



166166

A METHOD OF RESOLVING A NON-LATIN CHARACTER URLBACKGROUND AND FIELD OF THE INVENTION

5 This invention relates to URL address retrieval.

Each Internet website has a unique website URL (uniform resource location) address which a user enters using his web browser in order to access the site. The URL has two parts, a domain name, specifying the host or the IP (Internet
10 Protocol) address and the path to the resource. These are entered by the user in a word form which is then translated into a numerical address. Conventionally, URLs are composed from a standard character set comprising Latin alphabet characters and numerals and are resolved using "root" registry facilities provided by ICANN in conjunction with "branch" national registries in
15 each country.

It is a problem with such URL addresses that they use exclusively Latin characters and it is thus not possible to represent non-Latin character URL addresses using other language representations, such as using Chinese
20 characters, or non-Latin alphabets such as Thai, since these cannot be resolved. Proposals have been made for representing non-Latin character URL addresses hereinafter referred to as i18n (short for internationalization) addresses and two implementations have currently been proposed IDNS (Internationalized Domain Name System) and CNNIC (China Internet Network
25 Information Center) for implementation of i18n domains. Each implementation has a different name server. It is a problem that no standard implementation has yet been agreed leading to difficulties in accessing i18n domains using different implementations and i18n URL domains that use different implementations may not be unique leading to confusion, particularly if the
30 number of implementations and registries for non-Latin domain names grows in accordance with expected demand in countries which use non-Latin character sets.



G00002

It is also a problem for users to input non-Latin characters. Most computers in the world uses Operating Systems that have only input capabilities for the Latin character set. For users of these computers, the users has no facility to type in the non-Latin URL of the i18n domains, and cannot access these URLs. Some
5 Operating Systems have input capabilities for one (or a few) non-Latin character sets (e.g. Chinese Windows 95, Japanese Windows 98). Language input programs that integrate with the Operating System to allow input of non-Latin characters (e.g. HansVision for Chinese input) also exist. A problem which such
10 systems is that users cannot type the URL of an i18n domain, and cannot access URLs of an i18n site of a different character set (for example, a user using Chinese Operating System that wants to go to a Tamil URL site).

15 It is an object of the invention to provide a method of URL address retrieval that alleviates at least one problem of the prior art or provides the user with a useful choice.

SUMMARY OF THE INVENTION

20

According to the invention, in a first aspect, there is provided a method of resolving a non-Latin character URL, having a domain name, entered by a user comprising the steps of: selecting a character set in which the URL is written; inputting the URL in characters from the character set; determining, using the
25 character set of the input URL, one or more name servers that support domain names in the selected character set; querying at least one said name server with the domain name; and fetching contents from the URL if the query is satisfied.

30 According to the invention, in a second aspect, there is provided a method of resolving a non-Latin character URL, having a domain name, entered by a user using an Internet browser comprising the steps of: contacting a website

supporting non-Latin character URL resolution, downloading a non-Latin character URL input program from the website, inputting the URL using the program and sending the URL to the website; determining at the website, using the character set of the URL, one or more name servers that support domain
5 names in the selected character set; querying from the website at least one said name server with the domain name; and fetching contents from the URL if the query is satisfied and passing the contents to the user.

Apparatus for performing the method of the aspects of the invention noted
10 above is also envisaged as forming part of the invention and further features of the invention are mentioned in the claims appended hereto.

In the described method of the invention, a user is allowed to use any internet browser that has JavaScript and/or Java support to enter a non-Latin character
15 URL (uniform resource location) and be able to retrieve and view non-Latin character contents without installing any additional software on his computer and without changing any client-side settings. The embodiment allows different methods (solutions) of i18n (Internationalized) domains to be accommodated. A non-Latin character editor is provided to enter a non-Latin character URL and
20 to convert non-Latin character web pages to graphics dynamically for viewing without additional software. A user selects an input method with a language encoding. With each input method, a list of choices of the characters is displayed through graphics fetched from a web server to allow the user to select the character as the user types. Each character has a unique Unicode
25 (Universal Character Set ISO/IEC 10646 implementation) representation and this allows the embodiment to keep track of the Unicode and to display the character using a graphic having the Unicode as its filename.

