Internationalization of Domain Names: A history of technology development John C Klensin and Patrik Fältström	

First-generation Hostnames and Character Coding

Consideration of internationalization issues in the ARPANET/ Internet context goes back to the early 1970s. Some of the thinking that went into decisions about naming of hosts, "text" file contexts and data streams for transferring them, etc., was informed by considerations of languages other English and scripts that could not be represented in ASCII. The difficulty was that the available technology, for whatever reason, just was not up to the problem. The twelve "national use character positions" of ISO 646 Basic Version (BV)[1] came along later and were satisfactory only for small numbers of Roman-based characters in use in each country and then only for limited uses. For example, the square brackets ("[" and "]") fell within the national use positions and were hence replaced by national characters in several countries. But they were also special characters in some programming languages, so people writing in those programming languages in those countries needed to switch the interpretation of the codepoints back and forth depending on context. That never works well and often doesn't work at all: for example, since the contextual information is out of band, writing comments in the national language in a program that requires the brackets becomes almost impossible without using one set of printed characters in a peculiar way. Finally, because information about which particular set of national-used characters were intended was available only out of band, the whole picture was useless for international communications.

A little bit later, a number of national eight-bit character set standards came into being, typically organized as Latin-something, i.e., with control characters and ASCII approximations in the lower 128 code positions and the national characters (and sometimes additional control characters) in the upper 128 code positions. Many of those eight-bit character sets later evolved into the various parts of ISO 8859[2]. Competing coded character sets also came into being as IBM-specific, and later Microsoft-provided, code pages, many of them with characters in the lower 128 code positions that did not appear in ASCII or that were coded differently. All of those eight-bit sets shared the problem of the seven-bit ISO 646 BV: if one communicated across national boundaries, one needed some way to identify the character sets in use, not only within a coding family but between them[3]. Otherwise, bits were just bits, indistinguishable from other bits: there was no way to tell, in isolation, if a string of bits were intended to refer to an Arabic character, a Cyrillic one, or an Eastern European "Latin" one.

---

[1] International Organization for Standardization, "Information technology - ISO 7-bit coded character set for information interchange", ISO Standard 646, 1991.

[2] International Organization for Standardization, "Information processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1 (1998) - Part 2: Latin alphabet No. 2 (1999) - Part 3: Latin alphabet No. 3 (1999) - Part 4: Latin alphabet No. 4 (1998) - Part 5: Latin/Cyrillic alphabet (1999) - Part 6: Latin/Arabic alphabet (1999) - Part 7: Latin/Greek alphabet (2003) - Part 8: Latin/Hebrew alphabet (1999) - Part 9: Latin alphabet No. 5 (1999) - Part 10: Latin alphabet No. 6 (1998) - Part 11: Latin/Thai alphabet (2001) - Part 13: Latin alphabet No. 7 (1998) - Part 14: Latin alphabet No. 8 (Celtic) (1998) - Part 15: Latin alphabet No. 9 (1999) - Part 16: Part 16: Latin alphabet No. 10 (2001)", ISO Standard 8859, 2003.

[3] A good, practical, discussion of these issues and comparisons among some of the tables, appears in F. da Cruz and C. M. Gianone, *Using C-Kermit*, Digital Press, 1993.

Only when ISO 2022 and the notion of code table designation and switching came along[4] was it possible to identify, in-band, what character a particular code point was intended to represent. There were still two difficulties. The first was that the code table designation approach was problematic for bodies of text that needed several tables because the systems using it had to remember the last designation for an indefinite period (the Japanese developed a workaround to require the designation be renewed for every line[5], which limited the problem). Conversely, for short text, like a host name or DNS label, the general opinion was that including a code table designator at the beginning of every name or label was excessive overhead and, worse, would cause interoperability problems since not all systems would have the same sets of code tables available.

Most of those discussions were face to face and not recorded or, at least, the documents are not readily available. That history is important only to stress that the issues were not being ignored. The problem was that there was no character-coding technology available that would provide a foundation for addressing the problems. Even when appropriate national or language codings were available, their adoption would have induced interoperability difficulties: different character coding systems in use in different places with no standard way to communicate what was in use and to map between them. The result was that a number of bodies, including both the predecessors to IETF and the predecessor to ITU-T, made essentially the same decision: to restrict the names and identifiers to ASCII or its equivalents, ISO 646 IRV and ITU Recommendation T.50 (IA5)[6].

