Network Working Group
Internet-Draft
Intended status: Experimental
Expires: October 13, 2008

E. Lear
Cisco Systems GmbH
April 11, 2008

# NERD: A Not-so-novel EID to RLOC Database
## draft-lear-lisp-nerd-04.txt

Status of this Memo



Abstract

LISP is a protocol to encapsulate IP packets in order to allow end sites to multihome without injecting routes from one end of the Internet to another. This memo specifies a database and a method to transport the mapping of EIDs to RLOCs to routers in a reliable, scalable, and secure manner. Our analysis concludes that transport of of all EID/RLOC mappings scales well to at least 10^8 entries.

Table of Contents

## 1. Introduction

Locator/ID Separation Protocol (LISP) [1] separates an IP address used by a host and local routing system from the locators advertised by BGP participants on the Internet in general, and in the default free zone (DFZ) in particular. It accomplishes this by establishing a mapping between globally unique endpoint identifiers (EIDs) and routing locators (RLOCs). This reduces the amount of state change that occurs on routers within the default-free zone on the Internet, while enabling end sites to be multihomed.

In some mapping distribution approaches to LISP the mapping is learned via data-triggered control messages between ingress tunnel routers (ITRs) and egress tunnel routers (ETRs) through an alternate routing topology [14]. In other approaches of LISP, the mapping from EIDs to RLOCs is instead learned through some other means. This memo addresses different approaches to the problem, and specifies a Not-so-novel EID RLOC Database (NERD) and methods to both receive the database and to receive updates.

NERD is offered primarily as a way to avoid dropping packets, the underlying assumption being that dropping packets is bad for applications and end users. Those who do not agree with this underlying assumption may find that other approaches make more sense.

LISP and NERD are both currently experimental protocols. The NERD database is specified in such a way that the methods used to distribute or retrieve it may vary over time. Multiple databases are supported in order to allow for multiple data sources. An effort has been made to divorce the database from access methods so that both can evolve independently through experimentation and operational validation.

### 1.1. Base Assumptions

In order to specify a mapping it is important to understand how it will be used, and the nature of the data being mapped. In the case of LISP, the following assumptions are pertinent:

- The data contained within the mapping changes only on provisioning or configuration operations, and is not intended to change when a link either fails or is restored. Some other mechanism such as the use of LISP Reachability Bits with mapping replies handles healing operations, particularly when a tail circuit within an service provider's aggregate goes down. NERD can be used as a verification method to ensure that whatever operational mapping changes an ITR receives are authorized.

- While weight and priority are defined, these are not hop-by-hop metrics. Hence the information contained within the mapping does not change based on where one sits within the topology.
- A purpose of LISP being to reduce control plane overhead by reducing "rate X state" complexity, updates to the mapping will be relatively rare.
- Because LISP and NERD are designed to ease interdomain routing, their use is intended within the inter-domain environment. That is, LISP is best implemented at either the customer edge or provider edge, and there will be on the order of as many ITRs and EID Prefixes as there are connections to Internet Service Providers by end customers.
- As such, NERD cannot be the sole means to implement host mobility, although NERD may be in used in conjunction with other mechanisms. For instance, it would be possible for a mobile node to receive a local address that is an EID and pass that to the correspondent node, who could also make use of an EID. As such use of LISP in this case would be transparent, and no mapping entries are changed for mobility.

## 1.2. What is NERD?

NERD is a Not-so-novel EID to RLOC Database. It consists of the following components:

1. a network database format;
2. a change distribution format;
3. a database retrieval/bootstrapping method;
4. a change distribution method.

The network database format is compressible. However, at this time we specify no compression method. NERD will make use of potentially several transport methods, but most notably HTTP [2]. HTTP has restart and compression capabilities. It is also widely deployed.

There exist many methods to show differences between two versions of a database or a file, UNIX's "diff" being the classic example. In this case, because the data is well structured and easily keyed, we can make use of a very simple format for version differences that simply provides a list of EID/RLOC mappings that have changed using the same record format as the database, and a list of EIDs that are to be removed.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3].

## 1.3. Glossary

The reader is once again referred to [1] for a general glossary of terms related to LISP. The following terms are specific to this memo.

Base Distribution URI: An Absolute-URI as defined in Section 4.3 of [6] from which other references are relative. The base distribution URI is used to construct a URI to an EID/RLOC mapping database. If more than one NERD is known then there will be one or more base distribution URIs associated with each (although each such base distribution URI may have the same value).

EID Database Authority: The authority that will sign database files and updates. It is the source of both.

The Authority: Shorthand for the EID Database Authority.

NERD: (N)ot-so-novel (E)ID to (R)LOC (D)atabase.

AFI Address Family Identifier.

Pull Model: An architecture where clients pull only the information they need at any given time, such as when a packet arrives for forwarding.

Push Model: An architecture in which clients receive an entire dataset, containing data they may or may not require, such as mappings for EIDs that no host served is attempting to send to.

Hybrid Model: An architecture in which some information is pushed toward the receiver from a source and some information is pulled by the receiver.

# 2. Theory of Operation

Operational functions are split into two components: database updates and state exchange between ITR and ETR during a communication.

