IPNL Architecture and Protocol Description

Paul Francis

ACIRI
francis@aciri.org

May 8, 2000

Contents

1 Introduction

The paper proposes a solution to the IP address depletion problem. The protocol, called IPNL (for IP Next Layer), is proposed as an alternative to IPv6. IPNL is architected as a new layer above IPv4 (hereafter referred to as simply IP) rather than a replacement of IP.

IPNL uses domain names as global addresses, and connects multiple IP realms with IPNL routers (a.k.a. NAT boxes). This general approach was first concieved by Van Jacobsen years ago, but has recently been gaining currency with proponents citing its ease of transition as a primary benefit.

Within the two architectural principles of domain names as global addresses and multiple IP realms, there are a vast number of possible specific protocol designs. TRIAD is one such design (one that doesn't scale well enough). IPNL is another.

The design of IPNL is conservative in nature. It attempts to retain the look and feel of IP to the extent possible while still solving the problems at hand (address depletion, scalability, ease of transition). A key phrase in this last sentence is "to the extent possible". IPv6 looks more like IP than IPNL does, but it does so at the expense of not (yet) adequately solving all the problems at hand. Where possible, IPNL looks like IP. For instance IPNL headers carry a context-free address, not a tag. IPNL also minimizes the amount of state carried in the infrastructure. But where it is not possible, IPNL is willing to look different from IP.

Compared to "pure" IPv6, a key advantage of IPNL is that it preserves the existing IPv4 infrastructure forever, thus making transition far less expensive. ISPs can forget about ever having to evolve the core of the internet away from IP. Site network managers no longer need to think about transitioning their mission-critical router infrastructure. Even new technologies coming onto the internet, such as wireless networks, benefit because they can base most of their infrastructures on mature IP technology rather than new and no doubt buggy IPv6 technology.

Of course, the "transitional" forms of IPv6, where IPv6 runs over IPv4 in one way or another, also have the characteristic that the IPv4 infrastructure continues to exist for some amount of time. The target endpoint for IPv6 is to replace IPv4, and some IPv6 protocols and mechanisms, such as auto-address configuration, don't work unless there is an IPv6 router on the same wire as the host. In the remainder of this paper, unless otherwise stated, a reference to IPv6 refers to both its "pure" and "transitional" forms.

IPNL has two key advantages over IPv6. First, it decouples site (enterprise, SOHO, etc.) addressing from global addressing. This allows ISPs to renumber at will with zero impact on internal subscriber network operation, including multihomed subscriber networks. This allows ISPs to reduce routing table sizes, possibly by an order of magnitude or more, without requiring cooperation from or inconveniencing their subscribers. (IPv6 does this to some extent, through the use of site prefixes. All routers and hosts in a site, however, must still be renumbered when an ISP prefix changes.)

Second, the transition path for IPNL is easier. This is not simply because IPNL keeps IPv4 intact forever—it is also easier to run IPNL than to run IPv6 over IPv4. This is mainly because IPNL uses domain names as end-to-end routable addresses. As a result, the mapping of IPNL addresses (names) into global IPv4 address is no different than it is today. This makes it easy for islands of IPNL to cross the existing global IPv4 infrastructure.

IPNL also has specific support for connecting multiple IP address realms within a site. The result is that operation of a multi-realm network becomes simpler with the introduction of IPNL (even without host changes). This creates one positive incentive for transition to IPNL.

IPNL supports both multicast and anycast addressing in addition to unicast. It also includes generalized support for mobility.

2 Rough Architectural Overview

The basic architecture of IPNL is that of IP address realms interconnected by IPNL routers (called nl-routers). IP within each realm is fully self-contained, and operates unchanged and with no awareness of IPNL above it. IP addresses in IP headers never cross realm boundaries.

Within a realm, DNS protocols and implementations operate unchanged. Nl-routers understand DNS, communicate with DNS, and through DNS-like interactions between nl-routers extend DNS functionality to make it work for IPNL.

The topology of the IPNL internet is that of a huge global and public "middle realm" interconnecting much smaller "private realms". The middle realm is a continuation of today's global public IP internet. Multiple private realms may be reachable through a single middle realm IP address (MRIP).

These private realms may interconnect to form networks of their own. These networks are called realm groups. Traditional hierarchical routing algorithms operate within realm groups. For other destinations, however, packets are default routed to the middle realm. All the zones of realms that wish to be reachable outside of their realm group must be advertised to middle realm DNS.

2.1 Unicast Routing and Addressing

IPNL has two forms of address. Both are "overloaded" addresses in the sense that they contain both routing and identifying information. One is based on domain names, are called FQDNs (Fully Qualified Domain Names), and are grouped by zone. The other is a fixed-length address, and is called simply the IPNL address.

FQDNs share the following qualities with the original (pre-CIDR and pre-NAT) unicast IP address:

- They are relatively long-lived.
- While a host may have multiple FQDNs, for any given connection only a single FQDN may be used during the connection.
- They are not sensitive to the topology of the internet. A realm may attach to anywhere in the internet, and may have multiple attachments, without any effect on its FQDNs.
- They are preserved end-to-end. In particular, the transport pseudo-headers contain the source and destination FQDN.
- They are globally unique (unless intentionally assigned to multiple hosts).

The one quality of the original IP address that FQDNs don't share is that routing on FQDNs is not efficient in terms of lookup speed. This is in part because they are longer, variable length, and harder to parse. But more to the point it is because, for packets that cross the middle realm, routes have not been computed in advance. Instead, when a packet reaches the middle realm, the border nl-router must do a DNS lookup before it can further route the packet. (Keeping in mind that this operation is exactly analogous to current DNS usage in that a relatively slow DNS request must typically be made before an IP packet can be sent. And like current DNS usage, the learned address information is of course cached.)

The IPNL address compensates for this key shortcoming of the FQDN-as-address. Both forms of the IPNL address (global and local) are fixed length and simple, and one or the other appears in every IPNL packet header. The local IPNL address has two parts, and the global IPNL address has an additional third part:

- the host's local IP address (4 bytes)
- the host's realm number (2 bytes)

- the middle realm IP (MRIP) address (4 bytes, global only).

Packets routed within a realm group are local and do not require the MRIP address field. Locally addressed packets are completely insensitive to the "location" of the realms' attachment points to the rest of the internet. They do not change, for instance, if a different ISP (or ISPs) is used to connect to the internet.

Packets routed across the middle realm are global and do require the MRIP address field. Global addresses are sensitive to internet location, a host may have multiple of them (if the host's realm or realm group is multi-homed), they can change during the course of a connection, and they are not necessarily long lived.

Each of the address types (FQDNs, local address, global address) individually has limitations. FQDNs are stable and globally unique but slow. Global address are fast and globally unique but dynamic. Local addresses are fast and stable but not globally unique. By pairing the FQDN with the other two addresses, however, IPNL overcomes the limitations of each.

Basic usage of the multiple addresses is as follows. When a host sends its initial packet to another host, the source and dest MRIP and realm number fields are empty (they contain "unknown" default values), and the FQDNs are attached. Note that, from the host's perspective, there is no separate A record DNS lookup—the initial packet and return packet serve as the DNS lookup (though "third-party" lookups such as for MX records still take place). Note too that the host does not statically "know" its own realm numbers and MRIPs. Rather, these are learned on an as-needed basis just as remote hosts' IPNL addresses are learned as needed.

As the initial packet is routed through the internet, parts of the IPNL addresses are filled in. First the source IPNL address is filled in as the packet moves "up" in the topology (from realm to realm group to middle realm), and then the destination IPNL address is filled in as the packet moves back "down" in the topology. By the time the initial packet reaches the destination, the IPNL addresses are completed. In essence, an FQDN-to-address mapping has taken place. The destination stores the mapping, and can reverse the IPNL addresses and return a packet.

Although the return packet contains full IPNL addresses, it must still carry FQDNs as well. This gives the initiating host the FQDN-to-IPNL address mapping. Once mappings are learned, most subsequent packets do not contain the FQDNs. If the packets are locally addressed, they do not contain the MRIPs either. The FQDNs remain part of the transport pseudoheader, however, as a check against mis-addressed packets.

In the case of locally addressed packets, there will typically only be one IPNL address for each host during the course of a connection (barring mobility). Because traditional dynamic routing protocols run locally (within the realm group), the addresses will not change even though the path may change. As a result, address usage for locally addressed packets is not all that different from IPv4. By and large, only the FQDN-to-address mapping phase is different.

It is a different story for globally addressed packets. Unlike the case with local routing, it would be very difficult for nl-routers to maintain constant up-to-date routing information about every destination its hosts are talking to (much less all destinations globally). And in fact IPNL doesn't even try.

Instead, border nl-routers always "blindly" forward any globally addressed packet to the MRIP of the destination address. Except for cached FQDN-to-MRIP name records and cached ICMP destination unreachable messages, they don't maintain any state about border nl-routers for remote destinations. Indeed a border nl-router does not even need to keep a list of the other border nl-routers for its own realm groups. This is very much in keeping with the overall design approach for IP—to keep the middle as simple as possible.

IPNL requires that hosts themselves maintain state about how to reach global destinations they are actively communicating with. In essence, hosts keep a dynamic list of valid global addresses for destination hosts, with information on which addresses most recently worked and which, if any, have become invalid. Because the address dictates which border nl-router a packet goes through, the list of addresses in essence indicates which border nl-routers may be used to route a packet to a destination.

IPNL has a simple and robust mechanism for maintaining these lists. The IPNL packet header actually contains

two source MRIP fields. The first contains the value set by the source host, and is called the host source MRIP. The second contains the MRIP of the actual egress border nl-router of the realm group, and is set by that border nl-router. It is called the used source MRIP. All global packets contain a valid used source MRIP. The first packet(s) sent by an initiating host does not contain a valid host source MRIP—all other packets do.

The first packet received by a (non-initiating) host has nothing for the host source MRIP, a valid used source MRIP, and an FQDN. From this the host initiates a list for that FQDN containing the used MRIP, which is used to address the initiating host in return packets. In this way, the used source MRIP is conveyed back to the initiating host. This MRIP is set as the host source MRIP in subsequent packets, but without the FQDN. The non-initiating host, however, is able to recognize the identity of the initiating host from the host source MRIP, so the FQDN is not needed.

If a subsequent packet from the initiating host happens to get routed through a different egress border router, the packet will contain a new used source MRIP. In this way, the non-initiating host will learn of another global address for the initiating host, thus dynamically responding to failed border nl-routers and building up its list. This new global address is more likely to be for a working border nl-router, and so is used as the destination address in return packets.

Of course, the same thing is happening in the other direction too, so both hosts' lists get built up over time on an as-needed basis without a separate out-of-band protocol for maintaining the lists. Optionally a host can request that the list learned from DNS by the border nl-router be transmitted back to it.

2.2 Multicast Routing and Addressing

As with unicast, IPNL uses both FQDNs and addresses. Unlike unicast, hosts learn of the address associated with an FQDN when they join a group, not when they send a data packet.

There is no syntactic difference between FQDNs that define a multicast group and unicast FQDNs. Outside of protocol context, one can't be distinguished from the other.

IPNL multicast addresses are distinguished from IPNL unicast addresses in that the destination host IP address in the IPNL header is an IP multicast address (comes from the IP class D address space). Other than this, there is no syntactic difference between IPNL unicast and multicast addressed headers.

The source address of an IPNL multicast address is the unicast address of the sender. As with IPNL unicast packets, the source address is composed on the fly as the packet first leaves the source host's realm and then leaves the source host's realm group. As with IP multicast, the source address is used as part of the forwarding decision for source trees.

IPNL multicast FQDNs have home realms just as unicast FQDNs do. The role of the home realm in IPNL multicast is to assign an IP multicast address to the multicast group. This IP multicast address must be unique among all IP multicast addresses assigned by that realm, but can overlap with multicast IP addresses assigned by other realms. The total IPNL multicast address, then, consists of the IP multicast address, realm number and optionally an MRIP of the home realm. The MRIP is needed when the home realm of the multicast group is not local to one or more group members. Multicast packets are globally addressed when this is the case (whether or not a given sender is local to the home realm). The addressing mode for a multicast group can dynamically change from local to global when a member not local to the home realm joins. Such a join is detected by nl-router through a multicast routing protocol, and conveyed to senders through IPNL-ICMP destination unreachable messages.

When a host joins an IPNL multicast group, it transmits a join message containing the group FQDN to an nl-router on its realm. It is the existance of the join message itself that tells the nl-router that this FQDN is multicast rather than unicast. If the nl-router does not already know the address of the multicast group, it forwards the join to the home realm of the multicast FQDN. Depending on the multicast routing algorithm, the nl-router may at the same time initiate establishment of part of the multicast tree.

When the join reaches an nl-router attached to the home realm, it must assign an IP multicast address for the group if one has not already been assigned. This assignment must be coordinated with the other nl-routers for the realm so as to avoid multiple nl-routers assigning different IP multicast addresses to the same group at the same time.

The assigned address is finally return to the source host, which can thereafter transmit packets to and receive packets from the group without the FQDN attached.

Depending on the routing algorithm used, there are three ways to transmit packets into and across realms. First, use multicast IP for both. Second use unicast IP to transmit packets from one nl-router to another, but use multicast IP to transmit packets to host in the realm. Third, use unicast IP for both. (Note that through tunneling of IP over IP, both unicast and multicast options exist with IP as well.)

As a final note, the multicast architecture presented here is a simple extension of IP multicast, as opposed to a fundamentally new architecture. The use of FQDNs and extended addresses simplifies the multicast address assignment problem as compared to IP, but does not necessarily improve on the scaling limitations of multicast IP. The same can be said about IPv6 relative to IP.

It may make sense to, in addition to the extended multicast described here, to design a new multicast architecture based on a wholley different header format. Such work would be outside the scope of IPNL.

3 Terminology

Backdoor A direct link between two realm groups. A backdoor requires at least one, and typically two border nl-routers. An IP tunnel over the middle realm is considered to be a direct (backdoor) link.

Border IPNL Router An nl-router (or part thereof) that connects a realm group either with another realm group (backdoor) or with the middle realm (frontdoor).

Expat A host that is in a realm other than its home realm. In the case of anycast, multiple expats may have the same FQDN.

Expat FQDN The FQDN for a zone in the realm where an expat host is attached.

Frontdoor A connection between a realm group and the middle realm. A frontdoor requires at least one border nl-router.

Globally Addressable Realm A realm that can only be reached by using the MRIP field of the IPNL address. Globally addressable realms are by definition only reachable through the middle realm.

Home Realm Relative to a given zone, this is the realm where hosts with FQDNs from that zone nominally reside. Individual zone hosts may be in other realms. These hosts are called Expat hosts. A realm may be a home realm for multiple zones or zero zones. A given zone may have multiple home realms, for instance for anycast, but all individual hosts in the zone must be reachable at all such realms. In the case of multicast, the home realm is the realm where boxes that can assign multicast addresses to a multicast group reside.

Interior IPNL Router An nl-router (or part thereof) that connects two realms within a realm group.

IP IPv4.

IPNL IP Next Layer.

IPNL Address Or just address for short. This is the IP address, Realm Number, and optional MRIP address that appear in the IPNL header. Local IPNL addresses do not require the MRIP. Global IPNL addresses do.

IPNL Box Either an nl-router or host.

