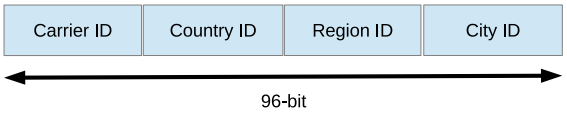

Fig. 10 – Hierarchical locator domain ID format.

is stored in a distributed hash table. This requires an end-host to perform a two-step query to obtain the HI, LD ID and locator of the destination host.

All Locator Domains are connected via Locator Domain Border Routers (LDBR). Each LDBR has at least one locator assigned in each LD to which it is connected. These locators have meaning and uniqueness only within that LD. Adjacent LDBRs exchange LD reachability information with an inter-LD routing protocol. The authors of HRA propose that BGP can be extended with a new address family to fill this need. They also claim that alternatively a new link-state protocol or distance-vector protocol can be designed as an inter-LD routing protocol.

When an end-host A wishes to communicate with another end-host B, it firstly obtains the locator and the LD ID information of destination host B from a distributed hash table before initializing a communication. Upon obtaining the LD ID and locator information, A fills in the destination IP address with the destination host locator of B, if the LD ID obtained is the same as its own. Otherwise, it fills the destination IP address in the IP header with that of its LDBR locator. In such a scenario, the LDBR of A rewrites the source IP address with its own locator, and destination IP address, with the LDBR address matching the LD ID of B. In this fashion, the packet is routed along different LDs. Once the packet arrives at the final LDBR of B in the destination LD, the LDBR fills in the destination IP address with the destination host locator, and source IP address with one of its locators that is routable in that Locator Domain.

HRA takes care of host mobility by enforcing end-hosts to register to the mapping system with their new location information when they change their points of attachment due to mobility or due to re-homing. The mapping of HI to LD ID and the corresponding locator also shows the multihoming status; multiple HI to LD ID mappings exist if the host is multihomed, but only one of the mappings can be used for communication.

3.2.8. *Mobile IP*

Mobile IP [42,43] supports mobility management for end-hosts. Essentially, MobileIP implements identifier–locator

split where a Home Address (HA) acts as the identifier and a Care-of Address (CoA) as the locator. In contrast to many other approaches, both the identifiers and the locators in Mobile IP are routable addresses.

Mobile IP makes use of a Home Agent located in the Home Network of the Mobile Node, which intercepts packets sent to the Home Address and tunnels them to the current Care-of Address of the mobile node. Several choices are available for the tunnel protocol, including IP-in-IP (for IPv4) and Routing Extension Headers (for IPv6).

The Care-of Address may be assigned to the Mobile Node itself or to a Foreign Agent, which forwards the packet to the Mobile Node over the visited network. The latter case is preferred in IPv4 because of address shortage. The Mobile Node has to detect changes in its Care-of Address, e.g. when it visits another network, and keep the Home Agent informed about its current Care-of Address.

The implementations for IPv4 and IPv6 differ in the way signaling is performed. In Mobile IPv4 [42], a dedicated protocol is used for registering. In Mobile IPv6 [43] registration signaling, called Binding Updates, is performed using a Mobility Header appended after the IPv6 header together with standard IPv6 security headers. Another major difference is that a Mobile IPv6 node is able to send binding updates directly to the corresponding node, which avoids triangular routing via the Home Agent.

Mobile IP primarily addresses end-host mobility, but indirectly enables also host multihoming. It is not a solution for multihomed networks and causes problems in site renumbering [4]. Mobile IP requires new infrastructure in the form of Home Agents.[6] Mobile IP requires support in the mobile node and, for IPv6 path optimization, also in the corresponding node. In 4G networks, Mobile IP is used in a proxy-based solution, i.e., the use of Mobile IP is hidden from the cellular devices, and the signaling is performed by network elements on the behalf of the devices.

The main benefit of Mobile IP is mobility support. With the help of a Home Agent, this can be achieved without any modifications at the server side, but this is a mixed blessing because additional infrastructure requires additional deployment, and triangular routing introduces additional latency. As another drawback, Mobile IP is not really a standalone protocol from the viewpoint of security, and typically is coupled with Internet Key Exchange (IKE).[7]

3.3. *Solutions based on core-edge separation*

In this section, we describe middlebox-based solutions that are realized, e.g., at edge-routers instead of deploying the solution at the end-hosts. We use the term "core-edge elimination" for such solutions [8].

