The Internet Protocol Journal

March 2008 Volume 11, Number 1

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at: **www.cisco.com/ipj**

From The Editor

The *Domain Name System* (DNS) was not designed to support anything beyond 7-bit ASCII characters. Thus my middle name, Jørgen, or my colleague's surname, Fältström, cannot be used in a domain name. In fact, even using such strings on the left side of the @-sign—or in the body of an e-mail message—is problematic. We often find ourselves ignoring this limitation, using either "Jorgen" and "Faltstrom" or in some cases the two-letter convention "Joergen" and "Faeltstroem." As Scandinavians, Mr. Fältström and I are relatively lucky in that our languages contain only three characters in addition to those that can be represented by 7-bit ASCII. This, of course, isn't true for such languages as Arabic, Chinese, Japanese, or Korean, to name just a few. The IETF, ICANN, and others have been working hard to design and deploy a system that will allow native characters to appear in the DNS. Our first article discusses these efforts, known collectively as *Internationalized Domain Names* (IDNs). Geoff Huston gives an overview of IDNs and describes the many technical and political challenges that must be overcome in order to deploy such a system.

Recent activities have focused much attention on IPv6 deployment. Experiments have been conducted at several major Internet events (NANOG, APRICOT, and IETF) to "turn off" IPv4 for a period of time to test connectivity and interoperability to the outside world. You can read more about these experiments in our "Fragments" section on page 41. Such experiments provide valuable information about what works and what doesn't, and several more IPv4 "outages" are planned for 2008 and beyond. At the same time, researchers have been looking at ways to scale the routing system of the Internet, regardless of IP protocol version. One such approach is the *Locator/Identifier Separation Protocol* (LISP), which Dave Meyer describes in our second article.

The next issue of *The Internet Protocol Journal*, to be published sometime in June 2008, will be our Tenth Anniversary issue. We would love to hear your reflections on the last ten years of this journal and about the Internet as a whole over the same time period. Send your Letters to the Editor to **ipj@cisco.com**

—Ole J. Jacobsen, Editor and Publisher
ole@cisco.com

Internationalizing the Domain Name System

by Geoff Huston, APNIC

Considering the global reach of the Internet, internationalizing the network sounds like a tautology. Surely the Internet is already truly "international," isn't it? The Internet reaches around the globe to every country, doesn't it? And no matter where you may travel these days, an Internet café is just around the corner. How much more "international" can you get?

But maybe I'm just being too parochial here when I call it a tautology. I use a dialect of the English language, and all the characters I need are contained in the *Western Latin* character set. Therefore, I avoid using a non-English language on the Internet; the only language I use on the Internet is English, and all the characters I need are encompassed in the ASCII character set. If I tried to use the Internet with a language that has a non-Latin character set and a different script, my experience would probably be different—and acutely frustrating. If my native language used a different script and a different text flow than English, I would probably give the Internet an extremely low score for ease of use. It is not as simple as managing glyph sets to represent the characters of the language; although it is relatively easy to present pictures of characters in a variety of fonts and scripts, using them in an intuitive and natural way in the context of the Internet becomes more challenging.

Mostly what is needed is good *localization,* or adapting the local computing environment to suit local linguistic needs. This environment may include support for additional character sets and additional language scripts, and perhaps altering the direction of text flow, or even the entire layout of the information.

For example, Japanese is traditionally written in a format called *Tategaki.* In this format, the text flows in columns going from top to bottom, with columns ordered from right to left. Modern Japanese also uses another writing format, called *Yokogaki.* This writing format is identical to that of European languages such as English, where the text flows from left to right in successive rows from top to bottom.

Today, the left-to-right direction is dominant in Japanese *Kana,* Chinese characters, and Korean *Hangul* for horizontal writing. This change is due partly to the influence of English, and partly to the increased use of computerized typesetting and word-processing software, most of which does not directly support right-to-left layout of East Asian languages. It would appear that even *Yokogaki* is an outcome of the lack of capability of IT systems to correctly cope with localization.[1]

One topic, however, does not appear to have a compellingly obvious localization solution in this multilingual environment: the *Domain Name System* (DNS). The subtle difference here is that the DNS is the "glue" that binds all users' language symbols together, and performing localized adaptations to suit local language use needs is not enough. The DNS spans the entire network, so what works for me in the DNS must also work for you. What we need is a means to allow the use of all of these language symbols within the same system, or *internationalization*.

The DNS is the most prevalent means of initiating a network transaction, whether it is a *BitTorrent* session, the Web, e-mail, or any other form of network activity. But the DNS name string is not just an arbitrary string of characters. What you find in the DNS is most often a sequence of words or their abbreviations, and the words are generally English words, using characters drawn from a subset of the Latin character set. Perhaps unsurprisingly, some implementations of the DNS also assume that all DNS names must be constructed only from this ASCII character set, and these implementations are incapable of supporting a larger character repertoire. If you want to use a larger character set in order to represent various diacritics, such as acute and grave symbols, umlauts and similar marks, then the deployed DNS can be resistant to this use, and may provide incorrect responses to queries that include such characters. And if you want to use words drawn from languages that do not use the western script for their characters, such as Japanese or Thai, for example, then the DNS is highly resistant to this form of multilingual use.

Latin and Roman Alphabets

The default Latin alphabet is the Roman[2] alphabet, supplemented with G, J, U, W, Y, Z, and lowercase variants. Additional letters may be formed:

- As *ligatures,* as W was from VV, for example Æ (*ash*) from AE, *oethel* Œ from OE, *eszett* ß from ſz (long s + z), *engma* ŋ from NG, *ou* Ȣ from OU, Ñ from NN, or ä from ae
- By *diacritics,* such as Å, Č, and Ų
- As *digraphs,* such as fi and fl
- By modification, as J was from I, G from C, Ø from O, *eth* Ð from D, *yogh* Ȝ from G, or *schwa* Ə from E
- By borrowing from another alphabet entirely, as *thorn* Þ and *wynn* Ƿ were from Futhark (Runic)

Over the years we have done a reasonable job of at least displaying non-Latin-based scripts within many applications, and although at times it appears to represent a less-than-reasonable compromise, it is possible to enter non-Latin characters on computer keyboards. So it appears to be possible to customise a local computing environment to use a language other than English in a relatively natural way.

But what happens when we extend the scope to consider multilingual support in the wider world of the Internet?

Again the overall story is not all that bad. We can use non-Latin character scripts in e-mail, in all kinds of Web documents, and in a wide variety of network applications. We can tag content with a language context to allow display of the content in the correct language using the appropriate character sets and presentation glyphs. However, until recently, one area continued to stick steadfastly to its ASCII roots: the DNS. This article addresses DNS internationalization, or *Internationalized Domain Names* (IDNs).

What do we mean when we talk of "internationalizing the DNS"? It refers to an environment where English, and the Latin character set, is just one of many languages and scripts in use, and where a communication is initiated in one locale and then the language and presentation are preserved wherever the communication is received.

Terminology

The following terms are used in this article:

Language: A language uses characters drawn from a collection of scripts.

Script: A script is a collection of characters that are related in their use by a language.

Character: A character is a unit of a script.

Glyph: The presentation of a character within the style of a font is called a glyph.

Font: A font is a collection of glyphs encompassing a script character set that share a consistent presentation style.

Multiple languages can use a common script, and any locale or country may use many languages, reflecting the diversity of its population and the evolution of local dialects within communities.

It is also useful to remember the distinction between internationalization and localization. *Internationalization* is concerned with providing a common substrate that many—preferably all—languages and all users can use, whereas *localization* is concerned with the use of a particular language within a particular locale and within a defined user population. Unsurprisingly, the two concepts are often confused, particularly when true internationalization is often far more difficult to achieve than localization.

Internationalizing the DNS

The objective is the internationalization of the DNS, such that the DNS can support the union of all character sets while preserving the absence of ambiguity and uncertainty in terms of resolution of any individual DNS name. We need to describe all possible characters in all languages and allow their use in the DNS. So the starting point is the "universal character set," and that appears to be Unicode.

One of the basic building blocks for internationalization is a character set that is the effective union of all character sets. *Unicode*[3] is intended to be such a universal encoding of characters (and symbols) in the contexts of all scripts and all languages. The current version of the *Unicode Standard,* Version 5.0, contains 98,884 distinct coded graphic characters.

A sequence of Unicode code points can be represented in multiple ways by using different character encoding schemes in a *Unicode Transformation Format* (UTF). The most commonly used schemes are UTF-8 and UTF-16.

UTF-8 is a variable-length encoding using 8-bit words, meaning that different code points require different numbers of bytes. The larger the index number of a code point, the more bytes are required to represent it using UTF-8. For example, the first 127 Unicode code points, which correspond exactly to the values used by the ASCII character set (which maps only 127 characters), can be represented using only 8 bits in UTF-8, using the same 8-bit values as in ASCII. UTF-8 can require up to 32 bits to encode certain code points. A criticism of UTF-8 is that it "penalizes" certain scripts by requiring more bytes to represent their code points. The IETF has made UTF-8 its preferred default character encoding for internationalization of Internet application protocols.

UTF-16 is a variable-length character encoding using 16-bit words. Characters in the *Basic Multilingual Plane* are mapped into a single 16-bit word, with other characters mapped into a pair of 16-bit words.

UTF-32 is a fixed-length encoding that uses 32 bits for every code point. This encoding tends to make for a highly inefficient coding that is, generally, unnecessarily large, because most language uses of Unicode draw characters from the Basic Multilingual Plane, making the average code size 16 bits in UTF-16 as compared to the fixed-length 32 bits in UTF-32. For this reason UTF-32 is far less commonly used than UTF-8 and UTF-16.

But languages, which we humans change in various ways every day, are not always definitive in their use of characters, and Unicode has some weaknesses in terms of identifying a context of a script and a language for a given character sequence. The common approach to using Unicode encodings in application software is to use an associated "tag," allowing content to be tagged with a script and an encoding scheme. For example, a content tag might read: "This text has been encoded using the KOI-8 encoding of the CYRILLIC script."

Tagging allows for decoding of the encoded characters in the context of a given script and a given language. This decoding has been useful for e-mail or Web page content, but tagging breaks down in the context of the DNS. There is no natural space in DNS names to contain language and script tags, implying that attempting to support internationalization in the DNS has to head toward a "universal" character set and a "universal" language context. Another way of looking at this situation is that the DNS must use an implicit tag of "all characters and all languages."

The contexts of the use of DNS names have numerous additional artefacts. What about domain-name label separators? This "dot" between DNS "words," or a DNS label separator, is an ASCII period character. In some languages, such as Thai, for example, there is no natural use of such a label separator. In a similar vein, are URLs intended to be visible to end users? If so, then we may have to transform the punctuation components of the URL into the script of the language. Therefore, we may need to understand how to manage protocol strings, such as "http:" and separators such as the "/" character. To complete the integrity of the linguistic environment, these elements may also require local presentation transformations.

For example, the Thai alphabet uses 44 consonants and 15 basic vowel characters, which are horizontally placed, from left to right, with no intervening space, to form syllables, words, and sentences. Vowels associated with consonants are nonsequential: they can be located before, after, above, or below their associated consonant, or in a combination of these positions. The latter in particular causes problems for computer encoding and text rendering[4].

