The New Internationalised Domain Name System

Gihan V. Dias
LK Domain Registry

What is IDNA?

- A system to allow applications such as web browsers, mail clients, etc. to handle non-ASCII domain names
  - Stands for *Internationalizing Domain Names in Applications*
- Does not make any changes to name servers or any other DNS infrastructure
  - Users type/paste in/click on names in native characters
  - Converted to ASCII and sent to DNS
  - Conversion happens in application

Why IDNA?

- Most of the world doesn't use Latin script
  - or use extended Latin script with characters such as ä and ø
- DNS only handles labels with letters (a-z), numbers (0-9) and hyphen (-)
- Changing DNS not considered feasible
- Support for IDN provided by applications
  - e.g. web browsers, IM clients, telephones

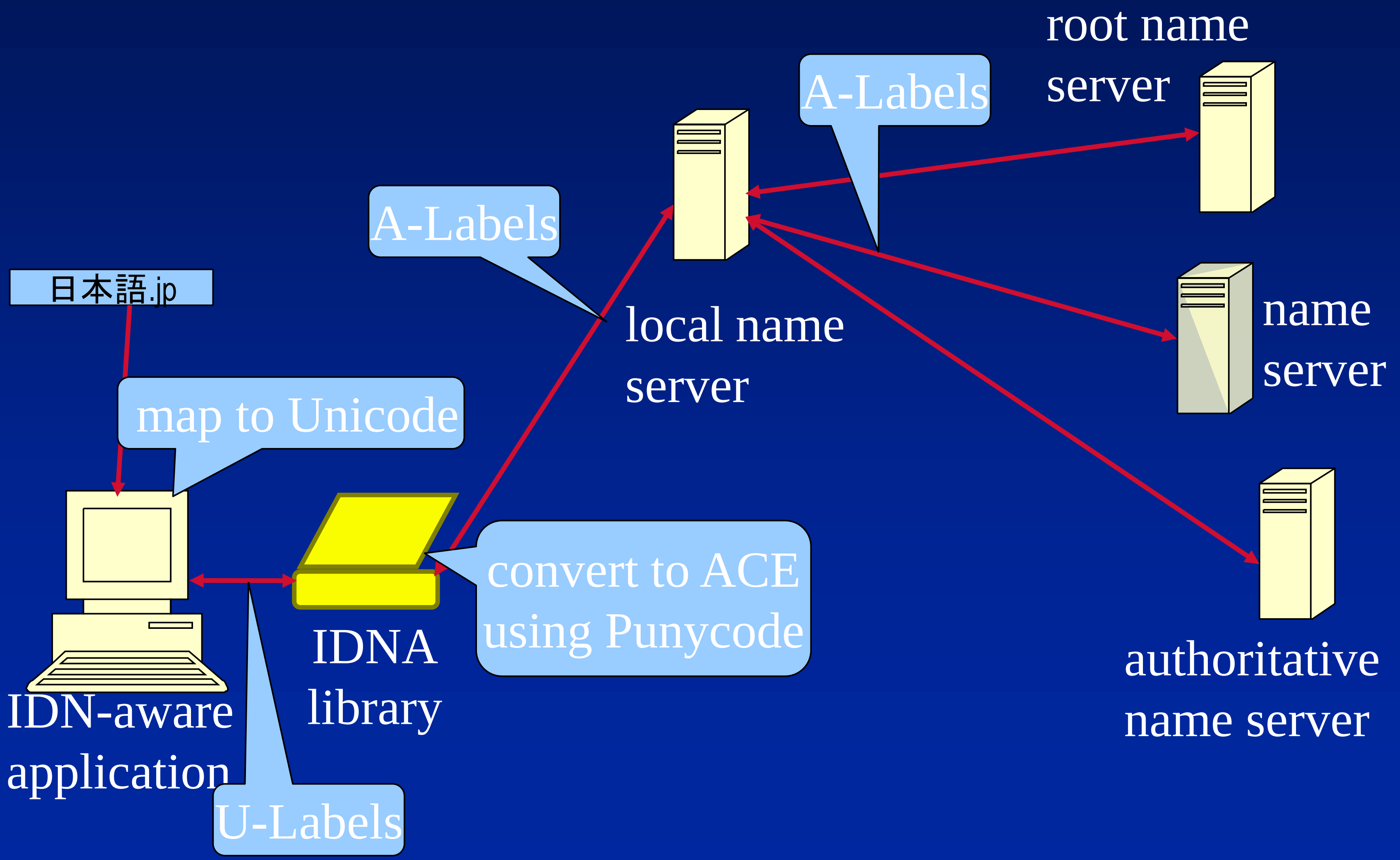
How IDNA Works: Name Resolution

- Name is entered in Unicode
    - possibly converted from other encoding to Unicode
- Name is separated into a sequence of labels at dots
    - Called U-Labels
- If a label has any non-ASCII characters, it is converted to an A-Label
    - using the Punycode algorithm
    - gives an ASCII string starting with “xn--”

Name Resolution (cont.)

- Sequence of A-Labels is sent to DNS
- DNS resolves name and returns requested info
- DNS does not “know” if the original name was ASCII or IDN
- Application getting an A-label will convert to Unicode (or other encoding) for display to user

Name Resolution (cont.)



How IDNA Works:
Name Registration

- Registrant provides name to be registered
  - may be converted to Unicode
- Name is separated to labels at dots
- Each label is validated
  - U-Label
- Each label is converted to ASCII using Punycode
  - A-Label
- Sequence of A-Labels is registered in the DNS

Use of IDN names

- Users will generally deal with names in their own language / script
- Either Unicode, or other encodings
- DNS works with A-labels
  - not User-Friendly e.g. xn—5zc6byczaxq
- Applications will generally display names in original script
