EXHIBIT A

U.S. PATENT NO. 6,233,608

US006233608B1

(12) **United States Patent**
Laursen et al.

(10) **Patent No.: US 6,233,608 B1**
(45) **Date of Patent: *May 15, 2001**

(54) **METHOD AND SYSTEM FOR SECURELY INTERACTING WITH MANAGED DATA FROM MULTIPLE DEVICES**

(75) Inventors: **Andrew L. Laursen**, San Mateo; **Bruce K. Martin, Jr.**; **Alain S. Rossmann**, both of Palo Alto, all of CA (US)

(73) Assignee: **Openwave Systems Inc.**, Redwood City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/320,296**

(22) Filed: **Jun. 7, 1999**

Related U.S. Application Data

(63) Continuation of application No. 08/987,346, filed on Dec. 9, 1997, now Pat. No. 6,065,120.

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **709/217**; 707/10; 713/201
(58) **Field of Search** 713/2, 201, 200; 709/203, 214, 217, 227, 218, 219, 10; 455/550, 556, 557

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,321,840	6/1994	Ahlin et al. .	
5,434,918	7/1995	Kung et al.	380/25
5,610,910 *	3/1997	Focsaneanu	370/351
5,673,322	9/1997	Pepe et al. .	
5,689,642	11/1997	Harkins et al. .	
5,689,825	11/1997	Averbuch et al. .	
5,703,942	12/1997	Pinard et al. .	
5,708,780	1/1998	Levergood et al.	709/229
5,717,923	2/1998	Dedrick .	

(List continued on next page.)

OTHER PUBLICATIONS

Jouni Mikkonen, Matti Turunen, "An Integrated QoS Architecture for GSM Networks", Nokia Wireless Data Library, sqosarchit.pdf, (PDF file, size 72 KB), Oct. 1997.*
Bartlett, Joel F., Digital WRL Technical Note TN–46, "Experience with a Wireless World Wide Web Client," Mar. 1995, pp. 1–7.
Gessler et al., Computer Networks & ISDN Systems, "PDAs as mobile WWW browsers," vol. 28, #1–2, 1995, pp. 53–59.

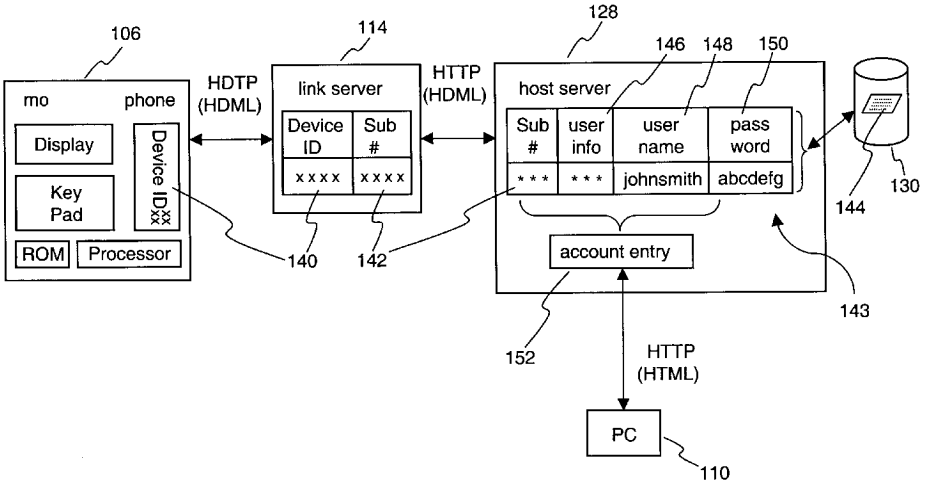
Primary Examiner—Thomas M. Heckler
(74) *Attorney, Agent, or Firm*—Beyer Weaver & Thomas, LLP

(57) **ABSTRACT**

The present invention has been made in consideration of thin devices efficiently communicating ideas and transactions into data networks by using other devices with fill functional user interface in the networks. According to one aspect of the present invention, the thin device exclusively controls the authentication of a rendezvous that is associated with a user account in a server. The thin device running a micro-browser provisions the rendezvous with a set of credential information in an authenticated and secure communication session so that the provisioning process is truly proprietary. To access the user account, the other devices equipped with well known browsers must submit the correct credential information to the rendezvous for verification in the server. Once admitted, the other devices can update managed information in the user account, individually and respectively, thereby the thin device is able to conduct desired transactions based on the managed information in the user account without the need to key in pertinent information of the transactions.

50 Claims, 12 Drawing Sheets

Microfiche Appendix Included
(2 Microfiche, 195 Pages)



U.S. PATENT DOCUMENTS

5,721,827		2/1998	Logan et al. .	
5,727,159		3/1998	Kikinis .	
5,732,074	*	3/1998	Spaur	370/313
5,740,430		4/1998	Rosenberg et al. .	
5,742,905		4/1998	Pepe et al. .	
5,754,939		5/1998	Herz et al. .	
5,764,235		6/1998	Hunt et al. .	
5,796,832		8/1998	Kawan .	
5,802,276	*	9/1998	Benantar	395/186
5,805,159		9/1998	Bertram et al.	707/507
5,805,803		9/1998	Birrell et al.	713/201
5,809,415		9/1998	Rossmann .	
5,815,665		9/1998	Teper et al.	709/229
5,825,759		10/1998	Liu .	
5,828,833		10/1998	Belville et al.	713/201
5,835,577	*	11/1998	Disanto	379/93.19
5,838,682		11/1998	Dekelbaum et al.	370/401
5,844,972		12/1998	Jagadish et al. .	
5,848,161		12/1998	Luneau et al.	380/49
5,857,201		1/1999	Wright, Jr. et al.	707/104
5,862,325		1/1999	Reed et al. .	
5,862,330	*	1/1999	Anupam	395/200.34
5,862,339		1/1999	Bonnaure et al.	709/227
5,867,153		2/1999	Grandcolas et al. .	
5,867,661		2/1999	Bittinger et al. .	
5,884,312		3/1999	Dustan et al.	707/10
5,887,171		3/1999	Tada et al.	709/303
5,890,155	*	3/1999	Steinman	707/9
5,896,444		4/1999	Perlman et al.	379/93.35
5,901,287		5/1999	Bull et al. .	
5,903,845		5/1999	Buhrmann et al. .	
5,905,251		5/1999	Knowles	235/472.01
5,907,547		5/1999	Foladare et al.	370/352
5,918,019		6/1999	Valencia	709/227
5,923,756		7/1999	Shambroom	380/21
5,926,624		7/1999	Katz et al.	705/707
5,926,636		7/1999	Lam et al.	709/303
5,954,799	*	9/1999	Goheen	709/250

OTHER PUBLICATIONS

Kaashock et al., Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, "Dynamic Documents: Mobile Wireless Access to the WWW," Dec. 1994, pp. 1–6.

Kaashock et al., Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, "Dynamic Documents: Extensibility and Adaptability in the WWW," Dec. 1994, pp. 1–10.

Liljeberg et al., SDNE, "Optimizing World–Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach," Jun. 5–6, 1995, pp. 1–8.

Schilit et al., Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Context–Aware Computing Applications, Dec. 1994, pp. 1–7.

Schilit et al., Fifth International World Wide Web Conference, "TeleWeb: Loosely Connected Access to the World Wide Web," May 6–10, 1996, pp. 1–14.

Voelker et al., Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, "Mobisaic: An Information System for a Mobile Wireless Computing Environment," Dec. 1994, pp. 53–59.

* cited by examiner

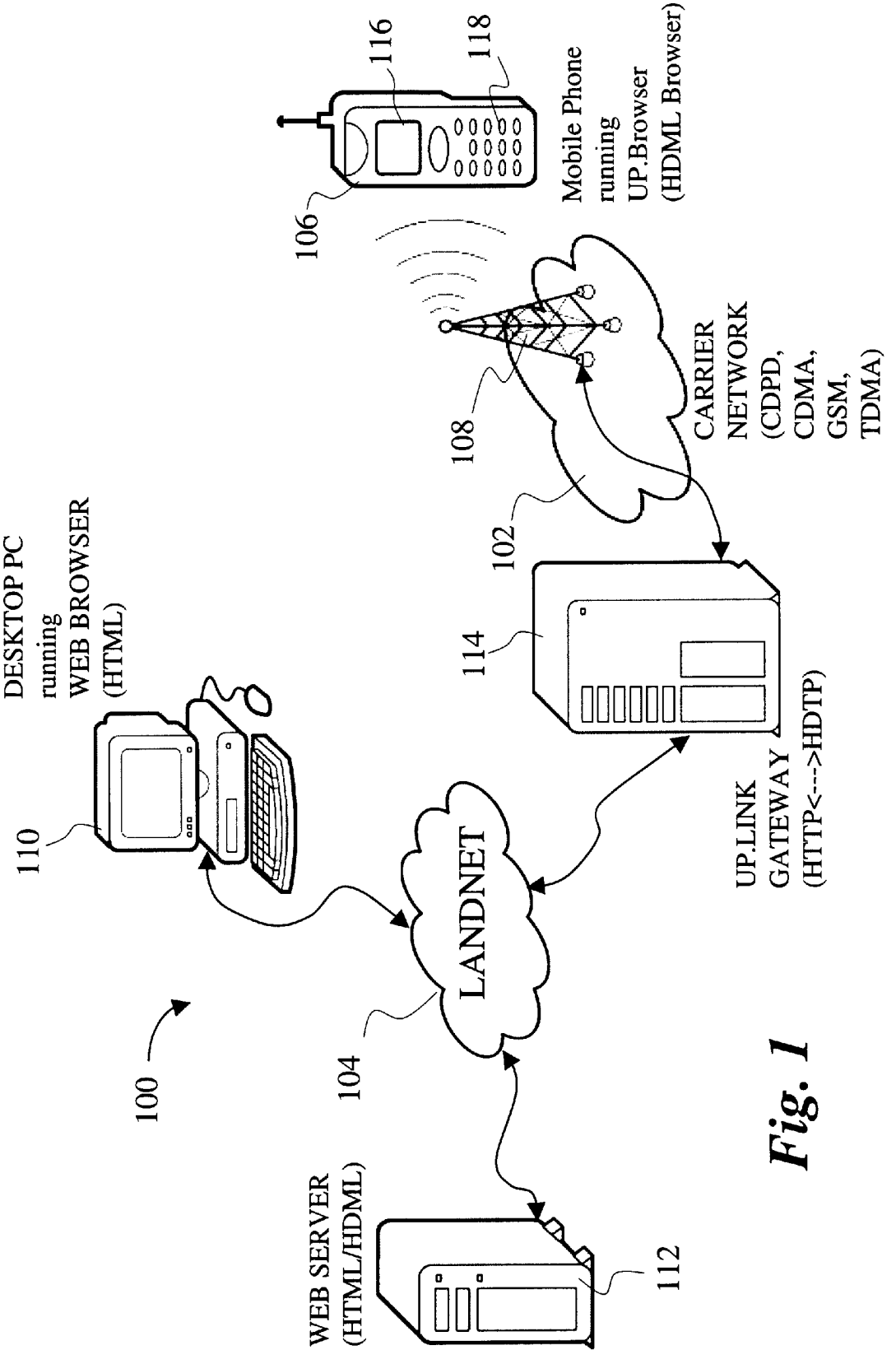

DESKTOP PC
running
WEB BROWSER
(HTML)
110
100
104
LANDNET
WEB SERVER
(HTML/HDML)
112
UP.LINK
GATEWAY
(HTTP<-->HDTP)
114
102
CARRIER
NETWORK
(CDPD,
CDMA,
GSM,
TDMA)
108
106
116
118
Mobile Phone
running
UP.Browser
(HDML Browser)

Fig. 1

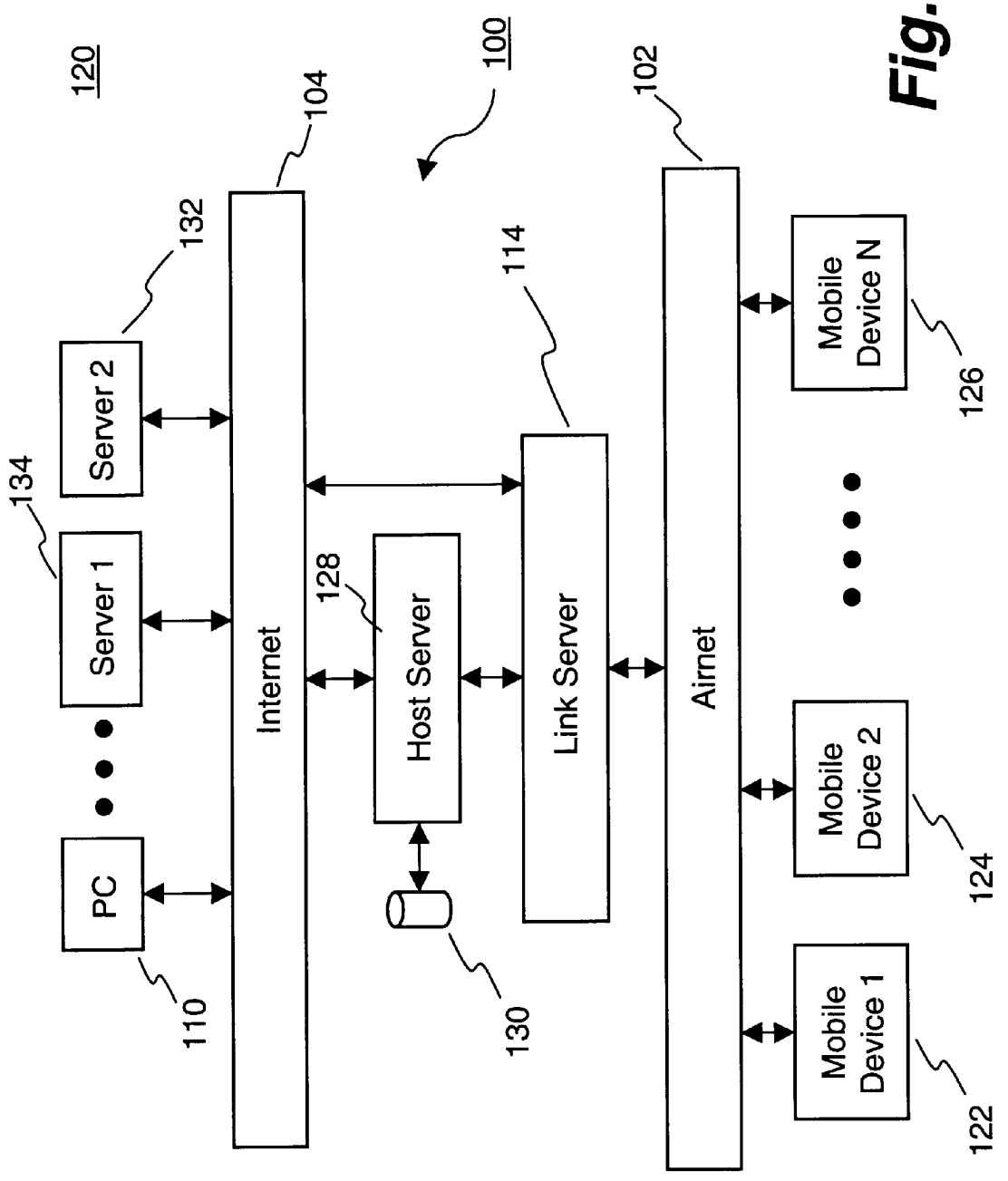

120
100
104
102
132
114
134
Server 2
Server 1
PC
110
Internet
128
Host Server
130
Link Server
Airnet
Mobile Device N
126
Mobile Device 2
124
Mobile Device 1
122

Fig. 2.a

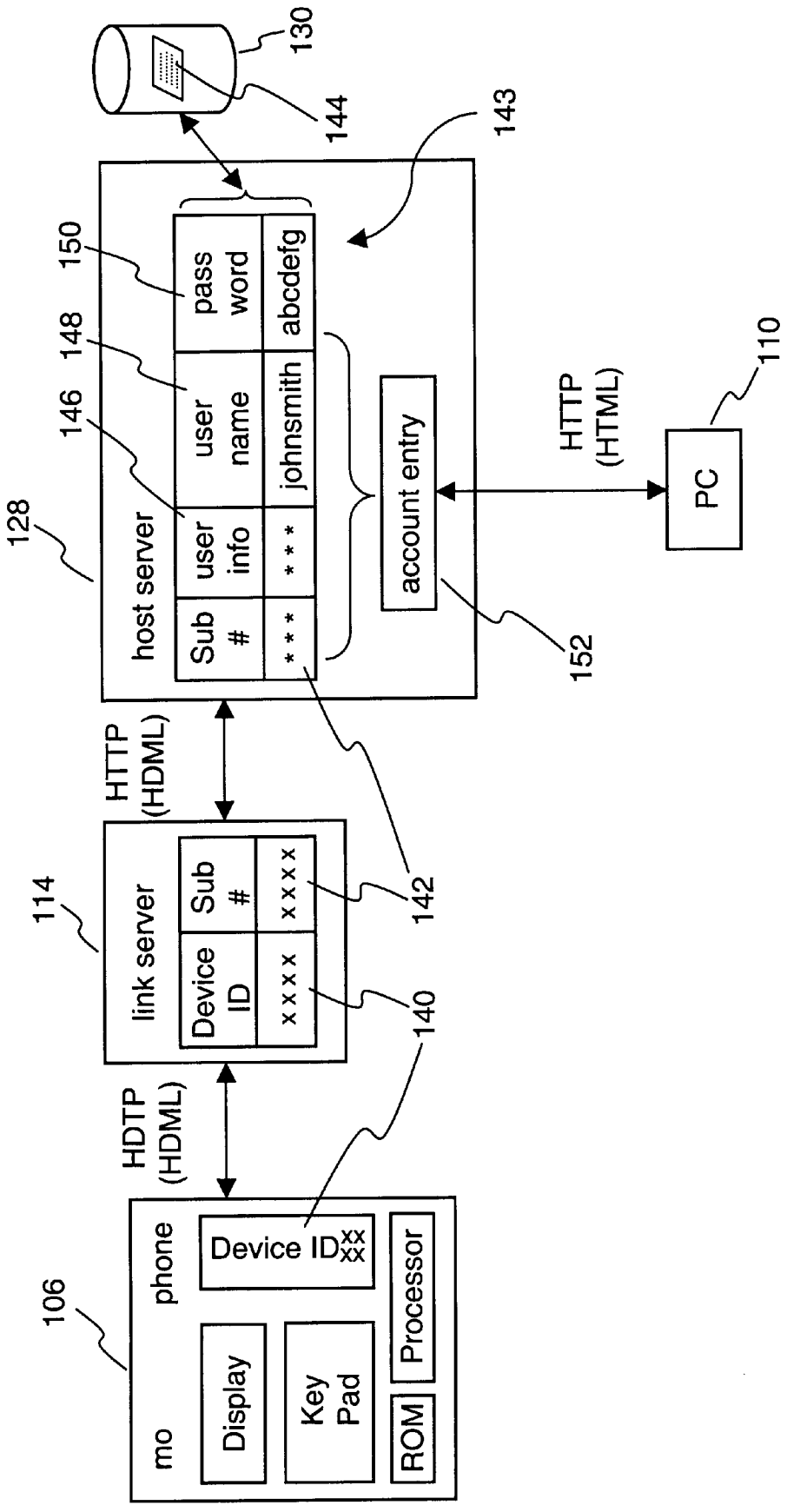

130
144
143
150
148
146
128
host server
Sub #
user info
user name
pass word
* * *
* * *
johnsmith
abcdefg
account entry
152
HTTP (HTML)
110
PC
HTTP (HDML)
114
link server
Device ID
Sub #
x x x x
x x x x
142
140
HDTP (HDML)
106
mo phone
Display
Key Pad
ROM
Processor
Device ID xx xx

Fig. 2.b

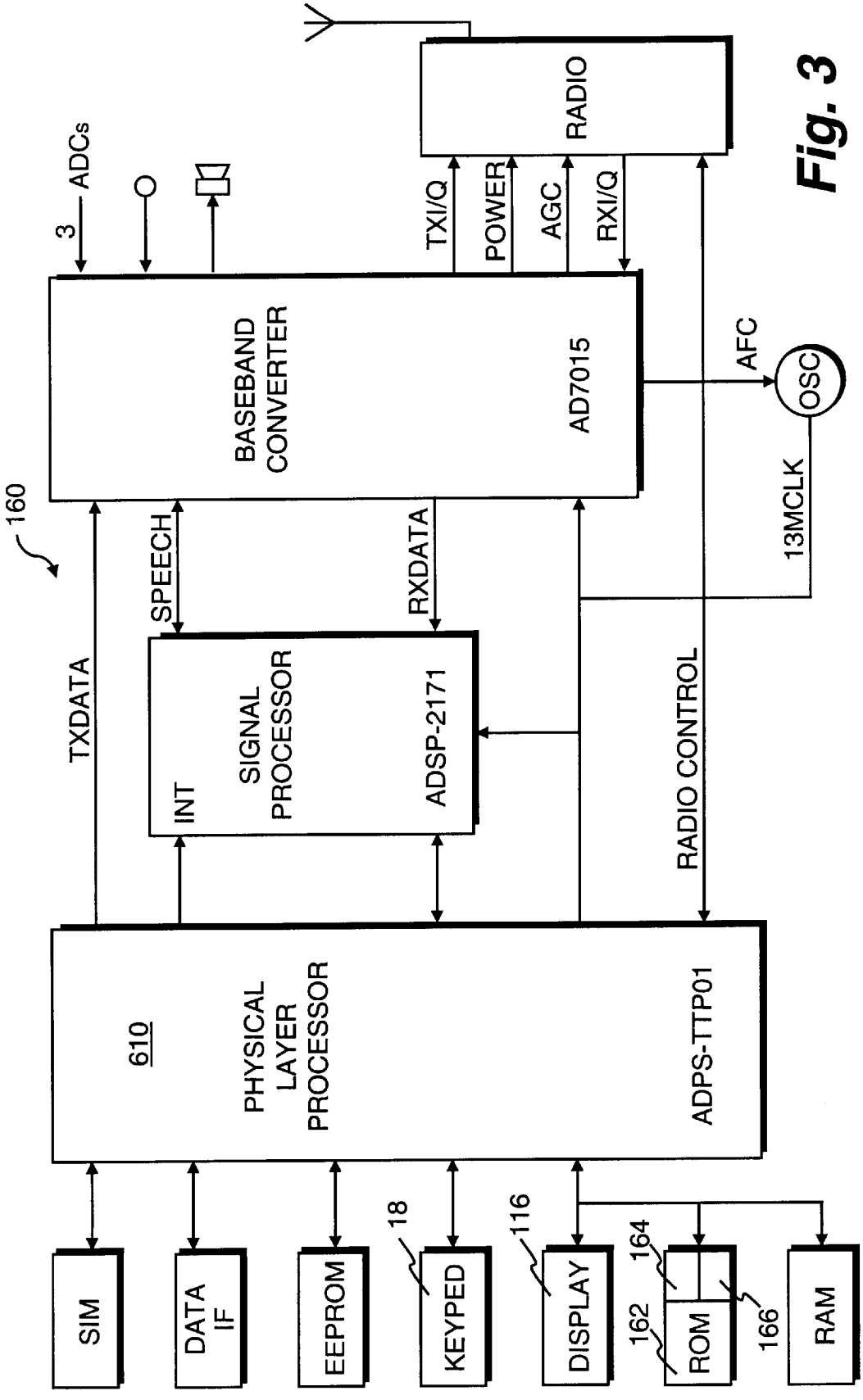

RADIO
TXI/Q
POWER
AGC
RXI/Q
3 ADCs
BASEBAND CONVERTER
AD7015
AFC
OSC
13MCLK
160
SPEECH
RXDATA
TXDATA
INT
SIGNAL PROCESSOR
ADSP-2171
RADIO CONTROL
610
PHYSICAL LAYER PROCESSOR
ADPS-TTP01
SIM
DATA IF
EEPROM
18
KEYPED
116
DISPLAY
164
162
ROM
166
RAM

Fig. 3

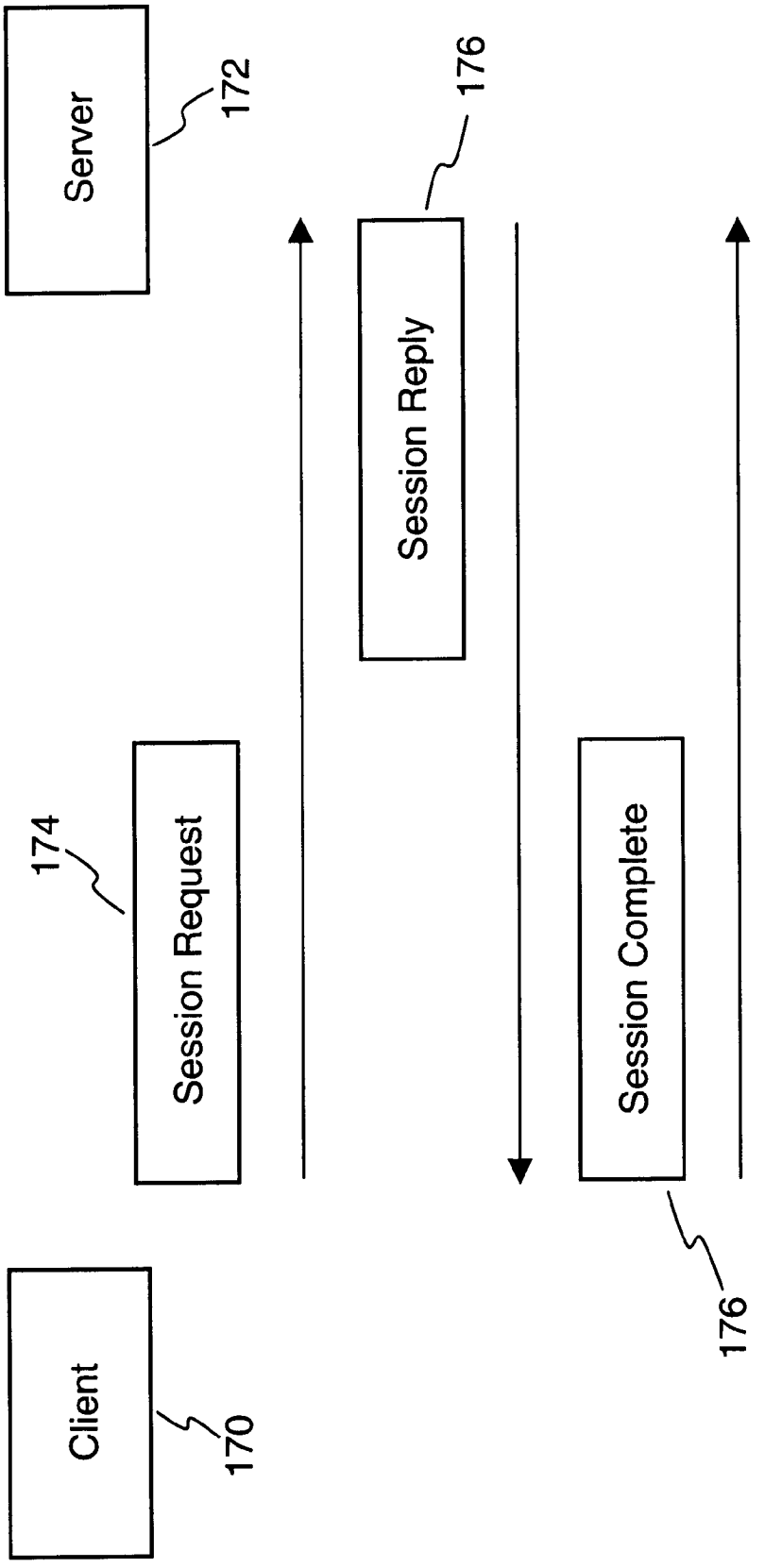

Server
172
176
Session Reply
174
Session Request
Session Complete
Client
170
176

Fig. 4

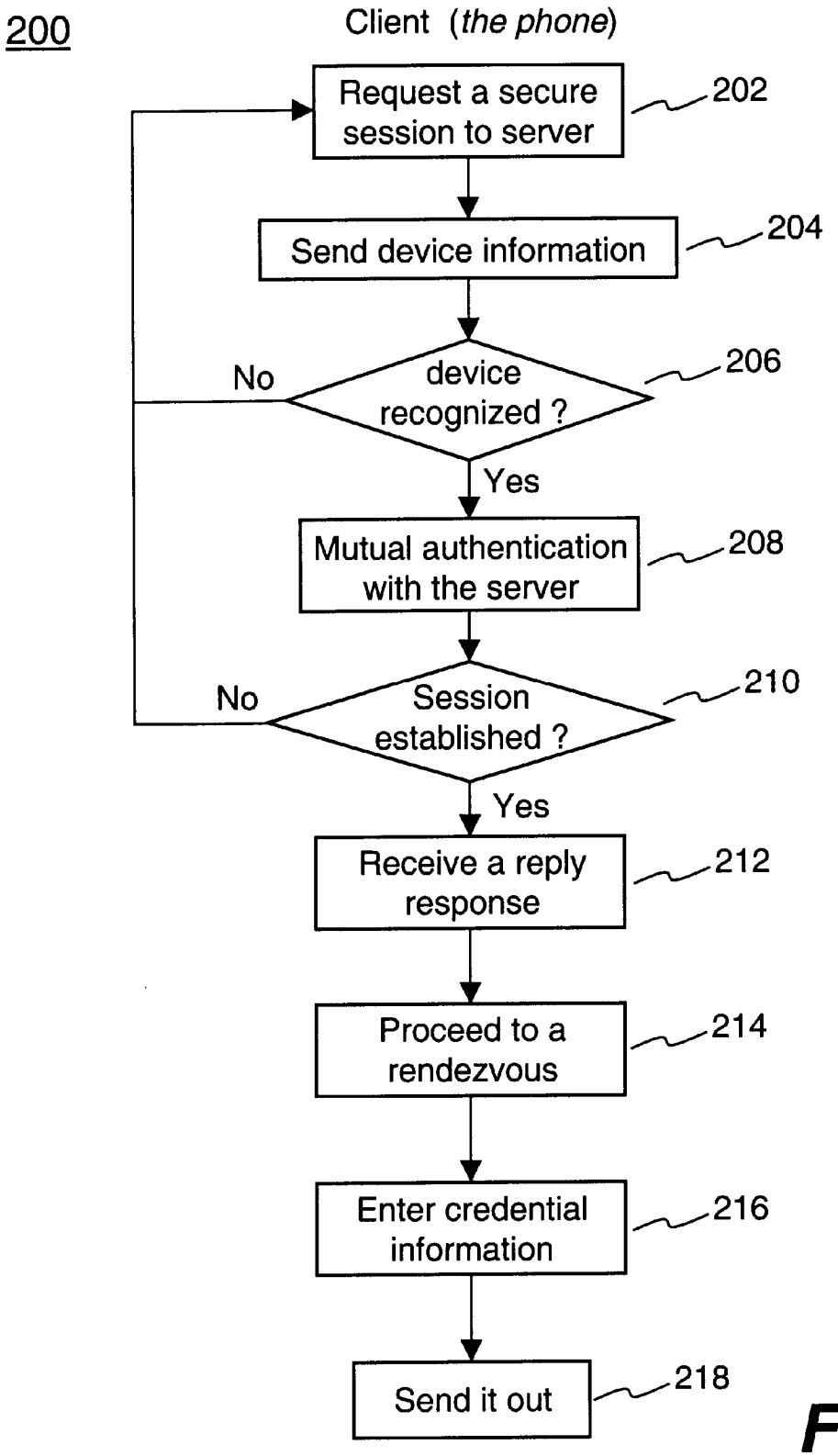

200
Client (the phone)
Request a secure session to server
202
Send device information
204
No
device recognized ?
206
Yes
Mutual authentication with the server
208
No
Session established ?
210
Yes
Receive a reply response
212
Proceed to a rendezvous
214
Enter credential information
216
Send it out
218

