

**Identifier-locator addressing for network virtualization
draft-herbert-nvo3-ila-02**

Abstract

This specification describes identifier-locator addressing (ILA) in IPv6 for network virtualization. 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

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/1id-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) 2016 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	3
2	Motivation	5
2.1	Network virtualization	5
2.1.1	Architecture	5
2.1.2	Multi-tenant virtualization	6
2.2	Data center virtualization	6
2.2.1	Address per task	6
2.2.2	Job scheduling	7
2.3	Alternative solutions in IPV6	8
2.3.1	Use flow label for VNID	8
2.3.2	Using an extension header	8
3	Address formats	9
3.1	ILA format	9
3.2	Identifier format	10
3.3	Identifier types	11
3.4	Interface identifiers	11
3.5	Locally unique identifiers	11
3.6	Virtual networking identifiers for IPV4	12
3.7	Virtual networking identifiers for IPV6	12
3.7.1	Virtual networking identifiers for IPV6 unicast	12
3.7.2	Virtual networking identifiers for IPV6 multicast	13
3.8	Standard identifier representation addresses	14
3.8.1	SIR for locally unique identifiers	15
3.8.2	SIR for virtual addresses	15
3.9	Locators	16
4	Operation	17
4.1	Identifier to locator mapping	17
4.2	Address translations	17
4.2.1	SIR to ILA address translation	17
4.2.2	ILA to SIR address translation	18
4.3	Virtual networking operation	18
4.3.1	Crossing virtual networks	18
4.3.2	IPv4/IPv6 protocol translation	19
4.4	Transport layer checksums	19
4.4.1	Checksum-neutral mapping	19
4.4.2	Sending an unmodified checksum	21

Herbert

Expires September 15, 2016

[Page 2]

4.5	Address selection	21
4.6	SIR address routing	21
4.7	Duplicate identifier detection	22
5	Communication scenarios	22
5.1	Terminology	22
5.2	Identifier objects	23
5.3	Reference network for scenarios	24
5.4	Scenario 1: Task to task	25
5.5	Scenario 2: Task to Internet	25
5.6	Scenario 3: Internet to task	25
5.7	Scenario 4: TS to service task	26
5.8	Scenario 5: Task to TS	26
5.9	Scenario 6: TS to Internet	26
5.10	Scenario 7: Internet to TS	26
5.11	Scenario 8: IPv4 TS to service	27
5.12	TS to TS in the same virtual network	28
5.12.1	Scenario 9: TS to TS in same VN using IPV6	28
5.12.2	Scenario 10: TS to TS in same VN using IPv4	28
5.13	TS to TS in a different virtual networks	28
5.13.1	Scenario 11: TS to TS in a different VNs using IPV6	28
5.13.2	Scenario 12: TS to TS in a different VNs using IPv4	28
5.13.3	Scenario 13: IPv4 TS to IPv6 TS in different VNs	29
6	Security Considerations	29
7	IANA Considerations	30
8	References	30
8.1	Normative References	30
8.2	Informative References	30
9	Acknowledgments	31
Appendix A	Task identifier generation	31
A.1	Globally unique identifiers method	31
A.2	Universally Unique Identifiers method	33
Appendix B	Task migration considerations	33
B.1	Address migration	33
B.2	Connection migration	34

[1](#) Introduction

This specification describes the data path, address formats, and expected use cases of identifier-locator addressing in IPv6 ([[RFC2460](#)]). The Identifier-Locator Network Protocol (ILNP) ([[RFC6740](#)], [[RFC6741](#)]) defines a protocol and operations model for identifier-locator addressing in IPv6. Many concepts here are taken from ILNP, however there are some differences in the context of network virtualization-- for instance in ILA a method to encode a virtual network identifier and virtual address within an identifier is defined.

In identifier-locator addressing, an IPv6 address is split into a

Herbert

Expires September 15, 2016

[Page 3]

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 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 forward the packet to the addressed application.

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.

In network virtualization, an identifier may further be split into a virtual network identifier and virtual host address. With identifier-locator addressing network virtualization 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 "encapsulation" 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.