Realm Group A group of realms under a single administrative control, running a coordinated routing protocol (even if that protocol is based on static entries), and with coordinated realm numbering (each realm number in the realm group is distinct from all others in the realm group). The connectivity between realms in a realm group may be via the middle realm.

Locally Addressable Realm A realm is locally addressable if it can be reached without the MRIP part of the address. If a realm is in the same realm group then it must be locally addressable. Realms in other realm groups can become locally addressable by assigning an internal realm number for the external realm, maintaining routing information to point to the appropriate border routers to reach the realm, and maintaining the appropriate realm number mappings and IP-level next-hop information at those border routers.

Middle Realm The single huge realm that constitutes the global backbone of the IPNL internet.

M-FQDN An FQDN that identifies a multicast group. Syntactically it is indestinguishable from a unicast FQDN.

PMTU Path MTU (Maximum Transmission Unit), exactly as used in IP [RFC1191].

Private Realm Any realm other than the middle realm.

Proxy An nl-router that runs IPNL on behalf of an IP-only host behind it. In this capacity, it is a proxy for the IP-only host.

Realm An IP internet isolated from other IP internets by nl-routers.

Realm Identifier An FQDN that uniquely identifies a realm. The realm identifier does not necessarily need to come from any zones associated with the realm nor does it need to be attached to any host. It is used in routing protocols and the like as a simple identifier.

Realm Name An FQDN that uniquely identifies a realm, is not associated with any individual host, and is globally routable. The realm name is used to convey the current location of expat hosts.

Reset (the address) "Resetting the address" or some equivalent phrase means to cause subsequent packets to be sent with the destination IPNL address written to "unset" and the FQDN header attached. This is how initial packets between two newly communicating hosts are transmitted.

One-way Frontdoor A frontdoor nl-router that can either send or receive over the middle realm, but not both. One-way frontdoors that cannot receive over the middle realm should write the MRIP of another frontdoor nl-router into the Used Source MRIP and Used Source Realm fields.

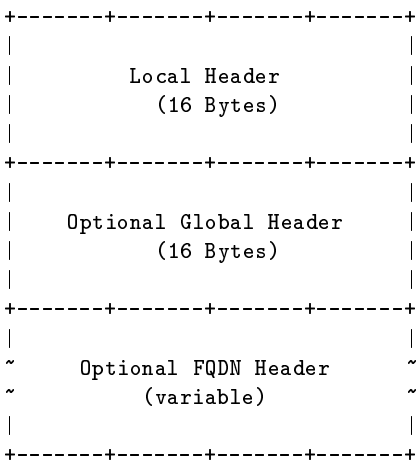
U-FQDN A unicast FQDN. "FQDN" (without a "U" or "M" preceding it) typically refers to the unicast FQDN.

Zone As used by DNS. Each bottom level zone typically has a single home realm, though it may have more if the entire zone is replicated in each realm.

4 IPNL Basic Header

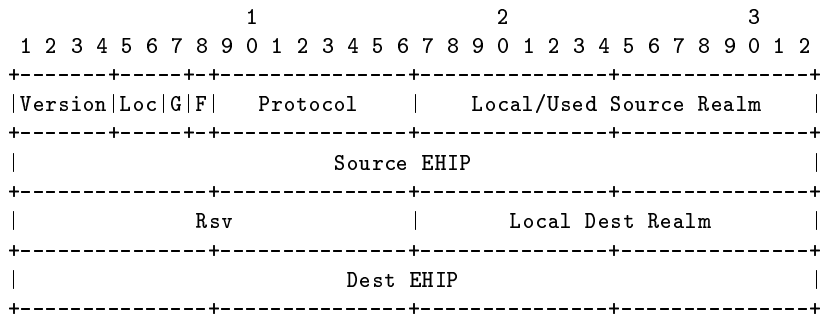
The IPNL header is encapsulated by IP. Every IPNL packet contains an IPNL header as well as an IP header.

The IPNL header consists of a 16-byte local header carried in all packets. For globally addressed packets, or packets that have the FQDN header attached, a 16-byte global header immediately follows the local header. The FQDN header, if included, immediately follows the global header.



4.1 IPNL Local Header

The IPNL local header is shown below.



The local header fields are described as follows.

Version The version of the header shown above is 1.

Loc (Location) This 2-bit field indicates a packet's location, relative to the middle realm, in its path from source to dest. This field is used by interior routers to determine if they should be routing a packet to the destination realm or to the middle realm. This field only applies to unicast (or anycast) packets. It is ignored for multicast packets.

In the case of globally addressed packet this is necessary because interior nl-routers do not know the MRIP addresses of their border nl-routers, and therefore cannot tell from inspection of a globally addressed packet whether it is on the way in or on the way out. While strictly speaking the field is not necessary for locally addressed packets, its consistent use for both globally and locally address packets simplifies header processing for interior nl-routers.

The Loc field has the following values:

Unknown (0) The location of the packet is unknown. This value may only be used when the FQDN header is attached.

Default Route (1) The packet is globally addressed and has not yet reached the middle realm.

Middle (2) The packet is globally addressed and is in the middle realm.

Route by Realm (3) The packet is either locally addressed, or has already passed through the middle realm (or both). This includes the case where the packet is locally address and is passing through the middle realm.

When a host not directly attached to the middle realm transmits its initial packets, before it knows whether the address will be global or local, it sets the value of Loc to "Unknown".

global header Present ("G" Bit) Set to 1 if the global header is attached, set to 0 otherwise. The global header, when present, always immediately follows the local header.

FQDN header Present ("F" Bit) This bit is set to 1 if the FQDN header is present, and set to 0 otherwise. If the FQDN header is present, the global header must also be present. The FQDN header must immediately follow the global header. Note that other "optional" headers are identified by using the Protocol field in the style adopted for IPv6. The FQDN and global headers are enough of a mainstream aspect of IPNL, however, that their presense is indicated with separate bits.

Protocol This field contains protocol number of the next higher layer. Its values have the same definitions as those of the IP protocol field. The next higher layer may be an options header of IPNL, in which case the options header itself will contain a protocol field for the layer after that.

Local/Used Source Realm This field has multiple roles, depending on whether the packet is globally or locally addressed, and if globally addressed, where it is in its path from source to destination. Its use in both unicast and multicast is the same.

If the packet is locally addressed, then the Local/Used Source Realm is the realm number of the source host's realm as locally defined. In this role, when the packet crosses realm group boundaries, the Local/Used Source Realm is modified from one realm group's local representation of the source realm to the other realm group's local representation. If the packet crosses the middle realm between realm groups, the Local/Used Source Realm is modified to its global value relative to the MRIP of the source frontdoor nl-router while in the middle realm. This MRIP is the source IP address of the packet. The Local/Used Source Realm field is copied into the Local Dest Realm field of return packets.

If the packet is globally addressed, it has two possible definitions. Before the packet has reached the middle realm (Loc value of 1, "Default Route"), the Local/Used Source Realm field contains the local realm number for the source realm. In other words, the same number that is used when the packet is locally addressed and still in the source realm group. This is necessary in case an interior nl-router needs to return an error packet to the source host, and is the only reason the Local/Used Source Realm field is used in this capacity for global addresses.

Once the packet reaches the middle realm, the Local/Used Source Realm field becomes the realm part of the global address defined by the Used Source MRIP field. In this context the Local/Used Source Realm field plays a strictly global role. The reason it is in the local header rather than the global header header is so that when an ICMP error messages received from the middle realm, the source host can be determined and a corresponding IPNL-ICMP message generated (see section 9).

The values for the Local/Used Source Realm and all other realm fields are defined as follows:

Reserved (0-255) These values are reserved for future use.

Unset (256) This value means that the realm number has not yet been set to any valid value.

Middle Realm (257) This value is reserved to mean the middle realm.

Valid (all other values) All other values are valid realm numbers assigned to private realms.

Source EHIP (Endhost IP Address) This is the IP address of the source host for the packet. That is to say, the IP address with which the source host is locally configured. This address is unique only in the source host's realm. It is carried intact end to end. This is true for both unicast and multicast packets.

For unicast packets, the Source EHIP is copied into the Dest EHIP of return packets.

Rsv Reserved. Transmit as 0 and ignore upon receipt.

Local Dest Realm In the case of unicast, the Local Dest Realm is the locally assigned realm number for the realm of the destination host. Whenever either i) a packet is locally addressed, or ii) a globally addressed packet has already crossed the middle realm, this field is used by interior nl-routers to route the packet to the destination realm. A packet can be determined to be in one of the above two states by a value of 3 ("Route by Realm") in the Loc field.

The Local Dest Realm field only uniquely identifies the destination realm in the local context. If the destination is within the same realm group as the source, then the Local Dest Realm field does not change once it is set to a valid value. If the destination realm is in another realm group, then the Dest Realm field may be modified as it passes from one realm group to another, and as it passes through the middle realm.

In the case of multicast, this field, along with the Dest EHIP, identifies the multicast group for locally addressed packets. The transmitting host sets this field to the value that its realm group locally uses to identify the home realm of the multicast group (that is, the realm that assigned the multicast address). When this group crosses realm group boundaries, this field is rewritten to the local value of the home realm for the new realm group. For globally addressed multicast packets, this field goes unused, and is transmitted at value "unset".

Dest EHIP (Endhost IP Address) In the case of unicast, this is the IP address of the destination host of the packet. That is to say, the IP address with which the destination host is locally configured. This address is unique only in the destination host's realm. A packet is identified as being unicast when this field is not a class D address.

This field is set to value zero if the destination host address is unknown. In this case, the packet must contain the FQDN header. A zero Dest EHIP field must be filled in by the first nl-router that knows the correct value. This will typically be the ingress nl-router of the destination realm. Once assigned (either by the source host or by an nl-router), it is carried to the destination host intact.

In the case of multicast, this is the low-order part of the multicast group address. In this case, it is always set at its correct value by the transmitting host. The host learns the correct value during the join operation.

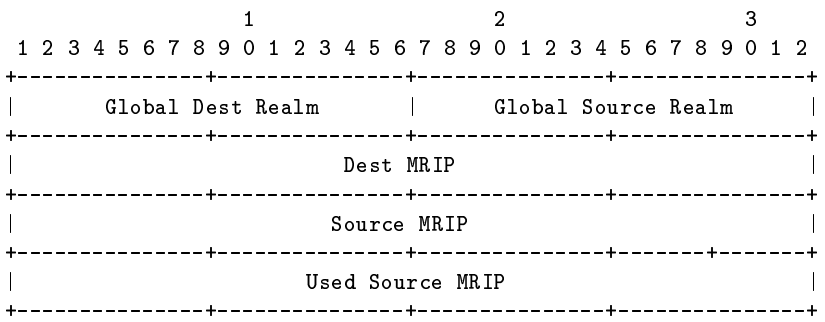
4.2 IPNL Global Header

The global header is attached for globally addressed packets, and for any packets that contain the FQDN header. Its presence can be detected by an G Bit value of 1 in the local header. It is shown below.

The fields of the global header are described as follows.

Global Dest Realm and Dest MRIP fields (Dest MRIP+RN) These two fields are discussed together because they form a unit. They are jointly refered to as the Dest MRIP+RN.

In the case of unicast, they constitute the higher-order part of the destination host's global address (with the Dest EHIP serving as the remainder of the global address). Specifically, the Dest MRIP field is a MRIP address through which the destination host is reachable. The Global Dest Realm field is the realm number that the frontdoor nl-router at that MRIP uses to identify the destination realm.



They are set by the source host unless the source host doesn't know the values (which is the case for the initial packet(s)), in which case the Dest MRIP field is typically set by the egress frontdoor nl-router, and the Global Dest Realm field is set by the ingress frontdoor nl-router. The FQDN header must be attached if these fields are transmitted as unset by the source host. Once they are set, they are carried intact to the destination host.

When set by the host, the values used may be that of any stored global addresses for the destination host. The actual values used are typically that of the most recently received valid Used Source MRIP and Local/Used Source Realm fields.

If a packet with "Unset" values in these fields is determined to be locally addressable, the values remain as "Unset". The Loc field, however, is changed from "Unknown" to "Route by Realm". The combination of a Loc value of "Route by Realm" and "Unset" Dest MRIP and Global Dest Realm values defines a packet as locally addressable.

The values for the Global Dest Realm are as defined for the Local Dest Realm field above. Values for the Dest MRIP field are defined as follows:

Reserved (0-255) These values are reserved for future use.

Unset (256) This value is used in various contexts where the actual MRIP value is either unknown or unused.

Valid MRIP (all other values) All other values may be valid MRIPs (unless disallowed by the IP protocol itself).

These MRIP definitions apply to all three MRIP field.

In the case of multicast, the Dest MRIP+RN, along with the Dest EHIP, identify the multicast group globally. The same Dest MRIP+RN must be used across all group members. A given MRIP+RN must not be assigned to another realm that is a multicast home realm until all multicast groups using that MRIP are gone.

Global Source Realm and Source MRIP fields (Source MRIP+RN) These two fields are discussed together because they form a unit. They are jointly refered to as the Source MRIP+RN. They constitute the higher-order part of the source host's global address (with the Source EHIP serving as the remainder of the global address). They are not used for multicast packets, and must be transmitted as "unused" in multicast packets and ignored upon receipt.

These fields are set by the source host. They are carried intact end to end. These fields may only be set to a valid value received previously in the Dest MRIP+RN fields, and will typically be set to the latest values received.

These two fields serve to identify the source host to the destination host.

Used Source MRIP In the case of unicast, this field is typically used to convey to the destination host the MRIP of the source realm group's frontdoor nl-router actually used by a packet. The exception to this is for one-way frontdoors, where the frontdoor used for outgoing packets cannot be used for incoming packets. In this case, the frontdoor must write the MRIP of a frontdoor nl-router that can handle incoming packets, and that is up with high probability.

In the case of multicast, this field is always used to convey the source of the packet after it has entered the middle realm.

This field, when not "Unset", is used in conjunction with the Local/Used Source Realm field. In this mode, they are jointly refered to as the Used Source MRIP+RN.

This field is set to value "Unset" by the source host. When a packet passes from a private realm to the middle realm, the frontdoor nl-router writes its MRIP (or a valid frontdoor nl-router's MRIP) into the Used Source MRIP field, and writes the realm number that the frontdoor uses to identify the source realm into the Local/Used Source Realm field. This allows the destination host to detect the new global address of the source host while at the same time being able to identify the source of the packet (from the Source MRIP+RN).