Fig. 5.a

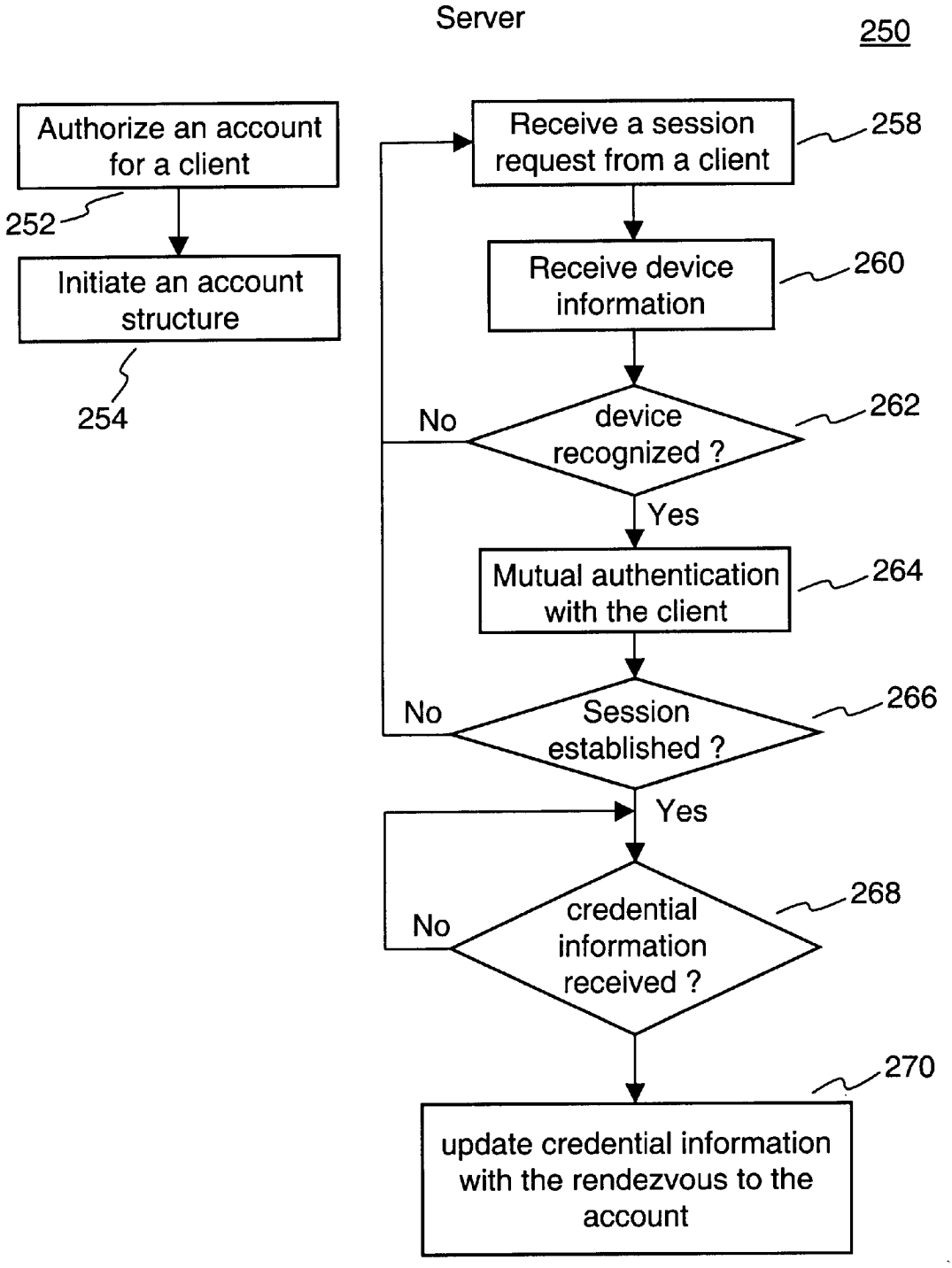

Server
250
Authorize an account for a client
252
Initiate an account structure
254
Receive a session request from a client
258
Receive device information
260
No
device recognized ?
262
Yes
Mutual authentication with the client
264
No
Session established ?
266
Yes
No
credential information received ?
268
270
update credential information with the rendezvous to the account

Fig. 5.b

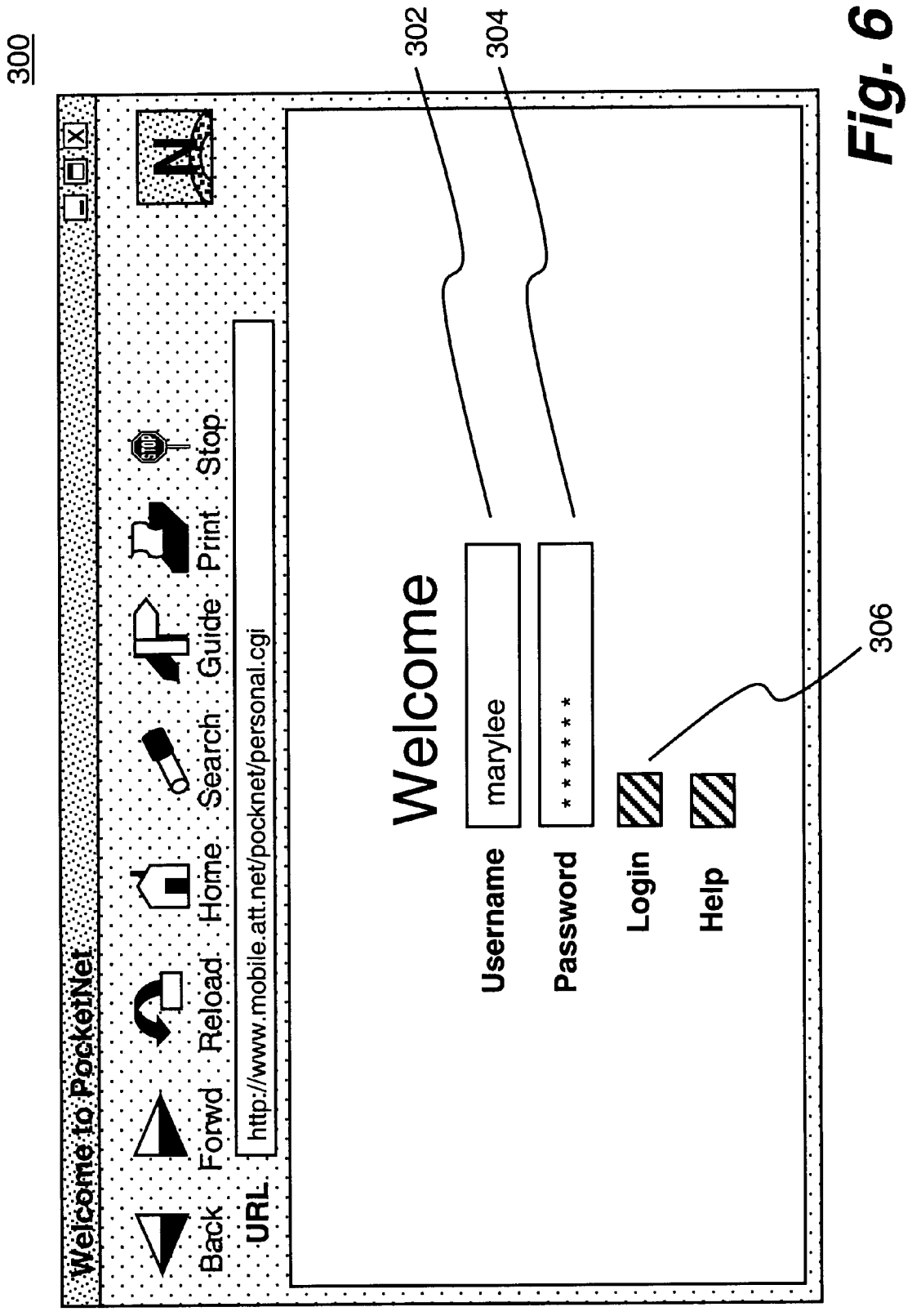

300
302
304
306
Welcome to PocketNet
Back
Forwd
Reload
Home
Search
Guide
Print
Stop
URL
http://www.mobile.att.net/pocknet/personal.cgi
Welcome
Username
marylee
Password
******
Login
Help

Fig. 6

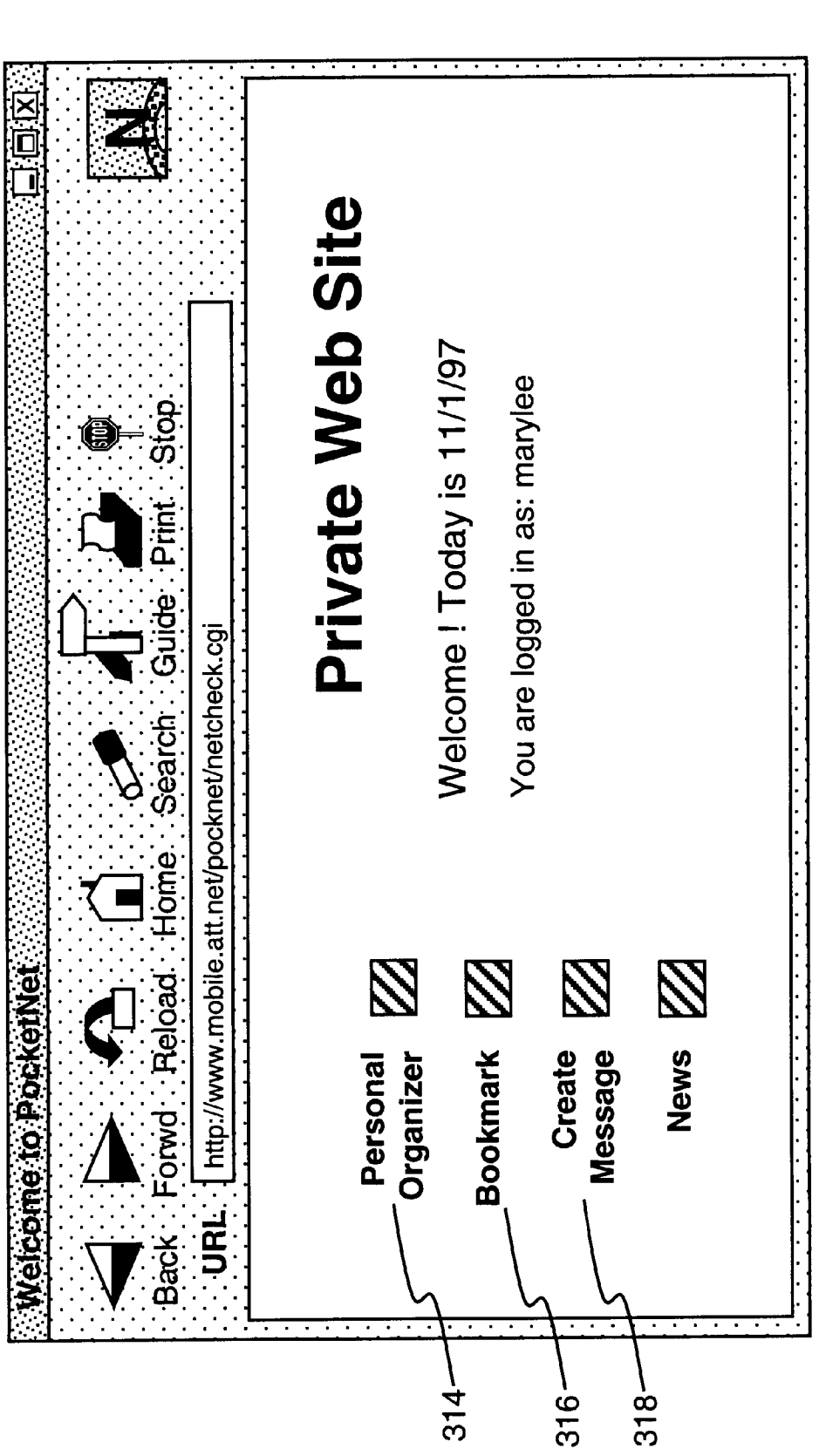

310
Welcome to PocketNet
Back
Forwd
Reload
Home
Search
Guide
Print
Stop
URL
http://www.mobile.att.net/pocknet/netcheck.cgi
Private Web Site
Welcome ! Today is 11/1/97
You are logged in as: marylee
Personal Organizer
Bookmark
Create Message
News
314
316
318

Fig. 7

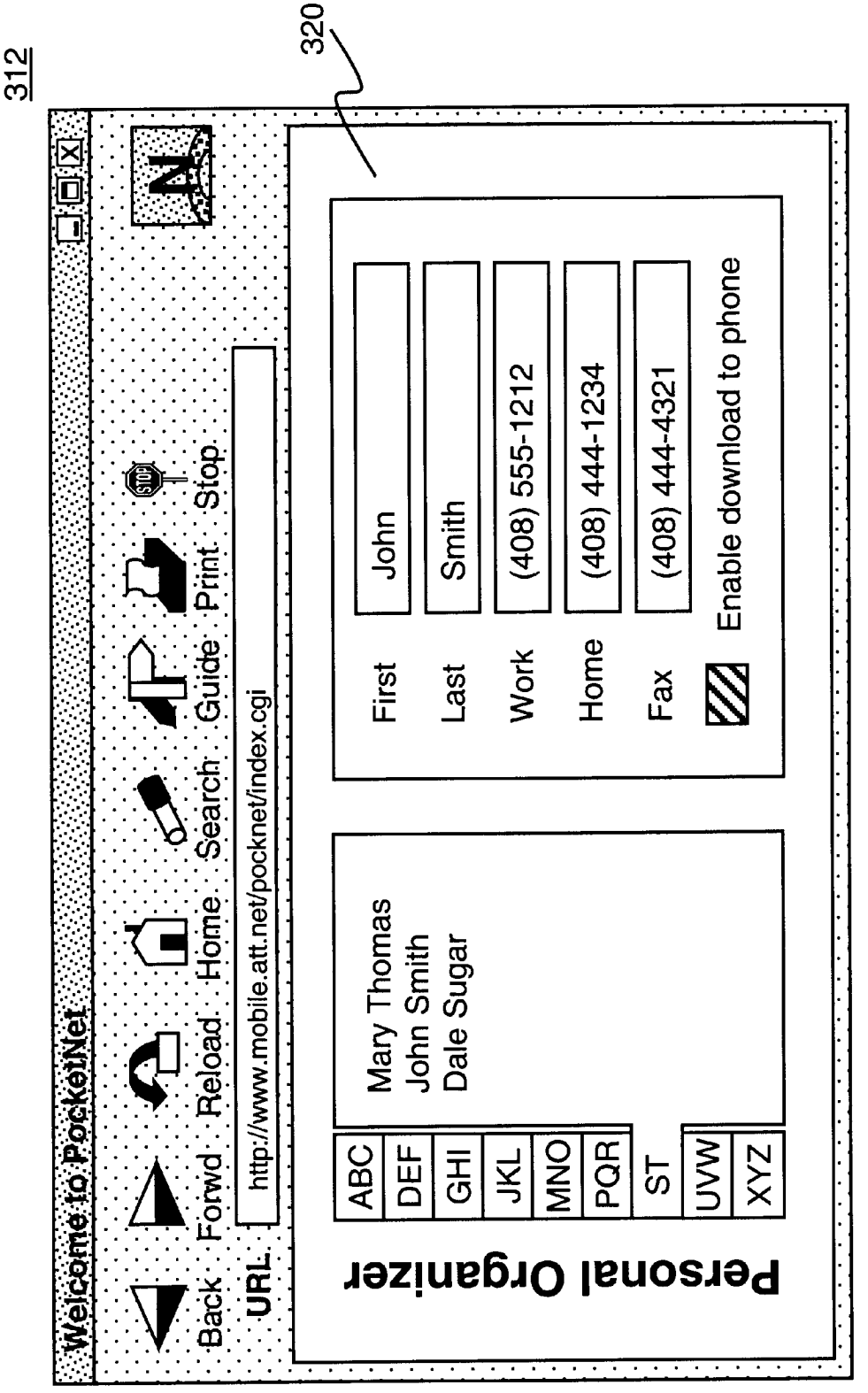

312
320
Welcome to PocketNet
Back
Forwd
Reload
Home
Search
Guide
Print
Stop
URL
http://www.mobile.att.net/pocknet/index.cgi
Personal Organizer
ABC
DEF
GHI
JKL
MNO
PQR
ST
UVW
XYZ
Mary Thomas
John Smith
Dale Sugar
First
John
Last
Smith
Work
(408) 555-1212
Home
(408) 444-1234
Fax
(408) 444-4321
Enable download to phone

Fig. 8

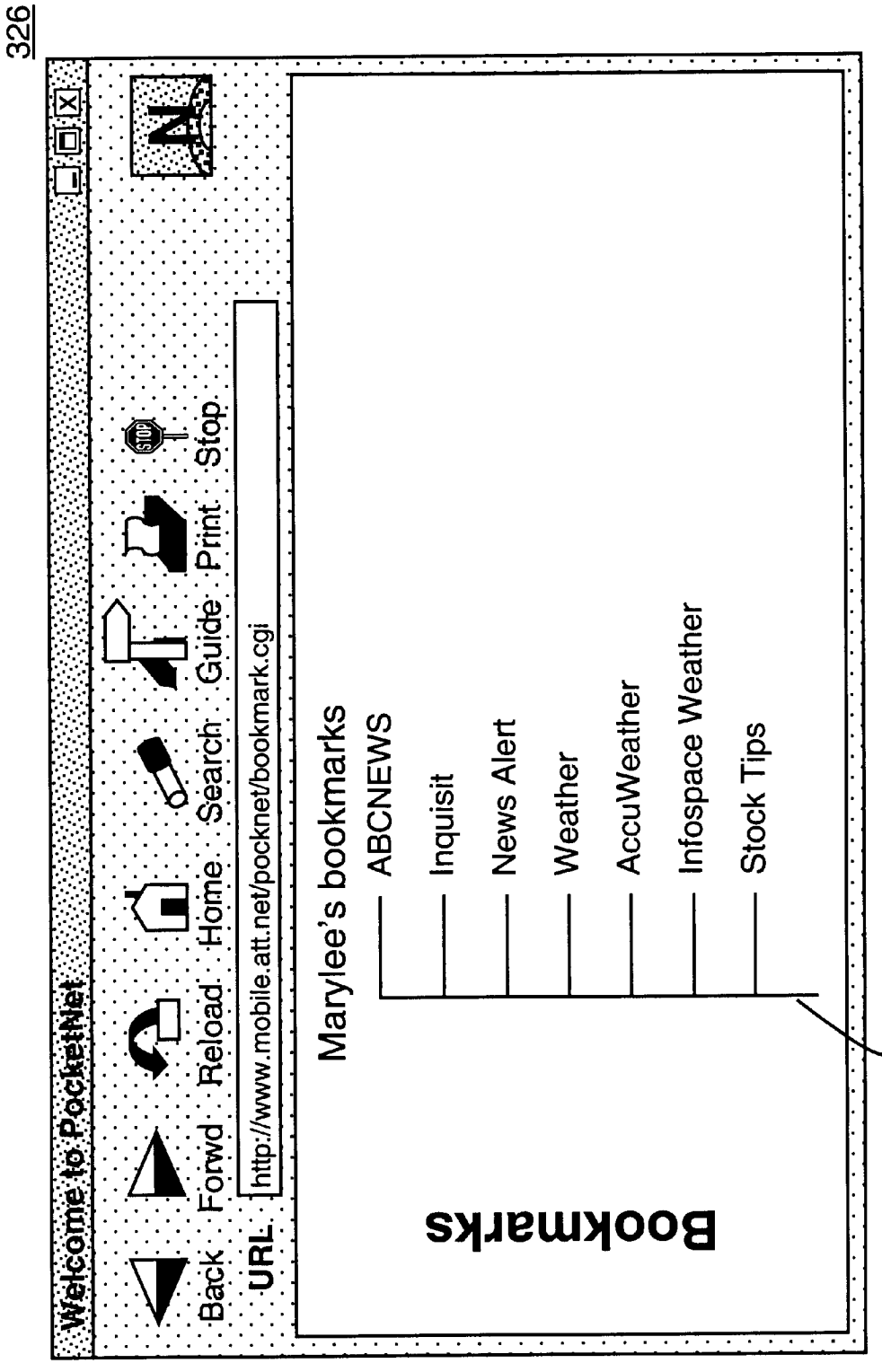

326
Welcome to PocketNet
Back
Forwd
Reload
Home
Search
Guide
Print
Stop
URL
http://www.mobile.att.net/pocknet/bookmark.cgi
Bookmarks
Marylee's bookmarks
ABCNEWS
Inquisit
News Alert
Weather
AccuWeather
Infospace Weather
Stock Tips
322

Fig. 9

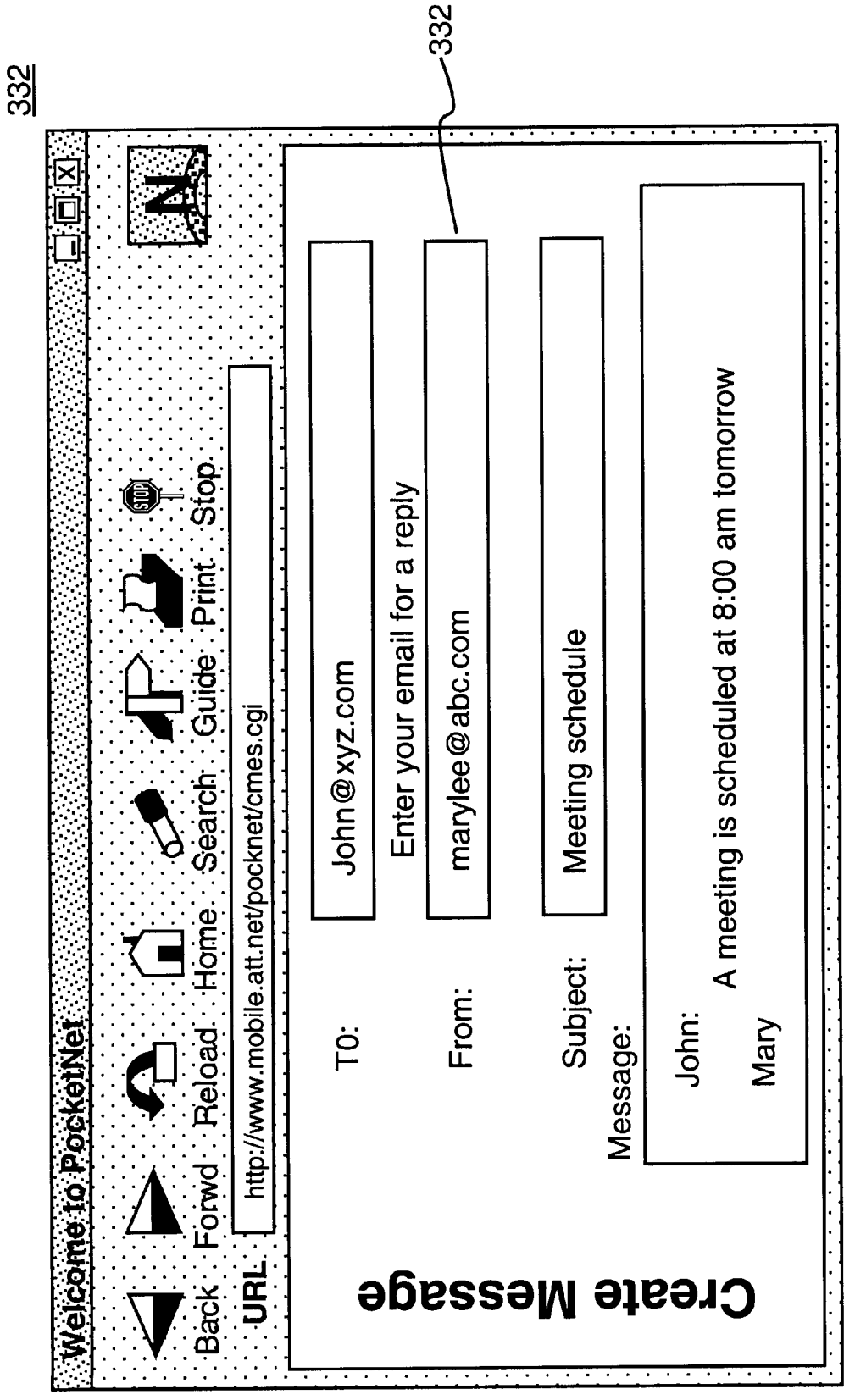

332
332
Welcome to PocketNet
Back
Forwd
Reload
Home
Search
Guide
Print
Stop
URL
http://www.mobile.att.net/pocknet/cmes.cgi
Create Message
TO:
John@xyz.com
Enter your email for a reply
From:
marylee@abc.com
Subject:
Meeting schedule
Message:
John:
A meeting is scheduled at 8:00 am tomorrow
Mary

Fig. 10

METHOD AND SYSTEM FOR SECURELY INTERACTING WITH MANAGED DATA FROM MULTIPLE DEVICES

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Application No. 08/987,346, filed Dec. 9, 1997, now U.S. Patent No 6,065,120, the content of which is hereby incorporated by reference.

REFERENCE TO APPENDIXES

Appendix A, which is a part of the present disclosure, is a microfiche appendix consisting of 2 sheets of microfiche having a total of 195 frames. The microfiche Appendix is a source code listing of one embodiment of the authentication and provisioning process in the present invention, which is described more completely below.

BACKGROUND OF THE INVENTION

1. Field of Invention

The invention relates to user authentication systems over data network systems, and more particularly relates to a method and system for self-provisioning, through a first device, a rendezvous to ensure secure access to managed information in a user account by other devices through the rendezvous in a data network, wherein the rendezvous is generally identified by a URL, the first device, coupled to the data network, runs a first browser under a first communication protocol and the other devices in the same data network run a second browser under a second communication protocol.

2. Description of the Related Art

The Internet is a rapidly growing communication network of interconnected computers around the world. Together, these millions of connected computers form a vast repository of hyperlinked information that is readily accessible by any of the connected computers from anywhere and anytime. To provide mobility and portability of the Internet, wireless computing devices were introduced and are capable of communicating, via wireless data networks, with the computers on the Internet. With the wireless data networks, people, as they travel or move about, are able to perform, through the wireless computing devices, exactly the same tasks they could do with computers on the Internet

The most common remote access paradigm is, as of today, the one in which a laptop personal computer is equipped with a wireless communication mechanism, for example, a wireless modem. This paradigm may remain useful for a considerable number of applications and users, but there has been a growing need for a mobile paradigm in which the Internet can be instantly accessed by mobile devices, such as cellular phones and personal digital assistants. The mobile devices are generally designed small in size and light in weight. With increasing data processing capabilities in the mobile devices, more and more users start carrying the devices around to materialize their unproductive time into

productive time. As more commonly seen, regular mobile phones can return calls, check voice mail or make users thereof available for teleconferences anywhere and anytime, but desired mobile phones, not just reactive to calls but also proactive, can meld voice, data, and personal information with manager-like functionality into a single handset that can effectively, through a host computer, access a myriad of public and enterprise information services in the Internet.

The evolution of the mobile phones or the mobile devices has been fueled by the demand of users for immediate access to the information they are looking for. For example, a traveler may request an exact flight schedule when he is on his way to airport, or a trader may purchase shares of stock at a certain price. The pertinent information from these ideas or transactions may include the airline and the flight number for the traveler as well as the number of shares and the price thereof being purchased by the trader. To be timely informed, a preferable way is to communicate the information requests electronically into the wireless data network. The data network, for example, connects to a flight information server or stock quote server so that the desired flight information or the current stock price can be retrieved therefrom on demand. However, it becomes troublesome or impractical to key in lengthy information queries electronically into the data network through a mobile device that typically has a keypad with a few buttons, much less functional compared to a keyboard in a personal computer system. There is therefore a great need for a method and system for efficiently communicating desired transactions into a data network through which the transaction can be performed or pertinent information can be retrieved without the need to key in such every time the transactions or the information are desired. In many cases the desired information in a user account, especially regarding personal matters, is preferred to be confidential. Thus there is further a need for a generic solution that provides a method and means for self-provisioning an account entry to a user account that has the proprietary information therein accessible only through the account entry.

SUMMARY OF THE INVENTION

The present invention has been made in consideration of the above described problems and has particular applications to systems of self-authentication by authorized users using devices that have limited computing power. Cellular phones are the typical example that has very little computing power and memory to satisfy the power long lasting and portability requirement, others include Internet-enabled electronic appliances that generally have computing powers at a minimum so as to reduce the cost thereof for market popularity. All these devices, considered as thin devices or clients herein, in data networks, provide users with portable, convenient, and instant access to information being sought in the Internet; for example, retrieving a list of stock quotes using a mobile phone or viewing a list of interested news stations on Internet-connected TVs. In both examples, the mobile phone and a remote control of the TV have very limited user interface to receive inputs from users. One of the important aspects of the present invention is to provide a generic solution for communicating desired ideas or transactions from other devices with rich user interface to such a thin client through a self-provisioned account entry.

While administrated user authentication systems over data networks have been used extensively in areas such as administered network computers and electronic commerce in the Internet, the present invention disclosing a method and system for self-provisioning, through a first device, e.g. the

cellular phone or the remote control, a rendezvous to ensure secure access to a user account by other devices through the rendezvous yields unexpected results. The administrated user authentication systems in computer networks generally require each account holder to remember his username and associated password. If the username and password were ever lost or forgotten, the corresponding account becomes abandoned or must be clarified by a system administer. The disclosed invention, however, allows a user to self-provision an account entry or a rendezvous with a set of credential information, which does not require the user to write down or remember the credential information in order to access his account. Further, the user is the only one who knows the credential information created in an authenticated and secure communication session for the rendezvous, thereby the account becomes truly proprietary. Moreover through the rendezvous, the present invention for the first time allows efficient means for communicating personalized information into a database by utilizing other computers running an HTML browser with more familiar graphic user interface while allowing a thin device running a micro browser to access the same personalized information stored in the database.