ILA exhibits properties of different networking techniques:

- o Network Address Translation
- o Source routing
- o Encapsulation

2 Motivation

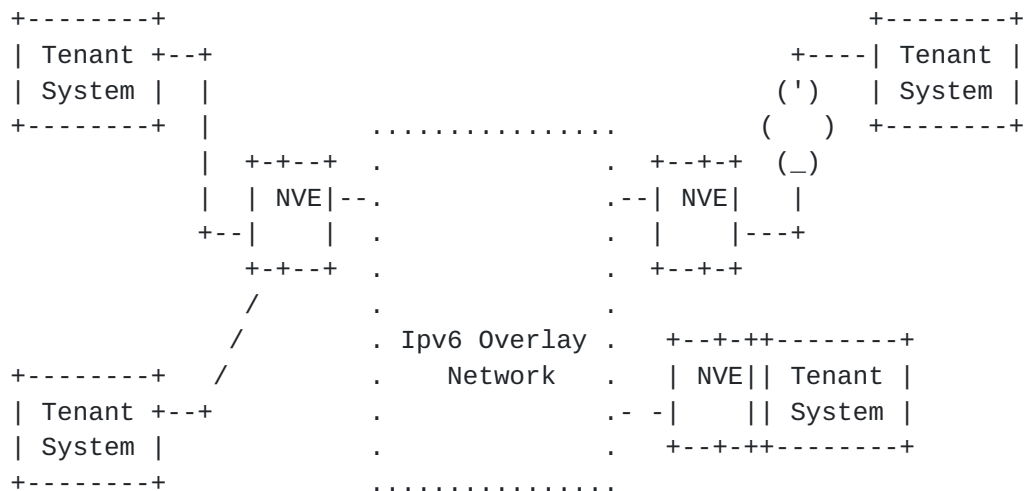
This section highlights the motivation for identifier-locator addressing.

2.1 Network virtualization

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

2.1.1 Architecture

The architecture for Network Virtualization over Layer 3 ([[NV03ARCH](#)]) can be applied to network virtualization with ILA.



A Network Virtualization Edge (NVE) [RFC7365] is the entity that implements the overlay functionality using ILA. An NVE resides at the boundary between a Tenant System and the IPV6 overlay network as shown above. An NVE creates and maintains local state about each Virtual Network for which it is providing service on behalf of a Tenant System.

As in traditional network virtualization, NVEs are responsible for transit of tenant's packets through the overlay network. With ILA, the NVEs perform address translation on packets as opposed to encapsulation. The ingress NVE will translate the virtual address of a destination to an ILA address. At the egress NVE, the reverse translation is performed.

Herbert

Expires September 15, 2016

[Page 5]

[2.1.2](#) Multi-tenant virtualization

Identifier-locator addressing may be used as an alternative to nvo3 encapsulation protocols (such as GUE [[GUE](#)]). In multi-tenant virtualization, overlay networks are established for various tenants to create virtual networks and a tenant's nodes are assigned virtual addresses. Virtual networking identifiers are used to encode a virtual network identifier and a virtual address in an ILA address.

An advantage of identifier-locator addressing is that the overhead of encapsulation is reduced and use of virtualization can be transparent to the underlying network. A downside is that some features that use additional data in an encapsulation aren't available (security option in GUE for instance [[GUESEC](#)]).

Identifier-locator addressing may be appropriate in network virtualization where the users are trusted, for instance if virtual networks were assigned to different departments within an enterprise. Network virtualization in this context provides a means of isolation of traffic belonging to different departments of a single tenant. In this scenario, if the isolation breaks and packets unintentionally crosses between virtual networks, it would not be considered a security risk.

[2.2](#) Data center virtualization

A primary motivation for identifier-locator addressing is data center virtualization. Virtualization within a data center permits malleability and flexibility in using data center resources. In particular, identifier-locator addressing virtualizes networking to allow flexible job scheduling and possibility of live task migration.

[2.2.1](#) Address per task

Managing the port number space for services within a data center 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.

Herbert

Expires September 15, 2016

[Page 6]

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.

Algorithms for the creation of unique address per task are described in [Appendix A](#).

[2.2.2](#) Job scheduling

In the usual data center 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 the data center in terms of utilization, cost, latency, etc. Data center 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.

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.

Herbert

Expires September 15, 2016

[Page 7]

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.

The networking state of interest to migrate are the addresses used by the task and open transport connections. The handling of these at task migration is discussed in [Appendix B](#).

[2.3](#) Alternative solutions in IPv6

A few alternative solutions have been proposed to provide network virtualization without encapsulation in IPv6.

[2.3.1](#) Use flow label for VNID

The IPv6 flow label could be used as a 20-bit Virtual Network Identifier. In this model the addresses may be virtual address within the specified virtual network. Presumably, the flow-label/addresses could be used by switches to forward virtually addressed packets. This has some issues:

- o Forwarding virtual packets to their physical location would require specialized switch support.
- 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 be covered by any transport (or other) checksums.

