Network Working Group   J. Peterson
Internet-Draft   NeuStar, Inc.
Intended status: Informational   O. Kolkman
Expires: January 17, 2013   NLnet Labs
                                                        H. Tschofenig
                                                Nokia Siemens Networks
                                                           B. Aboba
                                               Microsoft Corporation
                                                        July 16, 2012

Architectural Considerations on Application Features in the DNS
draft-iab-dns-applications-05

Abstract

   A number of Internet applications rely on the Domain Name System
   (DNS) to support their operations. Many applications use the DNS to
   locate services for a domain. For example, some applications
   transform identifiers other than domain names into formats that the
   DNS can process and then fetch application data or service location
   data from the DNS. Proposals to incorporate more sophisticated
   application behavior into the DNS, however, have raised questions
   about the applicability and extensibility of the DNS. This document
   explores the architectural consequences of using the DNS for storing
   and retrieving certain application features, and provides guidance to
   future application designers as to the sorts of ways that
   applications can make use of the DNS.

Table of Contents

1. Motivation

   The Domain Name System (DNS) has long provided a general means of
   translating easily-memorized domain names into numeric Internet
   Protocol addresses, which makes the Internet easier to use by
   providing a valuable layer of indirection between well-known names
   and lower-layer protocol elements. [RFC0974] documented a further
   use of the DNS: to manage application services operating in a domain
   with the Mail Exchange (MX) resource record, which helped email
   addressed to the domain to find a mail service for the domain
   sanctioned by the zone administrator.

   The seminal MX record served as a prototype for other DNS resource
   records that supported applications associated with a domain name.
   The SRV resource record [RFC2052] provided a more general mechanism
   for locating services in a domain, complete with a weighting system
   and selection among transports. The Naming Authority Pointer (NAPTR,
   originally [RFC2168]) resource record, especially as it evolved into
   the more general Dynamic Delegation Discovery System (DDDS, [RFC3401]
   passim) framework, added a generic mechanism for storing application
   data in the DNS - an algorithm for transforming a string into a
   domain name, which might then be resolved by the DNS to find NAPTR
   records. This enabled the resolution of identifiers that do not have
   traditional host components through the DNS; the best-known example
   of this are telephone numbers, as resolved by the DDDS application
   ENUM. Recent work such as Domain Keys Identified Mail (DKIM,
   [RFC4871]) has enabled security features of applications to be
   advertised through the DNS, via the TXT resource record.

   The amount of application intelligence that uses the DNS has thus
   increased over time. However, some proposed usages of, and
   extensions to, the DNS have become misaligned with both the DNS
   architecture and the DNS protocol. An example of such misalignment
   is illustrated through the expectation of confidentiality: In the
   global public DNS the resolution of domain names to IP addresses is
   public information with no expectation of confidentiality, and thus
   the underlying query/response protocol has no authentication or
   encryption mechanism - typically, any security required by an
   application or service is invoked after the DNS query, when the
   resolved service has been contacted. Only in private DNS
   environments (including split horizon DNS) where the identity of the
   querier is assured through some external means does the DNS maintain
   confidential records in this sense by providing answers to the
   private and public users of the DNS.

   Another type of differentiation in answers is done for load balancing
   reasons, localization or related optimizations, whereby the public
   DNS returns different addresses in response to queries from different

sources, or even no response at all, which is discussed further in
Section 3.1.1.

Applications in many environments require features such as
confidentiality, and as the contexts in which applications rely on
the DNS have increased, some application protocols have looked to
extend the DNS to include these sorts of capabilities.

This document therefore provides guidance to designers of
applications and application protocols on the use or extension of the
DNS to provide features to applications. It offers an overview of
ways that applications have used the DNS in the past, as well as
proposed ways that applications could use the DNS. It identifies
concerns and trade-offs and gives some reasons to decide the
question, "Should I store this information in the DNS, or use some
other means?" when that question arises during protocol development.
These guidelines remind application protocol designers of the
strengths and weaknesses of the DNS in order to make it easier for
designers to decide what features the DNS should provide for their
application.

The guidance in this document complements the guidance on extending
the DNS given in [RFC5507]. Whereas [RFC5507] considers the
preferred ways to add new information to the underlying syntax of the
DNS (such as defining new resource records or adding prefixes or
suffixes to labels), the current document considers broader
implications of applications that rely on the DNS for the
implementation of certain features, be it through extending the DNS
or simply reusing existing protocol capabilities - implications that
may concern the invocation of the resolver by applications, the
behavior of name servers, resolvers or caches, extensions to the
underlying DNS protocol, the operational responsibilities of zone
administrators, security, or the overall architecture of names. When
existing DNS protocol fields are used in ways that their designers
did not intend to handle new applications, those applications may
require further changes and extensions that are fundamentally at odds
with the strengths of the DNS.

2. Overview of DNS Application Usages

   [RFC0882] identifies the original and fundamental connection between
   the DNS and applications. It begins by describing how the
   interdomain scope of applications creates "formidable problems when
   we wish to create consistent methods for referencing particular
   resources that are similar but scattered throughout the environment."
   This motivated "the need to have a mapping between host names... and
   ARPA Internet addresses" transition from a global table (the original
   "hosts" file) to a "distributed database that performs the same
   function." [RFC0882] also envisioned some ways to find the resources
   associated with mailboxes in a domain: without these extensions, a
   user trying to send mail to a foreign domain lacked a discovery
   mechanism to locate the right host in the remote domain to which to
   connect.

   While a special-purpose discovery mechanism could be built by each
   such application protocol that needed this functionality, the
   universality of the DNS invites installing these features into its
   public tree. Over time, several other applications leveraged
   resource records for locating services in a domain or for storing
   application data associated with a domain in the DNS. This section
   gives examples of various types of DNS usage by applications to date.

2.1. Locating Services in a Domain

   The MX resource record provides the simplest motivating example for
   an application advertising its resources in the Domain Name System.
   The MX resource record contains the domain name of a server that
   receives mail on behalf of the administrative domain in question;
   that domain name must itself be resolved to one or more IP addresses
   through the DNS in order to reach the mail server. While naming
   conventions for applications might serve a similar purpose (a host
   might be named "mail.example.com" for example), approaching service
   location through the creation of a new resource record yields
   important benefits. For example, one can put multiple MX records
   under the same name, in order to designate backup resources or to
   load balance across several such servers (see [RFC1794]); these
   properties could not easily be captured by naming conventions (see
   [RFC4367]).

   While the MX record represents a substantial improvement over naming
   conventions as a means of service location, it remains specific to a
   single application. Thus, the general approach of the MX record was
   adapted to fit a broader class of application through the Service
   (SRV) resource record (originally [RFC2052]). The SRV record allows
   DNS resolvers to query for particular services and underlying
   transports (for example, HTTP running over TLS, see [RFC2818]) and to

learn a host name and port where that service resides in a given
domain. It also provides a weighting mechanism to allow load
balancing across several instances of a service.

