

INTERNET-DRAFT
Intended Status: Informational
Expires: September 14, 2017

Tom Herbert
Quantonium
Petr Lapukhov
Facebook
March 13, 2017

Identifier-locator addressing for IPv6
draft-herbert-nvo3-ila-04

Abstract

This specification describes identifier-locator addressing (ILA) for IPv6. Identifier-locator addressing differentiates between location and identity of a network node. Part of an address expresses the immutable identity of the node, and another part indicates the location of the node which can be dynamic. Identifier-locator addressing can be used to efficiently implement overlay networks for network virtualization as well as solutions for use cases in mobility.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	4
1.1	Terminology	4
2	Architectural overview	6
2.1	Addressing	6
2.2	Network topology	6
2.3	Translations and mappings	7
2.4	ILA routing	8
3	Address formats	9
3.1	ILA address format	9
3.2	Locators	9
3.3	Identifiers	9
3.3.1	Checksum neutral-mapping format	10
3.3.2	Identifier types	10
3.3.2.1	Interface identifiers	10
3.3.2.2	Locally unique identifiers	11
3.3.2.3	Virtual networking identifiers for IPv4	11
3.3.2.4	Virtual networking identifiers for IPv6 unicast	12
3.3.2.5	Virtual networking identifiers for IPv6 multicast	13
3.4	Standard identifier representation addresses	14
3.4.1	SIR for locally unique identifiers	15
3.4.2	SIR for virtual addresses	15
3.4.3	SIR domains	16
4	Operation	16
4.1	Identifier to locator mapping	16
4.2	Address translations	16
4.2.1	SIR to ILA address translation	16
4.2.2	ILA to SIR address translation	17
4.3	Virtual networking operation	17
4.3.1	Crossing virtual networks	18
4.3.2	IPv4/IPv6 protocol translation	18
4.4	Transport layer checksums	18
4.4.1	Checksum-neutral mapping	19
4.4.2	Sending an unmodified checksum	20
4.5	Address selection	20
4.6	Duplicate identifier detection	20

Herbert

Expires September 14, 2017

[Page 2]

4.7	ICMP error handling	21
4.7.1	Handling ICMP errors by ILA capable hosts	21
4.7.2	Handling ICMP errors by non-ILA capable hosts	21
4.8	Multicast	22
5	Motivation for ILA	22
5.1	Use cases	22
5.1.1	Multi-tenant virtualization	22
5.1.2	Datacenter virtualization	23
5.1.3	Device mobility	23
5.2	Alternative methods	24
5.2.1	ILNP	24
5.2.2	Flow label as virtual network identifier	24
5.2.3	Extension headers	25
5.2.4	Encapsulation techniques	25
6	IANA Considerations	26
7	References	27
7.1	Normative References	27
7.2	Informative References	27
8	Acknowledgments	28
	Appendix A: Communication scenarios	29
A.1	Terminology for scenario descriptions	29
A.2	Identifier objects	30
A.3	Reference network for scenarios	30
A.4	Scenario 1: Object to task	31
A.5	Scenario 2: Object to Internet	31
A.6	Scenario 3: Internet to object	31
A.7	Scenario 4: Tenant system to service	32
A.8	Scenario 5: Object to tenant system	32
A.9	Scenario 6: Tenant system to Internet	33
A.10	Scenario 7: Internet to tenant system	33
A.11	Scenario 8: IPv4 tenant system to object	33
A.12	Tenant to tenant system in the same virtual network	34
A.12.1	Scenario 9: TS to TS in the same VN using IPV6	34
A.12.2	Scenario 10: TS to TS in same VN using IPv4	34
A.13	Tenant system to tenant system in different virtual networks	34
A.13.1	Scenario 11: TS to TS in different VNs using IPV6	34
A.13.2	Scenario 12: TS to TS in different VNs using IPv4	35
A.13.3	Scenario 13: IPv4 TS to IPv6 TS in different VNs	35
	Appendix B: unique identifier generation	36
B.1	Globally unique identifiers method	36
B.2	Universally Unique Identifiers method	36
	Appendix C: Datacenter task virtualization	37
C.1	Address per task	37
C.2	Job scheduling	37
C.3	Task migration	38
C.3.1	Address migration	38
C.3.2	Connection migration	39

Herbert

Expires September 14, 2017

[Page 3]

1 Introduction

This specification describes the address formats, protocol operation, and communication scenarios of identifier-locator addressing (ILA). In identifier-locator addressing, an IPv6 address is split into a locator and an identifier component. The locator indicates the topological location in the network for a node, and the identifier indicates the node's identity which refers to the logical or virtual node in communications. Locators are routable within a network, but identifiers typically are not. An application addresses a peer destination by identifier. Identifiers are mapped to locators for transit in the network. The on-the-wire address is composed of a locator and an identifier: the locator is sufficient to route the packet to a physical host, and the identifier allows the receiving host to translate and forward the packet to the addressed application.

With identifier-locator addressing network virtualization and addressing for mobility can be implemented in an IPv6 network without any additional encapsulation headers. Packets sent with identifier-locator addresses look like plain unencapsulated packets (e.g. TCP/IP packets). This method is transparent to the network, so protocol specific mechanisms in network hardware work seamlessly. These mechanisms include hash calculation for ECMP, NIC large segment offload, checksum offload, etc.

Many of the concepts for ILA are adapted from Identifier-Locator Network Protocol (ILNP) ([\[RFC6740\]](#), [\[RFC6741\]](#)) which defines a protocol and operations model for identifier-locator addressing in IPv6.

[Section 5](#) provides a motivation for ILA and comparison of ILA with alternative methods that achieve similar functionality.

1.1 Terminology

ILA	Identifier-locator addressing.
ILA router	A network node that performs ILA translation and forwarding of translated packets.
ILA host	An end host that is capable of performing ILA translations on transmit or receive.
ILA node	A network node capable of performing ILA translations. This can be an ILA router or ILA host.
Locator	A network prefix that routes to a physical host.

Herbert

Expires September 14, 2017

[Page 4]

Locators provide the topological location of an addressed node. In ILA locators are a sixty-four bit prefixes.

Identifier A number that identifies an addressable node in the network independent of its location. ILA identifiers are sixty-four bit values.

ILA address

An IPv6 address composed of a locator (upper sixty-four bits) and an identifier (low order sixty-four bits).

SIR Standard identifier representation.

SIR prefix A sixty-four bit network prefix used to identify a SIR address.

SIR address

An IPv6 address composed of a SIR prefix (upper sixty-four bits) and an identifier (lower sixty-four bits). SIR addresses are visible to applications and provide a means to address nodes independent of their location.

SIR domain A unique identifier namespace defined by a SIR prefix. Each SIR prefix defines a SIR domain.

ILA translation

The process of translating the upper sixty-four bits of an IPv6 address. Translations may be from a SIR prefix to a locator or a locator to a SIR prefix.

Virtual address

An IPv6 or IPv4 address that resides in the address space of a virtual network. Such addresses may be translated to SIR addresses as an external representation of the address outside of the virtual network, or they may be translated to ILA addresses for transit over an underlay network.

Topological address

An address that refers to a non-virtual node in a network topology. These address physical hosts in a network.

Herbert

Expires September 14, 2017

[Page 5]

2 Architectural overview

Identifier-locator addressing allows a data plane method to implement network virtualization without encapsulation and its related overheads. The service ILA provides is effectively layer 3 over layer 3 network virtualization (IPv4 or IPv6 over IPv6).

2.1 Addressing

ILA performs translations on IPv6 address. There are two types of addresses introduced for ILA: ILA addresses and SIR addresses.

ILA addresses are IPv6 addresses that are composed of a locator (upper sixty-four bits) and an identifier (low order sixty-four bits). The identifier serves as the logical addresses of a node, and the locator indicates the location of the node on the network.

A SIR address (standard identifier representation) is an IPv6 address that contains an identifier and an application visible SIR prefix. SIR addresses are visible to the application and can be used as connection endpoints. When a packet is sent to a SIR address, an ILA router or host overwrites the SIR prefix with a locator corresponding to the identifier. When a peer ILA node receives the packet, the locator is overwritten with the original SIR prefix before delivery to the application. In this manner applications only see SIR addresses, they do not have visibility into ILA addresses.

ILA translations can transform addresses from one type to another. In network virtualization virtual addresses can be translated into ILA and SIR addresses, and conversely ILA and SIR addresses can be translated to virtual addresses.

2.2 Network topology

ILA nodes are nodes in the network that perform ILA translations. An ILA router is a node that performs ILA address translation and packet forwarding to implement overlay network functionality. ILA routers perform translations on packets sent by end nodes for transport across an underlay network. Packets received by ILA routers on the underlay network have their addresses reversed translated for reception at an end node. An ILA host is an end node that implements ILA functionality for transmitting or receiving packets.