[2.3.2](#) Using an extension header

To accomplish network virtualization an extension header, probably as a destination option, could be used that contains the virtual (destination) address of a packet. The destination address in the IPv6 header would be the physical address for the location of the virtual node.

This technique also has some issues:

- o Intermediate devices must not insert extension headers

Herbert

Expires September 15, 2016

[Page 8]

[[RFC2460bis](#)]. This would preclude using extension headers in an NVE that is not co-located with a source host.

- o Extension headers introduce additional packet overhead which may impact performance.
- o Extension headers are not covered by transport checksums (as the address 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

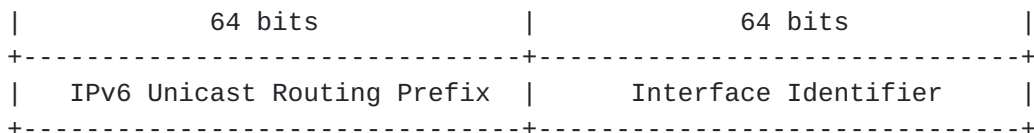
[3](#) Address formats

This section describes the address formats associated with identifier-locator addressing in network virtualization.

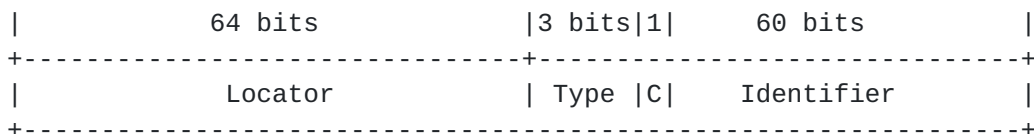
[3.1](#) ILA format

As described in ILNP ([[RFC6741](#)]) an IPv6 address may be encoded to hold a locator and identifier where each occupies sixty-four bits. In ILA, the upper three bits of the identifier indicate an identifier type. The fourth upper bit of the identifier, the C bit, is used to indicate that checksum-neutral mapping has been done (see [section 4.4](#)).

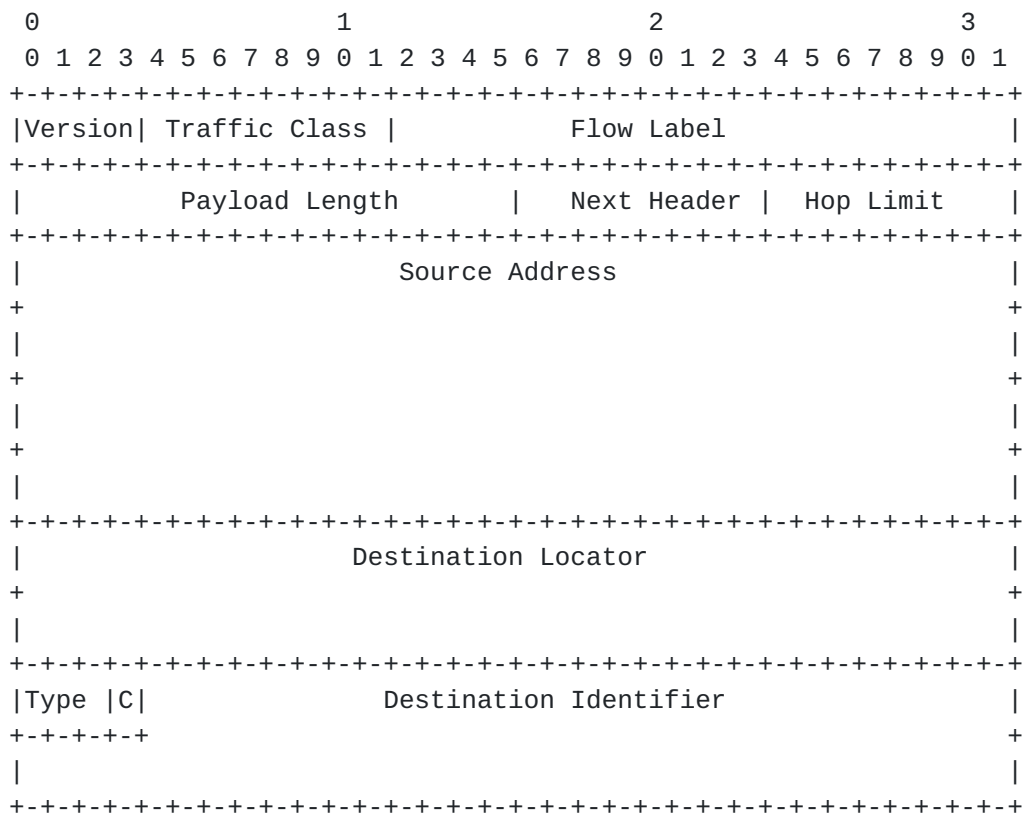
The IPv6 canonical address format is:



