Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

J. Gersch
Secure64 SW Corp
D. Massey
Colorado State University
E. Osterweil
Verisign
C. Olschanowsky
Colorado State University

February 25, 2013

Reverse DNS Naming Convention for CIDR Address Blocks
draft-gersch-dnsop-revdns-cidr-04.txt

Abstract

This draft proposes a naming convention for encoding CIDR address blocks into the reverse DNS namespace. The reverse DNS naming method is commonly used to specify a complete IP address. This document describes how to encode an IPv4 or IPv6 CIDR address block such as 129.82.128.0/17. By defining a common naming convention, one can associate information with a prefix. The convention builds on past work in RFC 1101 that associates network names with prefixes. However, this previous work pre-dated the introduction of CIDR and has several critical ambiguities. This convention corrects the ambiguities and enables new applications ranging from routing information to geolocation.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on August 29, 2013.

Table of Contents

1. Introduction

This draft proposes a common naming convention for entering CIDR prefixes into the Reverse DNS.

The Reverse DNS provides a naming convention for both IPv4 and IPv6 addresses. At this time, the most common use of the reverse-DNS is to associate an IP address with a PTR resource record that identifies the corresponding host name. For example, IP address 129.82.138.2 is encoded as 2.138.82.129.in-addr.arpa and a PTR resource record identifies the host name as alpha.netsec.colostate.edu. The Reverse DNS would be more expressive if we had a formal convention for encoding and returning information associated with a network address range, not just a unique IP address. For example, the naming convention in this document allows one to store and resolve resource records associated with a prefix range such as 129.82.128/17.

The association of prefixes and data using reverse DNS has existing applications. Specifically, [RFC1035] (section 3.5) uses the reverse DNS to identify gateways on a subnet and [RFC1101] associates a network name with an address block. The introduction of the CIDR addressing architecture created ambiguities for naming conventions such as RFC 1101. This document introduces a naming convention that resolves the ambiguities, restores the historical uses, and enables new uses such as the inclusion of routing data and geolocation data. This list of possible applications is not intended to be complete, but instead suggest some of the possibilities.

This draft proposes a naming convention for the prefix name and argues that applications would benefit from the use of consistent convention. Since it is only a naming convention, it requires no changes to the DNS servers or resolvers. It simply provides a way to express a prefix as a unique DNS name. DNS zone administrators can choose to associate any name with a prefix, but having a common convention facilities inter-operability between different applications. In fact, there is already DNS data using this document's naming convention in the reverse DNS. Standardizing the convention allows it to be improved, clearly documented, and allows other applications to make use of the same naming convention.

1.1. Aligning the DNS and IP Hierarchies

Both the DNS names and IP addresses are part of a hierarchical tree structure and any naming convention should respect and align with these tree structures whenever possible. In the DNS hierarchical tree structure 128.82.129.in-addr.arpa is logically below 82.129.in-addr.arpa, which is logically below 129.in-addr.arpa. Other "flat" approaches to naming, such as Distributed Hash Tables, have been

proposed, but the DNS tree structure remains a powerful abstraction. It forms the basis for the operation of DNS; caching, delegation, DNSSEC signing, and so forth all benefit from the DNS tree structure.

IP addresses also have a logical tree structure where 129.82.128.0/24 is subprefix (logically below) 129.82.0.0/16 which is a subprefix of 129.0.0.0/8. The reverse DNS aligns with the structure; 128.82.129.in-addr.arpa is logically below 82.129.in-addr.arpa which is logically below 129.in-addr.arpa. This alignment between the DNS hierarchy and the IP address hierarchy serves both systems well and allows one to easily encode prefixes that fall on an octet boundary (e.g. IPv4 prefixes whose mask length is a multiple of 8).

The challenge is to preserve this alignment even when even when CIDR prefixes do not fall on octet boundaries. For example, 129.82.128.0/19 is a subprefix of 129.82.128.0/18. The DNS name for 129.82.128.0/19 should be logically below the DNS name for 129.82.128.0/18. This document introduces a naming convention for CIDR prefixes that preserves this alignment while also building on the existing reverse DNS structure.