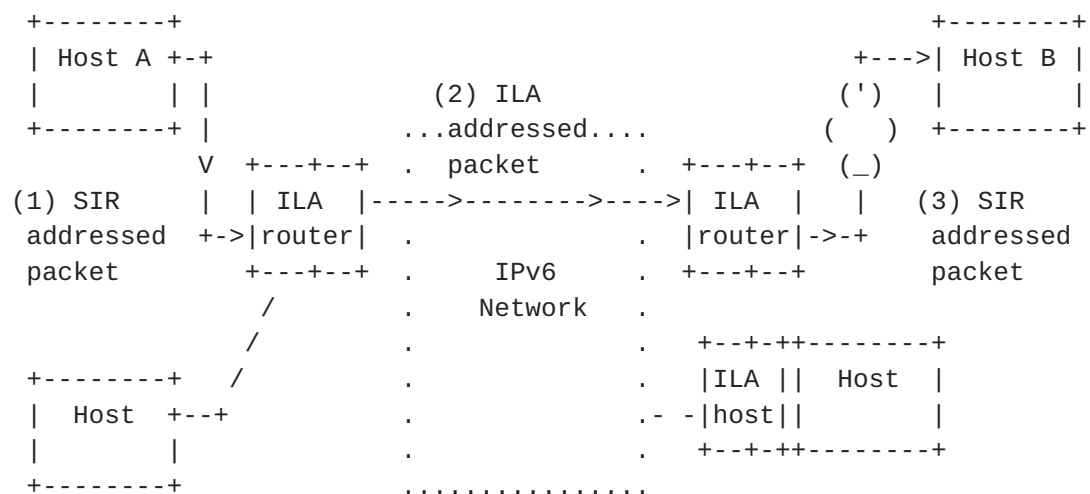
ILA nodes are responsible for transit of packets over an underlay network. On ingress to an ILA node (host or router) the virtual or SIR address of a destination is translated to an ILA address. At the a peer ILA node, the reverse translation is performed before handing packets to an application.

Herbert

Expires September 14, 2017

[Page 6]

The figure below provides an example topology using ILA. ILA translations performed in one direction between Host A and Host B are denoted. Host A sends a packet with a destination SIR address (step (1)). An ILA router in the path translates the SIR address to an ILA address with a locator set to Host B, referring to the location of the node indicated by the identifier in the SIR address. The packet is forwarded over the network and delivered to a peer ILA node (step 2). The peer ILA node, in this case another ILA router, translates the destination address back to a SIR address and forwards to the final destination (step 3).



2.3 Translations and mappings

Address translation is the mechanism employed by ILA. Logical or virtual addresses are translated to topological IPv6 addresses for transport to the proper destination. Translation occurs in the upper sixty-four bits of an address, the low order sixty-four bits contains an identifier that is immutable and is not used to route a packet.

Each ILA node maintains a mapping table. This table maps identifiers to locators. The mappings are dynamic as nodes with identifiers can be created, destroyed, or migrated between physical hosts. Mappings are propagated amongst ILA routers or hosts in a network using mapping propagation protocols (mapping propagation protocols will be described in other specifications).

Identifiers are not statically bound to a host on the network, and in fact their binding (or location) may change. This is the basis for network virtualization and address migration. An identifier is mapped to a locator at any given time, and a set of identifier to locator mappings is propagated throughout a network to allow communications. The mappings are kept synchronized so that if an identifier migrates to a new physical host, its identifier to locator mapping is updated.

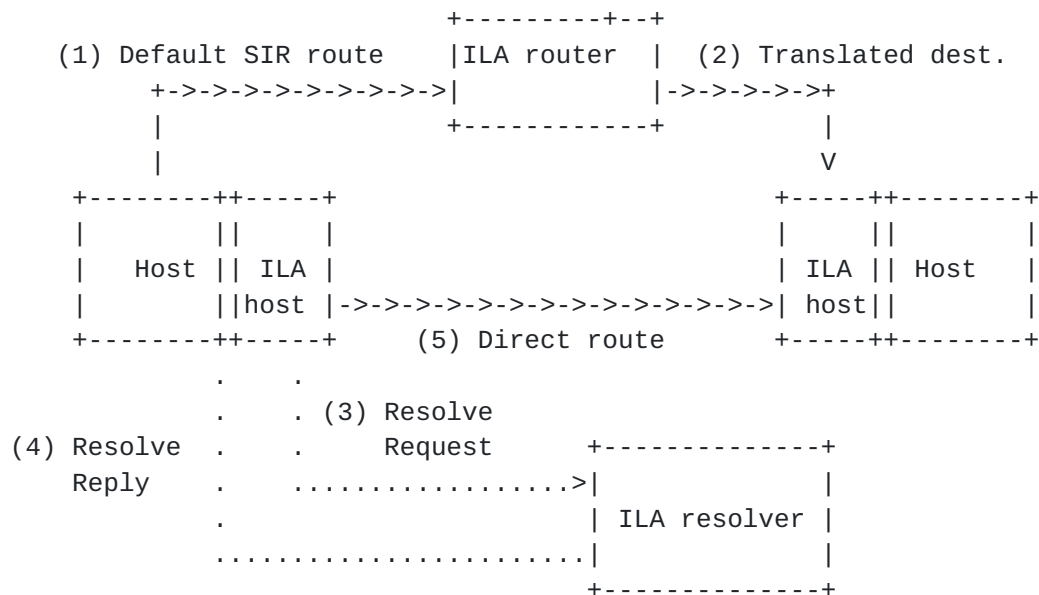
Herbert

Expires September 14, 2017

[Page 7]

2.4 ILA routing

ILA is intended to be sufficiently lightweight so that all the hosts in a network could potentially send and receive ILA addressed packets. In order to scale this model and allow for hosts that do not participate in ILA, a routing topology may be applied. A simple routing topology is illustrated below.



An ILA router can be addressed by an "anycast" SIR prefix so that it receives packets sent on the network with SIR addresses. When an ILA router receives a SIR addressed packet (step (1) in the diagram) it will perform the ILA translation and send the ILA addressed packet to the destination ILA node (step (2)).

If a sending host is ILA capable the triangular routing can be eliminated by performing an ILA resolution protocol. This entails the host sending an ILA resolve request that specifies the SIR address to resolve (step (3) in the figure). An ILA resolver can respond to a resolver request with the identifier to locator mapping (step (4)). Subsequently, the ILA host can perform ILA translation and send directly to the destination specified in the locator (step (5) in the figure). The ILA resolution protocol will be specified in a companion document.

In this model an ILA host maintains a cache of identifier mappings for identifiers that it is currently communicating with. ILA routers are expected to maintain a complete list of identifier to locator mappings within the SIR domains that they service.

Herbert

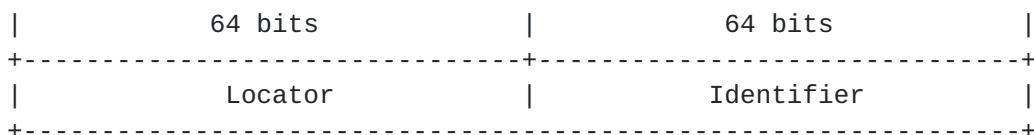
Expires September 14, 2017

[Page 8]

3 Address formats

3.1 ILA address format

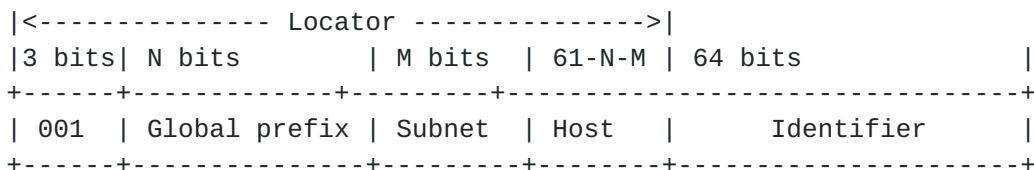
An ILA address is composed of a locator and an identifier where each occupies sixty-four bits (similar to the encoding in ILNP [[RFC6741](#)]).



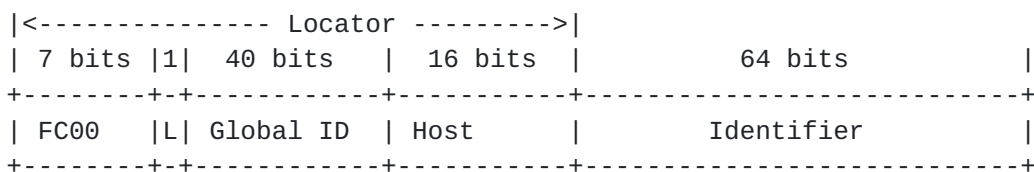
3.2 Locators

Locators are routable network address prefixes that create topological addresses for physical hosts within the network. They may be assigned from a global address block [[RFC3587](#)], or be based on unique local IPv6 unicast addresses as described in [[RFC4193](#)].

The format of an ILA address with a global unicast locator is:

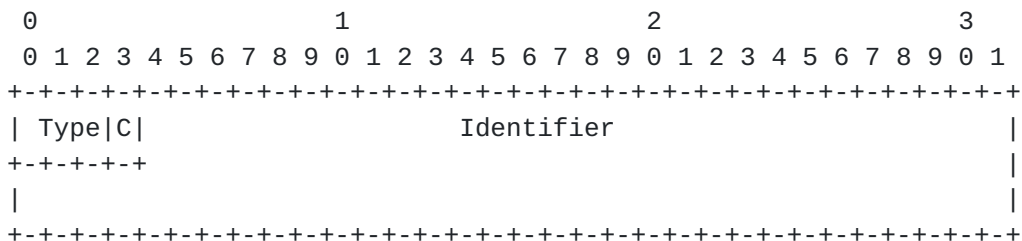


The format of an ILA address with a unique local IPv6 unicast locator is:



3.3 Identifiers

The format of an ILA identifier is:



Herbert

Expires September 14, 2017

[Page 9]

Fields are:

- o Type: Type of the identifier (see [section 3.3.2](#)).
- o C: The C-bit. This indicates that checksum-neutral mapping applied (see [section 3.3.1](#)).
- o Identifier: Identifier value.