The reliance of applications on the existence of MX and SRV records
has important implications for the way that applications manage
identifiers, and the way that applications pass domain names to
resolvers. Email identifiers of the form "user@domain" rely on MX
records to provide the convenience of simply specifying a "domain"
component rather than requiring an application to guess which
particular host handles mail on behalf of the domain. While for
applications like web browsing, naming conventions continue to abound
("www.example.com"), SRV records allow applications to query for an
application-specific protocol and transport. For HTTP, the SRV
service name corresponds to the URL scheme of the identifier invoked
by the application (e.g., when "http://example.com" is the
identifier, the SRV query is for "_http._tcp.example.com"); for other
applications, the SRV service that the application passes to the
resolver may be implicit in the identifier. The identifier used at
the application layer thus designates the desired protocol and
domain, but the application uses the DNS to find the location of the
host of that service for the domain, the port where the service
resides on that host, additional locations or ports for load
balancing and fault tolerance, and related application features.
Ultimately, applications that acquire MX or SRV records may use them
as intermediate transformations in order to arrive at an eventual
domain name that will resolve to the IP addresses to contact for the
service.

Locating specific services for a domain is thus the first major
function for which applications started using the DNS in addition to
simple name resolution. SRV broadened and generalized the precedent
of MX to make service location available to any application, rather
than just to mail. As the domain name of the located service might
require a second DNS query to resolve, [RFC1034] shows that the
Additional Data section of DNS responses may contain the
corresponding address records for the names of services designated by
the MX record; this optimization, which requires support in the
authoritative server and the resolver, is an initial example of how
support for application features requires changes to DNS operation.
At the same time this is an example of an extension of the DNS that
cannot be relied on: Many DNS resolver implementations will remove
the addresses in the additional section of the DNS answers because of
the trustworthiness issues described in [RFC2181].

2.2. NAPTR and DDDS

The NAPTR resource record evolved to fulfill a need in the transition
from Uniform Resource Locators (URLs) to the more mature URI
[RFC3986] framework, which incorporated Uniform Resource Names
(URNs). Unlike URLs, URNs typically do not convey enough semantics
internally to resolve them through the DNS, and consequently a
separate URI-transformation mechanism is required to convert these
types of URIs into domain names. This allows identifiers with no
recognizable domain component to be treated as DNS names for the
purpose of name resolution. Once these transformations result in a
domain name, applications can retrieve NAPTR records under that name
in the DNS. NAPTR records contain a far more rich and complex
structure than MX or SRV resource records. A NAPTR record contains
two different weighting mechanisms ("order" and "preference"), a
"service" field to designate the application that the NAPTR record
describes, and then two fields that can contain translations: a
"replacement" or "regular expression" field, only one of which
appears in given NAPTR record. A "replacement," like NAPTR's
ancestor the PTR record, simply designates another domain name where
one would look for records associated with this service in the
domain. The "regexp," on the other hand, allows regular expression
transformations on the original URI intended to transform it into an
identifier that the DNS can resolve.

As the Abstract of [RFC2915] says, "This allows the DNS to be used to
lookup services for a wide variety of resource names (including URIs)
which are not in domain name syntax." Any sort of hierarchical
identifier can potentially be encoded as a DNS name, and thus
historically the DNS has often been used to resolve identifiers that
were never devised as a name for an Internet host. A prominent early
example is found in the in-addr domain [RFC0882], in which IPv4
addresses are encoded as domain names by applying a string
preparation algorithm that required reversing the octets and treating
each individual octet as a label in a DNS name - thus, for example,
the address 192.0.2.1 became 1.2.0.192.in-addr.arpa. This allowed
resolvers to query the DNS to learn name(s) associated with an IPv4
address. The same mechanism has been applied to IPv6 addresses
[RFC3596] and other sorts of identifiers that lack a domain
component. Eventually, this idea connected with activities to create
a system for resolving telephone numbers on the Internet, which
became known as ENUM (originally [RFC2916]). ENUM borrowed from an
earlier proposal, the "tpc.int" domain [RFC1530], which provided a
means for encoding telephone numbers as domain names by applying a
string preparation algorithm that required reversing the digits and
treating each individual digit as a label in a DNS name - thus, for
example, the number +15714345400 became
0.0.4.5.4.3.4.1.7.5.1.tpc.int.

In the ENUM system, in place of "tpc.int" the special domain
"e164.arpa" was reserved for use. In its more mature form in the
Dynamic Delegation and Discovery Service (DDDS) ([RFC3401] passim)
framework, this initial transformation of the telephone number to a
domain name was called the "First Well Known Rule." The capability
to create these rules is thus a generalization of the string
preparation algorithm done for IP addresses. Its flexibility has
inspired a number of proposals beyond ENUM to encode and resolve
unorthodox identifiers in the DNS. Provided that the identifiers
transformed by the First Well Known Rule have some meaningful
structure and are not overly lengthy, virtually anything can serve as
an input for the DDDS structure: for example, civic addresses.
Though [RFC3402] stipulates of the identifier that "the lexical
structure of this string must imply a unique delegation path," there
is no requirement that the identifier be hierarchical, nor that the
points of delegation in the domain name created by the First Well
Known Rule correspond to any points of administrative delegation
implied by the structure of the identifier.

While this ability to look up names "which are not in the domain name
syntax" does not change the underlying DNS protocols - the names
generated by the DDDS algorithm are still just domain names - it does
change the context in which applications pass name to resolvers, and
can potentially require very different operational practices of zone
administrators(see Section 3.3). In terms of the results of a DNS
query, the presence of the "regexp" field of NAPTR records enabled
unprecedented flexibility in the types of identifiers that
applications could resolve with the DNS. Since the output of the
regular expression frequently took the form of a URI (in ENUM
resolution, for example, a telephone number might be converted into a
SIP URI [RFC3261]), anything that could be encoded as a URI might be
the result of resolving a NAPTR record - which, as the next section
explores, essentially means arbitrary data.

2.3. Arbitrary Data in the DNS

Since URI encoding has ways of carrying basically arbitrary data,
resolving a NAPTR record might result in an output other than an
identifier that would subsequently be resolved to IP addresses and
contacted for a particular application - it could give a literal
result consumed by the application. Originally, in [RFC2168], the
intended applicability of the regular expression field in NAPTR was
much more narrow: the regexp field contained a "substitution
expression that is applied to the original URI in order to construct
the next domain name to lookup," in order to "change the host that is
contacted to resolve a URI" or as a way of "changing the path or host
once the URL has been assigned." The regular expression tools
available to NAPTR record authors, however, grant much broader powers

   to alter the input string, and thus applications began to rely on
   NAPTR to perform more radical transformations. By [RFC3402], the
   output of DDDS is wholly application-specific: "the Application must
   determine what the expected output of the Terminal Rule should be,"
   and the example given in the document is one of identifying
   automobile parts by inputting a part number and receiving at the end
   of the process information about the manufacturer (though this
   example reflects that the DNS is only one database to which the DDDS
   framework might apply).