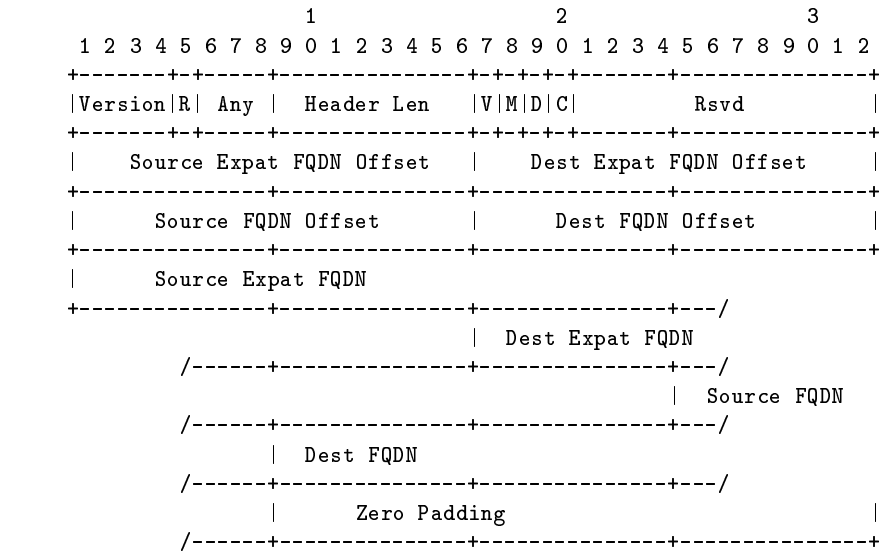
(An alternative approach might be to only write the MRIP into the Used Source MRIP field if it is different from that in the Source MRIP field. In any event, however, the Local/Used Source Realm field must be set in order to be able to determine the source of packets that generate ICMP error messages.)

In the case of multicast, the frontdoor nl-router MRIP is always different from the Source MRIP because the Source MRIP is always set at "unset" by the source host.

The egress frontdoor nl-router is the only box that may modify this field.

4.3 FQDN Header

The FQDN header contains the FQDNs of the source and destination host of the packet. It is present whenever the F Bit of the local header is 1. It is typically carried only on the initial packets passed between a pair of hosts. This binds the IPNL addresses to the FQDNs, after which the FQDNs are generally not needed. The FQDN header is formatted as follows:



The fields are defined as follows:

Version Set at 1 for this header.

R: Reserved. Transmit as 0, ignore upon receipt.

Any This field is set to a non-zero value if the packet should be "terminated" at an nl-router rather than transmitted to the destination host. In other words, the nl-router treats the packet as if it were destined for it. The values are defined as follows:

No Anycast (0): Do not terminate this packet at an nl-router.

First nl-router (1): The first nl-router to handle this packet should terminate it.

First border nl-router (2): The first border nl-router to handle this packet should terminate it.

Last border nl-router (3): The last border nl-router to handle this packet should terminate it.

Last nl-router (4): The last nl-router to handle this packet should terminate it.

Reserved (5-7): These values are reserved for future use, and should be treated the same as value 0 (no anycast).

Header Len The length of the FQDN header, in units of 32 bit words. The minimum valid value for this field is 2 (no FQDNs at all). This would be for a header that is simply acknowledging receipt a received FQDN header, but that otherwise hasn't changed from the last FQDN header sent.

Verify Dest MRIP ("V" Bit) This bit is set to 1 if the relm group's egress frontdoor nl-router should verify that the value in the Dest MRIP field matches one of those returned by DNS for the host named by the Dest FQDN.

MRIP List Request ("M" Bit) This bit is set if the source host wishes to receive the list of MRIPs for the Dest FQDN known by the frontdoor border nl-router (either from the current or a past DNS query). The default behaviour is to not request an MRIP List. Normally it is up to the application to tell the IPNL layer to request an MRIP List, for instance because the application knows that the connection will be long-lived, and therefore the overhead of obtaining the list is small.

Not in DNS ("D" Bit) This bit is set to 1 if the Source FQDN or Source Expat FQDN cannot be looked up in DNS. This saves the remote end the overhead of trying to look it up if it would otherwise do a reverse lookup, for instance for the sake of verifying the address of the source. The remote end may of course choose to reject any packet with the D Bit set. This bit is transmitted by the source host as 0, and set by any nl-router that knows that the source cannot be looked up in DNS.

Source FQDN Changed ("C" Bit) This bit may be set if the Source FQDN and/or Source Expat FQDN fields have been modified in transmit. This would typically be done by a border router for the sake of privacy. This bit must be passed up to the higher layers along with the FQDN itself. It alerts the application that the remote host may not recognize the supplied FQDN as its own, for instance in case the application protocol wished to make use of the FQDN. Of course, the fact that the Source FQDN has changed at all may be private information. Therefore the setting of this bit even when the Source FQDN has changed is entirely optional.

Rsvd: Reserved. Transmit as 0, ignore upon receipt.

Source Expat FQDN Offset The offset of the Source Expat FQDN, in units of bytes, counting from the 0th byte of the FQDN header. This value will be zero if the source host is not an expat (and there is therefore no Source Expat FQDN).

Dest Expat FQDN Offset The offset of the Dest Expat FQDN, in units of bytes, counting from the 0th byte of the FQDN header. This value will be zero if the source host is not an expat.

Source FQDN Offset The offset of the Source FQDN, in units of bytes, counting from the 0th byte of the FQDN header. This value will be zero if neither the source or dest hosts are known to be expats.

Dest FQDN Offset The offset of the Dest FQDN, in units of bytes, counting from the 0th byte of the FQDN header. This value is set to 0 if there is no Dest FQDN.

Source Expat FQDN The FQDN of a zone in the source host's expat realm. It is encoded as one (ASCII) byte per character, starting with the "left-most" character. This field is null if the source host is not an expat.

Dest Expat FQDN The FQDN of a zone in the dest host's expat realm. It is encoded the same as the Source Expat FQDN, and is null if the dest host is not known to be an expat. It immediately follows the Source Expat FQDN if there is one, or the Dest FQDN Offset field if there isn't. In other words, there is no padding between FQDNs.

Source FQDN The FQDN of the source host. It is encoded the same as the Source Expat FQDN. It is null for "Ack-only" FQDN Header Options.

Dest FQDN The FQDN of the destination host. It is encoded the same as the Source Expat FQDN. It is null for "Ack-only" FQDN Header Options.

Zero Padding The Dest FQDN is followed by zero or more bytes of value 0, to pad the header out to an integral 32-bit boundary.

4.4 "Missing" Fields

Of the commonly used fields of IP, IPNL is "missing" the Header Checksum, Time to Live, fragmentation fields, and the Type of Service field.

IPNL takes a similar approach regarding the header checksum as IPv6. That is, the important and unchanging IPNL fields are protected either by their direct incorporation in the TCP or UDP checksum as a pseudo-header (see section /refencap), or in checks against state stored by the host IPNL layer.

IPNL has no Time to Live or Type of Service because it uses those of IP itself. Specifically, when an nl-router forwards a packet, the Time to Live and Type of Service of the transmitted IP header must be set to that of the received IP header. ICMP Time to Live Expired messages are translated into IPNL-ICMP Time to Live Expired messages, which are sent back to the source host where the IPNL layer modifies the setting of the IP layer's Time to Live if necessary.

IPNL handles fragmentation/reassembly identically to IPv6:

- nl-routers cannot fragment or reassemble packets.
- Hosts can fragment packets at the IPNL layer, but are strongly encouraged to do PMTU discovery instead.
- when hosts fragment, the fragmentation information is carried in a options header.

IP packets transmitted by all IPNL boxes must have the DF flag (Don't Fragment) set to 1.

5 Extension Headers

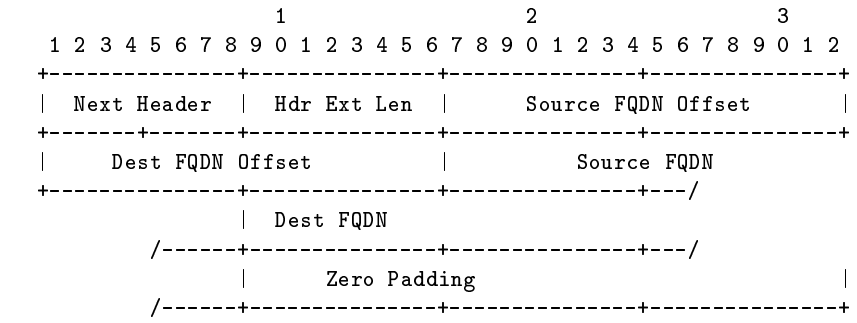
IPNL copies IPv6's extension header method of attaching optional headers to IPNL (with the exception of the MRIP and FQDN headers). IPNL includes the same extension headers as IPv6, plus more pertinent to IPNL. What's more, IPNL where possible uses the same formatting as IPv6, and has the same ordering rules.

The headers copied from IPv6 will be added to this spec in a later version. The headers unique to IPNL described below.

5.1 Additional FQDN Extension Header

This header is used by a host to inform a remote host of any additional FQDNs that may be used to identify it. It is commonly used in an anycast application to allow a remote host to know which host of a group of anycast hosts it is talking to.

It is formatted as follows:



The fields are defined as follows:

Next Header The header type of the header immediately following this one. The values for this field are as defined by the IP (and IPNL) Protocol field.

Hdr Ext Len The length, in units of 32-bit words, of this extension header.

Source FQDN Offset The offset of the Source FQDN, in units of bytes, counting from the 0th byte of the FQDN header. This value is set to 0 if there is no Source FQDN.

Dest FQDN Offset The offset of the Dest FQDN, in units of bytes, counting from the 0th byte of the FQDN header. This value is set to 0 if there is no Dest FQDN.

Source FQDN The Source FQDN is encoded as one (ASCII) byte per character, starting with the "left-most" character. This field is null if no Source FQDN is included.

Dest FQDN The Dest FQDN is encoded the same as the Source FQDN, and is null if no Dest FQDN is included. It immediately follows the Source FQDN if there is one, or the Dest FQDN Offset field if there isn't. In other words, there is no padding between FQDNs.

6 Overview of Unicast Operation

This description assumes hosts natively running IPNL. When a host is not running IPNL, a proxy operates IPNL on its behalf. Proxy operation and other transition issues are described in section /refsecTrans.

6.1 IPNL Names and Addresses

The core name in IPNL is the DNS name, refered to as the FQDN. The FQDN is overloaded in the same sense as the IP address in that it is both a global identifier and globally routable. It is the stable name used in all cases where the identity of or route to/from a host may be ambiguous or unknown. In addition, the transport pseudoheaders of all packets contain the FQDNs (and not any other address information).

As with DNS, FQDNs are grouped into zones. Zones in IPNL are fully valid DNS zones in the sense that a single domain name server has pointers to either i) all records for the zone or ii) name servers for all lower level zones.

All bottom level zones (zones with no lower level zones) are assigned to a realm, known as the home realm. Typically a zone has a single home realm, though it may be associated with multiple realms as long as all of its member FQDNs are represented in the realm. In this case, the zones become anycast zones. A bottom level zone may not be split between multiple home realms.

Each realm has an FQDN associated with it that i) is not the FQDN of any individual host, and ii) is routable (that is, a packet with the realm FQDN can be routed to the realm from anywhere). This is called the realm name.

An individual host does not necessarily need to be attached to its home realm. If it is not, then it is known as an expat host. Multiple hosts can be expats for the same FQDN, in this way providing anycast functionality. A host can change realms during the course of a single connection, making the expat mechanism suitable for providing mobility as well.

There are two ways that an expat host may be reached. First, the realm name of its adopted realm may be carried in packets along side the host's FQDN. Routing is based on the realm name until the packet reaches the adopted realm, after which it is based on FQDN. Additionally, the individual host's FQDN may be advertised into the (local) routing protocol.

Every realm may also have a realm identifier. This is an FQDN that uniquely identifies the realm globally, but is not necessarily globally routable. It is advantageous that this FQDN not come from any zones attached to the realm (or any other realm). This way the realm identifier may remain valid even when the zones in a realm change. The realm identifier is used as a simple identifier in routing protocols and other management tools.

Every host interface has an one or more IP addresses. These IP addresses are used in the IP header of all packets entering and leaving the interface, as with standard IP. These IP addresses also serve as the low-order component of all IPNL addresses.

There are two types of IPNL address, the local address and the global address. Typically a given IP address for a given host interface will have one local address and zero, one, or more global addresses associated with it. (There may be more than one local address, but it is not particularly advantageous.)

The local address consists of a realm number and an IP address. Globally the realm number is a reused number. It is unique within the scope of a realm group, where a realm group is somewhat circularly defined as a group of realms with internally unique realm numbers. (An additional constraint on a realm group is that all realms in a realm group be reachable via paths internal to the realm group. If a realm group is partitioned, packets from one partition may still be routed to another partition. They must, however, use global addresses.) The realm number used in a local address is called a local realm number.

A realm group may import routing information for zones in other realm groups. When it does, it assigns a local realm number to the the home realm of the external zone. When a locally addressed packet passes from one realm group to another realm group, the realm numbers in the packet header are changed from their local values in one realm group to their local values in the other realm group. Realm numbers of packets that stay within a realm group do not change. Local realm numbers are called internal or external depending on whether the realm is in the same realm group or in another realm group.

The global address consists of a middle realm IP address (MRIP), a realm number, and an IP address. All

unicast MRIPs are unique among other MRIPs (all middle realm interfaces have a unique IP address). (Anycast MRIPs are of course not unique.) Unicast MRIPs, however, are not unique among all IP interfaces. They can be reused in a private realm. There is no notion of private addresses [RFC1918] in IPNL. Any IP address can be used both in the middle realm and in private realms.

The realm number in the global address identifies a realm that is reachable behind its MRIP. It is unique only among other realms reachable behind the MRIP. Realms may, and typically will, have multiple global addresses, one for each MRIP they are reachable through. The realm number for each such global address is not related to the others. When a packet passes through a frontdoor nl-router from the middle realm, the local realm number of the destination realm is written into the packet. Thereafter the local realm number is used for routing by interior nl-routers. The only nl-router that ever uses the realm number in the global address is the nl-router attached to the corresponding MRIP.

The only address component that a host is configured to know is its own IP address. All other address components are learned on an as needed basis, and forgotten as soon as they are no longer needed. The mechanism for learning of the other address components is that the components are written into the packet as it makes its way through the internet such that they are completed by the time they reach the destination. The destination then copies the received source address into the destination address fields on return packets. A host learns its own address through reception of packets returned to it from other hosts. This approach allows global addresses to change frequently, and allows multiple global addresses to be used for the same host pair.

By the same token, hosts also dynamically learn the addresses of other hosts during the course of their communications. If only one frontdoor nl-router is used in talking to another host, then only one address will typically be learned. If multiple nl-routers are used (for instance because one crashed), then multiple addresses will be dynamically learned. There are mechanisms for learning new addresses and flushing no longer valid addresses on the fly.

A globally addressed packet may carry up to three global addresses and one local address. The local address is needed for the destination address while the packet is in the destination realm group, and for the source address while the packet is in the source realm group. This is necessary because interior nl-routers do not maintain routing information about global addresses, except for a default route to the middle realm.

One of the global addresses is the source address as set by the source host itself. This address is used to identify the source of the packet. Specifically, the address is used to lookup the FQDN of the source. This in turn is checked against the transport pseudoheader.

The second global addresses is the destination address. Upon receiving a packet, a host does not use the destination address to determine if the packet has been delivered correctly because a host does not assume that it knows its current global address (it may have changed recently). Rather, it uses the transport pseudoheader to weed out misdelivered packets. It does use the destination global address of the latest received packet, however, as the source global address in return packets.

The third global address is the source address as set by the egress frontdoor nl-router of the source host's realm group. This represents the actual path taken, is typically used as the destination address in return packets, and is subsequently used to identify received packets.

6.1.1 Notes on Other Design Choices

Conveying MRIPs

The use of four addresses in a single packet may seem excessive, so we try to justify that design choice here.

First, we point out that this is only the case for globally addressed packets. Locally addressed packets have a single source and destination address only.