3.3.1 Checksum neutral-mapping format

If the C-bit is set the low order sixteen bits of an identifier contain the adjustment for checksum-neutral mapping (see [section 4.4.1](#) for description of checksum-neutral mapping). The format of an identifier with checksum neutral mapping is:

[illegible]

3.3.2 Identifier types

Identifier types allow standard encodings for common uses of identifiers. Defined identifier types are:

- ```
0: interface identifier
1: locally unique identifier
2: virtual networking identifier for IPv4 address
3: virtual networking identifier for IPv6 unicast address
4: virtual networking identifier for IPv6 multicast address
5-7: Reserved
```

### 3.3.2.1 Interface identifiers

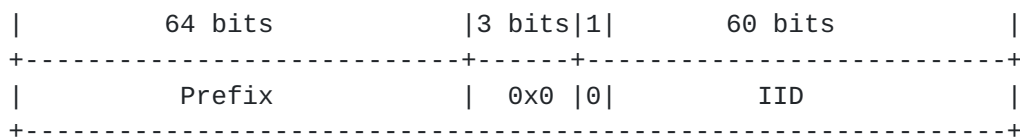
The interface identifier type indicates a plain local scope interface identifier. When this type is used the address is a normal IPv6 address without identifier-locator semantics. The purpose of this type is to allow normal IPv6 addresses to be defined within the same networking prefix as ILA addresses. Type bits and C-bit MUST be zero.

Herbert

Expires September 14, 2017

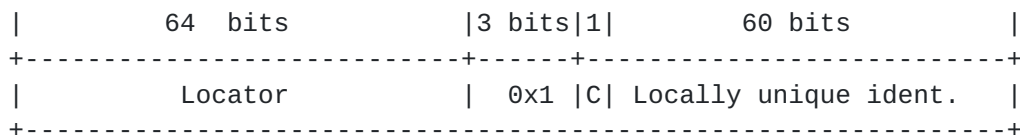
[Page 10]

The format of an ILA interface identifier address is:

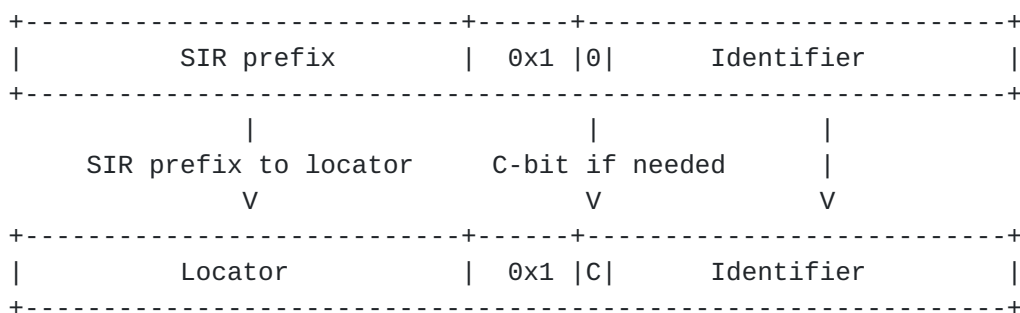


### [3.3.2.2](#) Locally unique identifiers

Locally unique identifiers (LUI) can be created for various addressable objects within a network. These identifiers are in a flat sixty bit space and must be unique within a SIR domain (unique within a site for instance). To simplify administration, hierarchical allocation of locally unique identifiers may be performed. The format of an ILA address with locally unique identifiers is:

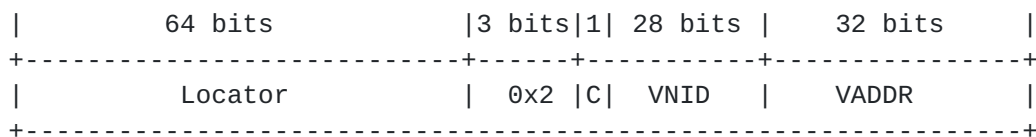


The figure below illustrates the translation from SIR address to an ILA address as would be performed when a node sends to a SIR address. Note the low order 16 bits of the identifier may be modified as the checksum-neutral adjustment. The reverse translation of ILA address to SIR address is symmetric.



### [3.3.2.3](#) Virtual networking identifiers for IPv4

This type defines a format for encoding an IPv4 virtual address and virtual network identifier within an identifier. The format of an ILA address for IPv4 virtual networking is:



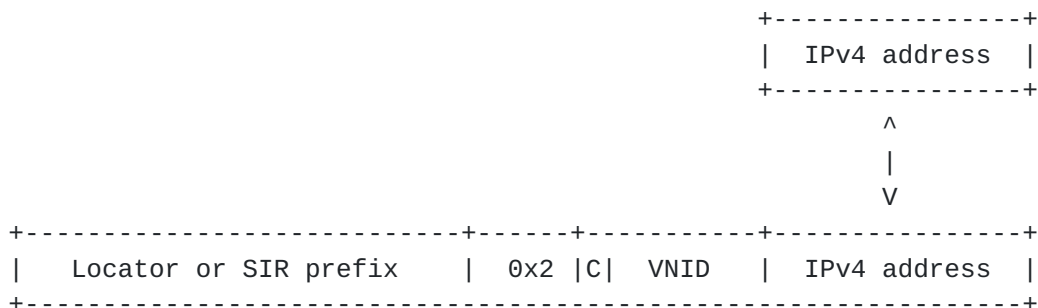
Herbert

Expires September 14, 2017

[Page 11]

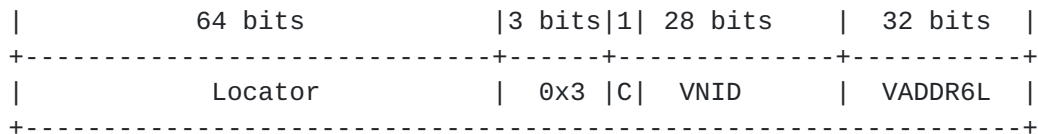
VNID is a virtual network identifier and VADDR is a virtual address within the virtual network indicated by the VNID. The VADDR can be an IPv4 unicast or multicast address, and may often be in a private address space (i.e. [[RFC1918](#)]) used in the virtual network.

Translating a virtual IPv4 address into an ILA or SIR address and the reverse translation are straight forward. Note that the low order 16 bits of the IPv6 address may be modified as the checksum-neutral adjustment and that this translation implies protocol translation when sending IPv4 packets over an ILA IPv6 network.



#### [3.3.2.4](#) Virtual networking identifiers for IPv6 unicast

In this format, a virtual network identifier and virtual IPv6 unicast address are encoded within an identifier. To facilitate encoding of virtual addresses, there is a unique mapping between a VNID and a ninety-six bit prefix of the virtual address. The format an IPv6 unicast encoding with VNID in an ILA address is:



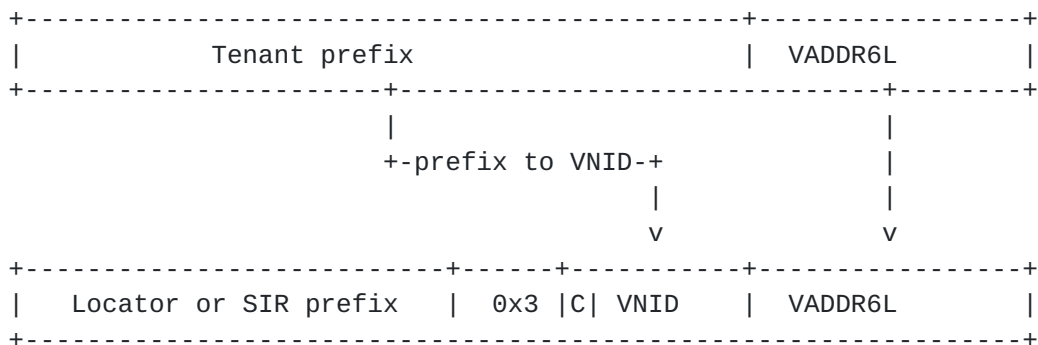
VADDR6L contains the low order 32 bits of the IPv6 virtual address. The upper 96 bits of the virtual address are inferred from the VNID to prefix mapping. Note that for ILA translations the low order sixteen of the VADDR6L may be modified for checksum-neutral adjustment.

The figure below illustrates encoding a tenant IPv6 virtual unicast address into a ILA or SIR address.

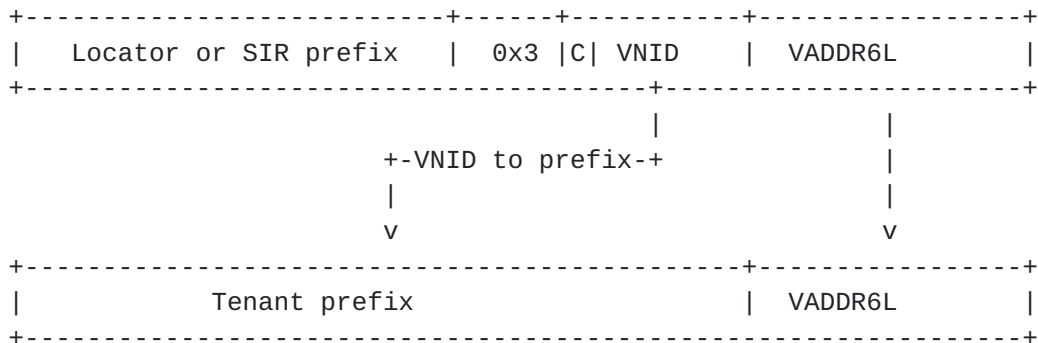
Herbert

Expires September 14, 2017

[Page 12]

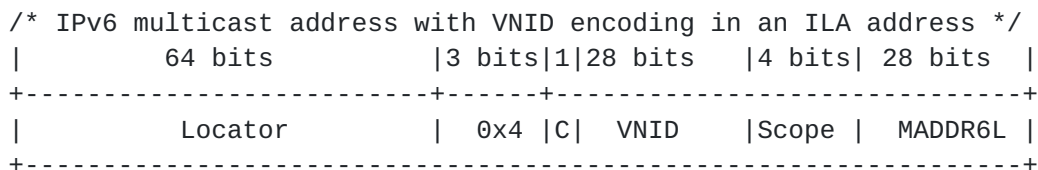


This encoding is reversible, given an ILA address, the virtual address visible to the tenant can be deduced:



### 3.3.2.5 Virtual networking identifiers for IPv6 multicast

In this format, a virtual network identifier and virtual IPv6 multicast address are encoded within an identifier.



