Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2513016 A1 2007/01/22

(21) 2 513 016

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2005/07/22

(41) Mise à la disp. pub./Open to Public Insp.: 2007/01/22

(51) Cl.Int./Int.Cl. *H04L 9/28* (2006.01), *H04L 29/02* (2006.01), *H04L 7/00* (2006.01), *H04L 9/32* (2006.01), *H04Q 7/22* (2006.01)

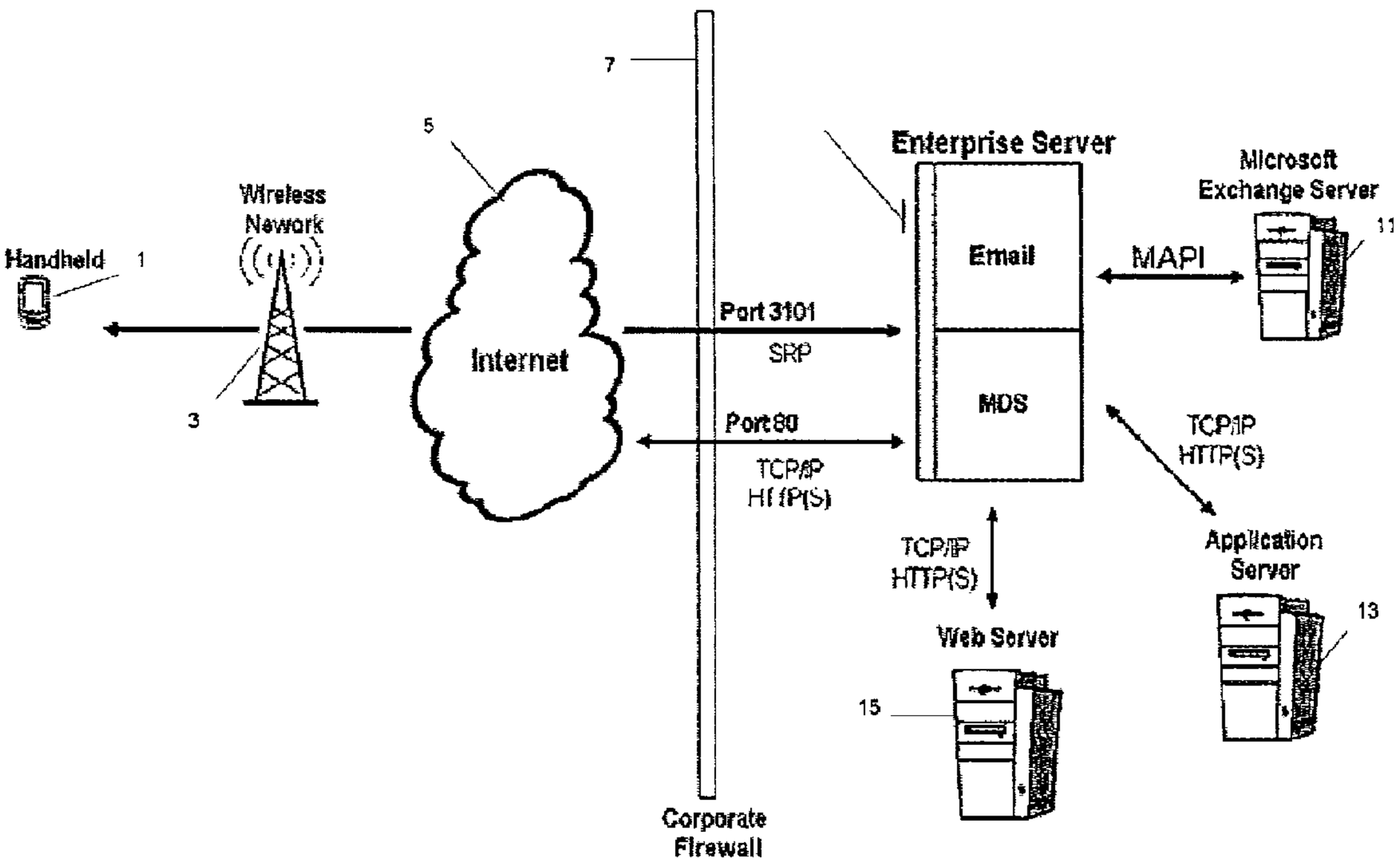
(71) Demandeur/Applicant:
RESEARCH IN MOTION LIMITED, UNKNOWN

(72) Inventeur/Inventor:
UNKNOWN, UNKNOWN

(74) Agent: PERRY + PARTNERS

(54) Titre : METHODE SECURISEE DE SYNCHRONISATION DE CONTENU DE CACHE D'UN NAVIGATEUR MOBILE AVEC UN SERVEUR MANDATAIRE

(54) Title: A SECURE METHOD OF SYNCHRONIZING CACHE CONTENTS OF A MOBILE BROWSER WITH A PROXY SERVER




(57) Abrégé/Abstract:
A secure method of synchronizing cache contents of a mobile browser with a proxy server includes initiating a session between the browser and proxy server, including transmission of browser state information regarding the cache contents and an authentication key to the proxy server; maintaining a record of data sent from the proxy server to the browser for storage in the cache; maintaining a record of the state information regarding the cache contents transmitted from the browser to the proxy server; and transmitting data requests from the browser to the proxy server, in response to which the proxy server uses the key as a seed generation function and accesses each the record of data and returns only data that does not already form part of the cache contents, and wherein the data includes a result of a hash of data generated by the generation function for authentication by the browser before updating the cache contents with the data.