According to one preferred embodiment of the present invention, a method for provisioning, through a thin device, a rendezvous to a user account in a server to ensure secure access to the user account by a networked computing device through the rendezvous having a URL, thereby the networked computing device can update managed information in the user account that is also accessible by the thin device, the method comprises:

initiating a transaction signal by the thin device to the server; the thin device having a client identification associated with the user account in the server and running a micro browser supported by a first communication protocol, wherein the transaction signal comprises the client identification and the URL of the rendezvous;

examining a communication session between the thin device and the server, wherein the session examination between the thin device and the server comprising:

creating the communication session between the thin device and the server if the communication session is not in existence or is not valid;

conducting mutual authentication between the thin device and the server; and

generating session credential information for the session such that subsequent transactions between the thin device and the server are encrypted by the session credential information; and

establishing user credential information for the rendezvous by the thin device; and

associating the user credential information with the rendezvous to the user account in the server.

Upon updating the user credential information to the rendezvous, the networked computing device with the correct user credential information can go through the rendezvous to the user account to edit, modify or update the managed information, e.g. a URL of a Web server, in the user account with a more convenient information entering means, such as an HTML browser. The thin device can immediately access the managed information, such as the specified URL, to retrieve pertinent information therefrom without the need to key in the URL that often has a number of alphabets.

The system for secure access to a user account in a server, through a rendezvous identified generally by a URL, the rendezvous being exclusively designated to the user account, the system comprising:

a data network comprising an airnet supporting a first communication protocol and a landnet supporting a second communication protocol, the landnet coupled to the server;

a first client device, remotely located with respect to the server device and coupled to the airnet using a first communication protocol, having a client identification exclusively associated with the rendezvous and running a first browser;

a second client device coupled to the landnet using a second communication protocol and running a second browser,

means for mapping the first communication protocol to the second communication protocol and the second communication protocol to the first communication protocol; the first client communicating with the server via the communication protocol means;

means for mapping the first communication protocol to the second communication protocol and the second communication protocol to the first communication protocol;

means for creating an authenticated and secure communication session between the first client device and the server through the data network; the session creating means comprising:

means for requesting the session by the first client device to the server if the session is not in existence or is not valid;

means for conducting mutual authentication between the first client device and the server; and

means for generating session credential information for the session in creation; and

means, by the first client and through the created session, for updating the rendezvous with user credential information by a first browser such that the user account is accessible by the second client through the rendezvous with the user credential information.

Accordingly, an important object of the present invention is to provide a generic solution for self-provisioning a rendezvous to a corresponding user account created and authorized in a server;

Another object of the present invention is to provide a method and system for efficient and secure access to a user account by self-provisioning a rendezvous to the account as such any computer with a more convenient information entering means may update managed information in the account; and

Other objects, together with the forgoing are attained in the exercise of the invention in the following description and resulting in the embodiment illustrated in the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

FIG. **1** shows a schematic representation of a data network in which the present invention may be practiced;

FIGS. **2**.*a* and **2**.*b* illustrate a representation of system architecture of the present invention and a layout of a corresponding user account in a server in communication with a mobile phone and a PC;

FIG. **3** shows a typical example of a mobile device that houses one portion of the linked and complied processes disclosed in the present invention;

FIG. **4** illustrates a schematic representation of a mutual authentication process between a mobile device and a host server to ensure subsequent information transacted therebetween is secured;

FIGS. **5**.*a* and **5**.*b* demonstrate a flowchart showing the corresponding processes in each of the involved devices, respectively; and

FIGS. **6**, **7**, **8**, **9** and **10** illustrate, respectively, examples of personal g a user account being accessed through a self-provisioned rendezvous.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will become obvious to those skilled in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuitry have not been described in detail to avoid unnecessarily obscuring aspects of the present invention.

The following detailed description of the present invention is presented largely in terms of procedures, steps, logic blocks, processing, and other symbolic representations that resemble the operations of data processing devices coupled to networks. These process descriptions and representations are the means used by those experienced or skilled in the art to most effectively convey the substance of their work to others skilled in the art. The present invention is a method and system for self-provisioning a rendezvous through a thin device to ensure secure access by other devices to information in a database in a data network. The method along with the system or architecture to be described in detail below is a self-consistent sequence of steps leading to a desired result These steps or processes are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities may take the form of electrical signals capable of being stored, transferred, combined, compared, displayed and otherwise manipulated in a computer system or electronic computing systems. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, operations, messages, terms, numbers or the like. It should be borne in mind that all of these similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following description, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "verifying" or "displaying" or the like, refer to the actions and processes of a computing system that manipulates and transforms data represented as physical quantities within the computing device's registers and memories into other data similarly represented as physical quantities within the computing device or other such as storage, transmission or display devices,

Referring now to the drawings, in which like numerals refer to like parts throughout the several views. FIG. **1** shows a schematic representation of a data network **100** in which the present invention may be practiced. The data network **100** comprises an airnet **102** that is generally called wireless network and a landnet **104** that is generally a landline network, each acting as a communication medium for data transmission therethrough. The airnet **102**, in which transmission is via the air, is sometimes referred to as a carrier network because each airnet is controlled and operated by a carrier, for example AT&T and GTE, each having its own communication scheme, such as CDPD, CDMA, GSM and TDMA for the airnet **102**. The landnet **104** or the Internet, used interchangeably herein, may be the Internet, the Intranet or other private networks. Referenced by **106** is a mobile data device, but resembling a mobile phone therein, in communication with the airnet **102** via an antenna **108**. It is generally understood that the airnet **102** communicates simultaneously with a plurality of mobile computing devices of which only a mobile or cellular phone **106** is shown in the figure. Similarly, connected to the Internet **104** are a plurality of desktop PCs **110** and a plurality of servers **112**, though only one representative respectively is shown in the figure. The PC **110**, as shown in the figure, may be a personal computer SPL 300 from NEC Technologies Inc. and runs a HTML Web browser via the Internet **104** using HTTP to access information stored in the web server **112** that may be a workstation from SUN Microsystems Inc.. It is understood by those skilled in the art that the PC **110** can store accessible information therein so as to become a web server as well. Between the Internet **104** and the airnet **102** there is a link server **114** performing data communication between the Internet **104** and the airnet **102**. The link server **114**, also referred to as link proxy or gateway, may be a workstation or a personal computer and performs mapping or translation functions, for example, communication protocol mapping from one protocol to another, thereby a mobile device **106** can be in communication with any one of the servers **112** or the PCs **110**, respectively.

The communication protocol in the Internet **104** is the well known HyperText Transfer Protocol or HTTP and runs on TCP and controls the connection of a well known HyperText Markup Language Web browser, or HTML Web browser, to a Web server and the exchange of information therebetween. The communication protocol between the mobile device **106** and the link server **114** via the airnet **102** is Handheld Device Transport Protocol (HDTP), or Secure Uplink Gateway Protocol (SUGP), which preferably runs on User Datagram Protocol (UDP) and controls the connection of a HDML Web browser to a link server, where HDML stands for Handheld Device Markup Language, it is similar to that of HTML and a set of commands or statements that specify how information is displayed. The specifications of both HDTP and HDML, being considered as the wireless network standards, are provided at http://www.w3.org or http://www.uplanet.com and incorporated herein by reference. Further a reference specification entitled "Magellan SUGP Protocol", a HTTP specification with network security features is incorporated herein by reference as Appendix B. The HDTP is a session-level protocol that resembles the HTTP but without incurring the overhead thereof and is highly optimized for use in mobile devices that have significantly less computing power and memory. Further it is understood to those skilled in the an that the UDP does not require a connection to be established between a client and a server before information can be exchanged, which eliminates the need of exchanging a large number of packets during a session creation between a client and a server. Exchanging a very small number of packets during a transaction is one of the desired features for a mobile device with very limited computing power and memory to effectively interact with a landline device.

Referring now to FIGS. **2**.*a* and **2**.*b*, there is depicted a representation of the architecture **120** of the present inven-

tion. As described above, the airnet **102** communicates simultaneously with a plurality of two-way mobile communication devices, **122**, **124** and **126**, generally from a group consisting of mobile phones, two-way pagers and telephones, such as a Duette cellular phone from Samsung Telecommunication America, Inc. Due to the increasing reduction in size and weight and high portability, most of the mobile devices, considered as thin clients, have a very limited computing power, typically equivalent to less than one percent of what is provided in a typical desktop or portable computer, the memory capacity in a thin client is generally less than 250 kilobytes and the LCD display thereof is perhaps four lines high by twelve or twenty characters, the graphics capabilities thereof are very limited or nearly nonexistent and the general user interface is a keypad having far less buttons than a PC keyboard does. Therefore many transactions desired by users through such clients are preferably predetermined or pre-entered in their user accounts in a host server **128** as such the users need only to select desired transactions to perform or at most key in one or two letters corresponding to desired entries through the keypads of their cellular phones. For example, if there is a list of stock symbols of interest in a user account that is designated to a mobile phone, a user of the mobile phone will not have to key in the symbols every time he desires to look up for the price thereof currently being traded in the stock market. The list of stock symbols is previously entered to the user account. Evidently the most available and convenient means for now is to use a computing device that has powerful and full functional information entering capabilities. A PC is a typical example of such computing device, the PC can be equipped with the well-known HTML browser that provides a rich graphic user interface and an ideal environment for the user to manage his personalized information in his account.

As is well known, the Internet **104** is typically a landline network connecting computers that are provided the HTML browser. Referenced by **110** is a PC representing one of the computers that use the HTML browser running on HTTP to hyperlink to other computers/servers **132** or **134** to update/fetch information on line or simply copy files therefrom. It should be noted that "user account" and "database" have been used herein sometimes interchangeably when only one account is being addressed. It is generally understood that a database or an allocation of memory, as referenced by **130** in the FIGS. **2**.*a* and **2**.*b*, hosts a plurality of user accounts, each designated to an authorized capacity in which managed or personalized information is kept. Further it is understood that the database **130** can be an independent storage or physically a part of the host server **128**. To access the personalized information therein from any computer on the Internet **104**, one has to provide an account entry, namely a rendezvous, to a user account in the host server **128** or database **130** with a set of credential information such as a username and a password thereof. FIG. **2**.*b* illustrates a layout of a typical user account assigned with a mobile phone **106**. Each mobile phone is assigned to a device ID **140** which can be a phone number of the phone or a combination of an IP address and a port number, for example: 204.163.165.132:01905 where 204.163.165.132 is the IP address and 01905 is the port number. The device ID **140** is further associated with a subscriber number (sub #) **142** authorized by a carrier in the link server **114** as part of the procedures to activate the phone **106**. The sub # may take the form, for example, of 861234567-10900_pn.mobile.att.net by AT&T Wireless Service, it is a unique identification to the phone **106**. In other words, each of the mobile devices **122**, **124** and **126** has a unique device ID that corresponds to a user account in a server, respectively. It may be appreciated by those skilled in the art that the link server **114** does not have to be a separate sever to perform the communication protocol mapping, it can be just a part of the host server **128** and the protocol mapping is a part of functions the host server **128** provides.

A corresponding account **144** in the database **130** is indexed by an account structure **143** comprising the sub #**142**, user information **146**, a username **148** and a password **150**. The sub #**142** is received from the link server **114** as an index to the account structure **143**, the user information **146** comprises the account configuration and other account related information. The username **148** and the password **150**, namely the user credential information, control the authentication to enter the account **144** in the database **130**. From the data network perspective, any computer can logon through HTTP to the rendezvous **152** identified by an address identifier, often a universal resource locator (URL) taking the form of www.xyz.com. In other words, each account in a database is exclusively associated with a rendezvous identified by a unique URL. As shown in the figure, the PC **110** establishes a communication session with the rendezvous **152** based on a given URL of the rendezvous **152**. However, to access the associated account **144** in the database **130**, the PC **110** must provide a set of correct username and password to the rendezvous **152** that performs a verification thereof with the account structure **143**. If the supplied username and password match those in the account structure **143**, the access requested by the PC **110** is allowed. Otherwise, the entry to the account **144** is denied.

The PC **110** can update information stored in the account **144** when the supplied username and password are verified. Using the powerful and familiar HTML browser in the PC **110**, a user can key in frequently request information, such as a list of stock symbols and a list of URLs of Web servers that provide services to the phone **106**. An example will be provided later. All the information entered through the PC **110** becomes immediately available to the phone **106**.

A process named webpwd.cpp in the code listing in the appended Microfiche Appendix A illustrates a provisioning process between the phone **106** and the link server **114** in one embodiment of the present invention. Upon the request of the phone **106**, the process, specifically in a subprocess called setNameAndPasswordState(), allows the phone **106** to supply a username and a password and then send the newly supplied credential information to a second subprocess called submitstate() that checks if the entered username and password are acceptable, namely the username and password should have a certain length and contain no spaces or unrecognized characters with respect to a general rule of being a username and password. If the username and password are not acceptable, the subprocess submitState() returns to the phone **106** with a corresponding message being either "You must enter a name" or "You must enter a password". Otherwise, the newly entered username and password are sent to another subprocess called SetUserAuth() in a process called HTTPDBMSUerDB. The subprocess SetUserAuth() updates the username and password in the account structure **143**, which immediately requires all subsequent logins to the account entry **152** with the newly supplied username and password. A subprocess Authenticate() examines a set of username and password supplied by the PC **106**, it compares the username and password from the PC **110** to the ones in the account structure **143**. If the comparison is successful, the subprocess Authenticate() returns a AuthPass flag that allows the

PC **110** to access the account in the database. Otherwise, it returns a flag that denies the admission of the PC **100** to the account.

It should be noted that the communication between the phone **106** and the link server **114** is through the airnet **102** in FIG. **1**. Message carrying proprietary information travelling in the air is not secure. To transact credential information over the open space to provision the rendezvous, user must have an efficient, reliable and secured manner to conduct private communications with the link server. According to one embodiment of the present invention, an authenticated and secure session between the cellular phone **106** and the link server **114** must be in place before the cellular phone, provisions the rendezvous through which the user accesses his/her account from other computers. It is necessary to refer to an architecture of a mobile phone before proceeding with the detailed description of creating the authenticated and secure communication between a user's phone (client) and a server. FIG. **3** is a block diagram of a typical GSM digital cellular phone **160**. Each of the hardware components in the cellular phone **160** is known to those skilled in the art and so the hardware components are not to be described in detail herein. Although the user interface of the phone **160** is not shown in detail in the figure, the mobile device **118**, resembling a cellular phone, in FIG. **1** may be referenced thereto, in which referenced by **116** is a LCD screen and **118** is a key button pad, respectively. The screen **116** prompts user what to proceed with the keypad **118**, with a sequence of key entries and through the phone **160**, a user can interactively communicate with a server through the airnet, link server and the Internet. According to one embodiment of the present invention, complied and linked processes of the present invention are stored in ROM **162** as a client module **164** and support module **166**. Upon activation of a predetermined key sequence utilizing the keypad **118**, a physical layer processor or microcontroller **118**, initiates a session communication to the server using the module **164** in the ROM **162**.

To establish a secured communication between a cellular phone (a client) and a server, an authentication process must be conducted first to ensure that only interested parties are actually in the communication therebetween. According to one embodiment of the present invention, the code listing thereof being provided in the appended Microfiche Appendix, the process is complete through two rounds of independent authentication, one being the client authenticated by the server, referred to as client authentication, and the other being the server authenticated by the client, referred to as server authentication. Further each authentication is completed in two separate steps for high grade of security, which will be described in detail below. The success of the mutual authentication processes provisions an evidence that the two communicating parties possess a valid shared secret encrypt key through a mutual decryption and a challenge/response mechanism. The mutual decryption mechanism comprises the steps of mutually recovering encrypted messages from two involved communicating parties. The challenge/response mechanism, referred to as nonce verification, verifies a predetermined relationship between a sent nonce and a received derivative thereof.

In one preferred embodiment of the present invention, the authentication process is conducted with three message exchanges; a Session Request (SR), a Session rePly (SP), and a Session Completion (SC). FIG. **4** illustrates a schematic representation of the authentication process. The client **170**, representing a mobile device or the cellular phone **106** of FIG. **1**, to conduct a transaction with the server **172**, representing a link server **114** of FIG. **2**, initiates a SR **174** to be sent to the server **172** by first creating a client proto-session. A client proto-session is a session data structure that gets initialized when a session creation starts. The initialized SR **174** comprises the following essential information:

- sessionID—an identifier identifying all requests from the client to the server; in the case of requesting a session creation, sessionID is always assigned to 0;
- cipher—a two-byte number representing the choice of the encryption the client is currently using as there are a number of encryption schemes available in a communication protocol;
- deviceID—a variable up to 255-byte, representing the device identifier or the client identifier, comprising a phone number of the device or an IP address and a port number, e.g. 204.163.165.132:01905; and
- C-nonce—a client nonce represented with a non-repeatable number, usually 2 bytes, used for the client to conduct a following server authentication.
- C-nonceModified—a modified version of the client nonce, used for the server to conduct a nonce verification in the following client authentication.

Further the cipher in the SR **174** includes an identifier to an encryption algorithm and associated parameters thereof. To be more specific, the first byte in the cipher represents an identifier to a combination of the encryption algorithm, the key size (e.g. 128-bit for US or 40-bit for foreign countries) and content of a security attachment thereto and the second byte in the cipher indicates the additional parameters related to the first byte. For example, value 1 in the first byte indicates that the encryption algorithm is block cipher RC**5**, the key size thereof is 128 bit, a two byte check-sum therein is used as the MAC (Message Authentication Code), no IV (Initialization Vector for block ciphers) therefor is transmitted over the network, and padding bytes are added if necessary. The block cipher algorithm RC**5** is part of the RSA's BSAFE product. It can be further appreciated that the identifier in the cipher may be assigned to a unique value to identify a non-secure session if so desired. The C-nonce is a non-repeatable number initially and randomly generated in the client and the modified version thereof, C-nonceModified, is generated from the C-nonce through an operational relationship; for example the Exclusive-OR relationship or expressed as follows:

C-nonceModified=2-byte-number⊕C-nonce.

It can be appreciated by those who are skilled in the art that there are many ways to get the C-nonceModified from a C-nonce, the Exclusive-OR is one of the operational relationships used in one embodiment of the present invention. Both C-nonce and C-nonceModified are encrypted using the shared secret encrypt key between the client **170** and the server **172**. The purpose of the C-nonceModified is to provide the server that receives the SR with means for ensuring that C-nonce is correctly decrypted and validated by examining the C-nonce and its relationship with the C-nonceModified. Both should not be altered after a successful decryption of the C-nonce and the C-nonceModified. In other words, a SR message or signal may be expressed as follows:

SR={session ID, cipher, device ID, Encry[nonce, nonceModified]};

where Encry[ ] means that the parameters or contents in the bracket are encrypted accordingly. When the SR is sent by

the client to the server to request a session creation, both C-nonce, C-nonceModified are encrypted according to the cipher the client is using at the time the SR is sent out.

Upon receiving the SR from the client **170**, the server **172** creates a server proto session for the client **170** with a session identifier, referred to as session ID, to identify the session context for the session just created in the server **172**. A server proto-session is a session entry marked as a proto status in a session table, which indicates that the session is not authenticated and is not able to conduct any transactions with the client. It is understood to those skilled in the art that the proto-session can be kept in the RAM of the server. If a proto-session already exists for that client, it is re-used. The information in the received SR is saved in the server proto-session. If the server **172** is satisfied with the fact that the client is known, namely Encry[C-nonce, C-nonceModified] in the received SR are successfully decrypted with the shared secret encrypt key, the step one in the client authentication is successful and a corresponding session key is generated and stored with the server proto session entry. It may be noted herein that many encryption schemes used in this invention, such as the scheme utilizing RC**5**, have a procedure that adds and validates the Message Authentication Code such as the check-sum, to assure that the encrypted message is correctly decrypted, the procedure, every time the decryption takes place, is used herein to examine the transaction integrity, namely to assure the received messages or signals are unaltered in the cause of data transmission. If the step one client authentication is not successful, namely Encry[C-nonce, C-nonceModified] in the received SR are not fully decrypted or supported, the proto session is aborted and removed from the proto session table, resulting in a failed session creation. What the support means herein is the cipher proposed or used by the client is also used by the server, for example the client uses the RC**5** encryption to encrypt Encry[C-nonce, C-nonceModified], to decrypt Encry[C-nonce, C-nonceModified], the server must be equipped with the same RC**5** encryption capability therein. If Encry[C-nonce, C-nonceModified] can not be successfully decrypted due to other reasons such as transmission errors, the client must reinitiate a new session request to the server in order to establish a secure communication with the server. To challenge the step two server authentication subsequently at the client side, a derivative of the client nonce or C-nonce, is generated therefor. In one embodiment of the present invention, the derivative is created by adding a constant to the client nonce, for example derivative=C-nonce+1. The purpose of the derivative is to provide the client with means for reassuring that the C-nonce is correctly decrypted by the server and the server is the correct server with which the client should be in communication.

Right after the successful step one client authentication, the server **172** responds to the client with a Session rePly (SP) **176** to begin a second round authentication; server authentication. The SP **176** comprises the following information:

- C-SID—a one byte number indicates the sessionID originally assigned in the client, to be more specific C-SID=0 indicates a clear text client session, C-SID=1 indicates a shared secret key encrypted session, and C-SID=2 indicates a session key encrypted session. In the context of the current description, C-SID=1.
- sessionID—a four-byte number representing an identification and parameters, such as a session encrypt key, of the session created by the server for the client;
- key—a session key to be used with a mutually acceptable encryption, and to be used for encryption and decryption in all transactions in the session;
- derivative—a number derived from the C-nonce for the client to perform the subsequent server authentication;
- S-nonce—a non-repeatable number, used for the server to conduct a following step-two client authentication; it should be noted that S-nonce is generated by the server and generally different from the C-nonce by the client; and
- cipher—a two-byte number representing the choice of the encryption the server proposes after the client proposed cipher is received, It may or may not be the same as the one used in the client, to be more specific, the cipher is the same as the one proposed by the client when the server supports the client proposed cipher, otherwise the cipher is the one currently used in the server.

In other words, the SP can be expressed as follows:

SP={C-SID, Encry[sessionID, key, S-nonce, derivative, cipher]};

When the client **170** receives the SP **176** from the server **172**, it performs the step one server authentication, which is considered successful if Encry[sessionID, key, S-nonce, derivative, cipher] in the received SP **176** is decrypted successfully with the shared encrypt key. If the step one server authentication fails, the client **170** discards the SP **176** and a new session creation may be started over again. Upon the success of the step one server authentication, the client **170** proceeds with the step two server authentication; namely the predetermined relationship between the C-nonce and the derivative thereof should be held for a successful step-two server authentication:

C-nonce=derivative−1

If the C-nonce derived from the SP **176** is the same as the C-nonce originally generated by the client, the step two server authentication is successful, hence the server **172** is considered authenticated, trusted from the viewpoint of the client, and the SP **176** is accepted as a valid message, which means that the client **170** then uses the session key and other information in the SP **176** for the session being created. Only with both successful steps of the server authentication, the client **170** marks the session as committed, which means that transactions can be conducted subsequently in the session, again only from the viewpoint of the client **170**. If the predetermined relationship between the client nonce and the derivative thereof does not hold, the step two server authentication fails and the received SP **176** is discarded. The client **170** may abort the session creation process if no further SP's are received and pass both steps of the server authentication during the time period allowed for a session creation. To provide the server with means for reassuring the client authentication by itself through the client, a derivative of the S-nonce, similar to the derivative of the C-nonce, is generated.

The client **170** then sends the server **172** a SC **178** to complete the session creation process. The SC **178** comprises the following information:

SC={Encry[derivative]};

where the derivative is the client's response to the server nonce challenge, namely the result of the verification, the derivative is used by the server **172** for step two client authentication. Further it is noted that the SC **178** is an encrypted message, meaning that the client encrypts the information in the SC **178** according to either its own cipher or the server proposed cipher. Generally the client **170** encrypts the information in the SC **178** according to the

server proposed cipher if it accepts the server proposed cipher, otherwise, it encrypts the SC according to its own cipher.

Upon receipt of Session Complete or SC **178**, the server **172** tests if the client **170** uses its own proposed cipher or the server proposed cipher by decrypting the SC twice using the two ciphers if necessary. If the server **172** decrypts the encrypted message in the SC **178** and verifies the relationship thereof with the S-nonce, the step two client authentication is succeeded. Subsequently the server **172** promotes the server proto session to the active session and the session creation process is completed, thereby an authenticated and secure communication session is established between the client and the server. Any transactions in the established communications session are now encrypted by the session key created in the server according to a cipher mutually agreed by both the client and the server, thereby the transactions between the client and the server are truly proprietary. A code listing of one embodiment of the mutual authentication is listed in the Appendix A.

Referring now to FIGS. **5**.*a* and **5**.*b*, each illustrates a flowchart showing the processes of the present invention in each involved device, respectively, in conjunction with FIGS. **6**, **7**, **8**, **9** and **10** demonstrating examples of personalizing a user account being accessed through a self-provisioned rendezvous. A client **200**, which can be a cellular phone, in FIG. **5**.*a* is one of the mobile devices communicating with a server **250** in FIG. **5**.*b* through a data network that is not shown in these figures but illustrated in FIG. **1** or FIGS. **2**.*a* and **2**.*b*. It should be noted that the server **250** functions as a link server and a host server. The functional flowcharts on the client and server sides are conjointly described in the following with respect to a cellular phone. Nevertheless it will be appreciated by those skilled in the art that a server, without reciting specifically a link server or a host server, as referenced by **250** can perform similar functions, this becomes evident when the client is a landline device having direct communication to the Internet.

As part of the procedures to activate a cellular phone, a user account, or sometimes called device account, is created in the server **250**, the account is exclusively associated with the phone or client **200**. In other words, each mobile device in the data network has its own account identified by a corresponding device ID and subsequently a sub # in the server **250**. The account for the client **200** is therefore created and configured at **252** according to services subscribed by the client **200**. Meanwhile a corresponding account structure, similar to **143** in FIG. **2***b*, is initiated at **254**. With an established account in the server **250**, the client **200** becomes one of the clients capable of communicating with any computers in a data network.

When a user desires to update his personalized information in his account, he needs to first self-provision the rendezvous associated with his account using the client **200**. The phone therefore requests a communication session to the server **250** at **202** for subsequent transactions to take place in an authenticated and secure communication session. From the session creation described above, it can be appreciated that the session creation requested by the client **200** includes a piece of device information assigned to the client **200**. If, at **204** and **206**, the device information sent to the host is not recognized by the contacting host **250**, no communication session can be possibly established therefor. Meanwhile the host **250** receives the session request from the client **200**, as part of the session creation process, the device information is examined at **260** and the session creation process proceeds when the device information is verified at **262**, that means that the device **200** has an authorized account therein. At **208** and **264** respectively, a mutual authentication process between the client **200** and server **250** takes place. As described above, the mutual authentication process comprises a client authentication and a server authentication, each further comprising two respective steps to ensure that the communicating party is authenticated. Resulting from the mutual authentication process or once the session is created and authenticated at **210** and **266** of the client **200** and the server **250**, respectively, a set of session credential information is generated. The session credential information comprises a session ID, a session key and a session cipher. The session ID is used to distinguish the session from other sessions that the host is creating or has already established with other mobile devices or clients, and the session key and the session cipher are to encrypt transactions between the client **200** and the server **250**. At **212**, the client **200** is acknowledged that there is a rendezvous associated with the account designated to the phone **250**. If the user desires to update his personalized information in the account created and authorized in the server **250**, he may proceed at **214** with the rendezvous that is generally identified by a URL provided by the host **250** and is subsequently prompted for a set of user credential information, such as a username and a password. At **216**, the user credential information is entered. The credential information is then sent to the host **250** at **218**, which includes a process of ensuring the newly supplied username and password satisfy a general rule of being a username and a password. The username/password ensuring process has been discussed above and the code listing thereof is in Appendix A. Meanwhile the host **250** has acknowledged that the client **200** is about to receive a set of new user credential information and expects it therefrom at **268**. As soon as the new user credential information has arrived, the server **250** updates the user credential information associated with the rendezvous at **270**. In other words, to pass through the rendezvous to the user account now by other devices, the new credential information must be provided.

With the newly updated user credential information, the user can now log onto the rendezvous from any computer in the data network. A PC, which is not shown, connected to the data network, is equipped with a familiar HTML-based browser, preferably from Netscape Communication Corporation or Microsoft Corporation. As an example, it is assumed that a user has just provisioned a rendezvous with a username being "marylee" and the corresponding password being "123456". The user now goes to a networked PC that runs a Navigator browser from Netscape Communication Corporation and logs onto the rendezvous based on the URL of the rendezvous. FIG. **6** shows an interactive web page **300** received from the server **250** after the PC made the connection to the rendezvous. It is understood to those skilled in the art that the page and subsequent pages can be constructed with HTML along with CGI script/Java applets, where the process, CGI stands for Common Gateway interface, to receive information entered from a user. To update his personalized information in his account, the user must provide the newly created username and password required at **302** and **304**. It should be noted that the password entered is generally not echoed at **304** and instead indicated with a asterisk corresponding to a letter entered. When the login icon **306** is activated, the entered username and password are retrieved and sent, through the network, to the server **250** in which the entered username and password are verified; namely the entered username and password match those entered and authorized by the user through the client

200. The user is then prompted with a second web page 310 shown in FIG. 7 in which the username is displayed as referenced by 312. To categorize personalized information in the account, the web page 310 comprises entries to other specific service pages, such as Personal Organizer 314, Bookmarks 316 and Create a Message 318. All these pages are accessible by the user to personalize his desired information therein. FIG. 8, for example, is a page 326 of the Personal Organizer 314 showing a personalized address book 320 that allows the user to edit his frequently contacted people's phone numbers and other information. FIG. 9 is a page of the Bookmarks 316 that allows the user to establish a list of web sites he may frequently visit through his cellular client 200, for example, StockTIPS referenced by 322 allows the user to keep a list of stock symbols there. With the personalized bookmarks, the user, when on the go, can quickly enter into the web pages having his list of the stock symbol to look up for the prices thereof currently being traded in the stock market without keying in any symbols at all. As a convenient feature, the page 330 in FIG. 10 allows the user to create an email message and be replied to a different address at 332 decided by the user, which eliminates the inconvenience of typing a lengthy message through a phone keypad and reading a replied message at the small screen in the client 200.

The contents in the exemplary pages respectively shown in FIGS. 6, 7, 8, 9 and 10 composed by HTML are accessible by an HDML browser through a server providing communication protocol mapping and markup language translation functions. Similarly information or messages entered on the client 200 composed by HDML are equally accessible by any computer equipped with an HTML browser through the same server in the data network. The duality of the information updating through two different mark-up languages provides a useful means for efficiently managing a personal account and solves substantially the problems of inconvenient data entry through a less functional keypad.