This format encodes an IPv6 multicast address in an identifier. The scope indicates multicast address scope as defined in [\[RFC7346\]](#). MADDR6L is the low order 28 bits of the multicast address. The full multicast address is thus:

ff0<Scope>::<MADDR6L high 12 bits>:<MADDR6L low 16 bits>

And so can encode multicast addresses of the form:

ff0X::0 to ff0X::0fff:ffff

The figure below illustrates encoding a tenant IPv6 virtual multicast

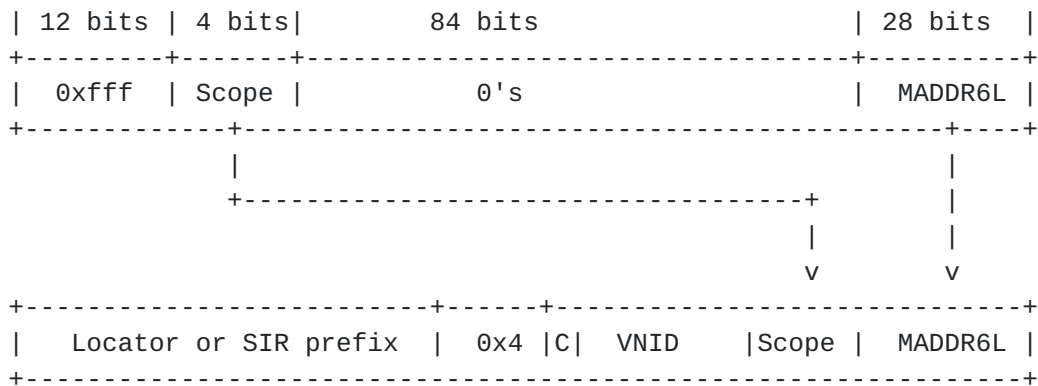


Herbert

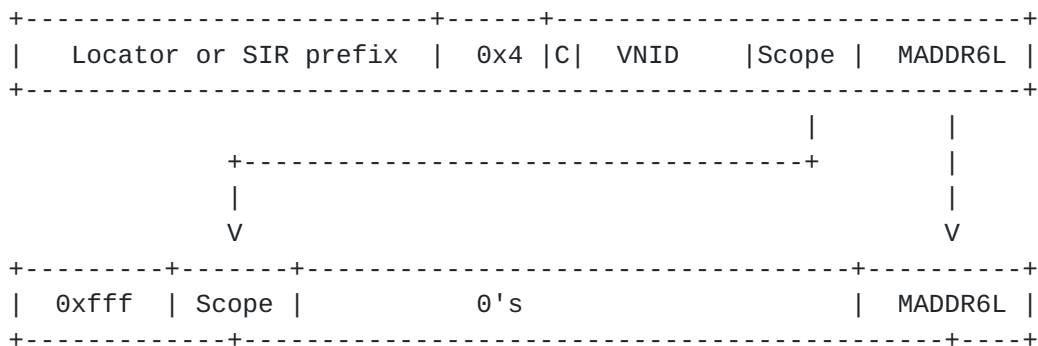
Expires September 14, 2017

[Page 13]

address in an ILA or SIR address. Note that low order sixteen bits of MADDR6L may be modified to be the checksum-neutral adjustment.



This translation is reversible:



### 3.4 Standard identifier representation addresses

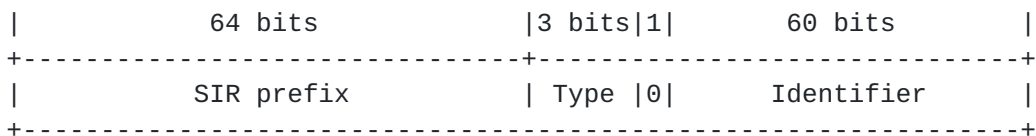
An identifier identifies objects or nodes in a network. For instance, an identifier may refer to a specific host, virtual machine, or tenant system. When a host initiates a connection or sends a packet, it uses the identifier to indicate the peer endpoint of the communication. The endpoints of an established connection context also referenced by identifiers. It is only when the packet is actually being sent over a network that the locator for the identifier needs to be resolved.

In order to maintain compatibility with existing networking stacks and applications, identifiers are encoded in IPv6 addresses using a standard identifier representation (SIR) address. A SIR address is a combination of a prefix which occupies what would be the locator portion of an ILA address, and the identifier in its usual location. The format of a SIR address is:

Herbert

Expires September 14, 2017

[Page 14]



The C-bit (checksum-neutral mapping) MUST be zero for a SIR address. Type may be any identifier type except zero (interface identifiers)

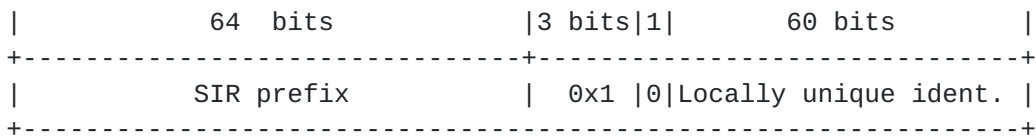
A SIR prefix may be site-local, or globally routable. A globally routable SIR prefix facilitates connectivity between hosts on the Internet and ILA nodes. A gateway between a site's network and the Internet can translate between SIR prefix and locator for an identifier. A network may have multiple SIR prefixes where each prefix defines a unique identifier space.

Locators MUST only be associated with one SIR prefix. This ensures that if a translation from a SIR address to an ILA address is performed when sending a packet, the reverse translation at the receiver yields the same SIR address that was seen at the transmitter. This also ensures that a reverse checksum-neutral mapping can be performed at a receiver to restore the addresses that were included in a pseudo header for setting a transport checksum.

A standard identifier representation address can be used as the externally visible address for a node. This can be used throughout the network, returned in DNS AAAA records [[RFC3363](#)], used in logging, etc. An application can use a SIR address without knowledge that it encodes an identifier.

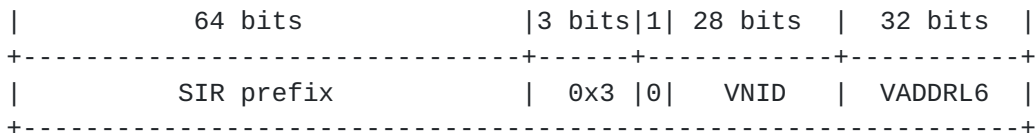
#### [3.4.1](#) SIR for locally unique identifiers

The SIR address for a locally unique identifier has format:



#### [3.4.2](#) SIR for virtual addresses

A virtual address can be encoded using the standard identifier representation. For example, the SIR address for an IPv6 virtual address may be:



Herbert

Expires September 14, 2017

[Page 15]

Note that this allows three representations of the same address in the network: as a virtual address, a SIR address, and an ILA address.

### **[3.4.3](#) SIR domains**

Each SIR prefix defines a SIR domain. A SIR domain is a unique name space for identifiers within a domain. The full identity of a node is thus determined by an identifier and SIR domain (SIR prefix). Locators **MUST** map to only one SIR domain in order to ensure that translation from a locator to SIR prefix is unambiguous.

## **[4](#) Operation**

This section describes operation methods for using identifier-locator addressing.

### **[4.1](#) Identifier to locator mapping**

An application initiates a communication or flow using a SIR address or virtual address for a destination. In order to send a packet on the network, the destination address is translated by an ILA router or an ILA host in the path. An ILA node maintains a list of mappings from identifier to locator to perform this translation.

The mechanisms of propagating and maintaining identifier to locator mappings are outside the scope of this document.

### **[4.2](#) Address translations**

With ILA, address translation is performed to convert SIR addresses to ILA addresses, and ILA addresses to SIR addresses. Translation is usually done on a destination address as a form of source routing, however translation on source virtual addresses to SIR addresses can also be done to support some network virtualization scenarios (see [appendix A.7](#) for example).

#### **[4.2.1](#) SIR to ILA address translation**

When translating a SIR address to an ILA address the SIR prefix in the address is overridden with a locator, and checksum neutral mapping may be performed. Since this operation is potentially done for every packet the process should be very efficient (particularly the lookup and checksum processing operations).

The typical steps to transmit a packet using ILA are:

- 1) Host stack creates a packet with source address set to a local address (possibly a SIR address) for the local identity, and

Herbert

Expires September 14, 2017

[Page 16]

the destination address is set to the SIR address or virtual address for the peer. The peer address may have been discovered through DNS or other means.