   NAPTR did not however pioneer the storage of arbitrary data in the
   DNS. At the start, [RFC0882] observed that "it is unlikely that all
   users of domain names will be able to agree on the set of resources
   or resource information that names will be used to retrieve," and
   consequently places little restriction on the information that DNS
   records might carry: it might be "host addresses, mailbox data, and
   other as yet undetermined information." [RFC1035] defined the TXT
   record, a means to store arbitrary strings in the DNS; [RFC1034]
   specifically stipulates that a TXT contains "descriptive text" and
   that "the semantics of the text depends on the domain where it is
   found." The existence of TXT records has long provided new
   applications with a rapid way of storing data associated with a
   domain name in the DNS, as adding data in this fashion requires no
   registration process. [RFC1464] experimented with a means of
   incorporating name/value pairs to the TXT record structure, which
   allowed applications to differentiate different chunks of data stored
   in a TXT record. In this fashion, an application that wants to store
   additional data in the DNS can do so without registering a new
   resource record type - though [RFC5507] points out that it is
   "difficult to reliably distinguish one application's record from
   otters, and for its parser to avoid problems when it encounters other
   TXT records."

   While open policies surrounding the use of the TXT record have
   resulted in a checkered past for standardizing application usage of
   TXT, TXT has been used as a technical solution for DKIM [RFC4871] to
   store necessary information about the security of email in DNS,
   though within a narrow naming convention (records stored under
   "_domainkey" subdomains). Storing keys in the DNS became the
   preferred solution for DKIM for several reasons: notably, because the
   public keys associated with email required wide public distribution,
   and because email identifiers contain a domain component that
   applications can easily use to consult the DNS. If the application
   had to negotiate support for the DKIM mechanism with mail servers, it
   would give rise to bid-down attacks (where attackers misrepresent
   that DKIM is unsupported in the domain) that are not possible if the
   DNS delivers the keys (provided that DNSSEC [RFC4033] guarantees
   authenticity of the data). However, there are potential issues with

storing large data in the DNS, as discussed in Section 3.2.1 as well
as with the DKIM namespace conventions that prevent the use of DNS
wildcards (as discussed in section 3.6.2.1 of [RFC4871] and in more
general terms in [RFC5507]).

[3](#). Challenges for the DNS

   The methods discussed in the previous section for transforming
   arbitrary identifiers into domain names, and for returning arbitrary
   data in response to DNS queries, both represent significant
   departures from the basic function of translating host names to IP
   addresses, yet neither fundamentally alters the underlying semantics
   of the DNS. When we consider, however, that the URIs returned by
   DDDS might be base 64 encoded binary data in the data URL [RFC2397],
   the DNS could effectively implement the entire application feature
   set of any simple query-response protocol. Effectively, the DDDS
   framework considers the DNS a generic database - indeed, the DDDS
   framework was designed to work with any sort of underlying database;
   as [RFC3403] shows, the DNS is only one potential database for DDDS
   to use. Whether the DNS as an underlying database can support the
   features that some applications of DDDS require, however, is a more
   complicated question.

   As the following subsections will show, the potential that
   applications might rely on the DNS as a generic database gives rise
   to additional requirements that one might expect to find in a
   database access protocol: authentication of the source of queries for
   comparison to access control lists, formulating complex relational
   queries, and asking questions about the structure of the database
   itself. DNS was not designed to provide these sorts of properties,
   and extending the DNS to encompass them would represent a fundamental
   alteration to its model. Ultimately, this document concludes that
   efforts to retrofit these capabilities into the DNS would be better
   invested in selecting, or if necessary inventing, other Internet
   services with broader powers than the DNS. If an application
   protocol designer wants these properties from a database, in general
   this is a good indication that the DNS cannot, or can only partly,
   meet the needs of the application in question.

   Since many of these new requirements have emerged from the ENUM
   space, the following sections use ENUM as an illustrative example;
   however, any application using the DNS as a feature-rich database
   could easily end up with similar requirements.

[3.1](#). Compound Queries

   Traditionally, DNS information is uniquely identified by the domain
   name, resource record type, and class. DNS queries are based on this
   3-tuple and the replies are resource record sets that are to be
   treated as atomic data elements (see [RFC2181]); to applications the
   behavior of the DNS has traditionally been that of an exact-match
   query response lookup mechanism. Outside of the DNS space, however,
   there are plenty of query-response applications that require a

compound or relational search, which takes into account more than one
factor in formulating a response or uses no single factor as a key to
the database. For example, in the telephony space, telephone call
routing often takes into account numerous factors aside from the
dialed number, including originating trunk groups, interexchange
carrier selection, number portability data, time of day, and so on.
All are considered simultaneously in generating a route. While in
its original conception, ENUM hoped to circumvent the traditional
PSTN and route directly to Internet-enabled devices, the
infrastructure ENUM effort to support the migration of traditional
carrier routing functions to the Internet aspires to achieve feature
parity with traditional number routing. However, [RFC3402]
explicitly states that "it is an assumption of the DDDS that the
lexical element used to make a delegation decision is simple enough
to be contained within the Application Unique String itself. The
DDDS does not solve the case where a delegation decision is made
using knowledge contained outside the AUS and the Rule (time of day,
financial transactions, rights management, etc.)." Consequently,
some consideration has been given to ways to add additional data to
ENUM queries to give the DNS server sufficient information to return
a suitable URI (see Section 3.1.1).

From a sheer syntactical perspective, however, domain names do not
admit of this sort of rich structure. Several workarounds have
attempted to instantiate these sorts of features in DNS queries. For
example, the domain name itself could be compounded with the
additional parameters: one could take a name like
0.0.4.5.4.3.4.1.7.5.1.e164.arpa and append a trunk group identifier
to it, for example, of the form
tg011.0.0.4.5.4.3.4.1.7.5.1.e164.arpa. While in this particular case
a DNS server can adhere to its traditional behavior in locating
resource records, the syntactical viability of encoding additional
parameters in this fashion is dubious, especially if more than one
additional parameter is required and the presence of parameters is
optional so that the application needs multiple queries to assess the
completeness of the information it needs to perform its function.

More commonly, proposals piggyback additional query parameters as
EDNS0 extensions (see [RFC2671]). This might be problematic for
three reasons. First, supporting EDNS0 extensions requires
significant changes to name server behavior that need to be supported
by the authoritative and recursive nameservers on which the
application relies and might be very hard to realize on a global
scale. In addition, the intended applicability of the EDNS0
mechanism, as [RFC2761] states, is to "a particular transport level
message and not to any actual DNS data," and consequently the OPT
Resource Records it specifies are never to be forwarded; the use of
EDNS0 for compound queries, however, clearly is intended to

discriminate actual DNS data rather than to facilitate transport-
layer handling. Finally, [RFC2761] also specifies that OPT Resource
Records are never cached (see the next paragraph). For these
reasons, this memo recommends against the use of EDNS0 as a mechanism
for crafting compound DNS queries.

