IETF IDN Working Group
Internet Draft
draft-ietf-idn-icu-00.txt
Expires: 14 January 2001

Seungik Lee, Hyewon Shin, Dongman Lee
ICU
Eunyong Park, Sungil Kim
KKU, Netpia.com

14 July 2000

Architecture of Internationalized Domain Name System

1. Abstract

For restrict use of Domain Name System (DNS) for domain names with alphanumeric characters only, there needs a way to find an Internet host using multi-lingual domain names: Internationalized Domain Name System (IDNS).

This document describes how multi-lingual domain names are handled in a new protocol scheme for IDNS servers and resolvers in architectural view and it updates the [RFC1035] but still preserves the backward compatibility with the current DNS protocol.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

"IDNS" (Internationalized Domain Name System) is used here to indicate a new system designed for a domain name service, which supports multi-lingual domain names.

"The current/conventional DNS" or "DNS" (Domain Name System) is used here to indicate the domain name systems currently in use. It fulfills the [RFC1034, RFC1035], but implementations and functional operations may be different from each other.

The "alphanumeric" character data used here is the character set that is allowed for a domain name in DNS query format, [a-zA-Z0-9-].

3. Introduction

Domain Name System (DNS) has eliminated the difficulty of remembering the IP addresses. As the Internet becomes spread over all the people, the likelihood that the people who are not familiar with alphanumeric characters use the Internet increases. The domain names in alphanumeric characters are difficult to remember or use for the people who is not educated English. Therefore, it needs a way to find an Internet host using multi-lingual domain name: Internationalized Domain Name System.

3.1 The current issues of IDNS

IDNS maps a name to an IP address as the typical DNS does, but it allows domain names to contain multi-lingual characters. The multi-lingual characters need to be encoded/decoded into one standardized format, and it needs changes in the conventional DNS protocol described in [RFC1034] and [RFC1035]. But it is required to minimize the changes in the present DNS protocol so that it guarantees the backward compatibility.

The IDNS issues have been discussed in IETF IDN Working Group. These issues are well described in [IDN-REQ]. The main issues are:

- Compatibility and interoperability. The DNS protocol is in use widely in the Internet. Although a new protocol is introduced for DNS, the current protocol may be used with no changes. Therefore, a new design for DNS protocol, IDNS must provide backward compatibility and interoperability with the current DNS.

- Internationalization. IDNS is on the purpose of using multi-lingual domain names. The international character data must be represented by one standardized format in domain names.

- Canonicalization. DNS indexes and matches domain names to look up a domain name from zone data. In the conventional DNS, canonicalization is subjected to US-ASCII only. However, every multi-lingual character data must be canonicalized in its own rules for a DNS standardized matching policy, e.g. case-insensitive matching rule.

- Operational issues. IDNS uses international character data for domain names. Normalization and canonicalization of domain names are needed in addition to the current DNS operations. IDNS also needs an operation for interoperability with the current DNS. Therefore, it is needed to specify the operational guidelines for IDNS.

3.2 Overview of the proposed scheme

Our proposed scheme for IDNS is also subjected on the issues described earlier to fulfill the requirements of IDN [IDN-REQ].

The proposed scheme can be summarized as following:

- The IN bit, which is reserved and currently unused in the DNS query/response format header, is used to distinguish between the queries generated by IDNS servers or resolvers and those of non-IDNS ones [Oscarsson]. This mechanism is also needed to indicate whether the query is generated by the appropriate IDNS operations for canonicalization and normalization or not.

- The multi-lingual domain names are encoded into UTF-8 as a wire format. UTF-8 is recommended as a default character encoding scheme (CES) in the creation of new protocols which transmit text in [RFC2130]. This scheme allows the IDNS server to handle the DNS query from non-IDNS servers or resolvers because the ASCII code has no changes in UTF-8.

- The UTF-8 domain names must be case-folded before transmission. It minimizes the overhead on server's operations of matching names in case-insensitive. It also guarantees that the result of caching queries can be used without any further normalization and canonicalization. If IDNS server gets non-IDNS query that is not case-folded, it case-folds the query before transmitting to another servers.