1.2. Purpose

In order to enable the association of prefixes and data using reverse DNS, one must map an IPv4 or IPv6 prefix into a reverse DNS name. There are various subtleties, advantages and disadvantages that emerge when trying to define a naming convention. This requires no DNS protocol changes and no modifications to resolvers, caches, or authoritative servers. Today, zone administrators can use their own individual approaches to encode a prefix in the reverse DNS. The emergence of different encoding standards complicates (but does not prevent) the design of systems that would make use of these resource records. The aim of this work is to introduce a standard convention.

1.3. Terminology

The following terms are used throughout out the document:

Reverse DNS:
: We use the term Reverse DNS to refer to the domains in-addr.arpa and ip6.arpa.

Prefix:
: A prefix refers to IPv4 or IPv6 address range specified by a network portion and mask length, as described in [RFC4632]. For example, 129.82.0.0/16 and 129.82.128/18 are examples of IPv4 prefixes.

Octet Boundary:
An IPv4 prefix falls on an octet boundary if its mask length is a multiple 8. For example, 129.82.0.0/16 is on an octet boundary while 129.82.128/18 is not. Prefixes that are on octet boundary naturally map to the reverse DNS. Prefixes that do not fall on an octet boundary are more complex and are the main challenge for any naming convention.

Nibble Boundary:
An IPv6 prefix falls on a nibble boundary if its mask length is a multiple 4. For example, 2607:fa88::/32 is on a nibble boundary while 2607:fa88::/33 is not. Prefixes that are on nibble boundary naturally map to the reverse DNS. Prefixes that do not fall on a nibble boundary are more complex and are the main challenge for any naming convention.

2. Conventions Used In This Document

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 [RFC2119].

3. Design Requirements

A naming convention to specify CIDR address blocks in the reverse-DNS must satisfy the following requirements:

1. Unambiguous: A prefix must have a unique name and a name must uniquely match a single prefix. It is very important that a prefix should have only one unique DNS name. If there are multiple DNS names for the same prefix, applications might need to query data at each of the multiple names. Worse still, the different names could contain conflicting information. To avoid this, we require each prefix have exactly one unique DNS name. It is equally important that a DNS name maps to only one prefix. If the same name maps to more than one prefix, applications cannot distinguish which records should be be associated with which prefix. To avoid this, we require each name in our naming convention maps to exactly one unique prefix.

2. Autonomy: The owner of a reverse-DNS zone file associated with a CIDR address block should be able to act independently from any other organization in order to create or modify data records within the DNS zone.

3. Coverage Authority: With the exception of data that has been sub-delegated to a child zone, the reverse DNS zone must be authoritative for all sub-prefixes below the covering prefix. Any query for a sub-prefix must be answered with a data record or NXDOMAIN specifying this zone as the authority.

4. Delegation: It must allow the zone owner to delegate smaller address blocks to a child zone which will be independently managed.

5. Conformance: It should align with naming conventions and delegation structures already in use by the RIRs for IN-ADDR.ARPA and IP6.ARPA.

6. Simplicity: The naming structure should be understandable, or at a minimum, able to be easily constructed by software provisioning tools and utilities such as DIG.

4. Reverse DNS CIDR Name Specification

The naming method described in this section is based on the well-known technique of performing a binary AND to a bit-mask and the low-order octet of an IP address. The result is then broken up into individual sub-names using the "." separator. The result looks like an ENUM or IPv6 reverse-DNS address; that is, a string of chained empty non-terminal sub-names.

This name-chaining creates the desired effect of enabling a DNS zone delegation at any point in the chain. For example, the naming scheme allows for the creation of two /17's from a /16, two /18's from a /17.

4.1. IPv4 Address Block Naming

The CIDR to Reverse-DNS name conversion is performed as follows:

1. Remove any octets that are not significant. An octet is significant if it includes any part of the network address. An octet is not significant if all bits correspond to the host portion of the address. For example, 129.82.0.0/16 --> 129.82 and 129.82.160.0/19 --> 129.82.160

2. If the prefix falls on an octet boundary: first, invert the address and insert a "m" label as the first label to indicate this is a prefix name and, then, append in-addr.arpa to the end e.g. 129.82 --> m.82.129.in-addr.arpa.

3. If the prefix does not fall on an octet boundary:

    A. Truncate the name to remove the least significant octet. Add a "m" label to this domain name to indicate "mask".

    B. Convert the least significant octet to binary, separating each bit into its own label (with a "." character).

    C. Truncate the binary labels to the N significant labels that correspond to the given prefix_length.

    D. Reverse the string and add ".in-addr.arpa."

Several examples illustrate this algorithm. These examples show the conversion to binary, followed by the truncation, followed by the name reversal.