The Design of the Domain Name System

The DNS was designed to be able to accommodate "binary" labels and data –not only non-ASCII text, but non-character material. The provisions for doing that were built into the original specifications[7] and are present in all conforming implementations. While that feature could have provided a platform for some IDN approaches, it was underspecified in two important ways. First, case-mapping was defined for ASCII (at least within Class IN (see below)), but it was not clear what should be done about non-ASCII character data if the relevant script had case or position-in-word differences. Second, use of ASCII and non-ASCII labels within the same Class was not clearly specified. Debates about the correct resolution of these issues have continued nearly to the present day and implementations are not completely consistent about how they are interpreted[8].

The ability of the DNS to carry binary labels and data implied that, if all of the users in some community were using the same (or a compatible) DNS implementation and could agree on what conventions to use, it was possible to simply store non-ASCII information in the DNS. A number of early "IDN" (or "multilingual name") demonstrations took advantage of that fact. However, problems arose between communities, communities that, in practice, overlap rather than having precise boundaries: Different DNS implementations handled non-ASCII strings in different ways. Different communities and registries used different coded character sets or

---

[4] International Organization for Standardization, "Information Processing: ISO 7-bit and 8-bit coded character sets: Code extension techniques", ISO Standard 2022, 1986 For registration of new character sets and escape sequences see ISO/IEC 2375:2003 and http://www.itscj.ipsj.or.jp/ISO-IR/index.html

[5] This arrangement is used in the Shift-JIS form defined in Appendix 1 of JIS X0208:1997.

[6] ITU-T Rec. T.50 (1992) "International Reference Alphabet (IRA)" (Formerly International Alphabet No. 5 or IA5 "Information technology - 7-bit coded character set for information interchange").

[7] See RFC 1034 and RFC 1035.

[8] RFC 2181 was written to resolve many of those issues but it just touched off more controversy about a few of them. See RFC 4033 and RFC 2673 for examples.

made different assumptions about comparison. When Unicode, or other coding systems that could represent a given character in more than one way was used, different implementations made different assumptions about the comparison process. And, of course, many implementations of applications that expected domain names to conform to the original "hostname" standards would reject non-ASCII names even if there were no difficulties in the DNS itself. So there was considerable potential for confusion and systems that would not interoperate or, in some cases, scale to the size and demands of the public Internet.

Standardizing Internationalized Content

The IETF standardized a mechanism for transmitting and processing email messages with non-ASCII characters in the early 1990s. The desire for a standard way to transport and identify those messages and the character sets being used was one of the key reasons the MIME effort was initiated[9]. For text, MIME is permissive: it identifies the character set or code page being used but does not specify conversions among types. However, unlike predecessor systems, the information about what is being used is transmitted in-band (with the message itself) in a way that does not require state preservation. The domain name, email address local part, and mail header internationalization issues were discussed at that time. But, with the exception of some parts of headers (those that did not carry any protocol information), explicit labels for character coding were judged to be inappropriate and Unicode just was not sufficiently ready. The MIME system for identifying the coding systems in use was adopted for use in the Web although HTML was defined as ISO 8859-1 compatible initially, with UTF-8 coming along with HTML 3.2[10]

Internationalizing Domain Names

When the IDN question came up again some years later, it was clear that standardization was required to avoid the use of different conventions in different communities without clear identification or other ways to achieve interoperability. The IETF IDN WG was formed, initially as a study effort and then as a development one. As part of that effort, a large family of possible paths were considered. Some, such as the original i-DNS.net proposal and some ideas about using phonetic, rather than script-based, approaches, were rejected on technical grounds: the conclusion was that they were unworkable on a global Internet or undeployable in an environment with a running DNS with extensively-deployed implementations and applications.. A number of approaches other than the one that evolved into the final IDN standard, IDNA[11], were discussed at great length[12]. The most important ones were:

1. Simply permitting DNS labels to be written in UTF-8 (or some other Unicode form). There is no fundamental bar to doing this in the DNS protocols, at least as they are usually implemented. However, it raised some critical issues about normalization and canonicalization, since many characters can be represented in multiple ways in Unicode. More important, while the DNS design did not present a barrier, many applications, including both email and the web, had their own, more restrictive, definitions of what a DNS name was required to look like. There was empirical evidence (not just theory) that some conforming implementations of those protocols

[9] The current versions of the MIME specifications are RFCs 2045, 2046, and 2047

[10] See http://www.w3.org/TR/html4/charset.html for a discussion of current character-handling techniques for the web.