  - users need not deal with funny names
  - may occasionally show A-labels

Phishing and other bad things

- IDNAs may be used for phishing
- Certain letters in one script are similar (basically identical) to other letters in another script
  - e.g. Latin a, Cyrillic а
- Same problem occurs with Latin
  - e.g. PaypaI.com

Capital I

- Browsers may restrict use of IDNs

IDNA2003

- First version of IDNA
- Unicode names and ASCII DNS
- Based on Unicode version 3.2

Operation of IDNA2003

- Split domain name into labels
- Process each label with either
  - ToASCII – convert Unicode to ASCII
  - ToUnicode – convert ASCII to Unicode
- ToASCII:
  - if label is already in ASCII format, do nothing
  - Do NAMEPREP processing
  - Convert to ASCII using PUNYCODE algorithm

NAMEPREP processing

- Map – map any input characters which have a mapping
  - may be to null (delete character)
- Normalize – Possibly normalize the result of step 1 using Unicode normalization.
- Prohibit – if any prohibited characters are present, return an error
- Check bidi – if any right-to-left characters, string should satisfy “bidi” requirements

Punycode Algorithm

- ASCII characters in the input string are at the beginning of the output string
- Non-ASCII characters are encoded to letters (a-z) and digits (0-9) and output after a hyphen '-'.
- The string is preceded by the ACE prefix xn--

Examples of Punycode Encoding

Unicode string	ACE string
ascii.com	ascii.com
日本語.jp	xn--wgv71a119e.jp
தமிழ்.in	xn--rlcus7b3d.in
bücher.de	xn--bcher-kva.de
සිංහලidn.lk	xn--idn-u4k9u8ai4i.lk

Issues with IDNA2003

- Limited to Unicode version 3.2
    - need to support new and future versions
    - applications need not be aware of latest version of Unicode
- Does not allow the use of joiners and a few other characters
- Mapping may confuse users who entered one character and got another
- Allows the use of symbols and other non-letter/digit characters
- Problems with bidi rules

IDNA2008

(Approved in 2010)

