Identifier-locator addressing for network virtualization
draft-herbert-nvo3-ila-00

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

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. In the context of virtualization, a virtual address serves as an identifier and the address of the host where the associated tenant system currently resides is a locator.

Table of Contents

1 Introduction

This document 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 we assume that a centralized control plane will be implemented that provides mappings of identifiers to locators.

In identifier-locator addressing, an IPv6 address is split into a locator and an identifier component. The locator indicates the physical location in the network for a node, and the identifier indicates the node's identity which is the logical or virtual endpoint 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.

2 Address formats

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

2.1 ILA format

As described in ILNP ([RFC6741]) an IPv6 address may be encoded to hold a locator and identifier where each occupies 64 bits. In ILA, the upper three bits of the identifier indicate an identifier type.

```
/* IPv6 canonical address format */
|             64 bits             |             64 bits            |
+---------------------------------+--------------------------------+
|   IPv6 Unicast Routing Prefix   |      Interface Identifier      |
+---------------------------------+--------------------------------+

/* ILA for IPv6 */
|             64 bits             |3 bits|       61 bits           |
+---------------------------------+--------------------------------+
|              Locator            | Type |     Identifier          |
+------------------------------------------------------------------+
```

An IPv6 header with ILA addresses would then have the format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Source Locator                       |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Type |                Source Identifier                        |
+-+-+-+                                                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Destination Locator                    |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Type |                Destination Identifier                   |
+-+-+-+                                                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note that there is no requirement that both the source and destination are identifier-locator addresses.

2.2 Identifier format

An ILA identifier includes a three bit type field and sixy-one bits for an identfier value.

```
   /* Identifier format for ILA */
    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|                   Identifier                            |
   +-+-+-+                                                         |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

o Type: Type of the identifier (see below).

o Identifier: Identifier value.

2.3 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

2.4 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.

```
   /* Local scope interface identifier */
   |         64 bits             |3 bits|       61 bits             |
   +-----------------------------+------+---------------------------+
   |          Address1           |  0x0 |       Address2            |
   +----------------------------------------------------------------+
```

2.5 Locally unique identifiers

Locally unique identifiers (LUI) can be created for various addressable nodes within a network. These identifiers are in a flat 61 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 done.

```
/* ILA with locally unique identifiers */
|         64  bits            |3 bits|         61 bits            |
+-----------------------------+------+----------------------------+
|          Locator            |  0x1 |   Locally unique ident.    |
+-----------------------------------------------------------------+
```

2.6 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|   29 bits      |   32 bits |
+-----------------------------+------+----------------+-----------+
|          Locator            |  0x2 |    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.

2.7 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 sixy-one bit identifier. Different encodings are used for unicast and multicast addresses.

2.7.1 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 96 bit prefix.

```
/* IPv6 unicast encoding with VNID in ILA */
|           64 bits           |3 bits|   29 bits    |  32 bits  |
+-----------------------------+------+--------------+-----------+
|            Locator          |  0x3 |    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 address into a ILA address.

```
/* IPv6 virtual address seen by tenant */
+-----------------------------------------------+----------------+
|            Tenant prefix                      |  VADDR6L       |
+-----------------------+-------------------------------+--------+
                        |                               |
                        +-prefix to VNID-+              |
                                         |              |
                                         v              v
+---------------------------+------+-----------+----------------+
|            Locator        |  0x3 |   VNID    |  VADDR6L       |
+---------------------------------------------------------------+
/* Encoded IPv6 virtual address with VNID in ILA */
```

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

```
/* ILA encoded virtual networking address */
+---------------------------+------+-----------+----------------+
|            Locator        |  0x3 |   VNID    |  VADDR6L       |
+-----------------------------------------+---------------------+
                                          |              |
                        +-VNID to prefix-+               |
                        |                                |
                        v                                v
+-----------------------------------------------+----------------+
|            Tenant prefix                      |  VADDR6L       |
+---------------------------------------------------------------+
/* IPv6 virtual address seen by tenant */
```

2.7.2 Virtual networking identifiers for IPv6 multicast

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

```
      /* IPv6 multicast address with VNID encoding in ILA */
      |         64 bits           |3 bits|  29 bits   |4 bits| 28 bits  |
      +---------------------------+------+-----------------------------+
      |          Locator          |  0x4 |    VNID    |Scope |  MADDR6L |
      +-----------------------------------------------------------------+
```