The implications of these sorts of compound queries for recursion and
caching are potentially serious. The logic used by the authoritative
server to respond to a compound query may not be understood by any
recursive servers or caches; intermediaries that naively assume that
the response was selected based on the domain name, type and class
alone might serve responses to queries in a different way than the
authoritative server intends.

3.1.1. Responses Tailored to the Originator

The most important subcase of the compound queries are DNS responses
tailored to the identity of their originator, where some sort of
administrative identity of the originator must be conveyed to the
DNS. We must first distinguish this from cases where the originating
IP address or a similar indication is used to serve a location-
specific name. For those sorts of applications, which generally lack
security implications, relying on factors like the source IP address
introduces little harm; for example, when providing a web portal
customized to the region of the client, it would not be a security
breach if the client saw the localized portal of the wrong country.
Because recursive resolvers may obscure the origination network of
the DNS client, a recent proposal suggested introducing a new DNS
query parameter to be populated by DNS recursive resolvers in order
to preserve the originating IP address (see
[I-D.vandergaast-edns-client-ip]). However, aside from purely
cosmetic uses, these approaches have known limitations due to the
prevalence of private IP addresses, VPNs and so on which obscure the
source IP address and instead supply the IP address of an
intermediary that may be very distant from the originating endpoint.
Implementing technology such as the one described by
[I-D.vandergaast-edns-client-ip] does require significant changes in
the operation of recursive resolvers and the authoritative servers
that would rely on the original source IP address to select Resource
Records, and moreover a fundamental change to caching behavior as
well. As a result such technology can only be deployed after
bilateral (authoritative-resolver) agreement and is not likely to
deploy to ubiquitous Internet scale use.

In other deployments in use today, including those based on the BIND
"views" feature, the source IP address is used to grant access to a
private set of resource records. The security implications of
trusting the source IP address of a DNS query have prevented most

solutions along these lines from being standardized (see [RFC6269]),
though the practice remains widespread in "split horizon" private DNS
deployment (see Section 4) which typically rely on an underlying
security layer, such as a physical network or an IPsec VPN, to
prevent spoofing of the source IP address. These deployments do have
a confidentiality requirement to prevent information intended for a
constrained audience (internal to an enterprise, for example) from
leaking to the public Internet -- while these internal network
resources may use private IP addresses which would not be useful on
the public Internet anyway, in some cases this leakage would reveal
topology or other information that the name server provisioner hopes
to keep private.

These first two cases, regardless of their acceptance by the
standards community, have widespread acceptance in the field. Some
applications, however, go even further and propose extending the DNS
to add an application-layer identifier of the originator rather than
an IP address; for example,
[I-D.kaplan-dnsext-enum-sip-source-ref-opt-code] provides a SIP URI
in an EDNS0 parameter. Effectively, the conveyance of application-
layer information about the administrative identity of the originator
through the DNS is a weak authentication mechanism, on the basis of
which the DNS server makes an authorization decision before sharing
resource records. This can parlay into a per-Resource Record
confidentiality mechanism, where only a specific set of originators
are permitted to see resource records, or a case where a query for
the same name by different entities results in completely different
resource record sets. The DNS, however, substantially lacks the
protocol semantics to manage access control lists for data, and
again, caching, forwarding, and recursion introduce significant
challenges for applications that attempt to offload this
responsibility to the DNS. Achieving feature parity with even the
simplest authentication mechanisms available at the application layer
would like require significant rearchitecture of the DNS.

3.2. Using DNS as a Generic Database

As previously noted, applications can use the DNS by transferring an
arbitrary string into a query (using a method like the First Well
Known Rule of DDDS) and receiving arbitrary data stored in custom
resource records (or potentially TXT RRs). Some query-response
applications, however, require queries and responses that simply fall
outside the syntactic capabilities of the DNS. Domain names
themselves must conform to certain syntactic constraints: they must
consist of labels that do not exceed 63 octets while the total length
of the encoded name may not exceed 255 octets, they must obey fairly
strict encoding rules (some characters are not legal in labels), and
so on. The DNS therefore cannot be a completely generic database.

Similar concerns apply to the size of DNS responses.

3.2.1. Large Data in the DNS

While the data URL specification [RFC2397] notes that it is "only
useful for short values," unfortunately it gives no particular
guidance about what "short" might mean. Many applications today use
quite large data URLs (containing a megabyte or more of data) as
workarounds in environments where only URIs can syntactically appear.
The meaning of "short" in an application context is probably very
different from how we should understand it in a DNS message.
Referring to a typical public DNS deployment, [RFC5507] observes that
"there's a strong incentive to keep DNS messages short enough to fit
in a UDP datagram, preferably a UDP datagram short enough not to
require IP fragmentation". And while EDNS0 allows for mechanisms to
negotiate DNS message sizes larger than the traditional 512 octets
there is a high risk that a long payload will cause UDP
fragmentation, in particular when the DNS message already carries
DNSSEC information. If EDNS0 is not available, or the negotiated
EDNS0 packet size is too small to fit the data, or UDP fragments are
dropped, the DNS will (eventually) resort to use TCP. While TCP
allows DNS responses to be quite long, this requires stateful
operation of servers which can be costly in deployments where servers
have only fleeting connections to many clients. Ultimately, there
are forms of data that an application might store in the DNS that
exceed reasonable limits: in the ENUM context, for example, something
like storing base 64 encoded mp3 files of custom ringtones.

Designs relying on storage of large amounts of data within DNS RRs
furthermore need to minimize the potential damage achievable in a
reflection attack (see [RFC4732], Section 3), in which the attacker
sends DNS queries with a forged source address, and the victim
receives the response. By generating a large response to a small
query, the attacker can magnify the bandwidth directed at the victim.
Where the responder supports EDNS0, an attacker may set the requester
maximum payload size to a larger value while querying for a large
Resource Record, such as a certificate [RFC4398]. Thus the
combination of large data stored in DNS RRs and responders supporting
large payload sizes has the potential to increase the potential
damage achievable in a reflection attack.

Since a reflection attack can be launched from any network that does
not implement source address validation, these attacks are difficult
to eliminate absent the ubiquitous deployment of source address
validation.

The bandwidth that can be mustered in a reflection attack directed by
a botnet reflecting off a recursive nameserver on a high bandwidth

network is sobering. For example, if the responding resolver could
be directed to generate a 10KB response in reply to a 50 octet query,
then magnification of 200:1 would be attainable. This would enable a
botnet controlling 10000 hosts with 1 Mbps of bandwidth to focus 2000
Gbps of traffic on the victim, more than sufficient to congest any
site on today's Internet.