## 2.1. Database Updates

What follows is a summary of how NERDs are generated and updated. Specifics can be found in Section 3. The general way in which NERD works is as follows:

1. A NERD is generated by an authority that allocates provider independent (PI) addresses (e.g., IANA or an RIR) which are used by sites as EIDs. As part of this process the authority generates a digest for the database and signs it with a private key whose public key is part of an X.509 certificate. [10] That signature along with a copy of the authority's public key is included in the NERD.
2. The NERD is distributed to a group of well known servers.
3. ITRs retrieve an initial copy of the NERD via HTTP when they come into service.
4. ITRs are preconfigured with a group of certificates whose private keys are used by database authorities to sign the NERD. This list of certificates should be configurable by administrators.
5. ITRs next verify both the validity of the public key and the signed digest. If either fail validation, the ITR attempts to retrieve the NERD from a different source. The process iterates until either a valid database is found or the list of sources is exhausted.
6. Once a valid NERD is retrieved, the ITR installs it into both non-volatile and local memory.
7. At some point the authority updates the NERD and increments the database version counter. At the same time it generates a list of changes, which it also signs, as it does with the original database.
8. Periodically ITRs will poll from their list of servers to determine if a new version of the database exists. When a new version is found, an ITR will attempt to retrieve a change file, using its list of preconfigured servers.
9. The ITR validates a change file just as it does the original database. Assuming the change file passes validation, the ITR installs new entries, overwrites existing ones, and removes empty entries, based on the content of the change file.

As time goes on it is quite possible that an ITR may probe a list of configured neighbors for a database or change file copy. It is equally possible that neighbors might advertise to each other the version number of their database. Such methods are not explored in depth in this memo, but are mentioned for future consideration.

## 2.2. Communications between ITR and ETR

[1] describes the basic approach to what happens when a packet arrives at an ITR, and what communications between ITR and ETR take place. NERD provides an optimistic approach to establishing communications with an ETR that is responsible for a given EID prefix. State must be kept, however, on an ITR to determine whether that ETR is in fact reachable. It is expected that this is a common requirement across LISP mapping systems, and will be handled in the

core LISP architecture.

## 2.3. Who are database authorities?

This memo does not specify who the database authority is. That is because there are several possible operational models. In each case the number of database authorities is meant to be small so that ITRs need only keep a small list of authorities, similar to the way a name server might cache a list of root servers.

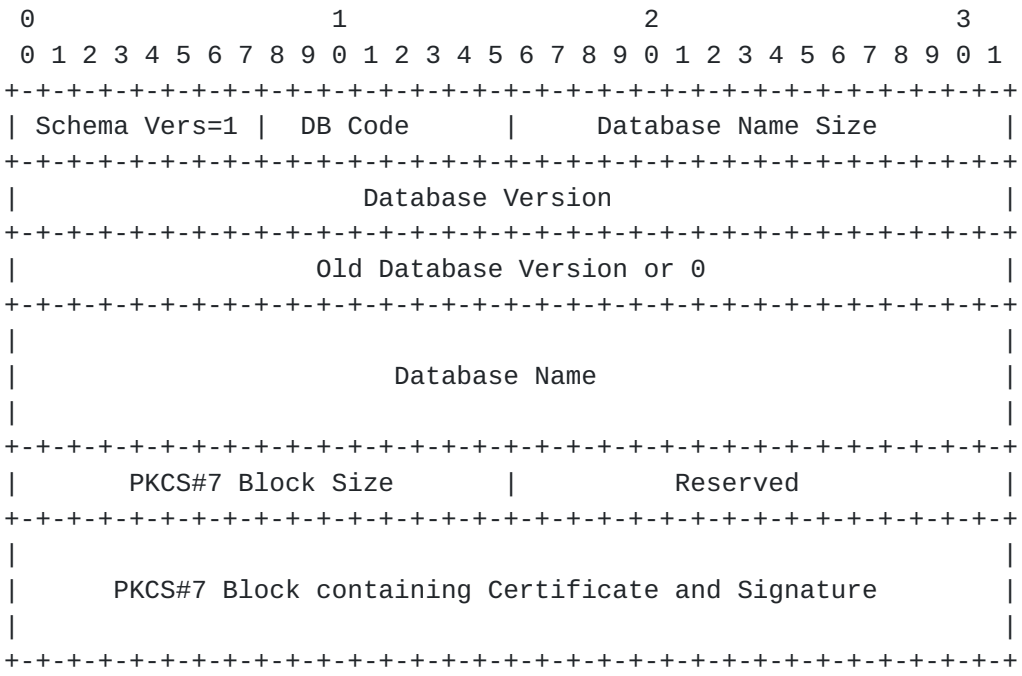
- A single database authority exists. In this case all entries in the database are registered to a single entity, and that entity distributes the database. Because the EID space is provider independent address space, there is no architectural requirement that address space be hierarchically distributed to anyone, as there is with provider-assigned address space. Hence, there is a natural affinity between the IANA function and the database authority function.
- Each region runs a database authority. In this case, provider independent address space is allocated to either Regional Internet Registries (RIRs) or to affiliates of such organizations of network operations guilds (NOGs). The benefit of this approach is that there is no single organization that controls the database. It allows one database authority to backup another. One could envision as many as ten database authorities in this scenario. One drawback to this approach, however, is that any reference to a region imposes a notion of locality, thus potentially diminishing the split between locator and identifier.
- Each country runs a database authority. This could occur should countries decide to regulate this function. While limiting the scope of any single database authority as the previous scenario describes, this approach would introduce some overhead as the list of database authorities would grow to as many as 200, and possibly more if jurisdictions within countries attempted to regulate the function. There are two drawbacks to this approach. First, as distribution of EIDs is driven to more local jurisdictions, an EID prefix is tied even tighter to a location. Second, a large number of database authorities will demand some sort of discovery mechanism.
- Independent operators manage database authorities. This has the appeals of being location independent, and enabling competition for good performance. This method has the drawback of potentially requiring a discovery mechanism.