The address format using ILA is:

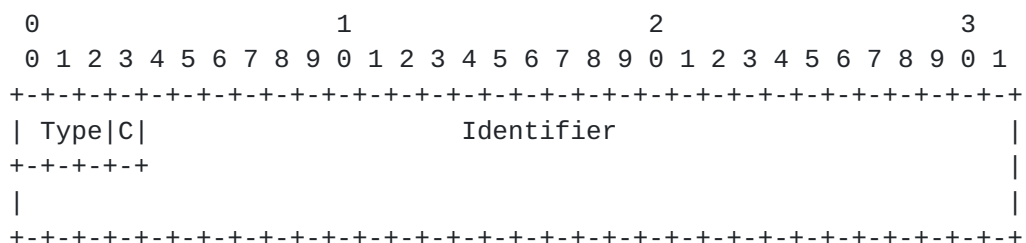


An IPv6 header with an ILA address would then have the format:



3.2 Identifier format

The format of an ILA identifier is:



o Type: Type of the identifier (see [section 3.3](#)).

o C: Checksum-neutral mapping applied

o Identifier: Identifier value.

If the C-bit is set the low order 16-bits of an identifier contain the adjustment for checksum-neutral mapping. The format of an identifier with checksum neutral mapping is:

Herbert

Expires September 15, 2016

[Page 10]

```

      0              1              2              3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Type|1|                                     Identifier                    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     | Checksum-neutral adjustment |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

3.3 Identifier types

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

3.4 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. The type bits and C-bit must be zero, and the format of the other bits (subnetting) would be site-defined. For example, the format of an interface identifier might be:

```

/* Local scope interface identifier */
|          64 bits          |3 bits|1|          60 bits          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Prefix          | 0x0 |0|          IID          |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

3.5 Locally unique identifiers

Locally unique identifiers (LUI) can be created for various addressable nodes within a network. These identifiers are in a flat sixty bit space and must be unique within a domain (unique within a site for instance). To simplify administration, hierarchical allocation of locally unique identifiers may be performed.

Herbert

Expires September 15, 2016

[Page 11]

```

/* ILA with locally unique identifiers */
|          64 bits          |3 bits|1|          60 bits          |
+-----+-----+-----+
|          Locator          | 0x1 |C| Locally unique ident. |
+-----+-----+-----+

```

[3.6 Virtual networking identifiers for IPv4](#)

This type defines a format for encoding an IPv4 virtual address and virtual network identifier within an identifier.

```

/* ILA for IPv4 virtual networking */
|          64 bits          |3 bits|1| 28 bits          | 32 bits |
+-----+-----+-----+
|          Locator          | 0x2 |C| VNID          | VADDR |
+-----+-----+-----+

```

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.

[3.7 Virtual networking identifiers for IPv6](#)

A virtual network identifier and an IPv6 virtual host address (tenant visible address) can be encoded within an identifier. Encoding the virtual host address involves mapping the 128 bit address into a sixty bit identifier. Different encodings are used for unicast and multicast addresses.

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

In this format, the 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.

```

/* IPv6 unicast encoding with VNID in ILA */
|          64 bits          |3 bits|1| 28 bits          | 32 bits |
+-----+-----+-----+
|          Locator          | 0x3 |C| VNID          | VADDR6L |
+-----+-----+-----+

```

VADDR6L contains the low order 32 bits of the IPv6 virtual address. The upper 96 bits of the virtual address inferred from the VNID to prefix mapping.

The figure below illustrates encoding a tenant IPv6 virtual unicast

Herbert

Expires September 15, 2016

[Page 12]

Herbert

Expires September 15, 2016

[Page 13]

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 address into an ILA address.

```

/* IPv6 multicast address */
| 12 bits | 4 bits|          84 bits              | 28 bits |
+-----+-----+-----+-----+
| 0xfff   | Scope |          0's              | MADDR6L |
+-----+-----+-----+-----+
                        |                               |
                        +-----+                       +-----+
                                |                               |
                                v                               v
+-----+-----+-----+-----+
|           Locator           | 0x4 |C| VNID       |Scope | MADDR6L |
+-----+-----+-----+-----+
/* IPv6 multicast address with VNID encoding in ILA */

```

3.8 Standard identifier representation addresses

An identifier serves as the external representation of a network node. 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.