- 2) An ILA router or host translates the packet to use the locator. If the original destination address is a SIR address then the SIR prefix is overwritten with the locator. If the original packet is a virtually addressed tenant packet then the virtual address is translated per [section 3.3.2](#). The locator is discovered by a lookup in the locator to identifier mappings.
- 3) The ILA node performs checksum-neutral mapping if configured for that ([section 4.4.1](#)).
- 4) Packet is forwarded on the wire. The network routes the packet to the host indicated by the locator.

#### **[4.2.2](#) ILA to SIR address translation**

When a destination node (ILA router or end host) receives an ILA addressed packet, the ILA address MUST be translated back to a SIR address (or tenant address) before upper layer processing.

The steps of receive processing are:

- 1) Packet is received. The destination locator is verified to match a locator assigned to the host.
- 2) A lookup is performed on the destination identifier to find if it addresses a local identifier. If match is found, either the locator is overwritten with SIR prefix (for locally unique identifier type) or the address is translated back to a tenant virtual address as shown in [appendix A.7](#).
- 3) Perform reverse checksum-neutral mapping if C-bit is set ([section 4.4.1](#)).
- 4) Perform any optional policy checks; for instance that the source may send a packet to the destination address, that packet is not illegitimately crossing virtual networks, etc.
- 5) Forward packet to application processing.

#### **[4.3](#) Virtual networking operation**

When using ILA with virtual networking identifiers, address translation is performed to convert tenant virtual network and virtual addresses to ILA addresses, and ILA addresses back to a



Herbert

Expires September 14, 2017

[Page 17]

virtual network and tenant's virtual addresses. Translation may occur on either source address, destination address, or both (see scenarios for virtual networking in [Appendix A](#)). Address translation is performed similar to the SIR translation cases described above.

#### **[4.3.1](#) Crossing virtual networks**

With explicit configuration, virtual network hosts may communicate directly with virtual hosts in another virtual network by using SIR addresses for virtualization in both the source and destination addresses. This might be done to allow services in one virtual network to be accessed from another (by prior agreement between tenants). See [appendix A.13](#) for example of ILA addressing for such a scenario.

#### **[4.3.2](#) IPv4/IPv6 protocol translation**

An IPv4 tenant may send a packet that is converted to an IPv6 packet with ILA addresses. Similarly, an IPv6 packet with ILA addresses may be converted to an IPv4 packet to be received by an IPv4-only tenant. These are IPv4/IPv6 stateless protocol translations as described in [[RFC6144](#)] and [[RFC6145](#)]. See [appendix A.12](#) for a description of these scenarios.

#### **[4.4](#) Transport layer checksums**

Packets undergoing ILA translation may encapsulate transport layer checksums (e.g. TCP or UDP) that include a pseudo header that is affected by the translation.

ILA provides two alternatives do deal with this:

- o Perform a checksum-neutral mapping to ensure that an encapsulated transport layer checksum is kept correct on the wire.
- o Send the checksum as-is, that is send the checksum value based on the pseudo header before translation.

Some intermediate devices that are not the actual end point of a transport protocol may attempt to validate transport layer checksums. In particular, many Network Interface Cards (NICs) have offload capabilities to validate transport layer checksums (including any pseudo header) and return a result of validation to the host. Typically, these devices will not drop packets with bad checksums, they just pass a result to the host. Checksum offload is a performance benefit, so if packets have incorrect checksums on the wire this benefit is lost. With this incentive, applying a checksum-

Herbert

Expires September 14, 2017

[Page 18]

neutral mapping is the recommended alternative. If it is known that the addresses of a packet are not included in a transport checksum, for instance a GRE packet is being encapsulated, then a source may choose not to perform checksum-neutral mapping.

#### **4.4.1 Checksum-neutral mapping**

When a change is made to one of the IP header fields in the IPv6 pseudo-header checksum (such as one of the IP addresses), the checksum field in the transport layer header may become invalid. Fortunately, an incremental change in the area covered by the Internet standard checksum [[RFC1071](#)] will result in a well-defined change to the checksum value [[RFC1624](#)]. So, a checksum change caused by modifying part of the area covered by the checksum can be corrected by making a complementary change to a different 16-bit field covered by the same checksum.

ILA can perform a checksum-neutral mapping when a SIR prefix or virtual address is translated to a locator in an IPv6 address, and performs the reverse mapping when translating a locator back to a SIR prefix or virtual address. The low order sixteen bits of the identifier contain the checksum adjustment value for ILA.

On transmission, the translation process is:

- 1) Compute the one's complement difference between the SIR prefix and the locator. Fold this value to 16 bits (add-with-carry four 16-bit words of the difference).
- 2) Add-with-carry the bit-wise not of the 0x1000 (i.e. 0xefff) to the value from #1. This compensates the checksum for setting the C-bit.
- 3) Add-with-carry the value from #2 to the low order sixteen bits of the identifier.
- 4) Set the resultant value from #3 in the low order sixteen bits of the identifier and set the C-bit.

Note that the "adjustment" (the 16-bit value set in the identifier in set #3) is fixed for a given SIR to locator mapping, so the adjustment value can be saved in an associated data structure for a mapping to avoid computing it for each translation.

On reception of an ILA addressed packet, if the C-bit is set in an ILA address:

- 1) Compute the one's complement difference between the locator in

Herbert

Expires September 14, 2017

[Page 19]

the address and the SIR prefix that the locator is being translated to. Fold this value to 16 bits (add-with-carry four 16-bit words of the difference).

- 2) Add-with-carry 0x1000 to the value from #1. This compensates the checksum for clearing the C-bit.
- 3) Add-with-carry the value from #2 to the low order sixteen bits of the identifier.
- 4) Set the resultant value from #3 in the low order sixteen bits of the identifier and clear the C-bit. This restores the original identifier sent in the packet.

#### **[4.4.2](#) Sending an unmodified checksum**

When sending an unmodified checksum, the checksum is incorrect as viewed in the packet on the wire. At the receiver, ILA translation of the destination ILA address back to the SIR address occurs before transport layer processing. This ensures that the checksum can be verified when processing the transport layer header containing the checksum. Intermediate devices are not expected to drop packets due to a bad transport layer checksum.

#### **[4.5](#) Address selection**

There may be multiple possibilities for creating either a source or destination address. A node may be associated with more than one identifier, and there may be multiple locators for a particular identifier. The choice of locator or identifier is implementation or configuration specific. The selection of an identifier occurs at flow creation and must be invariant for the duration of the flow. Locator selection must be done at least once per flow, and the locator associated with the destination of a flow may change during the lifetime of the flow (for instance in the case of a migrating connection it will change). ILA address selection should follow specifications in Default Address Selection for Internet Protocol Version 6 (IPv6) [[RFC6724](#)].

#### **[4.6](#) Duplicate identifier detection**

As part of implementing the locator to identifier mapping, duplicate identifier detection should be implemented in a centralized control plane. A registry of identifiers could be maintained (possibly in association the identifier to locator mapping database). When a node creates an identifier it registers the identifier, and when the identifier is no longer in use (e.g. task completes) the identifier is unregistered. The control plane should be able to detect a

Herbert

Expires September 14, 2017

[Page 20]

registration attempt for an existing identifier and deny the request.

#### **[4.7 ICMP error handling](#)**

A packet that contains an ILA address may cause ICMP errors within the network. In this case the ICMP data contains an IP header with an ILA address. ICMP messages are sent back to the source address in the packet. Upon receiving an ICMP error the host will process it differently depending on whether it is ILA capable.

##### **[4.7.1 Handling ICMP errors by ILA capable hosts](#)**

If a host is ILA capable it can attempt to reverse translate the ILA address in the destination of a header in the ICMP data back to a SIR address that was originally used to transmit the packet. The steps are:

- 1) Assume that the upper sixty-four bits of the destination address in the ICMP data is a locator. Try match these bits back to a SIR address. If the host is only in one SIR domain, then the mapping to SIR address is implicit. If the host is in multiple domains then a locator to SIR addresses table can be maintained for this lookup.
- 2) If the identifier is marked with checksum-neutral mapping, undo the checksum-neutral using the SIR address found in #1. The resulting identifier address is potentially the original address used to send the packet.
- 3) Lookup the identifier in the identifier to locator mapping table. If an entry is found compare the locator in the entry to the locator (upper sixty-four bits) of the destination address in the IP header of the ICMP data. If these match then proceed to next step.
- 4) Overwrite the upper sixty-four bits of the destination address in the ICMP data with the found SIR address and overwrite the low order sixty-four bits with the found identifier (the result of undoing checksum-neutral mapping). The resulting address should be the original SIR address used in sending. The ICMP error packet can then be received by the stack for further processing.

##### **[4.7.2 Handling ICMP errors by non-ILA capable hosts](#)**

A non-ILA capable host may receive an ICMP error generated by the network that contains an ILA address in an IP header contained in the ICMP data. This would happen in the case that an ILA router performed



Herbert

Expires September 14, 2017

[Page 21]

translation on a packet the host sent and that packet subsequently generated an ICMP error. In this case the host receiving the error message will attempt to find the connection state corresponding to the packet in headers the ICMP data. Since the host is unaware of ILA the lookup for connection state should fail. Because the host cannot recover the original addresses it used to send the packet, it won't be able any to derive any useful information about the original destination of the packet that it sent.

If packets for a flow are always routed through an ILA router in both directions, for example ILA routers are coincident with edge routes in a network, then ICMP errors could be intercepted by an intermediate node which could translate the destination addresses in ICMP data back to the original SIR addresses. A receiving host would then see the destination address in the packet of the ICMP data to be that it used to transmit the original packet.