The latter two approaches are not mutually exclusive. While this specification allows for multiple databases, discovery mechanisms are left as future work.

## 3. NERD Format

The NERD consists of a header that contains a database version and a signature that is generated by ignoring the signature field and setting the authentication block length to 0 (NULL). The authentication block itself consists of a signature and a certificate whose private key counterpart was used to generate the signature.

Records are kept sorted in numeric order with AFI plus EID as primary key and mask length as secondary. This is so that after a database update it should be possible to reconstruct the database to verify the digest signature, which may be retrieved separately from the database for verification purposes.




Database Header

The DB Code indicates 0 if what follows is an entire database or 1 if what follows is an update. The database file version is incremented each time the complete database is generated by the authority. In the case of an update, the database file version indicates the new database file version, and the old database file version is indicated in the "old DB version" field. The database file version is used by routers to determine whether or not they have the most current

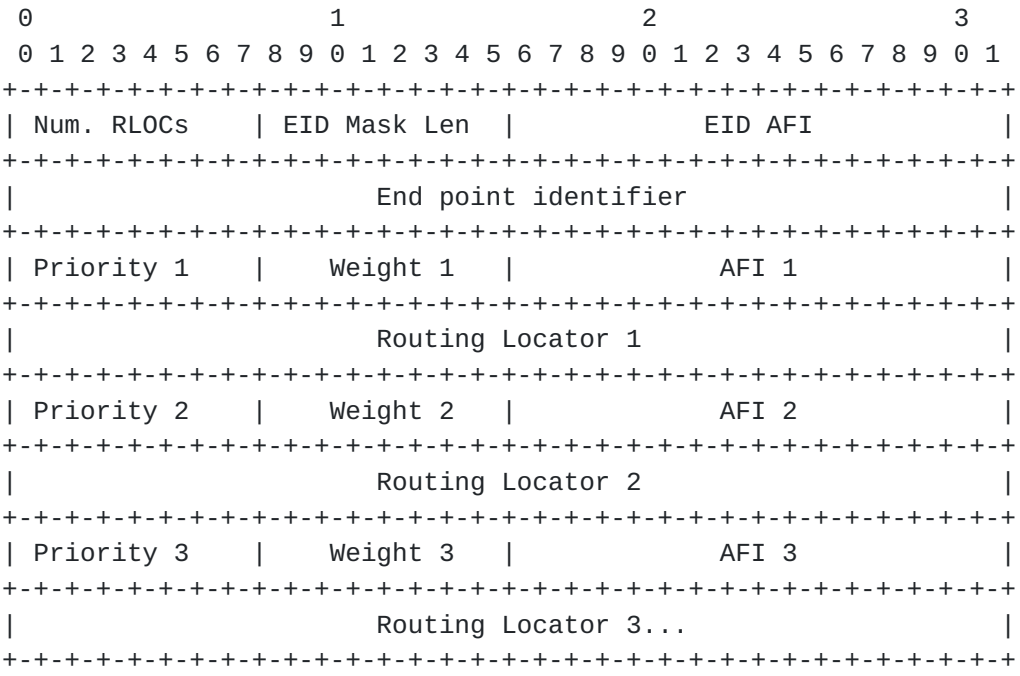
database.

The database name is a domain name. This is the name that will appear in the Subject field of the certificate used to verify the database. The purpose of the database name is to allow for more than one database. Such databases would be merged by the router. It is important that an EID/RLOC mapping be listed in no more than one database, lest inconsistencies arise. However, it may be possible to transition a mapping from one database to another. During the transition period, the mappings MUST be identical. When they are not, the resultant behavior will be undefined.

The PKCS#7 [4] authentication block contains a DER encoded [5] signature and associated public key.

### 3.1. NERD Record Format

As distributed over the network, NERD records appear as follows:




Priority N and Weight N, and AFI N are associated with Routing Locator N. There will always be at least one routing locator. The minimum record size for IPv4 is 16 bytes. Each additional IPv4 RLOC increases the record size by 8 bytes. The purpose of this format is to keep the database compact, but somewhat easily read. The meaning

of weight and priority are described in [1]. The format of the AFI is specified by IANA as "Address Family Numbers", with the exception of how IPv6 EID prefixes are stored.

In order to reduce storage and transmission amounts for IPv6, only the necessary number of bytes as specified by the prefix length are kept in the record, rounded to the nearest four byte (word) boundary. For instance, if the prefix length is /49, the nearest four-byte word boundary would require that eight bytes are stored. IPv6 RLOCs are represented as normal 128-bit IPv6 addresses.