## ABSTRACT

A secure method of synchronizing cache contents of a mobile browser with a proxy server includes initiating a session between the browser and proxy server, including transmission of browser state information regarding the cache contents and an authentication key to the proxy server; maintaining a record of data sent from the proxy server to the browser for storage in the cache; maintaining a record of the state information regarding the cache contents transmitted from the browser to the proxy server; and transmitting data requests from the browser to the proxy server, in response to which the proxy server uses the key as a seed generation function and accesses each the record of data and returns only data that does not already form part of the cache contents, and wherein the data includes a result of a hash of data generated by the generation function for authentication by the browser before updating the cache contents with the data.

# A SECURE METHOD OF SYNCHRONIZING CACHE CONTENTS OF A MOBILE BROWSER WITH A PROXY SERVER

## Copyright Notice



## Field

**[0002]** This specification relates generally to mobile data communication systems, and more particularly to a method for communicating state information between a data server and a mobile Internet browser

## Background

**[0003]** Mobile communication devices are becoming increasingly popular for business and personal use due to a relatively recent increase in number of services and features that the devices and mobile infrastructures support. Handheld mobile communication devices, sometimes referred to as mobile stations, are essentially portable computers having wireless capability, and come in various forms. These include Personal Digital Assistants (PDAs), cellular phones and smart phones.

**[0004]** It is known in the art to provide Internet browser functionality in such mobile communication devices. In operation, a browser user-agent in the handheld mobile communication device issues commands to an enterprise or proxy server implementing a Mobile Data Service (MDS), which functions as an acceleration server for browsing the Internet and transmitting text and images to the mobile device for display. Such enterprise or proxy servers generally do not store the state of their clients (i.e. the browser user-agent), or if they do, the state that is stored is minimal and limited to HTTP state (i.e. cookies). Typically, such enterprise or proxy servers fetch and transmit data to the browser user-agent when the browser makes a data request. In some enterprise or proxy servers, in order to improve the performance of the browser on the mobile device, the enterprise or proxy server fetches all the data required in order to fulfill the data request from the browser, aggregates the fetched data, and transmits the data to the device browser. For instance, if a hyper-text markup language (HTML) page is requested, the enterprise or proxy server would fetch any additional files referenced within the HTML page (ex: Images, inline CSS code, JavaScript, etc.). Since the proxy server fetches all the additional files within the HTML file, the device does not have to make additional data requests to retrieve these additional files. Although

this methodology is faster than having the device make multiple requests, the proxy server nonetheless has to send all of the data again if the site is later revisited. This is because the proxy server has no knowledge of the device cache (e.g. caches that are saved in persistent memory, for different types of data such as a content cache to store raw data that is cached as a result of normal browser activity, a channel cache containing data that is sent to the device by a channel or cache push, and a cookie cache containing cookies that are assigned to the browser by visited Web pages). For example, if a user browses to CNN.com, closes the browser to perform some other function (e.g. place a telephone call or access e-mail messages, etc.) and then later accesses the CNN.com web site (or follows a link from CNN.com to a news story), the banner "CNN.com" will be transmitted from the MDS to the device of browser each time the site is accessed, thereby consuming significant bandwidth, introducing delay, etc.

**[0005]** Thus, it would be desirable to provide a control channel for conveying information about the browser's current session, such as cookie management and cache state, to a proxy server, in a wireless mobile environment.

**[0006]** It is known in the art to provide local file caching. One approach is set forth in *GloMop: Global Mobile Computing By Proxy*, published September 13, 1995, by the GloMop Group, wherein PC Card hard drives are used as portable file caches for storing, as an example, all of the users' email and web cache. The user synchronizes the file cache and the proxy server keeps track of the contents. Mobile applications (clients) are able to check the file cache before asking for information from the proxy server by having the server verify that the local version of a given file is current.

## Summary

**[0007]** A method is set forth herein for communicating information between an enterprise or proxy server and a mobile Internet browser. The method invokes an out-of-band protocol, referred to herein as Browser Session Management (BSM) protocol, for providing a control channel between the proxy server, and a mobile communication device, so that the device can communicate to the server what data the device has stored in memory (from previous browsing). Meta data transmitted by the device is then used by the server when handling subsequent requests from the browser agent, to determine what data to send to the device, thereby reducing data transfer on subsequent requests by over half relative to the prior art methodology discussed above.