The DNS name string reads left to right, and not right to left or top to bottom as in other script and language cultures. How much of this string you can encode in the DNS and how much must be managed by the application is part of the problem here. Is the effort to internationalize the DNS with multiple languages restricted to the "words" of the DNS, leaving the implicit left-to-right ordering and the punctuation of the DNS unaltered? If so, how much of this ordering and punctuation is a poor compromise, in that these DNS conventions in such languages are not natural translations?

The Unicode UTF-8, UTF-16, and UTF-32 encodings all require an "8-bit clean" storage and transmission medium. Because "traditional" DNS domain names are representable with 7-bit ASCII characters, not all applications that process domain names preserve the status of the eighth bit; in other words, they are not 8-bit clean. This situation stimulated significant debate in the IETF's *IDN Working Group* and influenced the direction of the standards development into the area of application assistance: the group took a very conservative view of the capabilities of the DNS as a restricted ASCII code application.

Accordingly, we now see the DNS itself as a heavily restricted "language." The prudent use of the DNS specifies, in RFC 1035[5], a sequence of "words" (or "labels"), where each label conforms to the "*L*etter, *D*igit, *H*yphen" (LDH) restriction. Each DNS label must begin with a letter, restricted to the Latin character subset of "A" through "Z" and "a" through "z", followed by a sequence of letters, digits, or hyphens, with a trailing letter or digit, and no trailing hyphen. Furthermore, the case of the letter is not important to the DNS, so, within the DNS "a" is equivalent to "A", and so on, and all characters are encoded in monocase ASCII. The DNS uses a left-to-right ordering of these labels, with the ASCII period as the label delimiter. This restriction is often referred to as the *LDH Convention*.

The challenge posed with the effort of *internationalizing* the DNS is one of attempting to create a framework that allows Internet applications—and the DNS in particular—to be set in the user's own language in an entirely natural fashion, and yet allow the DNS to operate in a consistent and deterministic manner within its restricted "language." In other words, we all should be able to use browsers and e-mail systems using our own language and scripts, yet still be able to communicate naturally with others who may be using a different language interface.

The most direct way of stating the choice set of IDN design is that IDNs either change the "prudent use" of the deployed DNS into something quite different by permitting a richer character repertoire in all parts of the DNS, or IDNs change the applications that want to support a multilingual environment such that they have to perform some form of encoding transfer to map between a language string using Unicode characters and an "equivalent" string using the restricted DNS LDH character-set repertoire. It appears that options other than these two lead us into fragmented DNS roots, and having already explored that particular concept in the past, not many of us want to return to that subject. So if we want to maintain a cohesive and unified symbol space for the DNS, then either the deployed DNS has to become 8-bit clean, or applications have to do the work and present to the DNS an encoded form of the Unicode sequences that conform to the restricted DNS character repertoire.

The IDN Framework

If you are an English language user with the ASCII character set, the DNS name you enter into the browser—or the domain part of an e-mail address—is almost the same string as the string that is passed to the DNS resolver to resolve into an address (the difference is the conversion of the characters into monocase). If you want to send a mail message, you might send it to **`user@example.com`**, for example, and the domain name part of this address, **`example.com`**, is the string used to query the DNS for an *MX Resource Record* in order to establish how to actually deliver the message.

But what if you want to use a domain name that is expressed in another language? What if the e-mail address is **`user@`**記念**`.com`**? The problem here is that this domain name cannot be "naturally" expressed in the restricted syntax of the DNS, and although this domain name may have a perfectly reasonable Unicode code sequence, this encoded sequence is not a strict LDH sequence, nor is it case-insensitive (whatever "case" may mean in an arbitrary non-Latin script). It is here that IDNs depart from the traditional view of the DNS and use a hybrid approach to the task of mapping these language strings into network addresses.

The IDN Working Group of the IETF was formed in 2000 with the goal of developing standards to internationalize domain names. The working group's charter was to specify a set of requirements and develop IETF standards-track protocols to allow use of a broader range of characters in domain names. The outcome of this effort was the *IDN in Applications* (IDNA) framework, published as RFCs 3454, 3490, 3491, and 3492.[6,7,8,9]

Rather than attempting to expand the character repertoire of the DNS itself, the IDN working group used an *ASCII Compatible Encoding* (ACE) to encode the binary data of Unicode strings that would make up IDNs into an ASCII character encoding. The concept is similar to the Base64 encoding used by the *Multipurpose Internet Mail Extension* (MIME) e-mail standards, but whereas Base64 uses 64 characters from ASCII, including uppercase and lowercase, the ACE approach requires the smaller DNS-constrained LDH subset of ASCII.

The working group examined various ACE algorithms in its efforts to converge to a single standard (because different encoding algorithms have different compression goals and yields) and encode the data using slightly different subsets of ASCII. Most proposals specified a prefix to the ACE coding to tag the fact that this string was, in fact, an encoded Unicode string. The IETF adopted *punycode* as its standard IDN ACE[9]. Punycode was chosen for its efficient encoding compression properties that produce short ACE strings. For example, the domain name of 記念**`.com`** encodes with punycode to **`xn--h7tw15g.com`**.

IDN in Applications

Although an ASCII-compatible encoding of Unicode characters allows representation of an IDN in a form that will probably not be corrupted by the deployed DNS infrastructure on the Internet, an ACE alone is not a full solution. The IDN approach also needs to specify how and where the ACE should be applied.

The overall approach to IDNs is relatively straightforward. In IDN the application has a critical role to play. The application takes a domain name that is expressed in a particular language using a particular script—and potentially in a particular character and word order that is related to that language—and produces an ASCII-compatible LDH-encoded version of this DNS name. Equally, when presenting a DNS string to the user, the application should take the LDH-encoded DNS name and transform it to a presentation sequence of glyphs that correspond to the original string in the original script.

It is critical that all applications perform this encoding and decoding function correctly, deterministically, and uniformly. In fact, this capability is critical to the entire IDN framework.

The basic shift in the DNS semantics that IDNs bring to the DNS is that the actual name itself is no longer in the DNS. An encoded version of the canonical name form sits in the DNS, and applications need to perform the canonical name transformation, as well as the mapping between the Unicode character string and the encoded DNS character string. So we need to agree on what are the "canonical" forms of name strings in every language. We also need to agree on the encoding method, and our various applications must have precise equivalents of these canonical name and encoding algorithms, or the symbolic consistency of the DNS will fail. The problem here is that the DNS does not perform approximate matches or return a set of possible answers to a query. The DNS is a deterministic system that performs a precise match on the query in order to generate a response. The implication here is that if we want the same IDN character sequence to map to the same network response in all cases and all contexts, then all applications must perform precisely the same operations on the character sequence in order to generate the ACE-equivalent label sequence.

RFC 3454[6] defines a presentation layer in IDN-aware applications that is responsible for the punycode ACE encoding and decoding. This new layer in the application architecture is responsible for encoding any internationalized input in domain names into punycode format before the corresponding LDH encoded domain name is passed to the DNS for resolution. This presentation layer is also responsible for decoding the punycode format in IDNs and rendering the appropriate glyphs for the user.

It is a matter of personal perspective whether this solution is an elegant one or it simply shifts an unresolved problem from one area of the IETF to another. The IDNA approach assumes that it is easier to upgrade applications to all behave consistently in interpreting IDNs than it is to change the underlying DNS infrastructure to be 8-bit clean in a manner that would support direct use of Unicode code points in the DNS.

The Presentation Layer Transform for IDNs

The objective here is to define a reliable and deterministic algorithm that takes a Unicode string in a given language and produces a DNS string as expressed in the LDH character repertoire. This algorithm should not provide a unique 1:1 mapping, but should group "equivalent" Unicode strings, where "equivalence" is defined in the context of the language of use, into the same DNS LDH string. Any reverse mapping from the DNS LDH string into the Unicode string should deterministically select the single "canonical" string from the group of possible IDN strings.

Stringprep

The first part of the presentation layer transform is to take the original Unicode string and apply numerous transformations to it to produce a "regular" or "canonical" form of the IDN string. This form of the string is then transformed using the punycode ACE into an encoded DNS string form. The generic name of this process is, in IDN language, "stringprep,"[6] and the particular profile of transformations used in IDNAs is termed "nameprep."[8]

This transform of a Unicode string into a canonical format is based on the observation that many languages have a variety of ways to display the same text and a variety of ways to enter the same text. Although we humans are unconcerned about this concept of expressing an idea in multiple ways, the DNS is an exact equivalence match operation and it cannot tolerate imprecision. So how can the DNS tell that two text strings are intended to be identical, even though their Unicode strings are different? The IDN approach is to transform the string so that all equivalent strings are mapped to the same canonical form, or "stringprep" the string. The stringprep specification is not a complete algorithm, and it requires a "profile" that describes the applicability of the profile, the character repertoire (at the time of writing RFC 3454, it was Unicode 3.2, although the Unicode Consortium has subsequently released Unicode Version 4.0, 4.1, and 5.0), mapping tables normalization, and prohibited output characters.

Mapping

In converting from a string to a *normal,* or canonical, form, the first step is to map each character into its *normalized* equivalent, using a mapping table. This table is conventionally used to map characters to their lowercase equivalent value to ensure that the DNS string comparison is case-insensitive.

Other characters are removed from the string by using this mapping operation because their presence or absence in the string does not affect the outcome of a string-equivalence operation, such as characters that affect glyph choice and placement, but without semantic meaning.

The mapping function will create monocase (specifically lowercase) outcomes and also will eliminate non-significant code points (such as, for example, the Unicode code point 1806; MONGOLIAN TODO SOFT HYPHEN or the Unicode code point 200B; ZERO WIDTH SPACE, if you really wanted to know what a non-significant code point was).

Normalization

Numerous languages use different character sequences for the same meaning. Characters may appear the *same* in presentation format as a glyph sequence, yet have *different* underlying code points. This may be associated with variables ways of combining diacritics, or using canonical code points, or using compatibility characters, and, in some language contexts, performing character reordering. For example, the character Ä can be represented by a single Unicode code point 00C4; LATIN CAPITAL A WITH DIARESIS. Another valid representation of this character is the code point 0041; LATIN CAPITAL LETTER A followed by the separate code point 0398; COMBINING DIARESIS.

The intent of normalization is to ensure that every class of character sequences that are equivalent in the context of a language is translated into a single canonical, consistent format. This consistency of format allows the equivalence operator to perform at the character level using direct comparison without additional language-dependent equivalence operations.

Languages in daily use are not rigid structures, and human use patterns of languages change. Normalization is no more than a best-effort process to detect equivalences in a rigid, rule-managed manner, and it may not always produce predictable outcomes. This unpredictability can be a problem with regard to namespace collisions in the DNS, because it does not increase the confidence level of the DNS as a deterministic exact-match information-retrieval system. IDNs introduce some forms of name approximation into the DNS environment, and the DNS is extremely ill-suited to the related "fuzzy-search" techniques that accompany such approximations.

Filtering Prohibited Characters

The last phase in string preparation is removal of prohibited characters, including the various Unicode white-space code points, control code points and joiners, private-use code points, and other code points used as surrogates or tags.

Right-to-Left Characters

As an option for a particular stringprep profile, you can perform a check for right-to-left displayed characters, and if any are found, make sure that the whole string satisfies the requirements for bidirectional strings. The Unicode standard has an extensive discussion of how to reorder glyphs for display when dealing with bidirectional text such as Arabic or Hebrew. All Unicode text is stored in logical order as distinct from the display order.