The present invention has been described in sufficient detail with a certain degree of particularity. It is understood to those skilled in the art that the present disclosure of embodiments has been made by way of example only and that numerous changes in the arrangement and combination of parts as well as steps may be resorted without departing from the spirit and scope of the invention as claimed. For example, any mobile devices equipped with a micro browser, e.g. HDML browser, may be connected, using an adapter, to the Internet directly without going through the airnet, the emerging Internet-enabled electronic appliances are also Internet-connected, all have limited computing powers and keypads but are capable of communicating with a server in a data network. The mutual authentication between such devices and the server thus becomes less complicated. The mutual authentication needs a process of having the client, such as a controller of the electronic appliance, authenticated by the server and having the server authenticated by the client. The process can be carried out in existing encryption mechanisms in HTTPS (an extended version of HTTP with built-in security), in which case, the link server could be replaced by a built-in capability in the device, or the HTTPS or the transceiver or somewhere in the connection to the Internet. The principles of the present invention may still be practiced in such configuration. Accordingly, the scope of the present invention is defined by the appended claims rather than the foregoing description of one embodiment.

What is claimed is:

1. A method for accessing managed data contained in a data network system, the method comprising:

executing a first set of program instructions in a wireless telephone of a subscriber, the wireless telephone having a display screen and being in communication, over a wireless data network, with a server hosting the managed data, the managed data being uniquely associated with the wireless telephone of the subscriber and being accessible by a computing device executing a second set of program instructions and coupled to the server through a wired data network, the computing device being able to alter the managed data at the server via the wired data network, wherein the wireless data network and the data network utilize a first communication protocol and a second communication protocol, respectively;

sending a request to the server to retrieve the managed data after activation of a predefined key of the wireless telephone;

receiving, at the wireless telephone, the managed data from the server via the wireless data network, the managed data being presented in a first markup language interpretable by the first set of program instructions when presented to the wireless telephone and being presented in a second markup language interpretable by the second set of program instructions when presented to the computing device; and

displaying the managed data on the display screen of the wireless telephone.

2. The method as recited in claim 1, wherein the managed data comprises an address book and bookmarks entered from the computing device executing the second set of program instructions.

3. The method as recited in claim 1, wherein the first markup language is the same as the second markup language.

4. The method as recited in claim 1, wherein the first markup language is Handheld Device Markup Language (HDML) and the second markup language is Hypertext Markup Language (HTML).

5. The method as recited in claim 1; wherein the request comprises an address identifier identifying the server.

6. The method as recited in claim 5, wherein the address identifier is a universal resource locator (URL).

7. The method as recited in claim 1,

wherein the managed data comprises a plurality of selectable hyperlinks, each of the hyperlinks providing access to a resource in the data network; and

wherein the displaying comprises displaying at least one of the selectable hyperlinks on the display screen of the wireless telephone using the first set of program instructions.

8. The method as recited in claim 7, wherein the first set of program instructions is included in a first browser being operated in the wireless telephone and the second set of program instructions is included in a second browser being operated in the computing device.

9. The method as recited in claim 8, the method further comprising:

sending a new request from the wireless telephone to the server using the first set of program instructions to fetch information identified by one of the hyperlinks when the one of the hyperlinks being displayed is selected.

10. A method for accessing data contained in a data network system, the method comprising:

hosting, at a server, data associated with an account for a wireless telephone having a display screen, the data comprising a plurality of information categories and

being accessible by a computing device remotely located and coupled to a data network selected from a group consisting of the Internet, a private network and a network of private networks;

receiving a request from the wireless telephone through a wireless data network to access the data, the request comprising a selection of one of the information categories;

retrieving information pertaining to the selected category if such information is co-located with the account and after the request is authenticated with respect to the account; and

forwarding the information to the wireless telephone in a first format displayable on the display screen of the wireless telephone.

11. The method as recited in claim **10**, wherein the request is an update to the one of the information categories and causes the data to be updated with the update.

12. The method as recited in claim **10**, further comprising:

prompting the computing device for credential information when the computing device accesses the data;

providing access to the data in a second format after the credential information is verified; and

updating the data upon receiving updated information from the computing device.

13. The method as recited in claim **12**, wherein the first format is in a first markup language and the second format is in a second markup language.

14. The method as recited in claim **10**, wherein the data comprises a plurality of hyperlinks, and the selected category is one of the hyperlinks; and wherein the retrieving further comprises:

contacting a resource identified by the one of the hyperlinks over the data network;

fetching the information in a second format from the resource; and

converting the respective information in the second format to the first format.

15. The method as recited in claim **14**, wherein the first format is a first markup language and the second format is a second markup language.

16. A method for interacting with managed data from a wireless computing device or a wired computing device, the managed data being stored on a server coupled to a data network, said method comprising:

permitting access to the managed data in a secure manner via the wired computing device;

receiving user input from the wired computing device;

altering the managed data being stored at the server based on the user input from the wired computing device; and

thereafter permitting access to the managed data in a secure manner via the wireless computing device and then forwarding the managed data to the wireless computing device for use therein.

17. A method as recited in claim **16**, wherein the managed data is used on the wireless computing device to generate screen displays on a display screen of the wireless computing device.

18. A method as recited in claim **16**, wherein the managed data comprises at least one of address book data and bookmark data.

19. A method as recited in claim **16**, wherein the managed data comprises user account data.

20. A method as recited in claim **16**, wherein the wireless computing device is associated with a user; and

wherein the managed data is personalized information of the user.

21. A method as recited in claim **16**, wherein said method further comprises:

altering the managed data being stored at the server based on the user input from the wireless computing device.

22. A method as recited in claim **16**, wherein the wired computing device is a personal computer.

23. A method as recited in claim **16**, wherein said permitting access to the managed data in a secure manner via the wireless computing device comprises:

authenticating the wireless computing device to the server; and

authenticating the server to the wireless computing device.

24. A method as recited in claim **16**, wherein the wired computing device is a personal computer having a standard size keyboard, and the wireless computing device is a small, handheld device having a telephone-type keypad.

25. A method as recited in claim **24**, wherein the managed data represents frequently requested data, thereby improving ease of use of the wireless computing device by allowing entry of the frequently requested data through use of the standard size keyboard, yet being for use by the wireless computing device.

26. A method as recited in claim **16**, wherein the wired computing device is a personal computer having a substantially more powerful user input mechanism than the wireless computing device which is a small, handheld device,

wherein the managed data represents frequently requested data, and

wherein said method improves ease of use by allowing entry of the frequently requested data through use of the more powerful input mechanism of the wired computing device, yet the frequently requested data so entered being for use by the wireless computing device.

27. A method as recited in claim **26**, wherein a user input mechanism for the wireless computing device has ambiguous keys that require several key strokes to input a particular key, whereas the more powerful input mechanism has non-ambiguous keys require only a single keystroke to input a particular key.

28. A method as recited in claim **16**, wherein the wireless computing device is a cellular telephone.

29. A method as recited in claim **28**, wherein the wired computing device is a personal computer.

30. A method as recited in claim **16**, wherein said permitting access to the managed data in a secure manner via the wired computing device uses a self-provisioning rendezvous.

31. A method as recited in claim **30**, wherein the self-provisioning rendezvous is accessed by an address identifier.

32. A method as recited in claim **31**, wherein the address identifier is a universal resource locator (URL).

33. A computer readable medium containing program code for accessing data contained in a data network system, the computer readable medium comprising:

first program code for displaying the data on a display screen of a wireless device, the data comprising a plurality of selectable information categories and hosted in a server with at least one of the information categories hyperlinking to a resource on a data network, the data also being accessible though a computing device remotely located and coupled to the data network;

second program code for receiving a selection of one of the information categories when the one of the information categories is selected by a user;

third program code, executable in response to the selection, for sending a request for information identified by the selection to the server;

fourth program code for receiving the information from the server in a first format; and

fifth program code for displaying the respective information on the display screen.

34. The computer readable medium as recited in claim **33**, wherein the selection is made through a keypad with reference to the information categories being displayed on the display screen.

35. The computer readable medium as recited in claim **33**, wherein the first format is a markup language.

36. The computer readable medium as recited in claim **33**, wherein the computer readable medium further comprises:

sixth computer program code for receiving updated information entered from a telephone keypad; and

seventh computer program code for sending the updated information to the server, the data being updated with the updated information.

37. A computer readable medium containing program code for accessing data in a data network system, the program code comprising:

a first program code for receiving a request, through a wireless data network, sent from a first browser being executed in a wireless telephone to access the data hosted in a database; the data associated with the wireless telephone and being accessible via a second browser executing on a computing device coupled to a data network that is part of the data network system;

a second program code for authenticating the request with respect to an account associated to the wireless telephone; and

a third program code for forwarding the data in a format supported by the first browser, through the wireless data network, to the wireless telephone.

38. The computer readable medium as recited in claim **37**, wherein the request comprises an identification identifying the wireless telephone; and the program code further comprises a fourth program code for verifying if the wireless telephone is authorized by comparing the identification with the account.

39. The computer readable medium as recited in claim **37**, wherein the request comprises credential information; and the program code further comprises a fourth program code for verifying if the wireless telephone is authenticated by comparing the credential information with the account.

40. The computer readable medium as recited in claim **37**, wherein the first browser is executed by a processor of the wireless telephone, and further wherein the processor controls a telephony function of the wireless telephone.

41. The computer readable medium as recited in claim **37**, wherein the data includes a plurality of hyperlinks, each providing a link to a resource in the data network.

42. The computer readable medium as recited in claim **41**, wherein if the request indicates one of the hyperlinks, the program code further comprises:

a fourth program code for retrieving information identified by the one of the hyperlinks from the data network.

43. The computer readable medium as recited in claim **37**, wherein if the credential information is different from an existing credential information after the wireless telephone is authenticated, the program code further comprises a fourth program code for updating the account with the new credential information.

44. The computer readable medium as recited in claim **43**, wherein the new credential information must be provided when the second browser executing on the computing device attempts to access the data.

45. The computer readable medium as recited in claim **44**, wherein the format is in a first markup language supported by the first browser and the data is in a second markup language supported by the second browser.

46. The computer readable medium as recited in claim **45**, wherein the second markup language provides a graphic user interface so that said data can be updated from the computing device.

47. A wireless telephone for accessing data in a data network system the wireless telephone comprising:

a display screen;

a memory containing a set of program code for a first browser;

a processor, coupled to the display screen and the memory, executing the set of program code to enable the first browser to perform operations of:

sending a request to retrieve the data from a wireless data network, the data being hosted in a server coupled between the wireless data network utilizing a first communication protocol and a data network utilizing a second communication protocol;

receiving the data presented in a first markup language;

displaying the data on the display screen; and

wherein the data is accessible by a computing device operating a second browser and coupled to the data network, and wherein the data presented to the computing device is in a second markup language.

48. The wireless telephone as recited in claim **47**, wherein the request comprises an identifier identifying the wireless telephone so that the wireless telephone can be authenticated by the server when the request is received.

49. The wireless telephone as recited in claim **47**, wherein the first markup language and the second markup language are the same.

50. The wireless telephone as recited in claim **47**, wherein the processor controls a telephony operation of the wireless telephone.

* * * * *

EXHIBIT B

U.S. PATENT NO. 6,289,212

US006289212B1

(12) **United States Patent**
Stein et al.

(10) **Patent No.: US 6,289,212 B1**
(45) **Date of Patent: Sep. 11, 2001**

(54) **METHOD AND APPARATUS FOR PROVIDING ELECTRONIC MAIL SERVICES DURING NETWORK UNAVAILABILITY**

(75) Inventors: **Lawrence M. Stein**, San Jose; **Peter F. King**, Half Moon Bay; **Bruce K. Martin, Jr.**, Palo Alto; **Bruce V. Schwartz**, San Mateo; **Paul A. Smethers**, Cupertino, all of CA (US)

(73) Assignee: **Openwave Systems Inc.**, Redwood City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/172,105**

(22) Filed: **Oct. 13, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/100,663, filed on Sep. 16, 1998.

(51) **Int. Cl.**[7] **H04M 11/10**

(52) **U.S. Cl.** **455/412**; 455/557; 455/575

(58) **Field of Search** 455/412, 556, 455/557, 403, 555, 575, 567, 418, 419

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,438,611	*	8/1995	Campana, Jr. et al.	379/58
5,461,667	*	10/1995	Remilard	379/96
5,487,100	*	1/1996	Kane	379/57

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 845 894 A2		6/1998	(EP)	H04M/3/50
0 924 921 A1		6/1999	(EP)	H04M/11/08
EP0994608A2	*	4/2000	(GB)	H04L/12/58
WO98/10580		3/1998	(WO)	H04M/11/08

OTHER PUBLICATIONS

HDML 2.0 Language Reference, Version 2.0, Unwired Planet, Inc. Software Developer Kit, Jul. 1997.
"HDTP Draft Specification", Version 1.1, Unwired Planet, Inc. 1997.
"Wireless Application Protocol Wireless Session Protocol Specification" (WAP WSP), Version 30, Apr. 30, 1998.
"Wireless Application Protocol Wireless Markup Language Specification" (WAP WML), Version Apr. 30, 1998.
Document Object Model (DOM), W3C Specification, Jul. 20, 1998 http://www.w3.org/TR/1998/WD-DOM–19980720/.
Lambert, "PCMAIL: A Distributed Mail System for Personal Computers," RFC 1056, Jun. 1988.
Kaashoek et al., "Dynamic Documents: Mobile Wireless Access to the WWW," Proceedings of Workshop on Mobile Computing Systems and Applications, IEEE Computer Society Press, Dec. 1994, pp. 179–184.
Hild, "Mobilizing Applications," IEEE Personal Communications, vol. 4, No. 5, Oct. 1997, pp. 26–34.

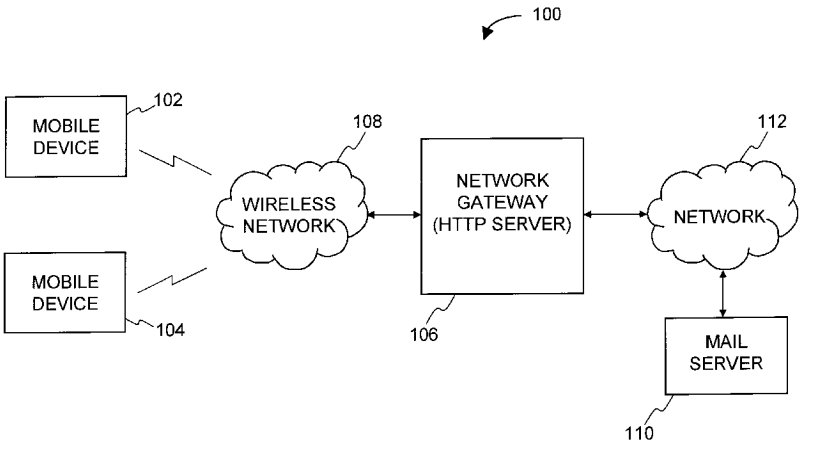
* cited by examiner

Primary Examiner—Daniel Hunter
Assistant Examiner—C. Chow
(74) *Attorney, Agent, or Firm*—Beyer Weaver & Thomas, LLP

(57) **ABSTRACT**

Improved techniques for providing electronic mail services across a network are disclosed. A mail server and its clients communicate through a network. Although the mail server centrally manages the electronic mail services, the clients are able to themselves locally perform certain electronic mail services when the network is unavailable. Accordingly, clients seeking to perform electronic mail services no longer endure significant delays when the network is unavailable. The network can be unavailable for a variety of reasons, including: congestion, out of range, network failure, etc. The network can be wired or wireless. The invention is particularly well suited for networks having sporadic connectivity, high latencies or excessive traffic.

38 Claims, 15 Drawing Sheets



U.S. PATENT DOCUMENTS

5,579,472	*	11/1996	Keyworth, II et al.	395/326
5,809,415		9/1998	Rossmann	455/422
5,822,692	*	10/1998	Krishan et al.	455/419
5,903,652	*	5/1999	Mital	380/25
5,933,478	*	8/1999	Ozaki et al.	379/93.24
5,987,609	*	11/1999	Hasebe	713/200
6,006,087	*	8/1999	Amin	455/413
6,014,559	*	1/2000	Amin	455/413
6,052,735	*	4/2000	Ulrich et al.	709/236
6,055,426	*	4/2000	Beasley	455/432
6,065,120	*	5/2000	Laursen et al.	713/201
6,094,681	*	7/2000	Shaffer et al.	709/224
6,101,244	*	8/2000	Okada	379/100.08
6,108,688	*	8/2000	Nielson	709/206
6,181,736	*	1/2001	Mclaughlin et al.	375/222

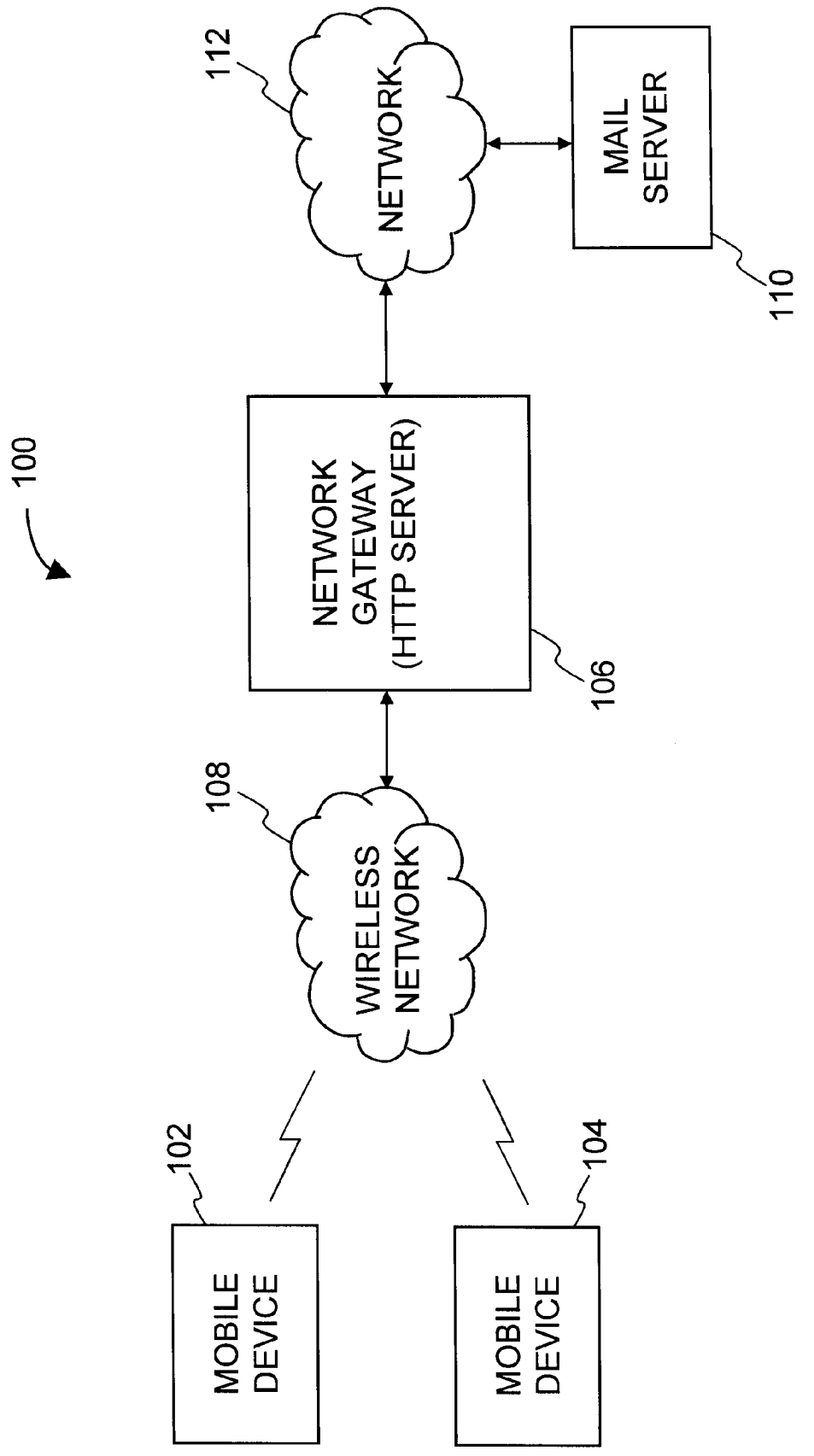

100
112
NETWORK
MAIL SERVER
110
NETWORK GATEWAY (HTTP SERVER)
106
108
WIRELESS NETWORK
102
MOBILE DEVICE
104
MOBILE DEVICE

Fig. 1

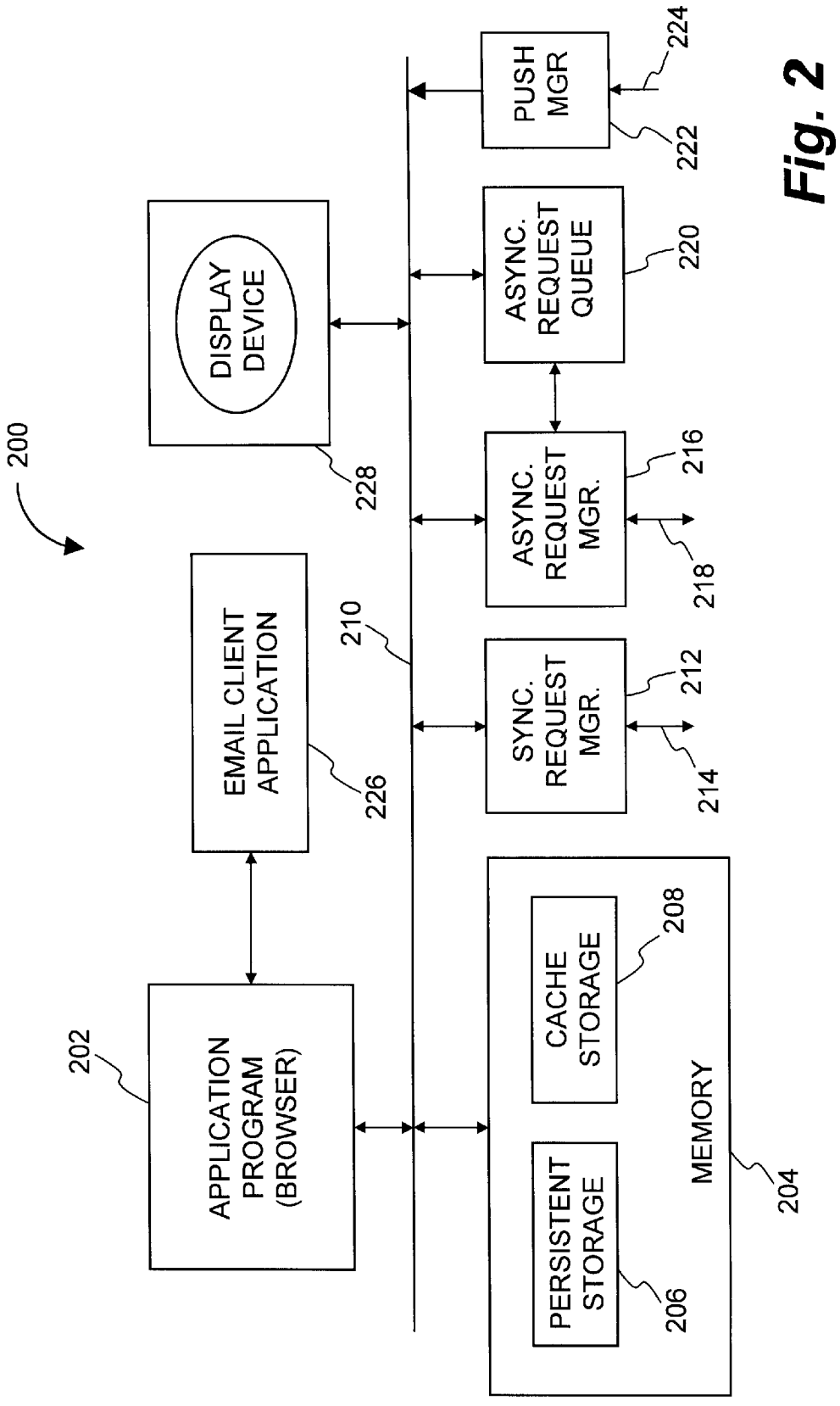

200
202
APPLICATION PROGRAM (BROWSER)
EMAIL CLIENT APPLICATION
226
DISPLAY DEVICE
228
210
MEMORY
204
CACHE STORAGE
208
PERSISTENT STORAGE
206
SYNC. REQUEST MGR.
212
214
ASYNC. REQUEST MGR.
216
218
ASYNC. REQUEST QUEUE
220
PUSH MGR
222
224

Fig. 2

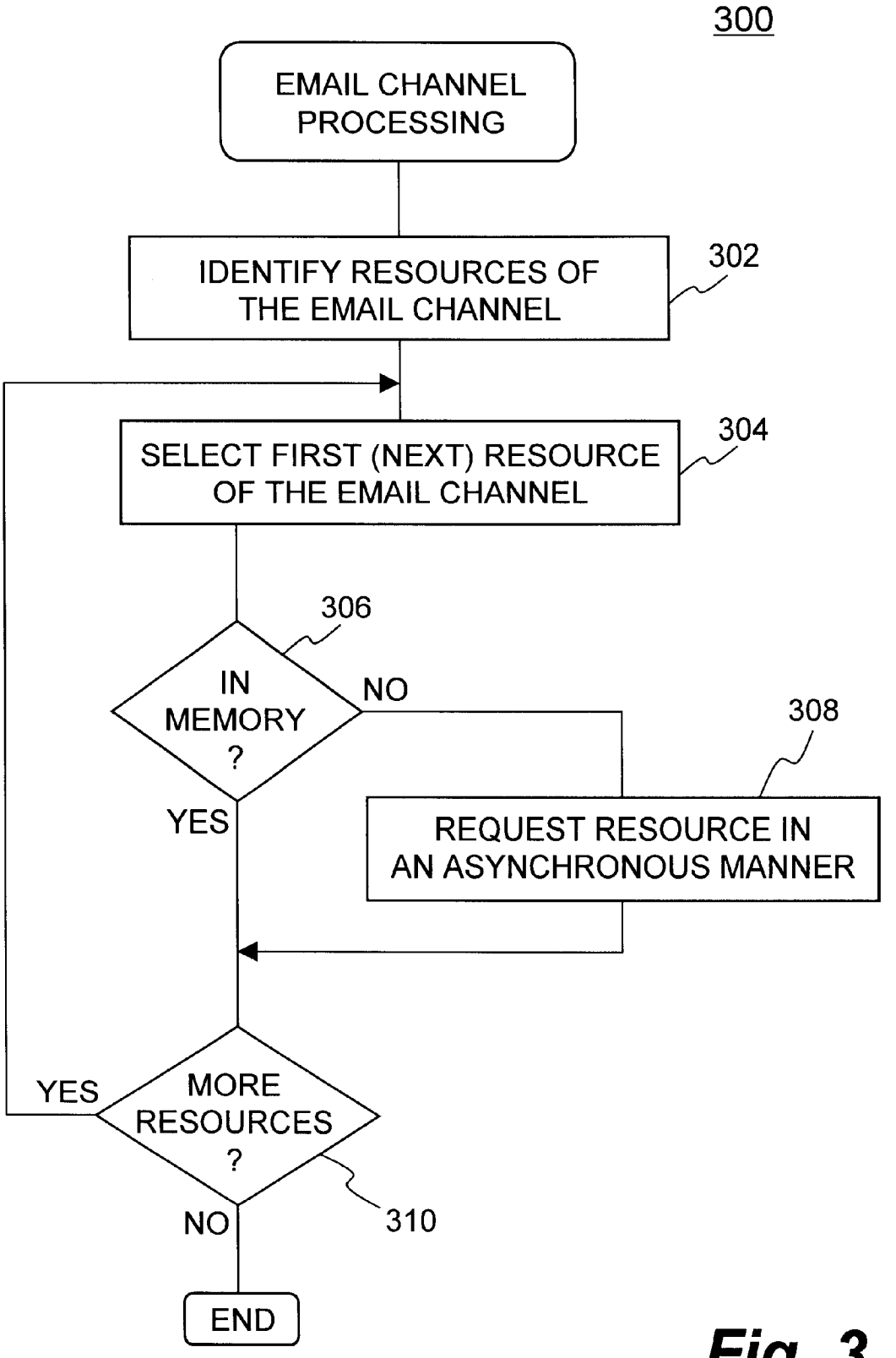

300
EMAIL CHANNEL PROCESSING
IDENTIFY RESOURCES OF THE EMAIL CHANNEL
302
SELECT FIRST (NEXT) RESOURCE OF THE EMAIL CHANNEL
304
306
IN MEMORY ?
NO
308
YES
REQUEST RESOURCE IN AN ASYNCHRONOUS MANNER
YES
MORE RESOURCES ?
310
NO
END