    129.82.0.0/16 --> m.82.129.in-addr.arpa. (at octet boundary)

```
129.82.64.0/18  --> 129.82.m.0.1.0.0.0.0.0.0
                --> 129.82.m.0.1 (N = 18 mod 8 = 2)
                --> 1.0.m.82.129.in-addr.arpa.

129.82.64.0/20  --> 129.82.m.0.1.0.0.0.0.0.0
                --> 129.82.m.0.1.0.0  (N = 20 mod 8 = 4)
                --> 0.0.1.0.m.82.129.in-addr.arpa.

129.82.160.0/20 --> 129.82.m.1.0.1.0.0.0.0.0
                --> 129.82.m.1.0.1.0 (N = 20 mod 8 = 4)
                --> 0.1.0.1.m.82.129.in-addr.arpa.

129.82.160.0/23 --> 129.82.m.1.0.1.0.0.0.0.0
                --> 129.82.m.1.0.1.0.0.0.0 (N = 23 mod 8 = 7)
                --> 0.0.0.0.1.0.1.m.82.129.in-addr.arpa.

15.192.0.0/12   --> 15.192.m.1.1.0.0.0.0.0.0
                --> 15.192.m.1.1.0.0    (N = 12 mod 8 = 4)
                --> 0.0.1.1.m.15.in-addr.arpa.
```

The Reverse-DNS name to CIDR conversion is performed as follows:

1. Drop ".in-addr.arpa" from the string.

2. Calculate the prefix length from the name using the formula:

    p_len = 8*(token count after "m") + (token count before "m")

    Each token comprises the digits (or letter) between periods.

3. Reverse the order of the tokens in the string.

4. Use the binary values at the end to calculate the final octet. To do this take the binary values represented and left shift them until there are 8 bits, convert to decimal.

5. Remove the m token and all tokens following it; replace them with the decimal representation of the binary value.

6. Append a "/" and the prefix length.

Examples:

    1.0.m.82.129.in-addr.arpa --> 129.82.64.0/18

(example has 2 octets + 2 binary digits, so mask length = 18)

0.0.1.0.m.82.129.in-addr.arpa --> 129.82.64.0/20
(example has 2 octets + 4 binary digits, so mask length = 20)

0.0.0.1.0.1.m.129.in-addr.arpa--> 129.160.0/14
(example has 1 octet + 6 binary digits, so mask length = 14)

4.2. IPv6 Address Block Naming

The IPv6 naming convention is similar, with the exception that 4-bit nibble boundaries are used instead of octets, and "ip6.arpa" is used as the suffix.

Examples:

2607:fa88::/32 --> m.8.8.a.f.7.0.6.2.ip6.arpa
(on nibble boundary)

2607:fa88:8000::/33 --> 2.6.0.7.f.a.8.8.m.1.0.0.0
--> 2.6.0.7.d.a.8.8.m.1 (33 mod 4 = 1)
--> 1.m.8.8.a.f.7.0.6.2.ip6.arpa

2607:fa88:e000::/35 --> 2.6.0.7.f.a.8.8.m.1.1.1.0
--> 2.6.0.7.d.a.8.8.m.1.1.1(35 mod 4 = 3)
--> 1.1.1.m.8.8.a.f.7.0.6.2.ip6.arpa

4.3. Maintaining one-to-one mapping

We note that the naming convention uses the letter "m" to indicate a transition from octet/nibble numbering to binary numbering for the remainder of the name. Nothing restricts a DNS administrator from creating a name in which the sequence of binary digits extends past the next octet or nibble boundary. Applications may actually find this to be a useful capability. Nevertheless, this document defines a naming convention where each prefix maps to a unique name, as described in section 5. We therefore add the restriction that any application looking for records associated with a prefix MUST check standard naming convention (e.g. m.0.82.129.in-addr.arpa at an octet boundary) and if the desired records are found, the application MUST prefer these records over any records found at a non-standard encoding.

5. Related Work

The process of mapping CIDR addresses into the reverse-DNS name space is difficult because the prefix length of an IPv4 CIDR address is an arbitrary number from 0 to 32. These numbers do not necessarily align with an IPv4 octet.