Nameprep: A Stringprep Profile for the DNS

The nameprep profile[8] specifies stringprep for internationalized domain names, specifying a character repertoire (in this case the specification references Unicode 3.2) and a profile of mappings, normalization (form "KC"), prohibited characters, and bidirectional character handling. The outcome is that two-character sequences can be considered equivalent in the context of IDNs if, by following the sequences of operations defined by the nameprep profile, the resultant sequences of Unicode code points are identical. These code point sequences are the "canonical" forms of names that the DNS uses.

The Punycode ASCII-Compatible Encoding

The next step in the processing of IDN names by the application is to transform this canonical form of the Unicode name string into a LDH-equivalent string using an ACE. The algorithm used, *punycode,* uses a highly efficient encoding, attempting to limit the extent to which Unicode sequences become extended-length ACE strings.

The algorithm first divides the input code points into a set of "basic" code points that require no further encoding, and the set of "extended" code points. The algorithm takes the basic code points and reproduces this sequence in the encoded string: the "literal portion" of the string. A delimiter is then added to the string. This delimiter is a basic code point that does not occur in the remainder of the string. The extended code points are then added to the string as a series of integers expressed through an encoding into the basic (LDH) code set.

These additions of the extended code points are done primarily in the order of their Unicode values, and secondarily in the order in which they occur in the string. The encoding of the code point and its insertion position is done by using a difference, or offset, encoding, so that sequences of clustered code points, such as would be found in a single language, encode efficiently.

For example, the German language string *bücher* uses basic codes for all characters except the *ü* character. The punycode algorithm copies all the basic codes, followed by a "-". The value and position of the *ü* insertion now has to follow.

The encoded form for *ü* (code 252) is at the position between the first and second basic characters. Using the punycode[10] algorithm gives a delta code of 745, a value that can be expressed in base 35 as (21 x 35) + 10. This code point and the position information are expressed in base 35 notation as (10,22,1), or in reverse notation, with the encoding **kva**. So the punycode encoding of *bücher* is **bcher-kva**. The internationalized domain-name format prepends the string **xn--** to the punycode string, resulting in the encoded IDN domain-name form of **xn--bcher-kva**.

IDNS and Our Assumptions About the DNS

At this stage it should be evident that we have the code points for characters drawn from all languages, and the means to create canonical forms of various words and express them in an encoded form that the DNS can resolve.

However, there is more to IDNs than the encoding algorithm. Although a massive number of discrete code points exist in the realm of Unicode, all these distinct characters are not necessarily displayed in unique ways. Indeed, given a relatively finite range of glyphs, the same glyph can display numerous discrete code points.

The often-quoted example with IDNs and name confusion is the name **paypal**. What is the difference between **www.paypal.com** and **www.paypal.com**? There is a subtle difference in the first "a" character, where the second domain name has replaced the Latin *a* with the Cyrillic *а*. Did you spot the difference? Of course not. These *homoglyphs* are cases where the underlying domain names are distinct, yet their appearance is indistinguishable. In the first case the domain name **www.paypal.com** is resolved in the DNS with the query string **www.paypal.com**, yet in the second case the query string **www.paypal.com** is translated by the application to the DNS query string **www.xn--pypal-4ve.com**. How can you tell one case from the other?

This example is by no means a unique case in the IDN realm. The reports "Unicode Security Considerations" (Unicode Technical Report 36) and "Unicode Security Mechanisms" (Unicode Technical Report 39) provide many more examples of postnormalization homographs.

There is no clear and unique relationship between characters and glyphs. Cyrillic, Latin, and Greek share numerous common glyphs. Glyphs may change their shape depending on the character sequence, multiple characters may produce a single glyph, such as the character pair *f l* being displayed as the single glyph *fl*, and a single character may generate multiple glyphs.

Homoglyphs extend beyond a conventional set of characters and include syntax elements as well. For example, the Unicode point 0244 FRACTION SLASH is often displayed using the slash glyph, allowing URLs of the form **http://a.com/e.com**. Despite its appearance, this is not a reference to **a.com** with a locator suffix of **e.com**, but is a reference to the domain **a.com/e.com**.

The basic response is that if you maintain IDN integrity at the application level, then the user just cannot tell. The punycode transform of **www.paypal.com** into **www.xn--pypal-4ve.com** is intended to be a secret between the application and the DNS, because this ASCII-encoded form is simply meaningless to the user. But if this encoded form remains invisible to the user, how can the user detect that the two identically presented name strings are indeed different? Sadly, the only true "security" we have in the DNS is the "look" of the DNS name that is presented to the user, and the user typically works on the principle that if the presented DNS string looks like the real thing, then it must be the real thing.

When this homoglyph problem was first exposed, the response from many browser implementations was to turn off all IDN support in their browser. The next response was to deliberately expose the punycode version of the URL in the browser address bar, so that directing the browser to **http://www.paypal.com** would display in the address bar the URL value of **http://www.xn--pypal-4ve.com**.

The distinction between the two equivalently displayed names was then visible to the user, but the downside was that we were back to displaying ASCII names again, and in this case ASCII versions of punycode-encoded names. If trying to "read" Base64 was difficult, then the displaying—and understanding—of displayed punycode names is surely equally as difficult, if not more so. The encoded names can be completely devoid of any form of useful association or meaning. Although the distinction between ASCII and Cyrillic may be evident by overt differences in their ASCII-encoded names, what happens when the homoglyph occurs across two non-Latin languages? The punycode strings are different, but which string is the "intended" one? Did you mean **http://xn--21bm4l.com** or **http://xn--q2buub.com** when you enter a Hindi script URL?

Using ASCII as the fall-back to resolve name confusion in response to the problem of ambiguities in non-ASCII script names appears to be a nonsensical solution. We appear to be back to guessing games in the DNS again, unfortunately, and particularly impossible guessing games at that.

These days most popular browsers display the glyphs, rather than the ASCII punycode, but once more we are back to the homoglyph problem.

If the intention in the IDN effort was to preserve the deterministic property of DNS resolution, such that a DNS query can be phrased deterministically and not have the query degenerate into a search term or require the application of fuzzy logic to complete the query, then we are not quite there yet.

The underlying observation is that languages are indeed human-use systems. They can be tricky, and they invariably use what appear to be rules in strange and inconsistent ways. They are also resistant to automated processing and the application of rigid rule sets. The canonical name forms that are produced by nameprep-like procedures are not comprehensive, nor does it appear that such a rigidly defined rule-driven system can produce the desired outcomes in all possible linguistic situations. And if the intention of the IDN effort was to create a completely "natural" environment using a language environment other than English and a display environment that is not reliant on ASCII and ASCII glyphs, while preserving all the other properties of the DNS, then the outcome does not appear to match our original IDN expectations.

The underlying weakness here is the implicit assumption that in the DNS "what you see is what you get," and that two DNS names that look identical are indeed references to the same name, and when resolved in the DNS produce precisely the same resolution outcome. When you broaden the repertoire of appearances of the DNS, such that the entire set of glyphs can be used in the DNS, then the mapping from glyph to underlying code point is not unique. Any effort to undertake such a mapping needs additional context in the form of a language and script context. But the DNS does not carry such a context, making the task of maintaining uniqueness and determinism of DNS name translation essentially impossible if we also want to maintain the property that it is the appearance, or presentation format, of DNS names to the user that is the foundation stone of the integrity of our trust in the DNS.

Some concerns still remain in this space, including the inclusion of various forms of character codes that are in effect invisible. In addition, homoglyphs could be better managed by using a refined definition of IDN labels that lists which Unicode code points can be used in the context of IDNs, excluding all others. It would be helpful if confusing and non-reversible character mappings were removed from the IDN space, including the consistent treatment of ligatures and diacritics, refining the treatment of right-to-left and left-to-right scripts, and removing the dependency on a particular version of the Unicode standard. This effort is under way in the IETF in the context of revisions to the IDNA specification documents.

IDNS, TLDs, and the Politics of the DNS

So why is there a very active debate, particularly within ICANN-related forums, about putting IDN codes into the root of the DNS as alternative *top-level domains* (TLDs)?

I have seen two major lines of argument here; namely the argument that favors the existence of IDNs in all parts of the DNS, including the TLDs, and the argument that favors a more restricted view of IDNs in the root of the DNS that links their use to that of an existing (ASCII-based) DNS label in the TLD zone.

Apparently, those who favor the approach of using IDNs in the top-level zone as just another DNS label see this as a natural extension of adding punycode-encoded name entries into lower levels of the DNS. Why should the root of the DNS be any different, in terms of allowing IDNs? Why should a non-Latin script user of the Internet have to enter the TLD code in its ASCII text form, while entering the remainder of the string in a local language? And in right-to-left scripts, where does this awkward ASCII appendage sit when a user attempts to enter it into an application?

Surely, goes the argument, the more natural approach is to allow any DNS name to be wholly expressible in the user's language, implying that all parts of the DNS should be able to carry native language-encoded DNS names. After all, コンピュータは予約する**.jp** looks wrong as a monolingual domain name. What is that **.jp** appendage doing there in that DNS name? Surely a Japanese user should not have to resort to an ASCII English abbreviation to enter in the country code for Japan, when 日本 is obviously more "natural" in the context of a Japanese user using Japanese script. If we had punycode TLDs then, goes the line of argument, users could enter the entire domain name in their language and have the punycode encoding happen across the entire name string, and then successfully perform a DNS lookup on the punycode equivalent. This way the user would enter the Japanese character sequence: コンピュータは予約する**.**日本 and have the application translate this entry to the DNS string **xn--88j0bve5g9-bxg1ewerdw490b930f.xn--wgv71a**. For this process to work in its entirety uniformly and consistently, the name **xn--wgv71a** needs to be a TLD name.

We can always take this thought process one step further and question the ASCII string **http** and the punctuation symbols **://** for precisely the same reason, but I have not heard (yet) calls for multilingual equivalents of protocol identifier codes. The multilingual presentation of these elements remains firmly in the provenance of the application, rather than attempting to alter the protocol identifiers in the relevant standards.

The line of argument also encompasses the implicit threat that if the root of the DNS does not embrace TLDs as expressed in the language of the Internet's users, then language communities will break away from a single DNS root and meet their linguistic community's requirements in their own DNS hierarchy. Admitting such encoded tags into the DNS root is the least problematic, including the consequence of inactivity, which is cited as being tantamount to condoning the complete fragmentation of the Internet's symbol set.

Of course having an entirely new TLD name in an IDN name format does not solve all of the potential problems with IDNs. How can a user tell what domain names are in the ASCII top level, and what are in the "equivalent" IDN-encoded TLDs? Are any two name spaces that refer to the same underlying name concept equivalent? Is **xn--88j0bve5g9bxg1ewerdw490b930f** appropriately a subdomain of **.jp**, or a subdomain of **xn--wgv71a**? Should the two domains be tightly synchronized with respect to their zone content and represent the same underlying token set, or should they be independent offerings to the marketplace, and allow registrants and the end-user base make implicit choices here? In other words, should the pair of domain names, namely **xn--88j0bve5g9bxg1ewerdw490b930f.xn--wgv71a** and **xn--88j0bve5g9bxg1ewerdw490b930f.jp**, reference precisely the same DNS zone, or should they be allowed to compete, and each find their own "natural" level of market support based on decoupled TLD names of **.jp** and **.xn--wgv71a**?