**[0008]** Because the proxy server is aware of what the browser has stored in its cache the amount of data sent to the browser may be reduced, thereby increasing the performance of the browser and reducing operational cost. For the example given above, if after the first

request the CNN.com banner is cached and if the proxy server "knows" that the information has been cached then there will be no need to send the CNN.com banner to the device upon subsequent visits to the CNN web site.

**[0009]** According to another aspect, messages from the device to the proxy server contain hash values of data (rather than the actual data) which are used by the server to detect state changes in the device and utilize that information in preparing a document for transmission to the device. A person of skill in the art will appreciate that a one-way hash function transforms data into a value of fixed length (hash value) that represents the original data. Ideally, the hash function is constructed so that two sets of data will not generate the same hash value. Examples of known hash functions include MD2, MD5 and SHA-1.

**[0010]** According to another aspect, each portion of the document downloaded from the server is authenticated by the device before adding such portion of the document to the device cache. This prevents a third party from creating their own document and sending it to the device for injecting cache entries that could be used to extract personal information from the user.

**[0011]** Therefore in general, there is provided a secure method of synchronizing cache contents of a mobile browser with a proxy server includes initiating a session between the browser and proxy server, including transmission of browser state information regarding the cache contents and an authentication key to the proxy server; maintaining a record of data sent from the proxy server to the browser for storage in the cache; maintaining a record of the state information regarding the cache contents transmitted from the browser to the proxy server; and transmitting data requests from the browser to the proxy server, in response to which the proxy server uses the key as a seed generation function and accesses each the record of data and returns only data that does not already form part of the cache contents, and wherein the data includes a result of a hash of data generated by the generation function for authentication by the browser before updating the cache contents with the data.

**[0012]** In contrast to the prior art *GloMop* caching methodology discussed above, the method set forth herein synchronizes the cache contents when the browser connects to the proxy in order to initiate a session, and keeps track of changes to the cache via knowledge of what data has been sent to the browser in combination with state information periodically received from the browser identifying what has actually been cached. Also, as set forth in greater detail below, the proxy server uses this cache knowledge to determine what to send back to the browser. In contrast, the prior art *GloMop* methodology does not contemplate sending any state information to the proxy server for identifying what has actually been cached in the device. Moreover, the prior art *GloMop* approach first checks the local cache,

and then queries the proxy server to determine whether a particular data item in the cache is current or not. According to the *GloMop* prior art, the proxy server does not use it's own knowledge of the browser cache to determine what to send back to the browser.

**[0013]** Additional aspects and advantages will be apparent to a person of ordinary skill in the art, residing in the details of construction and operation as more fully hereinafter described and claimed, reference being had to the accompanying drawings.

## Brief Description of the Drawings

**[0014]** A detailed description of the preferred embodiment is set forth in detail below, with reference to the following drawings, in which:

**[0015]** Figure 1 is a block diagram of a communication system for implementing Internet browsing functionality in a mobile communication device; and

**[0016]** Figure 2 is a flow chart showing the method for communicating information between a mobile data server and a mobile Internet browser, according to the preferred embodiment.

## Detailed Description

**[0017]** Figure 1 depicts the architecture of a system for providing wireless e-mail and data communication between a mobile device 1 and an enterprise or proxy server 9. Communication with the device 1 is effected over a wireless network 3, which in turn is connected to the Internet 5 and proxy server 9 through corporate firewall 7. When a new message is received in a user's mailbox within email server 11, enterprise or proxy server 9 is notified of the new message and copies the message out to the device 1 in a push-based format. Server 9 also provides access to data on an application server 13 and a Web server 15 via a Mobile Data Service (MDS). Additional details regarding e-mail messaging, MAPI sessions, attachment service, etc., are omitted from this description as they are not germane. Nonetheless, such details would be known to persons of ordinary skill in the art.

**[0018]** In terms of web browsing functionality, the device 1 communicates with an enterprise or proxy server 9 using HTTP over an IP protocol optimized for mobile environments. In some embodiments, the device 1 communicates with the proxy server 8 using HTTP over TCP/IP. The communication between the device 1 and server 9 is encrypted with Triple Data Encryption Standard (Triple DES), as is known in the art. The proxy server 9 enables Internet access, preprocesses and compresses HTML and XML content from the Web server 15 before sending it to the device 1, transcodes content type, stores HTTP cookies on behalf of the device 1, and supports certificate authority

authentications, etc.