3.3.1. *GSE*

Global, Site, and End-system address elements (GSE) [45] is an alternate IPv6 architecture. The GSE proposal proposes

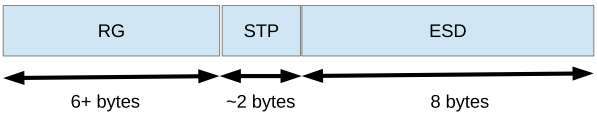

Fig. 11 – GSE IPv6 address structure.

that the original IPv6 specification continues to rely on provider based aggregation of addresses, which can break down with multihoming and re-homing as explained in the previous section. GSE proposal states that even if the current Classless Interdomain Routing (CIDR) aggregation levels were to continue at the same efficiency, the increase in the number of routes would still be a serious problem for route computations in the future because of the larger IPv6 address space.

GSE introduces a different IPv6 architecture that mitigates this routing state explosion problem by providing aggressive topological aggregation. GSE additionally supports multihoming and allows for independent evolution of routing and forwarding models with no impact on end-hosts. In GSE terminology, "Site" is a private, local network that forms the fundamental unit of attachment.

The GSE architecture separates the identification and location of end-hosts. The proposed IPv6 address contains a 6-Byte "Routing Goop (RG)", a 2-Byte "Site Topology Partition (STP)" and an 8-Byte "End System Designator (ESD)" as shown in Fig. 11. The Routing Goop is the point of attachment of a site to the Internet and is used for routing in the global Internet. When a site is multihomed, it associates one Routing Goop per attachment point. The ESD identifies an interface of the end system and STP (optionally) is used for routing within a site. ESD and STP are hidden from the global Internet, and intra-site routing is not affected by a change in the point of attachment/routing goop. The RG acts as a locator and is used to identify the site to which an end-host identified by the ESD belongs to.

When creating a packet, the source host fills the destination address with a 128-bit IPv6 address that it receives from a DNS lookup for the destination. This address includes the RG of the destination as provided in the DNS response. If the destination is not within this local site, the packet leaves the site through one of the border routers that modifies the source RG, which will be used for the packets on the return path of this communication session. The RG of an address is modified by a site border router to isolate the internal routers from any external changes that may occur. This also allows the ISPs to aggressively aggregate the RGs according to their needs. The transport layer and other layers above it, only use ESD for identification.

One potential flaw in the site isolation provided by GSE is that it does not apply to DNS servers. The DNS server must be updated according to RG so that they can correctly resolve names to sites. GSE also requires the DNS server to be "two faced" (split horizon), i.e., the server should be able to identify whether a request is from a local or remote host. This is required so that the DNS server can appropriately fill in the STP/RG in the address returned as a part of the response message.

[6] Foreign agents are usually mentioned in the literature, but rarely used in practice due to deployment issues.

[7] IKEv2 has its own mobility and multihoming extensions [44].

3.3.2. LISP

Locator/ID Separation Protocol (LISP) [43] is another approach to split the current Internet namespace into separate identifier and addressing entities that has been proposed. The driving factor behind this proposal is the issue of scalability that is faced by routing and addressing in the current Internet architecture [28]. One of the reasons behind this scalability problem is the steady increase in the number of multihomed sites that cannot be addressed as a part of provider-based aggregated prefixes. Some other causes also hindering aggregation include traffic engineering, end-site renumbering, and dual stacks for IPv4/IPv6 in routers.

In LISP, an Endpoint Identifier (EID) is used for identifying a host and a Routing Locator (RLOC) is used for routing the packets over the Internet. End-hosts work the same as they do currently and the IP addresses assigned to them are referred to as EIDs. EIDs can be assigned independently of the network topology and are used for numbering the end-hosts. On the contrary, RLOCs are topologically assigned to network attachment points of the routers, and are used for routing and forwarding packets.

LISP is based on edge routers to realize the identifier–locator split and tunneling information between the source and destination. LISP can be incrementally deployed by upgrading the software of the edge routers, albeit end-host mobility support requires also upgrading the end-hosts [46].