4. Design considerations

Our proposed scheme is designed to fulfill the requirements of IETF IDN WG [IDN-REQ]. All the methods for IDNS schemes must be approved by the requirements documents. The design described in this document is based on these requirements.

4.1 Protocol Extensions

To indicate an IDNS query format, we use an unallocated bit in the current DNS query format header, named 'IN' bit [Oscarsson]. All IDNS queries are set IN bit to 1. Without this bit set to 1, we cannot guarantee that the query is in the appropriate format for IDNS.

'IN' bit is to indicate whether the query is from IDNS resolvers/servers or not. It also reduces overhead on canonicalizing operation at IDNS server. It will be described further in <4.4. Canonicalization>.

We devise new operations and new structures of resolvers and name servers to add the multi-lingual domain name handling features into the DNS. This causes changes of all DNS servers and resolvers to use multi-lingual domain names. The new architectures for resolvers and servers will be described in <5. Architectures>

4.2 Compatibility and interoperability

The 'IN' bit is valid bit location of query for the conventional DNS protocol to be set to zero [RFC1035]. And operations and structures of IDNS preserve the conventional rules of DNS to guarantee the interoperability with the conventional DNS servers or resolvers so that the changes are optional. These make this scheme for IDNS compatible with the current protocol.

Although the current DNS protocol uses 7-bit ASCII characters only, the query format of the current DNS protocol set is 8 bit-clean. Therefore, we can guarantee the backward compatibility and interoperability with the current DNS using UTF-8 code because the ASCII code is preserved with no changes in UTF-8.

Note: There are also in use implementations that are compatible with the current DNS but extend their operations to use UTF-8 domain names. The IDNS described here interoperates well with these implementations. The interoperability with these implementations will be described in <5.4 Interoperability with the current DNS>.

4.3 Internationalization

All international character data must be represented in one standardized format and the standardized format must be compatible with the current ASCII-based protocols. Therefore, the coded character set (CCS) for IDNS protocol must be Unicode [Unicode], and be encoded using the UTF-8 [RFC2279] character encoding scheme (CES).

The client-side interface may allow the domain names encoded in any local character sets, Unicode, ASCII and so on. But they must be encoded into Unicode before being used in IDNS resolver. The IDNS resolver accepts Unicode character data only, and converts it to UTF-8 finally for transmission.

4.4 Canonicalization

In the current DNS protocol, the domain names are matched in case-insensitive. Therefore, the domain names in a query and zone file must be case-folded before equivalence test.

The case-folding issue has been discussed for a long time in IETF IDN WG. The main problem is for case folding in locale-dependent. Some different local characters are overlapped within case-folded format. For example, Latin capital letter I (U+0049) case-folded to lower case in the Turkish context will become Latin small letter dotless i (U+0131). But in the English context, it will become Latin small letter i (U+0069)

Therefore, we case-fold the domain names in locale-independent in our new IDNS design with a method defined in [UTR21].

Multi-lingual domain names should be case-folded in IDNS resolvers or IDNS servers before transmitting to other IDNS/DNS servers. That is, IDNS resolver should case-fold the domain name and converts it to UTF-8 before transmission. In case of IDNS server, if it gets a query with IN bit set to 1, then it needs not to make the multi-lingual domain name canonicalized anymore. If the IDNS server gets a query with IN bit set to 0, then it cannot determine the query is appropriate canonicalized format for IDNS server, so that it case-folds that multi-lingual domain name in the query, and set 'IN' bit to 1.

The current DNS queries contain the original case of domain names to preserve the original cases. To be consistent with this rule, all case-folded multi-lingual domain names should be stored by IDNS resolvers or servers before case-folding, and should be restored before sending response.

In the case of case-folding UTF-8 code, using the case-folding method in [UTR21], the UTF-8 should be converted to Unicode and it must be mapped to the mapping table finally. Of course that if we could make a case-folding mapping table of UTF-8 character data, this overhead could be reduced.

However it cannot avoid an overhead in IDNS servers for canonicalization, because the canonicalization of international character data is complicated.