This format encodes a multicast IPv6 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>::0<MADDRL6 high 12 bits>:<MADDRL6 low 16 bits>

This encoding permits encoding of 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 |    VNID    |Scope |  MADDR6L |
      +-----------------------------------------------------------------+
      /* IPv6 multicast address with VNID encoding in ILA */
```

2.8 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 nominally refer to 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             |             64 bits           |
+--------------------------------+-------------------------------+
|           SIR prefix           |            Identifier         |
+----------------------------------------------------------------+
```

A SIR prefix may be may be site-local, or globally routable. A globally routable SIR prefix allows 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, and may also allow tenant specific SIR prefixes in network virtualization.

The standard identifier representation can be used as the externally visible address for an node. This can 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.

2.8.1 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|        61 bits         |
+--------------------------------+-------------------------------+
|           SIR prefix           |  0x1 |  Locally unique ident. |
+----------------------------------------------------------------+
```

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

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

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

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

2.8.2 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:

```
   /* SIR with IPv6 virtual network encoding */
   |           64 bits            |3 bits| 29 bits     |  32 bits   |
   +------------------------------+------+-------------+------------+
   |     Tenant's SIR prefix      |  0x3 |    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 below). In this case the flow tuple at the peer may be:

```
  (local SIR 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.

2.9 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 */
   |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 */
   | 7 bits |1|  40 bits    |  16 bits |          64 bits              |
   +--------+-+------------+-----------+---------------------------+
   | FC00   |L| Global ID  | Host      |        Identifier         |
   +--------+-+------------+-----------+---------------------------+
```

3 Operation

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

3.1 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.

3.2 Address translations

With ILA, address translation is performed to convert SIR addresses to ILA addresses, and ILA addresses to SIR addresses. Translation may be done on either the source or destination address of a packet. Translation is stateless and is done per IPv6-to-IPv6 Network Prefix Translation (NPTv6) ([RFC6296]).

3.2.1 SIR to ILA address translation

When transmitting a packet, the locator for both the source and destination ILA addresses might need to be set before 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) Stack creates a packet with source address set to 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.

2) Stack overwrites the SIR prefix in the source address with an appropriate locator for the local host.

3) Stack 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.

4) If a transport checksum includes a pseudo header that covered the original addresses, the checksum needs to be updated. This should be akin to the checksum update needed in address translation for NAT ([RFC6296]).

5) Packet is sent on the wire. The network routes the packet to the host indicated by the locator.

3.2.2 ILA to SIR address translation

Upon reception, the identifier is used to match a valid address on the host or a connection context. In order to avoid having networking stack operate on a new address type, identifier-locator addresses may be translated to standard identifier representation addresses by overwriting the locator in the address with a SIR prefix.

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 match to a local identifier. If the lookup is address based, the SIR address can be created for the destination (overwrite locator with a SIR prefix).

3) Perform any checks as necessary. Validate locators, identifiers, and check that packet is not illegitimately crossing virtual networks (see below).

4) Forward packet to application processing. If necessary, the addresses in the packet can be converted to SIR addresses in place. Changing the addresses may also entail updating the checksum to reflect that (again similar to a NAT translation).

3.3 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. Address translation

is performed similar to the SIR translation cases described above.

A packet with virtual networking ILA addresses must be verified on reception. By default, the virtual network identifiers in the source and destination addresses must match or the packet is dropped. This would include the case that one address is using ILA with virtual network identifier and the other is not.

3.3.1 Crossing virtual networks

With explicit configuration, virtual network hosts may communicate directly with virtual hosts in another virtual network. This might be done to allow services in one virtual network to be accessed from another (by prior agreement between tenants). In this case, the virtual networking identifiers in the source and destination addresses won't match. This does require that identifiers are unique in a shared space.

3.3.2 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].

3.4 One sided ILA

It is not required that ILA be used for both and destination addresses. For instance a statically addressed server may provide service to virtual hosts or migratable jobs. Note that even though the server's address is static, locators for its ILA clients may change so the server will need identifier to locator mappings.

3.5 Checksum handling

TCP and UDP checksum includes a pseudo checksum that covers the IP addresses in a packet. In the case of identifier-locator addressing the checksum must include the actual addresses set in the packet on the wire. So when creating a checksum for transmit, or verifying a checksum on receive, identifier-locator addressing must be taken into account.