The problem of associating records with network names dates back to [RFC1035]. This RFC uses 10.in-addr.arpa to represent net 10 and uses PTR records to identify gateways on net 10. This works intuitively for simple classful network such as 10/8 and sets the stage for future work, but fails to fully specify a convention. For example, it does not show how to represent mask lengths that don't match the classful boundary and clearly does not address arbitrary mask lengths; CIDR addresses were not yet defined.

5.1. Naming via RFC 1101

[RFC1101] addressed how to add subnet masks by introducing both reverse DNS conventions and pointers to names in the forward DNS tree (e.g. DNS zones not under in-addr.arpa). However, this RFC also pre-dates the existence of CIDR addresses and some small ambiguities became more pronounced with the introduction of CIDR prefixes. These ambiguities make the convention infeasible for current applications.

To illustrate the problem, suppose one wants to associate a network name and some additional information with the address blocks 129.82.0.0/16, 129.82.0.0/18, and 129.82.0.0/20. RFC 1101 uses PTR records to encode the network name and A records to define the existence of a subnet with a specified mask length. We will use a TXT record to store the additional information. The TXT record is simply meant to represent an arbitrary RR type.

The following entries would be added to the appropriate enclosing zone:

```
;129.82.0.0/16 network name and additional information
0.0.82.129.in-addr.arpa    IN   PTR   topnet.myzone.example.
0.0.82.129.in-addr.arpa    IN   TXT   "my additional info"

; define and name the 129.82.0.0/18 network name
0.0.82.129.in-addr.arpa    IN   PTR   subnet1.myzone.example.
0.0.82.129.in-addr.arpa    IN   A     255.255.192.0;  /18 mask
0.0.82.129.in-addr.arpa    IN   TXT   "/18 additional info"

; define and name the 129.82.0.0/20 network name
0.0.82.129.in-addr.arpa    IN   PTR   subnet2.myzone.example.
0.0.82.129.in-addr.arpa    IN   A     255.255.240.0  ; /20 mask
```

```
0.0.82.129.in-addr.arpa    IN   TXT   "/20 additional info"
```

The first A record indicates there is a 129.82.0.0/18 subnet defined. The second A record indicates there is a 129.82.0.0/20 subnet defined. The ambiguity arises when wants to obtain the TXT (or PTR or any other RR type) associated with 129.82.0.0/18. A query will return the three records in the TXT RRSet at 0.0.82.129.in-addr.arpa. Only one of these TXT RRs is associated with 129.82.0.0/18 and there is no way to determine which of the three is the correct one.

This naming convention fails the "Unambiguous" requirement. We do not consider additional issues since the above ambiguity makes the RFC 1101 approach infeasible. The naming convention introduced later in this document builds on the main concepts in this RFC, resolves the ambiguity, explicitly expands to more than just network names, and addresses our design goals and operational concerns.

5.2. CIDR Naming via RFC 2317

According to [RFC2317], the purpose of the document is to describe "a way to do IN-ADDR.ARPA delegation on non-octet boundaries for address spaces covering fewer than 256 addresses." It is not a general naming scheme for prefixes. However, some would argue that it might be extended into a general naming convention.

To create a naming convention based on [RFC2317], we note a representative example maps prefix 192.0.2.0/25 to the DNS name 0/25.2.0.192.in-addr.arpa. More generally, a prefix of the form A.B.C.D/M is mapped to the name D/M.C.B.A.in-addr.arpa. IPv6 prefixes are not discussed and the IPv4 mask length is assumed to be strictly greater than 24.

This approach does not satisfy the unambiguous requirement for prefixes with a mask length smaller than 24. For example, the [RFC2317] style name for prefix 129.82.0.0/16 maps to the DNS name 0/16.0.82.129.in-addr.arpa. It also maps to the names 0/16.82.129.in-addr.arpa, 82/16.129.in-addr.arpa, and 82.129.in-addr.arpa. [RFC2317] does not defines which of these is correct. This is not a flaw in the RFC. Instead, [RFC2317] says it is designed for "address spaces covering fewer than 256 addresses". 129.82.0.0/16 covers over 65,000 addresses, is clearly out scope, and thus a name for this prefix is not specified.

To extend this to a general naming convention, let 129.82.0.0/16 map to the DNS name 82.129.in-addr.arpa, 129.82.0.0/18 map to the DNS name 0/18.82.129.in-addr.arpa, and 129.82.0.0/20 map to the DNS name 0/20.82.129.in-addr.arpa. This mapping improves on the previous [RFC1101] approach in that each prefix now has a unique name, but we

show the approach has several other critical flaws.