### 3.2. Database Update Format

A database update contains a set of changes to an existing database. Each AFI/EID/mask-length tuple may have zero or more RLOCs associated with it. In the case where there are no RLOCs, the EID entry is removed from the database. Records that contain EIDs and mask lengths that were not previously listed are simply added. Otherwise, the old record for the EID and mask length is replaced by the more current information. The record format used by the a database update is the same as described in Section 3.1.

## 4. NERD Distribution Mechanism

### 4.1. Initial Bootstrap

Bootstrap occurs when a router needs to retrieve the entire database. It knows it needs to retrieve the entire database because either it has none or an update too substantial to process, as might be the case if a router has been out of service for a substantially lengthy period of time.

To bootstrap the ITR appends the database name plus "/current/entiredb" to a Base Distribution URI and retrieves the file via HTTP. For example, if the configured URI is "http://www.example.com/eiddb/", and assuming a database name of "nerd.arin.net", the ITR would request "http://www.example.com/eiddb/current/nerd.arin.net/entiredb". Routers MUST check the signature on the database prior to installing it, and MUST check that the database schema matches a schema they understand. Once a router has a valid database it MUST store that database in some sort of non-volatile memory (e.g., disk, flash memory, etc).

N.B., the host component for such URIs MUST NOT resolve to a LISP EID, lest a circular dependency be created.

### 4.2. Retrieving Changes

In order to retrieve a set of database changes an ITR will have previously retrieved the entire database. Hence it knows the current version of the database it has. Its first step for retrieving changes is to retrieve the current version of the database. It does so by appending "current/version" to the base distribution URI and retrieving the file. Its format is text and it contains the integer value of the current database version.

Once an ITR has retrieved the current version it compares version of its local copy. If there is no difference, then the router is up to date and need take no further actions until it next checks.

If the versions differ, the router next sends a request for the appropriate change file by appending "current/changes/" and the textual representation of the version of its local copy of the database to the base distribution URI. For example, if the current version of the database is 1105503 and router's version is 1105500, and the base URI and database name are the same as above, the router would request "http://www.example.com/eiddb/nerd.arin.net/current/changes/1105500".

The server may not have that change file, either because there are too many versions between what the router has and what is current, or because no such change file was generated. If the server has changes from the routers version to any later version, the server SHOULD issue an HTTP redirect to that change file, and the router SHOULD retrieve and process it. Once it has done so, the router should then repeat the process until it has brought itself up to date. It is thus important for servers to expire old change files in the order in which they were generated.

By way of convention, it is suggested that the URIs issued in redirects be of the following form:

{base dist. URI}/{dbname}/{more-recent-version}/changes/ {older-version}

where "base dist. URI" is the base distribution URI, "dbname" is the name of the database, and each version is the textual representation of the integer version value.

For example, if the current database version was 1105503 and a router made a request for "http://www.example.com/eiddb/nerd.arin.net/current/changes/1105400" but there was no change file from 1105400 to 1105503, and the server had a group of change files to make the router current, it would

issue a redirect to
"http://www.example.com/eiddb/nerd.arin.net/110450/changes/1105400"
that the router would then process. The router would then make a
request for
"http://www.example.com/eiddb/nerd.arin.net/current/changes/110450"
that the server would have.

While it is unlikely that database versions would wrap, as they consists of 32 bit integers, should the event occur, ITRs MUST attempt first to retrieve a change file when their current version number is within 10,000 of 2^32 and they see a version available that is less than 10,000. Barring the availability of a change file, the ITR MUST still assume that the database version has wrapped and retrieve a new copy.

## 5. Analysis

We will start our analysis by looking at how much data will be transferred to a router during bootstrap conditions. We will then look at the bandwidth required. Next we will turn our concerns to servers. Finally we will ponder the effect of providing only changes.

In the analysis below we treat the overhead of the database header as insignificant (because it is). The analysis should be similar, whether a single database or multiple databases are employed, as we would assume that no entry would appear more than once.

### 5.1. Database Size

By its very nature the information to be transported is relatively static and is specifically designed to be topologically insensitive. That is, every ITR is intended to have the same set of RLOCs for a given EID. While some processing power will be necessary to install a table, the amount required should be far less than that of a routing information database because the level of entropy is intended to be lower.

For purposes of this analysis, we will assume that the world has migrated to IPv6, as this increases the size of the database, which would be our primary concern. However, to mitigate the size increase, we have limited the size of the prefix transmitted. For purposes of this analysis, we shall assume an average prefix length of 64 bits.

Based on that assumption, Section 3.1 states that mapping information for each EID/Prefix includes a group of RLOCs, each with an

associated priority and weight, and that a minimum record size with IPv6 EIDs with at least one RLOC is 30 bytes uncompressed. Each additional IPv6 RLOC costs 20 bytes.

| 10^n EIDs | 2 RLOC | 4 RLOC | 8 RLOC |
|---|---|---|---|
| 4 | 500 KB | 900 KB | 1.70 MB |
| 5 | 5.0 MB | 9.0 MB | 17.0 MB |
| 6 | 50 MB | 90 MB | 170 MB |
| 7 | 500 MB | 900 MB | 1.70 GB |
| 8 | 5.0 GB | 9.0 GB | 17.0 GB |

Database size for IPv6 routes with average prefix length = 64 bits

Table 1

Entries in the above table are derived as follows:

$$E * (30 + 20 * (R - 1 ))$$

where E = number of EIDs (10^n), R = number of RLOCs per EID.