Each communicating end-site has one or more edge routers which are termed as Ingress Tunnel Routers (ITRs) or Egress Tunnel Routers (ETRs), depending on their functionality. The choice of the ETR and ITR can be flexible. For example, the ETR can be the first hop router at the source, and the ITR can be the last hop router at the destination. Routers on the path forward packets based on EIDs. ITRs and ETRs are responsible for encapsulating and decapsulating data flows into LISP tunnels, with their LISP-specific headers.

When an end-host A with endpoint identifier EID1, within domain abc.com, wishes to communicate with another host B with endpoint identifier EID2 within the domain xyz.com, it performs a simple DNS look up for b.xyz.com to obtain the endpoint identifier of B. It then sends the IP packet destined for B to the default first hop router. This router can be configured as an ITR which performs a EID-to-RLOC mapping to obtain the RLOC (usually the address of the edge router of the destination xyz.com) by sending LISP Map-Request and receiving a Map-Reply message (over UDP).[8] With this mapping, the ITR has a route to the destination edge router and can tunnel packets. It then encapsulates IP packets from the local site into a LISP tunnel and forwards them towards the ETR over the tunnel. The ETR is responsible for finally decapsulating the LISP tunnel and forwarding the IP packet to destination B. The ETR may choose to cache the RLOC of the ITR for packets on the reverse path.

Despite of the overhead due to the increase in packet size, splitting IP addresses into EIDs and RLOCs brings a number of benefits. First, it reduces the size of routing tables in the Default-Free Zone (DFZ). Second, it makes multihoming for sites simpler, which allows sites to use multiple service providers simultaneously. Third, sites can change providers in an easy way.

LISP has also a few drawbacks. First, while LISP supports also end-host mobility, this requires software to be installed at the end-host, which can be challenging from the viewpoint of deployment. Second, LISP requires separate proxies to communicate with non-LISP sites, which might be challenging from the viewpoint of scalability.

[8] Currently, public deployments use LISP Delegated Database Tree [47] as the mapping system.

4. Qualitative analysis

In this section, we present a qualitative analysis of the architectures that have been discussed throughout this paper. The results are summarized in Tables 1 and 2.

In Table 1, the second and third columns (IPv4 and IPv6) indicate the IP version that is supported by the corresponding architecture at the network layer. Some architectures, such as LIN6, require the hosts to use IPv6, while architectures such as TRIAD aim to prolong the lifetime of IPv4 and, thereby, avoid switching to IPv6. The following four columns (Multihoming, Mobility, Site Renumbering and Internet Transparency) elucidate the challenges introduced in Section 2 are addressed by the architecture. The last two columns explain some characteristics of the identifier namespace used in the architecture: the identifier namespace is “disjoint” if it does not overlap with the locator namespace, and it is “structured” if it is based on hierarchical identifiers (i.e., the opposite of “flat” identifiers).

In Table 2, the architectures are typically based either on “tunneling”, “address rewriting” [48] or both as indicated by the second column. It should be noted that the tunneling approach is also referred as “map-and-encap” in the literature. With tunneling, the packets are encapsulated with an extra header: the inner header contains the identifiers and the outer header the locators. An end-host or a middlebox responsible for the identifier–locator split adds the header when the originating host sends the packet. Correspondingly, the header is removed at the destination by the responsible end-host or middlebox. In address rewriting schemes, the responsible nodes translate identifiers to locators when sending and translate locators back to identifiers at the destination. The translation can involve the whole address or just portions of it, such as the prefix. Since the hosts are responsible for translating identifiers, this approach is also referred to as “translation” as shown in the table.

The tunneling approach requires a mapping infrastructure from where to look up the locators corresponding to the identifiers. This approach can result in lost packets, especially when combined with the core-edge separation. When an edge router receives an outgoing packet with the identifiers, it has to look up the corresponding destination locator and, thus, may have to either drop the packet until the look-up is completed or buffer it. As the tunneling scheme adds an extra header, this lowers the Maximum Transfer Unit (MTU), which further increases the probability for fragmentation [2]. Tunneling can also interfere with geolocation based on IP addresses [1]. However, a benefit of the tunneling is that

Table 1 – Namespace related characteristics of the architectures.