What does the term *equivalence* really imply here? Is equivalence something as loose as the relationship between **.com** and **.biz**, namely being different abbreviations of words that reflect similar concepts with different name-space populations that reflect market diversity and a competitive supply industry? Or is equivalence a much tighter binding in that equivalent names share precisely the same subdomain name set, and a registration in one of these equivalence names is in effect a name registration across the entire equivalence set?

Even this subject is not readily resolvable given our various interpretations of *equivalence*. In theory, the DNS root zone is populated by ISO two-letter country codes and numerous "generic" TLDs. Under what basis, and under what authority, is **xn--wgv71a** considered an "equivalent" of the ISO 3166 two-letter country code JP? Are we falling into the trap once again of making up the rules as we go along? Is the distinction between **.com** and **.biz** apparent only in English? And why should this distinction apply only to non-Latin character sets? Surely it makes more sense for a native German language speaker to refer to commercial entities as *kommerze,* and the abbreviated TLD name as **.kom**? When we say "multilingual" are we in fact ignoring "multilingual" and looking exclusively at "multiscript"?

Let's put aside the somewhat difficult concept of name equivalence for a second, and assume that this equivalence problem is solved. Also suppose that we want tight coupling across equivalence sets of names.

In other words, what we want is that a name registered in any of the elements of the equivalent domain-name set in all scripts is, in effect, registered in all the equivalent DNS zones. The question is: how should it be implemented in the DNS? One approach that could support tight synchronization of equivalence is to use the DNAME record[11] to create these TLD name aliases for their ASCII equivalents, thereby allowing a single name registration to be resolvable using a root name expressed in any of the linguistic equivalents of the original TLD name. The DNAME entry for all but the "canonical" element of the equivalence set effectively translates all queries to a query on the canonical name. The positive aspects of such an approach is uniformity across linguistic equivalents of the TLD name form—a single name delegation in a TLD domain becomes a name within all the linguistic equivalents of the TLD name without any further delegation or registration required.

Using DNAME as a tool to support sets of equivalent names in the DNS is still in the early stages. The limited experience so far with DNAME indicates that CNAME synthesis places load back on the name servers that would otherwise not be there, and the combination of this synthetic record and DNSSEC starts to get very unwieldy. Also, the IETF is reviewing the DNAME specification with the intention to remove the requirement to perform CNAME synthesis. All of these factors may explain why there is no immediate desire to place DNAMEs in the DNS root zone.

Different interpretations of equivalence in IDN names are possible. The use of DNAMEs as aliases for existing TLDs in effect "locks up" IDNs into the hands of the incumbent TLD name-registry operators. Part of the IDN debate, is, as usual, a debate over the generic TLD registry operators and the associated perception of incumbent monopolies. An alternative approach is to associate a single registrar with each IDN variant of the same generic TLD, allowing a form of "competition" between the various registrars. From the perspective of a coherent symbol space where the same symbol, expressed in any language script, resolves in the same fashion, such independent registries are not overly consistent with such a model of registry diversity in a multilingual environment. In this case such an artifice of IDN "competition" may well do more harm than good for Internet users.

It appears that another line of argument is that the DNS top-level name space is very conservatively managed, and new entries into this space are not made lightly. There are concerns of stability of operation, of attempting to conserve a coherent namespace, and the ever-present consideration that if we manage to "break" the DNS root zone it would be an irrevocable act.

This line of argument recognizes the very hazy nature of name equivalence in a multilingual environment and is based on the proposition that the DNS is incapable of representing such imprecision with any utility. The DNS is not a search engine, and the DNS does not handle imprecision at all well. Again, goes the argument, if this is the case then can we push this problem back to the application rather than trying to bend the DNS? If an application is capable of translating, say, 日本 into **xn--wgv71a**, and considering that the TLD name space is relatively small, it appears that having the application performing a further translation of this intermediate form punycode string into the ASCII string **jp** is not a particularly challenging form of table lookup. In such a model no new TLD aliases or equivalences are required in the root zone of the DNS. If we are prepared to pass the execution of the presentation layer of the DNS to the application layer to perform, then why not also ask this same presentation layer to perform the step of further mapping the punycode ACE equivalents of the TLDs to the actual ASCII TLDs, using some richer language context that the application may be aware of that is not viable strictly within the confines of the DNS?

So, with respect to the question of whether IDN TLDS should be loaded into the DNS at all, and, if so, whether they should represent an opportunity for further diversity in name supply or be constrained to be aligned to existing names, and precisely how name equivalence is to be interpreted in this context, then it appears that ICANN has managed to place itself in a challenging situation. In not making a decision, those with an interest in having diverse IDN TLDs appear to derive some pleasure in pointing out that the political origins of ICANN and its strong linguistic bias to English are influencing it to ignore non-English language use and non-English language users of the Internet. Where dramatic statements are called for, such statements often use terms such as "cultural imperialism" to illustrate the nature of the linguistic insult. The case has been made repeatedly, in support of IDN TLDs, that an overwhelming majority of Internet users and commercial activity of the Internet is in languages other than native English, and the imposition of ASCII labels on the DNS is an unnatural imposition on the overwhelming majority of Internet users.

On the other hand, most decisions to permit some form of entry in the DNS are generally seen as irrevocable, and building a DNS that is littered with the legacy of various non-enduring name technologies and poor ad hoc decisions to address a particular concern or problem without any context of a longer-term framework seems also to represent a step along a direction leading to a heavily littered and fragmented Internet where, ultimately, users cannot communicate with each other.

What about global interoperability and the Internet? Should we just take the easy answer and simply give up on the entire concept? Well of course not! But, taking a narrower perspective, are IDNs simply not viable in the DNS? I would suggest that not only is this question one that was overtaken by events years ago, but even if we want to reconsider it now, then the answer remains that any users using their local language and local script should have an equally "natural" experience. IDNs are a necessary and valuable component of the symbol space of any global communications system, and the Internet is no exception. However, we also should recognize that we do need combinations of both localization and globalization, and that we are voicing some pretty tough objectives. Is the IDNA approach enough? Is our assumption that an unaltered DNS with application-encoded name strings represents a rich enough platform to preserve the essential properties of the DNS while allowing true multilingual use of the DNS? On the other hand, taking a pragmatic view of the topic, is what we have with IDNA enough for us to work on, and is the alternative of reengineering the entire fabric of the DNS into an 8-bit clean system just not a viable option?

I suspect that the framework of IDNA is now the technology for IDNs for the Internet, and we simply have to move on from here and deliberately take the stance of understanding the space from users' perspectives when we look at the policy concerns of IDNs. The salient questions from such perspectives include: "What is the "natural" thing to do?" and "What causes a user the least amount of surprise?" Because in this world, what works for the user is what works for the Internet as a whole.

Further IDN News

IDNs are by no means completed work. Development continues in the Unicode forum on elaboration of character sets, and there are further proposals in the IETF to continue a complementary standards activity of refining the IDN documents.

In February 2008 the *Applications Area* of the IETF announced a proposal for further work on IDNs. The proposal has noted that the existing RFC documents are tied to version 3.2 of Unicode, while the Unicode Consortium has released version 5.0.0.

The proposed work is to consider revision of the IDN documents to untie the Internet specifications that define validity based on Unicode properties from specific versions of Unicode using algorithms. It is also proposed that these updates study revision of bi-directional algorithms, and to permit the use of some scripts that were inadvertently excluded by the original Internet specification.

This is not intended to be a major rewrite of the IDN approach, and, in particular, IDNs will continue to use the **xn--** prefix, the same Punycode ASCII-compatible encoding, and the bidirectional algorithm is intended to follow the same design as presently specified.

Further Reading

It is possible to reference an overwhelming amount of commentary on this topic, so I have deliberately kept this list of further reading on the topic of IDNs relatively brief:

[A] John Klensin, "Internationalizing Top-Level Domain Names: Another Look," ISOC Member Briefing, September 2004, **http://www.isoc.org/briefings/018/**

[B] John Klensin, "National and Local Characters for DNS Top Level Domain (TLD) Names," RFC 4185, October 2005.

[C] Papers submitted to the ICANN IDN TLD workshop, held in November 2005: **http://www.icann.org/announcements/announcement-17nov05.htm**

[D] Internet Architecture Board, "Review and Recommendations for Internationalized Domain Names (IDNs)," RFC 4690, September 2006.

[E] "ICANN's IDN Roadmap Announcement—Progress and Future," **http://www.icann.org/announcements/announcement-1-01nov06.htm**

[F] "An Important Step Toward the Implementation of IDN Top-Level Domains: New Versions of IDNA Protocol Revision Proposals Posted," **http://www.icann.org/announcements/announcement-26nov07.htm**

[G] ICANN's IDN Evaluation Gateway. Eleven new internationalized domains representing the name **example.test** entirely in scripts other than the Latin characters: **http://idn.icann.org/**

References

[1] **http://en.wikipedia.org/wiki/Horizontal_and_vertical_writing_in_East_Asian_scripts**

[2] **http://en.wikipedia.org/wiki/Roman_script**

[3] **http://unicode.org**

[4] **http://www.omniglot.com/writing/thai.htm**

[5] Mockapetris, P., "Domain Names—Implementation and Specification," RFC 1035, November 1987.

[6] Hoffman, P., and Blanchet, M., "Preparation of Internationalized Strings ("stringprep")," RFC 3454, December 2002.

[7] Hoffman, P., Fältström, P., and Costello, A., "Internationalizing Domain Names in Applications (IDNA)," RFC 3490, March 2003.

[8] Hoffman, P., and Blanchet, M., "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)," RFC 3491, March 2003.

[9] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)," RFC 3492, March 2003.

[10] **http://en.wikipedia.org/wiki/Punycode**

[11] Crawford, M., "Non-Terminal DNS Name Redirection," RFC 2672, August 1999.

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. The author of numerous Internet-related books, he is currently the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of the Internet Society from 1992 until 2001. E-mail: **gih@apnic.net**

The Locator Identifier Separation Protocol (LISP)

by David Meyer, Cisco Systems

The *Internet Architecture Board's* (IAB)'s October 2006 *Routing and Addressing Workshop*[8] renewed interest in the design of a scalable routing and addressing architecture for the Internet. Many concerns prompted this renewed interest, including the scalability of the routing system and the impending exhaustion of the IPv4 address space. Since the IAB workshop, several proposals have emerged that attempt to address the concerns expressed both at the workshop and in other forums[7,9,12,13,14]. All of these proposals are based on a common concept: the separation of locator and identifier in the numbering of Internet devices, often termed the "Loc/ID split." This article focuses on one proposal for implementing this concept: the *Locator/Identifier Separation Protocol* (LISP)[3].

The basic idea behind the Loc/ID split is that the current Internet routing and addressing architecture combines two functions: *Routing Locators* (RLOCs), which describe how a device is attached to the network, and *Endpoint Identifiers* (EIDs), which define "who" the device is, in a single numbering space, the IP address. Proponents of the Loc/ID split argue that this "overloading" of functions makes it virtually impossible to build an efficient routing system without forcing unacceptable constraints on end-system use of addresses. Splitting these functions apart by using different numbering spaces for EIDs and RLOCs yields several advantages, including improved scalability of the routing system through greater aggregation of RLOCs. To achieve this aggregation, we must allocate RLOCs in a way that is congruent with the topology of the network ("Rekhter's Law"). Today's "provider-allocated" IP address space is an example of such an allocation scheme. EIDs, on the other hand, are typically allocated along organizational boundaries. Because the network topology and organizational hierarchies are rarely congruent, it is difficult (if not impossible) to make a single numbering space efficiently serve both purposes without imposing unacceptable constraints (such as requiring renumbering upon provider changes) on the use of that space.