/* SIR address in IPv6 */			
64 bits	3 bits	1	60 bits
+-----+-----+-----+			
SIR prefix	Type	0	Identifier
+-----+-----+-----+			

Note that the C-bit (checksum-neutral translation) is always 0 for a SIR address.

A SIR prefix may be site-local, or globally routable. A globally

Herbert

Expires September 15, 2016

[Page 14]

routable SIR prefix facilitates connectivity between hosts on the Internet and ILA endpoints. 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 translation can be performed at a receiver to restore the addresses that were included in a pseudo header for setting a transport checksum.

The 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.

3.8.1 SIR for locally unique identifiers

The SIR address for a locally unique identifier has format:

```
/* SIR address with locally unique identifiers */
|          64 bits          |3 bits|1|          60 bits          |
+-----+-----+-----+
|          SIR prefix          | 0x1 |0|Locally unique ident. |
+-----+-----+-----+
```

When using ILA with locally unique identifiers a flow tuple logically has the form:

```
(source address, source port,
 destination identifier, destination port)
```

Using standard identifier representation the flow is then represented with IPv6 addresses:

```
(source address, source port,
 destination SIR address, destination port)
```

3.8.2 SIR for virtual addresses

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

Herbert

Expires September 15, 2016

[Page 15]

```

/* SIR with IPv6 virtual network encoding */
|          64 bits          |3 bits|1| 28 bits  |  32 bits  |
+-----+-----+-----+-----+
|          SIR prefix          | 0x3 |0|   VNID   |  VADDRL6   |
+-----+-----+-----+-----+

```

In a tenant system, a flow tuple would have the form:

```
(local VADDR, local port, remote VADDR, remote port)
```

After translating packets for the flow into ILA, the flow would be identified on-the-wire as:

```
((local VNID, local VADDR), local port,
(remote VNID, remote VADDR), remote port)
```

A tenant may communicate with a peer in the network which is not in its virtual network, for instance to reach a network service (see [section 5](#)). In this case the flow tuple at the peer may be:

```
(local address, local port,
remote SIR address, remote port)
```

In this example, the remote SIR address is a SIR address for a virtual networking identifier, however from peer's connectivity perspective this is not distinguishable from a SIR address with a locally unique identifier or even a non-ILA address.

3.9 Locators

Locators are routable network address prefixes that address 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](#)].

```

/* ILA with a global unicast locator */
|<----- Locator ----->|
|3 bits| N bits          | M bits | 61-N-M | 64 bits          |
+-----+-----+-----+-----+
| 001  | Global prefix | Subnet | Host   | Identifier      |
+-----+-----+-----+-----+

/* ILA with a unique local IPv6 unicast locator */
|<----- Locator ----->|
| 7 bits |1|  40 bits   | 16 bits |          64 bits          |
+-----+-----+-----+-----+
| FC00  |L| Global ID  | Host   | Identifier      |
+-----+-----+-----+-----+

```

Herbert

Expires September 15, 2016

[Page 16]

4 Operation

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

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 identifier is mapped to a locator. The mappings are not expected to change frequently, so it is likely that locator mappings can be cached in the flow contexts.

Identifier to locator mapping is nearly identical to the mechanism needed in virtual networking to map a virtual network and virtual host address to a physical host. These mechanisms should leverage a common solution.

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 done on a destination address as a form of source routing.

4.2.1 SIR to ILA address translation

When transmitting a packet, the locator for the destination ILA address might need to be set before the packet is sent on the wire. In the case that packet was created using a standard identifier representation, the SIR prefix is overridden with a locator. Since this operation is potentially done for every packet the process should be very efficient. Presumably, a host will maintain a cache of identifier locator mappings with a fast lookup function. If there is a connection state associated with the communication, the locator information may be cached with the connection state to obviate the need to perform a lookup per packet.

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

Herbert

Expires September 15, 2016

[Page 17]

- 2) An NVE overwrites the SIR prefix in the destination address with a locator for the peer. This locator is discovered by a lookup in the locator to identifier mappings.
- 3) The NVE performs checksum-neutral mapping if configured for that ([section 4.4](#)).
- 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

Upon reception, an ILA address must be translated back to a SIR address before upper layer processing.