Architecture	IPv4	IPv6	Multihoming	Mobility	Site renumbering	Internet transparency	Disjoint	Structured
HIP	✓	✓	✓	✓	✓	✓	✓	
LIN6		✓	✓	✓			✓	✓
MAT	✓	✓		✓				✓
FARA	✓	✓		✓				✓
MILSA	✓	✓	✓	✓			✓	✓
SHIM6		✓	✓					✓
Six/One		✓	✓		✓		✓	✓
ILNP	✓	✓	✓	✓			✓	✓
TRIAD	✓					✓	✓	✓
IPNL	✓		✓	✓	✓	✓		✓
i3	✓	✓	✓	✓	✓		✓	
4+4	✓		✓		✓	✓	✓	✓
NodeID	✓	✓	✓	✓		✓	✓	
NUTSS	✓	✓	✓	✓	✓	✓	✓	✓
HRA	✓	✓	✓	✓	✓	✓		
Mobile IP	✓	✓		✓				✓
GSE		✓	✓		✓		✓	✓
LISP	✓	✓	✓	✓	✓	✓	✓	✓

Table 2 – Deployment related characteristics of the architectures.

Architecture	Method	Deployment	Legacy apps	Infrastructure changes
HIP	Both	Elimination	✓	New FQDN record per host (optional)
LIN6	Rewrite	Elimination		Mapping Agents for nodes, FQDN/MA
MAT	Rewrite	Elimination	✓	IMS for nodes with DNS entry for each IMS
FARA	Rewrite	Elimination		FARA directory service (fDS)
MILSA	Rewrite	Elimination		DNS like names for every node, RZBS servers
SHIM6	Rewrite	Elimination		IPv6 Options
Six/One	Rewrite	Elimination		Changes to edge network routers (Optional)
ILNP	Rewrite	Elimination	✓	FQDN/node, ARP modified for ILNPv4
TRIAD	Rewrite	Gateway	✓	FQDN/Node, WRAP supporting Relay Agents
IPNL	Both	Gateway	✓	DNS name for every node, upgraded Routers
i3	Rewrite	Gateway	✓	i3 servers
4+4	Rewrite	Gateway	✓	DNS name for every node, upgraded NATs
NodeID	Rewrite	Gateway	✓	NodeID Routers for routing in static core
NUTSS	Rewrite	Gateway	✓	P-boxes and M-boxes
HRA	Rewrite	Gateway		LDBRs, IPv6 ext. headers, new IPv4 payload, FQDN/host
Mobile IP	Tunnel	Gateway	✓	Home Agents
GSE	Rewrite	Separation		Two-faced DNS
LISP	Tunnel	Separation	✓	Tunnel Routers

it is stateless as each packet contains all the necessary information to process it.

As an alternative to tunneling, address rewriting/translation typically requires some extra state at the middleboxes. Typically, packets do not have to be dropped or buffered because the translation is known beforehand. Translation does not affect MTU as no extra header is added. It should be noted that some translation schemes, such as Six/One, ILNP and GSE, alter the semantics of IP addresses, for example, by splitting a single address into identifier and locator portions. Such schemes require changes to the application logic as most of them generally treat addresses as opaque tokens without additional semantics [2].

The third column (Deployment) of Table 2 indicates the mechanism used for implementing the identity–locator split by an architecture: the term "elimination" refers to core-edge elimination (end-host based solution), "separation" to core-edge separation (middlebox-based solution) and "gateway" to a hybrid solution (changes both at the end-host and middlebox). All the architectures discussed in this paper use IPv4/IPv6 for forwarding packets in the core, so the core routers are unaffected. As an example of an elimination based approach, HIP requires end-hosts on the Internet to have host identities and the applications on the end-hosts need to utilize these identities instead of IPv4 or IPv6 addresses. On the other hand, architectures such as LISP, which require changes at the middleboxes to perform tunneling of traffic, are abbreviated as separation.[9] Lastly, gateway-based architectures such as MobileIP, require changes to both the end-hosts (for "Home Address" and "Care-of Address", and the ability to handle binding updates) and middleboxes (for "Home Agent").

[9] Strictly speaking, mobility support in LISP requires changes at the end-hosts, so it could also be categorized as a gateway-based solution.