To minimize this overhead, we use 'IN' bit to indicate that the canonicalization for the query has been already handled. That means it needs not canonicalization operation anymore. The detailed operations according to the 'IN' bit are described later in <5. Architectures>.

With international character data, the canonicalization (e.g. case-folding) is much more complicated than the one with US-ASCII, and is different from each other's by their locale contexts.

But this document doesn't specify any method or recommendation more than case-folding. For canonicalization of international character data, [UTR15] is a good start. It must be discussed further and specified in the IDNS protocol specification.

4.5 Operational issues

In the current DNS scheme, it uses only ASCII code for a wire format. But our new IDNS scheme uses UTF-8 code for a wire format. All the IDNS resolvers must transmit queries encoded in UTF-8 and case-folded. This format can be guaranteed by checking the IN bit: if IN bit is set to 1, the query is encoded in UTF-8 and case-folded. Otherwise the IDNS server cannot assure that the query is encoded in UTF-8 and case-folded. Therefore it needs additional operations for encoding to UTF-8 and case-folding, etc in this case.

The current DNS resolvers transmit the queries in ASCII code. But it's not considerable in IDNS servers because the ASCII code is preserved with no changes in UTF-8.

Some applications and resolvers transmit the queries in UTF-8 although they don't fit on the new IDNS resolvers' structures, e.g. Microsoft's DNS servers. We cannot guarantee that those queries are case-folded correctly. Therefore, the IDNS servers should convert them to appropriate IDNS queries instead of the IDNS resolver in that case.

All detailed operations of IDNS servers and resolvers are described in <5. Architectures>.

5. Architectures

5.1 New header format

A new IDNS servers and resolvers must interoperate with the ones of current DNS. Therefore, we need a way to determine whether the query is for IDN or not. For this reason, we use a new header format as proposed in [Oscarsson].

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|IN|AD|CD|   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QDCOUNT                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     ANCOUNT                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     NSCOUNT                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     ARCOUNT                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

The IDNS resolvers and servers identify themselves in a query or a response by setting the 'IN' bit to 1 in the DNS query/response format header. This bit is defined to be zero by default in the current DNS servers and resolvers.

5.2 Structures of IDNS resolvers

To use multi-lingual domain names with IDNS servers, all the IDNS/DNS resolvers must generate the query in a format of UTF-8 or ASCII. The design of a resolver could be different with each other according to the local operating systems or applications. We propose new design guidelines of a resolver for a new standardization.

The IDNS resolver accepts Unicode from user interface for domain names. The other character sets should be rejected. It encodes all such character data into UTF-8 for transmission to name servers.

The procedures of the operation of an IDNS resolver are below:

<1>. If the resolver gets a domain name in Unicode or ASCII then it stores the original domain name query. Otherwise the request for lookup is rejected. In the current DNS protocol, the original case of the domain name should be preserved. Therefore, the resolver must store the original cases of the domain names before canonicalization (e.g. case-folding).

<2>. Make the domain name case-folded with locale-independent case-mapping table defined in [UTR21].

<3>. Convert it to UTF-8.

<4>. Set IN bit to 1. It indicates the query is from IDNS resolver and the format is UTF-8, case-folded.

<5>. Send request query to name servers.

<6>. Restore the original domain name query into the response query format.

<7>. Send response to the application.

5.3 Structures of IDNS servers

The operation of IDNS server is similar to the current one of DNS server, but the IDNS server accepts UTF-8 queries and converts them to the appropriate formats additionally.

The IDNS server distinguishes between the IDNS queries and DNS queries by checking IN bit in the query/response format header. According to the 'IN' bit, it operates differently.

The procedures of the operation of an IDNS server are below:

<1>. If the IN bit in the query/response format header is set to 1 then it matches the domain name within zone file data or forwards request query to resolve. It operates as same as the operations of the current DNS servers but retrieves UTF-8 code. In this case, it needs not to make domain name canonicalized because the domain name is already canonicalized in the previous procedures of IDNS resolvers or IDNS servers. Go to step <7>.

<2>. Set IN bit to 1.

<3>. Store the original domain name query.