Our scaling target is to accommodate 10^8 multihomed systems, which is one order magnitude greater than what is discussed in [8]. At 10^8 entries, a device could be expected to use between 5 and 17 gigabytes of RAM for the mapping. No matter the method of distribution, any router that sits in the core of the Internet would require near this amount of memory in order to perform the ITR function. Large enterprise ETRs would be similarly strained, simply due to the diversity of of sites that communicate with one another. The good news is that this is not our starting point, but rather our scaling target, a number that we intend to reach by the year 2050. Our starting point is more likely in the neighborhood of 10^4 or 10^5 EIDs, thus requiring between 500KB and 17 MB.

## 5.2. Router Throughput Versus Time

| Table Size (10^N) | 1mb/s | 10mb/s | 100mb/s | 1gb/s |
|---|---|---|---|---|
| 6 | 8 | 0.8 | 0.08 | 0.008 |
| 7 | 80 | 8 | 0.8 | 0.08 |
| 8 | 800 | 80 | 8 | 0.8 |
| 9 | 8,000 | 800 | 80 | 8 |
| 10 | 80,000 | 8,000 | 800 | 80 |
| 11 | 800,000 | 80,000 | 8,000 | 800 |

Number of seconds to process NERD

Table 2

The length of time it takes to process the database is significant in models where the device acquires the entire table. During this period of time, either the router will be unable to route packets using LISP or it must use some sort of query mechanism for specific EIDs as the rest it populates its table through the transfer. Table 2 shows us that at our scaling target, the length of time it would take for a router using 1 mb/s of bandwidth is about 80 seconds. We can measure the processing rate in small numbers of hours for any transfer speed greater than that. The fastest processing time shows us as taking 8 seconds to process an entire table of 10^9 bytes and 80 seconds for 10^10 bytes.

## 5.3. Number of Servers Required

As easy as it may be for a router to retrieve, the aggregate information may be difficult for servers to transmit, assuming the information is transmitted in aggregate (we'll revisit that assumption later).

| # Simultaneous Requests | 10 Servers | 100 Servers | 1,000 Servers | 10,000 Servers |
|---|---|---|---|---|
| 100 | 720 | 72 | 72 | 72 |
| 1,000 | 7,200 | 720 | 72 | 72 |
| 10,000 | 72,000 | 7,200 | 720 | 72 |
| 100,000 | 720,000 | 72,000 | 7,200 | 720 |
| 1,000,000 | 7,200,000 | 720,000 | 72,000 | 7,200 |
| 10,000,000 | 72,000,000 | 7,200,000 | 720,000 | 72,000 |

Retrieval time per number of servers in seconds. Assumes average 10^8 entries with 4 RLOCs per EID and that each server has access to 1gb/s and 100% efficient use of that bandwidth and no compression.

Table 3

Entries in the above table were generated using the following method:

For 10^8 entries with four RLOCs per EID, the table size is 9.0GB, per our previous table. Assume 1 Gb/s transfer rates and 100% utilization. Protocol overhead is ignored for this exercise. Hence a single transfer X takes 48 seconds and can get no faster.

With this in mind, each entry is as follows:

$$\max(1X, N*X/S)$$

where N=number of transfers, X = 72 seconds, and S = number of servers.

If we have a distribution model which every device must retrieve the mapping information upon start, Table 3 shows the length of time in seconds it will take for a given number of servers to complete a transfer to a given number of devices. This table says, as an example, that it would take 72,000 seconds (20 hours) for one million ITRs to simultaneously retrieve the database from one thousand servers. Should a cold start scenario occur, this number should be of some concern. Hence it is important to take some measures both to avoid such a scenario, and to ease the load should it occur. The primary defense should be for ITRs to first attempt to retrieve their databases from their peers or upstream providers. Secondary defenses could include data sanity checks within ITRs, with agreed norms for how much the database should change in any given update or over any given period of time. As we will see below, dissemination of changes

is considerably less volume.

| % Daily Change | 100 Servers | 1,000 Servers | 10,000 Servers |
|---|---|---|---|
| 0.1% | 300 | 30 | 3 |
| 0.5% | 1500 | 150 | 15 |
| 1% | 3000 | 300 | 30 |
| 5% | 15,000 | 1500 | 150 |
| 10% | 30,000 | 3000 | 300 |

Assuming 10 million routers and a database size of 9GB, resulting hourly transfer times are shown in seconds, given number of servers and daily rate of change.

Table 4

This table shows us that with 10,000 servers the average transfer time with 1Gb/s links for 10,000,000 routers will be 300 seconds with 10% daily change spread over 24 hourly updates. For a 0.1% daily change, that number is 3 seconds for a database of size 9.0GB.

The amount of change goes to the purpose of LISP. If its purpose is to provide effective multihoming support to end customers, then we might anticipate relatively few changes. If, on the other, service providers attempt to make use of LISP to provide some form of traffic engineering, we can expect the same data to change more often. We can probably not conclude much in this regard without additional operational experience. The one thing we can say is that different applications of the LISP protocol may require new and different distribution mechanisms. Such optimization is left for another day.