Receive processing may be:

- 1) Packet is received, the destination locator matches an interface address prefix on the host.
- 2) A lookup is performed on the destination identifier to find if it addresses a local identifier. If match is found, a SIR address can be created for the destination (overwrite locator with a SIR prefix).
- 3) Perform reverse checksum-neutral mapping if C-bit is set ([section 4.4](#)).
- 4) Perform any checks as necessary. Validate locators, identifiers, and check that packet is not illegitimately crossing virtual networks (see below).
- 5) Forward packet to application processing. If necessary, the addresses in the packet can be converted to SIR addresses in place.

[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 virtual network and tenant's virtual addresses. Translation may occur on either source address, destination address, or both (see scenarios for virtual networking in [section 5](#)). Address translation is performed similar to the SIR translation cases described above.

[4.3.1](#) Crossing virtual networks

Herbert

Expires September 15, 2016

[Page 18]

With explicit configuration, virtual network hosts may communicate directly with virtual hosts in another virtual network by using ILA 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).

[4.3.2](#) IPv4/IPv6 protocol translation

An IPv4 tenant may send a packet that is converted to an IPv6 packet with ILA addresses having IPv4 virtual networking identifiers. 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](#)].

[4.4](#) Transport layer checksums

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

ILA provides two alternatives to deal with this:

- o Perform a checksum-neutral mapping by performing a complementary change modification to a different 16-bit field covered by the checksum (as described in [[RFC6296](#)]).
- 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-neutral translation 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

Herbert

Expires September 15, 2016

[Page 19]

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 performs a checksum-neutral mapping when a SIR prefix is translated to a locator in an IPv6 address, and performs the reverse mapping when translating a locator back to a SIR prefix. The low order sixteen bits of the identifier contain the offset to produce a checksum-neutral translation in 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 bit-wise not of 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 and does not need to be computed for each translation.

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

- 1) Compute the one's complement difference between the locator in 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 bit-wise not of 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

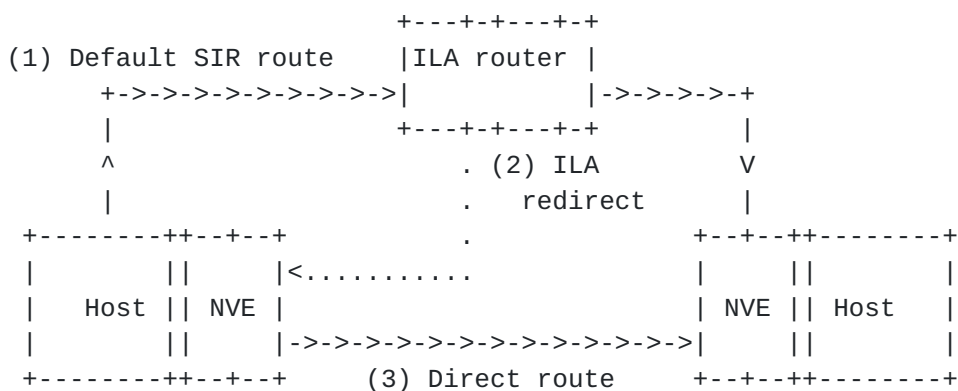
When sending an unmodified checksum, the checksum is technically incorrect as viewed in the packet on the wire. At the receiver, ILA translation of the destination ILA address back to the SIR address occurs before transport layer processing. In this way when the transport layer validates the checksum the pseudo header is based on that of the original used to set the checksum. As mentioned above, 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 selection of an identifier occurs at flow creation and must be invariant for the duration of the flow. Locator selection should be done once per flow, however may change (in the case of a migrating connection it will change). ILA address selection should follow guidelines in Default Address Selection for Internet Protocol Version 6 (IPv6) ([RFC6724]).

4.6 SIR address routing

ILA is intended to be sufficiently lightweight so that all the hosts in a data center 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 topology is illustrated below.



An ILA router is a node that implements both NVE and NVA (Network

Herbert

Expires September 15, 2016

[Page 21]

Virtualization Authority). Packets sent with a destination SIR address are routed to an ILA router (the SIR prefixes may "anycast" address prefixes to facilitate this routing). When an ILA router receives a SIR addressed packet it will perform the ILA translation and send the ILA addressed packet to the destination NVE.