Unicode is only one example of a possible character encoding that the invention
30 can use. In a preferred implementation, a character graphics web server may contain multiple character graphics sets in different languages and character codings, each in a different folder and having a naming standard that maps the

character code to a URL name. The fetched contents of the URL may be dynamically converted to a graphical representation which allows the user to view any non-Latin character contents without additional software.

- 5 The described method can resolve an i18n domain URL that can include any implementation of an i18n domain. (There are 2 known implementations currently, they being provided by I-DNS and CNNIC). A domain name query is sent to a name server that supports a particular implementation of i18n domains. On the client-side, a function to allow a user to select one of the i18n
10 implementation is preferably provided. A default ordering of the i18n methods is also preferably provided for resolving so that the IP address of the domain host of the first i18n domain method that returns an IP address successfully is used. After the contents are fetched, these may be filtered so that if a URL in the contents contains an i18n domain, this URL can be replaced with the JavaScript
15 codes that are able to resolve the i18n domain.

BRIEF DESCRIPTION OF THE DRAWINGS

- An embodiment of the invention will now be described by way of example with
20 reference to the accompanying drawings in which:

Figure 1 is a schematic diagram illustrating the principal components used in the embodiment of the invention.

- Figure 2 is a flowchart illustrating an embodiment of the method of the
25 invention.

Figure 3 is a flowchart to explain the step of obtaining the multi-lingual URL of Fig. 2.

Figure 4 is a flowchart to explain the step of resolving the i18n domain of Fig. 2.

Figure 5 is a flowchart to explain the fetch and filter contents step of Fig. 2 .

30

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

To illustrate the principles of the method of the present invention, FIG. 1 shows a high-level schematic view of the different components used. An Internet browser 110 is provided that is opened by a user whose computer is connected to the Internet 100. The user is connected to a web server 140 which contains
5 downloadable programs namely a non-Latin character editor toolbar 120 and also a JavaScript and Java applet 125 supporting the method of the present invention. A name resolver program 145 is also disposed on the server 140 and is required to do domain name resolving on the server-side. Three different types of server may be connected to the Internet namely a normal name server
10 10 (e.g. Bind), a name server 11 using the i18n domain solution provided by IDNS and a name server 12 using the i18n domain solution provided by CNNIC.

An overview of the operation of the method will first be described. A user can open up the Internet browser 110 and browse to the web server site 140. The
15 browser 110 retrieves the non-Latin character editor toolbar web page 120 with the associated JavaScript and the hidden Java applet 125. The non-Latin character editor 120 works in Unicode for essential resolution of international characters but may use any other coding scheme or allow the user to select the coding scheme to be used. The user selects an input method for the characters
20 forming the address and any non-Latin character domain URL can then be entered by running the JavaScript program 125 in the client browser 110 which causes the toolbar to be displayed allowing character entry. The JavaScript program captures the address that the user has entered and decides the Unicode character the user has selected based on the input method the user
25 has chosen. The JavaScript then displays the Unicode character by fetching a graphics file (which has a filename equivalent to the Unicode) from the web server 140 allowing a Unicode character to be dynamically converted to its graphical representation without requiring the user to install any non-Latin character software on his computer.

30

The user then hits <enter> with the intention of browsing to the non-Latin character domain URL that he has just entered. It is important that the software

of the present invention is used for entering the URL. For example, if the user has instead entered this URL with the help of other software into the browser 110 address bar and hit <enter>, the browser 110 will proceed to fetch the URL directly using the default name server as configured in the Operating System (usually) or the browser. If that name server is not capable of resolving i18n addresses, a host name lookup failure will occur. Even if the default name server is i18n capable, it is very likely that it comply to only one implementation of i18n domain name standard (e.g. IDNS or CNNIC, not both), and a host name lookup failure may occur (if the user wants a i18n domain in the iDNS standard, but the default name server setting is set to CNNIC, for example).