## 5.4. Security Considerations

Whichever the answer to our previous question, we must consider the security of the information being transported. If an attacker can forge an update or tamper with the database, he can in effect redirect traffic to end sites. Hence, integrity and authenticity of the NERD is critical. In addition, a means is required to determine whether a source is authorized to modify a given database. No data privacy is required. Quite to the contrary, this information will be necessary for any ITR.

The first question one must ask is who to trust to provide the ITR a mapping. Ultimately the owner of the EID prefix is most authoritative for the mapping to RLOCs. However, were all owners to sign all such mappings, ITRs would need to know which owner is

authorized to modify which mapping, creating a problem of O(N^2) complexity.

We can reduce this problem substantially by investing some trust in a small number of entities that are allowed to sign entries. If authority manages EIDs much the same way a domain name registrar handles domains, then the owner of the EID would choose a database authority she or he trusts, and ITRs must trust each such authority in order to map the EIDs listed by that authority to RLOCs. This reduces the amount of management complexity on the ETR to retaining knowledge of O(#authorities), but does require that each authority establish procedures for authenticating the owner of an EID. Those procedures needn't be the same.

There are two classic methods to ensure integrity of data:

- secure transport of the source of the data to the consumer, such as Transport Layer Security (TLS) [7]; and
- provide object level security.

These methods are not mutually exclusive, although one can argue about the need for the former, given the latter.

In the case of TLS, when it is properly implemented, the objects being transported cannot easily be modified by interlopers or so-called men in the middle. When data objects are distributed to multiple servers, each of those servers must be trusted. As we have seen above, we could have quite a large number of servers, thus providing an attacker a large number of targets. We conclude that some form of object level security is required.

Object level security involves an authority signing an object in a way that can easily be verified by a consumer, in this case a router. In this case, we would want the mapping table and any incremental update to be signed by the originator of the update. This implies that we cannot simply make use of a tool like CVS [9]. Instead, the originator will want to generate diffs, sign them, and make them available either directly or through some sort of content distribution or peer to peer network.

### 5.4.1. Use of Public Key Infrastructures (PKIs)

X.509 provides a certificate hierarchy that has scaled to the size of the Internet. The system is most manageable when there are few certificates to manage. The model proposed in this memo makes use of one current certificate per database authority. The three pieces of information necessary to verify a signature, therefore, are as follows:

- the certificate of the database authority, which can be provided along with the database;
- the certificate authority's certificate; and
- A table of database names and distinguished names (DNs) that are allowed to update them.

The latter two pieces of information must be very well known and must be configured on each ITR. It is expected that both would change very rarely, and it would not be unreasonable for such updates to occur as part of a normal OS release process.

The tools for both signing and verifying are readily available. OpenSSL [16] provides tools and libraries for both signing and verifying. Other tools commonly exist.

Use of PKIs is not without implementation, operational complexity or risk. The following risks and mitigations are identified with NERD's use of PKIs:

If a NERD database authority private key is exposed:

In this case an attacker could sign a false database update, either redirecting traffic, or otherwise causing havoc. In this case, the NERD database administrator must revoke its existing key and issue a new one. The certificate is added to a certificate revocation list (CRL), which may be distributed with both this and other databases, as well as through other channels. Because this event is expected to be rare, and the number of database authorities is expected to be small, a CRL will be small. When a router receives a revocation, it checks it against its existing databases, and attempts to update the one that is revoked. This implies that prior to issuing the revocation, the database authority MUST sign an update with the new key. Routers SHOULD discard updates they have already received that were signed after the revocation was generated. If a router cannot confirm that whether the authority's certificate was revoked before or after a particular update, it MUST retrieve a fresh new copy of the database with a valid signature.

The private key associated with the CA that signed the Authority's certificate is compromised:

In this case, it becomes possible for an attacker to masquerade as the database authority. To ameliorate damage, the database authority SHOULD revoke its certificate and get a new certificate issued from a CA that is not compromised. Once it has done so, the previous procedure is followed. The compromised certificate

can be removed during the normal operating system upgrade cycle.

An algorithm used if either the certificate or the signature is cracked:

This is a catastrophic failure and the above forms of attack become possible. The only mitigation is to make use of a new algorithm. In theory this should be possible, but in practice has proved very difficult. For this reason, additional work is recommended to make alternative algorithms available.

The Database Authority loses its key or disappears:

In this case nobody can update the existing database. There are few programmatic mitigations. If the database authority places its private keys and suitable amounts of information escrow, under agreed upon circumstances, such as no updates for three days, for example, the escrow agent would release the information to a party competent of generating a database update.

### 5.4.2. Other Risks

Because this specification does not require secure transport, if an attacker prevents updates to an ITR for the purposes of having that ITR continue to use a compromised ETR, the ITR could continue to use an old version of the database without realizing a new version has been made available. If one is worried about such an attack, a secure channel such as SSL to a secure chain back to the database authority should be used. It is possible that after some operational experience, later versions of this format will contain additional semantics to address this attack.

As discussed above, substantial risk would be a cold start scenario. If an attacker found a bug in a common operating system that allowed it to erase an ITR's database, and was able to disseminate that bug, the collective ability of ITRs to retrieve new copies of the database could be taxed by collective demand. The remedy to this is for devices to share copies of the database with their neighbors, thus making each potential requester a potential service.