LISP, as a specific instance of the Loc/ID split, aims to decouple location and identity. This decoupling will facilitate improved aggregation of the RLOC space, implement persistent identity in the EID space, and, in some cases, increase the security and efficiency of network mobility.

Implementing the Locator/ID Separation

There are two basic approaches to implementing the Loc/ID split: *map-and-encap* and *address rewriting*. Each is briefly discussed in the following sections.

Map-and-encap

In the map-and-encap scheme (genenerally considered to have evolved from Bob Hinden's ENCAPS protocol[24]), when a source sends a packet to the EID of a destination outside of the source domain, the packet traverses the domain infrastructure to a border router (or other border element). The border router maps the destination EID to a RLOC that corresponds to an entry point in the destination domain (hence an EID-to-RLOC mapping system is needed; proposals are discussed later in the article). This phase is the "map" phase of map-and-encap. The border router then encapsulates the packet and sets the destination address to the RLOC returned by the mapping infrastructure (if any; it may be statically configured as well). This phase is the "encap" phase of the map-and-encap model.

Thus map-and-encap works by appending a new header to the existing packet; the "inner-header" source and destination addresses are EIDs, and the "outer-header" source and destination addresses are in most cases RLOCs. When an encapsulated packet arrives at the destination border router, the router decapsulates the packet and sends it on to its destination. Note that this process suggests that EIDs may need to be routable in some scope (likely scoped to the domain).

Map-and-encap schemes have the desirable property that they do not in general require host changes or changes to the core routing infrastructure. In addition, map-and-encap schemes work with both IPv4 and IPv6, and retain the original source address (a feature that is useful in various filtering scenarios). Controversy remains, however, as to whether or not the encapsulation overhead of map-and-encap schemes is problematic; opinions exist on both sides of this topic (see, for example, [18]).

Address Rewriting

The basic idea behind the address-rewriting schemes, originally proposed by Dave Clark and later by Mike O'Dell in his 8+8/GSE specification[11], is to take advantage of the 128-bit IPv6 address and use the top 64 bits as the routing locator ("Routing Goop," or RG), and the lower 64 bits as the endpoint identifier (hence rewriting works only for IPv6). In this scheme, when a host emits a packet destined for another domain, the source address contains its identifier (frequently a IEEE MAC address) in the lower 64 bits, and a special value (meaning unspecified) in the RG. The destination address contains the fully specified destination address (RG and EID).

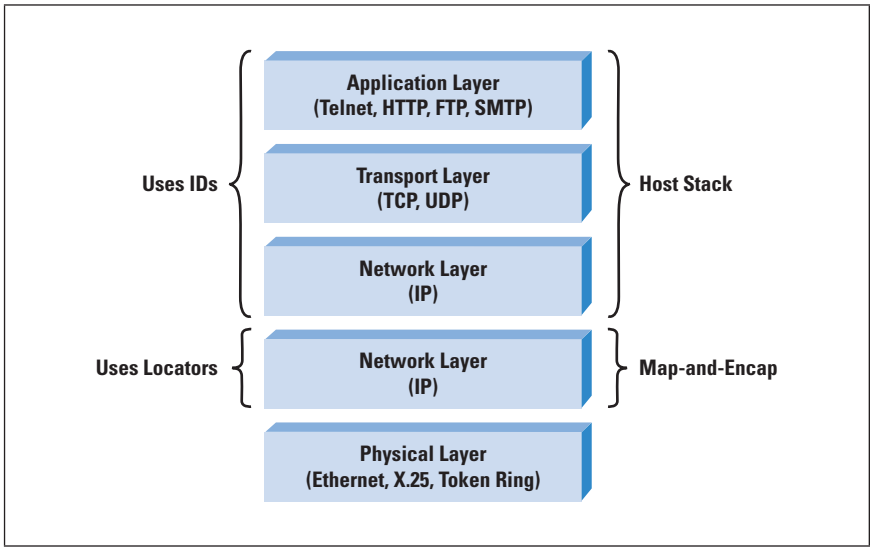
When a packet destined for a remote domain arrives at the local domain egress router, the source RG is filled in (forming a full 128-bit address), and the packet is routed to the remote domain. On ingress to the remote domain, the destination RG is rewritten with the unspecified value, ensuring that the host does not know what its RG is.

This process, in theory, would enable the ease of renumbering that would be required to maintain congruence between prefix assignment and physical network topology that is required for the kind of "aggressive" renumbering envisioned in the 8+8/GSE specification.

The Locator/Identifier Separation Protocol (LISP)

LISP is designed to be a simple, incremental, network-based map-and-encap protocol that implements separation of Internet addresses into EIDs and RLOCs. Because LISP is a map-and-encap protocol, it requires no changes to host stacks and no major changes to existing database infrastructures. It is designed to be implemented in a relatively small number of routers. LISP is also an instance of what is architecturally called a "jack-up," because the existing network layer is "jacked up" and a new network layer is inserted below it (the term "jacked up" is attributed to Noel Chiappa). The LISP jack-up is depicted in Figure 1.

Figure 1: LISP is a Jack-Up



The LISP design aims to improve site multihoming (for example, by controlling site ingress without complex protocols), improve *Internet Service Provider* (ISP) multihoming, decouple site addressing from provider addressing, and reduce the size and dynamic properties of the core routing tables.

The LISP data plane (the map-and-encap operation) and the LISP control plane (the EID-to-RLOC mapping system) are very modular. In particular, although the base LISP specification defines the format of messages to query the mapping system and to receive responses from that system, it makes no assumptions on the architecture of potential mapping systems. As a result, several mapping systems have been proposed[0,1,4,5,6,10].

LISP Network Elements

The LISP specification defines two network elements: The Egress Tunnel Router (ETR) and the Ingress Tunnel Router (ITR).

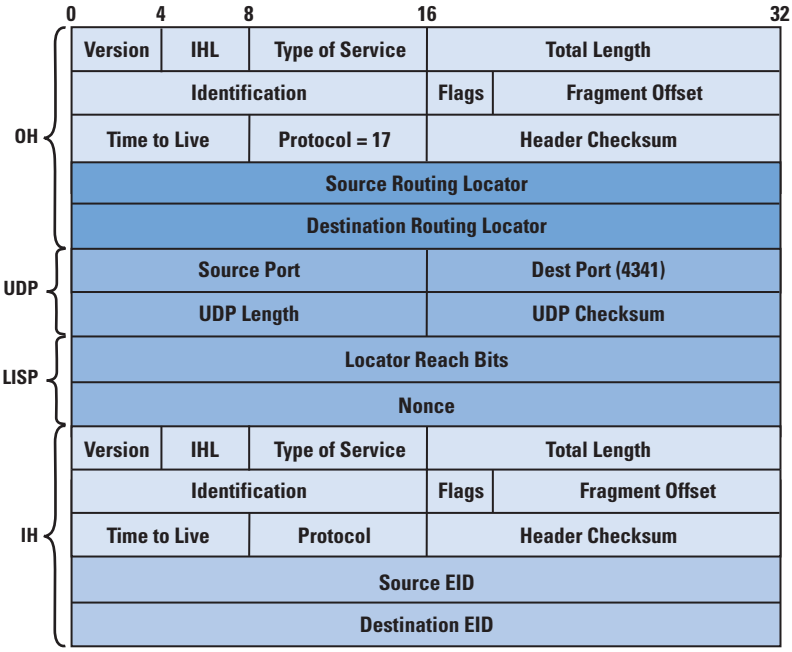
A LISP *Egress Tunnel Router* (ETR) receives LISP-encapsulated IP packets from the Internet on one side and sends decapsulated IP packets to site end systems on the other side. In particular, an ETR accepts an IP packet where the destination address in the "outer" IP header is one of its own RLOCs. The router strips the "outer" header and forwards the packet based on the next IP header found.

A LISP *Ingress Tunnel Router* (ITR) accepts IP packets from site end systems on one side and sends LISP-encapsulated IP packets toward the Internet on the other side. In particular, an ITR accepts an IP packet with a single IP header (more precisely, an IP packet that does not contain a LISP header). The router treats this "inner" IP destination address as an EID and performs an EID-to-RLOC mapping lookup if necessary (that is, it does not already have an EID-to-RLOC mapping for the EID). The router then prepends an "outer" IP header with one of its globally routable RLOCs in the Source Address field and the result of the mapping lookup in the Destination Address field. Note that this destination RLOC may be an intermediate, proxy device that has better knowledge of the EID-to-RLOC mapping closest to the destination EID.

LISP Data-Plane Operation

When a host in a LISP-capable domain emits a packet, it puts its EID in the packet source address, and EID of the correspondent host in its destination address (note that hosts will typically look up EIDs in the *Domain Name System* [DNS]). If the destination of the packet is in another domain, the packet traverses the source domain infrastructure to one of its ITRs. The ITR maps destination EID to a RLOC that corresponds to an ETR that is either in the destination domain or a proxy for the destination domain (how this mapping is accomplished in LISP is discussed later in the article). The ITR then encapsulates the packet, setting the destination address to the RLOC of the ETR returned by the mapping infrastructure or by static configuration. Note that LISP is address family-agnostic and as such can be used with both IPv4 and IPv6 (or any other address family). Figure 2 depicts the LISP IPv4 in IPv4 encapsulation.

Figure 2: LISP Header Format



When the packet arrives at the destination ETR, it decapsulates the packet and sends it on to its destination. Again, note that this scenario implies that EIDs need to be routable in some scope (likely scoped to the domain).

As mentioned previously, the LISP specification defines three packet types designed to support an EID-to-RLOC mapping system. The first type of packet, the *Data Probe*, is a data packet that an ITR may send into the mapping system to probe for the mapping; the authoritative ETR responds to the ITR with a Map-Reply message when it receives such a data packet. Note that in this case the ETR detects that the packet is a Data Probe by noticing that the inner *Destination Address* (DA) was copied to the outer DA by the ITR, that is, the inner DA equals the outer DA and is an EID. The second type of LISP packet used to support the mapping system is the *Map Request*. An ITR may query the mapping system by sending a Map-Request message into the mapping system to request a particular EID-to-RLOC mapping. As in the Data Probe case, the authoritative ETR responds with a Map-Reply message.

The third type of LISP packet used to support the mapping system is the *Map Reply*. An ETR emits a Map Reply under two conditions. First, if the ETR receives a LISP-encapsulated packet in which the outer-header destination address is the same as that of the inner header, it knows that the packet is a Data Probe and can respond with a Map Reply to the source ITR. The ETR may also receive a Map Request, in which case it replies to the requesting ITR with the mapping.

LISP Control Plane

Both map-and-encap and address-rewriting models rely on an additional of level of indirection in the addressing architecture to make the routing system scale reasonably. Because packets are sourced with an EID in the Destination Address field and EIDs are not in general routable on the global Internet, the destination EID must be mapped to an RLOC in order to deliver the packet to another domain (that is, across the Internet). In the case of the map-and-encap schemes, it is a direct translation: an EID is mapped to a RLOC. The situation is subtly different for the rewriting schemes; in general such schemes must look up the entire destination address (usually proposed to reside in the DNS)[11,13], but must somehow determine the source RG when rewriting the source address at the domain border.