Considering globally addressed packets only, we could eliminate the use of the local address if we allowed interior nl-routers to know about MRIPs and the realm number assignments they have made. Doing this would further

eliminate the need for the Loc field, because interior routers could use the Dest MIRP field to determine if a packet was in the destination realm group or not. While this would complicate realm group routing, it could simplify other operations, such as certain kinds of debugging, because packets would be less context sensitive. This approach should probably be given more consideration. (Note, however, that this approach only saves 2 bytes in the header.)

The reason we have so chosen not to take this approach is to try to isolate internal site network operation from the global internet as much as possible. A specific design goal is to limit the effect of global addressing to the border nl-routers to the extent possible. For these reasons, even though configuring interior nl-routers with border MRIP and realm number information is perfectly feasible (for instance, it could be piggy-backed on routing information), IPNL here errs on the side of site isolation.

The use of three global addresses rather than just two is motivated by i) the desire to keep the border nl-routers as simple as possible, and ii) the desire to minimize load on the global DNS infrastructure.

To see why this is the case, consider an approach that does not write the actual source MRIP used into the packet header. Such an approach would require that out-of-band messages be sent to hosts to inform them of the full set of addresses for a remote host (because they would not be dynamically learned in IPNL packet headers). This would need to include both the MRIP and the realm number. This in turn requires either that:

- DNS maintain both MRIP records (which it does today in the form of A records) and records for realms (which would be a new function), or
- nl-routers themselves maintain this information for all other nl-routers attached to the same realm group, and themselves pass the information to remote hosts, possibly piggybacked on the initial return packet.

The former approach would require a DNS lookup in each direction of communications, more-or-less doubling its load. The latter approach involves that much more coordination and configuration between nl-routers.

Further, there would need to be some mechanism for informing hosts when new addresses for remote destinations became available (for instance, because a new frontdoor nl-router was installed). What would happen here is that, when a host received a packet from a remote host through the new frontdoor, it would not recognize the source address (i.e., that of the new frontdoor). It would then return an error message to the source telling it to attach an FQDN to subsequent packets (or to send an out-of-band packet with the FQDN if the FQDN would cause the normal packets to segment). A subsequent packet would contain the FQDN, allowing the host host to identify the remote host and learn the new address. (Unless of course that packet didn't in fact go through the new frontdoor, for instance because packets were being load-shared between two frontdoors. So we'd probably have to outlaw such load splitting, or something.)

This is not to say that an alternative approach isn't possible, but that it adds a lot of protocol and configuration complexity at a savings of 6 bytes per packet. Again, one might reasonably choose to make this trade-off, but for now we err on the side of operational and protocol simplicity.

As long as we are on the topic of alternative design choices, it is worth mentioning that the current design is the 5th or 6th attempt. The earliest design had no realm numbers or MRIPs, but rather simple 64-bit tags that were mapped hop-by-hop. Realm numbers was an early modification, to simplify site operations. Several iterations, however, tried to make use of simple 32-bit tags to identify realm groups (rather than actual middle realm addresses).

The first of these attempts tried to hide the complexity of multiple frontdoors from hosts. These approaches, however, resulted in extensive coordination required between border nl-routers. This coordination was especially intense when border nl-routers crashed and lots of update activity ensued. Latter attempts at pulling some of the complexity from nl-routers and pushing it to hosts helped, but still resulted in unsatisfyingly complex designs.

Support of Mobile Devices

Another interesting design choice would be to increase the size of the EHIP fields from 32 bits to 64 bits. This would allow IPNL to subsume networking technologies other than just IP. The most important example of this would be to incorporate the [XXXX ref needed] numbers used for identification by mobile telephone devices. This could make auto-address assignment for these devices easier.

This would considerably expand the scope of IPNL, beyond simply extending IP. While this might be useful, on the other hand it seems that the long-term goal of mobile devices is to get IP addresses, so perhaps better to just keep IPNL simple and let other technologies conform to it, so to speak.

Separate DNS Lookups

In the current design of IPNL, initial packets are overloaded in that they serve both to carry application data and to trigger a DNS lookup. An alternative design would be to have the host send a DNS lookup before transmitting any data packets. This would trigger the DNS lookup at the frontdoor nl-router.

The advantage of this approach is that the frontdoor nl-router does not have to store the data packet while it is waiting for the DNS answer. This advantage is probably as not as strong as it may seem. Since the frontdoor nl-router needs to maintain state for the DNS lookup itself, the complexity of its operation is not significantly reduced by using a separate DNS lookup packet. Since the first packet sent by a host is typically small (a TCP SYN, for instance), the extra overhead of holding a packet is probably not excessive.

The disadvantage of this approach is that it forces a DNS lookup on locally addressed packets as well as globally addressed packets, since a host does not necessarily know in advance whether a destination is global or local. As a result, connections that turn out to be local will suffer an unecessary delay in delivering the first packet.

On the other hand, another downside of the overloaded approach is that any application that starts sending lots of data without any kind of initial handshake will force the frontdoor nl-router to either store a lot of big packets, or drop a lot of packets. Such applications would have to recognize this problem and either do a handshake or at least start slow. On the other hand, getting the frontdoor nl-router to drop packets for this kind of application might be seen as a feature (ala RED) since sending a burst of packets into the network is not friendly to other connections.

This issue deserves further consideration, but for now IPNL errs on the side of overloading initial packets.

6.2 Maintained Databases

This section outlines the databases that must be continuously maintained by various boxes.

Nl-routers attached directly to realms should maintain a database of FQDN to IP address mappings for hosts in the realm. This can best be done by configuring them to be stub domain name servers for the zones in the realm. They would get zone transfers from the DNS servers in the realm (or, operate as primary or secondary DNS servers themselves). If they can't maintain a full database, they must at least know how to query the DNS servers within their attached realms.

Nl-routers attached to realms must know the zones associated with the realm, and the realm identifier. They must know the realm names of the current realms for each expat host from their realms. They must also know the FQDN to IP address mapping for every expat host from other realms currently in one of their realms. This expat information is obtained through a separate protocol similar to the protocol used to communicate with local and home stations in mobility protocols.

All nl-routers must maintain a database of routes to all locally addressable realms and their associated realm numbers and zones. In addition, they must maintain routes to all individual expat FQDNs for which local anycast functionality is required. It is envisioned that a BGP-like protocol (almost certainly a modification of BGP itself) would be used for this purpose. This is primarily because the mechanisms for passing routing information across realms has similarities with that of passing routing information across autonomous systems. This routing protocol must have the ability to "hole punch", that is, route by longest FQDN "postfix". (A postfix is a prefix for the "righthand" end of a string.)

It is envisioned that the assignment of local realm numbers will be manual and static. Nl-routers attached to realms must originate routing updates for their attached realms. Border nl-routers must originate routing updates for realms in other realm groups.

All non-frontdoor nl-routers must maintain a database of default routes (that is, routes to frontdoor nl-routers). It is envisioned that this will be part of the same BGP-like routing algorithm, with all frontdoor nl-router originating the default routing updates.

Frontdoor nl-routers must maintain a database of realm numbers associated with each of their MRIPs for each realm reachable through them. It is envisioned that the assignment of these realm numbers will be automatic, since the nl-router does not need to coordinate the assignment with any other boxes.

Some middle realm-attached box must maintain a database of zone to MRIP address mappings for the realm groups they are responsible for. This could either be done by modified DNS servers, or preferably by the frontdoor nl-routers themselves. In the latter case, the DNS servers for the realm group would be configured to view all of the nl-routers for the realm group as primaries for the zones in the realm group. The nl-routers would get zone transfers from the DNS servers and this way learn about all the other nl-routers. The nl-routers would have the ability to return DNS answers containing MRIPs to queries for individual hosts within their realms.

Note finally that all IPNL boxes must know where each of their interfaces connect: to a realm within the same realm group, to the middle realm, or to a backdoor realm. If a given interface reaches multiple different realms (for instance by virtue of being connected to a LAN whose routers belong to different realms), the IPNL box must be able to distinguish the realms reachable over that interface.

6.3 Autoconfiguring Hosts

Before a host can transmit an IPNL packet, it must know the following:

- Its own IP address
- Its own FQDN
- The IP address of an nl-router for its realm

The latter can be learned through DNS without any apriori configuration information, as described in the transition section below. A summarization of the basic approach is as follows. Before an IPNL host learns any nl-routers, it uses DNS. Whatever IP address DNS returns for a given destination, the host transmits IPNL pings to determine if the box at the IP address is IPNL-capable. If the box is an nl-router for the host's private realm, then the nl-router will inform the host of this.

The host can of course learn its own IP address through mechanisms in existence today (DHCP etc.). Not all hosts today have FQDNs, for instance home PCs that access the internet through dial-up modems. Even where hosts today do have an FQDN, they are typically configured manually.

Since autoconfiguration is an increasingly important requirement, the remainder of this section discusses autoconfiguration of FQDNs and IP addresses.

Because FQDNs can be almost arbitrarily large, autoconfiguration of unique (if not necessarily very human friendly) FQDNs is as a general rule straightforward. Without getting into details, dial-up modems at an ISP can assign FQDNs to subscribers at the same time they assign the IP address, but without ever using the same FQDN twice. Or, the FQDN could include the assigned IP address.

As another example, a host without an FQDN in a private realm could be assigned one by the first nl-router it contacts. The nl-router could then update DNS and the other nl-routers in the private realm.

As yet another example, small devices could have unique FQDNs burned into them at the factory. Such an FQDN might consist of a serial number plus the domain name of the manufacturer. If the device never had to talk outside its local environment (the dentist's office), the burned-in FQDN would suffice. If it did, the burned-in FQDN could be concatinated with an FQDN suffix by an nl-router to create a globally routable but still unique FQDN.

Autoconfiguration of IP addresses is not as easy as autoconfiguration of FQDNs because there is not infinitely many of them in any given environment. IP address autoconfiguration in an IPNL environment, however, is easier than in today's IP-only environment, for the following reasons:

- Because a given realm can assign IP addresses from the entire IP address space (not just the RFC1918 private addresses), the pool to assign from (for instance, by a DHCP server) can be made much larger.
- Because a device can already have a burned-in unique FQDN, identification of nodes in the context of a distributed advertise-and-challenge algorithm for assigning IP addresses is made easier [NoDHCP].

Autoconfiguration of IP addresses in an IPNL environment is still not as easy as autoconfiguration of IPv6 addresses where unique host identifiers are available. Never-the-less, it is not a tremendously hard problem.

6.4 Setting Address Fields

When a host I is to initiate communications with another host R, host I will start with one of the following:

- The FQDN of host R only. This is the typical case when host I has not exchanged packets with host R before.
- The FQDN and one or more complete addresses for host R. This is typically the case where host I has exchanged packets with host R in the recent past, and has remembered the learned address information.
- The FQDN and one or more MRIP addresses for host R. This is typically the case where host I has not exchanged packets with host R, but has done a DNS lookup for the MX record for another host R' for which host R is the mail server. In the case where host R is globally addressed, the DNS answer contains the FQDN and MRIP of host R (see Section 6.4.1).
- The address of host R but not the FQDN. This would typically be for network monitoring or other debugging purposes.

Whenever host I has not recently received a packet from host R, it must attach the FQDN header to the packet. (Recently here is defined in terms of minutes.) If it has partial or full address information for host R, this is written into the appropriate destination address fields as well. If host R is no longer at the supplied address, this will typically be learned via an IPNL-ICMP destination unreachable message, usually transmitted by an nl-router.

In what follows, we assume the case where host I starts with only the FQDN of host R. The first packet sent by host I contains host R's FQDN, but the destination address information for host R (EHIP, realm number, and MRIP) are initially in the "unset" values. Except for the source EHIP address, the source address information is also in the "unset" values. The Loc field is set to "unknown".

Nl-routers use the dest FQDN to route the packet, and do filtering on the source FQDN to prevent spoofing. They also fill in the unset address fields as the packet makes its way to the destination.

The first nl-router to receive the packet knows whether the packet is globally or locally addressable. This is because it either has an explicit forwarding table entry for the destination FQDN, in which case the packet is locally addressable, or not, in which case it is assumed to require global addressing. Either way, this nl-router writes the local realm number of the source realm into the Local/Used Source Realm field.

If the packet requires global addressing, the Loc field is set to "default route", and the packet is forwarded towards the middle realm. The egress frontdoor border nl-router, possibly after doing a DNS lookup over the middle realm, will fill in the Dest MRIP field and transmit the packet to that MRIP. It also sets the Loc field to "middle", and fills in the Used Source MRIP+NR fields. Finally, if the "M" bit of the FQDN header is set, the frontdoor transmits the list back to the source host.

The Global Dest Realm field remains in the unset value until the packet reaches the ingress frontdoor border router. This border router sets the Loc field to "route by realm", fills in the Global Dest Realm and Dest Realm fields, and forwards the packet towards the dest realm. Once the packet reaches the dest realm, the ingress interior router of the dest realm is able to fill in the dest EHIP address (possibly but preferentially not by doing a local DNS lookup), thus completing the global dest address. This whole process is summerized in the following picture.

```
        |<----Source Realm Group------>| MR  |<-----Dest Realm Group---->|

                  first      egress          ingress    last
         Host I   interior   frontdoor       frontdoor  interior   Host R
                  nl-router  nl-router       nl-router  nl-router
         ------>  -------->  -------->       -------->  --------> -----|
Loc      unknown  default    default  (mid)  realm      realm     realm

D EHIP   unset    unset      unset      ||   unset      unset      set
LD RN    unset    unset      unset      ||   local      local     local
S EHIP   set      set        set        ||   set        set        set
LUS RN   unset    local      global     ||   global     global    global

GD RN    unset    unset      unset      ||   global     global    global
D MRIP   unset    unset      global     ||   global     global    global
GS RN    unset    unset      unset      ||   unset      unset     unset
S MRIP   unset    unset      unset      ||   unset      unset     unset
NS MRIP  unset    unset      global     ||   global     global    global
```

Note that the source MRIP+RN remain unset. The receiving host learns the actual global source address from the Used Source MRIP+RN.

If the packet is determined by the first nl-router to be locally routable, that nl-router will write the Loc field to "route by realm", and leave all of the global fields at unset. It will also set the Local/Used Source and Dest Realm numbers to their appropriate locally known values. If the destination is in the same realm group as the source, the source and dest realm numbers will not change thereafter. The dest EHIP address is filled in by the last nl-router as with the example above.

If the destination is in a different realm group, the source and dest realm numbers will be changed. If the dest realm group is reachable via a direct backdoor connection, the border nl-router will rewrite the realm numbers from the local values used by the sending realm group to those used by the receiving realm group. The values are known either through a routing protocol exchange between the realm groups, or by manually configured entries. This rewriting takes place for every realm group boundary crossed, though typically only one boundary is crossed.