## 6. Why not use XML?

Many objects these days are distributed as either XML pages or something derived as XML [11], such as SOAP [12],[13]. Use of such well known standards allows for high level tools and library reuse. XML's strength is extensibility. Without a doubt XML would be more

extensible than a fixed field database. Why not, then, use these standards in this case? There are two answers to this question. First, the obvious concern is that XML is not known for efficiency of data transport. Being based in text, an IPv4 address is expanded from one octet to three octets, plus either an attribute and quotes or element tags and end tags. Let us presume for the moment a very simple schema that might cause a record to be represented as follows:

```
<r e="10.1.1.0" m="24">
  <l w="10" p="15">
    <v4>
    192.168.1.1
    </v4>
 </l>
  <l w="5" p="15">
    <v4>
    192.168.1.2
    </v4>
 </l>
</r>
```

With white space removed the uncompressed XML represents 120 bytes versus 20 bytes for the record specified in Section 3.1, representing a five fold expansion. The expansion rate for IPv6 is a bit more complex. Because the textual representation can be shortened, it's hard to say exactly what length an IPv6 address would be.

The other concern about XML is that version 1.0 of the specification is silent on the order of sibling elements. Specifications other than the base specification state that order is significant. Order is significant to LISP and NERD because once an update is applied to the database it should be possible to verify the signature of the entire database. Prior to applying the signature the XML generator would need to ensure the order of information. That same sort would be required of the router. This seems to add unnecessary fragility to a critical system without much benefit. While there may indeed be uses of an XML representation of the database, these uses are likely to be outside of a router.

## 7. Perhaps use a hybrid model?

Perhaps it would be useful to use both a prepopulated database such as NERD and a query mechanism (perhaps LISP+ALT, LISP-CONS [15], or DNS) to determine an EID/RLOC mapping. One idea would be to receive a subset of the mappings, say, by taking only the NERD for certain

regions. This alleviates the need to drop packets for some subset of destinations under the assumption that one's business is localized to a particular region. If one did not have a local entry for a particular EID one would then make a query.

One approach to using DNS to query live would be to periodically walk "interesting" portions of the network, in search of relevant records, and caching them to non-volatile storage. While preventing resource attacks, the walk itself could be viewed as an attack, if the algorithm was not selective enough about what it thought was interesting. A similar approach could be applied to LISP+ALT or LISP-CONS by forcing a data-driven Map Reply for certain sites.

## 8. Deployment Issues

While LISP and NERD are intended as experiments at this point, it is already obvious one must give serious consideration to circular dependencies with regard to the protocols used and the elements within them.

### 8.1. HTTP

In as much as HTTP depends on DNS, either due to the authority section of a URI, or due to the configured base distribution URI, these same concerns apply. In addition, any HTTP server that itself makes use of provider independent addresses would be a poor choice to distribute the database for these exact same reasons.

One issue with using HTTP is that it is possible that a middlebox of some form, such as a cache, may intercept and process requests. In some cases this might be a good thing. For instance, if a cache correctly returns a database, some amount of bandwidth is conserved. On the other hand, if the cache itself fails to function properly for whatever reason, end to end connectivity could be impaired. For example, if the cache itself depended on the mapping being in place and functional, a cold start scenario might leave the cache functioning improperly, in turn providing routers no means to update their databases. Some care must be given to avoid such circumstances.

## 9. Open Questions

Do we need to discuss reachability in more detail? This was clearly an issue at the IST-RING workshop. There are two key issues. First, what is the appropriate architectural separation between the data plane and the control plane? Second, is there some specific way in

which NERD impacts the data plane?

Should we specify a (perhaps compressed) tarball that treads a middle ground for the last question, where each update tarball contains both a signature for the update and for the entire database, once the update is applied.

Should we compress? In some initial testing of databases with 1, 5, and 10 million IPv4 EIDs and a random distribution of IPv4 RLOCs, the current format in this document compresses down by a factor of between 35% and 36%, using Burrows-Wheeler block sorting text compression algorithm (bzip2). The NERD used random EIDs with mask lengths varying from 19-29, with probability weighted toward the smaller masks. This only very roughly reflects reality. A better test would be to start with the existing prefixes found in the DFZ.

## 10. Conclusions

This memo has specified a database format, an update format, a URI convention, an update method, and a validation method for EID/RLOC mappings. We have shown that beyond the predictions of 10^8 EID-prefix entries, the aggregate database size would likely be at most 17GB. We have considered the amount of servers to distribute that information and we have demonstrated the limitations of a simple content distribution network and other well known mechanisms. The effort required to retrieve a database change amounts to between 3 and 30 seconds of processing time per hour at at today's gigabit speeds. We conclude that there is no need for an off box query mechanism today, and that there are distinct disadvantages for having such a mechanism in the control plane.

Beyond this we have examined alternatives that allow for hybrid models that do use query mechanisms, should our operating assumptions prove overly optimistic. Use of NERD today does not foreclose use of such models in the future, and in fact both models can happily co-exist.

We leave to future work how the list of databases is distributed, how BGP can play a role in distributing knowledge of the databases, and how DNS can play a role in aggregating information into these databases.