Host NVEs might not be initialized with ILA identifier to locator mappings. When a host sends a SIR addressed packet, the packet is routed to an ILA router based on the SIR prefix. The ILA router provides ILA translation for the SIR prefix (this is shown in (1) above). In addition to forwarding the ILA packet, the ILA router may send an "ILA redirect" back to the source (at (2) above). The redirect indicates the locator to use for the associated identifier. Subsequently, the NVE at the source host can perform ILA translation to send directly to the destination NVE thus eliminating triangular routing (as shown in (3)). The specification of the ILA redirect message is outside the scope of this document.

[4.7](#) Duplicate identifier detection

As part of implementing the locator to identifier mapping, duplicate identifier detection may 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 registration attempt for an existing identifier and deny the request.

[5.](#) Communication scenarios

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

[5.1](#) Terminology

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

Herbert

Expires September 15, 2016

[Page 22]

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 transport endpoint IPv6 address with a locally unique identifier with 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 transport visible addresses to ILA addresses for transmission on the network and back to transport endpoint addresses at the receiver has notation:

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

5.2 Identifier objects

Identifier-locator addressing is broad enough in scope to address many different types of networking objects within a data center. For descriptive purposes we classify these objects as tasks or tenant systems.

A task is a unit of execution that runs in the data center networks. These do not run in a virtual machine, but typically run in the native host context perhaps within containers. Tasks are the execution mechanism for native jobs in the data center.

A network service is a task that provides some network wide service such as DNS, remote storage, remote logging, etc. A network service

Herbert

Expires September 15, 2016

[Page 23]

may be accessed by tenant systems as well as other tasks.

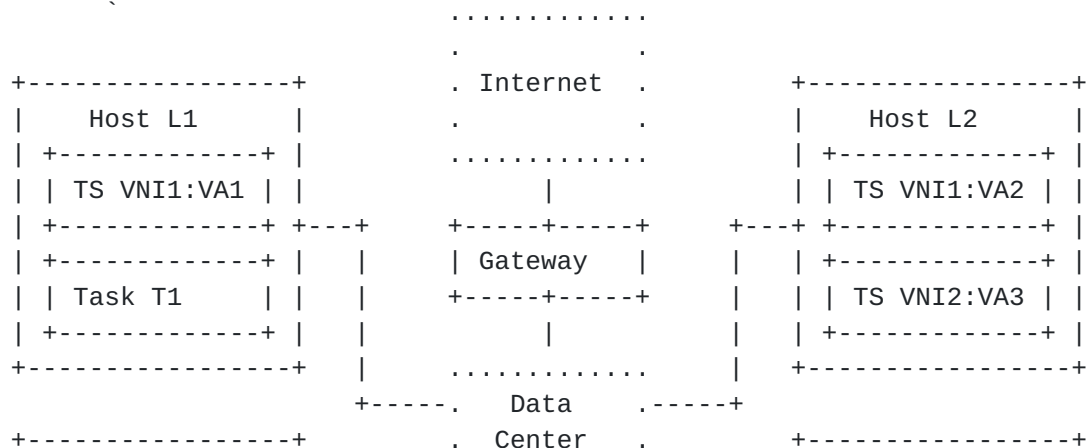
A tenant system, or TS, is a unit of execution which runs on behalf of a tenant in network virtualization. A TS may be implemented as a virtual machine or possibly using containers mechanisms. In either case, a virtual overlay network is implemented on behalf of a tenant, and isolation between tenants' virtual networks is paramount.

5.3 Reference network for scenarios

Several communication scenarios can be considered:

- 1) Task to task (service)
- 2) Task to Internet
- 3) Internet to task
- 4) TS to service
- 5) Task to TS
- 6) TS to Internet
- 7) Internet to TS
- 8) IPv4 TS to service
- 9) TS to TS in same virtual network using IPv6
- 10) TS to TS in same virtual network using IPv4
- 11) TS to TS in different virtual network using IPv6
- 12) TS to TS in different virtual network using IPv4
- 13) IPv4 TS to IPv6 TS in different virtual networks

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 tasks with identifiers T1, T2, and T3, as well as a networking service task with identifier T4. The identifiers for these tasks may be locally unique identifiers. 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.