[11] RFC 3490, 3491, and 3491

[12] The archives of the IDN WG electronic mailing list contain the discussions of many of these options. A subset of them were described at more length in Internet-Drafts (whose identification can also be found in the archives).

would behave very badly when confronted with UTF-8 strings. In addition, there are some cases in which the DNS specification itself is not absolutely clear and interpretations of it differ. Taking this path and making a bad decision implied the risk of seriously damaging DNS operations: not only IDN-specific operations and with no real way back.

2. A second group of proposals involved using the "class" structure of the DNS[13] to permit migration from the current Class=IN (and the associated "LDH" hostnames and ASCII-based matching rules) to a new Class with different, more internationally - compatible, rules. That proposal was received favorably in parts of the actual DNS expert community (a community that was underrepresented in the IDN WG), but ultimately went nowhere for the reasons discussed below. A reasonably mature, although still incomplete, proposal along those lines was posted as draft-klensin-i18n-newclass-02.txt (Google should be able to find a copy for you or check with me). Earlier versions of that draft (...newclass-00.txt, ...newclass-01.txt) go back to discussions that preceded the first public posting in December 2000.

   The "class"-based ideas also carried risks of compatibility with the installed base. While many DNS experts felt comfortable about them (and were not comfortable with the "just use UTF-8" strategy discussed above), there has been only limited actual experience with deployment. When different DNS classes have been used in the past, they have been applied to very different applications and name spaces. We have no experience in trying to transition between one class and another and there are, for example, some issues about what a query for data from "any" class would mean, especially since, nominally, each class can have its own root structure. While none of those problems appeared to be insurmountable, it was clear that it would take some time to work them out well and carefully.

3 There were also a series of "above DNS" approaches as well as general discussion about why they might be a better idea. The unifying theme of those proposals was that the DNS, even for ASCII and the languages that use it, was never going to be a good way to represent user-friendly names and that the community needed to look at other sorts of naming systems (including keywords), directory systems, search systems, and so on, and solve the "multilingual" problems there, in environments that could handle the needs of diverse languages and scripts better, and localization issues better, than would even be possible under the constraints of the DNS. RFC 3467, published in February 2003 (before the IDNA documents), discusses some of those issues vis-a-vis the capabilities of the DNS. Versions of that document were circulated as public drafts as early as the summer and fall of 2000; it and other discussions of the topic go back in less-formal documents somewhat longer. Some related discussion, and proposals for additional work, appear in an unfinished draft[14]

---

[13] In addition to the widely-know resource record "Types", such as "A", "MX", and so on, the DNS supports the idea of a "Class" of such records. Classes may have different matching and label-formation rules and different Types. They typically will have different hierarchical structure. The Class concept has been used for a few specialized applications that are unrelated to conventional DNS naming but not extensively used or tested otherwise: all records in the DNS normally encountered by Internet users are in Class "IN".

[14] draft-klensin-dns-search. Documents mentioned here with names of the form "draft-…" are Internet Drafts. These are working documents that formally expire six months after they are posted and are subsequently useful, if at all for context and historical research. While there is no formal public archive of them, old versions are easily located on the network using the usual search tools. The actual names always contain a version number

and some related documents. They explore some of the possible "above DNS" technologies from the standpoint of what could be made to work and what some of the pieces might be, rather than the "what is the DNS suitable for" approach of RFC 3467. Those drafts date back to the spring and summer of 2001, again with private discussions (of which many participants in the IDN WG were aware, with some participation) occurring even earlier. A more general, but complementary, discussion of Internet navigation issues appears in the US National Research Council Report *Signposts in Cyberspace*[15]

While the first of these options was taken off the table largely for technical and deployment reasons, the other two appeared to be viable. The general consensus today is that it is too late for the "class" proposal or variations on it. Both it and the various "above DNS" approaches succumbed, at least temporarily, to the belief that IDNA would deploy quickly and without problems (since punycode[16] is compatible with legacy applications and implementations). That decision was also driven by immense pressure placed on the IETF from several sources, notably MINC and a few governments, to do something within the DNS and to do something that could, in principle, be deployed quickly, lest people go in their own directions and irretrievably break the DNS. The discussions behind those pressures did not exhibit a great deal of concern for large-scale DNS use in an environment that mixed upgraded DNS implementations (on both servers and user machines) with older ones.