3.5.1 Transmit checksum

If the source and destination locators are available when the transport checksum is being set, these can be used to calculate the

pseudo checksum for the packet. This might be applicable in cases where locator information is cached within the context for a transport connection.

If the locators are set after the transport layer processing, the checksum can be updated following NAT procedures for address translation.

3.5.2 Receive checksum

Similar to the transmit case, if address translation occurs before transport layer processing the checksum must be adjusted per NAT. An implementation may verify a transport checksum before converting addresses to standard identifier representation to potentially obviate modifying the transport checksum to account for translation.

3.6 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 constant 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) ([RFC6742]).

4. Communication scenarios

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

4.1 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
VAX = An IPv6 networking identifier (IPv6 VA mapped to VAX)
```

SIR = Prefix for standard identifier representation
EXA = An Internet routable prefix, may be use as a SIR
VNET = IPv6 prefix for a tenant

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,L:I -> A,A

4.2 Identifier objects

Identifier-locator addressing is broad enough in scope to address may 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. Task are the execution mechanism for native jobs in the data center.

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 virtual networks is paramount.

A network service is a task that provides some network wide service such as DNS, remote storage, remote logging, etc. A network service may be accessed by tenant systems as well as other tasks.

4.2 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. Three tasks with identifiers T1, T2, and T3 exist 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.

```
         `                     .............
                               .           .
   +-----------------+         . Internet  .         +-----------------+
   |    Host L1      |         .           .         |    Host L2      |
   | +-------------+ |         .............         | +-------------+ |
   | | TS VNI1:VA1 | |               |               | | TS VNI1:VA2 | |
   | +-------------+ +---+     +-----+-----+     +---+ +-------------+ |
   | +-------------+ |   |     | Gateway   |     |   | +-------------+ |
   | | Task T1     | |   |     +-----+-----+     |   | | TS VNI2:VA3 | |
   | +-------------+ |   |           |           |   | +-------------+ |
   +-----------------+   |     .............     |   +-----------------+
                         +-----.   Data    .-----+
   +-----------------+         .  Center   .         +-----------------+
   |   Host L3       |   +-----.  Network  .---+     |    Host L4      |
   | +-------------+ |   |     .............   |     | +-------------+ |
   | |  Task T2    | |   |                     |     | | VM VNI2:VA4 | |
   | +-------------+ +---+                     +-----| +-------------+ |
   | +-------------+ |                               | +-------------+ |
   | |  Task T3    | |                               | | Serv. T4    | |
   | +-------------+ |                               | +-------------+ |
   +-----------------+                               +-----------------+
```

There are several communications scenario that 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

4.3 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 locators are set in source and destination addresses of the packet. On reception the source and destination addresses are converted back to SIR representations 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
L1:T1,L3:T2 ->                       // ILA used on the wire
SIR:T1,SIR:T2                        // Received at T2
```

4.4 Scenario 2: Task to Internet

Communication from a task to the Internet is accomplished through use of a gateway that translates the internal locator for the task source to an externally routable prefix.

If task T1 is sending to an address Iaddr on the Internet, the ILA translation sequence would be:

```
SIR:T1,Iaddr ->                      // Transport endpoints at T1
L1:T1,Iaddr ->                       // On the wire in data center
EXA:T1,Iaddr                         // In the Internet
```

EXA is a globally routable prefix usable on the Internet. On egress from the data center network, a gateway sets EXA in the source address. If the SIR prefix is globally routable then this may be the same as EXA.

4.5 Scenario 3: Internet to task

An Internet host transmits packet to a task using an externally routable prefix and an identifier. The subnet 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 sends a packet to task T3, the ILA translation sequence would be:

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

EXA is a globally routable prefix usable on the Internet. On ingress into the data center, a gateway overwrites this with a locator. If the SIR prefix for T3 is globally routable then this may be the same as EXA.

4.6 Scenario 4: TS to service task

A tenant can communicate with a data center service using the SIR address of the service. The source address is translated from the tenant's address and prefix to VNID and VADDR. Locators must be set properly for transmission.