One limitation is that the scheme is flat rather than hierarchical. In the prefix hierarchy, 129.82.0.0/18 is descendant of 129.82.0.0/16. In the DNS tree, the name 0/18.82.129.in-addr.arpa is a descendant of 82.129.in-addr.arpa. This is the desired property, but is only preserved at the octet boundary.

In the prefix hierarchy, 129.82.0.0/20 is descendant of 129.82.0.0/18. In the DNS tree, the name 0/18.82.129.in-addr.arpa and 0/20.82.129.in-addr.arpa are siblings, both are direct descendants of 82.129.in-addr.arpa. The hierarchical prefix structure is not preserved and mapped into a single flat space. Worse still, all /17, /18, /19, /20, /21, /22, and /23 prefixes are siblings. There is no hierarchical relationship whatsoever for prefixes that don't fall on an octet boundary, violating the Coverage Authority and Delegation requirements.

This document proposes a naming convention that adds as much hierarchy as possible, while still preserving the existing reverse DNS tree structure. In our approach, the name for 129.82.0.0/20 is a descendant of the name for 129.82.0.0/18.

Overall, [RFC2317] was not intended to encode IPv4 prefixes with a mask length smaller than 24 and it does not consider IPv6. Because the namespace is flat, it fails to meet the design requirements of Coverage Authority, Allowing Delegation, and arguably Simplicity.

5.3. Prior Work on CIDR Names for Routing

Over a decade ago, [I-D.bates-bgp4-nlri-orig-verif] proposed to use the reverse DNS to verify the origin AS associated with a prefix. This requires both a naming convention for converting the name into a prefix and additional resource record types for storing origin information, along with recommendations on their use.

Our focus in this draft is on the naming convention. Draft [I-D.bates-bgp4-nlri-orig-verif] as well as other subsequent work on BGP security, extends [RFC2317] style names to encode a prefix. For example, the draft proposes to encode the prefix 10.1.128/20 as the DNS name 128/20.1.10.bgp.in-addr.arpa.

In [I-D.bates-bgp4-nlri-orig-verif], the DNS hierarchy and the IP address hierarchy diverge and the approach fails to meet the Coverage Authority requirement. To see this, consider the prefixes 10.1.128/20 and 10.1.128/21. in CIDR terminology, 10.1.128/21 is covered by 10.1.128/20, but this relationship is not captured in the DNS hierarchy. 10.1.128/21 is encoded as 128/21.1.10.bgp.in-addr.arpa

and thus 10.1.128/20 and 10.1.128/21 are siblings in the DNS tree structure.

This can be overcome by introducing a large number of CNAME records; one for every potential subprefix. We instead provide an approach where the CIDR hierarchy and DNS hierarchy align.

6. Additional Considerations

This draft proposes a naming convention for IPv4 and IPv6 prefixes. With the introduction of a such a convention, a number of new possibilities are enabled and a number of issues have been raised. In this section, we summarize some of the main discussions. Though these are not directly part of the naming convention, they do help to review issues that may help application designers make better use of prefix names and help operators manage reverse zones. We first discuss how the naming convention interacts with the current octet (IPv4) or nibble (IPv6) based reverse DNS tree structure and then turn to the problem of prefix enumeration and find the longest match for a prefix.

6.1. Splitting a /16 into two /17s

Suppose organization X has been allocated 10.10.0.0/16 by SomeRIR. Organization X assigns 10.10.0/17 to Organization Y and assigned 10.10.128/17 to Organization Z. Concerns have been raised that Organization X needs to create 256 delegations. More precisely, Organization X needs to delegate 0.10.10.in-addr.arpa, 1.10.10.in-addr.arpa, up to 127.10.10.in-addr.arpa to Organization Y. Similarly, 128.10.10.in-addr.arpa up to 255.10.10.in-addr.arpa need to be delegated to Organization Z. This is the current practice and the naming convention here does not change this.

The naming convention described in this document requires no changes to the existing delegation operations. The naming convention does add one additional delegation, 0.m.10.10.in-addr.arpa, to Organization Y. Similarly, the naming convention does add one additional delegation, 1.m.10.10.in-addr.arpa, to Organization Z.