The other advantage of the IDNA approach, as contrasted with others, was that, if it turned out in practice that a serious error had been made, only those labels that started with the chosen prefix (ultimately "xn--") would be impacted: in principle, one could then abandon those names, choose a different prefix or a different method, and move on. An abbreviated version of that approach was actually used in the transition of the early "RACE" coding used in a "testbed" for several domains and the final IDNA coding, but the two approaches were essentially similar except for the way in which characters were normalized and matched and the actual coding used. There would be no going back from a "just install UTF-8" or "new class" approach, whether the fundamental approach or just the character-matching rules were wrong.

Any of these approaches would have left us with a problem that manifested itself in several was, including the "IDN Spoofing" claims of the last year or two and difficulties in transitioning from Unicode 3.2 (as specified with IDNA) and Unicode 4.x and 5.0 (needed for a number of important scripts that have been encoded more recently). That problem is that comparison between Unicode strings created at different times, and potentially with different versions of the character set, are not reliably possible without either significant restrictions on those strings or very complex and detailed context description and the facilities to interpret and take advantage of those descriptions. There will always be issues about whether two sequences of characters are equivalent or not, or look alike or not, and most of those issues will relate to the ways in which Unicode defines characters and strings,

---

starting with "00" and a type identifier, such as ".txt". Hence, e.g., draft-klensin-dns-search-01.txt would be the second version of the "draft-klensin-dns-search" group.

[15] National Research Council, *Signposts in Cyberspace: The Domain Name System and Internet Navigation*, Washington, DC: National Academy Press, 2005. ISBN 0309-09640-5 (Book) 0309-54979-5 (PDF). Chapters 6 through 8 are particularly relevant. http://www7.nationalacademies.org/cstb/pub_dns.html .

[16] The internal, ASCII-compatible, encoding used in IDNA and specified in RFC 3942. The IETF IDN working group expected that, under normal circumstances, ordinary users would neither see punycode encodings nor hear the term. Both of those assumptions have been disproven by actual experience.

not with the particularly coding used, whether that coding be punycode, UTF-8, UCS-4, or something else, nor with the particular normalization technique chosen. Consideration was given to an IETF-specified normalization in order to have a clear, DNS-appropriate, set of rules, but the idea was rejected both because of concerns about inadequate expertise in the IETF and in the expectation that the Unicode Consortium would be better at keeping things up to date (and as sensitive as the IETF believed necessary to forward compatibility issues). That decision may yet need to be reconsidered. In any event, one can make the matching algorithm clear by drastically limiting the number of codepoints that can be used, but such limits conflict directly with the often-expressed view that someone has the "right" to use a particular string as a DNS name and, less directly, with keeping the representation of names culturally and linguistically correct.

Looking Forward

If current trends continue, almost regardless of what we do in the near term, ten (or even five) years from now, most of the users, most of the time, will not be using IDNs, or DNS names more generally, as their principal navigation tool for the Internet. We are already seeing a steadily-accelerating shift from remembering and guessing names to general use of search engines and portals. We will presumably see the growth in the use of search engines continue. Probably we will see more keyword systems and more intentionally-populated directories with or without associated portals and electronic business cards. The reasons will be simple usability by users who, on average, will be less technically sophisticated and careful as the Internet grows: URLs are not, and cannot be made, user-friendly. Internationalized and generalized forms of URL, known as IRIs will not improve things significantly since they preserve the same basic syntax. The addition of more domain names, especially at the top level, will make name-guessing harder. Name-guessing will also become more dangerous in the presence of phishers, spyware-installers, virus-spreaders, and so on. Most important, it is clear that we will need to get beyond the global syntax and structure constraints of the DNS if we are really going to optimize naming and navigation systems to the cultures in which they are used. Getting around those constraints probably implies new systems rather than DNS retrofitting within the existing name spaces: the transition problems with the latter are extremely difficult, perhaps impossibly so, and, if we manage to foul up the DNS, there is no way back for the Internet. A serious mistake would leave us completely without a fallback plan.

Recommendations

While work to deploy IDNs should certainly continue, ISOC members and the broader community should understand that there are alternate approaches to culturally-appropriate naming and access to objects and resources on the network. In particular, there are problems that IDNs will not solve and that might be solved better by other types of systems. In this area, it is also desirable, as it is in so many other areas that involve the Internet, to remember that our most important accomplishments occur through working together to develop solutions that are locally appropriate but still globally scalable and interoperable.