The fourth column (Legacy apps) indicates whether the architecture supports legacy IPv4 applications or not. The fifth column (infrastructure changes) in Table 2 indicates the changes or additional functionality that are required to the current Internet infrastructure. These changes or additions can be new functionality in the network (routers/DNS/NAT), or extensions or new options in existing communication protocols. As an example, LISP requires edge routers in the current Internet architecture to perform additional new functionality for tunneling data from source to destination. On the other hand, SHIM6 requires new IPv6 options for providing multihoming support.

5. Discussion

This paper begins by providing an overview of the challenges such as mobility and multihoming, associated with addressing in the current TCP/IP architecture. It then presents and compares a variety of solutions that have been proposed for these challenges. In his book [49], John Day, re-iterates the problem and states that the current TCP/IP architecture is a "kludge" and that it has survived the aggregation problems only due to Moore's laws through cheap hardware compensating for the bad design. He proposes a counter design based on Salzer's model [50] that proposes the separation of logical ("who") and physical addressing ("where"), but with a subtle correction: there may be multiple routes to a node (Saltzer had missed multihoming in his original design). Basically, John Day advocates the Open Systems Interconnection (OSI) model, which has been commercially used only in Intermediate System to Intermediate System (IS–IS) based routers. However, in this paper, we do not elaborate on his proposed solution and neither do we discuss architectural solutions as Nimrod [51] and Plutarch [52], since a lot of details for these solutions were extremely coarse and many of these solutions are unverified with no current software implementation or simulation results available. We also chose to exclude clean-slate (non-IP based) architectures, such as Layered Naming Architecture [13] and Routing On Flat Labels (ROFL) [53], from our survey because the Internet architecture has ossified [54] in the sense that making changes at the IP layer or below is very hard, as the deployment of IPv6 has proven.

John day also states that HIP and SHIM6 are merely band aids to an already broken architecture and discusses the design of a new protocol with naming architectures. Over the years, we have witnessed that it is non-trivial for any of the architectures to impact the current Internet irrespective of their novelty or the amount of change required.

Levä et al. [55] provide an interesting case study on the adoption barriers that were faced by HIP. The authors state that despite of its extensive feature set, legacy application compatibility and theoretically correct placement in the protocol stack, HIP faced a number of non-technical challenges. For instance, HIP appeared rather late in the market, where other protocols already provided point solutions to specific problems. Thus, it would have been useful for HIP to find a new problem where other solutions would have been less optimal, but such a "killer application" was not discovered. Later, HIP has been proposed to solve more specific problems related to Internet of Things [56], security for vehicular communications [57], SDN security issues [58] and data center naming [59]. The protocol is also in production in two companies (Boeing and Tofino Security), but it has not been adopted Internet wide according to its initial vision.

Based on detailed analysis on HIP, perhaps it can be generalized that protocol adoption should be driven by specific business use scenarios rather than technical improvements. However, this does not mean that protocols should be "cemented" to specific use cases. As described in RFC5218 [60], sometimes protocols are "wildly successful" and are adopted beyond their original use case, which means that protocols should be designed with some modularity in mind. The RFC states also two other success factors for protocol deployment. First, early adopter should get the benefits. This usually implies that the protocol is going to face adoption difficulties if the early adopters have to deploy additional infrastructure before obtaining any benefit, which we have also analyzed for specific protocols in this paper. Avoiding infrastructural changes is a nice generalization, and the RFC mentions Secure SHell (SSH) servers as an example of avoiding infrastructural changes altogether. At the other extreme, many of the protocols described in the paper require new infrastructure to be deployed (LIN6, MAT, FARA, MILSA, TRIAD, i3, NodeID, NUTSS, HRA, Mobile IP). A middle ground is found in the remaining protocols: upgrades in end-hosts are required in HIP, SHIM6 and ILNP (and also LISP when mobility is employed); upgrades in middleboxes are required in GSE, ILNP, Six/One, IPNL, LISP, 4+4; extensions to DNS are required in HIP, GSE, ILNP, 4+4, IPNL.