In either Loc/ID split model, an EID-to-RLOC mapping service is needed to make the system scale reasonably and to make it operationally viable. There are three important scale parameters to consider when architecting a mapping service: the rate of updates to the mapping database, the state of the mapping service required, and the latency incurred during database lookup. The scaling properties of the database are frequently characterized as a (*Rate* × *State*) problem (ignoring for the moment the subject of lookup latency); because most estimates put the size of the mapping database at $O(10^{10})$, the database update rate must be small (note that this situation is a primary reason that current mapping proposals do not incorporate reachability information into the mapping database). In addition, the choice of push vs. pull also affects latency: if you push the entire database close to the edge, you improve lookup latency at the cost of increased state; if you architect a service that requires a mapping request and you find an authoritative server for that mapping (that is, pull), you reduce state at the cost of increased lookup latency.

LISP-Alternative-Topology: A LISP Control Plane

The basic idea behind *LISP-Alternative-Topology* (LISP-ALT)[4] is to build an alternative logical topology for managing EID-to-RLOC mappings for LISP. This logical topology uses existing technology and tools, specifically the *Border Gateway Protocol* (BGP)[17] and its multiprotocol extension[15], along with the *Generic Routing Encapsulation* (GRE)[16] protocol to construct an overlay network of devices that advertise EID prefixes only.

As was the case for the LISP data plane, an important design goal of LISP-ALT is to minimize the number of changes to existing hardware and software that are required to deploy the mapping system. Therefore, LISP-ALT requires modifications to neither BGP nor GRE.

Note that LISP-ALT is a hybrid push/pull architecture. Aggregated EID prefixes are "pushed" among the LISP-ALT routers and, optionally, to ITRs (which may elect to receive the aggregated information, as opposed to simply using a default mapping). Specific EID-to-RLOC mappings are "pulled" by ITRs either by Map Requests or Data Probes, both of which are routed over the alternate topology and result in Map Replies being generated by ETRs.

The basic idea behind in LISP-ALT, then, is to use BGP running over a GRE overlay to build the reachability required to route Data Probes, Map Requests, and Map Replies over the alternate topology. The ALT *Routing Information Base* (RIB) comprises EID prefixes and associated next hops. The LISP-ALT routers talk *External BGP* (eBGP) to each other in order to propagate EID prefix update information, which is learned either over eBGP connections from the authoritative ETR or by configuration. ITRs may also eBGP peer with one or more LISP-ALT routers in order to route Data Probe packets or Map Requests.

In summary, the LISP-ALT uses BGP to propagate EID-prefix reachability information used by ITRs and ETRs to forward Map Requests, Map Replies, and Data Probes. This reachability is carried as IPv4 or IPv6 *Network Layer Reachability Information* (NLRI) without modification (because the EID space has the same syntax as IPv4 or IPv6). LISP-ALT routers eBGP peer with one another, forming the overlay network. A LISP-ALT router near the edge learns EID prefixes that originate with authoritative ETRs. In general then, LISP-ALT routers aggregate EID prefixes, and forward Data Probes, Map-Requests, and Map-Replies.

Threat Models and Mitigation

As in any Loc/ID split approach, a critical operation is the creation of locator-to-ID binding state that devices will use over time. In the case of LISP, the critical operation is the creation of EID-to-RLOC mappings in the ITR and the ETR. We can obtain these mappings in three ways:

- By using the information obtained from a LISP data packet
- By using the information contained in the Map-Reply message
- By using an EID-to-RLOC mapping database

LISP mitigates attacks on the first two techniques by including a *nonce* in the LISP header; the nonce is a 32-bit randomly generated number (generated by the source ITR) that is used to test route returnability.

More specifically, an ETR echoes the nonce back to the ITR in a Map-Reply message. That is, the nonce, combined with the ITR accepting only solicited Map Replies, provides a base level of authentication for Map Replies. Note however, that these techniques do not protect against man-in-the-middle attacks.

The LISP design assumes that many (if not most) security mechanisms are part of the mapping database service when using control-plane procedures for obtaining EID-to-RLOC mappings. *Denial-of-Service* (DoS) attack prevention, on the other hand, depends on the ability of an implementation to rate-limit Map Requests and Map Replies (in the control plane), as well as its ability to rate limit the number of data-triggered Map Replies (for example, in response to Data Probe packets).

Refer to [19] for a more detailed preliminary threat analysis for LISP.

LISP and Fast Endpoint Mobility

Fast endpoint mobility occurs when an endpoint moves relatively rapidly, changing its IP layer network attachment point, and maintenance of session continuity is a goal. Mobile IPv4[20] and Mobile IPv6[21,22,27] mechanisms can be used in this case; note however, that the interaction of Mobile IP with LISP needs further exploration. Refer to the LISP specification[3] for additional details.

In summary, the major problem introduced by a Loc/ID split scheme is that as an endpoint moves, changes to the mapping between its EID and a set of RLOCs for its new network location may be required. When this change is added to the overhead of mobile IP binding updates, some packets might be delayed or dropped. In general, the problem is controlling the update rate (that is, the [*Rate* × *State*] product described previously), and is an area of ongoing research.

Multicast

A multicast group address, as defined in the original Internet architecture, is an identifier of a grouping of topologically independent receiver host locations. The address encoding itself does not determine the location of the receiver(s). The multicast routing protocol and the network-based state the protocol creates determine the location of the receivers.

In the LISP context, a multicast group address is both an EID and a RLOC. As such, no specific action is necessary for destination addresses; a group address that appears in an inner IP header (built by a source host) is used as the destination EID by an ITR as a destination address when it LISP-encapsulates the packet (that is, the ITR uses the same group address as the destination RLOC).

The source RLOC, as is usually the case, is the ITR IP address (that is, one of its RLOCs).

At the receiving side, *Protocol Independent Multicast* (PIM)[23] has to translate the source-address Join/Prune messages from RLOCs to EIDs when multicast packets are forwarded by the ETR. However, in contrast to the unicast case (where a Map Request is sent by the ITR at forwarding time), a Map Request can be sent when the multicast tree is being built.

Putting It All Together: A Day in the Life of a LISP Packet

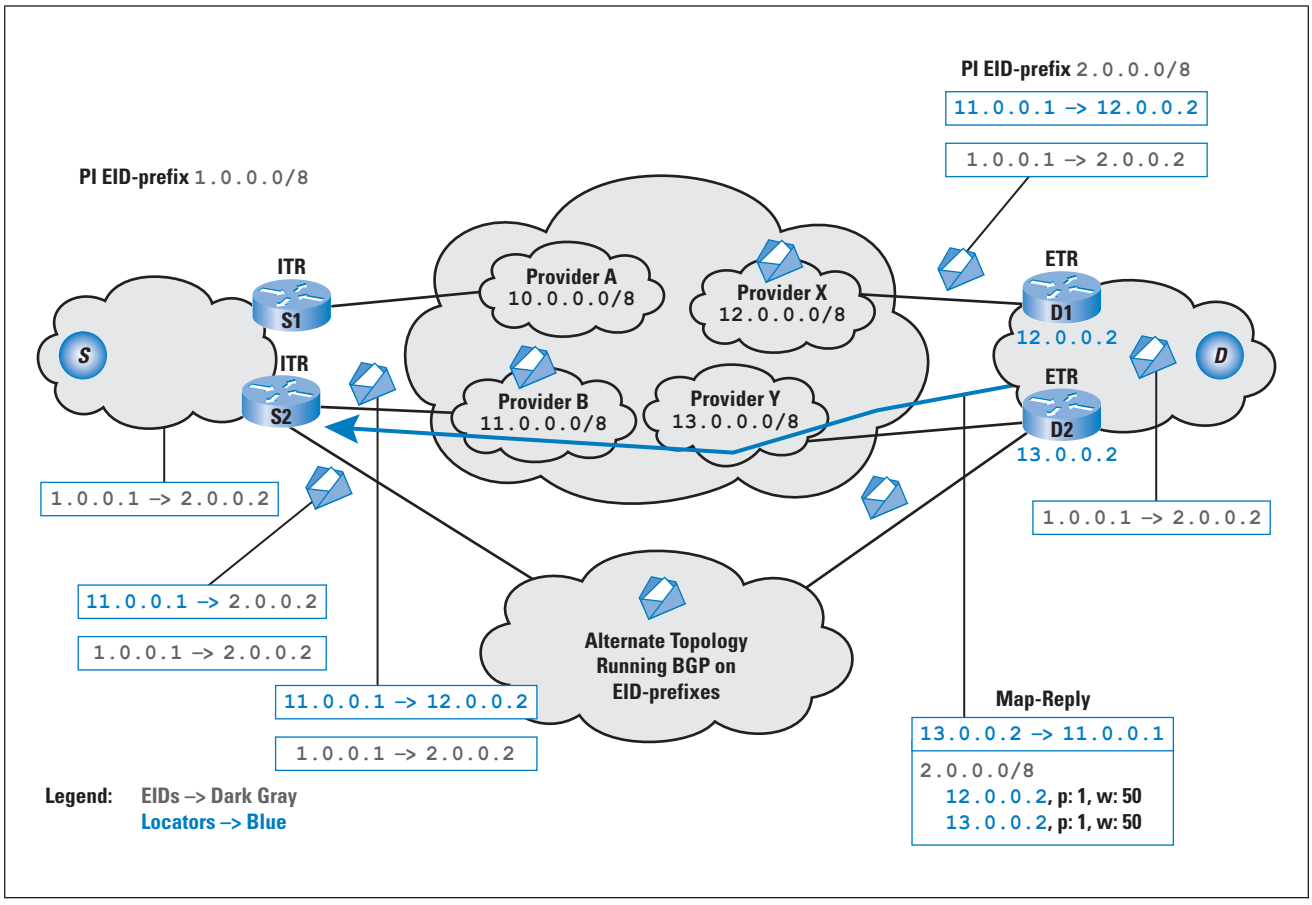
When a host in a LISP-capable domain wants to send a packet, it first looks up the correspondent host's EID in the DNS. It then puts its EID in the packet source address, and EID of the correspondent host in its destination address; if the destination of the packet is in another domain, the packet traverses the source domain infrastructure to one of the domain ITRs.

If the ITR has cached the EID-to-RLOC mapping for the destination EID, it sets the destination RLOC in the outer (encapsulated) header to the cached RLOC, and the source RLOC to its RLOC (note that the inner header has the source host's EID as the source and the destination's EID in the Destination field). The packet is then sent over the Internet to the ETR indicated in the destination RLOC, which decapsulates the packet and sends it on to the destination EID.

If, on the other hand, the ITR does not have a EID-to-RLOC mapping for the destination EID, it encapsulates the packet in a LISP header in which the destination address is the same as the inner header destination address, namely, the EID of the destination host. This packet is a Data Probe packet, and is routed over the LISP-ALT topology to the LISP-ALT router (typically an ETR, but this type of router is not required) that is authoritative for the EID-to-RLOC mapping. When the ETR receives the Data Probe packet, it decapsulates the packet and sends it on to the destination EID and sends a Map Reply to the source ITR so subsequent packets are sent natively over the Internet (as opposed to over the LISP-ALT overlay network). This query/response transaction is required only for the first packet sent between sites; all subsequent packets are sent LISP-encapsulated directly between the ITR and the ETR (and in particular, not over the LISP-ALT topology). Finally, note that the ITR could also preload its cache with mappings for popular destinations using the Map-Request message, avoiding the Data Probe packet (and associated latency, if any) altogether.