Fig. 3

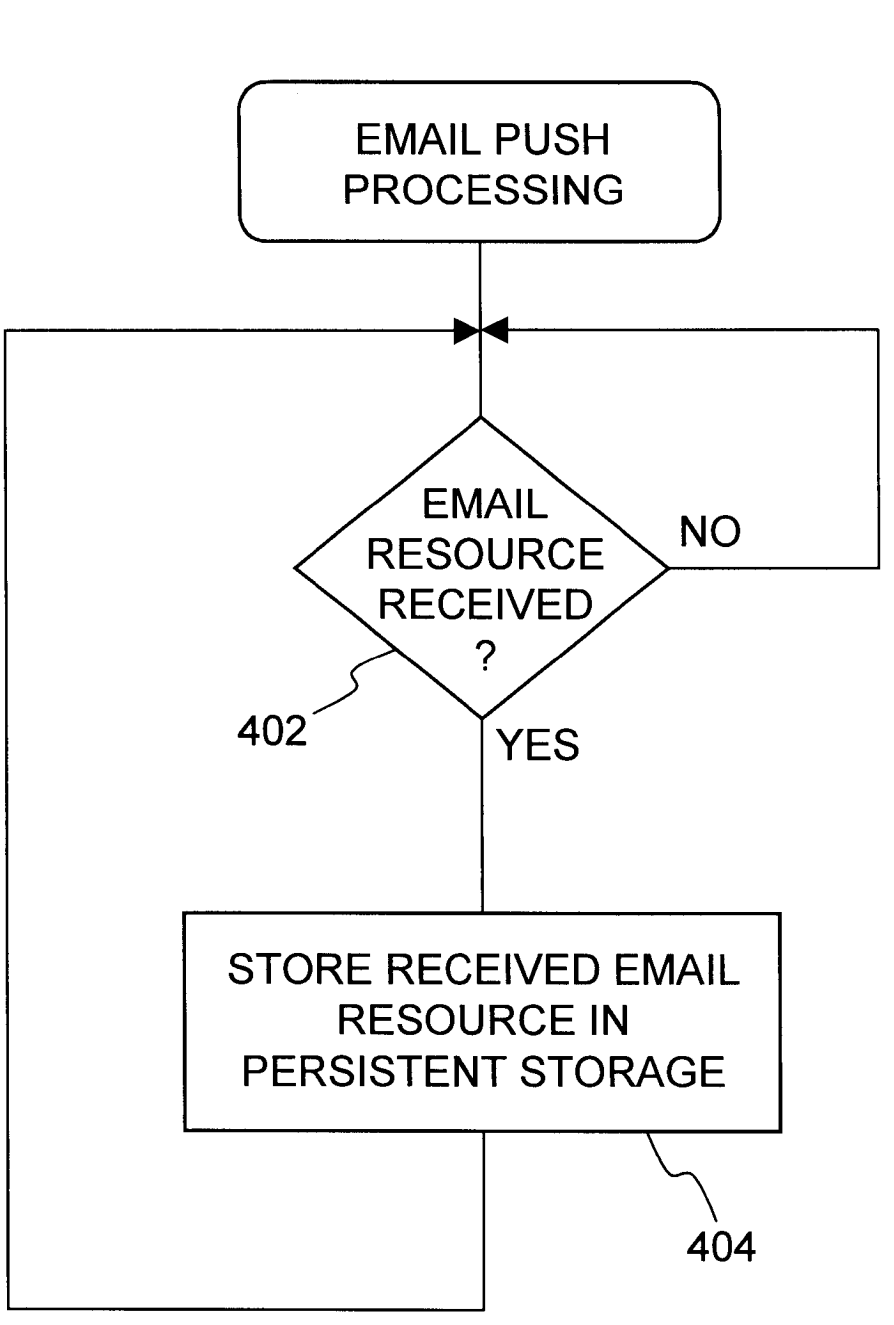

400
EMAIL PUSH PROCESSING
EMAIL RESOURCE RECEIVED ?
NO
YES
402
STORE RECEIVED EMAIL RESOURCE IN PERSISTENT STORAGE
404

Fig. 4

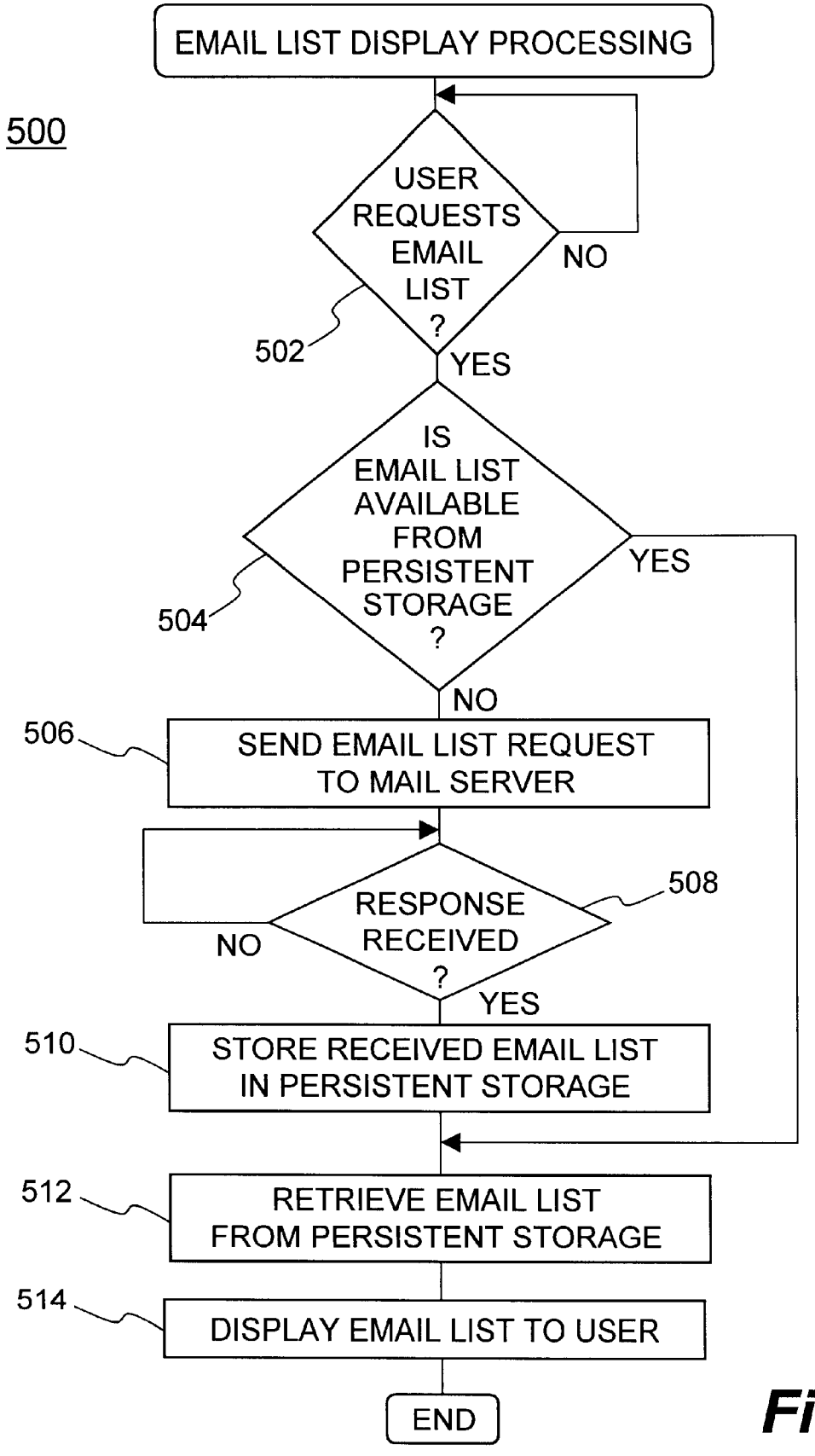

EMAIL LIST DISPLAY PROCESSING
500
USER REQUESTS EMAIL LIST ?
NO
502
YES
IS EMAIL LIST AVAILABLE FROM PERSISTENT STORAGE ?
YES
504
NO
506
SEND EMAIL LIST REQUEST TO MAIL SERVER
RESPONSE RECEIVED ?
508
NO
YES
510
STORE RECEIVED EMAIL LIST IN PERSISTENT STORAGE
512
RETRIEVE EMAIL LIST FROM PERSISTENT STORAGE
514
DISPLAY EMAIL LIST TO USER
END

Fig. 5

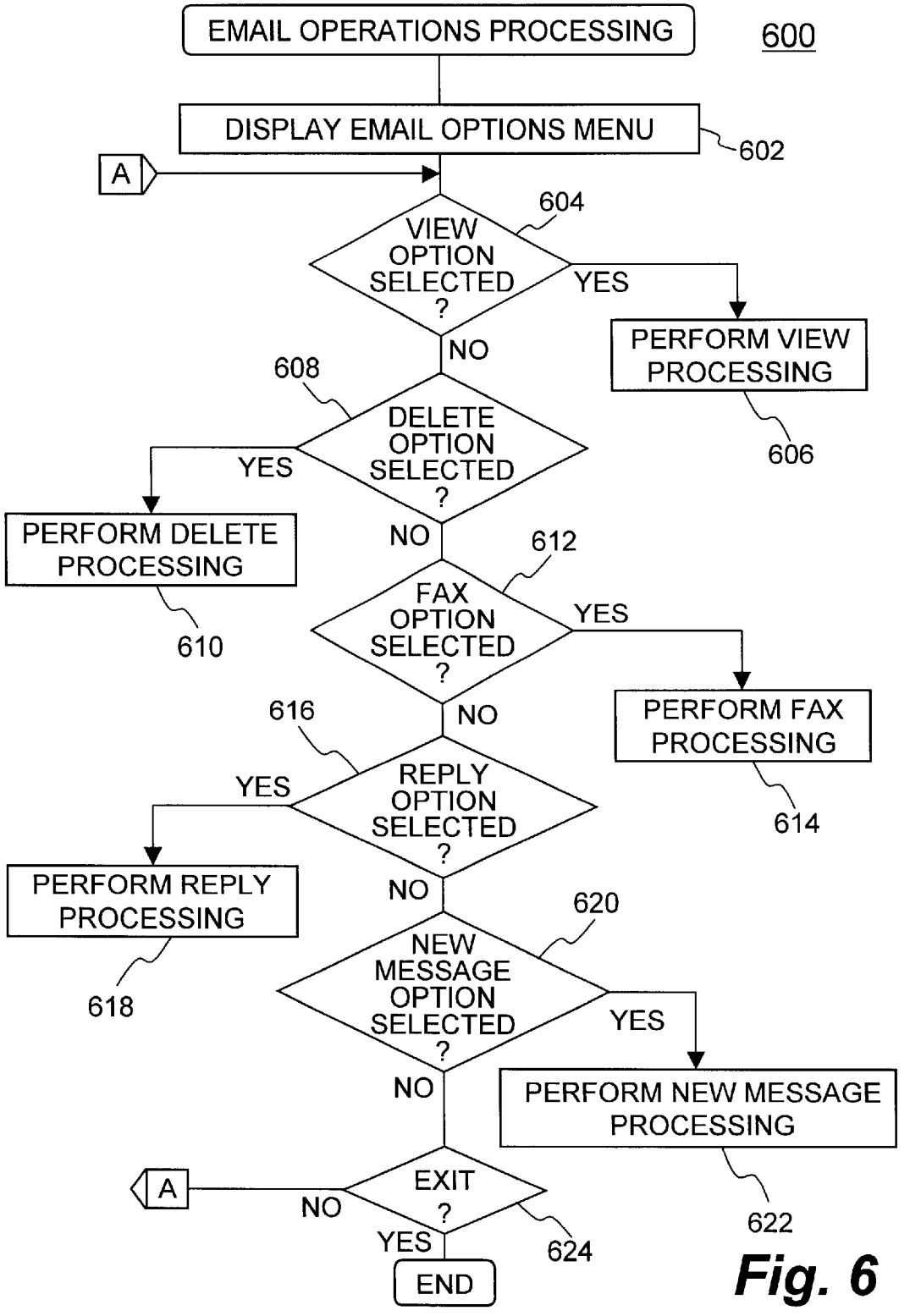

EMAIL OPERATIONS PROCESSING
600
DISPLAY EMAIL OPTIONS MENU
602
A
604
VIEW OPTION SELECTED ?
YES
PERFORM VIEW PROCESSING
606
NO
608
DELETE OPTION SELECTED ?
YES
PERFORM DELETE PROCESSING
610
NO
612
FAX OPTION SELECTED ?
YES
PERFORM FAX PROCESSING
614
NO
616
REPLY OPTION SELECTED ?
YES
PERFORM REPLY PROCESSING
618
NO
620
NEW MESSAGE OPTION SELECTED ?
YES
PERFORM NEW MESSAGE PROCESSING
622
NO
A
EXIT ?
NO
624
YES
END

Fig. 6

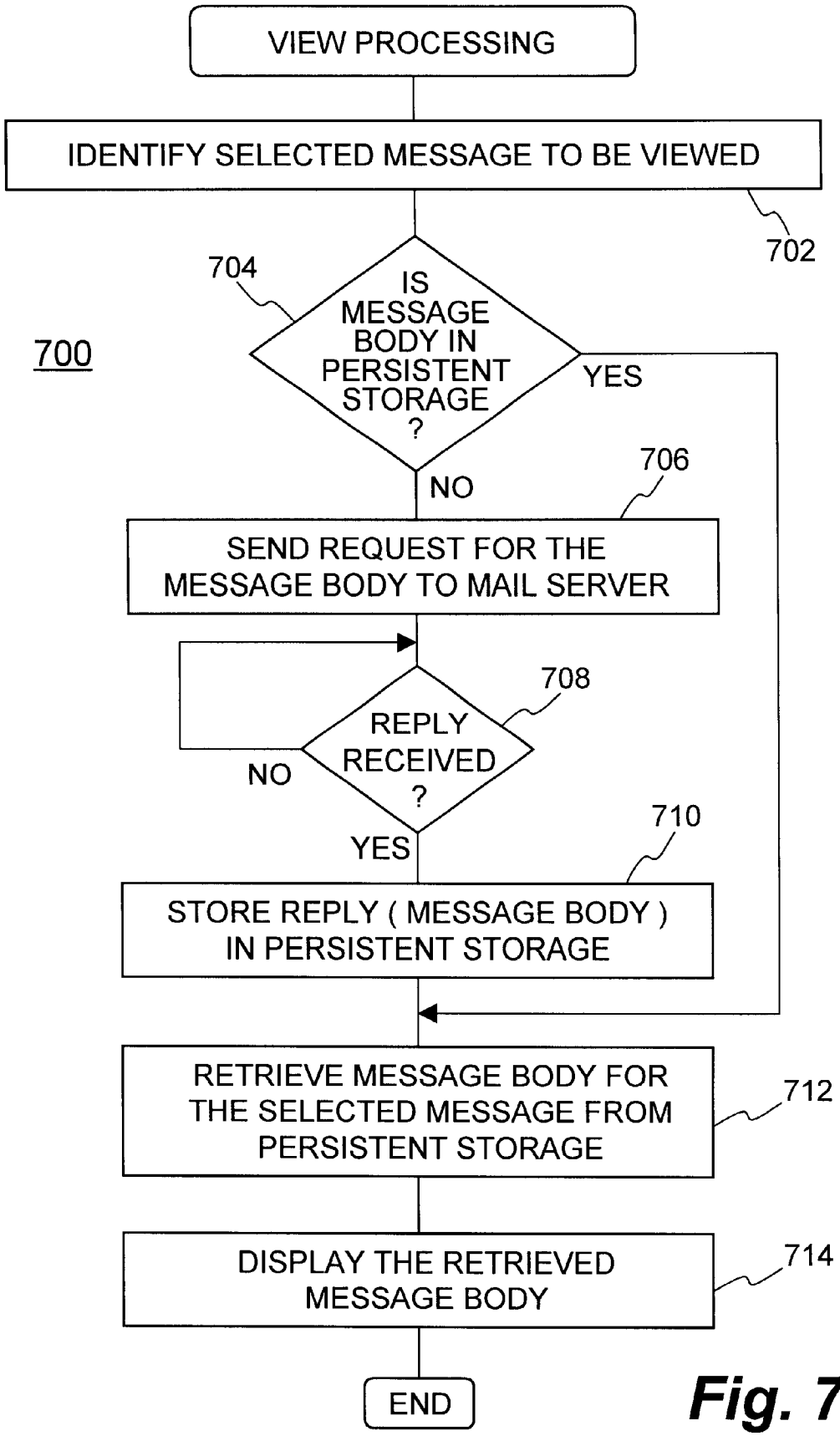

VIEW PROCESSING
IDENTIFY SELECTED MESSAGE TO BE VIEWED
702
704
IS MESSAGE BODY IN PERSISTENT STORAGE ?
700
YES
NO
706
SEND REQUEST FOR THE MESSAGE BODY TO MAIL SERVER
708
REPLY RECEIVED ?
NO
YES
710
STORE REPLY (MESSAGE BODY) IN PERSISTENT STORAGE
RETRIEVE MESSAGE BODY FOR THE SELECTED MESSAGE FROM PERSISTENT STORAGE
712
DISPLAY THE RETRIEVED MESSAGE BODY
714
END

Fig. 7A

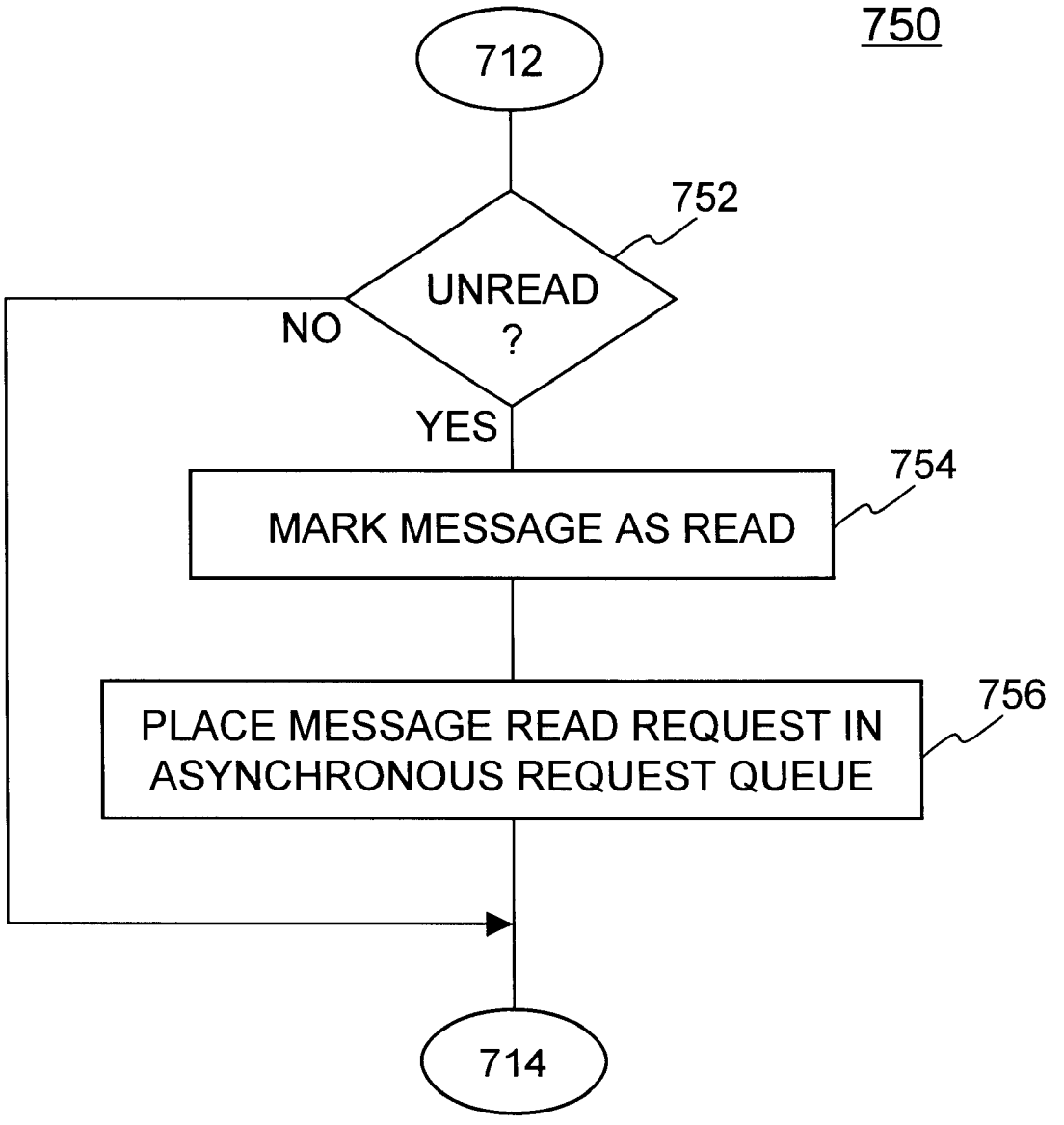

750
712
752
UNREAD ?
NO
YES
754
MARK MESSAGE AS READ
756
PLACE MESSAGE READ REQUEST IN ASYNCHRONOUS REQUEST QUEUE
714

Fig. 7B

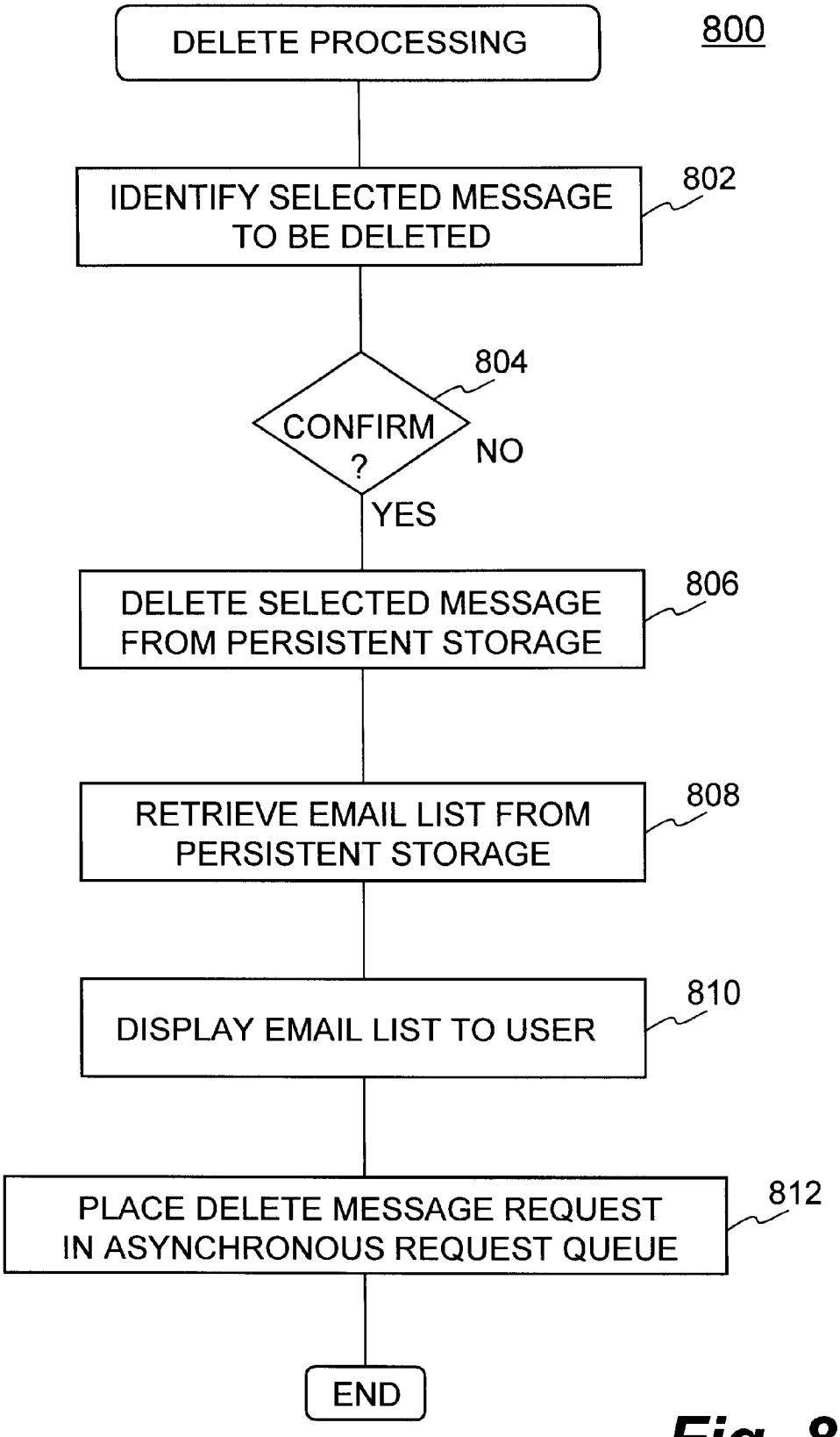

DELETE PROCESSING
800
IDENTIFY SELECTED MESSAGE TO BE DELETED
802
CONFIRM ?
804
NO
YES
DELETE SELECTED MESSAGE FROM PERSISTENT STORAGE
806
RETRIEVE EMAIL LIST FROM PERSISTENT STORAGE
808
DISPLAY EMAIL LIST TO USER
810
PLACE DELETE MESSAGE REQUEST IN ASYNCHRONOUS REQUEST QUEUE
812
END

Fig. 8

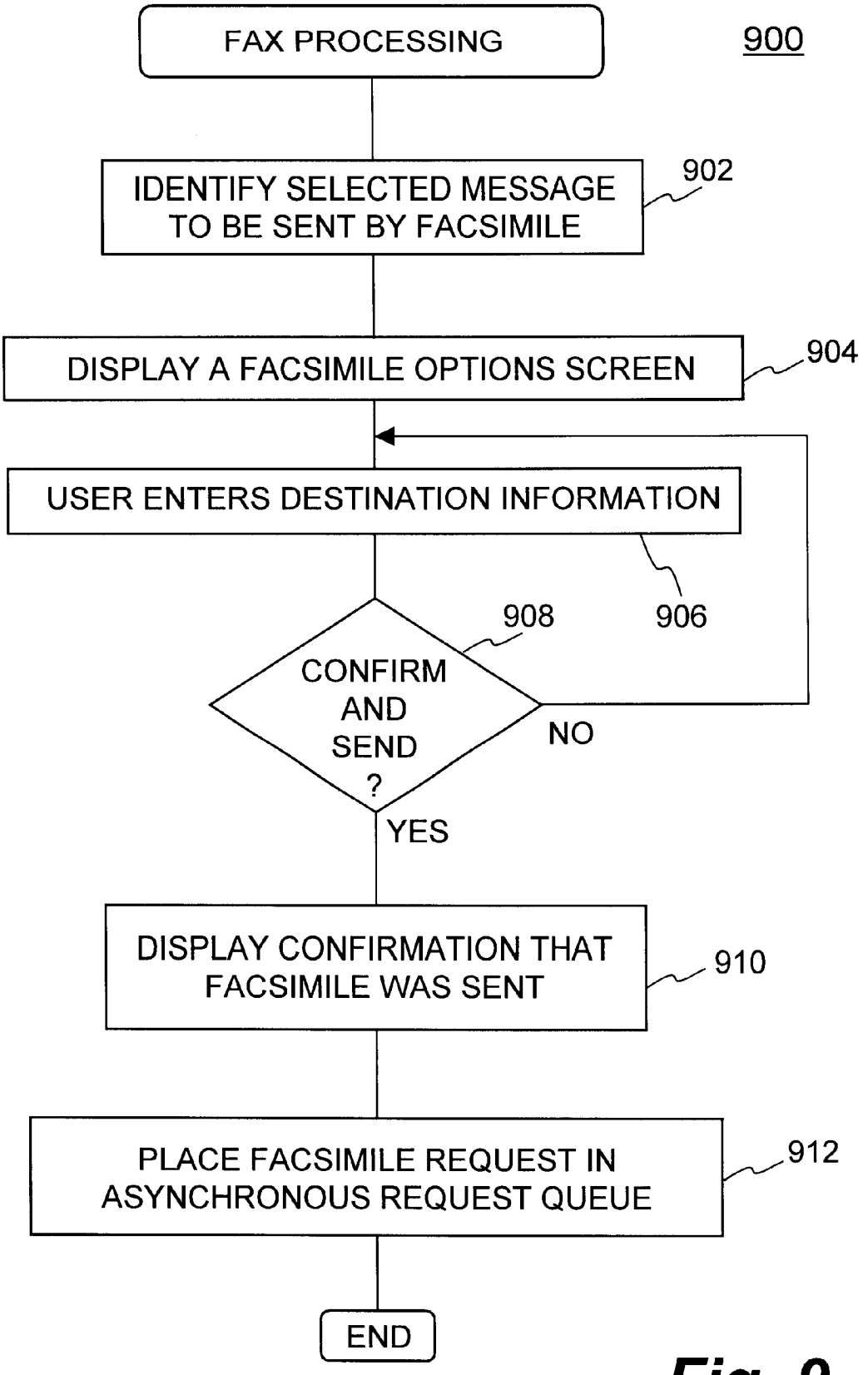

FAX PROCESSING
900
IDENTIFY SELECTED MESSAGE TO BE SENT BY FACSIMILE
902
DISPLAY A FACSIMILE OPTIONS SCREEN
904
USER ENTERS DESTINATION INFORMATION
906
908
CONFIRM AND SEND ?
NO
YES
DISPLAY CONFIRMATION THAT FACSIMILE WAS SENT
910
PLACE FACSIMILE REQUEST IN ASYNCHRONOUS REQUEST QUEUE
912
END

Fig. 9

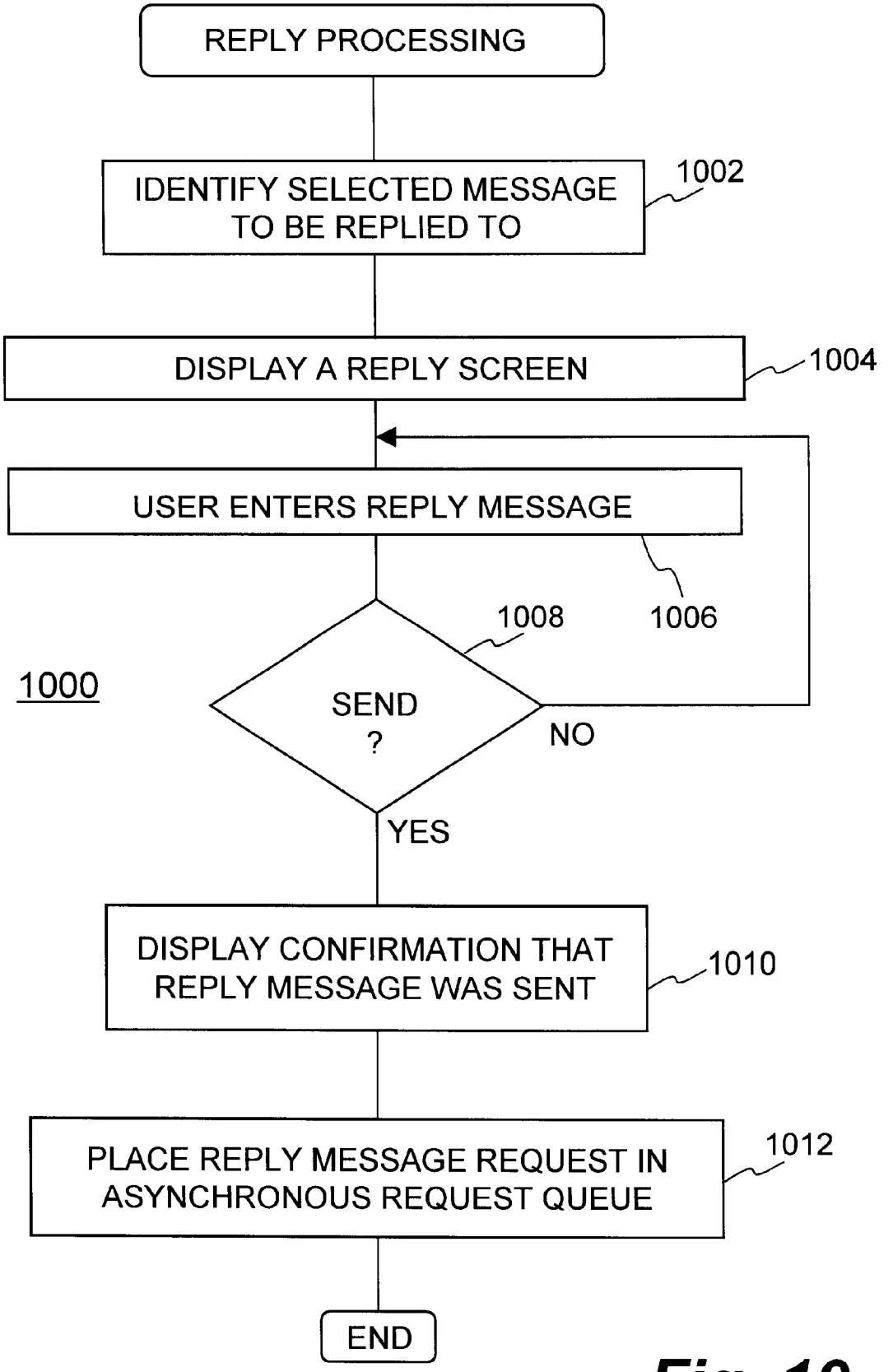

REPLY PROCESSING
IDENTIFY SELECTED MESSAGE TO BE REPLIED TO
1002
DISPLAY A REPLY SCREEN
1004
USER ENTERS REPLY MESSAGE
1006
1008
1000
SEND ?
NO
YES
DISPLAY CONFIRMATION THAT REPLY MESSAGE WAS SENT
1010
PLACE REPLY MESSAGE REQUEST IN ASYNCHRONOUS REQUEST QUEUE
1012
END