It should be noted that many of the network-layer solutions for mobility and multihoming remain marginally adopted and deployed. Thus, many application layer specific solutions have emerged to tolerate mobility at some level. For instance, many web browsers, including Mozilla Firefox, support pausing and resuming of downloads. Email client software such as Mozilla Thunderbird can tolerate disconnectivity and automatically reconnect to the email server. Internet telephony as supported by Session Initiation Protocol (SIP) includes session mobility [61]. Most web services identify HTTP sessions with browser cookies and, thus, can tolerate IP address changes for non-streaming applications.[10] In web browsers, the use of persistent HTTP, i.e., the reuse of the same TCP connection is less common nowadays [63], perhaps to tolerate mobility better. For multihoming, TCP has been extended to support it in the form of multipath TCP [64], and Apple has been one of the first commercial vendors to adopt it. Finally, more generic application layer solutions based on, e.g., libraries [65], by introducing a new session layer [66] and overlays [67] have also emerged but have not yet been embraced by application developers.

The aim of this paper was to provide an overview of the addressing problems prevalent on the Internet and present a survey on a number of existing solutions. We hope that our work benefits the networking research community when new architectures or protocols are being designed and proposed as we have categorized some technical properties and explained

[10] According to some measurements [62], HTTP traffic can amount to nearly 60% of Internet traffic.

some deployment related trade-offs from the prior art. To our knowledge, our survey on identifier–locator split architecture is more extensive than others; earlier surveys either looked at a particular problem such as mobility [68] or multihoming [69] or did not encompassed only a few architectural solutions [70].

Acknowledgments

We would like to express our gratitude for Tuomas Aura, who provided useful input to our work. This work was supported by TEKES as part of the Internet of Things program of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business).