Herbert

Expires September 15, 2016

[Page 24]



5.4 Scenario 1: Task to task

The transport endpoints for task to task communication are the SIR addresses for the tasks. 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 task T1 is communicating with task T2, the ILA translation sequence would be:

```
SIR:T1,SIR:T2 ->           // Transport endpoints on T1
SIR:T1,L3:T2 ->           // ILA used on the wire
SIR:T1,SIR:T2             // Received at T2
```

5.5 Scenario 2: Task to Internet

Communication from a task 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 task T1 is sending to an address Iaddr on the Internet, the packet addresses would be:

```
SIR:T1,Iaddr
```

5.6 Scenario 3: Internet to task

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

If a host on the Internet with address Iaddr sends a packet to task T3, the ILA translation sequence would be:

```
Iaddr,SIR:T3 ->           // Transport endpoint at Iaddr
Iaddr,L1:T3 ->           // On the wire in data center
Iaddr,SIR:T3             // Received at T3
```


Herbert

Expires September 15, 2016

[Page 25]

5.7 Scenario 4: TS to service task

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

If TS VA1 is communicating with service task T4, the ILA translation sequence would be:

```
VNET:VA1,SIR:T4->           // Transport endpoints in TS
VNET:VA1,L3:T4->           // On the wire
VNET:VA1,SIR:T4             // Received at T4
```

Where VNET is the address prefix for the tenant.

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.

5.8 Scenario 5: Task to TS

A task can communicate with a TS through it's externally visible address.

If task T2 is communicating with TS VA4, the ILA translation sequence would be:

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

5.9 Scenario 6: TS 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
```

5.10 Scenario 7: Internet to TS

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 data center. 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:

Herbert

Expires September 15, 2016

[Page 26]

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

5.11 Scenario 8: IPv4 TS to service

A TS that is IPv4-only may communicate with a data center network service using protocol translation. The network service 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 service task T4, the ILA translation sequence would be:

```
VA2,ADDR4 ->               // IPv4 endpoints at TS
SIR:(VNI1:VA2),L4:T4 ->    // On the wire in data center
SIR:(VNI1:VA2),SIR:T4      // 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 service task T4 to communicate with TS VA2, the ILA translation sequence would be:

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


5.12 TS to TS 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.

5.12.1 Scenario 9: TS to TS in 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
```

5.12.2 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
```

5.13 TS to TS in a 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.

5.13.1 Scenario 11: TS to TS in a 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
```

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

5.13.2 Scenario 12: TS to TS in a 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 the 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.

5.13.3 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 VM VA4 using IPv4 to communicate with VM 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.

6 Security Considerations

Security must be considered when using identifier-locator addressing. In particular, the risk of address spoofing or address corruption must be addressed. To classify this risk the set possible destinations for a packet are classified as trusted or untrusted. The set of possible destinations includes those that a packet may inadvertently be sent due to address or header corruption.

If the set of possible destinations are trusted then packet misdelivery is considered relatively innocuous. This might be the case in a data center if all nodes were tightly controlled under single management. Identifier-locator addressing can be used in this case without further additional security.

If the set of possible destinations contains untrusted hosts, then packet misdelivery could be a risk. This may be the case that virtual machines with untrusted third party applications or OSes are running in the network. A malicious user may be snooping for misdelivered packets, or may attempt to spoof addresses. Identifier-locator

Herbert

Expires September 15, 2016

[Page 29]

addressing should be used with stronger security and isolation mechanisms such as IPsec or GUESEC.

[7](#) IANA Considerations

There are no IANA considerations in this specification.

[8](#) References

[8.1](#) 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] Rijsinghani, 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.

[8.2](#) 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 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

[9 Acknowledgments](#)

The author would like to thank Mark Smith, Lucy Yong, Erik Kline, Saleem Bhatti, Petr Lapukhov, Blake Matheny, Doug Porter, 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: Task identifier generation

Potentially every task in a data center could be migratable as long as each task is assigned a unique identifier. Since an ILA identifier is sixty bits it is conceivable that identifiers could be allocated using a shared counter or based on a timestamp.

[A.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.

Herbert

Expires September 15, 2016

[Page 31]

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	4.3368×10^{-13}
10000	4.3368×10^{-11}
100000	4.3368×10^{-9}
1000000	4.3368×10^{-7}

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.

A.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 task 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 a task identifier could be created on the host that initially runs a task. The identifier might be composed of a twenty-four bit host identifier followed by a thirty-six bit timestamp. Assuming that a host can start up to 100 tasks per second, this allows about 21.8 years before wrap around.

```

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

```

Appendix B: Task migration considerations

B.1 Address migration

ILA facilitates address (specifically identifier) 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 an ILA redirect 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 redirect mechanism.

[B.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
Facebook
1 Hacker Way
Menlo Park, CA
EMail: tom@herbertland.com