We also leave to future work whether HTTP is the best protocol for the job, and whether the scheme described in this document is the most efficient. One could easily envision that when applied in high delay or high loss environments, a broadcast or multicast method may prove more effective.

## 11. IANA Considerations

This memo makes no requests of IANA.

## 12. Acknowledgments

Dino Farinacci, Patrik Faltstrom, Dave Meyer, Joel Halpern, Dave Thaler, Mohamed Boucadair, Robin Whittle, Max Pritikin, and Scott Brim were very helpful with their reviews of this work. Thanks also to the participants of the Routing Research Group and the IST-RING workshop held in Madrid in December of 2007 for their incisive comments. The astute will notice a lengthy References section. This work stands on the shoulders of many others' efforts.

## 13. References

### 13.1. Normative References

[1] Farinacci, D., "Locator/ID Separation Protocol (LISP)", draft-farinacci-lisp-06 (work in progress), February 2008.

[2] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[5] International Telecommunications Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, ISO Standard 9594-8, March 2000.

[6] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

### 13.2. Informational References

[7] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[8] Carpenter, B., "IETF Plenary Presentation: Routing and

Addressing: Where we are today", March 2007.

[9] Grune, R., Baalbergen, E., Waage, M., Berliner, B., and J. Polk, "CVS: Concurrent Versions System", November 1985.

[10] International International Telephone and Telegraph Consultative Committee, "Information Technology - Open Systems Interconnection - The Directory: Authentication Framework", CCITT Recommendation X.509, November 1988.

[11] Bray, T., Paoli, J., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML) 1.0 (2nd ed)", W3C REC-xml, October 2000, <http://www.w3.org/TR/REC-xml>.

[12] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and H. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework", W3C Working Draft soap12-part1, June 2002, <http://www.w3.org/TR/soap12-part1>.

[13] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and H. Nielsen, "SOAP Version 1.2 Part 2: Adjuncts", W3C Working Draft soap12-part2, June 2002, <http://www.w3.org/TR/soap12-part2>.

[14] Farinacci, D., "LISP Alternative Topology (LISP-ALT)", draft-fuller-lisp-alt-01 (work in progress), November 2007.

[15] Brim, S., "LISP-CONS: A Content distribution Overlay Network Service for LISP", draft-meyer-lisp-cons-03 (work in progress), November 2007.

URIs

[16] <http://www.openssl.org>

## Appendix A. Generating and verifying the database signature with OpenSSL

As previously mentioned, one goal of NERD was to use off-the-shelf tools to both generate and retrieve the database. To many, PKI is magic. This section is meant to provide at least some clarification as to both the generation and verification process, complete with command line examples. Not included is how you get the entries themselves. We'll assume they exist, and that you're just trying to sign the database.

To sign the database, to start with, you need a database file that

has a database header described in Section 3. Block size should be zero, and there should be no PKCS#7 block at this point. You also need a certificate and its private key with which you will sign the database.

The OpenSSL "smime" command contains all the functions we need from this point forth. To sign the database, issue the following command:

```
openssl smime -binary -sign -outform DER -signer yourcert.crt \
        -inkey yourcert.key -in database-file -out signature
```

-binary states that no MIME canonicalization should be performed. -sign indicates that you are signing the file that was given as the argument to -in. The output format (-outform) is binary DER, and your public certificate is provided with -signer along with your key with -inkey. The signature itself is specified with -out.

The resulting file "signature" is then copied into to PKCS#7 block in the database header, its size in bytes is recorded in the PKCS#7 block size field, and the resulting file is ready for distribution to ITRs.

To verify a database file, first retrieve the PKCS#7 block from the file by copying the appropriate number of bytes into another file, say "signature". Next, zero this field, and set the block size field to 0. Next use the "smime" command to verify the signature as follows:

```
openssl smime -binary -verify -inform DER -content database-file
        -out /dev/null -in signature
```

Openssl will return "Verification OK" if the signature is correct. OpenSSL provides sufficiently rich libraries to accomplish the above within the C programming language with a single pass.

## Appendix B. Changes

This section to be removed prior to publication.

- 04: Analysis change: IPv6 RLOCs are 128 bits. While they can be shortened to 64 bits, that involves substantial ETR changes and expenditure of IPv6 networks, which is probably unnecessary, and can be left as a later optimization. Added an option of independent operators. Processed all but two of Dino's comments. Addressed Scott's comments. Removed existing work analysis. Saving that for another day. Clarified OpenSSL Appendix.
- 03: Change dbname to a domain name, indicate that is what is in the subject of the X.509 certificate, and list editorial changes, update acknowledgments.
- 02: Incorporate some of Dave Thaler's comments. Add authentication block detail. Modify analysis to take IPv6 into account, along with a more realistic number of RLOCs per EID. Add some comments about potential risks of a cold start. Add S/MIME example as appendix A and take out old ToDo. Provide some amount of compression of IPv6 addresses by limiting their size to significant bytes rounded to a four byte word boundary.
- 01: Massive spelling correction, URI example correction.
- 00: Initial Revision.

## Author's Address

Eliot Lear
Cisco Systems GmbH
Glatt-com
Glattzentrum, ZH CH-8301
Switzerland

Phone: +41 44 878 7525
Email: lear@cisco.com

## Full Copyright Statement



## Intellectual Property