**[0019]** In response to a request from the device browser, the proxy server 9 retrieves content from Web server 15 and creates a custom document containing both images to be displayed on the device and data in the form of compressed versions of requested portions of the document. The document is preferably of "multi-part" format to improve transmission to and processing efficiency within the device 1. Specifically, in order to display composite Web pages (i.e. pages composed of a main WML or HTML page and one or more related auxiliary files, such as style sheets, JavaScript files, or image files) the device browser is normally required to send multiple HTTP requests to the server 9. However, according to the multi-part generation feature the server 9 posts all necessary parts of a composite Web page in a single bundle, enabling the browser to download all the required content with a single request. The header in the server response identifies the content as a multi-part bundle.

**[0020]** In order to indicate device browser state information to the proxy server 9, three transitional state messages are defined herein, as follows: CONNECT, UPDATE and DISCONNECT, each of which is referred to as a "browser message" in step 21 of the flowchart depicted in Figure 2.

**[0021]** The CONNECT transitional message creates a new session with a connection identifier carried in the payload, device information and state data (e.g. current cache and device information) in the form of a hash function for use by the server in preparing a response.. Specific care is taken not to identify to the proxy server 9 what cookies or cache entries are contained on the device 1. Only hash values of the state data are sent to the proxy server 9 in order to protect the identity of state data on the device 1.

**[0022]** The CONNECT message also contains a unique key for a generating a MAC (Message Authentication Code) using an algorithm (HMAC) that incorporates a cryptographic hash function in combination with the secret key. Each portion of a mulit-part document from the server 9 also contains a MAC used for authenticating the proxy server 9 before adding that portion to the device cache. This prevents a third party from creating their own multi-part document and sending it to the device 1 for injecting cache entries that could be used to extract personal information from the user.

**[0023]** Upon receipt of the CONNECT message, the proxy server 9 uses the state information to regulate or control the transmission of content retrieved from Web server 15 (step 23) to the device 1. One example of an application where this information can be used is when the proxy server 9 is pre-fetching images, inline Cascading Style Sheets (CSS), JavaScript, and the like for a html document. If the proxy server 9 already knows that the device 1 has the image, inline CSS, or JavaScript document, there is no need for resending

the documents.

**[0024]** The UPDATE transition message notifies the proxy server 9 of changes that have occurred on the device 1 since the last connect session between the device 1 and server 9 (e.g. new cache entries added because of a push, or invoking the "Low Memory Manager" (LMM) on the device and purging items from the cache).

**[0025]** The DISCONNECT transition message notifies the proxy server 9 that the device 1 will no longer send any more messages using the connection identifier specified in the payload. The proxy server 9 can then de-allocate any memory reserved for the connect session between the device 1 and server 9. Upon receiving the disconnect message, the proxy server 9 deletes any session cookies for the device 1 (if it is processing cookies) along with state information. Receiving a request on the identified connection after the DISCONNECT has been received is defined as an error.

**[0026]** Since state is indicated from the device 1 to the proxy server 9, and state may be stored in transient memory within server 9, a mechanism is provided for the proxy server 9 to return to the device 1 a message indicating that the session the device is trying to use is not valid. Once this occurs, the device 1 issues a new CONNECT message and establishes a new session with the server 9, and re-issues the original request.

**[0027]** The data protocol set forth herein is similar to HTTP in order to reduce complexity and to reuse code that already exists for the HTTP protocol. Thus, data transmission according to this protocol begins with a STATE keyword; followed by a BSM (Browser Session Management) protocol identifier and a "Content-Length" header. The end of the "headers" is indicated by a double CRLF (a sequence of control characters consisting of a carriage return (CR) and a line feed (LF)), much like HTTP. After the double CRLF pair (i.e. \r\n) a WBXML (WAP Binary Extensible Markup Language) encoded document is inserted as the message payload. The WBXML document is later decoded using a DTD (Document Type Definition) and codebook, as discussed in greater detail below. The indication of the protocol version refers to what version of the DTD to validate the request against (ie. BSM/1.1 stipulates using version 1.1 of the DTD).

**[0028]** The following is an example communication using the protocol according to the preferred embodiment:

```
CONNECT BSM/1.0\r\n
Content-Length: 40\r\n
\r\n
<WBXML Encoded document of length 40 bytes>

BSM/1.0 200\r\n
r\n
```

**[0029]** In the foregoing, the first four lines form the CONNECT message from the device 1 to the proxy server 9, and the last two lines are the response from the proxy server 9.