Since it is prohibitively difficult to complete a TCP three-way
handshake begun from a forged source address, DNS reflection attacks
utilize UDP queries. Unless the attacker uses EDNS0 [RFC2671] to
enlarge the requester's maximum payload size, a response can only
reach 576 octets before the truncate bit is set in the response.
This limits the maximum magnification achievable from a DNS query
that does not utilize EDNS0. As the large disparity between the size
of a query and size of the response creates this amplification,
implementations should limit EDNS0 responses to a specific ratio
compared to the request size. The precise ratio can be configured on
a per-deployment basis (taking into account DNSSEC response sizes),
but without this limitation, EDNS0 can cause significant harm.

In summary, there are two operational forces that tend to drive the
practically available EDNS0 sizes down: possible UDP fragmentation
and minimizing magnification in case of reflection attacks. DNSSEC
data will use a significant fraction of the available space in a DNS
packet. Therefore -- appreciating that given the current DNSSEC and
EDNS0 deployment experience precise numbers are impossible to give --
the generic payload available to other DNS data, given the premise
that TCP fallback is to be minimized, is likely to be closer to of
several hundreds than a few thousand octets.

3.3. Administrative Structures Misaligned with the DNS

While the DDDS framework enables any sort of alphanumeric data to
serve as a DNS name through the application of the First Well Known
Rule, the delegative structure of the resulting DNS name may not
reflect the division of responsibilities for the resources that the
alphanumeric data indicates. While [RFC3402] requires only that
"Application Unique String has some kind of regular, lexical
structure that the rules can be applied to," DDDS is first and
foremost a delegation system: its abstract stipulates that "Well-
formed transformation rules will reflect the delegation of management
of information associated with the string." Telephone numbers in the
United States, for example, are assigned and delegated in a
relatively complex manner. Historically, the first six digits of a
nationally specific number (called the "NPA/NXX") reflected a point
of administrative delegation from the number assignment agency to a
carrier; from these blocks of ten thousand numbers, the carrier would
in turn delegate individual assignments of the last four digits (the

"XXXX" portion) to particular customers. However, after the rise of
North American telephone number portability in the 1990s, the first
point of delegation went away: the delegation is effectively from the
number authority to the carrier for each complete ten-digit number
(NPA/NXX-XXXX). While technical implementation details differ from
nation to nation, number portability systems with similar
administrative delegations now exist worldwide.

To render these identifiers as domain names in accordance with the
DDDS Rule for ENUM yields a large flat administrative domain with no
points of administrative delegation from the country-code
administrator, e.g. 1.e164.arpa, down to the ultimate assignee of a
number. Under the assumption that one administrative domain is
maintained within one DNS zone containing potentially over five
billion names, scalability difficulties manifest in a number of
areas: The scalability that results from caching depends on points of
delegation so that cached results for intermediate servers take the
load off higher tier servers. If there are no such delegations, then
as in the telephone number example the zone apex server must bear the
entire load for queries. Worse still, number portability also
introduces far more dynamism in number assignment, where in some
regions updated assignees for ported numbers must propagate within
fifteen minutes of a change in administration. Jointly, these two
problems make caching the zone extremely problematic. Moreover,
traditional tools for DNS replication, such as the zone transfer
protocol AXFR [RFC1034] and IXFR [RFC1995], might not scale to
accommodate zones with these dimensions and properties.

In practice, the maximum sizes of telephone number administrative
domains are closer to 300M (the current amount of allocated telephone
numbers in the United States today - still more than three times the
number of .com domain names) and one can alleviate some of the
scalability issues mentioned above by artificially dividing the
administrative domain into a hierarchy of DNS zones. Still, the fact
that traditional DNS management tools no longer apply to the
structures that an application tries to provision in the DNS, is clue
that the DNS might not be the right place for an application to store
its data.

DDDS provides a capability that transforms arbitrary strings into
domain names, but those names play well with the DNS only when the
input string have an administrative structure that maps on DNS
delegations. In the first place, delegation implies some sort of
hierarchical structure. Any mechanism to map a hierarchical
identifier into a DNS name should be constructed such that the
resulting DNS name does match the natural hierarchy of the original
identifier. Although telephone numbers, even in North America, have
some hierarchical qualities (like the geographical areas

corresponding to their first three digits), after the implementation
of number portability these points no longer mapped onto an
administrative delegation. If the input string to the DDDS does not
have a hierarchical structure representing administrative delegations
that can map onto the DNS distribution system, then that string
probably is not a good candidate for translating into a domain name.

While DDDS is the most obvious example of these concerns, the point
is more generic: were address portability to be implemented for IP
addresses, and their administration thus to become non-hierarchical,
the same concerns would apply to in-addr.arpa names. The difficulty
of mapping the DNS to administrative structures can even occur with
traditional domain names, where applications expect clients to infer
or locate zone cuts. In the web context, for example, it can be
difficult for applications to determine whether two domains represent
the same "site" when comparing security credentials with URLs (see
Section 3.4 below for more on this).

3.3.1. Metadata about Tree Structure

There are also other ways in which the delegative structure of an
identifier may not map well onto the DNS. Traditionally, DNS
resolvers assume that when they receive a domain name from an
application, that the name is complete - that it is not a fragment of
a DNS name that a user is still in the middle of typing. For some
communications systems, however, this assumption does not apply.
ENUM use cases have surfaced a couple of optimization requirements to
reduce unnecessary calls and queries; proposed solutions include
metadata in the DNS that describes the contents and structure of ENUM
DNS trees to help applications handle incomplete queries or queries
for domains not in use. In particular, the "send-n" proposal
[I-D.bellis-enum-send-n] hopes to reduce the number of DNS queries
sent in regions with variable-length numbering plans. When a dialed
number potentially has a variable length, a telephone switch
ordinarily cannot anticipate when a dialed number is complete, as
only the numbering plan administrator (who may be a national
regulator, or even in open number plans a private branch exchange)
knows how long a telephone number needs to be. Consequently, a
switch trying to resolve such a number through a system like ENUM
might send a query for a telephone number that has only partially
been dialed in order to test its completeness. The "send-n" proposal
installs in the DNS a hint informing the telephone switch of the
minimum number of digits that must be collected by placing in zones
corresponding to incomplete telephone numbers some Resource Records
which state how many more digits are required - effectively how many
steps down the DNS tree one must take before querying the DNS again.
Unsurprisingly, those boundaries reflect points of administrative
delegation, where the parent in a number plan yields authority to a

child. With this information, the application is not required to
query the DNS every time a new digit is dialed, but can wait to
collect sufficient digits to receive a response. As an optimization,
this practice thus saves the resources of the DNS server - though the
call cannot complete until all digits are collected, so at best this
mechanism only reduces the time the system will wait before sending
an ENUM query after collecting the final dialed digit. A
tangentially related proposal, [I-D.ietf-enum-unused], similarly
places resource records in the DNS that tell the application that it
need not attempt to reach a number on the telephone network, as the
number is unassigned - a comparable general DNS mechanism would
identify, for a domain name with no records available in the DNS,
whether or not the domain had been allocated by a registry to a
registrant.