It also takes place over the middle realm. That is, the egress frontdoor router will change them to "global" values under the source and dest MRIPs, and the ingress frontdoor router will change them again to the local values for the destination realm group. The packet is still considered to be locally addressed, however, with the Loc field remaining as "route on realm". (As this implies, IPNL boxes must always know what realm any given packet came from.) This is illustrated as follows:

```
         |<----Source Realm Group----->|  MR    |<-----Dest Realm Group--->|

                   first      egress             ingress    last
          Host I   interior   frontdoor          frontdoor  interior  Host R
                   nl-router  nl-router          nl-router  nl-router
          ------>  -------->  -------->          -------->  --------> -----|
Loc       unknown  realm      realm      realm   realm      realm     realm

D EHIP    unknown  unknown    unknown            unknown    unknown   set
LD RN     unknown  local1     local1     global  local2     local2    local2
S EHIP    set      set        set                set        set       set
LUS RN    unknown  local1     local1     global  local2     local2    local2

(all MRIP fields remain unset)
```

The reason it is necessary to temporarily use global addressing across the middle realm is that the receiving ingress nl-router may have a large number of different realm groups behind it. Each realm group will have its own local realm numbering which will not be consistent with that of the other realm groups. The only consistent realm numbering across those realm groups will be that defined by the ingress nl-router's MRIP. (Note as well that tunneling over the middle realm can be used as well. In this case, the tunnel behaves identically to a direct backdoor link.)

In both the globally and locally addressable cases, the packet addresses will be completely filled in by the time they reach the destination. Host R (the destination host) can now bind host I's address and FQDN. Although host R could at this point successfully return a packet without the FQDN attached, its initial return packet must also contain the FQDN header so that host I can bind host R's address and FQDN.

Host R composes the return packet as follows. Host R can determine if the packet is locally or globally addressed by examining the Dest MRIP field. If it is at value "unset", then the packet is locally addressed. Otherwise the packet is globally addresed.

If the packet is locally addressed, Host R sets the Loc field to "route by realm". It swaps the Source and Dest EHIP fields. It swaps the Local/Used Source Realm and Local Dest Realm fields. It writes all other fields to the "unset" value. (If the FQDN is not being attached, then the global header is not attached and there are no other fields.)

Note that if either realm number changes during the course of a locally addressed "connection", the packet will not be delivered. The transmitting host will receive a destination unreachable, and will attach the FQDN on subsequent packets (with the Local/Used Source Realm and Local Dest Realm fields unset). This establishies the new realm number.

If the packet is globally addressed, it sets the Loc field to "default". It swaps the Source and Dest EHIP fields. It writes the Local Dest Realm field to "unset" (this is always the case for globally addressed packets, since they will be default routed anyway). The Used Source MRIP+RN fields are also written to "unset" (they also always start this way). The Dest MRIP+RN fields are set to the same values as the received Used Source MRIP+RN if not unset, and the Source MRIP+RN otherwise. (They can also in theory be set to any previously received valid address, but typically in practice they should be set to the latest received one.) Finally, it sets the Source MRIP+RN to the Dest MRIP+RN of the previously received packet.

Either way (local or global) the return packet contains complete source and destination addresses. The destination address is sufficient to get the packet routed to the destination. To inhibit spoofing, however, Nl-routers still use the destination FQDN to route the packet, and use the source FQDN to do source filtering. The exception to the former is that egress frontdoor nl-routers won't verify that the return dest MRIP matches that given by DNS unless explicitly requested to do so by the host.

Once both ends have their respective bindings, they may send packets without the FQDN header.

When a host returns a packet, if the remote host is in the same realm, it sets the destination IP address to the Source EHIP of the received packet. It can tell that the remote host is in the same realm because i) the IPNL address is local, and ii) the Local Dest Realm and Local/Used Source Realm fields are identical. If the remote host is in a different realm, it typically sets the destination IP address to the source address of the latest received packet. If the nl-router at that address is down (as determined by the receipt of an ICMP destination unreachable message or lack of response from a ping after a timeout), it uses the IP address of any other nl-router it knows about.

The transport pseudoheader contains the FQDNs but not the addresses. Hosts must continue to store the established address to FQDN binding in order to identify incoming packets. This is true for both TCP and UDP transports. The address is allowed to change during the course of a "connection" (using the word loosely here to include a series of UDP packets), but the FQDN must not.

6.4.1 DNS Lookups for MX and other Third Party Records

Certain (IPv4) internet applications use DNS to learn about services available on other "third party" hosts, for instance by looking up the MX record of a host to learn its mail server. In these cases, DNS answers with both the FQDN and IP addresses of the third party host. By returning the IP address along with the FQDN, a second DNS lookup is unecessary.

IPNL is also able to avoid a second DNS lookup. In the case where the third party host is local to the initiating host, the second DNS lookup is not required because no DNS lookup is necessary for locally addressed packets. In the case where the third party host is not local to the initiating host, the second DNS lookup is avoided by returning an MRIP for the third party host in the answer of the original DNS lookup.

The MRIP is encoded as a new DNS record, here called the MRIP Record (until a permanent name for it is decided). As described below, this new record is supplied by nl-routers rather than by DNS servers.

Third party DNS lookups in IPNL operate as follows. Assume that a source host S is searching for a third party host P for a target host T. (In other words, in the case of MX records, P would be the mail server for host T.) Host S starts by transmitting the original DNS query over IPNL. The Dest FQDN field of the FQDN header contains the FQDN of the target host T. The Any field of the FQDN header is set to "Last nl-router" (4), meaning that the packet should be delivered to an nl-router attached to the home realm of host T.

When such an nl-router receives the packet, it is either able to answer the DNS query immediately (because it is an authoritative server or secondary server for host T's zone), or able to query the realm's DNS to obtain the answer. The answer will contain the FQDN and IP address of the third party host P.

Next, the nl-router transmits the DNS answer to host S as follows:

1. If host P is in a different realm than host T, then the DNS answer is transmitted as is.
2. If host P is in the same realm as host T, and host S is locally addressable, then the DNS answer is transmitted as is.
3. If host P is in the same realm as host T, and host S is globally addressable, then an MRIP Record is added to the DNS answer. The MRIP Record is set to the Dest MRIP of the IPNL packet received from host S (that is, the packet that contained the DNS query).

When host S receives the answer, it now knows the FQDN of host P and can transmit packets to host P. The Dest EHIP of the packet to host P may contain the IP address returned in the DNS answer. If the MRIP Record was returned, then this is written into the Dest MRIP field of the packet. All other fields are set as usual for the first packet transmitted to a remote host.

The only case where a second DNS lookup would be required is the case where the third party host was both i) in a different realm from the target host, and 2) globally addressable by the source host.

6.4.2 Packets with Address but No FQDN

It is perfectly legal to transmit an initial IPNL packet with a full address and no Dest FQDN attached. Obviously the address would have to be learned through some mechanism outside of IPNL.

The target application on the destination host would also have to configure IPNL to accept such packets. The default behavior of hosts is to reject packets for which FQDNs have not been established.

An example of an application that might accept packets with no FQDN is network management software.

6.5 Dealing with Topology Dynamics

In discussing how to deal with topology dynamics, such as nl-routers crashing, once again there are differences between the locally addressed and globally addressed cases.

The locally addressed case is similar to typical dynamic routing today. Nl-routers constantly maintain forwarding information to all (locally addressed) destinations that they care about. If an nl-router crashes or some link goes down, the dynamic routing algorithm will route around it. No modifications to the addresses used is necessary. As such, the end hosts will not notice changes in the paths taken by packets.

In the case of globally addressed packets, the address itself determines which frontdoor nl-router is used to enter a realm group. When frontdoor nl-routers change, including failure induced changes as well as nl-routers being brought in and out of service, the addresses used by hosts must correspondingly change. This is a significant change from how hosts work today, and much of the design of IPNL deals with how to manage frontdoor nl-router dynamics in a simple and scalable way.

To the extent possible, IPNL does this by allowing regular data packets to discover frontdoor nl-router changes and convey these changes to hosts. Explicit "out of band" actions are rarely needed. When a frontdoor nl-router is added or deleted, this is automatically conveyed to the realm group dynamic routing protocol. When a packet is transmitted by a host, it will be routed to a new frontdoor. The frontdoor will tag the packet with its MRIP (in the Used Source MRIP), thus conveying the change to the remote host. The change is subsequently echo'd back in the Dest MRIP of a return packet.

This mechanism obviously works only when a packet is transmitted by a host behind the changed frontdoor nl-router. Until the change is conveyed to the remote host, packets from the remote host will continue to be routed to the same frontdoor nl-router.

Most unicast applications have periodic "keep alive" packets running in both directions (in the absense of application packets and their acknowledgements). These keep alives serve to discover and convey most changes in frontdoor nl-routers as described above. They won't necessarily discover multiple simultaneous changes, for instance where the frontdoor nl-routers on both ends crash at the same time.

To deal with both applications that don't have keep alive traffic and simultaneous failures, the IPNL layer in a host assumes that it should receive a packet from a remote host within a reasonable time after it has transmitted a packet to the remote host. The timeframe here is on the order of 10 or 20 seconds. If the IPNL layer does not receive a return packet in that timeframe, or if it receives a destination unreachable for the remote frontdoor nl-router, it sets out to discover a working remote frontdoor nl-router.

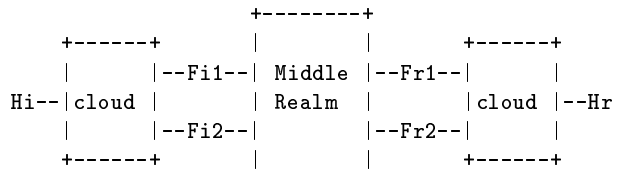
If the host already has other MRIPs for the remote host in its list, it tries these by transmitting echo packets to the remote hosts with these MRIPs as the Dest MRIP. If the host doesn't have other MRIPs, or all of the ones it has are not working, the host generates a DNS query for the remote host. Normally the answer to this query will still be cached at the host's frontdoor nl-router, so middle realm DNS is not burdened with additional DNS lookups. Even if the actual requested record is not cached at the frontdoor nl-router, the intermediate

results of the previous DNS lookup may be cached, so the root DNS servers do not need to be re-queried. As such, this approach does not result in a substantial increase in load on middle realm DNS, and can be pursued relatively agressively. In other words, the timeout can probably be relatively small (closer to 10 than to 20 seconds, though this obviously needs to be tested).

Note that the frontdoor nl-routers do almost nothing but blindly forward packets received from hosts. The only exception to this is that frontdoor nl-router may cache destination unreachables before passing them on to hosts, and use these as hints in deciding which of multiple MRIPs to use when forwarding the initial packet from a host.

Note also that a host can always request the complete DNS answer in advance of any failure in communications. Most of the time, however, this is not necessary and would only amount to extra overhead, especially for very shortlived TCP connections.

The following describes the above process in more detail. The explanations are based on the following topology: Assume that packets are already flowing between host I (Hi) and host R (Hr) via frontdoor nl-routers Fi1



and Fr1. Denote the MRIP+RNs used by Fr1 and Fi1 to be Ar1 and Ai1 respectively. Assume also that the initiating host did not request a list of MRIPs from the frontdoor nl-router, and so both hosts know only a single address for the other.

Lets say that Fi1 crashes. Consider first the simpler case of packets from Hi. Dynamic routing in Hi's cloud will, sooner or later, automatically route a packet to Fi2. This packet will contain the source MRIP+RN Ai1, which Fi2 will recognize to be different from its own. Fi2 therefore writes its own MRIP+RN (Ai2) realm (Ai2) into the Used Source MRIP+RN fields.

This packet contains the MRIP of Fr1 in the dest MRIP field and so is routed to Fr1 and on to Hr. Hr adds the new address Ai2 to its list of addresses valid for Hi, and uses Ai2 in subsequent return packets. These are routed to Hi via Fr1 and Fi2. Here, flow between the two hosts is reestablished in roughly the time it takes for the dynamic protocol to respond without requiring any additional DNS lookups or coordination between frontdoor nl-routers. This method works as long as periodic packets are being transmitted from Hi to Hr, for instance as would be the case with most TCP connections.

Now consider the more complex case of packets from Hr to Hi. These will not reach Hi, and will not be automatically rerouted to Fi2 because there is no dynamic routing across the middle realm. As a result, Hr will eventually time out by virtue of not having received a packet back from Hi.

Hr marks address Ai1 as being unreachable. Since Hr has no other MRIP+RN's for Ai1, it transmits a DNS request to its default frontdoor nl-router (Fr1 in this case). This is done by setting the Any field of the FQDN header to value "First border nl-router" (2). Fr1 has no DNS entry for Hi, and so does a DNS query for Hi. It stores the result (the MRIPs for Fi1 and Fi2) and passes this on to Hr.

Hr knows that Fi1 is unreachable, and so composes the next packet with the Dest MRIP set to Fi1's and the Dest Realm Number unset. The Source MRIP+RN is set to the same value in use (Fr1). The FQDN is attached. Fi2 fills in the realm number and passes it on to Hi. Hi's return packet uses address Ai2 as the Source MRIP+RN, but does not need to contain the FQDN.

In the above example, Hr determined that Hi was unreachable via Ai1 because of a timeout. It is possible, however, that Hr could have learned that Hi was unreachable much more quickly—by receiving a destination

unreachable. Hr's response depends on exactly what is unreachable. If either Hr's own default frontdoor (Fr1) or an nl-router behind Fi1'is unreachable, Hr should wait for the respective intra-realm group routing protocol to respond, but otherwise not change its use of address. If the unreachable shows that in fact Fi1 is down, Hr should still wait, because there is a good chance that Hi will transmit a packet and convey Ai2 to Hr. Only if this does not happen within the timeout period does Hr initiate the DNS lookup.

If both Fi1 and Fr1 go down at the same time, Hr will not receive packets from Hi even if Hi transmits. The same is true in the reverse direction. In this case, both hosts will timeout as described above and initiate DNS requests, though one host may timeout enough before the other to have done a DNS request and reestablished connectivity before the other host times out.

6.5.1 Other Design Choices

Recovery from frontdoor nl-router failure works best when the host behind the frontdoor nl-router transmits a packet, and in practice this is how most such failures will be detected. In spite of that, the fallback approach specified above assumes that the host on the remote side of the failed frontdoor nl-router take action after not receiving return packets.

An alternative approach would be to insure that hosts behind nl-routers transmit packets when necessary. This would work roughly as follows. The IPNL layer would trigger a timeout event whenever the application did not transmit a packet after a certain time (again, 10 to 20 seconds). The IPNL layer would then query its first-hop nl-router to determine if the middle-realm default route changed. If it did not, then the host would do nothing (and timeout again later). If it did, then the host would transmit an echo packet to the remote host in order to convey the change to the remote host.

The upside of this approach is that it would reduce the number of middle realm DNS lookups. The downside is that it would cause hosts to bang on their first-hop nl-routers more than necessary. Both approaches are probably reasonable (scale well, are simple), but which is better is a good issue for further consideration.

6.6 Mobility

Mobility in IPNL is defined as occuring when a host 1) obtains a new IP address, or 2) changes realms. The latter may occur without the former.

When a host changes realms, it is treated as any expat host. It first contacts an nl-server on its adopted realm. (It is able to learn about one or more realm-attached nl-routers through DNS, as described in Section 6.3.) It both registers its presence and learns the realm name of the adopted realm. After this, it contacts an nl-server on its home realm and announces its new realm name. Finally, it should contact an nl-server on its former realm (if not the home realm) and deregister itself.

The protocol for updating the nl-routers is envisioned as being similar to the protocol used to update the home and foriegn agents in mobile IP [RFC2002]. The nl-routers contacted by the expat must also inform all other nl-routers on their realm of the expat's change in status.

For the purpose of expository clarity, the following refers to the two hosts as stationary and mobile. It is understood that both hosts could perfectly well be mobile.