For example, consider the scenario depicted in Figure 3. In this case, a source *S* with EID `1.0.0.1` wants to send a packet to destination *D* whose EID is `2.0.0.2`. The packet arrives at ITR S2, which does not have an EID-to-RLOC mapping for `2.0.0.2`. S2 LISP-encapsulates the packet with the outer header having its RLOC (`11.0.0.1`) as the source address, copies the destination EID (`2.0.0.2`) from the inner header to the outer-header destination, and sends the data packet (a Data Probe) into the LISP-ALT topology. The packet follows the paths computed by BGP in the LISP-ALT topology to ETR D2. When D2 receives the packet, it decapsulates it and forwards the packet to the destination `2.0.0.2`; D2 also responds with a Map-Reply message that tells S2 (`11.0.0.1`) that the EID-to-RLOC mapping for `2.0.0.0/8` has two elements, ETR D1 (whose RLOC is `12.0.0.2`) and ETR D2 (whose RLOC is `13.0.0.2`). After receiving the Map Reply, ITR S2 can send LISP-encapsulated packets natively over the Internet (that is, not over the ALT topology).

Figure 3: A Day in the Life of a LISP Packet



Note that the mapping has priority (p) and weight (w) attributes. Priorities tell the ITR which ETRs to use in which order, and weights tell the ITR how to split load across ETRs of a given priority (w is a percentage of traffic that should go to each ETR). In this case, both ETRs have the same priority (1), and have weight 50 (that is, each ETR should receive 50 percent of the traffic).

New Functions Enabled by the Mapping System

Weights and priorities provide new capabilities for multihomed sites, which can use these features to control how traffic ingressing to the site is spread across its links without the complexity and overhead of running BGP. In particular, a multihomed site can configure its mapping database so that its links are used in an "active-active" configuration (that is, both links are in use). This situation is depicted in Figure 3, where the mapping databases entry `2.0.0.0/8` has two ETRs at the same priority that are equally weighted, meaning that the ITR will spread flows equally among the two ETRs.

This function is particularly attractive for *Small Office or Home Office* (SOHO) sites that desire both redundancy in their Internet connections and the ability to easily load share across those links in an active-active configuration, without the complexity and operational expense of running BGP.

Another interesting functionality enabled by the LISP control plane is the ability to mitagate some types of DoS attacks. In particular, if an ETR notices that it the subject of a DoS attack from behind an ITR (that is, DoS packets are destined to an EID-prefix for which it is authorative), it can use the LISP locator reachability bits (see Figure 2) to tell the the source ITR that the RLOC for that EID-prefix is not available. The ETR accomplishes this by sending a locator-reachability bit of zero for the RLOC to the offending ITR. Note that this functionality is similar to Ioannidis and Bellovin's "ICMP Pushback" proposal[25].

Performance Considerations

LISP and its associated mapping protocol(s) have two primary performance concerns:

- Encapsulation overhead
- EID-to-RLOC lookup latency and packet loss

In the case of encapsulation overhead, the concern is that the addition of the LISP header will cause the encapsulate packet to exceed the path *Maximum Transmission Unit* (MTU). As mentioned previously, this area of research is still active (see, for example, [18]).

In the case of lookup latency and packet loss, because LISP-ALT uses BGP to find a particular EID-to-RLOC mapping, there could be latency associated with the first few packets in the first flow between sites (note that it is only the first flow; subsequent flows can use the mapping installed in the ITR). However, this latency is mitigated, and the initial packets are not lost because LISP can send the first few data packets over the control plane; these packets are the Data Probe packets. There is additional latency associated with the time required for the destination ETR to return the Map Reply. However, after this initial transaction is completed, no additional latency is injected by the mapping system.

As mentioned previously, there is a trade-off in the mapping system among the state required to be held by network elements, the rate of updates to the mapping system, and the latency incurred when looking up an EID-to-RLOC mapping. LISP-ALT is a hybrid (push/pull) architecture that attempts to minimize the state requirements on ITRs, while at the same time minimizing lookup latency.

Conclusions

LISP is a new protocol that implements the Loc/ID split using a map-and-encap protocol. It obtains the advantages of the level of indirection afforded by the Loc/ID split while minimizing changes to hosts and to the core routing system. In addition, LISP enables new functions such as BGP-free multihoming in an active-active configuration.

Acknowledgments

The LISP specification and supporting documents are the work of many people, including Scott Brim, Noel Chiappa, Dino Farinacci, Vince Fuller, Eliot Lear, Darrel Lewis, and Dave Oran.

References

[0] Brim, S., et al., "EID Mappings Multicast Across Cooperating Systems for LISP," Internet Draft, Work in Progress, **draft-curran-lisp-emacs-00.txt**

[1] Brim, S., et al., "LISP-CONS: A Content distribution Overlay Network Service for LISP," Internet Draft, Work in Progress, **draft-meyer-lisp-cons-03.txt**

[2] Chiappa, N., "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture," **http://ana.lcs.mit.edu/~jnc//tech/endpoints.txt**

[3] Farinacci, D., et al., "Locator/ID Separation Protocol (LISP)," Internet Draft, Work in Progress, **draft-farinacci-lisp-06.txt**

[4] Fuller, V., et al., "LISP Alternative Topology (LISP-ALT)," Internet Draft, Work in Progress, **draft-fuller-lisp-alt-01.txt**

[5] Jen, D., et al., "APT: A Practical Transit Mapping Service," Internet Draft, Work in Progress, **draft-jen-apt-01.txt**

[6] Lear, E., "NERD: A Not-so-Novel EID to RLOC Database," Internet Draft, Work in Progress, **draft-lear-lisp-nerd-03.txt**

[7] Massey, D., Wang, L., Zhang, B., and L. Zhang, "A Proposal for Scalable Internet Routing and Addressing," Internet Draft, Work in Progress, **draft-wang-ietf-efit-01.txt**

[8] Meyer, D., et al., "Report from the IAB Workshop on Routing and Addressing," RFC 4984, September 2007.

[9] Narten, T., et al., "Routing and Addressing Problem Statement," Internet Draft, Work in Progress, **draft-narten-radir-problem-statement-01.txt**

[10] Nordmark, E., "Shim6: Level 3 Multihoming Shim Protocol for IPv6," Internet Draft, Work in Progress, **draft-ietf-shim6-proto-09.txt**

[11] O'Dell, M., "GSE - An Alternate Addressing Architecture for IPv6," **http://www.watersprings.org/pub/id/draft-ietf-ipngwg-gseaddr-00.txt**

[12] Templin, F., "The IPvLX Architecture," Internet Draft, Work in Progress, **draft-templin-ipvlx-08.txt**

[13] Vogt, C., "Six/One: A Solution for Routing and Addressing in IPv6," Internet Draft, Work in Progress, **draft-vogt-rrg-six-one-01.txt**

[14] Whittle, R., "Ivip (Internet Vastly Improved Plumbing) Architecture," Internet Draft, Work in Progress, **draft-whittle-ivip-arch-01.txt**

[15] Bates, T., et al., "Multiprotocol Extensions for BGP-4," RFC 2858, June 2000.

[16] Farinacci, D., et al., "Generic Routing Encapsulation (GRE)," RFC 2784, March 2000.

[17] Rekhter, Y., (Ed.), et al., "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, January 2006.

[18] Templin, F., "Subnetwork Encapsulation and Adaptation Layer," Internet Draft, Work in Progress, **draft-templin-seal-02.txt**

[19] Bagnulo, M., "Preliminary LISP Threat Analysis," Internet Draft, Work in Progress, **draft-bagnulo-lisp-threat-01.txt**

[20] Perkins, C., "IP Mobility Support for IPv4, revised," Internet Draft, Work in Progress, **draft-ietf-mip4-rfc3344bis-05.txt**

[21] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6," RFC 3775, June 2004.

[22] Arkko, J., Vogt, C., and W. Haddad, "Enhanced Route Optimization for Mobile IPv6," RFC 4866, May 2007.

[23] Fenner, B., et al., "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601, August 2006.

[24] Hinden, R., "New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG," RFC 1955, June 1996.

[25] Ioannidis John, and Bellovin, S., "Pushback: Router-Based Defense Against DDoS Attacks," **http://citeseer.ist.psu.edu/420554.html**

[26] Huston, G., "More ROAP: Routing and Addressing at IETF68," *The Internet Protocol Journal,* Volume 10, No. 2, June 2007.

[27] Carlos J. Bernardos, Ignacio Soto, and María Calderón, "IPv6 Network Mobility," *The Internet Protocol Journal,* Volume 10, No. 2, June 2007.

DAVID MEYER is currently a Director in the Advanced Research and Technologies Group at Cisco Systems, where he works on future directions for Internet technologies. E-mail: **dmm@cisco.com**

Book Review

Patterns in Network Architecture

Patterns in Network Architecture: A Return to Fundamentals, by John Day, ISBN-10: 0132252422, ISBN-13: 9780132252423, Prentice Hall, 2007. **http://www.informit.com/store/product.aspx?isbn=0132252422**

It isn't every day (pun intended) that one of the true Old Guard writes and publishes a book, and it behooves us to take notice. In this case, the author's expertise and his subject matter are of particular timeliness, because of the worldwide resurgence of activities with regard to next-generation network architectures, that is, a replacement, or upgrade to the Internet (dare one say "Internet 2.0"?).

John Day is a well-known scholar of historical cartography, and this book, in a way, is a roadmap of network architecture. The roadmap starts back in 1970, tracing from the roots of connectionless packet-switched dynamically routed systems such as Cyclades, and the ARPANET, through to recent discussions on multihoming, multicast, and mobility, with a view along the way of naming, addressing, protocol stack design, protocol design, and concepts of layering.

That description makes the book sound fairly standard in terms of structure and content, but it isn't. The book includes many discursive elements whose intent is to provide a collection of *patterns*. Design patterns originated in the building trade as a way for crafts people to pass on successful methods of construction (in the sense of affordable and noncollapsing) to less-inventive people (or people who want to spend their inventive efforts in different areas). Software engineers picked up on this idea, applying the techniques in both the microscopic world: patterns allow you to decide what algorithm is applicable in solving a problem in the small; and the macroscopic world: architectural patterns allow you to decide on an approach to breaking down a large system into the right kind of components.

Essentially, this book does the same thing, at the protocol stack level, and at the system level, with a collection of historical and contemporary examples to support the arguments.

The book makes interesting reading, especially as it represents a fair balance in reporting the early ideas that came not just from the United States, and restates the importance of the *Opens Systems Interconnection* (OSI) model (not the ISO protocols) in understanding layering and beads-on-a-string, as well as reasserting the use of the model in clarifying the perennially confusing concepts of names, addresses, and routes.