**[0030]** An exemplary XML document, is as follows:

```
<?xml version="1.0"?>
<!DOCTYPE bsm PUBLIC "-//RIM//DTD BSM 1.0//EN"
          "http://www.blackberry.com/go/mobile/BSM/bsm_1.0.xml">
<bsm id="2" hmac="12345678901234567890">
<cache>
<size>123012</size>
<entry urlHash="FEEDDEED01" etag="SomeEtag" expiry="256712323"/>
</cache>
<device>
<version>4.0.1.123</version>
<memfree>12342342</memfree>
</device>
</bsm>
```

**[0031]** In the example, the state data includes the URL of a html page within the device cache. It will be noted that the XML document payload includes a connection identifier (i.e. bsm id="2"), a value indicating when the document was last modified (i.e. etag="SomeEtag"), a page expiry (i.e. expiry="256712323"), and hash values for a URL (i.e. entry urlHash="FEEFFEEF01") rather than transmitting the actual URL itself. Thus, in operation, the hash of the URL of the cached page is sent to the proxy server 9 in the CONNECT string (step 21). The server 9 then fetches the requested page from Web server 13 (step 23) and compares a hash of the URL of the requested page with the hashed URL of the cached page, and also compares the time stamps/expiration information in order to determine whether the cached page is current. The server 9 sends the new page (or component of a page if the URL references an image) only if the cached page (or component) is not current within the device cache. If there has been no change in state (i.e. the hash values are found to be equal at step 29), then the proxy server merely indicates to the device 1 that the content has already been cached (step 31).

**[0032]** Although not indicated in Figure 2, it will be appreciated that each inline part to be added to a document to be displayed at the device 1 is fetched. If the response code indicates a "304" then the part is written as a block in the multipart document. On the other hand, if the server 9 returns a "200" then the hash compare operation is performed.

**[0033]** An exemplary DTD, according to the preferred embodiment, is as follows:

```
<!ELEMENT bsm (cache?, device)>
<!ATTLIST bsm
          id       NMTOKEN       #REQUIRED
>
<!ELEMENT cache (size, (entry)+)>
<!ATTLIST cache
          action   (add|remove|remove_all|quick_add)   "add"
>
```

```
<!ELEMENT entry EMPTY>
<!ATTLIST entry
          urlHash          CDATA      #REQUIRED
          etag             CDATA      #IMPLIED
          expiry           NMTOKEN    #IMPLIED
          size             NMTOKEN #IMPLIED
          last-modified NMTOKEN #IMPLIED
>
<!ELEMENT size (#PCDATA)>
<!ELEMENT device (version, memfree)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT memfree (#PCDATA)>
<!ELEMENT hmac (#PCDATA)>
```

Element/Code
HMAC 12

Attribute/Code
size 9 (instead of action)
lastModified 10
actionAdd 11
actionRemove 12
actionRemoveAll 13
actionQuickAdd 14

**[0034]** Finally, an exemplary codebook, is as follows:

| Element | Code |
|---|---|
| Session | 5 |
| Cache | 6 |
| Size | 7 |
| Entry | 8 |
| Device | 9 |
| Version | 10 |
| MemFree | 11 |
| HMAC | 12 |

| Attribute | Code |
|---|---|
| Id | 5 |
| UrlHash | 6 |
| ETag | 7 |
| Expiry | 8 |
| Action | 9 |

**[0035]** As is well known in the art, the codebook is used as a transformation for compressing the XML document to WBXML, wherein each text toke is represented by a single byte from the codebook.

**[0036]** As discussed above, the proxy server 9 transmits multi-part documents in a

proprietary format of compressed HTML, interspersed with data for images and other auxiliary files (which may or may not be related to the main HTML Web page). However, in a departure from conventional HTML, each document part may also include a response code (e.g. "200" for OK, or "304" for "not modified" to indicate that the specified document part has already been cached in the device 1). This may be used for selective downloading of document parts rather than entire documents and for indicating when a part (e.g. image) is about to expire. This is useful, for example, when one web page links to another page containing one or more common elements.

**[0037]** Of course, certain device requests (e.g. page refresh) will always result in a full document download, irrespective of device state information stored in the server 9.

**[0038]** It is contemplated that the inclusion of response codes may be used by heuristic processes within the proxy server 9 to learn user behaviour and modify downloading of documents based on tracking the history of certain changes reflected in the hash value (e.g. the server 9 may learn to download a certain page (e.g. CNN news) at a particular time each day based the user's history of issuing requests for that page at regular times. As discussed above, because the downloaded documents are multi-part and contain embedded response codes, only those portions of the document that have changed are actually downloaded.

**[0039]** As indicated above, the protocol of the preferred embodiment is preferably carried over the IPPP transport layer, but can also be easily adopted to run over TCP/IP on a specific port. The protocol is preferably implemented as a handler in the proxy server 9, thereby simplifying any currently existing protocol. (e.g. to avoid overloading a current HTTP protocol).

**[0040]** A person skilled in the art, having read this description of the preferred embodiment, may conceive of variations and alternative embodiments. For example, the conditional transfer of data based on communication of state information, as set forth above, may also be applied to separately transmitting individual components of the multipart document as opposed to transmitting the entire document at once.