There are two scenarios we must consider. The first is where the mobile host is the first to transmit a packet after moving, and the second is where the stationary host is the first to transmit. In either case, the behaviour of the mobile host is the same, so we consider it first.

When a mobile host either changes IP address or changes realm, it must, for each destination, attach the FQDN header to all packets sent until it has received a return packet with the new IP address or realm number.

If the host did not change realms, then the only change in the transmitted packets themselves is the different Source EHIP. This is all that is needed, from the hosts' perspectives, to maintain communications.

If the host did change realms, then the packets transmitted with FQDNs are transmitted with the Source MRIP+RN fields as unset. These are learned afresh, by both ends, as though it were their initial communications. In addition, the adopted realm name is included in the Source Expat FQDN field of the FQDN Options header.

When the stationary first host transmits a packet to the old address of a mobile host that has moved, it will either get an IPNL-ICMP destination unreachable or no reply after some timeout. This can happen even when the host has not deregistered, because in that case an ICMP unreachable may be sent by an IP router (or by a new host with that address) to the nl-router, which will in turn generate an IPNL-ICMP destination unreachable.

The stationary host then transmits a packet with the IPNL destination address fields "unset" and with the FQDN header, just as it would if it were sending its first packet. The nl-router at the home realm responds with an IPNL-ICMP redirect message giving the realm name of the mobile host's adopted realm. The stationary host then transmits another packet, also with the IPNL destination address fields "unset" and the FQDN header, but this time with the adopted realm name in the Dest Expat FQDN field. This in turn gets routed to the mobile host just as the initial packet in an exchange does.

7 Overview of Anycast Operation

Anycast in IPNL is defined as multiple hosts having the same FQDN. There are two general approaches for delivering a packet to an anycast host. The first is at the level of FQDN, and the other is at the level of address.

At the level of address, anycast within a realm of course can just use IP anycast. Across realms, IP anycast can be used to advantage in the middle realm. In theory, realm numbers could be anycast, but because local routing between realms uses both realm numbers and FQDN, and because realm numbers can change during the course of a connection, there appears to be no particular advantage to doing anycast by realm number versus doing anycast by FQDN.

The use of address-based anycast as described above (EHIP or MRIP level) may be a valid part of an overall anycast strategy, and must obviously be coordinated with FQDN assignment. Never the less, such usage is largely invisible to IPNL and so is not discussed further in this document.

Anycast FQDNs are syntactically identical to regular unicast FQDNs. There are two types of anycast in IPNL, implicit and explicit. In what follows, for descriptive clarity we refer the two hosts communicating as the anycast host and the unicast host. It should be understood that both hosts can be anycast hosts (of either type).

In implicit anycast, the unicast host does not know that anycast is being used. (The anycast host may not either, but it is what the unicast host knows that is of interest here.) If, during the course of communications, the unicast host receives a packet from a different host, then the received IPNL address will be different. The unicast host cannot tell, however, if the packet is from a different host or if the host simply moved.

In explicit anycast, the unicast host does know that anycast is being used, and it knows the identity of of the source host of each received packet. For explicit anycast to work, the anycast host must have a globally unique FQDN in addition to its shared (anycast) FQDN. It must also know which of its FQDNs are unique and which are shared.

Assuming that the anycast host is being addressed by its shared FQDN, whenever either host transmits a packet containing the FQDN header, it attaches the Additional FQDN Extension Header to the packet. The FQDN header contains the shared FQDN (either as the source or destination, depending on the direction of the packet), and the Additional FQDN Extension Header contains the unique FQDN.

The anycast host must continue to transmit its unique FQDN to the unicast host until it sees the unique FQDN returned by the unicast host. The unicast host may stop using the anycast FQDN and switch to the unique FQDN at any time. This prevents further packet from being transmitted to other hosts in the anycast groups.

The transport pseudo header must contain both the unique and shared FQDNs.

There are two general FQDN-based mechanisms for delivering packets to anycast hosts. One is to propogate FQDN destinations in the routing protocol. This would be primarily the routing protocol for local addresses, but one can imagine a protocol for the purpose of advertising anycast FQDNs between frontdoor nl-routers. The other general mechanism is the expat host mechanism.

7.0.1 Expat Host Mechanism

The expat host mechanism works exactly as it does with single expat hosts (section 6.6), except that there are multiple of them. In other words, the nl-routers at the adopted realm need to know of the expat hosts in their realms, and the nl-routers at the home realm need to have the complete list of expat from their realm. When a unicast host wishes to contact an anycast host, its initial packet goes to the home realm. The home realm nl-router replies with an IPNL-ICMP redirect message giving a list of adopted realms. The stationary host picks one and transmits its initial packet as described in section 6.6.

The list returned by the home realm nl-router may be a complete list, or may be a partial list of prefered anycast hosts, i.e. those least loaded and closest to the source. How prefered hosts are determined is outside the scope of this document. We would note, however, that widespread adoption of IPNL should lead to much better address agregation in the middle realm, with the effect that simple comparison of IP addresses should become an effective means of determining rough proximity between two hosts.

7.0.2 Routing Protocol Mechanism

This mechanism is similar to how IP anycast routing is done today.

At the local level, nl-routers at each realm with an anycast host advertise the host's FQDN into local routing. There are more options for how to do this routing in IPNL local routing than in IP routing. In IP anycast, each router views each anycast address as a single destination, though it may maintain more than one path to that destination. IPNL local routing may maintain information about each realm that contains a given FQDN. For instance, each realm listed in a routing advertisement could include its anycast FQDNs as well as its zones. Alternatively the anycast FQDN itself could be a separate advertisement, and nl-routers maintain an entry for only the closest one.

Either way, when a packet is transmitted by the unicast host, nl-routers will forward it to a local realm containing one of the named anycast hosts. The unicast host does not get a redirect and list of realm names as with the Expat host mechanism. Except for the presence of the Additional FQDN extension header if it is explicit anycast, packet exchange proceeds just as it would for local unicast.

The above description applies to local addressing only. It is possible to get the same effect for global addressing by populating frontdoor nl-routers with information about the locations of anycast FQDNs. This could be done through static configuration, caching of IPNL-ICMP redirects (and subsequent cache flushing triggered by IPNL-ICMP destination unreachables), and even some kind of distributed protocol. These techniques are issues for further study.

8 Overview of Multicast Operation

This description assumes hosts natively running IPNL. When a host is not running IPNL, a proxy operates IPNL on its behalf. Proxy operation and other transition issues are described in section /refsecMultTrans. In particular, a form of proxy that extends IP multicast across multiple realms without requiring changes to IP multicast hosts is presented.

8.1 IPNL Names and Addresses

IPNL multicast uses both FQDNs and addresses. The multicast FQDN (M-FQDN) is syntactically identical to the unicast FQDN. It is recognized as an M-FQDN only because it is carried in a group join request rather than in data packets themselves as is the case with unicast.

The M-FQDN serves two roles:

- It is the global identifier for the multicast group.
- It is the address of the realm whose nl-routers are able to assign IPNL multicast addresses to the multicast group.

In the latter role, the M-FQDN is identical to a unicast FQDN (U-FQDN, or just FQDN where the context is clear). In other words, it is a fully routable address. The U-FQDN, however, must not carry anycast semantics (must not be assigned to more than one realm). Other than this, it has the same properties and restrictions as a U-FQDN.

It its former role, the M-FQDN globally identifies one and only one multicast group. It says nothing about where the group members may be. In other words, there need not be any group members at the M-FQDN's home realm.

The multicast address is considerably simpler than its unicast counterpart. The primary simplification is that, from the point of view of hosts, the multicast group may not have multiple addresses. Multicast addresses do however have the global and local forms, where the local form does not require the global header.

The destination address of a multicast packet is the group multicast address. The source address is the address of the source host. The destination address is known by the host (after it does a join), and is set by the host. The source address is dynamically composed as the packet works its way out of the source host's realm and realm group.

The local multicast address consists of a realm number and an EHIP. The local multicast address consists of a realm number, an EHIP, and an MRIP. In both cases, the EHIP is an IP multicast (class D) address. The realm number of the local address is a local representation of the multicast home realm, and is rewritten when the packet passes between realm groups, just as its unicast counterpart does. The multicast MRIP+RN of the global address is globally unique and is not modified end to end.

As with unicast, the transport pseudoheader contains FQDNs only (the M-FQDN and the FQDN of the source).

8.1.1 Notes on Other Design Choices

For multicast, the global IPNL header can be shrunk by 8 bytes if we get rid of the unused fields. Whether this is worth doing is an open issue. By using the same format as unicast, much of the unicast machinery (such as that of writing the source address fields) can be reused.

8.2 Maintained Databases

There are not necessarily any constantly maintained databases required for IPNL multicast per se. Nl-routers may obviously have some kind of configured information, such as the block of IP multicast addresses from which they can make assignments, or some policy information about who can and cannot request assignments, and so on. But in terms of some particular routing or addressing information, nothing beyond what is needed for unicast is required.

All routing information is established dynamically on an as-needed basis.

8.3 Setting Address Fields

When a host wishes to join a multicast group, it starts with the M-FQDN of the multicast group. (Note that since this is a complete departure from how multicast is used today, the transition section (10.4) discusses how IP multicast can be used across multiple realms.)

The host transmits a join request containing the M-FQDN to any nl-router attached to its realm. From here two things happen. First, the message is transmitted to the home realm of the multicast group so that a multicast IPNL address may be learned and if necessary assigned. Second, the state needed in nl-routers to forward packets containing the multicast address is setup.

Exactly how this is done depends on the multicast routing protocols used which in turn depends on the type of multicast (shared tree, single source, etc.). These protocols are outside the scope of this specification per se. In any event, the join message can be transmitted unicast to the multicast group's home realm via normal unicast mechanisms. This ability could be used in multiple ways—to establish the actual delivery tree (which might be rooted at the home realm), or to contact a "rendesvous" box that is located at the home realm and is able to provide information about, for instance, the location of other members.

At some point in time, the necessary routing information for the group will be established. The following information is transmitted to the joining host as part of the join acknowledgement:

- The multicast IPNL address for the group.
- The IP address that may be used to transmit to the group.

The IP address to use may be either unicast or multicast. If unicast, then the destination IP addresses that may be used to transmit to the group are:

- the IP address of the nl-router that answered the join request
- IP addresses from the list in the join request, and
- IP addresses received in the source IP address field of packets received from the group. The latest received IP address is prefered.

Note too that the IP address could be an anycast IP address for all nl-routers, as long as all nl-routers are able to properly forward the packet.

If multicast, then the host must separately join the IP multicast group before it can send and receive packets. (As an enhancement, one might want to consider allowing hosts to start with IP unicast and later switch to IP multicast, for instance if a sufficient number of hosts in the same realm joined the group.)

If the group is locally addressed, the host transmits packets with the multicast address in the Dest EHIP and Local Dest Realm fields. The Local Dest Realm field is the realm number locally used to identify the home realm. The Source EHIP field contains the host's EHIP address. The Local/Used Source Realm is transmitted as "unset". The first nl-router to forward the packet writes the source realm number into the Local/Used Source Realm field.

If the packet crosses realm group boundaries, then both the Local Dest Realm and Local/Used Source Realm fields are rewritten to the local numbers used to identify the respective realms. Within a realm group the realm numbers remain unchanged once set.

If the group is globally addressed, the host transmits packets with the multicast address in the Dest EHIP, Global Dest Realm, and Dest MRIP fields. These fields are transmitted intact end to end.

The Source EHIP field contains the host's EHIP address. All other source fields are transmitted as unset. The Global Source Realm and Source MRIP fields remain unset end to end. The Local/Used Source Realm and Used Source MRIP fields are set exactly as with unicast packets.

8.4 Dealing with Topology Dynamics

Except for changing the destination unicast IP address used to transmit packets to the group, hosts do not see the effect of topology dynamics. The multicast routing algorithms must be able to respond to nl-routers going up and down, or other factors that may change the distribution tree.

9 Router-Generated ICMP Error Messages

IPNL has its own version of ICMP, called IPNL-ICMP. IPNL-ICMP replicates most of ICMPs messages, plus has some additional messages of its own. ICMP has several messages that are generated by routers in response to an error condition, namely the the Destination Unreachable, Time Exceeded, and Parameter Problem messages. When operating under IPNL, these messages are transmitted back to the last nl-router to handle the offending packet. For the first two messages, the nl-router must in turn generate the corresponding IPNL-ICMP message and transmit this back to the originating host.

This in turn requires that the address of the originating host be determined. This can be done for all ICMP messages derived from locally addressed packets, and for all ICMP message received from the middle realm. This is because all ICMP messages must contain at least 8 bytes of upper layer data, which is the first 8 bytes of the IPNL header. These 8 bytes contain both the Local/Used Source Realm and Source EHIP fields.

For locally addressed packets, these fields provide the source host address. For ICMP packets received from the middle realm, the destination IP address of the ICMP packet identifies the Used Source MRIP. This in combination with the Local/Used Source Realm and Source EHIP fields provide the source host address.

It is not possible to determine the source host address from the first 8 bytes of an IPNL header for globally addressed packets that have already passed through the middle realm. For these cases, each nl-router must know the PMTU and minimum expected TTL from itself to its neighbor nl-router, and also whether the nl-router is reachable or not. They must determine in advance if a packet to be transmitted to a given neighbor nl-router will fail, and if so transmit the appropriate IPNL-ICMP error message back to the source. Since local neighbor nl-routers maintain routing protocol state about each other anyway, maintaining this extra information is not excessive.

Some ICMP error messages (or equivalent determination that a given packet's TTL would expire before reaching the next nl-router) trigger IPNL-ICMP messages to be sent back to the original host while others don't. Of those that go back to the host, some result in resetting or modifying the IPNL address while others don't. This is summarized in the following table:

ICMP Error Message	Triggered IPNL-ICMP Message	Address Change
Time Exceeded	Time Exceeded	No
Fragmentation	Fragmentation	No
Unreachable from Middle Realm	Dest Unreachable	Yes
Other Unreachable	Dest Unreachable if no alt route, otherwise none.	No

10 Transition

Because IPNL was designed bottom-up to work with the existing IP infrastructure, the only real issue regarding transition is how to deal with IP-only hosts. The short answer is for nl-routers to be proxies for IP-only hosts. That is to say, an nl-router talks IPNL on one side and IP on the other, and is able to make it appear that the IP host it proxies is actually speaking IPNL. To do this, the proxy needs to know the FQDN and IP address of the IP host it is proxying, and the FQDN of the remote host. Once it knows these things, it can generate IPNL packets on behalf of the host.

There are two basic proxy scenarios:

1. The proxy knows about the IP host in advance of any packet exchange between them. This can always be arranged within private realms, and can sometimes be arranged across the middle realm.
2. The proxy only learns about the IP host when the IP host tries to contact a host "behind" the proxy. This is the case where the IP host is an external host somewhere "on the internet", the proxy is a border nl-router, and the host being reached is in the proxy's realm group.

In both cases, it is necessary for the proxy and the host to "coordinate", so that the proxy knows the FQDN of the remote host that the IP host wishes to communicate with. Because of the difference in the scenarios, different techniques are used to execute this coordination.