Both proposals optimize application behavior by placing metadata in
the DNS that predicts the success of future queries or application
invocation by identifying points of administrative delegation or
assignment in the tree. In some respects, marking a point in the
tree where a zone begins or end has some features in common with the
traditional parent zone use of the NS record type, except instead of
pointing to a child zone, these metadata proposals point to distant
grandchildren. While this does not change resolver behavior as such
(instead, it changes the way that applications invoke the resolver),
it does have implications for the practices for zone administrators.
Metadata in one administrative domain would need to remain
synchronized with the state of the resources it predicts in another
administrative domain in the DNS namespace, in a case like overlap
dialing where the carrier delegates to a zone controlled by an
enterprise. When dealing with external resources associated with
other namespaces, like number assignments in the PSTN or the
databases of a registry operator, other synchronization requirements
arise; maintaining that synchronization requires that the DNS have
semi-real time updates that may conflict with scale and caching
mechanisms of the DNS.

It may also raise questions about the authority and delegation model.
Who gets to supply records for unassigned names? While in the
original but little-used e164.arpa root of ENUM, this would almost
certainly be a numbering plan administrator, it is far less clear who
that would be in the more common and successful "infrastructure" ENUM
models (see Section 4). Ultimately, these metadata proposals share
some qualities with DNS redirection services offered by ISPs (for
example, [I-D.livingood-dns-redirect]) which "help" web users who try
to browse to sites that do not exist - as proposals like [I-D.ietf-
enum-used] also create DNS records for unallocated zones that
redirect to a service provider's web page - though in the
[I-D.livingood-dns-direct] cases, at least the existence or non-

existence of a domain name is a fact about the Internet namespace,
rather than about an external namespace like the telephone's e.164
namespace which must be synchronized with the DNS tree in the
metadata proposals. In send-n, different leaf zones, who administer
telephone numbers of different lengths, can only provision their
hints at their own apex, which provides an imperfect optimization:
they cannot install it themselves in a parent, both because they lack
the authority and because different zones would want to provision
contradictory data. The later the hint appears in the tree, however,
the less optimization will result. An application protocol designer
who wants to manage identifiers whose administrative model does not
map well onto the DNS namespace and delegations structure would be
better served to implement such features outside the DNS.

[3.4](). **Domain Redirection**

Most Internet application services provide a redirection feature -
when you attempt to contact a service, the service may refer you to a
different service instance, potentially in another domain, that is
for whatever reason better suited to address a request. In HTTP and
SIP, for example, this feature is implemented by the 300 class
responses containing one or more better URIs that may indicate that a
resource has moved temporarily or permanently to another service.
Several tools in the DNS, including the SRV record, can provide a
similar feature at a DNS level, and consequently some applications as
an optimization offload the responsibility for redirection to the
DNS; NAPTR can also provide this capability on a per-application
basis, and numerous DNS resource records can provide redirection on a
per-domain basis. This can prevent the unnecessary expenditure of
application resources on a function that could be performed as a
component of a DNS lookup that is already a prerequisite for
contacting the service. Consequently, in some deployment
architectures this DNS-layer redirection is used for virtual hosting
services.

Implementing domain redirection in the DNS, however, has important
consequences for application security. In the absence of universal
DNSSEC, applications must blindly trust that their request has not
been hijacked at the DNS layer and redirected to a potentially
malicious domain, unless some subsequent application mechanism can
provide the necessary assurance. By way of contrast, application-
layer redirections protocols like HTTP and SIP have available
security mechanisms such as TLS that can use certificates to vouch
that a 300 response came from the domain that the originator
initially hoped to contact.

A number of applications have attempted to provide an after-the-fact
security mechanism that verifies the authority of a DNS delegation in

the absence of DNSSEC. The specification for dereferencing SIP URIs
([RFC3263], reaffirmed in [RFC5922]), requires that during TLS
establishment, the site eventually reached by a SIP request present a
certificate corresponding to the original URI expected by the user
(in other words, if example.com redirects to example.net in the DNS,
this mechanism expects that example.net will supply a certificate for
example.com in TLS, per the HTTP precedent in [RFC2818]), which
requires a virtual hosting service to possess a certificate
corresponding to the hosted domain. This restriction rules out many
styles of hosting deployments common in the web world today, however.
[I-D.barnes-hard-problem] explores this problem space, and
[I-D.saintandre-tls-server-id-check] proposes a solution for all
applications that use TLS. Potentially, new types of certificates
(similar to [RFC4985]) might bridge this gap, but support for those
certificates would require changes to existing certificate authority
practices as well as application behavior.

All of these application-layer measures attempt to mirror the
delegation of administrative authority in the DNS, when the DNS
itself serves as the ultimate authority on how domains are delegated
(Note: changing the technical delegation structure by changing NS
records in the DNS is not the same as administrative delegation e.g.
when a domain changes ownership). Synchronizing a static instrument
like a certificate with a delegation in the DNS, however, is
problematic because delegations are not static: revoking and
reissuing a certificate every time an administrative delegation
changes is cumbersome operationally. In environments where DNSSEC is
not available, the problems with securing DNS-layer redirections
would be avoided by performing redirections in the application-layer.

4. Private DNS and Split Horizon

   The classic view of the uniqueness of DNS names is given in
   [RFC2826]:

   "DNS names are designed to be globally unique, that is, for any
   one DNS name at any one time there must be a single set of DNS
   records uniquely describing protocol addresses, network resources and
   services associated with that DNS name. All of the applications
   deployed on the Internet which use the DNS assume this, and Internet
   users expect such behavior from DNS names."

   However, [RFC2826] "does not preclude private networks from operating
   their own private name spaces," but notes that if private networks
   "wish to make use of names uniquely defined for the global Internet,
   they have to fetch that information from the global DNS naming
   hierarchy." There are various DNS deployments outside of the global
   public DNS, including "split horizon" deployments and DNS usages on
   private (or virtual private) networks. In a split horizon, an
   authoritative server gives different responses to queries from the
   public Internet than they do to internal resolvers; as they often
   differentiate internal from public queries by the source IP address,
   the concerns in Section 3.1.1 relating to trusting source IP
   addresses apply to such deployments. When the internal address space
   range is private [RFC1918], this both makes it easier for the server
   to discriminate public from private and harder for public entities to
   impersonate nodes in the internal network. Networks that are made
   private physically, or logically by cryptographic tunnels, make these
   private deployments more secure. The most complex deployments along
   these lines use multiple virtual private networks to serve different
   answers for the same name to many networks.