The next step is to resolve the non-Latin character domain to an IP address. The name server settings on the computer will normally point to a normal name server 11 at this current date. If a domain name query is send to such a name server 11, the name server 11 will return an error since the non-Latin character domain name is not a valid standard domain name (which is allowed only to consist of Latin alphabet, digits and hyphen). In the present method, the domain name query is performed by either the Java applet 125 or the name resolver 145 which determines from the Unicode representation, if the URL is in Latin character form or not. The name resolver keeps a list of well-known IP servers that provides domain name service for each type of i18n domain implementation. Thus, the domain name query will be send to the correct name server that knows how to resolve the non-Latin character domain name. For example, the non-Latin character domain name query can be sent to either the IDNS name server 11 or/and the CNNIC name server 12.

Since the servers 11,12 are not mutually exclusive the same non-Latin character domain may be registered by different parties, resulting in a conflict. This is resolved in the present method by allowing the user to choose the i18n domain solution provider.

A client-side domain name query will use the Java applet 125 to perform the resolution while a server-side domain name query will use the name resolver to perform the resolution. Both solutions basically are similar except that the server-side solution is more flexible and can have better features. For example, it will be easier to update the name server list information on the name resolver 145 and it can support advanced caching. For the client-side solution, the Java applet 125 proceeds to fetch the contents of the URL based on the IP address that is resolved earlier.

After the contents are fetched, the Java applet filters the contents so that if a URL in the contents contains an i18n domain, this URL is replaced with JavaScript codes which will forward the request to the Java applet 125 to resolve i18n domain URL when a user clicks on it. The Java applet 125 then displays the contents in another frame in the browser 110. For the server-side solution, the contents can be fetched and then filtered before the web server 140 returns the contents to the client browser 110. Thus the present method is able to resolve any i18n domain names from different i18n domain implementations as long as their name server solutions supports the standard name server solution, that is to say, the concept and design of the name server should follow what is described in RFC 1034 and RFC 1035. In particular, DNS queries and responses should be carried in a standard message format as described in RFC 1034 and RFC 1035 so that the same resolver may be used to query each of the name servers independent of the i18n domain implementation.

FIG. 2 presents a flow chart of the complete method. In the first step, a user uses the non-Latin character editor to enter a non-Latin character URL. (shown in block 20) Next, the domain part of the URL is resolved to obtain an IP address. (shown in block 21) The contents of the URL is fetched based on the IP address and the contents are filtered to replace URLs that contains an i18n domain with JavaScript codes. (shown in block 22) The contents are displayed

on the user's browser even if the browser does not have non-Latin character support. (shown in block 23).

The process indicated by block 20, which is initiated by the user is shown in greater detail in FIG. 3. In block 201, the non-Latin character URL editor toolbar downloaded from a web page on server 140 is displayed in the browser frame. A user can conveniently connect to this web page by using an internet bookmark or the user can enter the URL directly in the internet address bar of the browser. The current Internet Explorer browser supports custom toolbars such as this that can be placed at the left or bottom side of the browser and which can be invoked from the links toolbar for easy access. The non-Latin character editor does not require any client-side installation or any configuration settings on the user computer and can be extended to support any kind of language and input methods and all of this can be updated in the server without the user being aware of it. The user's computer is not required to install any language fonts and the browser is not required to support non-Latin character languages. Each non-Latin character character is displayed using a graphics file that is stored in the web server. Using the language encoding and knowing the Unicode representation of the character, the representing graphics file can be named using the Unicode and thus, this allows any non-Latin character web page to be displayed.