10.1 Advance Coordination

Where it is possible to arrange advance coordination, DNS is used as the coordination mechanism. For this to work, the following must be set up:

1. The proxy must have the complete DNS A Records for the hosts in the realm (or, at least, the hosts that it will proxy for). Note that this is something the proxy should have anyway in its usual role as an nl-router. This can be arranged by making the nl-router a secondary DNS server for zones in the private realm, thus causing it to receive zone transfers.
2. DNS requests from the IP-only host must be forwarded to the proxy. This can be arranged by making the nl-router a forwarder.
3. The proxy has a pool of private realm addresses that it owns and that it can dynamically map to hosts outside of the realm. The pool can be large, so this assignment can be relatively long-lived.
4. IP packets with destination addresses from this pool get routed to the proxy that owns the pool. This can be done with routing protocol configuration, or possibly by having the nl-router run the routing protocol.

Now consider the case where the IP host behind the proxy (call it the inside host) initiates communications with a host outside the private realm (call it the outside host). When the inside host makes a DNS query for the outside host, the DNS query arrives at the proxy by virtue of its being the forwarder. The DNS query of course contains the FQDN of the outside host. The proxy assigns a private-realm IP address from the pool to this FQDN. It answers the query with this assigned IP address as the A Record address for the searched FQDN.

When the inside host receives this DNS answer, it transmits a packet to the assigned IP address, which is routed to the proxy. The proxy can deduce the destination FQDN from the destination IP address from the mapping it assigned. It can also derive the source FQDN from the source IP address because of the zone transfer(s) it received for hosts in the private realm.

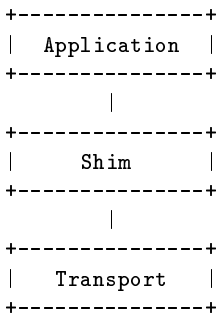
Once the proxy has the destination FQDN, it can use DNS to learn the IP address used to reach the outside host. This IP address may be that of the outside host, or may be that of a proxy for the outside host—it doesn't matter. If the former and the outside host is IP-only, then the proxy communicates with it using IP directly.

Note that the proxy may directly bridge two private realms. In this case it may act as a proxy for IP-only hosts in both realms.

10.2 On-demand Coordination

When an outside IP-only host for which advance coordination is not possible initiates communications with an inside host, there must be some explicit coordination between the outside host and the proxy at the time the outside host wishes to communicate with an inside host. This explicit coordination in turn requires some new protocol.

The need for a new protocol would seem to imply that we are not actually talking about IP-only hosts here. In fact, we are talking about hosts that are IP-only from the transport layer down, but that have a "shim layer" inserted between transport and the layer above it (lets call it the application layer). DNS resolvers in these hosts also go unmodified.



The idea is that in many cases it is easier to introduce a shim layer above transport than it is to change protocol implementations below transport. The shim layer can be introduced as part of new applications or modifying existing applications, or a library that is called by applications.

A good example of such a new application is WAP. Here there are a large number of wireless devices either behind an IPNL proxy or using IPNL natively. These wireless devices wish to access WAP-enabled web servers, and wish to access and be accessed by PCs. These new applications can specify the shim layer as a standard part of their stack, enabling connectivity between large numbers of new wireless and legacy wired boxes. New PC applications could then be written without assuming the existence of IPNL support in either the PC OS or the PC's access network.

The shim protocol works as follows. Assume an outside IP-only host with a shim initiating communications with an inside host. It doesn't matter if the outside host has a global IP address or gets its connectivity via a NAT box. It also doesn't matter if the inside host is IP-only or not—this is hidden from the outside host by the proxy.

The outside host does its usual DNS query and obtains an IP address. It sends a shim protocol packet to a well-known UDP port (not yet assigned) at the IP address containing:

1. the FQDN of the destination host,
2. the IP address it used as the source IP address in the packet it sent,
3. the transport protocol it will be using (TCP or UDP), and
4. the source and destination port numbers it will be using.

The first item is used of course to identify the inside host. The second is used to determine if the outside host is going through a NAT box (by checking whether the source address of the packet matches the address given in the shim protocol). If the outside host is not going through a NAT box, then the remaining information is sufficient to identify the inside host from packets received from the middle realm. That is, packets from the middle realm with the supplied source IP address, transport protocol, and source and destination ports must be for the inside host identified by the FQDN. (We are making the implicit assumption here that if the outside host's IP address is not being translated, then its port numbers are not either.)

If the box at the IP address is not a proxy, it will either reply with an IPNL shim protocol message indicating that it is an endhost and so the shim protocol is unnecessary, or with an ICMP destination port unreachable message. Either way, the host knows that the destination host is on the middle realm and proceeds with regular TCP or UDP (no shim protocol).

Assuming that the remote box is a proxy, it first determines if the packet was received through a NAT box or not, by matching the IP address in the shim protocol header with the actual received IP address. If they match, the proxy tells the outside host that no NAT box is being used, and the outside host proceeds with regular TCP or UDP.

If the outside host is determined to be going through a NAT box, then it gets a bit ugly (as is typically the case with NAT). The proxy assigns a 128-bit identifier to the inside FQDN, stores this, and returns the identifier to the outside host.

The outside host then starts sending data packets to the inside host. If the packets are UDP, then there is a shim header between UDP and the layer above it. The shim header contains the 128-bit identifier, so the proxy can determine both the source and destination of the packet. The first time one of these packets is received, the proxy also stores the IP address and port numbers of the incoming packet for use in return packets. The shim header is stripped, the relevent UDP fields are modified (Length and Checksum fields), and the packet forwarded to the inside host. The shim header is likewise inserted into packets from the inside host to the outside host by the proxy.

If the protocol is TCP, things are uglier still because the application layer does not know how the TCP stream will be packetized and cannot therefore insert the identifier into individual packets. The beginning of the TCP stream must contain the identifier. The proxy must setup its end of the connection itself, and wait until the beginning of the stream is received. The proxy now recognizes that the TCP stream is for the inside host from this identifier alone.

It then sets up the connection with the inside host as though it were the outside host. Once this is done, it strips the identifier from the stream and sends the stream on to the inside host. Thereafter, individual packets are forwarded between the two hosts—the proxy does not intercept the stream. The proxy must, however, modify the TCP seq and ack fields to account for the fact that it modified the size of the stream (just as NAT boxes do today).

10.3 Choosing Between IP, IPNL, and the Application Shim

When a host first starts running IPNL, either natively or with an application shim, it must determine if it is on the middle realm or the private realm. It starts by assuming that it is on the middle realm. While it believes that it is on the middle realm, it initiates every packet exchange with a standard DNS lookup. For each DNS answer, it must determine if the remote box with the returned IP address is IP-capable or not.

If the initiating host has native IPNL, it determines this by sending an IPNL-ICMP echo query (ping). If the initiating host is IP-only but using the application shim, it determines this by transmitting the shim protocol packet as described in the previous section.

If the remote box is not IPNL capable, then either an ICMP destination unreachable (protocol or port, depending) is returned, or nothing is returned. In the latter case, the initiating host can send a few IPNL and ICMP

pings. If only the ICMP pings are answered, it can assume that the remote box is IP-only.

If on the other hand the remote box is IPNL-capable, then it will answer either message properly. If furthermore the remote box is an nl-router, and the initiating host is on a private realm, then the answer indicates as much. In this way and in only this way does the initiating host determine that it is in a private realm.

Assume for now that the initiating host did not determine that it is in a private realm. If the remote box box was found to be IP-only, then it is assumed to be the destination host (not a proxy), and communications proceeds with straight IP. If the remote box was found to be IP-only with an application shim, then the application shim protocol tells it whether or not the remote box is the destination host or not. Either way, communications proceeds as described in the previous section.

If the remote box is found to have native IPNL, then IPNL is used. The remote box may either be an nl-router or the destination host, but either way IPNL is used.

This is summarized in the following table:

	Protocol Selection if Middle Realm		
Initiating Host Protocol	Remote Box Protocol	Remote Box Type	Protocol Used
Native IPNL	Native IPNL	nl-router or destination host	IPNL
	IP-only	destination host	IP-only
App Shim	App Shim	proxy	IP-only or App Shim
		destination host	IP-only
	IP-only	destination host	IP-only

Assume now that the initiating host just discovered that it was on a private realm. The first thing it does is transmit an IPNL-ICMP Realm Information query to the nl-router it just discovered. The answer contains a list of the nl-routers for the realm, as well as other information such as the realm name. Thereafter the host periodically transmits another Realm Information query to an nl-router to keep its list of nl-routers up to date.

If the host is IP-only with an app shim, then thereafter it uses straight IP. This is because the nl-routers are able to proxy for the host using advance coordination.

If the host has native IPNL capability, or is upgraded from app shim to IPNL capability, then the host then announces itself to all of the nl-routers. Each nl-router tags the host as being native IPNL-capable. This way, the nl-router knows to use IPNL when forwarding packets received from outside and destined for the host.

It does not periodically announce itself to all nl-routers. Rather, nl-routers continue to assume that the host is IPNL-capable. If the host later becomes IP-only, the nl-routers will discover this either by later receiving an ICMP destination protocol unreachable, or by not receiving any replies after transmitting a number of IPNL packets (but receiving ICMP ping replies).

Once an IPNL host determines that it is on a private realm, it stops using DNS, and instead transmits all packets directly to an nl-router. As part of normal IPNL operation, it may sometimes be redirected to another nl-router, or may be redirected to a host within the private realm. In the latter case, the redirect itself indicates whether the host is IP-only or IPNL.

If all nl-routers known to a host in a private realm are found to be either unreachable or no longer private realm nl-routers, the host reverts back to assuming that it is on the middle realm. It may of course still be on the private realm, but by assuming that it is on the middle realm, it uses DNS as usual and can at least

communicate with other hosts on the private realm.

From the above description, it can be seen that nl-routers assume that all hosts on their private realms are IP-only unless explictly told otherwise. Once an nl-router tags a host as IPNL-capable, they continue under this assumption until explicitly told otherwise. In other words, nl-routers are not pro-active in determining the IPNL capability of hosts on their private realms. It is entirely up to the hosts to inform the nl-routers.

Frontdoor nl-routers must, on the other hand, actively determine the capability of boxes they communicate with over the middle realm when hosts behind them are initiating. Typically a frontdoor nl-router is both a traditional NAT box and an nl-router. Because IPNL doesn't consume MRIP addresses or port numbers like NAT does, it is preferable that the frontdoor nl-router use IPNL where it can.

A simple way to do this is to send IPNL and ICMP pings to determine the capability of the box. A better way to do this is to create a new DNS record that indicates simply whether the box with the IP address is IPNL capable or not. This saves the cost and delay of sending IPNL and/or ICMP pings. In the absense of this, however, IPNL and ICMP pings can suffice.

10.4 Multicast Transition

To be completed.

11 API

As suggested by the above descriptions, the API for IPNL typically hides addresses and deals in FQDNs only. Rather than have separate calls for mapping DNS names to addresses, and then binding the addresses to sockets, the IPNL API would directly bind FQDNs to sockets. The application never has to see the addresses at all.

Having said that, the IPNL API should allow for the application to specify the addresses directly. This is especially but not only useful within realm groups and realms, where addresses may be stable for long periods of time.

11.1 Transition

When an old application (one that uses the traditional IP API) runs over IPNL, it will naturally use the old calls. If the old application uses an IP address directly (makes no gethostbyname() or equivalent call to map an FQDN to an address), then IPNL has no choice but to assume that the destination is in the same realm, and transmit the packet over straight IP.

If the old application does start with an FQDN, then IPNL would have to return some IP address to the application for the later socket bind. Ideally this IP address is that of the actual destination, but alternatively it could be a locally manufactured number unique to that socket.

To obtain the actual IP address in this case, IPNL could transmit an IPNL ping packet to the destination, thus learning the IP address of the destination. This would be returned to the application. The danger here is that the same IP address could apply to different destinations. If this were the case, IPNL may still have to give the application a locally manufactured but unique number.

11.2 Details

To be completed later.

12 Host Algorithm

To be completed based on above descriptions.

13 Nl-router Algorithm

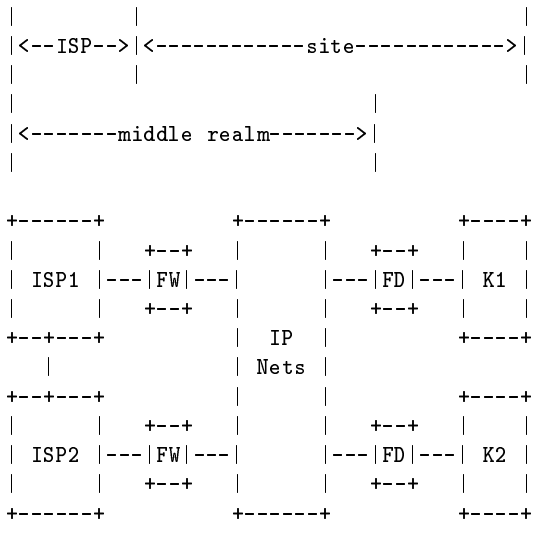
To be completed based on above descriptions.

14 Relationship Between Topological and Administrative Entities

The topological entities of IPNL are realms (private and middle) and realm groups. Roughly speaking, the administrative entities are ISPs and private networks, here called sites. Except for the fact that a realm group would typically fall under a single administrative entity, topological boundaries do not necessarily coincide with administrative boundaries. This section outlines some of the various possible configurations, and their impact on realm numbering.

In the following series of figures, a topology is shown. In each case, the administrative boundary between site and ISP is given, with the ISPs on the left and the site(s) on the right. In addition, the middle realm boundary is shown. The boundaries are shown even though both of them can be inferred. The ISP/site boundary can be inferred by the location of the firewall (FW). The middle realm boundary can be inferred by the location of the frontdoor nl-routers (FD).

In topology 1 below, a substantial amount of a large site's topology is part of the middle realm. This part is represented by the box labeled "IP Nets". This part of the site is separated from its ISPs by firewalls. The site has a couple of realm groups (K1 and K2) separated from the middle realm part of the site by frontdoor nl-routers (FD). Even though this site is multi-homed, whether or not it is effective multihoming (i.e., both ISPs can be used by hosts in the site) depends on factors outside of IPNL per se. For instance, if the site has one (middle realm) IP prefix that is advertised by both ISPs, then either ISP can be used.

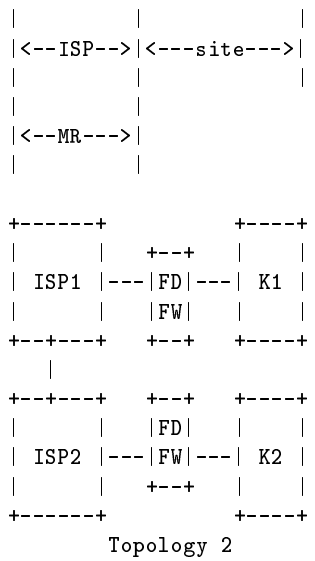


Topology 1

In topology 2 below, the site and middle realm boundaries just happen to coincide. While this is a "natural"

way to envision IPNL, it is as likely to be the exception as the rule. Indeed, much of the purpose of this section is to dispel the notion that the middle realm boundary has anything to do with site boundaries. Note that the two realm groups shown may be part of the same administration or not. For instance, they might be corporate sites in different countries that communicate between themselves via the public internet infrastructure. They may also have a backdoor nl-router connecting them.