   The use cases that motivate split horizon DNS typically involve
   restricting access to some network services - intranet resources such
   as internal web sites, development servers or directories, for
   example - while preserving the ease of use offered by domain names
   for internal users. While for many of these resources, the split
   horizon would not return answers to public resolvers for those
   internal resources (those records are kept confidential from the
   public), in some cases, the same name (e.g., "mail.example.com")
   might resolve to one host internally but another externally. The
   requirements for multiple-VPN private deployments, however, are
   different: in this case the authoritative server gives different, and
   confidential, answers to a set of resolvers querying for the same
   name. While these sorts of use cases rarely arise for traditional
   domain names, where, as [RFC2826] says, users and applications expect
   a unique resolution for a name, they can easily arise when other
   sorts of identifiers have been translated by a mechanism such as DDDS

into "domain name syntax." Telephone calls, for example, are
traditionally routed through highly mediated networks, and an attempt
to find a route for a call often requires finding an appropriate
intermediary based on the source network and location rather than
finding an endpoint (see the distinction between the Look-Up Function
and Location Routing Function in [RFC5486]). Moreover, the need for
responses tailored to the originator and confidentially is easily
motivated when the data returned by the DNS is no longer "describing
protocol addresses, network resources and services," but instead
arbitrary data. Although, for example, ENUM was originally intended
for deployment in the global public root of the DNS (under
e164.arpa), the requirements of maintaining telephone identifiers in
the DNS quickly steered operators into private deployments.

In private environments, it is also easier to deploy any necessary
extensions than it is in the public DNS: in the first place,
proprietary non-standard extensions and parameters can easily be
integrated into DNS queries or responses as the implementations of
resolvers and servers can likely be controlled; secondly,
confidentiality and custom responses can be provided by deploying,
respectively, underlying physical or virtual private networks to
shield the private tree from public queries, and effectively
different virtual DNS trees for each administrative entity that might
launch a query; thirdly, in these constrained environments, caching
and recursive resolvers can be managed or eliminated in order to
prevent any unexpected intermediary behavior. While these private
deployments serve an important role in the marketplace, there are
risks in using techniques intended only for deployment in private and
constrained environments as the basis of a standard solution. When
proprietary parameters or extensions are deployed in private
environments, experience teaches us that these implementations will
begin to interact with the public DNS, and that the private practices
will leak.

While such leakage is not a problem when using the mechanisms
described in Sections 3.1, 3.2, and 3.5 (with private RR types) of
[RFC5507], other extension mechanisms might cause confusion, or harm
if leaked. However, the use of a dedicated suffix (Section 3.3
[RFC5507]) in a private environment might cause confusion if leaked
to the public internet, for example.

That this kind of leakage of protocol elements from private
deployments to public deployments does happen has been demonstrated,
for example, with SIP: SIP implemented a category of supposedly
private extensions (notably the "P-" headers) which saw widespread
success and use outside of the constrained environments for which
they were specifically designed. There is no reason to think that
implementations with similar "private" extensions to the DNS

protocols would not similarly end up in use in public environments.

5. **Principles and Guidance**

   The success of the DNS relies on the fact that it is a distributed
   database, one that has the property that it is loosely coherent and
   that it offers lookup instead of search functionality. Loose
   coherency means that answers to queries are coherent within the
   bounds of data replication between authoritative servers (as
   controlled by the administrator of the zone) and caching behavior by
   recursive name servers. It is critical that application designers
   who intend to use the DNS to support their applications consider the
   implications that their uses have for the behavior of resolvers,
   intermediaries including caches and recursive resolvers, and
   authoritative servers.

   It is likely that the DNS provides a good match whenever applications
   needs are aligned with the following properties:

      Data stored in the DNS can be propagated and cached using
      conventional DNS mechanisms, without intermediaries needing to
      understand exceptional logic (considerations beyond the name, type
      and class of the query) used by the authoritative server to
      formulate responses

      Data stored in the DNS is indexed by keys that do not violate the
      syntax or semantics of domain names

      Applications invoke the DNS to resolve complete names, not
      fragments

      Answers do not depend on an application-layer identity of the
      entity doing the query

      Ultimately, applications invoke the DNS to assist in communicating
      with the domain of the name they are resolving

   Whenever one of the five properties above does not apply to one's
   data, one should seriously consider whether the DNS is the best place
   to store actual data. On the other hand, good indicators that the
   DNS is not the appropriate tool for solving problems is when you have
   to worry about:

      Populating metadata about domain boundaries within the tree - the
      points of administrative delegation in the DNS are something that
      applications should in general not be aware of

      Domain names derived from identifiers that do not share a
      semantics or administrative model compatible with the DNS

   Selective confidentiality of data stored in and provided by the
   DNS

   DNS responses not fitting into UDP packets, unless EDNS0 is
   available, and only then with the caveats discussed in
   Section 3.2.1

In cases where applications require these sorts of features, they are
simply better instantiated by independent application-layer protocols
than the DNS. The query-response semantics of the DNS are similar to
HTTP, for example, and the objects which HTTP can carry both in
queries and responses can easily contain the necessary structure to
manage compound queries. Many applications already use HTTP because
of widespread support for it in middleboxes. Similarly, HTTP has
numerous ways to provide the necessary authentication, authorization
and confidentiality properties that some features require, as well as
the redirection properties discussed in Section 3.4. The overhead of
using HTTP may not be appropriate for all environments, but even in
environments where a more lightweight protocol is appropriate, DNS is
not the only alternative.

Where the administrative delegations of the DNS form a necessary
component in the instantiation of an application feature, there are
various ways that the DNS can bootstrap access to an independent
application-layer protocol better suited to field the queries in
question. For example, since NAPTR or URI resource
[I-D.faltstrom-uri] records can contain URIs, those URIs can in turn
point to an external query-response service such as an HTTP service
where more syntactically and semantically rich queries and answers
might be exchanged.

6. Security Considerations

   Many of the concerns about how applications use the DNS discussed in
   this document involve security. The perceived need to authenticate
   the source of DNS queries (see Section 3.1.1) and authorize access to
   particular resource records also illustrates the fundamental security
   principles that arise from offloading certain application features to
   the DNS. As Section 3.2.1 observes, large data in the DNS can
   provide a means of generating reflection attacks, and without the
   remedies discussed in that section (especially the use of EDNS0 and
   TCP) the presence of large records is not recommended. Section 3.4
   discusses a security problem concerning redirection that has surfaced
   in a number of protocols (see [I-D.barnes-hard-problem]).

   While DNSSEC, were it deployed universally, can play an important
   part in securing application redirection in the DNS, DNSSEC does not
   provide a means for a resolver to authenticate itself to a server,
   nor a framework for servers to return selective answers based on the
   authenticated identity of resolvers. The existing feature set of
   DNSSEC is however sufficient for providing security for most of the
   ways that applications traditionally have used the DNS. The
   deployment of DNSSEC ([RFC4033] and related specifications) is
   therefore encouraged. Nothing in this document is intended to
   discourage the implementation, deployment, or use of Secure DNS
   Dynamic Updates ([RFC2136], though this document does recommend that
   large data in the DNS be treated in accordance with the guidance in
   Section 3.2.1.