Fig. 10

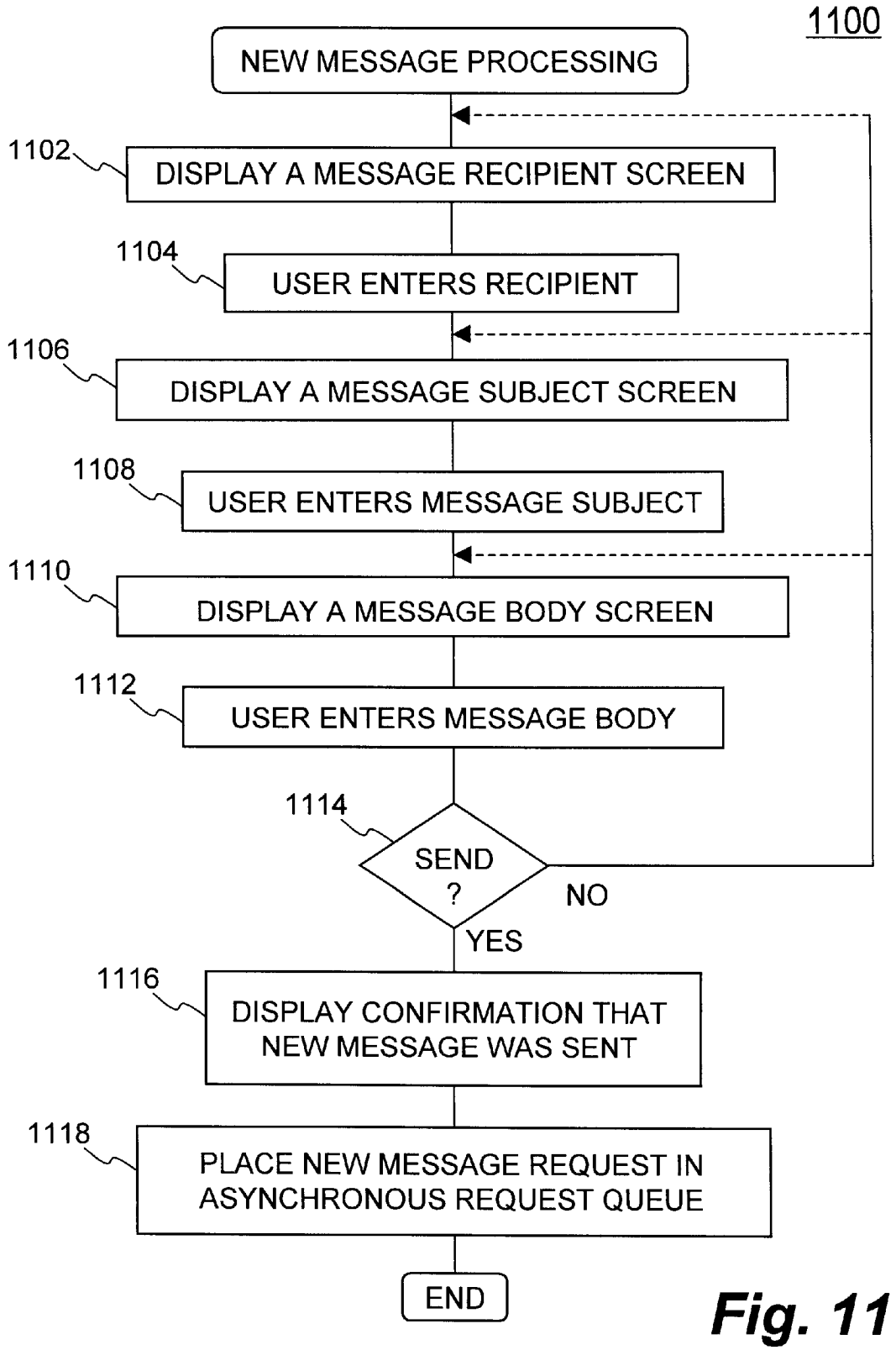

1100
NEW MESSAGE PROCESSING
1102
DISPLAY A MESSAGE RECIPIENT SCREEN
1104
USER ENTERS RECIPIENT
1106
DISPLAY A MESSAGE SUBJECT SCREEN
1108
USER ENTERS MESSAGE SUBJECT
1110
DISPLAY A MESSAGE BODY SCREEN
1112
USER ENTERS MESSAGE BODY
1114
SEND ?
NO
YES
1116
DISPLAY CONFIRMATION THAT NEW MESSAGE WAS SENT
1118
PLACE NEW MESSAGE REQUEST IN ASYNCHRONOUS REQUEST QUEUE
END

Fig. 11

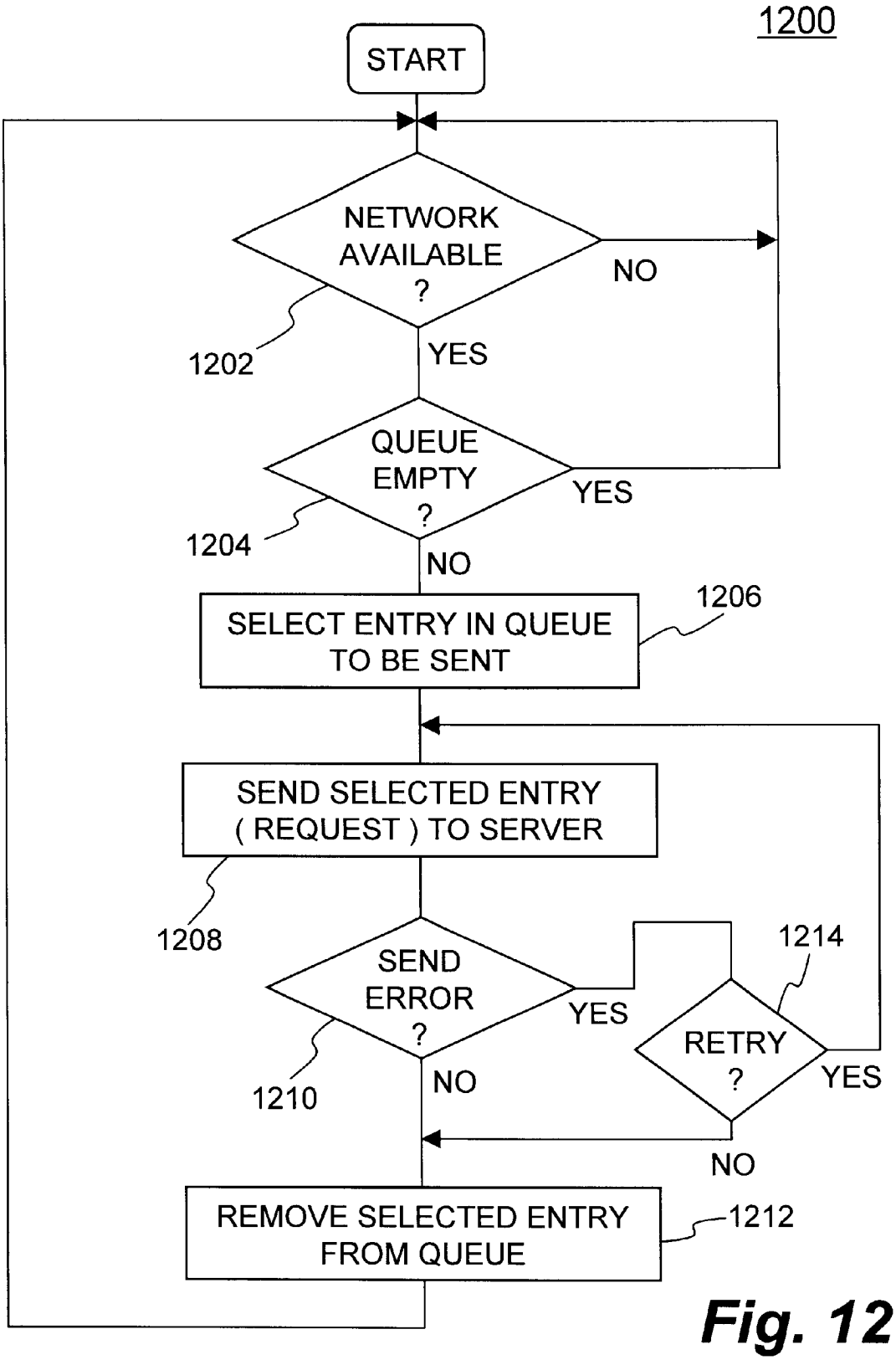

1200
START
NETWORK AVAILABLE ?
NO
1202
YES
QUEUE EMPTY ?
YES
1204
NO
SELECT ENTRY IN QUEUE TO BE SENT
1206
SEND SELECTED ENTRY (REQUEST) TO SERVER
1208
SEND ERROR ?
YES
1214
RETRY ?
YES
1210
NO
NO
REMOVE SELECTED ENTRY FROM QUEUE
1212

Fig. 12

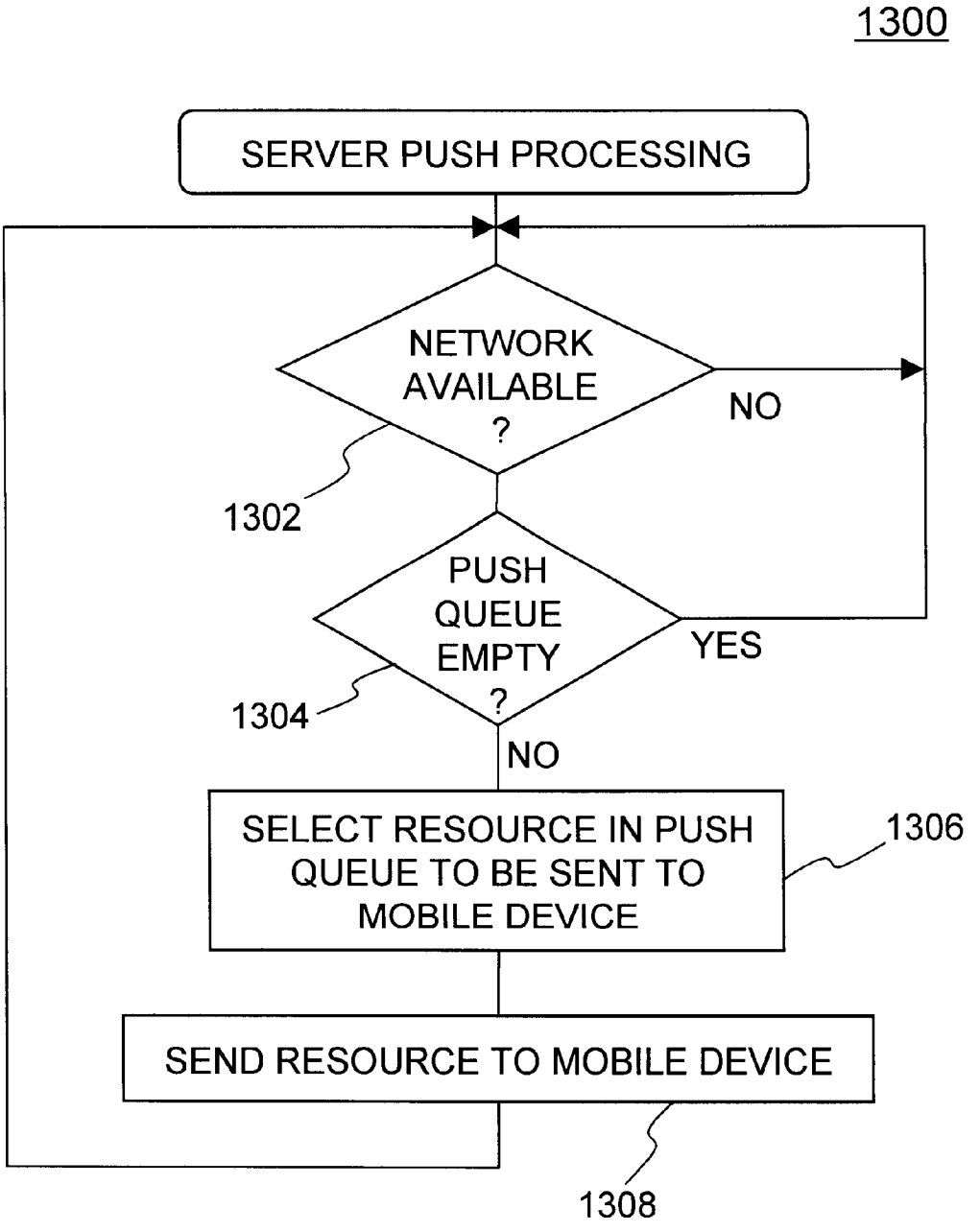

1300
SERVER PUSH PROCESSING
NETWORK AVAILABLE ?
NO
1302
PUSH QUEUE EMPTY ?
YES
1304
NO
SELECT RESOURCE IN PUSH QUEUE TO BE SENT TO MOBILE DEVICE
1306
SEND RESOURCE TO MOBILE DEVICE
1308

Fig. 13

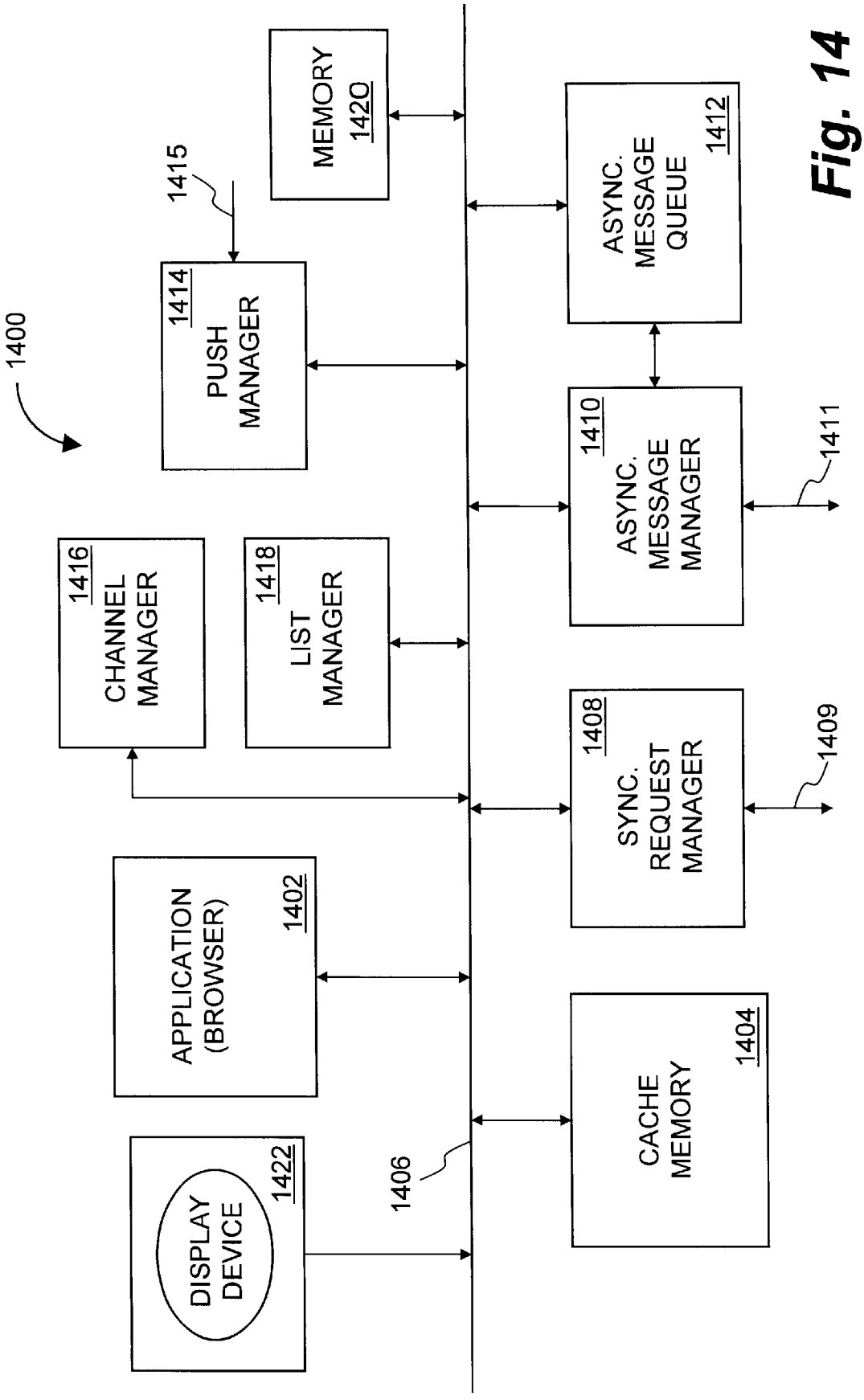

1400
MEMORY
1420
1415
1414
PUSH
MANAGER
ASYNC.
MESSAGE
QUEUE
1412
1410
ASYNC.
MESSAGE
MANAGER
1411
1416
CHANNEL
MANAGER
1418
LIST
MANAGER
1408
SYNC.
REQUEST
MANAGER
1409
APPLICATION
(BROWSER)
1402
CACHE
MEMORY
1404
1406
DISPLAY
DEVICE
1422

Fig. 14

METHOD AND APPARATUS FOR PROVIDING ELECTRONIC MAIL SERVICES DURING NETWORK UNAVAILABILITY

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/100,663, filed Sep. 16, 1998, and entitled "WIRELESS MOBILE DEVICES HAVING IMPROVED OPERATION DURING NETWORK UNAVAILABILITY", the content of which is hereby incorporated by reference. This application is also related to U.S. application Ser. No. 09/170,879, filed concurrently herewith, and entitled "WIRELESS MOBILE DEVICES HAVING IMPROVED OPERATION DURING NETWORK UNAVAILABILITY", the content of which is hereby incorporated by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to providing electronic mail services, and more particularly, to providing electronic mail services during network unavailability.

2. Description of the Related Art

Electronic mail (email) is a popular way to communicate with others. Electronic mail systems operate to send messages over a network. The network can include internal networks and external (e.g., public) networks. An example of an internal network is a corporate network, and an example of an external network is the Internet. Typically, the electronic mail systems are corporate wide systems that reside on an internal network but also permit coupling to an external network so that messages can be exchanged with other electronic mail systems.

Recently, Internet-based electronic mail systems have been developed and implemented to provide electronic mail services over the Internet. In such systems, there is no internal network because the electronic mail system resides on the Internet. The electronic mail system has a mail server that interacts with users' network browsers so that users are able to request electronic mail services which are performed by the mail server. Typically, the mail server is a Hyper Text Transfer Protocol (HTTP) server.

These Internet-based electronic mail systems have been implemented over wired networks as well as wireless networks. The availability of the network determines whether a client application (e.g., network browser) is in communication with the mail server. The client application operates on a local machine, whereas the mail server operates on a remote machine. In the case of wired networks, the local machine is, for example, a desktop computer. In the case of wireless networks, the local machine is a mobile device. For example, the mobile device can be a mobile telephone, a personal digital assistant (PDA) or a portable computer that has wireless access to the mail server.

One problem with the conventional electronic mail systems is that the operation of client applications are very much dependent on availability of their network. In other words, when the network is unavailable, the client applications can become "hung", namely stopping further processing until a response from a remote server via the network is received. The network can be a wired network or a wireless network. Unavailability of a wired network can be due to high congestion or server failure. Unavailability of a wireless network can result from a user of a mobile device

(supporting the client application) exceeding the geographic range of coverage. Unavailability also effectively results from a wireless network having high latencies, sporadic connectivity, high congestion or server failure. Because of the dependency of the operation of client applications on the availability of networks, client applications often have to wait for resources from a mail server. These wait times are unacceptably long when the network is unavailable to the client applications.

Computers or mobile devices are often provided with cache memories that temporarily store previously requested and obtained resources from remote servers. A cache memory is helpful in reducing the dependency of the computers or mobile devices on network availability. However, the cache memory is only helpful when the newly requested resource happens to reside in the cache memory. Hence, if the newly requested resource was not previously requested, then the newly requested resource would not be stored in the cache memory. Also, even if the newly requested resource were at one point in time stored in the cache memory, a reclamation or clean-up algorithm could have removed it from the cache memory to provide space for newer requested resources. Still further, the cache memory has to be relatively large to store all the resources likely to be requested. However, mobile devices (particularly hand-held mobile devices) need to keep cache memories relatively small due to power, cost and space limitations.

In the case of Internet-based electronic mail systems, a network browser operates on the computer or mobile device to enable access and manipulation their electronic mail residing on a mail server over the Internet. Since the network browsers often request electronic mail resources from remote mail servers over the network, the network browsers are particularly sensitive to network unavailability (e.g., due to out of coverage, high latencies, or sporadic connectivity). Consequently, even with a conventional cache memory, it is common for network unavailability to induce significant delays for the users of network browsers on computers or mobile devices.

Thus, there is a need for techniques to reduce delays faced by users seeking to perform electronic mail services with a mail server across a network.

SUMMARY OF THE INVENTION

Broadly speaking, the invention relates to improved techniques for providing electronic mail services across a network. A mail server and its clients communicate through the network. Although the mail server centrally manages the electronic mail services, the clients are able to themselves locally perform certain electronic mail services when the network is unavailable. Accordingly, clients seeking to perform electronic mail services no longer endure significant delays when the network is unavailable. The network can be at least temporarily unavailable for a variety of reasons, including: congestion, out of range, network failure, etc. The network can be wired or wireless. The invention is particularly well suited for networks having sporadic connectivity, high latencies or excessive traffic.

The invention can be implemented in numerous ways, including as a method, a computer readable medium, an apparatus, and a system. Several embodiments of the invention are discussed below.

As a mobile device for use with a wireless data communication network, an embodiment of the invention includes: a memory storage device that stores electronic mail resources; an electronic mail processor that performs an

electronic mail operation with respect to the electronic mail resources stored in said memory storage device, and where the electronic mail operation can be carried out at said mobile device even when the wireless data communication network is not available to said mobile device; and a display device that displays at least a part of one or more of the electronic mail resources.

As a method for interacting with electronic mail messages on a mobile device, the mobile device being able to communicate with a mail server at least in part through a wireless data network, an embodiment of the invention includes the acts of: pre-loading electronic mail message resources into a storage device of the mobile device; receiving a request to view an electronic mail list; determining whether the electronic mail list is available from the storage device of the mobile device; receiving the electronic mail list from the storage device when the electronic mail list is determined to be available from the storage device of the mobile device; requesting and subsequently receiving the electronic mail list from the mail server when the electronic mail list is determined not to be available from the storage device of the mobile device; and displaying the received electronic mail list.

As a method for interacting with electronic mail messages on a mobile device, the mobile device being able to connect to a remote mail server through a wireless data network, an embodiment of the invention includes: displaying an electronic mail list on a display screen of the mobile device, the electronic mail list including one or more entries that identify particular electronic mail messages; selecting one of the entries of the electronic mail list being displayed on the display screen of the mobile device; performing an operation on the electronic mail message associated with the selected entry without delay due to the unavailability of the wireless data network to the mobile device; and asynchronously sending a notification to the remote mail server based on the operation performed on the electronic mail message associated with the selected entry when the wireless data network is available to the mobile device.

As a computer readable medium including computer program code for interacting with electronic mail messages on a computing device, the computing device being able to communicate with a mail server at least in part through a data network, an embodiment of the invention includes: computer program code configured to pre-load electronic mail message resources into a storage device of the computing device; computer program code configured to receive a request to view an electronic mail list; computer program code configured to determine whether the electronic mail list is available from the storage device of the computing device; computer program code configured to receive the electronic mail list from the storage device when the electronic mail list is determined to be available from the storage device of the computing device; computer program code configured to request and subsequently receive the electronic mail list from the mail server when the electronic mail list is determined not to be available from the storage device of the computing device; and computer program code configured to display the received electronic mail list.

As a computer readable medium for interacting with electronic mail messages on a computing device, the computing device being able to connect to a remote mail server through a data network, an embodiment of the invention includes: computer program code configured to display an electronic mail list on a display screen of the computing device, the electronic mail list including one or more entries that identify particular electronic mail messages; computer program code configured to select one of the entries of the electronic mail list being displayed on the display screen of the computing device; computer program code configured to perform an operation on the electronic mail message associated with the selected entry without delay due to the unavailability of the wireless data network to the computing device; and computer program code configured to asynchronously send a notification to the remote mail server based on the operation performed on the electronic mail message associated with the selected entry when the data network is available to the computing device.

The advantages of the invention are numerous. Several advantages that embodiments of the invention may include are as follows. One advantage of the invention is that electronic mail services can be performed on electronic mail messages even when the network is unavailable. Clients, e.g., mobile devices, are able to perform electronic mail services regardless of network availability. As a result, clients experience better responsiveness and less waiting. Another advantage of the invention is that a mail server located on the network is able to be kept current by use of asynchronous messaging. Still another advantage of the invention is the intelligent management of the memory resources of the clients which are being consumed by the electronic mail services.

Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings which illustrate, by way of example, the principles of the invention.

BRIEF DESCRIPTION Of THE DRAWINGS

The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

FIG. **1** is a block diagram of a wireless data communications system according to an embodiment of the invention;

FIG. **2** is a block diagram of a mobile device according to an embodiment of the invention;

FIG. **3** is a flow diagram of electronic mail channel processing;

FIG. **4** is a flow diagram of electronic mail push processing according to an embodiment of the invention;

FIG. **5** is a flow diagram of electronic mail list display processing according to an embodiment of the invention;

FIG. **6** is a flow diagram of electronic mail operations processing according to an embodiment of the invention;

FIG. **7**A is a flow diagram of view processing according to an embodiment of the invention;

FIG. **7**B is a flow diagram of an optional enhancement to the view processing **700** according to an embodiment of the invention;

FIG. **8** is a flow diagram of delete processing according to an embodiment of the invention;

FIG. **9** is a flow diagram of fax processing according to an embodiment of the invention;

FIG. **10** is a flow diagram of reply processing according to an embodiment of the invention;

FIG. **11** is a flow diagram of new message processing according to an embodiment of the invention;

FIG. **12** is a flow diagram of asynchronous request send processing according to an embodiment of the invention;

FIG. **13** is a flow diagram of server push processing according to an embodiment of the invention; and

FIG. **14** is a block diagram of a mobile device according to another embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention relates to improved techniques for providing electronic mail services across a network. A mail server and its clients communicate through the network. Although the mail server centrally manages the electronic mail services, clients are able to themselves locally perform certain electronic mail services when the network is unavailable. Accordingly, clients seeking to perform electronic mail services no longer endure significant delays when the network is unavailable. The network can be at least temporarily unavailable for a variety of reasons, including: congestion, out of range, network failure, etc. The network can be wired or wireless. The invention is particularly well suited for networks having sporadic connectivity, high latencies or excessive traffic.

While the invention is useful for both wired and wireless networks, the invention is described below with reference to a wireless communication system since wireless networks more often suffer network unavailability. In a wireless communication system, a wireless network (wireless carrier network) generally supports connection of a plurality of mobile devices to a wired network. The mobile devices communicate with server machines on the wired network to request and receive various resources. The wired network can be of different types. One type of wired network is the Internet. The invention pertains to the facilitating of operation of mobile devices when the wireless or wired network is unavailable such that the mobile devices are unable to communicate with the server machines, such as a mail server.

Embodiments of the invention are discussed below with reference to FIGS. **1–14**. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for explanatory purposes as the invention extends beyond these limited embodiments.

FIG. **1** is a block diagram of a wireless data communications system **100** according to an embodiment of the invention. The wireless data communication system **100** includes mobile devices **102** and **104**. Normally, the wireless data communication system **100** supports a large number of mobile devices, and thus the mobile devices **102** and **104** are representative of the mobile devices used. These mobile devices **102** and **104** can couple to a network gateway **106** through a wireless network **108**. The network gateway **106** can also be referred to as a proxy server or wireless data server. The network gateway **106** is able to exchange information with a mail server **110**. The network gateway **106** and the mail server **110** are interconnected through a network **112**. The mail server **110** manages the storage and delivery of electronic mail messages to the appropriate location (e.g., the mobile devices **102** and **104**). Typically, the network **112** is a wired network. As an example, the network **112** can be a local area network (LAN), a wired area network (WAN), the Internet, or some combination thereof. In one embodiment, the network **112** is the Internet and the network gateway **106** and the mail server **110** are HTTP servers.

Conventionally, all the resources associated with the electronic mail service were stored only on the mail server. Hence, the mobile devices had to be in network communication with the mail server in order to perform electronic mail operations, such as viewing, deleting or creating electronic mail messages. This is also true for wired communication systems where local computers are required to be in network communication with a mail server in order to perform electronic mail operations. When in network communication, the mobile devices (or local computers) would request a list of electronic mail from the mail server through the network. Once the request list of electronic mail is received, the list would be displayed to a user. The user could then perform one of the electronic mail operations on one of the listed electronic mail messages. However, the electronic mail operations were all performed by the mail server and thus network communication was required. For example, if a user requested to delete a certain message from the list of electronic mail, then the request would be sent to the mail server. If the network was unavailable, the request would not be sent and thus the mobile device (or local computer) would wait for the network availability to return or eventually the mobile device (or local machine) would give up. Once received at the mail server, the mail server would process the delete request and then modify the list of electronic mail for the mobile device (or local machine). The modified list of electronic mail would then be sent to the mobile device (or local machine), again requiring network availability.

Due to sporadic connectivity, high latencies, congestion, range limitations, obstructions, network failures, etc., networks are occasionally unavailability (at least temporarily) to mobile devices (or local machines). During such network unavailability, conventional electronic mail systems do not allow the mobile machines (local machines) to complete electronic mail operations. Often, this causes the mobile device (or local machine) to incur delays which are unsatisfactory to the users of the mobile devices (or local machines). Thus, the conventional approaches to providing electronic mail to local devices are unsatisfactory. Namely, when the wireless network is not available to the mobile devices, the mobile devices are not able to perform any electronic mail functions.

According to the invention, the mobile devices **102** and **104** are able to perform many electronic mail functions even though the network is unavailable to the mobile devices. The mobile devices **102** and **104** are no longer dependent on the availability of network connectivity to the mail server **110** in order to provide electronic mail functions on the mobile devices **102** and **104**.

The mobile devices can take a variety of forms. Examples of mobile devices include mobile computing devices, cellular or mobile phones, portable computer devices, personal digital assistant (PDA) devices.