If the realm groups shown are different administrations, then their local realm numbers need be no different from their global realm numbers. Both are administered by the same body and can be set to the same value—translation at the realm group boundary is not necessary. Note that this fact is pointed out not because it is particularly advantageous. Rather, it is pointed out to make the example clear.

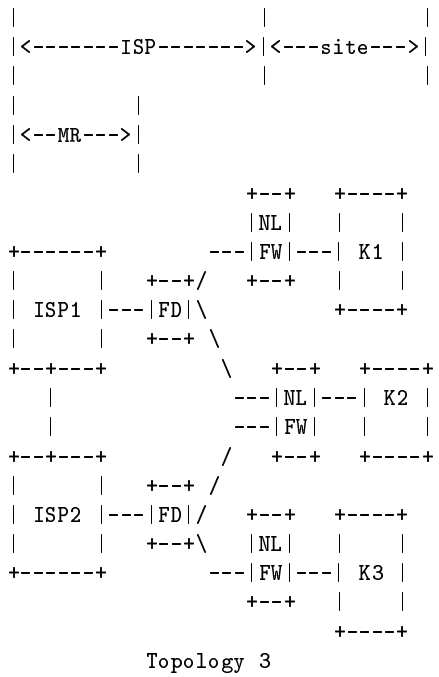


Topology 2

Topology 3 illustrates the case where multiple different sites are reachable through a single FRIP. Here, NL means nl-router. In this topology, the dominion of the ISP is pushed beyond the middle realm. The ISP operates a frontdoor nl-router, and assigns global realm numbers to the realms in each realm group. Within each realm group, a (different) set of realm numbers are assigned for local use. The realm numbers are translated at packets crossing the frontdoor nl-router (FD). Once again, the realm groups shown may belong to the same or different administrations, and may have backdoor nl-routers connecting them.

Topology 4 pushes the dominion of the ISP still further. Here, individual small sites (e.g. SOHO) constitute a single realm or even part of a single realm. This configuration allows an ISP to bring on very large numbers of customers while still using only a small number of middle realm IP addresses. This configuration isn't particularly useful for general purpose IP hosts over the near term, because they still need a unique MRIP/port number combination to talk to pure IP hosts across the middle realm. For new devices that are limited to a certain application domain, such as mobile PDAs limited to WAP, this configuration works well.

Notice that in this configuration, once again the assignment of realm numbers falls under the jurisdiction of a single administrative domain—this time the ISP itself. As such, global and local realm numbers match and there is no need for realm number translation.



Topology 3

15 Comparison with IPv6

When comparing IPNL with IPv6, comparisons must be made both with pure IPv6, and to some mixture of IPv6 and IPv4, whereby the global infrastructure is still mainly IPv4, and sites (enterprises, SOHOs, etc.) are either IPv6, IPv4, or some mix of the two. These two modes of IPv6 are hereafter called IPv6pure and IPv6on4 (understanding that there are a number of variations of IPv6on4, which of course is part of the difficulty).

Below is a summary of the comparison. This is followed by more detailed explanations.

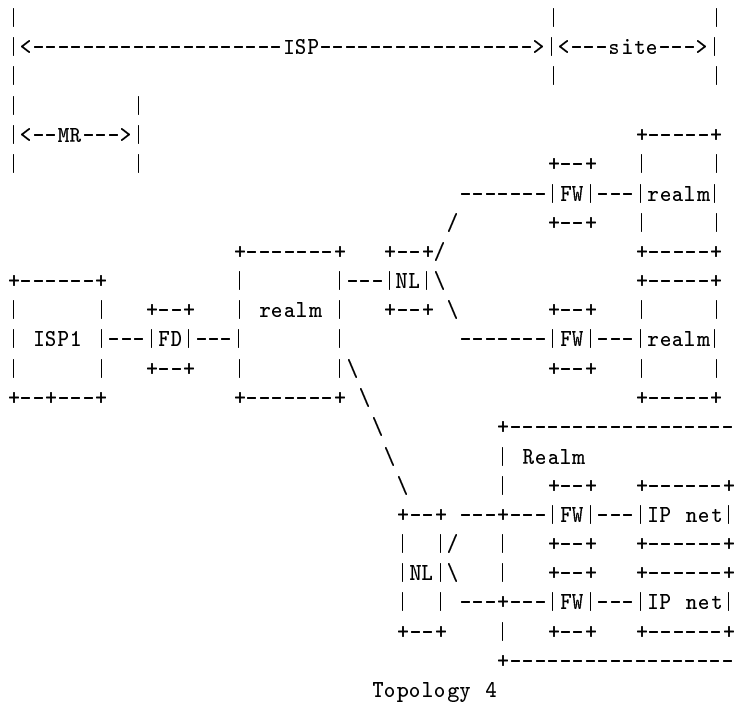
15.1 Comparison Summary

Compared to either IPv6pure or IPv6on4, IPNL has the major advantage that it completely isolates internal site operation from ISP numbering decisions. This results in the following tangible benefits:

- Sites can change provider ISPs easily.
- ISPs can renumber at will, allowing for orders-of-magnitude better aggregation than what is possible today.
- Sites can effectively multihome (use multiple different ISPs) without effecting scalability in the ISP routing infrastructure.

Compared to IPv6pure, IPNL is far less expensive. IPv6pure ultimately requires upgrade of the entire routing infrastructure, both for ISPs and sites. IPNL requires no upgrades to the routing infrastructure, and indeed allows smaller sites (those that can fit comfortably in a single realm) to effectively use IPNL without changing anything except their ISP access router/firewall.

Even compared to IPv6on4, IPNL offers a simpler and almost certainly less expensive transition. For instance, [TransIntro] outlines no less than 5 distinct tunneling mechanisms using 3 distinct address structures, and



Topology 4

6 distinct translation mechanisms. It outlines 6 different case scenarios where the 11 mechanisms are used individually or in various combinations. These mechanisms variously require upgrade of DNS, upgrade of host stacks, and manual configuration of tunnels (both at routers and hosts).

IPNL, on the other hand, has two mechanisms, proxy of IP-only hosts, and proxy of IP-only hosts with an application shim. Neither require changes to DNS, nor do they require new manual configuration at hosts or for crossing the IPv4 global infrastructure. Their usage scenarios are clearly defined and simple.

IPv6pure has the following potential advantages over IPNL. For packets between sites (realm groups in IPNL), IPNL packets undergo more header rewriting than IPv6, and are therefore probably slower to process. On the other hand, this re-writing only occurs at the edges of the network, and at firewall boxes that in any event are doing significant amounts of header work (L4 filtering, for instance), so doesn't seem to be a particurly serious drawback.

IPv6 auto-address configuration is simpler than IPNL, because IPv6 devices can be hard-wired with their own globally unique identifier. All that is required is that the IPv6 address prefix be assigned to the box. No individual address management per se is needed.

We hasten to note, however, that auto-address configuration in IPNL is still relatively straight-forward, primarily because individual realm IP address spaces can be sparsely populated, thus allowing address assignment to come from large pools of addresses. As an aside, an increase in the size of the EHIP fields in IPNL would allow for IPv6-style address auto-configuration. Section 6.1.1 discusses this.

In any event, both of the above advantages of IPv6pure disappear in an IPv6on4 environment.

15.2 Isolation of Sites and ISPs

Isolation of "sites" and "ISPs" is one of the original features of IP (though of course the latter term, at least, was not in use at that time). An IP network site administrator could get a block of IP addresses (class A, B,

or C), and use there without regard to how the site obtained its global internet connectivity.

This original feature was broken when CIDR was adopted to improve routing infrastructure scalability. By blocking addresses above the "site" level, CIDR created dependencies between address prefix and global internet access point. CIDR has been partially effective. Routing tables are huge and growing, but not nearly as big as they otherwise would be. By and large sites have avoided the dependencies created by CIDR through the widespread use of NAT. Without NAT, sites must either convince their ISP to advertise their individual site prefix, or renumber when they change ISPs. Without NAT, they cannot effectively multi-home (use multiple simultaneous ISPs) unless their ISPs advertise their site prefix.

The fact that IPv6 does not address this problem at an architectural level is a major failing. Instead, IPv6 proposes mechanisms for making it easier to live with the dependencies, such as site-wide router renumbering [renum] and DNS extensions [dnsExtend].

IPv6 also proposes the use of site-local prefixes so that internal communications may continue in the face of renumbering. This is a substantial improvement over the alternative, but it places a severe constraint on certain topology reconfigurations. For instance, when two sites merge, one or the other must renumber its subnet IDs. This procedure is not subject to the automatic site-wide prefix renumbering mechanisms.

Never-the-less, for reasons described in the following, a likely outcome of this is that IP administrators will force site/ISP isolation at an architectural level by using IPv6-IPv6 NAT (here-after called NATv6) at site boundaries rather than renumber internally. While this would be a huge improvement over IPv4-IPv4 NAT, it it nevertheless not a desirable outcome.

Both IPv6pure and IPv6on4 have the characteristic that one of the following three must be true:

1. Sites must completely renumber from time to time, or
2. Sites must use NAT between themselves and the global infrastructure, or
3. The global routing infrastructure will not scale (to the number of sites).

The reason is that, in order to scale reasonably in the global routing infrastructure, multiple administratively distinct sites must be aggregated under a single address prefix. Because of changes in topology (sites changing providers, or ISPs modifying their aggregations in order to keep up with growth), sites must from time-to-time be given new address prefixes by their ISPs.

The third characteristic is not an option. The global routing infrastructure must scale or the internet becomes unusable. This means that either sites must renumber occasionally (which if not necessary in IPNL), or they must use NATv6.

While renumbering of a large site appears technically feasible, it is in any event risky and will almost certainly require significant manpower to insure that the process goes smoothly (even if the actual mechanisms are automatic). Given this, it seems highly likely that a site administrator would prefer to simply install a NATv6 box at the site boundary than undergo the risk and cost of renumbering.

After all, most site administrators are already comfortable with NAT, and some even like it. Furthermore, NATv6 is far easier to use and safer than IPv4 NAT, because all NATv6 address assignments are static one-to-one assignments between pairs of globally unique addresses that typically involve only rewriting the address prefix. Add to this the fact that NATv6 gives the site some level of simple multihoming, and the fact that NAT will almost certainly already be in use for IPv6 transition, and the use of NATv6 over site-wide renumbering becomes an easy decision to make.

Just in case the reader is thinking that this doesn't sound too bad, consider the implications. For single-homed or multi-homed sites, it breaks IPsec. For multi-homed sites, additional header hacking is required. This is because packets exiting the site at different ISPs will be given different prefixes, causing the remote host to think they came from a different source. This could perhaps be fixed by having the border routers attach some option used by mobile IP to make it look like the host simply moved.

Whether or not all of this would work seems besides the point. It is ugly, counter to the architecture of IPv6, and anyway IPNL solves the multihoming problem simply and cleanly.

As a final note, it should be pointed out that previous proposals have been made on how to get multi-homing and scalability at the same time (not necessarily just for IPv6). None of these approaches eliminates renumbering. RFC2260 suggests BGP mechanisms and tunneling to minimize or eliminate the scaling overhead at the cost of some protocol complexity in ISP and site BGP. [Yu] proposes limiting the scope of advertisements to improve scalability. [dupont] proposes the use of IPv6 mobility mechanisms at the host to reroute packets around failed ISPs. This latter work comes closest in spirit to IPNL's approach.

15.3 Cost of Transition

It should be self-evident that transition to IPNL is less costly than transition to IPv6pure. The latter requires changes to all routers and all of the systems that support routers (management, address assignment, etc.), whereas the former doesn't.

It is almost certainly also the case that transition to and operation of IPNL is less costly than transition to and operation of IPv6on4. IPv6on4, at least as it stands now, offers a bewildering array of different mechanisms for use in different environments with no clear architectural principles guiding the whole thing. It pushes credulity to think that widespread deployment of these ad hoc mechnisms can be anything but difficult to manage and maintain.

15.4 Load on DNS

This is not a comparison with IPv6 per se, but a commentary on the effect of IPNL on DNS usage, particularly the global (middle realm) DNS infrastructure. DNS usage over the middle realm is virtually unchanged from how it operates today. That is, DNS lookups are made on a per-host-pair-communications basis. Applications that are inclined to make reverse DNS lookups today are equally inclined to do DNS lookups on the reverse path in IPNL.

Having said that, IPNL creates an opportunity to really reduce the load on DNS.

This is because DNS as used today returns A records for individual hosts. In IPNL, DNS returns A records that apply to entire zones or collections of zones. It should be possible to make this fact explicit so that nl-routers can reuse a given DNS answer for many hosts in the same zones, thus reducing the number of DNS queries. One way of doing this is to attach extra records to a DNS answer that indicate which zones are reachable behind a given set of IP addresses.

Nl-routers could cache these answers for long periods of time, because the answers can be explicitly flushed by remote nl-routers. When a remote nl-router gets a packet for an FQDN that does not exist in any zones behind it, it returns a destination unreachable that says as much, and the originating nl-router can flush its entry and do a new DNS query.

The downside of this approach is that an incorrect unreachable would have a greater impact, so there is a potential denial-of-service attack that would have to be countered. These are issues that need further study. Since DNS usage with IPNL is in any event no worse than today, however, unsuccesful resolution of these issues wouldn't be a factor in rejecting IPNL as compared to IPv6.

16 Acknowledgements

To be written.

References

dnsExtend Crawford, M., Huitema, C., Thomson, S., "DNS Extensions to Support IPv6 Address Aggregation and Renumbering," draft-ietf-ipngwg-dns-lookups-07.txt, Mar. 2000.

dupont Dupont, F., "Multihomed routing domain issues for IPv6 aggregatable scheme," draft-ietf-ipngwg-multi-isp-00.txt, Sep. 1999.

GSE Crawford, M., Mankin, A., Narten, T., Stewart, J., Zhang, L., "Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6," draft-ietf-ipngwg-esd-analysis-05.txt, Oct. 1999.

IPv6 Deering, S., and Hinden, B., "Internet Protocol, Version 6 (IPv6)", RFC 2460, December 1998

NoDHCP Troll, R., "Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network," draft-ietf-dhc-ipv4-autoconfig-05.txt.

renum Crawford, M., "Router Renumbering for IPv6," draft-ietf-ipngwg-router-renum-10.txt, Mar. 2000.

RFC1191 Mogul, J., Deering, S., "Path MTU Discovery", RFC 1191, Nov. 1990

RFC1918 Rekhter Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., Lear, E., "Address Allocation for Private Internets", RFC 1918, Feb. 1996.

RFC2002 Perkins, C. (editor), "IP Mobility Support," RFC 2002, Oct. 1996.

RFC2260 Bates, T., Rekhter, Y., "Scalable Support for Multi-homed Multi-provider Connectivity," RFC 2260, Jan., 1998.

TransIntro Biemolt, W., Kaat, M., Larder, T., Steenman, H., van der Pol, R., Tsirtsis, G., Sekiya, Y., Durand, A., "A Guide to the Introduction of IPv6 in the IPv4 World," draft-ietf-ngtrans-introduction-to-ipv6-transition-03.txt, Mar. 2000.

Yu Yu, J., "IPv6 Multihoming with Route Aggregation," draft-ietf-ipngwg-ipv6multihome-with-aggr-00.txt, Nov. 1999.