## **[4.8](#) Multicast**

ILA is generally not intended for use with multicast. In the case of multicast, routing of packets is based on the source address. Neither the SIR address nor an ILA address is suitable for use as a source address in a multicast packet. A SIR address is unroutable and hence would make a multicast packet unroutable if used as a source address. Using an ILA address as the source address makes the multicast packet routable, but this exposes ILA address to applications which is especially problematic on a multicast receiver that doesn't support ILA.

If all multicast receivers are known to support ILA, a local locator address may be used in the source address of the multicast packet. In this case, each receiver will translate the source address from an ILA address to a SIR address before delivering packets to an application.

## **[5](#) Motivation for ILA**

### **[5.1](#) Use cases**

#### **[5.1.1](#) Multi-tenant virtualization**

In multi-tenant virtualization overlay networks are established for tenants to provide virtual networks. Each tenant may have one or more virtual networks and a tenant's nodes are assigned virtual addresses within virtual networks. Identifier-locator addressing may be used as an alternative to traditional network virtualization encapsulation protocols used to create overlay networks (e.g. VXLAN [[RFC7348](#)]). [Section 5.2.4](#) describes the advantages of using ILA in lieu of

Herbert

Expires September 14, 2017

[Page 22]

encapsulation protocols.

Tenant systems (e.g. VMs) run on physical hosts and may migrate to different hosts. A tenant system is identified by a virtual address and virtual networking identifier of a corresponding virtual network. ILA can encode the virtual address and a virtual networking identifier in an ILA identifier. Each identifier is mapped to a locator that indicates the current host where the tenant system resides. Nodes that send to the tenant system set the locator per the mapping. When a tenant system migrates its identifier to locator mapping is updated and communicating nodes will use the new mapping.

### **[5.1.2](#) Datacenter virtualization**

Datacenter virtualization virtualizes networking resources. Various objects within a datacenter can be assigned addresses and serve as logical endpoints of communication. A large address space, for example that of IPv6, allows addressing to be used beyond the traditional concepts of host based addressing. Addressed objects can include tasks, virtual IP addresses (VIPs), pieces of content, disk blocks, etc. Each object has a location which is given by the host on which an object resides. Some objects may be migratable between hosts such that their location changes over time.

Objects are identified by a unique identifier within a namespace for the datacenter (appendix B discusses methods to create unique identifiers for ILA). Each identifier is mapped to a locator that indicates the current host where the object resides. Nodes that send to an object set the locator per the mapping. When an object migrates its identifier to locator mapping is updated and communicating nodes will use the new mapping.

A datacenter object of particular interest is tasks, units of execution for applications. The goal of virtualizing tasks is to maximize resource efficiency and job scheduling. Tasks share many properties of tenant systems, however they are finer grained objects, may have a shorter lifetimes, and are likely created in greater numbers. [Appendix C](#) provides more detail and motivation for virtualizing tasks using ILA.

### **[5.1.3](#) Device mobility**

ILA may be applied as a solution for mobile devices. These are devices, smart phones for instance, that physically move between different networks. The goal of mobility is to provide a seamless transition when a device moves from one network to another.

Each mobile device is identified by unique identifier within some

Herbert

Expires September 14, 2017

[Page 23]

provider domain. ILA encodes the identifier for the device in an ILA identifier. Each identifier is mapped to a locator that indicates the current network or point of attachment for the device. Nodes that send to the device set the locator per the mapping. When a mobile device moves between networks its identifier to locator mapping is updated and communicating nodes will use the new mapping.

## **[5.2](#) Alternative methods**

This section discusses the merits of alternative solution that have been proposed to provide network virtualization or mobility in IPv6.

### **[5.2.1](#) ILNP**

ILNP splits an address into a locator and identifier in the same manner as ILA. ILNP has characteristics, not present in ILA, that prevent it from being a practical solution:

- o ILNP requires that transport layer protocol implementations must be modified to work over ILNP.
- o ILNP can only be implemented in end hosts, not within the network. This essentially requires that all end hosts need to be modified to participate in mobility.
- o ILNP employs IPv6 extension headers which are mostly considered non-deployable. ILA does not use these.
- o Core support for ILA is in upstream Linux, to date there is no publicly available source code for ILNP.
- o ILNP involves DNS to distribute mapping information, ILA assumes mapping information is not part of naming.

### **[5.2.2](#) Flow label as virtual network identifier**

The IPv6 flow label could conceptually be used as a 20-bit virtual network identifier in order to indicate a packet is sent on an overlay network. In this model the addresses may be virtual addresses within the specified virtual network. Presumably, the tuple of flow-label and addresses could be used by switches to forward virtually addressed packets.

This approach has some issues:

- o Forwarding virtual packets to their physical location would require specialized switch support.

Herbert

Expires September 14, 2017

[Page 24]

- o The flow label is only twenty bits, this is too small to be a discriminator in forwarding a virtual packet to a specific destination. Conceptually, the flow label might be used in a type of label switching to solve that.
- o The flow label is not considered immutable in transit, intermediate devices may change it.
- o The flow label is not part of the pseudo header for transport checksum calculation, so it is not covered by any transport (or other) checksums.

### **[5.2.3](#) Extension headers**

To accomplish network virtualization an extension header, as a destination or routing option, could be used that contains the virtual destination address of a packet. The destination address in the IPv6 header would be the topological address for the location of the virtual node. Conceivably, segment routing could be used to implement network virtualization in this manner.

This technique has some issues:

- o Intermediate devices must not insert extension headers [[RFC2460bis](#)].
- o Extension headers introduce additional packet overhead which may impact performance.
- o Extension headers are not covered by transport checksums (as the addresses would be) nor any other checksum.
- o Extension headers are not widely supported in network hardware or devices. For instance, several NIC offloads don't work in the presence of extension headers.

### **[5.2.4](#) Encapsulation techniques**

Various encapsulation techniques have been proposed for implementing network virtualization and mobility. LISP is an example of an encapsulation that is based on locator identifier separation similar to ILA. The primary drawback of encapsulation is complexity and per packet overhead. For, instance when LISP is used with IPv6 the encapsulation overhead is fifty-six bytes and two IP headers are present in every packet. This adds considerable processing costs, requires considerations to handle path MTU correctly, and certain network accelerations may be lost.



Herbert

Expires September 14, 2017

[Page 25]

## **6 IANA Considerations**

There are no IANA considerations in this specification.

## **7 References**

### **7.1 Normative References**

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 2460](#), December 1998.
- [RFC2460bis] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [draft-ietf-6man-rfc2460bis-03](#), January 2016.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", [RFC 4291](#), February 2006.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", [RFC 6296](#), June 2011.
- [RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", [RFC 1071](#), September 1988.
- [RFC1624] Rijssinghani, A., "Computation of the Internet Checksum via Incremental Update", [RFC 1624](#), May 1994.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", [RFC 6724](#), September 2012.

### **7.2 Informative References**

- [RFC6740] RJ Atkinson and SN Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description", [RFC 6740](#), November 2012.
- [RFC6741] RJ Atkinson and SN Bhatti, "Identifier-Locator Network Protocol (ILNP) Engineering Considerations", [RFC 6741](#), November 2012.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", [BCP 5](#), [RFC 1918](#), February 1996.
- [RFC3363] Bush, R., Durand, A., Fink, B., Gudmundsson, O., and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", [RFC 3363](#), August 2002.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global

Herbert

Expires September 14, 2017

[Page 27]

Unicast Address Format", [RFC 3587](#), August 2003.

- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", [RFC 4193](#), October 2005.
- [RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", [RFC 6144](#), April 2011.
- [NVO3ARCH] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and Narten, T., "An Architecture for Overlay Networks (NVO3)", [draft-ietf-nvo3-arch-03](#)
- [GUE] Herbert, T., and Yong, L., "Generic UDP Encapsulation", [draft-herbert-gue-02](#), work in progress.
- [GUESEC] Yong, L., and Herbert, T. "Generic UDP Encapsulation (GUE) for Secure Transport", [draft-hy-gue-4-secure-transport-00](#), work in progress

## **[8](#) Acknowledgments**

The author would like to thank Mark Smith, Lucy Yong, Erik Kline, Saleem Bhatti, Petr Lapukhov, Blake Matheny, Doug Porter, Pierre Pfister, and Fred Baker for their insightful comments for this draft; Roy Bryant, Lorenzo Colitti, Mahesh Bandewar, and Erik Kline for their work on defining and applying ILA.



## Appendix A: Communication scenarios

This section describes the use of identifier-locator addressing in several scenarios.

### [A.1](#) Terminology for scenario descriptions

A formal notation for identifier-locator addressing with ILNP is described in [[RFC6740](#)]. We extend this to include for network virtualization cases.