7. IANA Considerations

   This document contains no considerations for the IANA.

8. IAB Members at Time of Approval

Internet Architecture Board Members at the time this document was
approved were:

[TO BE INSERTED]

9. Acknowledgements

The IAB appreciates the comments and often spirited disagreements of Ed Lewis, Dave Crocker, Ray Bellis, Lawrence Conroy, Ran Atkinson, Patrik Faltstrom and Eliot Lear.

10. Informative References

   [I-D.barnes-hard-problem]
             Barnes, R. and P. Saint-Andre, "High Assurance Re-
             Direction (HARD) Problem Statement",
             draft-barnes-hard-problem-00 (work in progress),
             July 2010.

   [I-D.bellis-enum-send-n]
             Bellis, R., "IANA Registrations for the 'Send-N'
             Enumservice", draft-bellis-enum-send-n-02 (work in
             progress), June 2008.

   [I-D.faltstrom-uri]
             Faltstrom, P. and O. Kolkman, "The Uniform Resource
             Identifier (URI) DNS Resource Record",
             draft-faltstrom-uri-06 (work in progress), October 2010.

   [I-D.ietf-enum-unused]
             Stastny, R., Conroy, L., and J. Reid, "IANA Registration
             for Enumservice UNUSED", draft-ietf-enum-unused-04 (work
             in progress), March 2008.

   [I-D.kaplan-dnsext-enum-sip-source-ref-opt-code]
             Kaplan, H., Walter, R., Gorman, P., and M. Maharishi,
             "EDNS Option Code for SIP and PSTN Source Reference Info",
             draft-kaplan-dnsext-enum-sip-source-ref-opt-code-03 (work
             in progress), October 2011.

   [I-D.livingood-dns-redirect]
             Creighton, T., Griffiths, C., Livingood, J., and R. Weber,
             "DNS Redirect Use by Service Providers",
             draft-livingood-dns-redirect-03 (work in progress),
             October 2010.

   [I-D.saintandre-tls-server-id-check]
             Saint-Andre, P. and J. Hodges, "Representation and
             Verification of Domain-Based Application Service Identity
             in Certificates Used with Transport Layer Security",
             draft-saintandre-tls-server-id-check-09 (work in
             progress), August 2010.

   [I-D.vandergaast-edns-client-ip]
             Contavalli, C., Gaast, W., Leach, S., and D. Rodden,
             "Client IP information in DNS requests",
             draft-vandergaast-edns-client-ip-01 (work in progress),
             May 2010.

   [RFC0882] Mockapetris, P., "Domain names: Concepts and facilities",
              RFC 882, November 1983.

   [RFC0974] Partridge, C., "Mail routing and the domain system",
              RFC 974, January 1986.

   [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
              STD 13, RFC 1034, November 1987.

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

   [RFC1464] Rosenbaum, R., "Using the Domain Name System To Store
              Arbitrary String Attributes", RFC 1464, May 1993.

   [RFC1530] Malamud, C. and M. Rose, "Principles of Operation for the
              TPC.INT Subdomain: General Principles and Policy",
              RFC 1530, October 1993.

   [RFC1794] Brisco, T., "DNS Support for Load Balancing", RFC 1794,
              April 1995.

   [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and
              E. Lear, "Address Allocation for Private Internets",
              BCP 5, RFC 1918, February 1996.

   [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995,
              August 1996.

   [RFC2052] Gulbrandsen, A. and P. Vixie, "A DNS RR for specifying the
              location of services (DNS SRV)", RFC 2052, October 1996.

   [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound,
              "Dynamic Updates in the Domain Name System (DNS UPDATE)",
              RFC 2136, April 1997.

   [RFC2168] Daniel, R. and M. Mealling, "Resolution of Uniform
              Resource Identifiers using the Domain Name System",
              RFC 2168, June 1997.

   [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS
              Specification", RFC 2181, July 1997.

   [RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397,
              August 1998.

   [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)",
              RFC 2671, August 1999.

   [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

   [RFC2826] Internet Architecture Board, "IAB Technical Comment on the
              Unique DNS Root", RFC 2826, May 2000.

   [RFC2915] Mealling, M. and R. Daniel, "The Naming Authority Pointer
              (NAPTR) DNS Resource Record", RFC 2915, September 2000.

   [RFC2916] Faltstrom, P., "E.164 number and DNS", RFC 2916,
              September 2000.

   [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic
              Update", RFC 3007, November 2000.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              June 2002.

   [RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation
              Protocol (SIP): Locating SIP Servers", RFC 3263,
              June 2002.

   [RFC3401] Mealling, M., "Dynamic Delegation Discovery System (DDDS)
              Part One: The Comprehensive DDDS", RFC 3401, October 2002.

   [RFC3402] Mealling, M., "Dynamic Delegation Discovery System (DDDS)
              Part Two: The Algorithm", RFC 3402, October 2002.

   [RFC3403] Mealling, M., "Dynamic Delegation Discovery System (DDDS)
              Part Three: The Domain Name System (DNS) Database",
              RFC 3403, October 2002.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, January 2005.

   [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "DNS Security Introduction and Requirements",
              RFC 4033, March 2005.

   [RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J.,
              and T. Wright, "Transport Layer Security (TLS)
              Extensions", RFC 4366, April 2006.

   [RFC4367] Rosenberg, J. and IAB, "What's in a Name: False
              Assumptions about DNS Names", RFC 4367, February 2006.

   [RFC4732] Handley, M., Rescorla, E., and IAB, "Internet Denial-of-
              Service Considerations", RFC 4732, December 2006.

   [RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton,
              J., and M. Thomas, "DomainKeys Identified Mail (DKIM)
              Signatures", RFC 4871, May 2007.

   [RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure
              Subject Alternative Name for Expression of Service Name",
              RFC 4985, August 2007.

   [RFC5486] Malas, D. and D. Meyer, "Session Peering for Multimedia
              Interconnect (SPEERMINT) Terminology", RFC 5486,
              March 2009.

   [RFC5507] IAB, Faltstrom, P., Austein, R., and P. Koch, "Design
              Choices When Expanding the DNS", RFC 5507, April 2009.

   [RFC5922] Gurbani, V., Lawrence, S., and A. Jeffrey, "Domain
              Certificates in the Session Initiation Protocol (SIP)",
              RFC 5922, June 2010.

   [RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P.
              Roberts, "Issues with IP Address Sharing", RFC 6269,
              June 2011.

Authors' Addresses

    Jon Peterson
    NeuStar, Inc.

    Email: jon.peterson@neustar.biz

    Olaf Kolkman
    NLnet Labs

    Email: olaf@nlnetlabs.nl

    Hannes Tschofenig
    Nokia Siemens Networks

    Email: Hannes.Tschofenig@gmx.net

    Bernard Aboba
    Microsoft Corporation

    Email: bernarda@microsoft.com