**[0041]** In some embodiments, the proxy server 9 uses heuristic algorithms to learn what additional data requests the device may make based on knowledge of the current request, and knowledge of past activity. In some instances, the device may follow a pattern of requesting a first web page, and then a second web page. For example, the device may first request the "cnn.com" web page, and then request the "cnn.com/news" web page. The proxy server 9 learns this pattern, and whenever the device requests the first web page, the proxy server 9 determines that the device is likely to then request the second web page. The proxy server 9 then fetches the second web page, and uses its knowledge of the data

cached on the device 1 (i.e. From the state information transferred to the proxy server 9 during initiation of the present connection) to determine whether the second web page already exists within the data cached on the device. If so, the proxy server 9 includes information about the second web page via response codes embedded within the response provided for the first web page. If the device 1 requires the second web page, then the device 1 can reference it's cache and can avoid having to make a request to the proxy server 9 for the second web page.

**[0042]** In other embodiments, heuristic processes within the proxy server 9 learn user behaviour and modify downloading of documents based on tracking the history of certain changes reflected in the hash value (e.g. the server 9 may learn to download a certain page (e.g. CNN news) at a particular time each day based the user's history of issuing requests for that page at regular times. As discussed, because the downloaded documents are multi-part and contain embedded response codes, only those portions of the document that have changed are actually downloaded.

**[0043]** All such variations and alternative embodiments are believed to be within the ambit of the claims appended hereto.

What is claimed is:

1. A secure method of synchronizing cache contents of a mobile browser with a proxy server, comprising:

initiating a session between said browser and proxy server, including transmission of browser state information regarding said cache contents and an authentication key to said proxy server;

maintaining a record of data sent from the proxy server to the browser for storage in said cache;

maintaining a record of said state information regarding said cache contents transmitted from the browser to said proxy server; and

transmitting data requests from said browser to said proxy server, in response to which said proxy server uses said key as a seed generation function and accesses each said record of data and returns only data that does not already form part of said cache contents, and wherein said data includes a result of a hash of data generated by said generation function for authentication by said browser before updating the cache contents with said data.