REFERENCES

[1] D. Thaler, Evolution of the IP model, RFC 6250 (Informational), 2011.
[2] D. Meyer, L. Zhang, K. Fall, Report from the IAB workshop on routing and addressing, RFC 4984 (Informational), 2007.
[3] P. Faltstrom, G. Huston, A survey of Internet identities: draft-iab-identities-02, Internet Draft, Internet Engineering Task Force, 2004.
[4] B. Carpenter, R. Atkinson, H. Flinck, Renumbering still needs work, RFC 5887 (Informational), 2010.
[5] J. Manner, M. Kojo, Mobility related terminology, RFC 3753 (Informational), 2004.
[6] C. Ng, P. Thubert, M. Watari, F. Zhao, Network mobility route optimization problem statement, RFC 4888 (Informational), 2007.
[7] J. Ylitalo, Secure mobility at multiple granularity levels over heterogeneous datacom networks, Aalto University, 2008.
[8] T. Li, Recommendation for a routing architecture, RFC 6115 (Informational), 2011.
[9] T. Bates, Y. Rekhter, Scalable support for multi-homed multi-provider connectivity, RFC 2260 (Informational), 1998.
[10] B. Carpenter, Internet transparency, RFC 2775 (Informational), 2000.
[11] L. Popa, A. Ghodsi, I. Stoica, HTTP as the narrow waist of the future Internet, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX, ACM, New York, NY, USA, 2010, pp. 6:1–6:6.
[12] T. Li, Design goals for scalable Internet routing, RFC 6227 (Informational), 2011.
[13] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, M. Walfish, A layered naming architecture for the Internet, SIGCOMM Comput. Commun. Rev. 34 (2004) 343–352.
[14] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, I. Stoica, A data-oriented (and beyond) network architecture, ACM SIGCOMM Comput. Commun. Rev. 37 (2007) 181–192.
[15] R. Moskowitz, P. Nikander, Host Identity Protocol (HIP) architecture, RFC 4423 (Informational), 2006.
[16] P. Nikander, J. Wall, J. Ylitalo, Integrating security, mobility, and multi-homing in a HIP way, in: Proceedings of Network and Distributed Systems Security Symposium, Internet Society, San Diego, California, 2003, pp. 87–99.
[17] T. Henderson, A. Gurtov, The Host Identity Protocol (HIP) experiment report, RFC 6538 (Informational), 2012.
[18] M. Kunishi, M. Ishiyama, K. Uehara, H. Esaki, F. Teraoka, LIN6: A new approach to mobility support in IPv6, in: Proc. of the Third International Symposium on Wireless Personal Multimedia Communications.
[19] R. Inayat, R. Aibara, K. Nishimura, T. Fujita, Y. Nomura, K. Maeda, MAT: An end-to-end mobile communication architecture with seamless IP handoff support for the next generation Internet, in: Web and Communication Technologies and Internet-Related Social Issues HSI 2003, 2003, Vol. 2713, pp. 465–475.
[20] D. Johnson, C. Perkins, J. Arkko, Mobility support in IPv6, RFC 3775 (Proposed Standard), 2004. Obsoleted by RFC 6275.
[21] D. Clark, R. Braden, A. Falk, V. Pingali, FARA: Reorganizing the addressing architecture, in: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, ACM, 2003, pp. 313–321.
[22] J. Pan, S. Paul, R. Jain, M. Bowman, MILSA: A mobility and multihoming supporting identifier locator split architecture for naming in the next generation Internet, in: Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE, 2008, IEEE, pp. 1–6.
[23] E. Nordmark, M. Bagnulo, Shim6: Level 3 multihoming shim protocol for IPv6, Internet Draft, Internet Engineering Task Force, 2009.
[24] J. Arkko, I. van Beijnum, Failure detection and locator pair exploration protocol for IPv6 multihoming, Internet Draft, Internet Engineering Task Force, 2009.
[25] M. Bagnulo, Hash-Based Addresses (HBA), Internet Draft, Internet Engineering Task Force, 2009.
[26] T. Aura, Cryptographically generated addresses, Internet Draft, Internet Engineering Task Force, 2005.
[27] C. Vogt, Six/One—A solution for routing and addressing in IPv6: draft-vogt-rrg-six-one-02, Internet Draft, Internet Engineering Task Force, 2009.
[28] T. Narten, On the scalability of Internet routing: draft-narten-radir-problem-statement-05.txt, Internet Draft, Internet Engineering Task Force, 2010.
[29] R. Atkinson, S. Bhatti, Identifier-Locator Network Protocol (ILNP) architectural description, RFC 6740 (Experimental), 2012.
[30] D.R. Cheriton, M. Gritter, TRIAD: A scalable deployable NAT-based Internet architecture, Technical Report, 2000, Citeseer.
[31] P. Francis, R. Gummadi, IPNL: A NAT-extended Internet architecture, ACM SIGCOMM Comput. Commun. Rev. 31 (2001) 69–80.
[32] S. Paul, J. Pan, R. Jain, A survey of naming systems: Classification and analysis of the current schemes using a new naming reference model, Technical Report, 2009, Citeseer.
[33] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, Internet indirection infrastructure, ACM SIGCOMM Comput. Commun. Rev. 32 (4) (2002) 73–86.
[34] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications, ACM SIGCOMM Comput. Commun. Rev. 31 (2001) 149–160.
[35] P. Nikander, J. Arkko, B. Ohlman, Host Identity Indirection Infrastructure (Hi3): draft-nikander-hiprg-hi3-00, Internet Draft, Internet Engineering Task Force, 2004.
[36] Z. Turanyi, A. Valkó, A. Campbell, 4 + 4: An architecture for evolving the Internet address space back toward transparency, ACM SIGCOMM Comput. Commun. Rev. 33 (2003) 43–54.
[37] B. Ahlgren, J. Arkko, L. Eggert, J. Rajahalme, A node identity internetworking architecture, in: Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM, IEEE, 2006, pp. 1–6.
[38] L. Mathy, L. Iannone, LISP-DHT: Towards a DHT to map identifiers onto locators, in: Proceedings of the 2008 ACM CoNEXT Conference, ACM, 2008, p. 61.
[39] B. Ford, Unmanaged Internet protocol: Taming the edge network management crisis, ACM SIGCOMM Comput. Commun. Rev. 34 (2004) 93–98.