<4>. Make the domain name case-folded with locale-independent case-mapping table defined in [UTR21].

<5>. Match the domain name within zone file data or send request query to lookup.

<6>. Restore the original domain name query into the response query format.

<7>. Send response for the query to the resolver or the other server requested.

5.4 Interoperability with the current DNS

The DNS servers and resolvers accept domain names in ASCII only. But IDNS servers and resolvers accept domain names in UTF-8. Therefore, the queries from DNS ones to IDNS ones can be well handled because the UTF-8 is a superset of ASCII code. But the queries from IDNS ones to DNS ones will be rejected because the UTF-8 code is beyond the range of ASCII code.

Note: There are some implementations which can handle UTF-8 domain names although they don't fit on this specification of IDNS and fully implemented with DNS protocol specification, e.g. Microsoft's DNS server and resolvers. In this case, we cannot guarantee that the queries from these 3rd-party implementations are encoded into UTF-8 and well canonicalized. But this queries are set 'IN' bit to 0, so that the IDNS evaluates whether the domain name is the range of UTF-8 or not, and converts it into UTF-8 and makes it canonicalized finally.

6. Security Considerations

This architecture of IDNS uses 8bit-clean queries for transmission and the UTF-8 code is handled instead of ASCII. The DNS protocol has already allocated 8bit query format for domain names Therefore, the IDNS protocol inherits the security issues for the current DNS.

Canonicalization of IDNS is defined in [UTR15] and case folding in [UTR21]. All security issues related with canonicalization or normalization inherits ones described in [UTR15, UTR21].

As always with data, if software does not check for data that can be a problem, security may be affected. As more characters than ASCII is allowed, software only expecting ASCII and with no checks may now get security problems.

7. References

[IDN-REQ]	James Seng, "Requirements of Internationalized Domain Names," Internet Draft, June 2000
[KWAN]	Stuart Kwan, "Using the UTF-8 Character Set in the Domain Name System," Internet Draft, February 2000
[Oscarsson]	Dan Oscarsson, "Internationalisation of the Domain Name Service," Internet Draft, February 2000
[RFC1034]	Mockapetris, P., "Domain Names - Concepts and Facilities," STD 13, RFC 1034, USC/ISI, November 1987
[RFC1035]	Mockapetris, P., "Domain Names - Implementation and Specification," STD 13, RFC 1035, USC/ISI, November 1987
[RFC2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, March 1997
[RFC2130]	C. Weider et. Al., "The Report of the IAB Character Set Workshop held 29 February - 1 March 1996," RFC 2130, Apr 1997.
[RFC2279]	F. Yergeau, "UTF-8, a transformation format of ISO 10646," RFC 2279, January 1998

[RFC2535] D. Eastlake, "Domain Name System Security Extensions," RFC 2535, March 1999

[UNICODE] The Unicode Consortium, "The Unicode Standard - Version 3.0," http://www.unicode.org/unicode/

[UTR15] M. Davis and M. Duerst, "Unicode Normalization Forms", Unicode Technical Report #15, Nov 1999, http://www.unicode.org/unicode/reports/tr15/

[UTR21] Mark Davis, "Case Mappings," Unicode Technical Report #21, May 2000, http://www.unicode.org/unicode/reports/tr21

8. Acknowledgments

Kyoungseok Kim <gimgs@asadal.cs.pusan.ac.kr>
Chinhyun Bae <piano@netpia.com>

9. Author's Addresses

Seungik Lee
Email: silee@icu.ac.kr

Hyewon Shin
Email: hwshin@icu.ac.kr

Dongman Lee
Email: dlee@icu.ac.kr

Information & Communications University
58-4 Whaam-dong Yuseong-gu Taejon, 305-348 Korea

Eunyong Park
Email: eunyong@eunyong.pe.kr
Konkuk University
93-1 Mojindong, Kwangjin-ku Seoul, 143-701 Korea

Sungil Kim
Email: clicky@netpia.com
Netpia.com
35-1 8-ga Youngdeungpo-dong Youngdeungpo-gu Seoul, 150-038 Korea