Basic terms are:

- A = IP Address
- I = Identifier
- L = Locator
- LUI = Locally unique identifier
- VNI = Virtual network identifier
- VA = An IPv4 or IPv6 virtual address
- VAX = An IPv6 networking identifier (IPv6 VA mapped to VAX)
- SIR = Prefix for standard identifier representation
- VNET = IPv6 prefix for a tenant (assumed to be globally routable)
- Iaddr = IPv6 address of an Internet host

An ILA IPv6 address is denoted by

L:I

A SIR address with a locally unique identifier and SIR prefix is denoted by

SIR:LUI

A virtual identifier with a virtual network identifier and a virtual IPv4 address is denoted by

VNI:VA

An ILA IPv6 address with a virtual networking identifier for IPv4 would then be denoted

L:(VNI:VA)

The local and remote address pair in a packet or endpoint is denoted

A,A

An address translation sequence from SIR addresses to ILA addresses

Herbert

Expires September 14, 2017

[Page 29]



for transmission on the network and back to SIR addresses at a receiver has notation:

A,A -> L:I,A -> A,A

## A.2 Identifier objects

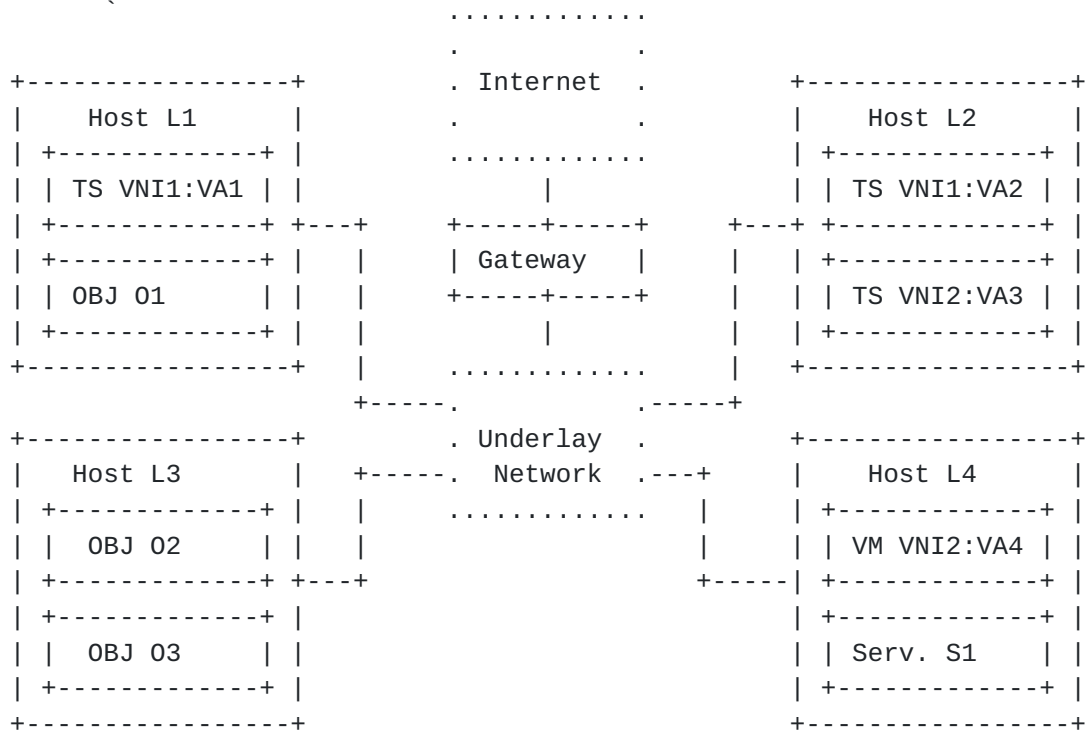
Identifier-locator addressing is broad enough in scope to address many different types of networking entities. For the purposes of this section we classify these as "objects" and "tenant systems".

Objects encompass uses where nodes are address by local unique identifiers (LUI). In the scenarios below objects are denoted by OBJ.

Tenant systems are those associated with network virtualization that have virtual addresses (that is they are addressed by VNI:VA). In the scenarios below tenant systems are denoted by TS.

## A.3 Reference network for scenarios

The figure below provides an example network topology with ILA addressing in use. In this example, there are four hosts in the network with locators L1, L2, L3, and L4. There three objects with identifiers 01, 02, and 03, as well as a common networking service with identifier S1. There are two virtual networks VNI1 and VNI2, and four tenant systems addressed as: VA1 and VA2 in VNI1, VA3 and VA4 in VNI2. The network is connected to the Internet via a gateway.





Several communication scenarios can be considered:

- 1) Object to object
- 2) Object to Internet
- 3) Internet to object
- 4) Tenant system to local service
- 5) Object to tenant system
- 6) Tenant system to Internet
- 7) Internet to tenant system
- 8) IPv4 tenant system to service
- 9) Tenant system to tenant system same virtual network using IPv6
- 10) Tenant system to tenant system in same virtual network using IPv4
- 11) Tenant system to tenant system in different virtual network using IPv6
- 12) Tenant system to tenant system in different virtual network using IPv4
- 13) IPv4 tenant system to IPv6 tenant system in different virtual networks

#### [A.4](#) Scenario 1: Object to task

The transport endpoints for object to object communication are the SIR addresses for the objects. When a packet is sent on the wire, the locator is set in the destination address of the packet. On reception the destination addresses is converted back to SIR representation for processing at the transport layer.

If object 01 is communicating with object 02, the ILA translation sequence would be:

|                  |                              |
|------------------|------------------------------|
| SIR:01,SIR:02 -> | // Transport endpoints on 01 |
| SIR:01,L3:02 ->  | // ILA used on the wire      |
| SIR:01,SIR:02    | // Received at 02            |

#### [A.5](#) Scenario 2: Object to Internet

Communication from an object to the Internet is accomplished through use of a SIR address (globally routable) in the source address of packets. No ILA translation is needed in this path.

If object 01 is sending to an address Iaddr on the Internet, the packet addresses would be:

SIR:01,Iaddr

#### [A.6](#) Scenario 3: Internet to object

Herbert

Expires September 14, 2017

[Page 31]

An Internet host transmits a packet to a task using an externally routable SIR address. The SIR prefix routes the packet to a gateway for the datacenter. The gateway translates the destination to an ILA address.

If a host on the Internet with address Iaddr sends a packet to object 03, the ILA translation sequence would be:

```
Iaddr,SIR:03 -> // Transport endpoint at Iaddr
Iaddr,L1:03 -> // On the wire in datacenter
Iaddr,SIR:03 // Received at 03
```

#### **[A.7 Scenario 4: Tenant system to service](#)**

A tenant can communicate with a datacenter service using the SIR address of the service.

If TS VA1 is communicating with service S1, the ILA translation sequence would be:

```
VNET:VA1,Saddr-> // Transport endpoints in TS
SIR:(VNET:VA1):Saddr-> // On the wire
SIR:(VNET:VA1):Saddr // Received at S1
```

Where VNET is the address prefix for the tenant and Saddr is the IPv6 address of the service.

The ILA translation sequence in the reverse path, service to tenant system, would be:

```
Saddr,SIR:(VNET:VA1) // Transport endpoints in S1
Saddr,L1:(VNET:VA1) // On the wire
Saddr,VNET:VA1 // Received at the TS
```

Note that from the point of view of the service task there is no material difference between a peer that is a tenant system versus one which is another task.

#### **[A.8 Scenario 5: Object to tenant system](#)**

An object can communicate with a tenant system through it's externally visible address.

If object 02 is communicating with TS VA4, the ILA translation sequence would be:

```
SIR:02,VNET:VA4 -> // Transport endpoints at T2
SIR:02,L4:(VNI2:VAX4) -> // On the wire
```

Herbert

Expires September 14, 2017

[Page 32]

SIR:02,VNET:VA4 // Received at TS

#### [A.9](#) Scenario 6: Tenant system to Internet

Communication from a TS to the Internet assumes that the VNET for the TS is globally routable, hence no ILA translation would be needed.

If TS VA4 sends a packet to the Internet, the addresses would be:

VNET:VA4,Iaddr

#### [A.10](#) Scenario 7: Internet to tenant system

An Internet host transmits a packet to a tenant system using an externally routable tenant prefix and address. The prefix routes the packet to a gateway for the datacenter. The gateway translates the destination to an ILA address.

If a host on the Internet with address Iaddr is sending to TS VA4, the ILA translation sequence would be:

Iaddr,VNET:VA4 -> // Endpoint at Iaddr  
Iaddr,L4:(VNI2:VAX4) -> // On the wire in datacenter  
Iaddr,VNET:VA4 // Received at TS

#### [A.11](#) Scenario 8: IPv4 tenant system to object

A TS that is IPv4-only may communicate with an object using protocol translation. The object would be represented as an IPv4 address in the tenant's address space, and stateless NAT64 should be usable as described in [[RFC6145](#)].

If TS VA2 communicates with object 03, the ILA translation sequence would be:

VA2,ADDR3 -> // IPv4 endpoints at TS  
SIR:(VNI1:VA2),L3:03 -> // On the wire in datacenter  
SIR:(VNI1:VA2),SIR:03 // Received at task

VA2 is the IPv4 address in the tenant's virtual network, ADDR4 is an address in the tenant's address space that maps to the network service.

The reverse path, task sending to a TS with an IPv4 address, requires a similar protocol translation.

For object 03 communicate with TS VA2, the ILA translation sequence would be:

Herbert

Expires September 14, 2017

[Page 33]



```
SIR:03,SIR:(VNI1:VA2) -> // Endpoints at T4
SIR:03,L2:(VNI1:VA2) -> // On the wire in datacenter
ADDR4,VA2 // IPv4 endpoint at TS
```

#### **[A.12](#) Tenant to tenant system in the same virtual network**

ILA may be used to allow tenants within a virtual network to communicate without the need for explicit encapsulation headers.

##### **[A.12.1](#) Scenario 9: TS to TS in the same VN using IPV6**

If TS VA1 sends a packet to TS VA2, the ILA translation sequence would be:

```
VNET:VA1,VNET:VA2 -> // Endpoints at VA1
VNET:VA1,L2:(VNI1,VAX2) -> // On the wire
VNET:VA1,VNET:VA2 -> // Received at VA2
```

##### **[A.12.2](#) Scenario 10: TS to TS in same VN using IPv4**

For two tenant systems to communicate using IPv4 and ILA, IPv4/IPv6 protocol translation is done both on the transmit and receive.