6.2. Allocating a /16 and then assigning the /16

Suppose organization X has been allocated 10.10.0.0/16 by SomeRIR. Organization X assigns 10.10.0/16 to Organization Y. SomeRIR needs to update the delegation (NS and DS records) in the 10.in-addr.arpa zone to point to Organization Y. Again, this is the current practice and the naming convention here does not change this.

6.3. Delegations that Span Octet boundaries

Suppose organization X has been allocated 10.0.0.0/10 by SomeRIR. Organization X assigns 10.0.128/17 to Organization Y. SomeRIR allocates 10.0/10 to Organization X, SomeRIR should also delegate the 64 zones 0.10.in-addr.arpa, 1.10.in-addr.arpa, ... 63.10.in-addr.arpa to Organization X. Organization Y should be delegated the 128 zones from 128.0.10.in-addr.arpa to 255.0.10.in-addr.arpa. Note all these

delegations come from the 0.10.in-addr.arpa zone, so SomeRIR is not involved in the delegation. Again, this is the current practice and the naming convention here does not change this.

SomeRIR should also delegate the 0.0.m.10.in-addr.arpa namespace to Organization X. Similarly, Organization X should also delegate the namespace 1.m.0.10.in-addr.arpa to Organization Y. Note this delegation comes from the 0.10.in-addr.arpa zone run by Organization X and SomeRIR is not involved in the delegation.

The addition of the new naming convention did not obsolete the need to add to delegate the various octet boundary zones. In other words, one still needs to continue the practice of delegating zones like 0.10.in-addr.arpa and 128.0.10.in-addr.zones. This draft intentionally works with the existing reverse DNS tree and does not change the practices for existing octet boundary zones.

6.4. Legacy Behavior at Octet Boundaries

The existing reverse DNS structure is aligned on octet boundaries for IPv4 and nibble boundaries for IPv6. The naming convention introduced here adds to the existing reverse DNS tree; it does not change the existing structure. This is a deliberate choice not to reinvent the reverse DNS but rather to enhance the existing structure. The naming convention proposed here builds on the existing reverse DNS structure and thus inherits both advantages and disadvantages from the existing system.

6.5. The Naming Convention and Zone Structures

The naming convention does not impose any semantics on zone structure. As with any DNS name, a resolver need not be aware of how the zone cuts are structured and no specific requirements are added for zone management. For example, some sites may choose to delegate at a subprefix boundary while others maintain one large zone. Names can make use of CNAME and DNAME records if the zone administrator so desires. This is simply a naming convention and does not change any existing resolver or server behavior.

6.6. Separation of Prefix Data and PTR Records

Some organization may want to separate the administration of prefix related data (geolocations, prefix ownership, and so forth) from the management of traditional PTR records. Note that all prefix related data is stored at a name that includes the "m" label. This "m" label could be used as delegation point to separate the administration of prefix data from the administration of PTR records.

To illustrate this, suppose the owner of 129.82.128/17 would like one to keep the management of prefix related data distinct from the management of their PTR records. Note that for all prefixes with a mask length between 17 and 23 are part of the zone 1.m.82.129.in-addr.arpa. This zone can simply be delegated to the group managing prefix related data while the group managing PTR records continues to be responsible for the zones 128.82.129.in-addr.arpa to 255.82.129.in-addr.arpa.

If prefix data is also to be stored at mask lengths ranging between 24 and 32, then m.128.82.129.in-addr.arpa to m.255.82.129.in-addr.arpa can also be delegated to the group managing prefix data. In this sense, an organization can keep a complete separation between groups managing prefix data and groups managing PTR records for host names.

During the discussion of the draft, some organizations expressed a desire to achieve this type of separation in operational practice. In particular, groups associated with routing and prefix management might manage the prefix related records while other groups associated with DHCP and IP address management currently manage the PTR records. This example simply illustrates these groups can be kept distinct if an organization so desires. As with any DNS deployment, an organization makes its own decisions on where to make zone cuts and how to manage their own delegation.

6.7. Prefix Enumeration

This document introduces a convention for naming IPv4 and IPv6 prefixes. It is not an enumeration technique. To illustrate the difference between lookup and enumeration we consider a hypothetical application that uses LOC resource records to associate geographic locations with prefixes. Note the use of the LOC record is simply to make the example concrete and the same argument applies to any type of data stored at a prefix.