Objectives of IDNA2008

- Allow IDNA to be updated with later versions of Unicode
- Fix problems with a small number of code points
- Reduce dependency on mapping
- Fix some details if bidirectional algorithm

Principles of IDNA2008

- Character mapping moved out of IDNA to a pre-processing step
  - case mapping also in pre-processing
  - good or bad?
- Permitted characters defined by rules
  - mostly by Unicode properties
  - short list of exceptions

Principles of IDNA2008

- No NAMEPREP stage
- Input should be a valid U-label
  - should be in Unicode normalised form
  - should only have valid characters
- Converted to ASCII using Punycode algorithm
  - no change in Punycode
- Compatible with IDNA2003
  - except in a few specific cases

Principles of IDNA2008

- Reversible one-to-one mapping between each U-label and A-label
  - either one is an exact representation of a name
- U-labels displayed to users and used by IDN-aware applications
- A-labels used by IDN-unaware applications, including DNS

How IDNA2008 works

- pre-processing
- name resolution
- name registration

Pre-Processing

- IDNA assumes that the characters submitted to it are in the correct form
- If the original string is not in Unicode, it must be converted to Unicode
- Mappings may be applied to the string to make it compatible with IDNA2008
- Mappings are not specified in IDNA2008
  - although some guidance is provided in the mappings document

Suggested Mappings

- Map upper-case characters to lower case
- Map “full-width” and “half width” characters to their decomposition mapping
- Map all characters using Unicode Normalization Form C (NFC)
- Map Ideographic Full Stop to Full Stop
- In addition, an application may do additional mappings based on language or locale

Vagueness on Mappings

- IDNA2008 is intentionally vague on mappings
- The idea is that applications should "do the right thing"
- on the other hand, this also creates opportunities for confusion, as different applications may behave differently
- Unicode Technical Standard 46 (UTS46) (also called TR46) attempts to define a standard mapping (discussed later)

Front End and User Interface

- Domain names may be
  - typed in a URL bar
  - read / OCRed from a businesss card
  - spoken (voice recognition)
  - in a URL embedded in a document
- The O.S. input method converts input to Unicode
- IDNA preprocessing may further map the input
- Result should be what the user expects

IDNA Permitted Characters

- IDNA2008 has an inclusion model
  - a character is valid only if it meets the rules
  - or is included as an exception
- Permitted characters
  - Letters and modifiers (in any script) in Unicode NFC form
  - digits
  - hyphen-minus
- Non-permitted characters
  - punctuation, symbols, pictographs

IDNA Character Categories

- IDNA divides all Unicode characters into four categories
- PROTOCOL VALID (PVALID)
  - The character is generally valid
  - may be subject to other rules (e.g. bidi)
- DISALLOWED
  - should never appear in a u-label
  - problematic chars, symbols, etc.
  - no DISALLOWED character will ever be valid

Character Categories (cont.)

- UNASSIGNED
  - not assigned in the current version of Unicode
  - should not be used at present
  - may become PVALID, CONTEXT or DISALLOWED in a future version of Unicode
- CONTEXTUAL RULE REQUIRED
  - two sub-categories

Contextual Restrictions

- CONTEXT-JOINER (CONTEXTJ)
    - zero-width joiner (ZWJ)
    - zero-width non-joiner (ZWNJ)
  - used in Arabic and Indic scripts in a specific context
  - valid in such contexts, invalid otherwise
- CONTEXT-OTHER (CONTEXTO)
  - special characters used in specific languages
  - Should only be registered in such contexts

Name Resolution

The name resolution process is as follows:

- An IDN name is obtained by the application
- The name is divided into labels
- Case folding, normalization and any other mappings are applied
- Each character in each label is considered, and if it is DISALLOWED or UNALLOCATED then error

Name Resolution (cont.)