The book begins with a discussion of seven principles that emerged through the early history of networking (I won't spoil the book for readers by listing them here), and ends in the tenth and final chapter, entitled "Backing Out of a Blind Alley," with an appeal to fundamentals. Essentially, the author points out that researchers (especially academics) are strongly motivated to keep moving on with claims of ever-newer tricks, but rarely to consolidate these tricks into a set of principles that stand for a long time (because then they would have to completely change the topic of their research). Thus uncovering a foundational theory of networking would put a whole generation of networkers out of work (or funding at least).

The book is peppered (saltily) with fine quotes and fascinating asides from philosophy (for this reader, especially, the Chinese diversions were most novel and illuminating). Illustrative of the range is that one finds Wittgenstein and Dave Clark, Confucius, and Dr. Seuss—Frege's useful reminder that "The sign '=' should be read as 'is easily confused with'" would make an excellent IETF T-shirt.

I found the book extremely readable and enjoyable, and although I might argue with some of the opinions in the book, I think that this is just more evidence that I should recommend the book to anyone interested in knowing why we are where we are in networking, and being better informed about where we should go next.

—Jon Crowcroft, University of Cambridge
`Jon.Crowcroft@cl.cam.ac.uk`

Read Any Good Books Lately?

Then why not share your thoughts with the readers of IPJ? We accept reviews of new titles, as well as some of the "networking classics." In some cases, we may be able to get a publisher to send you a book for review if you don't have access to it. Contact us at `ipj@cisco.com` for more information.

Fragments

ICANN Recovers Large Block of Internet Address Space

The *Internet Corporation for Assigned Names and Numbers* (ICANN) has found a little breathing room in the IPv4 address space with its recovery of a block of 16 million IPv4 addresses.

The IP addresses recovered were once used to connect older protocol packet-data networks with the fledgling Internet. The block of addresses, technically referred to as `14.0.0.0/8`, is also known as "Net-14."

"Net-14 was the easiest network to reclaim, the so-called low hanging fruit," said Barbara Roseman, General Manager with the *Internet Assigned Numbers Authority* (IANA), which is operated by ICANN. "None of the other legacy assignments in the IPv4 space are likely to be completely reclaimed as they are all in active use."

A small percentage of the addresses in Net-14 had been assigned, most more than 15 years ago. The assignments were so old that finding people who knew about them was a lengthy process. Nearly 50 organizations worked cooperatively with ICANN staff throughout 2007 to confirm that the 984 registrations were no longer in use. IANA undertook the reclamation effort to ensure that the greatest number of IPv4 addresses can be made available to Internet users as the overall free pool of IPv4 addresses is depleted. IANA allocates IPv4 and IPv6 addresses to *Regional Internet Registries* (RIRs). The five RIRs allocate addresses to network operators in their local regions. IANA allocated more than one /8 (16m IPv4 addresses) per month in 2007 and the rate of allocation is not expected to slow in 2008. The reclamation of Net-14 means there are now 43 unallocated /8s left.

"The recovery of these addresses offers some breathing room as the four billion addresses in IPv4 space are depleted, but it is only a temporary solution," added Roseman. "The real and lasting solution is the technical move to IPv6—the protocol that will make 340 trillion trillion trillion unique IP addresses available."

IPv6 Address Added for Root Servers in the Root Zone

ICANN recently took another step along the path of deployment for the next-generation IPv6 Internet addressing system. IPv6 addresses were added for six of the world's 13 *root server* networks (A, F, H, J, K, M) to the appropriate files and databases. This move allows for the possibility of fuller IPv6 usage of the *Domain Name System* (DNS). Prior to today, those using IPv6 had needed to retain the older IPv4 addressing system in order to be able to use domain names.

"The ISP community welcomes this development as part of the continuing evolution of the public Internet," said Tony Holmes, chair of ICANN's Internet Service and Connectivity Provider Constituency. "IPv6 will be an essential part our future and support in the root servers is essential to the growth, stability, and reliability of the public Internet."

Name server software relies on the root servers as a key part in translating domains like icann.org into the routing identifiers used by computers to connect to one another. In 2007 the ICANN *Security and Stability Advisory Committee* concluded that ICANN should move forward with the enhancement of the DNS root service by adding IPv6 addresses for the root servers."The addition of IPv6 addresses for the root servers enhances the end-to-end connectivity for IPv6 networks, and furthers the growth of the global interoperable Internet," added David Conrad, ICANN's Vice President of Research and IANA Strategy. "This is a major step forward for IPv6-only connectivity and the global migration to IPv6."

Further technical information on the move is available at:

`http://www.iana.org/reports/root-aaaa-announcement.html`

RIPE NCC Publishes Case Study of YouTube Hijack

As you may be aware from recent news reports, traffic to the `youtube.com` Website was "hijacked" on a global scale on Sunday February 24, 2008. The incident was a result of the unauthorized announcement of the prefix `208.65.153.0/24` and caused the popular video sharing Website to become unreachable from most, if not all, of the Internet. The RIPE NCC conducted an analysis into how this incident was seen and tracked by the RIPE NCC's *Routing Information Service* (RIS) and has published a case study at:

`http://www.ripe.net/news/study-youtube-hijacking.html`

The RIPE NCC RIS is a service that collects *Border Gateway Protocol* (BGP) routing information from roughly 600 peers at 16 *Internet Exchange Points* (IXPs) across the world. Data is stored in near real-time and can be instantly queried by anyone to provide multiple views of routing activity for any point in time. The RIS forms part of the RIPE NCC's suite of Information Services, which together provide a deeper insight into the workings of the Internet. The RIPE NCC is a neutral and impartial organization, and commercial interests therefore do not influence the data collected. The RIPE NCC Information Services suite also includes the *Test Traffic Measurement* (TTM) service, the *DNS Monitoring* (DNSMON) service and Hostcount. All of these services are available to anyone, and most of them are offered free of charge.

More information about RIPE NCC Information Services can be found at: `http://is-portal.ripe.net`

IETF Examines Future of the Internet by Going IPv6 Native

The *Internet Engineering Task Force* (IETF) put a spotlight on the next generation of Internet addressing when it switched off attendees' access to IPv4 during its March 2008 meeting. For an hour, Internet engineers at the meeting could only access the Internet using an IPv6 network.

During this event, IETF participants were encouraged to explore the Internet as it appears today in the IPv6 environment. The purpose of this exploration was to determine the next steps necessary toward deployment of IPv6 as the next generation of Internet addressing. The IETF undertook this activity at a time when IPv6-implementation is becoming a matter of global importance for the Internet. The event provided all IETF meeting attendees a first-hand opportunity to work with the Internet over an exclusive IPv6 network. "We get a lot of reports from members of our community who use IPv6, but this was an opportunity for everyone to observe and discuss the technical issues as a group," said Russ Housley, Chair of the IETF. "This first-hand data helps to inform our engineering decisions."

Some members of the Internet technical community assert that the ongoing deployment of IPv6 has been held back by a lack of IPv6-accessible Websites, creating the classic first-step dilemma for network operators. "It has been incredible to observe as members of the community organized themselves and updated their home networks to be ready for this event," said Leslie Daigle, Chief Internet Technology Officer at the Internet Society. "As we continue to solve the engineering and implementation obstacles to IPv6 deployment, creative engineers around the world will develop new uses for the Internet, through IPv6, in ways we can't yet imagine."

The IETF has provided dual stack IPv4/IPv6 network connectivity at its meetings for years, which has been useful for its regular IPv6-using attendees. The difference during this meeting was that a strictly IPv6 network was made available as well, and all attendees were encouraged to explore and experiment with the Internet as seen from IPv6. This focus was heightened when IPv4 access was deliberately shut off for an hour, leaving only IPv6 for connectivity. Following this—and other similar experiments—the engineering community expects to have a better understanding of the next steps necessary in the development of protocols and standards to support the continued deployment of IPv6 in support of the global Internet. The Comcast Corporation provided the facilities to conduct the live test of IPv6 and was the host sponsor of IETF-71 in Philadelphia.

For more information about this event, and similar events please see:

http://www.isoc.org/educpillar/resources/ipv6_faq.shtml

http://wiki.tools.isoc.org/IETF71_IPv4_Outage

http://www.civil-tongue.net/clusterf/

Postel Network Operator's Scholarship 2008

The *North American Network Operators' Group* (NANOG) and the *American Registry for Internet Numbers* (ARIN) have been unique and successful cooperative fora for Internet builders in North America and other parts of the world. Senior practitioners from around the world contribute their time to NANOG and ARIN as presenters, teachers and trainers, to produce consistent non-commercial conferences of high-quality.

Since 2007, the generosity of an anonymous donor and the administration of the Internet Society, have allowed NANOG and ARIN to provide financial support to a person from a developing country to participate in the October joint NANOG/ARIN meeting through the *Postel Network Operator's Scholarship*.

The Scholarship Committee cordially invites suitable applicants to apply for fellowship funding to participate in the October 2008 joint NANOG/ARIN meeting. The Scholarship targets personnel from developing countries who are actively involved in Internet development, in any of the following roles: Engineers (Network Builders), Operational and Infrastructure Support Personnel, and Educators, Teachers, and Trainers

Successful applicants will be provided with transportation to and from the meetings and a reasonable allowance for food and accommodation. In addition all fees for participation in the conferences, tutorials, and social events will be waived. Applicants from any part of the world will be considered. The deadline for application is June 1, 2008, and the awardee will be informed by July 1, 2008.

To apply for the fellowship please read **`http://www.nanog.org/postel-scholarship.html`** and submit your application by e-mail to **`PostelNOS@nanog.org`**

For more information about NANOG and ARIN meetings, see: **`http://www.nanog.org/`** and **`http://www.arin.net/`**

JPNIC Releases IPv4 Exhaustion Report

The *Japan Network Information Center* (JPNIC) has released a report entitled "Study Report on the IPv4 Address Space Exhaustion Issue (Phase I)." The report can be downloaded from the following link:

`http://www.nic.ad.jp/en/ip/ipv4pool/ipv4exh-report-071207-en.pdf`

Call for Papers

The Internet Protocol Journal (IPJ) is published quarterly by Cisco Systems. The journal is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. The journal carries tutorial articles ("What is...?"), as well as implementation/operation articles ("How to..."). It provides readers with technology and standardization updates for all levels of the protocol stack and serves as a forum for discussion of all aspects of internetworking.

Topics include, but are not limited to:

- Access and infrastructure technologies such as: ISDN, Gigabit Ethernet, SONET, ATM, xDSL, cable, fiber optics, satellite, wireless, and dial systems
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, fire-walls, troubleshooting, and mapping
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, network computing, and Quality of Service
- Application and end-user issues such as: e-mail, Web authoring, server technologies and systems, electronic commerce, and application management
- Legal, policy, and regulatory topics such as: copyright, content control, content liability, settlement charges, "modem tax," and trademark disputes in the context of internetworking

In addition to feature-length articles, IPJ will contain standardization updates, overviews of leading and bleeding-edge technologies, book reviews, announcements, opinion columns, and letters to the Editor.

Cisco will pay a stipend of US$1000 for published, feature-length articles. Author guidelines are available from Ole Jacobsen, the Editor and Publisher of IPJ, reachable via e-mail at `ole@cisco.com`

The Internet Protocol Journal is published quarterly by the Chief Technology Office, Cisco Systems, Inc.
www.cisco.com
Tel: +1 408 526-4000
E-mail: ipj@cisco.com

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

PRSRT STD
U.S. Postage
PAID
PERMIT No. 5187
SAN JOSE, CA