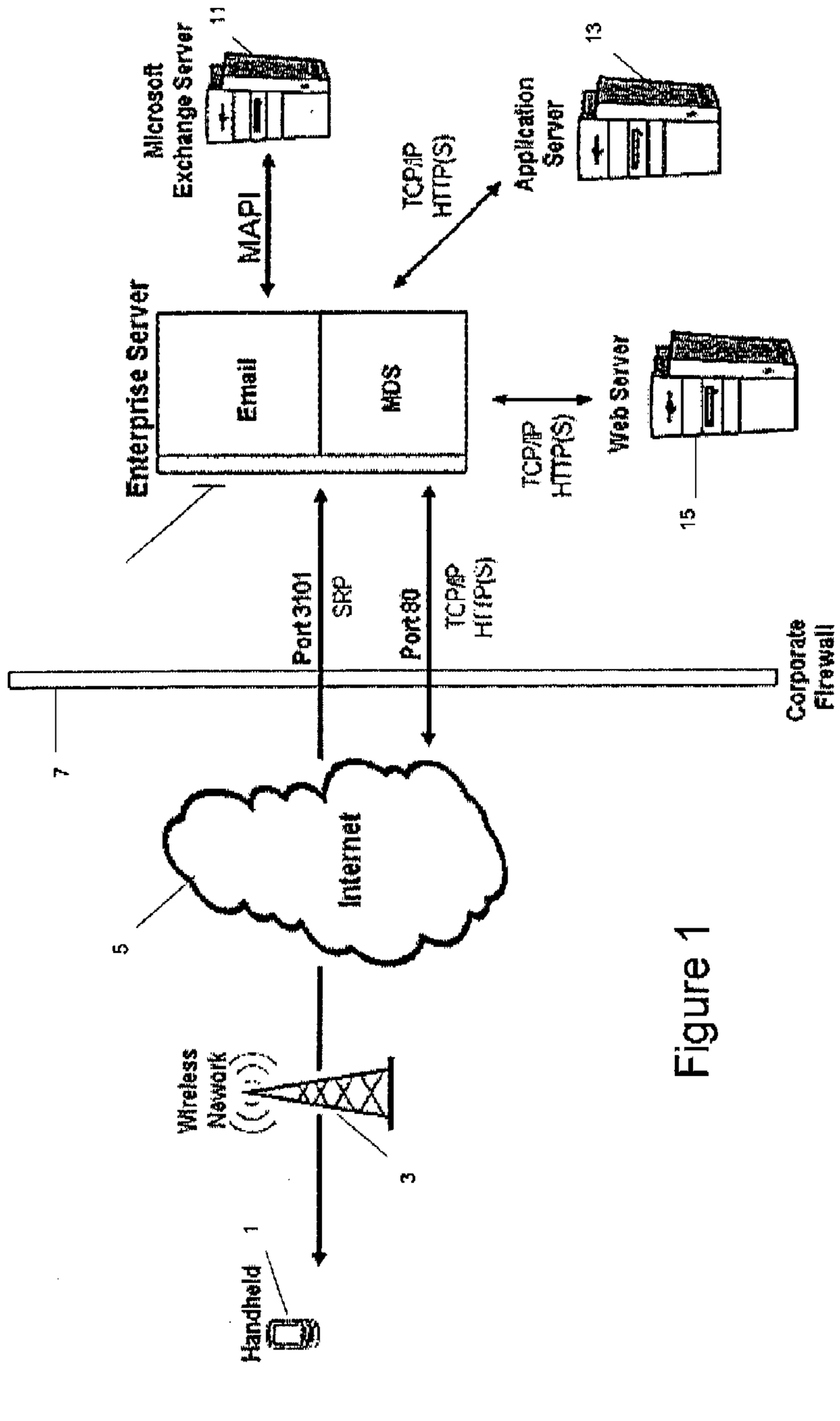

Microsoft Exchange Server
11
MAPI
API
Enterprise Server
Email
MDS
TCP/IP
HTTP(S)
Application Server
13
TCP/IP
HTTP(S)
Web Server
15
Port 3101
SRP
Port 80
TCP/IP
HTTP(S)
Corporate Firewall
7
Internet
5
Wireless Network
3
Handheld
1