An application can easily lookup the LOC resource record associated with a prefix using this naming convention. The application simply converts the prefix (IPv4 or IPv6) into a DNS name as described in the previous sections and queries for the LOC record associated with that name. Using DNSSEC, an application can also authenticate the LOC record or provide authenticated denial of existence proving that no such LOC record exists.

A distinct question is how one might enumerate all possible prefixes that have LOC records. Techniques to provide enumeration of prefixes in the DNS are outside the scope of this document.

6.8. Finding Longest Matches

Another distinct question is how one could find the longest match for a given IP address or prefix. For example, the application might want to find the most specific prefix (longest match) that has a LOC record and covers a particular IP address. Similar to enumeration, the naming convention does not directly provide longest match. Applications might develop strategies for searching all covering prefixes using variations of brute force searches, exploiting NSEC records (if used), using NXDOMAIN queries to find zone boundaries, or by adding additional record types to aid in finding related prefixes. [RFC1101], for example, uses an A record to specify a mask length for a contained subnet. It also shows how to chase such A record until the longest match is found. This scheme could be used with the naming convention proposed here as well. Nevertheless, these techniques are application-dependent. The naming convention proposed here in itself does not provide an explicit mechanism to find the longest matching prefix for an IP address.

The naming convention proposed here provides a way to name a prefix. Once one has this name, all the advantages (and disadvantages) of DNS apply. One can easily issue queries for the name and retrieve resource records associated with that name. For many applications, this is sufficient. Applications that require more complex prefix related functions, such as enumerating all prefixes of a given type or finding the longest prefix match, need to build this functionality into their application. The naming convention provides the necessary building blocks to achieve this, but does not dictate how a particular application will assemble the building blocks.

7. Security Considerations

This document only introduces a naming convention. Applications that make use of this naming convention may require the use of DNSSEC to validate the resource records stored at these names.

8. IANA Considerations

This document does not request any IANA action.

9. Acknowledgments

The authors would like to thank Danny McPherson (Verisign), Lixia Zhang (UCLA), and Kim Claffy (CAIDA) for their comments and suggestions. This document was aided via numerous discussions at NANOG, IETF and private meetings with ISPs, telecomm carriers, and research organizations too numerous to mention by name. Finally, the naming convention has been in use by some organizations for over a year at the time of this draft. Thanks to all for your comments and advice.

10. Change History

Changes from version 03 to 04

No changes were made to the naming convention, requirements, or document scope.

Minor changes to fix two typos and also to improve grammar throughout.

Clarified the description of Related Work based on feedback received.

Simplified the Additional Considerations based on feedback received.

No changes in the convention itself were added or removed.

Changes from version 02 to 03

Added detail regarding [RFC1101] and how it historically defined a method to name subnets; explained how CIDR introduced ambiguities into [RFC1101] creating the need for a more comprehensive naming convention.

Added similar explanatory material for [RFC2317].

Added "unambiguous" as a design objective.

Added definition of "nibble boundary".

Expanded and clarified the discussion of operational procedures required for maintaining the existing reverse DNS tree as subnets are delegated within or across octet boundaries.

Showed how largest enclosing prefix could be found using [RFC1101] A record semantics within the proposed naming structure.

Added clarification that the convention requires creating a name in which the sequence of binary digits does not extend past the next octet or nibble boundary.

Added Cathie Olschanowsky as a co-author.

Changes from version 01 to 02

Concerns were raised at the IETF 83 meeting that the document appeared too specific to the routing application. Several other

applications were mentioned. We clarified the introduction to show that the naming convention is application agnostic.

Expanded the related work discussion to include RFC 1101.

The "m" label is now added even when on an octet boundary.

Moved all other discussion into the Additional Considerations section; removing the alternate naming and replacing it with a discussion of existing delegations, adding a section on separating prefix and PTR records, added a section on enumerating prefixes and finding longest matches. All these changes reflect comments from the mailing list, IETF 83 discussions, and other comments. They do not change the naming scheme itself.

To emphasize the approach is application agnostic, the appendix examples were changed from using routing security records to LOC records. Any record type could be used, but LOC records were chosen as they were viewed as easy to understand.

Changes from version 00 to 01

Introduction added an additional subsection on aligning the DNS hierarchy with the IP address hierarchy.

Clarified step 1 of the naming algorithm on removing octets that are not significant.

Expanded and clarified the discussion of alternate name encoding for prefixes on an octet boundary.