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

```
VNET:VA1,SIR:T4->                     // Transport endpoints in TS
L1:(VNI1:VAX1),L3:T4->                // On the wire
SIR:(VNI1:VAX1),SIR:T4                // Received at T4
```

VNET is the address prefix for the tenant. Alternatively, the service may map the tenant's address to its SIR representation to use VNET for the endpoint:

```
VNET:VA1,SIR:T4->                     // Transport endpoints in TS
L1:(VNI1:VAX1),L3:T4->                // On the wire
VNET:VA1,SIR:T4                       // Received at T4
```

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

4.7 Scenario 5: Task to TS

A task can communicate with a TS through it's externally visible address, or by its virtual networking identifier and virtual address.

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

```
SIR:T2,SIR:(VNI2:VA4) ->           // Transport endpoints at T2
L3:T2,L4:(VNI2:VA4) ->             // On the wire
```

```
SIR:T2,VNET:VA4                    // Received at TS
```

Alternatively, the task can use the VNET prefix to address a TS:

```
SIR:T2,VNET:VA4 ->                 // Transport endpoints at T2
L3:T2,L4:(VNI2:VAX4) ->            // On the wire
SIR:T2,VNET:VA4                    // Received at TS
```

4.8 Scenario 6: TS to Internet

Communication from a TS to the Internet is accomplished through use of a gateway that translates the locator in the TS's source address back to the tenant's prefix. This assumes that the tenant's prefix is properly routed to the data center network.

If TS VA4 transmits a packet to address Iaddr on the Internet, the ILA translation sequence would be:

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

4.9 Scenario 7: Internet to TS

An Internet host transmits a packet to a tenant system using an externally routable tenant prefix and a tenant system identifier. 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:

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

4.10 Scenario 8: IPv4 TS to service

A TS that is IPv4-only may communicate with a data center network service using NAT protocol translation. The network service would 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
L2:(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
L4:T4,L2:(VNI1:VA2)   ->           // On the wire in data center
ADDR4,VA2 ->                       // IPv4 endpoint at TS
```

4.11 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.

4.11.1 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
L1:(VNI1:VAX1),L2:(VNI1,VAX2) ->    // On the wire
VNET:VA1,VNET:VA2 ->                // Received at VA2
```

4.11.2 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
L1:(VNI1:VA1),L2:(VNI1,VA2) ->      // On the wire
VA1,VA2                             // Received at VA2
```

4.12 TS to TS in a different virtual network

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.

4.12.1 Scenario 11: TS to TS in a different VN using IPV6

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

```
VNET2:VA4,VNET1:VA1->                // Endpoints at VA4
L4:(VNI2:VA4),L1:(VNI1,VA1)->        // On the wire
SIR:VA4,VNET1:VA1                    // Received at VA1
```

Alternatively, the the VNET prefix can address a TS:

```
VNET2:VA4,VNET1:VA1->                // Endpoint at VA4
L4:(VNI2:VAX4),L1:(VNI1,VAX1)->      // On the wire
VNET2:VA4,VNET1:VA1                  // Received at VA1
```

4.12.2 Scenario 12: TS to TS in a different VN 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
L4:(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.

4.12.3 Scenario 13: IPv4 TS to IPv6 TS in different VNs

Communication may also be mixed so that an IPv4 tenants 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
L4:(VNI2:VA4),L1:(VNI1,VAX1)->       // On the wire
SIR:VA4,VNET1:VA1                    // Received at VA1
```

Alternatively the task can use the VNET prefix to address a TS:

```
VA4,SADDR1 ->                        // IPv4 endpoint at VA4
```

```
L4:(VNI2:VA4),L1:(VNI1,VA1)->        // On the wire
VNET2:VA4,VNET1:VA1                  // Received at VA1
```

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

5. Use cases

This section highlights some use cases for identifier-locator addressing.

5.1 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.

5.1.1 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 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 run time state of the job. The run time state is

derived from the containers for the jobs.
3) Send the run time 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 run time 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.

5.1.1 Address migration

To allow for task migration, each migratable task is assigned a unique address which be moved to a new location at task migration.

With identifier-locator addressing, tasks are assigned locally unique identifiers (see below for assignment techniques). A LUI is combined with a SIR prefix to give each task its own IPv6 address. To communicate with a running task, the LUI is mapped to a locator which is placed in the on-the-wire packet as discussed above. When a task migrates to a new machine, the identifier to locator mapping for the task is updated to reflect the change.

5.1.2 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.

5.1.3 Task identifier generation

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

5.1.3.1 Gobally 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 61-bit length.

Identifiers	Probability of Collision
1000	2.1684*10^-13
10000	2.1684*10^-11
100000	2.1684*10^-09
1000000	2.1684*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.