FIG. **2** is a block diagram of a mobile device **200** according to an embodiment of the invention. The mobile device **200** is, for example, suitable for use as the mobile device **102** or the mobile device **104** illustrated in FIG. **1**. The mobile device **200** is particularly suited for wireless communications through a wireless network where connectivity is sporadic or high-latency conditions are present.

The mobile device **200** includes an application program that operates on the mobile device **200**. In one embodiment, the application program **202** is a network browser. In one embodiment, the network browser is a micro-browser. A micro-browser is a network browser designed for a small screen interface such as with hand-held mobile devices. As an example, a micro-browser is produced by Unwired Planet, Inc. located at 800 Chesapeake Drive, Redwood City, Calif. 94063. The application program **202** interacts with a memory **204** that stores data for use by the application

program **202**. The memory **204** includes a persistent storage **206** and a cache storage **208**. The persistent storage **206** is an area of the memory **204** that is protected from cache replacement or clean-up processing. The cache storage **208**, on the other hand, is subject to cache replacement and cach-cleanup processing as is normal with cache type memory operations. A communication link **210** (or interface) connects with the memory **204** to the application program **202**. The memory **204** is also often of a limited size due to limitations on size and power for mobile devices, particularly hand-held mobile devices.

The mobile device **200** also includes a synchronous request manager **212** that couples to the communication link **210**. The synchronous request manager **212** manages synchronous requests for resources by the mobile device **200** to the mail server **110** (or other remote servers) via a communication link **214**. In response to the synchronous requests, the synchronous request manager **212** also receives resources from the mail server **110** (or other remote servers) via the communication link **214**. An asynchronous request manager **216** is also coupled to the communication link **210**. The asynchronous request manager **216** operates to manage asynchronous requests for resources from the mobile device **200** to the mail server **110** (or other remote servers) via a communication link **218**. The asynchronous request manager is connected to an asynchronous request queue **220** that is also connected to the communication link **210**. The asynchronous request queue **220** stores asynchronous requests that are to be transmitted by the asynchronous request manger **216** to the mail server **110** (or other remote servers) when the network is available to the mobile device **200**. Also, when the network is available to the mobile device **200**, a push manager **222** receives incoming resources over a communication link **224**. The push manager **222** is coupled to the communication bus **210** and is thus able to store the incoming resources in the memory **204**.

The mobile device **200** also includes an electronic mail client application **226**. The electronic mail client application **226** is coupled with the application program **202** so that electronic mail functions (operations) can be performed on the mobile device **200**. As is explained in more detail below, the electronic mail client application **226** is able to operate properly even when the network is unavailable. The mobile device **200** also includes a display device (or screen) **228** that is coupled to the communication link **210**. Among other things, the display device **228** operates to display electronic mail information for a user of the mobile device **200**.

FIG. **3** is a flow diagram of electronic mail channel processing **300**. The electronic mail channel processing **300** is performed by a mobile device when operating to pre-load channel resources into a memory of the mobile device. As an example, the channel resources can be pre-loaded into the memory **204** (namely, the persistent storage **206**) of the mobile device **200** illustrated in FIG. **2**. The advantage of the pre-loading is that the resources needed by the mobile device in interacting with the electronic mail are rapidly available regardless of network availability because the resources are resident on the mobile device.

The electronic mail channel processing **300** initially identifies **302** the resources of the electronic mail channel that is to be pre-loaded. Typically, various channels are available to the mobile device. Hence, a user of the mobile device is able to select those channels they desire to have resident in the memory (namely, the persistent storage **206**) of the mobile device **200**. Here, it is assumed that the electronic mail channel is to be pre-loaded into the memory of the mobile device. The identification of the resources of the electronic mail channel are those resources that would be needed by the mobile device when performing operations associated with the electronic mail on the mobile device. In one embodiment, the resources of the electronic mail channel include: electronic mail message list; contents of the messages; menu and data entry screens for electronic mail operations.

Following block **302**, a first resource of the electronic mail channel is selected **304**. Then, a decision block **306** determines whether the selected resource is found within the memory. Here, the memory pertains to the memory **204** (namely, the persistent storage **206**) illustrated in FIG. **2**. When the decision block **306** determines that the selected resource is not stored in the memory, then the resource is requested **308** from a remote server (e.g., the mail server **110**) in an asynchronous manner. For example, with respect to FIG. **2**, the resource being requested would be forwarded to the asynchronous request queue **220** and then eventually transmitted to the mail server **110** by the asynchronous request manager **216** when network availability permits. Because the request **308** for the resource is performed in an asynchronous manner, there is no need for the electronic mail channel processing **300** to await the arrival of the requested resource. Thus, the processing continues once the request for the resource is placed in the asynchronous request queue **220**. The arrival of the previously requested resources are received in the background, for example, via the push manager **222**.

Following block **308**, as well as following the decision block **306** when the selected resource is in the memory, a decision block **310** determines whether there are more resources of the electronic mail channel to be pre-loaded. When the decision block **310** determines that there are more resources to be processed, the electronic mail channel processing **300** returns to repeat the block **304** and subsequent blocks. When repeating the block **304**, the next resource of the electronic mail channel is selected. Alternatively, when the decision block **310** determines that there are no more resources to be processed, the electronic mail channel processing **300** is complete and ends.

FIG. **4** is a flow diagram of electronic mail push processing **400** according to an embodiment of the invention. The electronic mail push processing **400** is performed on a mobile device. The electronic mail push processing **400** is the basic operation of the mobile device when receiving an incoming resource from the mail server over a push channel and arriving at the push manager **222** of the mobile device **200**.

The electronic push processing **400** begins a decision block **402**. The decision block **402** determines whether an electronic mail resource has been received. Typically, the mobile device can receive a variety of different types of resources over the push channel. In the case of the mobile device **200**, the incoming resources to the mobile device **200** would be received by the push manager **222** and stored in the memory **204**. In any event, the decision block **402** operates to invoke the electronic mail push processing **400** when an electronic mail resource has been received at the mobile device. When it is determined that an electronic mail resource has been received, the received electronic mail resource is stored **404** in persistent storage within the mobile device. For example, with respect to FIG. **2**, the received electronic mail resource can be stored in the persistent storage **206** of the memory **204**. Following block **404**, the electronic mail push processing **400** returns to the beginning of the electronic mail push processing **400** so as to process the next received electronic mail resource.

FIG. **5** is a flow diagram of electronic mail list display processing **500** according to an embodiment of the invention. The electronic mail list display processing **500** is, for example, performed by the mobile device **200** under the control of the electronic mail client application **226** and the application program **202**.

The electronic mail list display processing **500** begins with a decision block **502** that determines whether a user has requested an electronic mail list. When a user has not requested the electronic mail list, the decision block **502** causes the electronic mail list display processing **500** to be inactive. Once a user has requested the electronic mail list, then the electronic mail list display processing **500** is invoked.

Once evoked, a decision block **504** determines whether the electronic mail list is available from the persistent storage in the mobile device. For example, the electronic mail list will normally be stored in the persistent storage **206** of the memory **204**. While it is anticipated that in many cases the electronic mail list will be available from the persistent storage due to the pre-loading (see FIG. **3**), there are times in which the electronic mail list may not be available from the persistent storage. Hence, when the decision block **504** determines that the electronic mail list is not available from the persistent storage, an electronic mail list request is sent **506** to the mail server. Then, a decision block **508** determines whether a response to the electronic mail list request has been received. The response to the electronic mail list request would include the electronic mail list from the mail server. When the decision block **508** determines that the response has not yet been received, the electronic mail list display processing **500** awaits the reception of the response. In one embodiment, the electronic mail list request is a synchronous request that causes the mobile device to await the reception of the requested resource before processing continues. Such synchronous requests are, for example, handled by the synchronous request manager **212** illustrated in FIG. **2**. Once the decision block **508** determines that the response has been received, the received electronic mail list is stored **510** in the persistent storage of the mobile device. Alternatively, when the decision block **504** determines that the electronic mail list is available from the persistent storage (due to pre-loading), then blocks **506–510** are bypassed.

Following block **510**, as well as directly following the decision block **504** when the electronic mail list is available from the persistent storage, the electronic mail list is retrieved **512** from the persistent storage. Next, the electronic mail list is displayed **514** to the user. At this point, the electronic mail list has been displayed to the user and the system thereafter waits for additional operations to be requested by the user. Following block **514**, the electronic mail list display processing **500** is complete and ends.

FIG. **6** is a flow diagram of electronic mail operations processing **600** according to an embodiment of the invention. The electronic mail operations processing **600** assumes that the electronic mail list is presently displayed to the user of the mobile device (see FIG. **5**). When the electronic mail operations processing **600** begins, an electronic mail options menu is displayed **602** on the mobile device. The electronic mail options menu can be displayed concurrently or separately with the electronic mail list. Next, the electronic mail operations processing **600** effectively waits for a user input that selects a particular electronic mail operation (function) to be performed with respect to the electronic mail list. These electronic mail operations include viewing, deleting, faxing, replying and creating electronic mail messages.

Once a user input has been entered, the electronic mail operations processing **600** determines the particular electronic mail operation to be performed with respect to the electronic mail list. A decision block **604** determines whether a view option has been selected. When the decision block **604** determines that the view option has been selected, view processing is performed **606**. On the other hand, when the decision block **604** determines that the view option has not been selected, a decision block **608** determines whether a delete option has been selected. When the decision block **608** determines that the delete option has been selected, delete processing is performed **610**. When the decision block determines that the delete option has not been selected, then a decision block **612** whether a fax (facsimile) option has been selected. When the decision block **612** determines that the fax option has been selected, then fax processing is performed **614**. When the decision block **612** determines that the fax option has not been selected, then a decision block **616** determines whether a reply option has been selected. When the decision block **616** determines that a reply option has been selected, then reply processing is performed **618**. When the decision block **616** determines that the reply option has not been selected, a decision block **620** determines whether a new message option has been selected. When a decision block **620** determines that the new message option has been selected, new message processing is performed **622**. When the decision block **620** determines that the new message option has not been selected, a decision block **624** determines whether the user desires to exit the electronic mail operations processing **600**. When the decision block **624** determines that the user desires to exit the electronic mail operations processing **600**, then the electronic mail operations processing **600** is complete and ends. On the other hand, when the decision block **624** determines that the user does not desire to exit the electronic mail operations processing **600**, then the electronic mail operations processing **600** returns to repeat the decision block **604** and subsequent blocks so that user inputs with respect to electronic mail operations can be processed.

FIG. **7A** is a flow diagram of view processing **700** according to an embodiment of the invention. The view processing **700** is, for example, performed by the block **606** illustrated in FIG. **6**.

The view processing **700** initially identifies **702** the selected message that is to be viewed. Typically, the user has acted to select one of the electronic mail messages in the electronic mail list being displayed (see FIG. **5**), and then to request to view the selected message. In one embodiment, the selected message is identified by a message identifier. Next, a decision block **704** determines whether the message body associated with the selected message is stored in the persistent storage of the mobile device. When the message body is not stored in the persistent storage, a request for the message body is sent **706** to the mail server. This request for the message body is thus sent over the wireless data network to the mail server. Hence, when the network is unavailable to the mobile device, the request cannot be successfully transmitted and delays can occur. Following block **706**, a decision block **708** determines whether a reply to the request for the message body has been received from the mail server. When the decision block **708** determines that the reply (the message body) has not yet been received from the mail server, the view processing **700** awaits the arrival of the reply. Once the decision blocks **708** determines that the reply (including the message body) has been received, the reply is stored **710** in the persistent storage of the mobile device. In one embodiment, the synchronous request manager **212**

illustrated in FIG. **2** transmits the request and receives the reply from the mail server.

Alternatively, the blocks **706** through **710** of the view processing **700** are able to be bypassed when the decision block **704** determines that the message body already resides in the persistent storage of the mobile device. Thus, when the persistent storage has been pre-loaded to include the message body (i.e., a resource of the electronic mail channel) that is desired, the view processing **700** can operate without regard to network availability, and thus without network delays.

Following block **710**, as well as directly following the decision block **704** when the message body is found within the persistent storage, the message body for the selected message is retrieved **712** from the persistent storage. The retrieved message body is then displayed **714** on the display device of the mobile device. Following block **714**, the view processing **700** is complete and ends.

FIG. **7**B is a flow diagram of an optional enhancement to the view processing **700** according to an embodiment of the invention. The enchancement pertains to message marking processing **750** that can be performed between the block **712** and **714** of the view processing **700** illustrated in FIG. **7**A. The message marking processing **750** allows the mobile device to asynchronously (i.e., background process) inform the mail server of those messages that have been read. The message marking processing **750** initially determines at decision block **752** whether the selected message has already been read. If the selected message has not already been read, then the message is marked as having been read because the message is displayed in block **712** for the user to read the message. After marking the message (i.e., the local copy) as being read, a message read request is placed **756** in the asynchronous message queue. The message read request will serve to inform the mail server that a particular message has been read, and thus allow the mail server to keep track of those message the user has read.

FIG. **8** is a flow diagram of the delete processing **800** according to an embodiment of the invention. The delete processing **800** is, for example, performed by block **610** illustrated in FIG. **6**.

The delete processing **800** initially identifies **802** a selected message to be deleted. Typically, the user has acted to select one of the electronic mail messages in the electronic mail list being displayed (see FIG. **5**), and then to request to delete the selected message. Next, a decision block **804** determines whether the user has confirmed the deletion of the selected message. When the user does not confirm the deletion of the selected message, the delete processing **800** is complete and ends because the user has not confirmed its deletion. On the other hand, when the decision block **804** determines that the user has confirmed the deletion of the selected message, then the selected message is deleted **806** from the persistent storage. The electronic mail list is then retrieved **808** from the persistent storage. The electronic mail list is retrieved **808** from the persistent storage so that an updated electronic mail list is provided. The retrieved electronic mail list is then displayed **810** to the user. As an example, with respect to the mobile device **200** illustrated in FIG. **2**, the electronic mail list can be displayed on the display device **228**. Additionally, a delete message request is placed **812** in an asynchronous request queue. For example, with respect to FIG. **2**, the delete message request can be created by the electronic mail client application **226** and forwarded by the application program **202** to the asynchronous request queue **220**. Thereafter, the asynchronous request manager **216** can process the delete message request by sending it to the mail server when the network is available so that the mail server can update its database of messages. Following block **812**, the delete processing **800** is complete and ends.

FIG. **9** is a flow diagram of fax processing **900** according to an embodiment of the invention. The fax processing **900** is, for example, performed by the block **614** illustrated in FIG. **6**.

The fax processing **900** initially identifies **902** a selected message to be sent by facsimile. Typically, the user has acted to select one of the electronic mail messages in the electronic mail list being displayed (see FIG. **5**), and then to request to transmit the selected message to a facsimile machine. A facsimile option screen is then displayed **904**. The facsimile option screen allows a user to enter a destination information for a facsimile to be sent. The facsimile to be sent includes the contents of the selected message. Next, a user enters **906** the destination information which, for example, would include a facsimile telephone number and any suitable cover letter. A decision block **908** then determines whether the user has confirmed and requested to send the facsimile. When a decision block **908** determines that the user has not confirmed and requested to send the facsimile, the fax processing **900** awaits the user's confirmation and permits the user to alter the destination information. On the other hand, when the decision block **908** determines that the user has confirmed and requested to send the facsimile, a confirmation that the facsimile was sent is displayed **910**. Additionally, a facsimile request is placed **912** in an asynchronous request queue. For example, with respect to FIG. **2**, the facsimile request can be generated by the electronic mail client application **226** and forwarded by the application program **202** to the asynchronous request queue **220**. Thereafter, the asynchronous request manager **216** can send the facsimile request to the mail server when network availability permits. Following block **912**, the facsimile processing **900** is complete and ends.

FIG. **10** is a flow diagram of reply processing **1000** according to an embodiment of the invention. The reply processing **1000** is, for example, performed by block **618** illustrated in FIG. **6**.

The reply processing **1000** initially identifies **1002** a selected message that is to be replied to. Typically, the user has acted to select one of the electronic mail messages in the electronic mail list being displayed (see FIG. **5**), and then to request to reply to the selected message. A reply screen is then displayed **1004**. Next, the user then enters **1006** a reply message using the reply screen. Then, a decision block **1008** determines whether the user has requested to send the reply message. When the decision block **1008** determines that the user has not yet requested to send the reply message, the reply processing **1000** awaits the users request to send the reply message and permits the user to alter the reply message. Once the user has requested to send the reply message, a confirmation that the message was sent is displayed **1010**. Additionally, a reply message request is placed **1012** in an asynchronous request queue. For example, with respect to FIG. **2**, the reply message request can be generated by the electronic mail client application **226** and forwarded by the application program **202** to the asynchronous request queue **220**. Thereafter, the asynchronous request manager **216** can send the reply message request to the mail server when network availability permits. Following block **1012**, the reply processing **1000** is complete and ends.

FIG. **11** is a flow diagram of new message processing **1100** according to an embodiment of the invention. The new

message processing **1100** is, for example, performed by block **622** illustrated in FIG. **6**.

The new message processing **1100** initially displays **1102** a message recipient screen. A user then enters **1104** a recipient into the message recipient screen. Next, a message subject screen is displayed **1106**. A user then enters **1108** a message subject into the message subject screen. Next, a message body screen is displayed **1110**. A user then enters **1112** a message body into the message body screen. Following block **1112**, a decision block determines whether the user has requested to send the new message. When the user has not yet requested to send the message, the new message processing **1100** can return to repeat any of the prior blocks **1102–1112** so as to alter any previously entered information concerning the new message. On the other hand, once the decision block **1114** determines that the user has requested to send the message, then a confirmation that the new message was sent is displayed **1116**. Additionally, a new message request is placed **1118** in an asynchronous request queue. For example, with respect to FIG. **2**, the new message request can be generated by the electronic mail client application **226** and forwarded by the application program **202** to the asynchronous request queue **220**. Thereafter, the asynchronous request manager **216** can send the new message request to the mail server when network availability permits. Following block **1118**, the new message processing **1100** is complete and ends.

The asynchronous transmissions from the mobile device to the remote server (e.g., mail server) enable the mobile device to continue processing while these asynchronous transmissions occur in the background. Additionally, asynchronous reception (referred to as "push ") can also occur in the background. Asynchronous transmissions are described further below with respect to FIG. **12**, and asynchronous receptions are described further below with respect to FIG. **13**.

Although not shown in FIG. **6**, the electronic mail operations processing **600** can also include locking and unlocking electronic mail messages as another electronic mail operation. When a user interacts with the mobile device to "lock" a particular electronic mail message in an electronic mail list, the particular electronic mail message will remain locally stored and available in the persistent storage. In contrast, when an electronic mail message is not locked, it can be bumped out of the persistent storage to make room for a newer electronic mail message. A locked electronic mail message will remain in the persistent storage until deleted. Typically, the particular electronic mail message is locked by a user selecting a "lock" menu option when the electronic mail list is displayed on the display device. For visual feedback and notification, once a electronic mail message is locked, a distinctive icon can be displayed next to the electronic mail message in the electronic mail list.

Additionally, to facilitate easy entry of information into the various data entry screens for reply messages, fax destination information or new message information, one or more lists of most recently used data can be maintained (e.g., on the mobile device). A user is then able to enter the data for data input screen by selecting an entry in the appropriate list of most recently used data. For example, in FIG. **11**, with respect to block **1104**, a recent recipient list could be available to the user so that the user merely selects an entry in the recent recipient list without having to identify the recipient and their electronic mail address. Similarly, in FIG. **9**, with respect to block **906**, a recent facsimile numbers list and/or a recent recipient list could be available to the user for easy entry of data. In one embodiment, the number of entries stored in the list is small (e.g., 5–10) because of memory consumption and screen sizes with mobile devices. In any case, the entries in the various most recently used lists can be locked so that they are not removed from the lists.

FIG. **12** is a flow diagram of asynchronous request send processing **1200** according to an embodiment of the invention. The asynchronous request send processing **1200** is, for example, performed by the asynchronous request manager **216** illustrated in FIG. **2**.

The asynchronous request send processing **1200** begins with a decision block **1202** that determines whether a wireless network is available to the mobile device. Wireless networks often have sporadic connectivity or high latency due to out-of-range, congestion, etc. and thus are temporarily unavailable to mobile devices. When the decision block **1202** determines that the wireless network is not available, then the asynchronous request send processing **1200** simply awaits the availability of the wireless network. Once the wireless network becomes available, then a decision block **1204** determines whether an asynchronous request queue is empty. The asynchronous request queue is within the mobile device and is, for example, the asynchronous request queue **220** illustrated in FIG. **2A**.

When the decision block **1204** determines that the asynchronous request queue is empty, then the asynchronous request send processing **1200** returns to repeat the decision block **1202** and subsequent blocks because there is presently no requests waiting to be processed in the asynchronous request queue. On the other hand, when the asynchronous request queue is not empty, then an entry in the asynchronous request queue that is to be sent is selected **1206**. The selected entry in the asynchronous request queue is then sent **1208** to a server via the wireless network. The server can be a proxy server or a remote server (e.g., mail server) on a remote network. Typically, according to the invention, the selected entry is a request for a electronic mail resource located at a mail server on the remote network.

After sending the selected entry to the server, a decision block **1210** determines whether a send error has occurred. In other words, the decision block **1210** waits for an acknowledgment that the server has received the selected entry that has been sent. When the decision block **1210** determines that no send error occurred during the sending of the selected entry to server, then the selected entry is removed **1212** from the asynchronous request queue. On the other hand, when the decision block **1210** determines that a send error has occurred, a decision block **1214** determines whether a retry is desired to re-send the selected entry to the server. When the decision block **1214** determines that a retry is desired, then processing returns to repeat the block **1208** and subsequent blocks. Alternatively, when the decision block **1214** determines that a retry is not desired, then processing proceeds to the block **1212** where the selected entry is removed **1212** from the asynchronous request queue. Following block **1212**, the asynchronous request send processing **1200** returns to repeat the decision block **1202** and subsequent blocks so that additional entries in the queue can be processed.

FIG. **13** is a flow diagram of server push processing **1300** according to an embodiment of the invention. The server push processing **1300** is, for example, performed by a network gateway (or proxy server) such as the network gateway **106** illustrated in FIG. **1**.

The server push processing **1300** begins with a decision block **1302**. The decision block **1302** determines whether the network is available. Here, the decision block **1302** is

determining whether a wireless connection from the network gateway through the wireless network to the appropriate mobile device is available. When such network is not available, the server push processing **1300** is not evoked and merely awaits the availability of the network. When the network is available, a decision block **1304** determines whether a push queue is empty. The push queue contains resources (or replies to requests) that had been previously issued by the mobile device. In the case of electronic mail, the resources or replies to requests temporarily stored in the push queue are electronic mail resources. When the decision block **1304** determines that the push queue is empty, then there are no resources to be transmitted from the network gateway to the mobile device and thus the server push processing **1300** returns to repeat the decision block **1302** and subsequent blocks.

On the other hand, when the decision block **1304** determines that the push queue is not empty, then a resource in the push queue is selected **1306** to be sent to the mobile device. Then, the selected resource is sent **1308** to the mobile device. Following block **1308**, the server push processing **1300** returns to repeat the decision block **1302** so that additional push requests can be forwarded to the mobile device. At the mobile device, the resources pushed are received and managed by a push manager, such as the push manager **222** illustrated in FIG. **2**.

In one embodiment, the resources (e.g., electronic mail resources) stored in the push queue are limited in size. The resources can be limited in size anytime prior to being sent **1308** to the mobile device. For example, the mail server or the network gateway can operate to limit the size of the electronic mail resources. As an example, the size of the electronic mail resources can be limited to a predetermined maximum size. The maximum size can be set by the network gateway, the mail server, or the mobile device. While normally fixed, the maximum size could also be dynamically changed. In one exemplary embodiment, 400 bytes has been used as a maximum size. Then, once the electronic mail resources are sent **1308** to the mobile device they do not consume a large portion of the memory (namely, persistent storage) because they are of limited size. By limiting the electronic mail resources, the memory of the mobile device is able to store more electronic mail messages (which are of the limited size). Hence, the memory of the mobile device (which itself is of limited capacity) is intelligently used.

Typically, the limiting operates to limit the size of the message bodies. Thus, when a user views a message body only the first portion of the message body is initially displayable to the user. When there is additional portions of the message body, the user can be informed via the display screen that additional portions can be requested. If the user requests a next portion, then a synchronous request for the next portion is sent to the mail server and the next portion of the message body is thereafter received and displayed. Again, the next portion can also be of limited size. Additionally, a percentage of the portion of the total message that has been read can be displayed on the display screen for the user. The percentage would increase with each next portion read until all (100%) of the message is read.

FIG. **14** is a block diagram of a mobile device **1400** according to another embodiment of the invention. Like the mobile device **200** illustrated in FIG. **2**, the mobile device **1400** is particularly suited for wireless communications through a wireless network where connectivity is sporadic or high-latency conditions are present.

The mobile device **1400** includes an application **1402** that couples to a cache memory **1404** through an interface **1406**. The application **1402** is, for example, a network browser application that allows a user of the mobile device **1400** to request and receive resources provided on a remote network (e.g., Internet) that the mobile device **1400** is able to communicate with. The cache memory **1404** stores resources that have been previously requested and received by the mobile device **1400**. Additionally, the cache memory **1404** can be used to store various channels and lists that are used to improve performance of the mobile device **1400**.

The mobile device **1400** also includes a synchronous request manager **1408** and an asynchronous message manager **1410**. The synchronous request manager **1408** manages the synchronous sending and receiving of messages with respect to the remote network a wireless communication link **1409**. The asynchronous message manager **1410** manages the asynchronous sending of messages with respect to the network via a wireless communication link **1411**. The asynchronous message manager **1410** is provided so that the mobile device **1400** can communicate with the remote network in an asynchronous manner. The ability of the mobile device **1400** to communicate in an asynchronous manner is particularly useful in cases where the wireless network has high-latency conditions or suffers from sporadic connectivity.

The asynchronous message manager **1410** connects to an asynchronous message queue **1412** that stores messages that are to be sent to the remote network via the asynchronous message manager **1410** and the wireless communication link **1411**. In general, the messages awaiting transmission to the remote network remain in the asynchronous message queue **1412** until the asynchronous message manager **1410** determines that the wireless network is available, and then proceeds to service the particular messages stored in the asynchronous message queue **1412** such that they are sent to the remote network through the wireless communication link **1411** of the wireless network when the wireless network is available.

The push manager **1414** receives pushed messages (e.g., resources) from the remote network over a wireless communication link **1415**. These pushed messages that are received by the push manager **1414** over the wireless communication link **1415** are either in response to requests that have been asynchronously sent by the mobile device **1400** or provided ("pushed") by a remote server to the mobile device **1400** connects via the wireless network.

The mobile device **1400** also includes a channel manager **1416** and a list manager **1418**. The channel manager **1416** operates to load certain content channels into the cache memory **1404** of the mobile device **1400**. According to the invention, one pertinent content channel is an electronic mail channel, which includes electronic mail resources associated with providing electronic mail services the mobile device **1400**. The content channels are loaded into the cache memory **1404** by the channel manager **1416** so that the performance of the mobile device **1400** is improved with respect to the particular content associated with the content channels loaded in the cache memory **1404**. In particular, should a user of the mobile device **1400** request (via the application **1402**) a resource associated with a content channel stored in the cache memory **1404**, then the requested resource can be rapidly supplied to application **1402** via the cache memory **1404**. Otherwise, when a requested resource is not found in the cache memory **1404**, the availability of the requested resource at the mobile device is relatively slow because the request must be sent to either the synchronous request manager **1408** or the asynchronous message manager **1410** to obtain the resource from a remote server located on the remote network.

The list manager **1418** operates to store one or more lists in a memory **1420** (e.g., random-access memory). The lists that are stored in the memory **1420** are used to enable the list manager **1418** to manipulate various lists that are used by user interfaces associated with the application **1402**. The user interfaces operate to display the various lists on a display **1422**. Such lists are thus able to be modified locally within the mobile device **1400** without the need for availability of the wireless network. According to the invention, one pertinent list that is managed by the list manager **1418** is an electronic mail list that is displayed in the display **1422** and able to be locally modified without regard to network availability. The memory **1420** can also store resources from the remote network that are used to configure or operate the mobile device **1400**.

The advantages of the invention are numerous. Several advantages that embodiments of the invention may include are as follows. One advantage of the invention is that electronic mail services can be performed on electronic mail messages even when the network is unavailable. Clients, e.g., mobile devices, are able to perform electronic mail services regardless of network availability. As a result, clients experience better responsiveness and less waiting. Another advantage of the invention is that a mail server located on the network is able to be kept current by use of asynchronous messaging.

The many features and advantages of the present invention are apparent from the written description, and thus, it is intended by the appended claims to cover all such features and advantages of the invention. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation as illustrated and described. Hence, all suitable modifications and equivalents may be resorted to as falling within the scope of the invention.

What is claimed is:

1. A mobile device for use with a wireless data communication network, said mobile device comprising:

a memory storage device that stores electronic mail resources;

an electronic mail processor that performs an electronic mail operation with respect to the electronic mail resources stored in said memory storage device, and wherein the electronic mail operation can be carried out at said mobile device even when the wireless data communication network is not available to said mobile device; and

a display device that displays at least a part of one or more of the electronic mail resources.

2. A mobile device as recited in claim **1**, wherein the storage of the electronic mail resources in said memory storage device is persistent and thus not subjected to cache removal processing.

3. A mobile device as recited in claim **1**, wherein in performing the electronic mail operation when the wireless data communication network is not available, said electronic mail processor modifies the electronic mail resources stored in said memory storage device in accordance with the electronic mail operation being performed.

4. A mobile device as recited in claim **3**, wherein a mail server is coupled to the wireless data communication network,

wherein the electronic mail resources are pre-stored in said memory storage device,

wherein the wireless data communication network is occasionally unavailable to said mobile device, and

wherein said mobile device further comprises:

an asynchronous message manager that sends an asynchronous message to the mail server when the wireless data communication network is available to said mobile device, the asynchronous message informing the mail server of the modification to the electronic mail resources that occurred while the wireless data communication network was not available.

5. A mobile device as recited in claim **4**, wherein said mobile device further comprises:

an asynchronous message queue operatively connected to said asynchronous message manager, said asynchronous request queue stores outgoing asynchronous messages to be sent from said mobile device to the mail server.

6. A mobile device as recited in claim **1**, wherein a remote server is coupled to the wireless data communication network when the wireless data communication network is not available, and

wherein in performing the electronic mail operation, said electronic mail processor causes an asynchronous message to be created and thereafter sent to the remote server when the wireless data communication network is available to said mobile device, the asynchronous message being based on the electronic mail operation being performed.

7. A mobile device as recited in claim **6**, wherein the electronic mail operation is one of delete message, view message, reply message or new message.

8. A mobile device as recited in claim **1**, wherein the electronic mail operation is one of delete message, view message, reply message or new message.

9. A mobile device as recited in claim **1**, wherein said mobile device is one of a mobile telephone, a mobile pager, a mobile personal digital assistant, and a mobile computer.