Added Eric Osterweil as a co-author

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

11.2. Informative References

[I-D.bates-bgp4-nlri-orig-verif]
Bates, T., Bush, R., Li, T., and Y. Rekhter, "DNS-based NLRI origin AS verification in BGP", draft-bates-bgp4-nlri-orig-verif-00 (work in progress), January 1998.

[I-D.gersch-grow-revdns-bgp]
Gersch, J., Massey, D., Osterweil, E., and L. Zhang, "DNS Resource Records for BGP Routing Data", draft-gersch-grow-revdns-bgp-00 (work in progress), February 2012.

[I-D.howard-isp-ip6rdns]
Howard, L. and A. Durand, "Reverse DNS in IPv6 for Internet Service Providers", draft-howard-isp-ip6rdns-04 (work in progress), September 2010.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1101] Mockapetris, P., "DNS encoding of network names and other types", RFC 1101, April 1989.

[RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, March 1998.

Appendix A. Example Zone Files

A.1. Example 1

This example shows several DNS records added to an existing reverse-DNS zone file at octet boundary 129.82.0.0/16. The records show how LOC records could be specified in the zone file to be associated with an address block. Otherwise no other changes were made. This example has added records with LOC information pertinent to address blocks 129.82/16 and the four /18's at 129.82.0.0/18, 129.82.64.0/18, 129.82.128.0/18, and 129.82.192.0/18.

```
$TTL 3600
$ORIGIN 82.129.in-addr.arpa.

@    IN    SOA     rush.colostate.edu.  dnsadmin.colostate.edu. (
                        2012021300      ; serial number
                        900             ; refresh, 15 minutes
                        600             ; update retry, 10 minutes
                        86400           ; expiry, 1 day
                        3600            ; minimum, 1 hour
                       )

     IN    NS      dns1.colostate.edu.
     IN    NS      dns2.colostate.edu.

m                  IN   LOC   latitude/longitude info for the /16
; 129.82.0.0/16

0.0.m              IN   LOC   lat/long for North campus
; 129.82.0.0/18

1.0.m              IN   LOC   lat/long for South campus
; 129.82.64.0/18

0.1.m              IN   LOC   lat/long for Denver campus
; 129.82.128.0/18

1.1.m              IN   LOC   lat/long for Boulder campus
; 129.82.192.0/18


;  delegations required for 256 /24 zones which contain PTR records

1   IN  NS  dns1.colostate.edu.
    IN  NS  dns2.colostate.edu.
2   IN  NS  dns1.colostate.edu.
    IN  NS  dns2.colostate.edu.

;  continuation to 255 is left out for the sake of brevity
```

A.2. Example 2

This example illustrates the creation of a new zone for 216.17.128.0/17 which is not at an octet boundary. The existing 256 zones delegated at IN-ADDR.ARPA for the range 0.17.128 through 255.17.216.in-addr.arpa remain unchanged; they contain PTR records maintained by the appropriate zone owners.

In this example we have added several records all at the same domain name with information pertinent to address block 216.17.128.0/17.

Only a single new delegation needs to be added to IN-ADDR.ARPA:

```
1.m.17.216.in-addr.arpa  NS   ns.frii.net
```

This delegation refers to the new /17 zone and is not in conflict with any of the pre-existing /24 zones.

```
$TTL 3600
$ORIGIN 1.m.17.216.in-addr.arpa.

@    IN   SOA     ns1.frii.net.  hostmaster.frii.net. (
                       2012021300      ; serial number
                       14400           ; refresh, 4 hours
                       3600            ; update retry, 1 hour
                       604800          ; expiry, 7 days
                       600             ; minimum, 10 minutes
                      )

     IN   NS      ns1.frii.net.
     IN   NS      ns2.frii.net.

$ORIGIN 17.216.in-addr.arpa.

1.m              LOC       lat/long for main office
;216.17.128.0/17


; no other delegations or PTR records are needed in this zone file
```

Authors' Addresses

Joe Gersch
Secure64 SW Corp
Fort Collins, CO
US

Email: joe.gersch@secure64.com

Dan Massey
Colorado State University
Fort Collins, CO
US

Email: massey@cs.colostate.edu

Eric Osterweil
Verisign
Reston, VA
US

Email: eosterweil@verisign.com

Cathie Olschanowsky
Colorado State University
Fort Collins, CO
US

Email: cathie@cs.colostate.edu