If TS VA1 sends an IPv4 packet to TS VA2, the ILA translation sequence would be:

```
VA1,VA2 -> // Endpoints at VA1
SIR:(VNI1:VA1),L2:(VNI1,VA2) -> // On the wire
VA1,VA2 // Received at VA2
```

Note that the SIR is chosen by an ILA node as an appropriate SIR prefix in the underlay network. Tenant systems do not use SIR address for this communication, they only use virtual addresses.

#### **[A.13](#) Tenant system to tenant system in different virtual networks**

A tenant system may be allowed to communicate with another tenant system in a different virtual network. This should only be allowed with explicit policy configuration.

##### **[A.13.1](#) Scenario 11: TS to TS in different VNs using IPV6**

For TS VA4 to communicate with TS VA1 using IPv6 the translation sequence would be:

```
VNET2:VA4,VNET1:VA1-> // Endpoint at VA4
VNET2:VA4,L1:(VNI1,VAX1)-> // On the wire
VNET2:VA4,VNET1:VA1 // Received at VA1
```

Herbert

Expires September 14, 2017

[Page 34]

Note that this assumes that VNET1 and VNET2 are globally routable between the two virtual networks.

#### **A.13.2 Scenario 12: TS to TS in different VNs using IPv4**

To allow IPv4 tenant systems in different virtual networks to communicate with each other, an address representing the peer would be mapped into each tenant's address space. IPv4/IPv6 protocol translation is done on transmit and receive.

For TS VA4 to communicate with TS VA1 using IPv4 the translation sequence may be:

```
VA4, SADDR1 -> // IPv4 endpoint at VA4
SIR:(VNI2:VA4), L1:(VNI1,VA1)-> // On the wire
SADDR4, VA1 // Received at VA1
```

SADDR1 is the mapped address for VA1 in VA4's address space, and SADDR4 is the mapped address for VA4 in VA1's address space.

#### **A.13.3 Scenario 13: IPv4 TS to IPv6 TS in different VNs**

Communication may also be mixed so that an IPv4 tenant system can communicate with an IPv6 tenant system in another virtual network. IPv4/IPv6 protocol translation is done on transmit.

For TS VA4 using IPv4 to communicate with TS VA1 using IPv6 the translation sequence may be:

```
VA4, SADDR1 -> // IPv4 endpoint at VA4
SIR:(VNI2:VA4), L1:(VNI1,VAX1)-> // On the wire
SIR:(VNI2:VA4), VNET1:VA1 // Received at VA1
```

SADDR1 is the mapped IPv4 address for VA1 in VA4's address space.

In the reverse direction, TS VA1 using IPv6 would communicate with TS VA4 with the translation sequence:

```
VNET1:VA1, SIR:(VNI2:VA4) // Endpoint at VA1
VNET1:VA1, L4:(VNI2:VA4) // On the wire
SADDR1, VA4 // Received at VA4
```



## Appendix B: unique identifier generation

The unique identifier type of ILA identifiers can address  $2^{60}$  objects. This appendix describes some method to perform allocation of identifiers for objects to avoid duplicated identifiers being allocated.

### **[B.1](#) Globally unique identifiers method**

For small to moderate sized deployments the technique for creating locally assigned global identifiers described in [[RFC4193](#)] could be used. In this technique a SHA-1 digest of the time of day in NTP format and an EUI-64 identifier of the local host is performed. N bits of the result are used as the globally unique identifier.

The probability that two or more of these IDs will collide can be approximated using the formula:

$$P = 1 - \exp(-N^2 / 2^{(L+1)})$$

where P is the probability of collision, N is the number of identifiers, and L is the length of an identifier.

The following table shows the probability of a collision for a range of identifiers using a 60-bit length.

| Identifiers | Probability of Collision |
|-------------|--------------------------|
| 1000        | $4.3368 \cdot 10^{-13}$  |
| 10000       | $4.3368 \cdot 10^{-11}$  |
| 100000      | $4.3368 \cdot 10^{-09}$  |
| 1000000     | $4.3368 \cdot 10^{-07}$  |

Note that locally unique identifiers may be ephemeral, for instance a task may only exist for a few seconds. This should be considered when determining the probability of identifier collision.

### **[B.2](#) Universally Unique Identifiers method**

For larger deployments, hierarchical allocation may be desired. The techniques in Universally Unique Identifier (UUID) URN ([[RFC4122](#)]) can be adapted for allocating unique object identifiers in sixty bits. An identifier is split into two components: a registrar prefix and sub-identifier. The registrar prefix defines an identifier block which is managed by an agent, the sub-identifier is a unique value within the registrar block.

For instance, each host in a network could be an agent so that unique identifiers for objects could be created autonomously by the host.

Herbert

Expires September 14, 2017

[Page 36]

The identifier might be composed of a twenty-four bit host identifier followed by a thirty-six bit timestamp. Assuming that a host can allocate up to 100 identifiers per second, this allows about 21.8 years before wrap around.

```

/* LUI identifier with host registrar and timestamp */
|3 bits|1| 24 bits | 36 bits |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0x1 |C| Host identifier | Timestamp Identifier |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

## Appendix C: Datacenter task virtualization

This section describes some details to apply ILA to virtualizing tasks in a datacenter.

### [C.1](#) Address per task

Managing the port number space for services within a datacenter is a nontrivial problem. When a service task is created, it may run on arbitrary hosts. The typical scenario is that the task will be started on some machine and will be assigned a port number for its service. The port number must be chosen dynamically to not conflict with any other port numbers already assigned to tasks on the same machine (possibly even other instances of the same service). A canonical name for the service is entered into a database with the host address and assigned port. When a client wishes to connect to the service, it queries the database with the service name to get both the address of an instance as well as its port number. Note that DNS is not adequate for the service lookup since it does not provide port numbers.

With ILA, each service task can be assigned its own IPv6 address and therefore will logically be assigned the full port space for that address. This is a dramatic simplification since each service can now use a publicly known port number that does not need to be unique between services or instances. A client can perform a lookup on the service name to get an IP address of an instance and then connect to that address using a well known port number. In this case, DNS is sufficient for directing clients to instances of a service.

### [C.2](#) Job scheduling

In the usual datacenter model, jobs are scheduled to run as tasks on some number of machines. A distributed job scheduler provides the scheduling which may entail considerable complexity since jobs will often have a variety of resource constraints. The scheduler takes these constraints into account while trying to maximize utility of

Herbert

Expires September 14, 2017

[Page 37]



the datacenter in terms utilization, cost, latency, etc. Datacenter jobs do not typically run in virtual machines (VMs), but may run within containers. Containers are mechanisms that provide resource isolation between tasks running on the same host OS. These resources can include CPU, disk, memory, and networking.

A fundamental problem arises in that once a task for a job is scheduled on a machine, it often needs to run to completion. If the scheduler needs to schedule a higher priority job or change resource allocations, there may be little recourse but to kill tasks and restart them on a different machine. In killing a task, progress is lost which results in increased latency and wasted CPU cycles. Some tasks may checkpoint progress to minimize the amount of progress lost, but this is not a very transparent or general solution.

An alternative approach is to allow transparent job migration. The scheduler may migrate running jobs from one machine to another.

### **[C.3](#) Task migration**

Under the orchestration of the job scheduler, the steps to migrate a job may be:

- 1) Stop running tasks for the job.
- 2) Package the runtime state of the job. The runtime state is derived from the containers for the jobs.
- 3) Send the runtime state of the job to the new machine where the job is to run.
- 4) Instantiate the job's state on the new machine.
- 5) Start the tasks for the job continuing from the point at which it was stopped.

This model similar to virtual machine (VM) migration except that the runtime state is typically much less data-- just task state as opposed to a full OS image. Task state may be compressed to reduce latency in migration.

#### **[C.3.1](#) Address migration**

ILA facilitates address (specifically SIR address) migration between hosts as part of task migration or for other purposes. The steps in migrating an address might be:

- 1) Configure address on the target host.
- 2) Suspend use of the address on the old host. This includes handling established connections (see next section). A state may be established to drop packets or send ICMP destination



unreachable when packets to the migrated address are received.

- 3) Update the identifier to locator mapping database. Depending on the control plane implementation this may include pushing the new mapping to hosts.
- 4) Communicating hosts will learn of the new mapping via a control plane either by participation in a protocol for mapping propagation or by the ILA resolution protocol.

### **C.3.2 Connection migration**

When a task and its addresses are migrated between machines, the disposition of existing TCP connections needs to be considered.

The simplest course of action is to drop TCP connections across a migration. Since migrations should be relatively rare events, it is conceivable that TCP connections could be automatically closed in the network stack during a migration event. If the applications running are known to handle this gracefully (i.e. reopen dropped connections) then this may be viable.

For seamless migration, open connections may be migrated between hosts. Migration of these entails pausing the connection, packaging connection state and sending to target, instantiating connection state in the peer stack, and restarting the connection. From the time the connection is paused to the time it is running again in the new stack, packets received for the connection should be silently dropped. For some period of time, the old stack will need to keep a record of the migrated connection. If it receives a packet, it should either silently drop the packet or forward it to the new location.

#### **Author's Address**

Tom Herbert  
Quantonium  
4701 Patrick Henry  
Santa Clara, CA  
EMail: tom@herbertland.com

Petr Lapukhov  
1 Hacker Way  
Menlo Parck, CA  
EMail: petr@fb.com