In block 202, the user selects the language and input method. By doing so, the JavaScript codes in the web page download the appropriate dictionary corresponding to the selected input method so that it is able to map what the user keys in to a character code. JavaScript is used as it is supported by most of the popular browsers including Internet Explorer and Netscape browser. In block 203, the running JavaScript codes capture the keys entered by the user and the dictionary is checked for correspondence to any possible choices of characters. If there is only a single choice, then the character code is determined, otherwise the list of choices are displayed for the user to select as shown in block 204. All characters are displayed by fetching their representing

graphics files from the web server. In block 205, the user selects the graphic character from the list of choices displayed and thus the character code is determined. In block 206, the JavaScript codes append the Unicode character code to an internal URL string and also display the graphics character in the URL editor. This process is then repeated as the user keys in the full URL before the user hits the <enter> key to signal that he has finished and wants to browse the web page indicated by the URL. Block 207 illustrates this repeat process. The user is able to switch to another language and input method and thus input a URL in multiple character sets at block 202 provided each character has a Unicode equivalent, although the existing language and method remaining as a default unless changed by the user in block 202. When the user switches to another language and/or input method, a different dictionary may be automatically downloaded by the Javascript codes. The JavaScript codes will store the Unicode string URL that the user has keyed in at the end of the process of block 20.

In block 21, the domain part of the URL is resolved to obtain the host IP address. This process is illustrated in greater detail in FIG. 4. There are currently at least two proposals for i18n domain solutions, one is offered by IDNS and the other is offered by CNNIC. As there is huge business value for non-Latin character domain names, it is anticipated that more i18n domain proposals might be forthcoming. As such, it is desirable that the domain name can be resolved correctly against any i18n domain solutions. Provided the i18n domain solution is compatible with the current standard domain name servers, the present method can accommodate this and if the i18n domain name server can accept the same kind of the standard domain name query and returns the IP address to the client that sends the query, it will be compatible. As of now, both IDNS and CNNIC solutions are compatible. The present method needs to query the correct domain name server that supports the selected i18n domain solution or simply, all the name servers that support the i18n domain solution provided by IDNS or CNNIC can be queried.

Since in general, it is not possible to distinguish in which i18n domain solution (domain registrar) a domain is registered, the user can select the i18n domain solution in the non-Latin character editor toolbar. A default ordering of the i18n domain solutions is also provided so that when the software queries a domain name this is done against the name servers of one domain solution before the name servers of another solution. In practice, the implementation may query name servers of different i18n domain solutions in parallel to speed things up and then return the IP addresses with the default ordering. If there is a conflict of a domain name registered in different domain solutions, the user is given the option to try the other hosts (IP addresses) if the default is not the one he wants. In FIG. 4, the diagram depicts the process of the domain name query is carried out in series although this can be done in parallel also.

In block 211, the process checks whether the domain is in standard URL characters, if it is, the contents of the URL are retrieved as normal. If the domain is an i18n domain, the present method proceeds to resolve the domain by first retrieving the full list of i18n domain solutions shown in block 212. (e.g. IDNS and CNNIC). In block 213, a domain solution is selected and a list of domain name servers that support the selected domain name solution is retrieved. Currently, such a list of server IP addresses can be compiled manually and configured for the process to use. For each domain name server (as shown in block 214) in the list, the process sends a domain name query against it as shown in block 215. If the domain name server returns an IP address, the process is complete (as shown in block 216) although in practice, more than one IP address from different i18n domain solutions can be returned as described earlier to solve domain name conflicts. If the domain name server returns an error, the block 217 is executed to repeat the domain name query against another of the name servers in the list. In block 218, the process is repeated for another domain name solution by sending the same domain name query against another list of name servers supporting the solution. (In practice, this can be done in parallel).