[40] S. Guha, P.P. Francis, An end-middle-end approach to connection establishment, in: Proceedings of SIGCOMM 2007, pp. 193–204.
[41] X. Xu, D. Guo, Hierarchical Routing Architecture (HRA), in: Next Generation Internet Networks NGI, IEEE, 2008, pp. 92–99.
[42] C. Perkins, IP mobility support for IPv4, revised, RFC 5944 (Proposed Standard), 2010.
[43] C. Perkins, D. Johnson, J. Arkko, Mobility support in IPv6, RFC 6275 (Proposed Standard), 2011.
[44] P. Eronen, IKEv2 mobility and multihoming protocol (MOBIKE), RFC 4555 (Proposed Standard), 2006.
[45] M. O'Dell, GSE—An alternate addressing architecture for IPv6, Internet Draft, Internet Engineering Task Force, 1997.
[46] D. Saucez, L. Iannone, O. Bonaventure, D. Farinacci, Designing a deployable Internet: The locator/identifier separation protocol, IEEE Internet Comput. 16 (2012) 14–21.
[47] V. Fuller, D. Lewis, V. Ermagan, A. Jain, LISP delegated database tree, Internet Draft, Internet Engineering Task Force, 2014.
[48] G. Huston, Architectural approaches to multi-homing for IPv6, RFC 4177 (Informational), 2005.
[49] J. Day, Patterns in Network Architecture: A Return to Fundamentals, 2008.
[50] J. Saltzer, On the naming and binding of network destinations, Internet Draft, Internet Engineering Task Force, 1993.
[51] I. Castineyra, N. Chiappa, M. Steenstrup, The Nimrod routing architecture, RFC 1992 (Informational), 1996.
[52] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, A. Warfield, Plutarch: An argument for network pluralism, ACM SIGCOMM Comput. Commun. Rev. 33 (2003) 258–266.
[53] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, ROFL: Routing on flat labels, ACM SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 363–374.
[54] S. Akhshabi, C. Dovrolis, The evolution of layered protocol stacks leads to an hourglass-shaped architecture, SIGCOMM Comput. Commun. Rev. 41 (2011) 206–217.
[55] T. Levä, M. Komu, A. Keränen, S. Luukkainen, Adoption barriers of network layer protocols: The case of host identity protocol, Comput. Netw. 57 (2013) 2218–2232.
[56] R. Hummen, J. Hiller, M. Henze, K. Wehrle, Slimfit—A HIP DEX compression layer for the IP-based Internet of things, in: 9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2013, Lyon, France, October 7–9, 2013, pp. 259–266.
[57] S. Soderi, H. Viittala, J. Saloranta, M. Hamalainen, J. Iinatti, A. Gurtov, Security of Wi-Fi on-board intra-vehicular communication: Field trials of tunnel scenario, in: 13th International Conference on ITS Telecommunications, ITST, pp. 278–283.
[58] S. Namal, I. Ahmad, A. Gurtov, M. Ylianttila, Enabling secure mobility with OpenFlow, in: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1–5.
[59] M. Komu, M. Sethi, R. Mallavarapu, H. Oirola, R. Khan, S. Tarkoma, Secure networking for virtual machines in the cloud, in: In the Proceedings of IEEE Cluster 2012, IEEE Computer Society, 2012, pp. 88–96.
[60] D. Thaler, B. Aboba, What makes for a successful protocol? RFC 5218 (Informational), 2008.
[61] R. Shacham, H. Schulzrinne, S. Thakolsri, W. Kellerer, Session Initiation Protocol (SIP) session mobility, RFC 5631 (Informational), 2009.
[62] G. Maier, A. Feldmann, V. Paxson, M. Allman, On dominant characteristics of residential broadband Internet traffic, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, IMC'09, ACM, New York, NY, USA, 2009, pp. 90–102.
[63] T. Callahan, M. Allman, V. Paxson, A longitudinal view of HTTP traffic, in: Proceedings of the 11th International Conference on Passive and Active Measurement, PAM'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 222–231.
[64] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, TCP extensions for multipath operation with multiple addresses, RFC 6824 (Experimental), 2013.
[65] V.C. Zandy, B.P. Miller, Reliable network connections, in: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, MobiCom'02, ACM, New York, NY, USA, 2002, pp. 95–106.
[66] A.C. Snoeren, H. Balakrishnan, M.F. Kaashoek, Reconsidering Internet mobility, in: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, HOTOS'01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 41–46.
[67] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, The design and implementation of an intentional naming system, in: Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles, SOSP'99, ACM, New York, NY, USA, 1999, pp. 186–201.
[68] A. Mungur, C. Edwards, Scalability of the locator identity split mapping infrastructure to support end-host mobility, in: IEEE 35th Conference on Local Computer Networks, LCN, 2010, pp. 352–355.
[69] P. Savola, T. Chown, A survey of IPv6 site multihoming proposals, in: Proceedings of the 8th International Conference of Telecommunications, ConTEL, pp. 41–48.
[70] B.M. Martins, A.M. Alberti, Host identification and location decoupling: A comparison of approaches, in: International Workshop on Telecommunications, 2011.