5.1.3.2 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-one 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 the same host, the sub-identifier is a unique value within the registrar block.

For instance, a task identifier could be created on the initial running host that runs a task. The identifier could be composed of a 24 bit host identifier followed by a 37 bit timestamp. Assuming that a host can start up to 100 tasks per second, this allows 43.5 years before wrap around.

```
/* Task identifier with host registrar and timestamp  */
|3 bits|     24 bits       |               37  bits             |
+------+-------------------+------------------------------------+
| 0x1  |  Host identifier  |        Timestamp Identifier        |
+---------------------------------------------------------------+
```

Hierarchical allocation may also be used to support hierarchical locator lookup.

5.1.3.3 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 would be maintained. 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 able to detect a registration attempt for an existing identifier and deny the request.

5.2 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. If this isolation is broken and traffic illegitimately crosses between virtual networks, this is not considered a significant security risk.

The communication scenarios section above describes communication within a virtual network, communications with network services, and communication with hosts on the Internet.

5.2.1 IPv6 over IPv6 network virtualization

In a canonical implementation of overlay networks for network virtualization, encapsulation headers are used between outer and IP inner headers which contains a virtual network identifier and possibly other data. Typical encapsulation of an IPv6 packet using GUE is illustrated below:

```
+-------------------------------+
|                               |
|          IPv6 header          |
|                               |
|-------------------------------|
|                               |
|          UDP/GUE Header       |
|                               |
|-------------------------------|
|                               |
|          IPv6 header          |
|                               |
|-------------------------------|
|                               |
|           TCP header          |
|                               |
+-------------------------------+
```

The addresses in the outer IPv6 header indicate the physical nodes (source and destination NVEs) in the network. The inner IPv6 addresses are IPv6 addresses within the virtual network specified by the VNID in the GUE header.

Using ILA eliminates the encapsulation headers and inner IP headers:

```
+-------------------------------+
|                               |
|          IPv6 header          |
|                               |
|-------------------------------|
|                               |
|           TCP header          |
|                               |
+-------------------------------+
```

The IPv6 addresses are ILA addresses with virtual networking IPv6

identifiers. The encoded VNID indicates the virtual network the address belongs to, and the encoded VADDR provides the low order 32 bits of the virtual address for both source and destination. The tenant visible upper 96 bits of the IPv6 address is inferred from the VNID.

If the destination is multicast, the appropriate multicast identifier can be used in the destination address.

5.2.2 IPv4 over IPv6 network virtualization

The figure below illustrates the protocol headers when encapsulating a tenant's IPv4 packet using GUE.

```
+-------------------------------+
|                               |
|         IPv6 header           |
|                               |
|-------------------------------|
|                               |
|         UDP/GUE Header        |
|                               |
|-------------------------------|
|                               |
|         IPv4 header           |
|                               |
|-------------------------------|
|                               |
|          TCP header           |
|                               |
+-------------------------------+
```

The addresses in the outer IPv6 header indicate the physical nodes (source and destination NVEs) in the network. The inner IPv4 addresses are in the virtual network specified by the VNID in the GUE header.

Using ILA eliminates the encapsulation headers and inner IP headers:

```
+-------------------------------+
|                               |
|         IPv6 header           |
|                               |
|-------------------------------|
|                               |
|          TCP header           |
|                               |
+-------------------------------+
```

The IPv6 addresses are ILA addresses with virtual networking IPv4 identifiers. The encoded VNID indicates the virtual network the addresses belongs to, and the encoded VADDRs provide the IPv4 virtual addresses for both source and destination. The IPv4 virtual address are visible to the tenant systems.

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 this case without further additional security.

If the set of possible destinations are untrusted, then packet misdelivery is considered detrimental. This may be the case that virtual machines with third party applications and OS are running in the network. A malicious user may be snooping for misdelivered packets, or may attempt to spoof addresses. Identifier locator 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.

[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.

[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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[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.

[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 authors would like to thank Mark Smith, Lucy Yong, and Erik Kline for their insightful comments for this draft; Roy Bryant, Lorenzo Colitti, Mahesh Bandewar, and Erik Kline for their work on defining

and applying ILA.

Authors' Addresses

Tom Herbert
Google
1600 Amphitheatre Parkway
Mountain View, CA
EMail: therbert@google.com