The domain name resolving process can be implemented in either the client-side or the server-side. A client-side solution can be implemented using a Java applet. A server-side solution is more flexible and can have better features. For example, it will be easier to update the name server list information on the name resolver 145 and it can support advanced caching for faster name resolution. If the client-side resolution is used, the process of fetching and filtering the contents of the URL as shown in block 22 can also be implemented within the same Java applet. If the server-side name resolver 145 is used instead, there is a choice of either using client-side or server-side fetching and filtering. Client-side fetching and filtering has the advantage of offloading the server but it will make the client-side codes more complicated. Server-side filtering can resolve all i18n domains in the URLs of the contents all at once with good caching.

In block 22, the contents of the URL are fetched based on the IP address and the contents is filtered to replace URLs that contains an i18n domain with Javascript codes which will forward the request to the Java applet to resolve i18n domain URL when a user clicks on it. If the original URL that contains an i18n domain is clicked on, the browser will use the default domain name query which may fail against a standard name server. Thus, the contents are filtered to solve this problem. Even if all such i18n domains are resolved before displaying the contents by replacing the domains with their IP addresses, it is not sufficient to ensure that it will work correctly in all cases. In the case when a web server at an IP address chooses to host multiple domains, the loss of the original domain part in the URL will prevent the web server retrieving the URL contents. The solution used in the present method is to replace the URL with a Javascript function call and pass the parameter of the original URL to it. (An additional parameter for the host IP address can be passed if already resolved) When a user clicks on the replaced URL, the Javascript function is initiated to fetch the contents and this functionality is similar to fetching the contents of the URL the user keys in with the non-Latin character editor. This process of fetching and filtering is illustrated in FIG. 5 in greater detail.

In block 221, the contents of the URL are fetched. The process uses the resolved host IP address to connect to the web server and then fetches the contents of the original URL. The fetched contents are then parsed in block 222. In block 223, each URL in the contents is extracted. The URL is checked to see whether it contains an i18n domain name in block 224. If it does, it is replaced by the Javascript function call as shown in block 225. Block 226 illustrates the process is repeated for each of the extracted URL in the contents. The updated contents are then returned and the process is completed. Next, the contents are written back to the browser frame as shown in block 23.

10

The embodiment of the present invention described is not to be construed as limitative. For example, although the described embodiment uses UNICODE character coding, JavaScript as a scripting language and Java as a programming language, use of other suitable coding schemes instead of or as well as Unicode, other scripting languages such as Perlscript or Vbscript and other bytecode interpreted programming languages is envisaged.

20

For example, a character graphics web server may contain multiple character graphics sets in different languages and character codings, each in different folder and a naming standard that maps the character code to the URL name.

e.g. the word 六 (for the word 'six' in Chinese) is code 0xC1F9 in GB2312 code, and its graphics can be stored in the web server under the folder gb, and thus have the URL [http://\[graphics_web_server_domain\]/gb2312/C1F9.gif](http://[graphics_web_server_domain]/gb2312/C1F9.gif)

25

The same word 六 is also code 0x516D in Unicode version 3.0, and its graphics can be stored in the same web server under the folder unicode3, and have the URL [http://\[graphics_web_server_domain\]/unicode3/516D.gif](http://[graphics_web_server_domain]/unicode3/516D.gif)

30

The naming for the folder and file name for each character is determined by the Javascript/Java code with consultation from the dictionary loaded. A "Hanyu-pinyin input method" dictionary built for the GB2312 coding may point the graphics for the input "liu4" for the word 'six' in Chinese to [http://\[graphics_web_server_domain\]/gb2312/C1F9.gif](http://[graphics_web_server_domain]/gb2312/C1F9.gif), while another "Hanyu-pinyin input method" dictionary built for the Unicode coding may point the

graphics for the same input "liu4" for the word 'six' in Chinese to
[http://\[graphics_web_server_domain\]/unicode3/516D.gif](http://[graphics_web_server_domain]/unicode3/516D.gif).