- If any CONTEXTJ chars, then check context rules
- If any leading combining marks, then error
- If any Right-to-Left characters, then apply bidi rules
- If no errors, apply Punycode algorithm
- Lookup resulting A-Label in DNS

Name Registration

The name registration process is similar to the resolution process, except

- If char. is CONTEXTO, then do contextual processing
- Check if all chars in each label are in the appropriate character table
- Do any additional checks required by the zone
- Each zone should have a character table and also additional rules if needed

Name Registration (cont.)

- If a label begins with xn--, then assume it is an A-label and convert it to a U-label
- Else assume it is a U-label and convert to A-label
- If any errors, exit.
- Else display the U-label and A-label
- IDNA2008 does not recommend any mappings for registrations, but requires registrants to submit valid A- or U-labels

differences between IDNA2003 and IDNA2008

Count	IDNA2003	IDNA2008	Comments and Samples
86676	Valid	Valid	*e.g. U+00E0 (à)*
3302	Valid	Disallowed	*e.g U+2665 (♥)*
4	Mapped / Ignored	Contextj	U+200C (ZWJ) U+200D (ZWNJ) U+00DF (ß) U+03C2 (ς)
4648	Mapped / Ignored	Disallowed	*e.g. U+00C0 (À)*
431	Disallowed	Disallowed	*e.g. +FF01 (！)*

UTS46

- IDNA2008 vague on mappings
  - Does not provide guidance for application developers
- UTS46 (Unicode Technical Standard 46) proposes a standard mapping

  http://www.unicode.org/reports/tr46/
- Maps many characters as in IDNA2003
- Transitionally supports symbols and punctuation
- Four characters marked as “deviation”

Issues with IDNA2008

- Case folding
  - only lowercase allowed in DNS
- Phishing possibilities
- Previously allowed chars disallowed
- Localised mappings for each language / locale

Watch Out: Registrants and Name owners

- Variants
  - different ways of encoding “same” string
- Confusables
  - similar looking letters / sequences in different or same script
  - including ZWJ/ZWNJ
- Label invalid or different in either IDNA2003 or IDNA2008
  - applications which only support IDNA2003 will be around for a while

Watch Out:
Users

- Applications not configured for your script
  - may show A-Labels on URL bar
- Phishing attempts
  - so what's new?
- How do I type this in?
- Funky language/locale-based mapping
  - is *that* what I entered?
- IDN URLs in documents
  - what am I clicking on?

Watch Out: Registries

- Need to define Language table
  - for each zone
- Only register scripts you are familiar with
- Need to define registration policies
  - bundling
  - identification and activation of variants
- Only register U-labels
  - not A-labels
  - may do mapping as a service, but get confirmation of U-label before registration

Watch Out: Application Developers

- Use consistent mapping
  - may be based on UTS46
  - if doing localised mappings, make sure both you and your users understand what you are doing
- Fully support IDNA2008
- Provide IDNA2003 compatibility mode if needed
  - especially for German and Greek

Conclusion

- IDNA2008 solves problems some communities had with IDNA2003
- Designed to be "less confusing"
- May end up creating more confusion if applications are inconsistent
- Proper applications localisation needed for users to benefit
- Lack of uppercase in labels a drawback?

Draft IDNA2008 Documents

Overview Document - IDNA Background, Explanation, and Rationale

- http://tools.ietf.org/html/draft-ietf-idnabis-rationale

IDNA2008 Definitions

- IDNA Definitions and Document Framework
  http://tools.ietf.org/html/draft-ietf-idnabis-defs
- IDNA Protocol
  http://tools.ietf.org/html/draft-ietf-idnabis-protocol
- The Unicode code points and IDNA
  http://tools.ietf.org/html/draft-ietf-idnabis-tables
- Right-to-left scripts for IDNA
  http://tools.ietf.org/html/draft-ietf-idnabis-bidi

Informative document - Mapping Characters in IDNA

http://tools.ietf.org/html/draft-ietf-idnabis-mapping

Gihan Dias
gihan@uom.lk