Figure 1

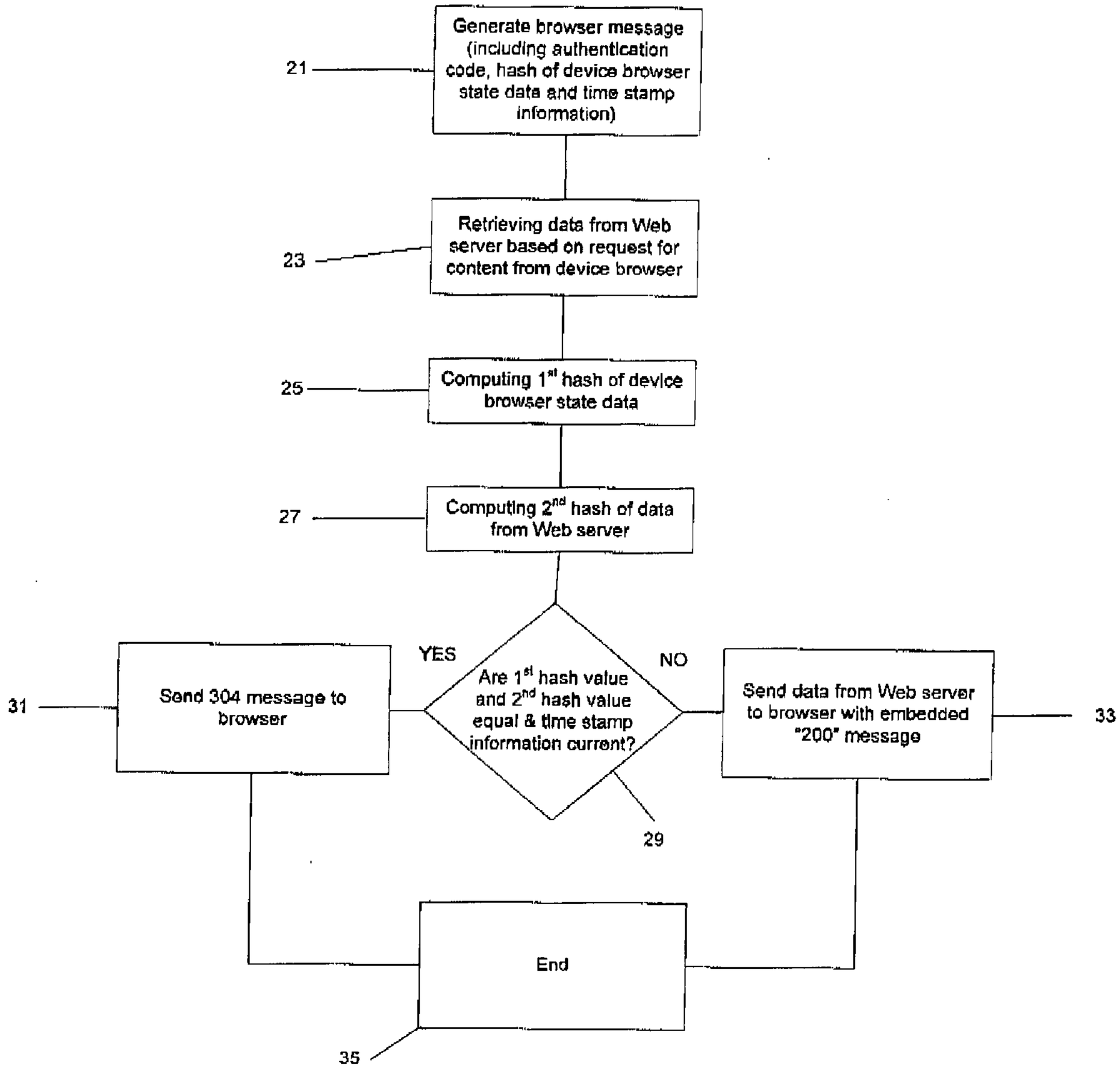
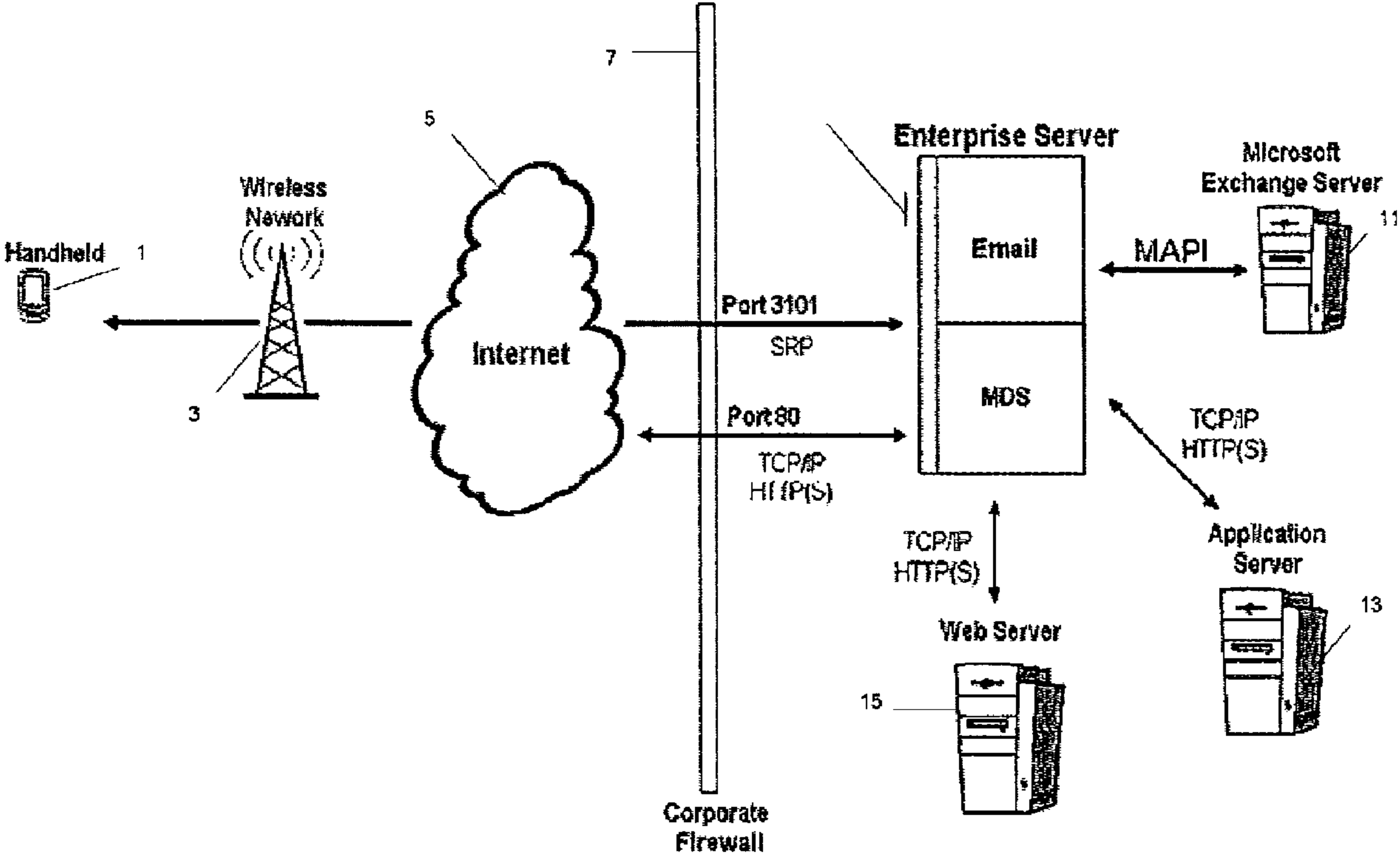

Generate browser message (including authentication code, hash of device browser state data and time stamp information)
21
Retrieving data from Web server based on request for content from device browser
23
Computing 1st hash of device browser state data
25
Computing 2nd hash of data from Web server
27
Are 1st hash value and 2nd hash value equal & time stamp information current?
YES
NO
29
Send 304 message to browser
31
Send data from Web server to browser with embedded "200" message
33
End
35

Figure 2


7
5
Enterprise Server
Microsoft
Exchange Server
11
Wireless
Nework
Handheld
1
Email
MAPI
Port 3101
SRP
Internet
MDS
Port 80
TCP/IP
HTTP(S)
TCP/IP
HTTP(S)
3
TCP/IP
HTTP(S)
Application
Server
13
Web Server
15
Corporate
Firewall