- For practical usage, in a preferred form, all the character graphics files may be
5 arranged in Unicode, and all dictionaries customized to Unicode as far as
possible (since Unicode contains characters of multiple languages), so as to
minimize maintenance of a large number of different graphics sets and
dictionaries.
- 10 Reference to a non-Latin character URL includes reference to a URL which has
a non-Latin character domain name and a Latin or non-Latin character path
portion.

References

UTF-5, a transformation format of Unicode and ISO 10646

5 <http://www.idns.org/technical/draft-jseng-utf5-00.txt>

iDNS, an Experimental DNS System with Unicode Support

<http://www.idns.org/technical/iuc14-paper.doc>

10 P. V. Mockapetris, Domain names - Implementation and specification.,
RFC1035, November 1987

P. V. Mockapetris, Domain names Concepts and Facilities, RFC1034,
November 1987

15

Martin Duerst, Internationalization of Domain Names, Internet Draft, July 1997,
extended in 1998.

The Unicode Standard, URL <http://www.unicode.org>

20.

CLAIMS

1. A method of resolving a non-Latin character URL, having a domain name,
entered by a user comprising the steps of:
5 selecting a character set in which the URL is written;
 inputting the URL in characters from the character set;
 determining, using the character set of the input URL, one or more name
 servers that support domain names in the selected character set;
 querying at least one said name server with the domain name; and
10 fetching contents from the URL if the query is satisfied.
2. A method as claimed in claim 1 further comprising the step of converting
each character of the input URL to a corresponding numerical value.
- 15 3. A method as claimed in claim 2 wherein the value is a Unicode numerical
code.
4. A method as claimed in any one of the preceding claims wherein the step of
inputting the URL uses a character input program .
- 20 5. A method as claimed in claim 4 wherein the program is downloaded from a
remote location at which the determining, querying and fetching steps are
performed.
- 25 6. A method as claimed in claim 5 wherein the remote location comprises a
website.
7. A method as claimed in any one of claims 4 to 6 wherein the program
displays a URL entry toolbar on an Internet browser of the user.
- 30 8. A method as claimed in any one of the preceding claims wherein the
inputting step uses a method of input of each character selected by the user.

9. A method as claimed in any one of the preceding claims wherein each character is identified by consulting a database at a remote location.

5 10. A method as claimed in claim 9 wherein the database includes a graphic representation of each character, the graphic representation of the identified character being sent to the user for display.

10 11. A method as claimed in any one of claims 4 to 6 wherein the identified characters are stored together to form an identified version of the URL for use in the determining step.

15 12. A method as claimed in any one of the preceding claims wherein, in the determining step, more than one name server supporting domain names in the selected language is determined.

13. A method as claimed in claim 8 wherein each name server is queried with the domain.

20 14. A method as claimed in claim 8 wherein only one or more name servers selected by the user are queried.

25 15. A method as claimed in any one of claims 7 to 10 wherein the name servers are queried in an order selected by the user.

16. A method as claimed in any one of claims 7 to 11 wherein, in the determining step, a list of name servers is queried.

30 17. A method as claimed in claim 16 wherein the list is held by the user.

18. A method as claimed in claim 16 wherein the list is held on a remote server.

19. A method as claimed in any one of the preceding claims further comprising the step of searching the fetched contents of the URL and replacing a non-Latin character URL found therein with means for resolving the found URL.

5 20. A method as claimed in claim 19 wherein said means comprises replacing the found URL with a JavaScript function call and passing the parameter of the found URL to the function call.

21. A method of resolving a non-Latin character URL, having a domain name,
10 entered by a user using an Internet browser comprising the steps of:
contacting a website supporting non-Latin character URL resolution,
downloading a non-Latin character URL input program from the website,
inputting the URL using the program and sending the URL to the website;
determining at the website, using the character set of the URL, one or more
15 name servers that support domain names in the selected character set;
querying from the website at least one said name server with the domain
name; fetching contents from the URL if the query is satisfied and passing
the contents to the user.

20 22. Apparatus for performing the method of any one of the preceding claims.