10. A mobile device as recited in claim **1**, wherein said mobile device is a mobile telephone.

11. A mobile device as recited in claim **1**, wherein said memory storage device has a limited capacity, and wherein the size of the electronic mail resources being stored in said memory storage device are limited to be less than a predetermined maximum size.

12. A mobile device as recited in claim **1**, wherein a mail server is coupled to the wireless data communication network,

wherein the electronic mail resources are pre-stored in said memory storage device,

wherein the wireless data communication network is occasionally unavailable to said mobile device, and

wherein said electronic mail processor comprises a network browser that operates to retrieve the electronic mail resources from said memory storage device or the mail server.

13. A mobile device as recited in claim **12**, wherein said mobile device is a mobile telephone, and wherein the network browser is a micro-browser.

14. A method for interacting with electronic mail messages on a mobile device, the mobile device being able to communicate with a mail server at least in part through a wireless data network, said method comprising:

pre-loading electronic mail message resources into a storage device of the mobile device;

receiving a request to view an electronic mail list;

determining whether the electronic mail list is available from the storage device of the mobile device;

receiving the electronic mail list from the storage device when the electronic mail list is determined to be available from the storage device of the mobile device;

requesting and subsequently receiving the electronic mail list from the mail server when the electronic mail list is determined not to be available from the storage device of the mobile device; and

displaying the received electronic mail list.

15. A method as recited in claim **14**, wherein the availability of the wireless data network to the mobile device is sporadic, and

wherein when the electronic mail list is available from the storage device, the received electronic mail list can be displayed regardless of the availability of the wireless data network to the mobile device.

16. A method as recited in claim **14**,

wherein the electronic mail list identifies at least one electronic mail message, and

wherein the electronic mail resources include at least the electronic mail list, and a message body for the at least one electronic mail message.

17. A method as recited in claim **16**, wherein the electronic mail list includes a sender identifier and a subject for at least one electronic mail message.

18. A method as recited in claim **14**,

wherein the electronic mail list identifies a plurality of electronic mail messages, and

wherein said method further comprises:

performing an operation on one of the electronic mail messages in the electronic mail list; and

modifying the electronic mail resources stored in the storage device in accordance with the operation performed on the one of the electronic mail messages.

19. A method as recited in claim **18**, wherein the operation is one of delete, view or reply.

20. A method as recited in claim **18**, wherein the availability of the wireless data network to the mobile device is sporadic,

wherein said method further comprises updating the electronic mail list being displayed after said modifying, and

wherein when the electronic mail list is available from the storage device, the received electronic mail list can be displayed and modified regardless of the availability of the wireless data network to the mobile device and thus without waiting for the availability of the wireless data network.

21. A method as recited in claim **18**, wherein said method further comprises:

forming an asynchronous request to the mail server to inform the mail server of the modification to the one of the electronic mail messages.

22. A method as recited in claim **18**, wherein said method further comprises:

forming an asynchronous request to the mail server to inform the mail server of the modification to the one of the electronic mail messages; and

thereafter sending the asynchronous request to the mail server when the wireless data network becomes available.

23. A method as recited in claim **17**, wherein said method further comprises:

subsequently updating the electronic mail list being displayed.

24. A method as recited in claim **14**, wherein the electronic mail message resources being pre-loaded into the storage device of the mobile device are previously limited to a predetermined maximum size.

25. A method as recited in claim **14**, wherein the mobile device communicates with the mail server using a network browser.

26. A method as recited in claim **25**, wherein the network browser is a micro-browser.

27. A method as recoted in claim **26**, wherein said mobile device is a mobile telephone.

28. A method for interacting with electronic mail messages on a mobile device, the mobile device being able to connect to a remote mail server through a wireless data network, said method being performed by said mobile device and comprising:

displaying an electronic mail list on a display screen of the mobile device, the electronic mail list including one or more entries that identify particular electronic mail messages;

selecting one of the entries of the electronic mail list being displayed on the display screen of the mobile device;

performing an operation on the electronic mail message associated with the selected entry without delay due to the unavailability of the wireless data network to the mobile device; and

asynchronously sending a notification to the remote mail server based on the operation previously performed on the electronic mail message associated with the selected entry when the wireless data network later becomes available to the mobile device.

29. A method as recited in claim **28**, wherein when the operation performed on the electronic mail message requires the electronic mail list to be updated, the electronic mail list is re-displayed in its modified form without regard to whether the wireless data network is available to the mobile device.

30. A method as recited in claim **28**, wherein said asynchronously sending of the notification message is deferred until the wireless data network is available to the mobile device.

31. A method as recited in claim **28**,

wherein the operation is a view operation,

wherein a message body for the electronic mail message is stored in a storage device within the mobile device, and

wherein said performing of the view operation on the electronic mail message operates to retrieve the message body from the storage device and then to display the message body on the display screen, without delay due to the unavailability of the wireless data network to the mobile device.

32. A method as recited in claim **28**,

wherein the operation is a delete operation,

wherein data for the electronic mail message is stored in a storage device within the mobile device, and

wherein said performing of the delete operation on the electronic mail message operates to delete the data of the electronic mail message associated with the selected entry from the storage device and then to re-display the electronic mail list on the display screen of the mobile device such that the selected entry is no longer present.

33. A method as recited in claim **32**, wherein the notification asynchronously sent is a delete message request that is sent to the mail server via the wireless data network, and the delete message request is sent to the mail server in a background mode so that unavailability of the wireless data network to the mobile device does not delay the delete operation or subsequent operations from completing on the mobile device.

34. A method as recited in claim 28,

wherein the operation is a facsimile operation,

wherein said performing of the facsimile operation on the electronic mail message operates to display a facsimile options screen and to receive destination information for a facsimile of the electronic mail message, and

wherein the notification asynchronously sent is a facsimile request that is sent to a remote server or the mail server via the wireless data network, and the facsimile request message is sent to the remote mail server in a background mode so that unavailability of the wireless data network to the mobile device does not delay the facsimile operation or subsequent operations from completing on the mobile device.

35. A method as recited in claim 28,

wherein the operation is a reply operation,

wherein said performing of the reply operation on the electronic mail message operates to display a reply screen and to receive reply message information for a reply message to the electronic mail message, and

wherein the notification asynchronously sent is a reply message request that is sent to the mail server via the wireless data network, and the reply message request is sent to the mail server in a background mode so that unavailability of the wireless data network to the mobile device does not delay the reply operation or subsequent operations from completing on the mobile device.

36. A method as recited in claim 28,

wherein the operation is a new message operation,

wherein said performing of the new message operation on the electronic mail message operates to display a message data entry screen and to receive new message information for a new electronic mail message, and

wherein the notification asynchronously sent is a new message request that is sent to the mail server via the wireless data network, and the new message request is sent to the mail server in a background mode so that unavailability of the wireless data network to the mobile device does not delay the new message operation or subsequent operations from completing on the mobile device.

37. A computer readable medium including computer program code for interacting with electronic mail messages on a computing device, the computing device being able to communicate with a mail server at least in part through a data network, said computer readable medium comprising:

computer program code configured to pre-load electronic mail message resources into a storage device of the computing device;

computer program code configured to receive a request to view an electronic mail list;

computer program code configured to determine whether the electronic mail list is available from the storage device of the computing device;

computer program code configured to receive the electronic mail list from the storage device when the electronic mail list is determined to be available from the storage device of the computing device;

computer program code configured to request and subsequently receive the electronic mail list from the mail server when the electronic mail list is determined not to be available from the storage device of the computing device; and

computer program code configured to display the received electronic mail list.

38. A computer readable medium for interacting with electronic mail messages on a computing device, the computing device being able to connect to a remote mail server through a data network, said computer readable medium comprising:

computer program code configured to display an electronic mail list on a display screen of the computing device, the electronic mail list including one or more entries that identify particular electronic mail messages;

computer program code configured to select one of the entries of the electronic mail list being displayed on the display screen of the computing device;

computer program code configured to perform an operation on the electronic mail message associated with the selected entry without delay due to the unavailability of the wireless data network to the computing device; and

computer program code configured to asynchronously send a notification to the remote mail server based on the operation performed on the electronic mail message associated with the selected entry when the data network is available to the computing device.

* * * * *

EXHIBIT C

U.S. PATENT NO. 6,405,037

US006405037B1

(12) **United States Patent**
Rossmann

(10) **Patent No.: US 6,405,037 B1**
(45) **Date of Patent: Jun. 11, 2002**

(54) **METHOD AND ARCHITECTURE FOR AN INTERACTIVE TWO-WAY DATA COMMUNICATION NETWORK**

(75) Inventor: **Alain Rossmann**, Menlo Park, CA (US)

(73) Assignee: **Openwave Systems Inc.**, Redwood City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/199,121**

(22) Filed: **Nov. 24, 1998**

Related U.S. Application Data

(63) Continuation of application No. 08/978,701, filed on Nov. 26, 1997, and a continuation of application No. 08/570,210, filed on Dec. 11, 1995, now Pat. No. 5,809,415.

(51) **Int. Cl.**[7] **H04Q 7/20**

(52) **U.S. Cl.** **455/426**; 455/422; 455/550; 455/552; 455/424; 455/412

(58) **Field of Search** 455/403, 411, 455/412, 422, 426, 445, 528, 550, 517, 413, 414, 552, 466, 418, 419, 557, 556, 566, 424

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,787,028 A	11/1988	Finfrock et al.	
4,812,843 A	3/1989	Champion, III et al.	
5,008,925 A	4/1991	Pireh	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 646 856 A2	4/1995	
EP	0 646 856 A3	12/1996	
EP	0812120 A3	12/1997	
EP	0812120 A2	12/1997	
FI	100137	9/1997	..... G07F/19/00
FI	102020	9/1998	..... G07F/19/00
JP	5-233191	9/1993	
JP	6-110637	4/1994	
JP	6-175764	6/1994	
JP	7-13671	1/1995	
JP	7-263187	10/1995	
JP	59-41047	3/1997	
JP	5-35421	2/1999	
WO	WO 93/16550	8/1993	..... H04M/11/00
WO	WO 96/13814	5/1996	..... G07F/7/08
WO	WO-97-14244	4/1997	
WO	WO 97/41654	11/1997	

OTHER PUBLICATIONS

HDTP Specification, Version 1.1–Draft, pp. 1–40, Redwood Shores, CA, Unwired Planet, Inc., Jul. 15, 1997.

(List continued on next page.)

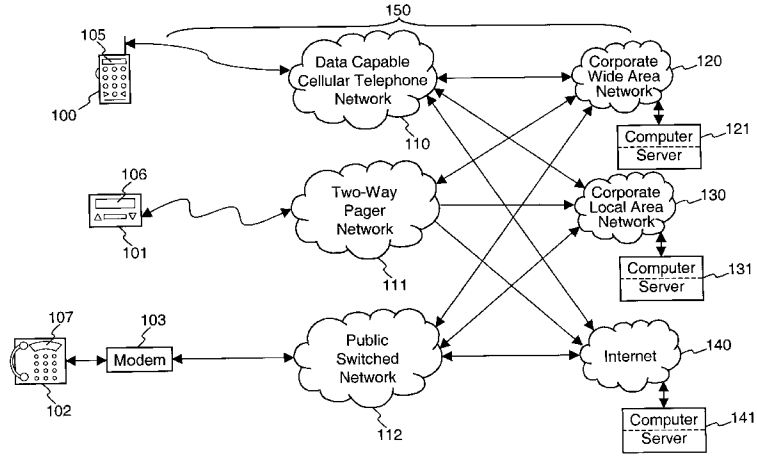
Primary Examiner—William Trost
Assistant Examiner—Keith Ferguson
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

A two-way data communication device such as a data ready cellular telephone, a two-way pager, or a telephone communicates via a two-way data communication network with a server computer on a computer network that has an interface to the two-way data communication network, i.e, is coupled to the two-way data communication network. For example, the computer network can be a corporate wide area network, a corporate local area network, the Internet, or any combination of computer networks. The two-way data communication device utilizes a client module to transmit message including a resource selector chosen by the user to a server on a server computer on the computer network. The server processes the message and transmits a response over the two-way data communication network to the client module. The client module interprets the response and presents the response to the user via a structured user interface. Alternatively, the user transmits a request that directs the server to transmit the response to the request to another location or to another user.

28 Claims, 36 Drawing Sheets

Microfiche Appendix Included
(6 Microfiche, 369 Pages)



U.S. PATENT DOCUMENTS

5,128,672	A		7/1992	Kaehler	341/23
5,335,276	A		8/1994	Thompson et al.	
5,465,401	A		11/1995	Thompson	
5,491,605	A		2/1996	Roeder	455/422
5,491,745	A		2/1996	Roeder	
5,548,636	A		8/1996	Bannister et al.	
5,555,446	A		9/1996	Jasinski	
5,560,008	A		9/1996	Johnson et al.	
5,577,100	A		11/1996	McGregor et al.	
5,577,103	A		11/1996	Foti	
5,577,209	A		11/1996	Boyle et al.	
5,579,535	A		11/1996	Orlen et al.	
5,581,595	A		12/1996	Iwashita et al.	455/566
5,606,786	A		3/1997	Gordon	379/100
5,608,786	A		3/1997	Gordon	
5,623,605	A		4/1997	Keshav et al.	
5,625,605	A		4/1997	Keshav et al.	370/392
5,671,354	A		9/1997	Ito et al.	
5,675,507	A		10/1997	Bobo, II	364/514 R
5,708,828	A		1/1998	Coleman	
5,727,159	A		3/1998	Kikinis	
5,740,252	A	*	4/1998	Minor et al.	455/422
5,742,668	A		4/1998	Pepe et al.	
5,745,706	A		4/1998	Wolfberg et al.	
5,751,798	A		5/1998	Mumick et al.	
5,854,936	A		12/1998	Pickett	
5,867,153	A		2/1999	Grandcolas et al.	
5,884,284	A		3/1999	Peters et al.	
5,909,485	A		6/1999	Martin et al.	
5,918,013	A		6/1999	Mighdoll et al.	
5,940,589	A		8/1999	Donovan et al.	
5,958,006	A		9/1999	Eggleston et al.	
6,049,711	A		4/2000	Ben-Yehezkel et al.	
6,085,105	A		7/2000	Becher	
6,119,137	A		9/2000	Smith et al.	
6,122,403	A		9/2000	Rhoads	
6,157,823	A		12/2000	Fougnies et al.	
6,167,253	A	*	12/2000	Farris et al.	455/412
6,185,184	B1		2/2001	Mattaway et al.	

OTHER PUBLICATIONS

HDML 2.0 Language Reference, Version 2.0, pp. 1–56, Redwood Shores, CA, Unwired Planet, Inc., Jul. 1997.

M. Liljeberg et al.: "Optimizing World–Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach" International Workshop on Sercvices in Distributed and Networked Environments, Jun. 5, 1995. pp. 132–139, XP000764774.

M. F. Kaashoek et al.: "Dynamic Documents: Mobile Wireless Access to the WWW" Proceedings, Workshop on Mobile Computing Systems and Applications, Dec. 8, 1994, pp. 179–184, XP0002016896.

S. Gessler et al.: "PDAs as mobile WWW browsers"Computer Networks and ISDN Systems, vol. 28, No. 1, Dec. 1, 1995, pp. 53–59, XP004001210.

G. M. Voelker et al.: "Mobisaic: An Information System for a Mobile Wireless Computing Environment" Proceedings, Workshop on Mobile computing Systems and Application, vol. 5, No.10, Jan. 1, 1995, pp. 185–190, XP002062595.

Bekre et al.; Institute of Electrical and Electronics Engineers, Inc.; "Distributed Computing Systems"; May 30—Jun. 2, 1995; pp. 136–143; *IEEE Computer Society*.

Japanese Publication; "Netscape Navigator for Windows"; Chapters 1–3; Jul. 31, 1995.

Japanese Publication; "Nifty Serve"; p. 61; Oct. 5, 1992.

Japanese Article; "Mosaic"; *Unix Magazine*; pp. 36–44; Mar. 1994.

Japanese Publication; "Introduction to HTML–WWW Publishing"; pp. 9,14,15, 20, 32–34, 163, 269, 286, 287 and 270; Jun. 30, 1995.

Japanese Publication; Larry Aronson; "Introduction to HTML"; Jul. 1, 1995.

Nikkei Multimedia Magazine; pp. 109–111; Nov. 1995.

"Introduction to HTML (3)"; *Unix Magazine*; Dec. 1994; pp. 49–52.

"NCSA HTTP Defined and Introduction to HTML"; *Unix Magazine*; Oct. 1994; pp. 52–60.

ASCII Magazine, vol. 19, No.7; Jul. 1995; pp. 331–353.

Mac Power Magazine; Jun. 1995; pp. 105 and 268.

Patent application EP 95935472.1–2207.

Behruz Vazvan: "Real Time Tele–payment System" Distributed in an EU's organized international seminar "Race Mobile Summit," Cascais–Portugal, 22–24.11.1995.

* cited by examiner

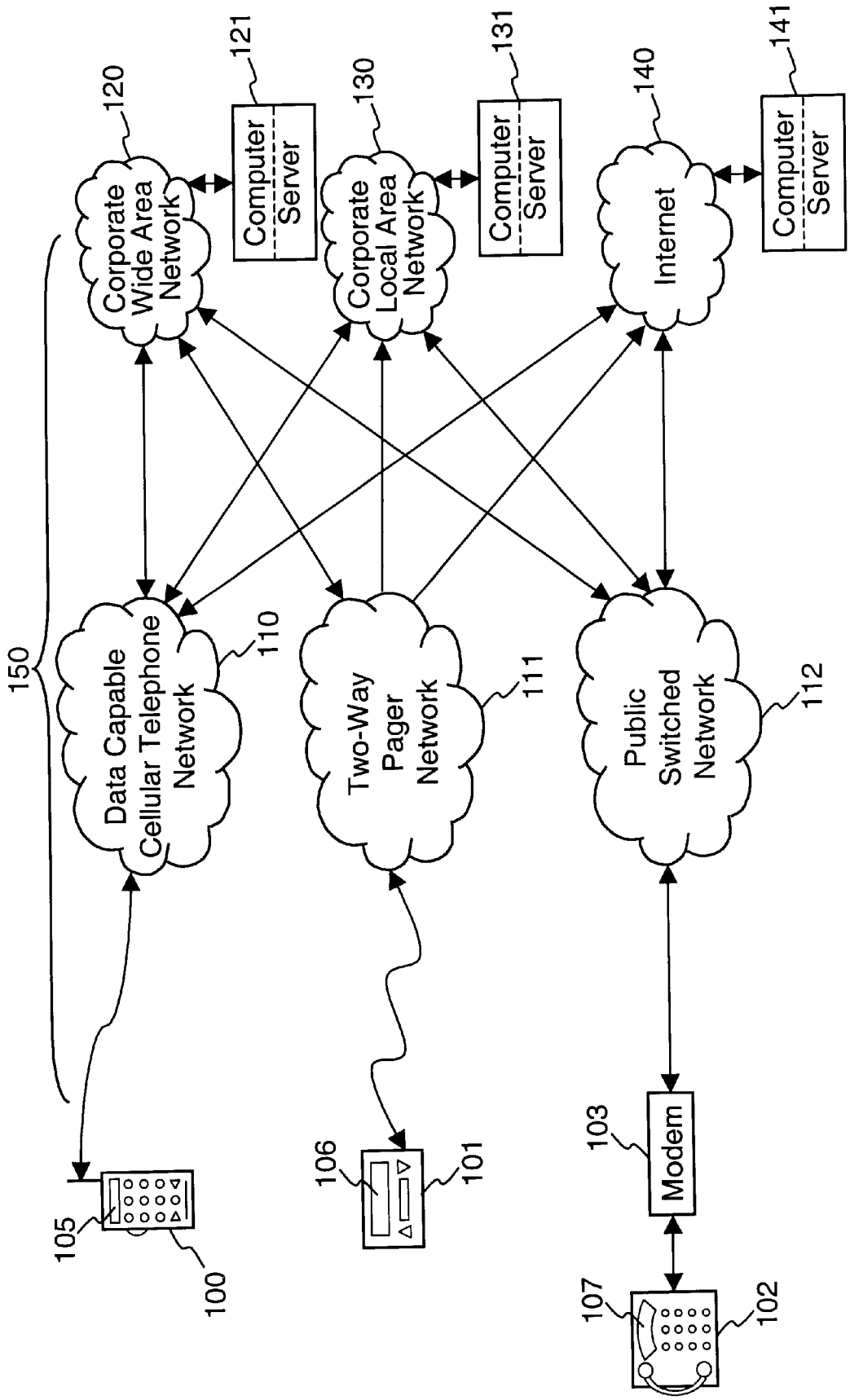

Corporate Wide Area Network
120
Computer Server
121
Corporate Local Area Network
130
Computer Server
131
Internet
140
Computer Server
141
Data Capable Cellular Telephone Network
110
Two-Way Pager Network
111
Public Switched Network
112
150
105
100
106
101
103
Modem
107
102

FIG. 1

FIG. 2A

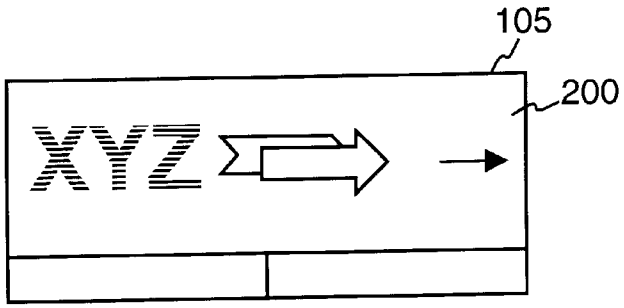


FIG. 2B

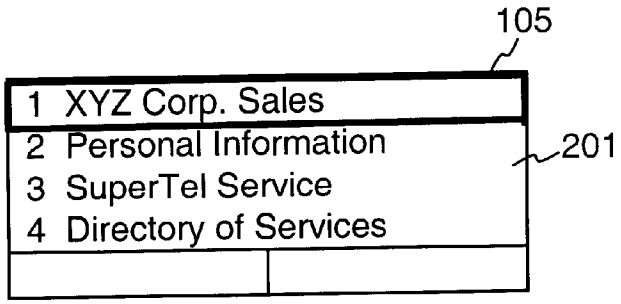


FIG. 2C

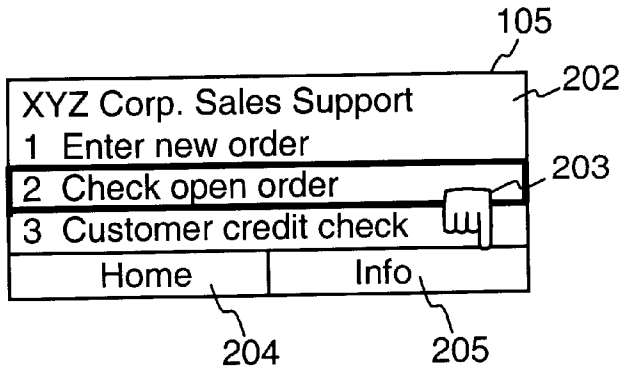


FIG. 2D

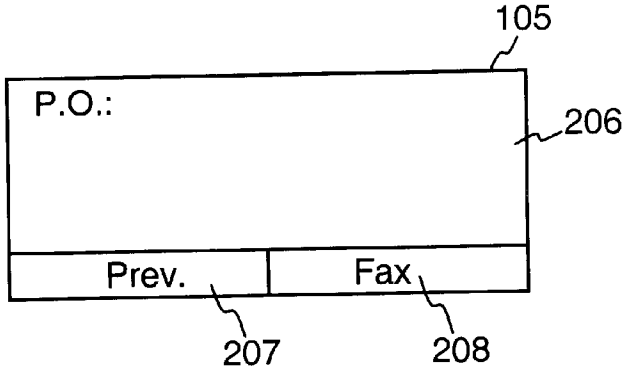


FIG. 2E

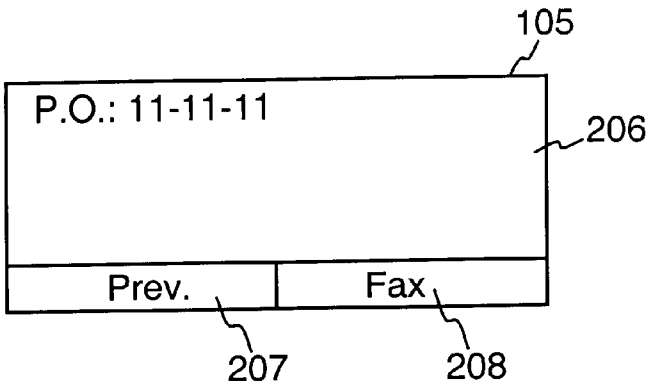


FIG. 2F

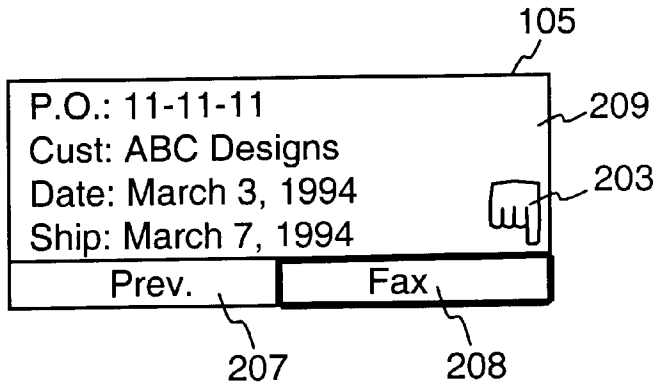


FIG. 2G

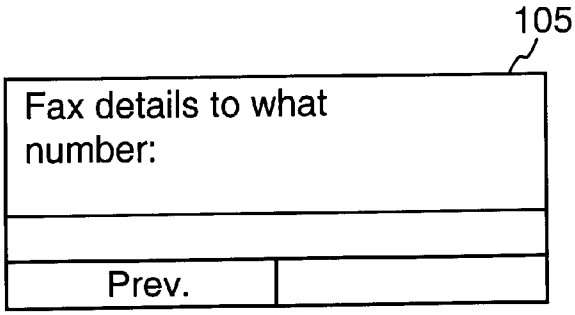
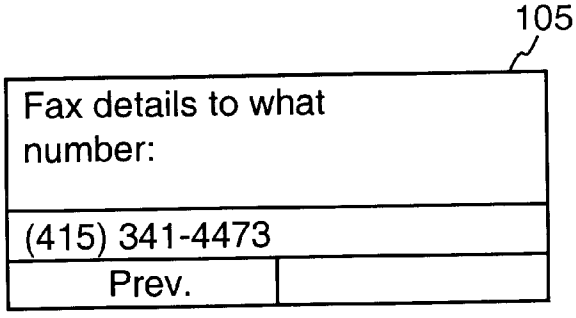


FIG. 2H



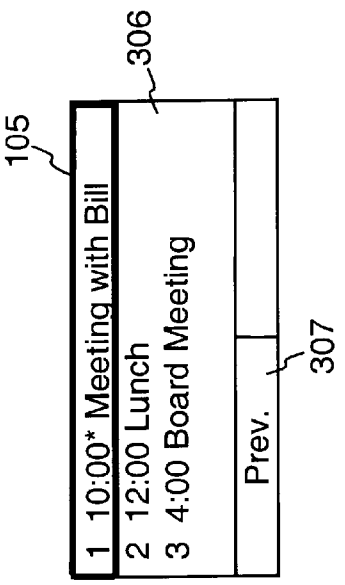


FIG. 3D

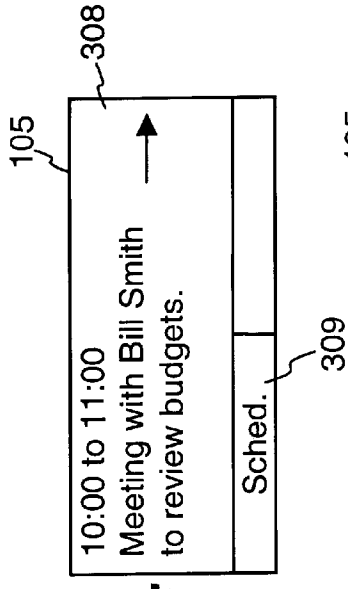


FIG. 3E

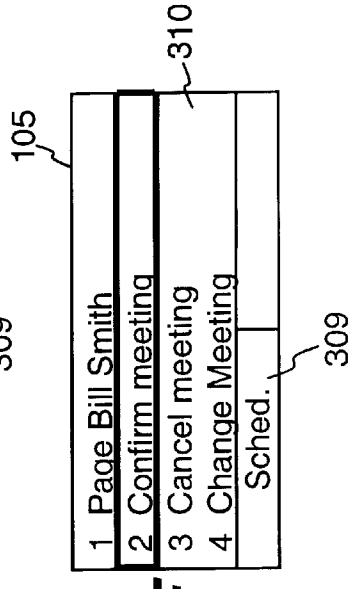


FIG. 3F

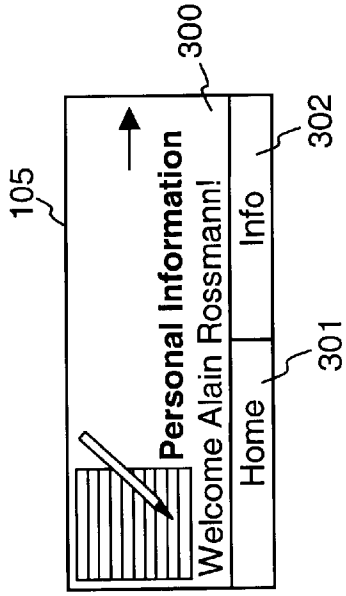


FIG. 3A

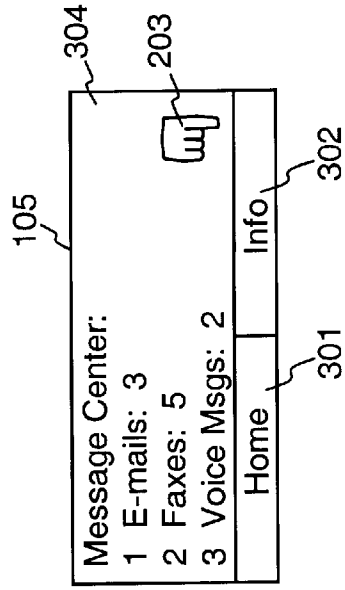


FIG. 3B

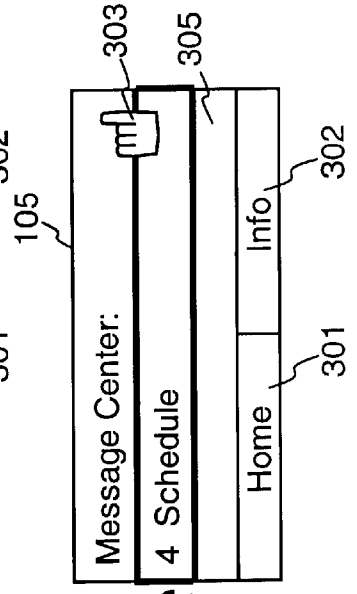
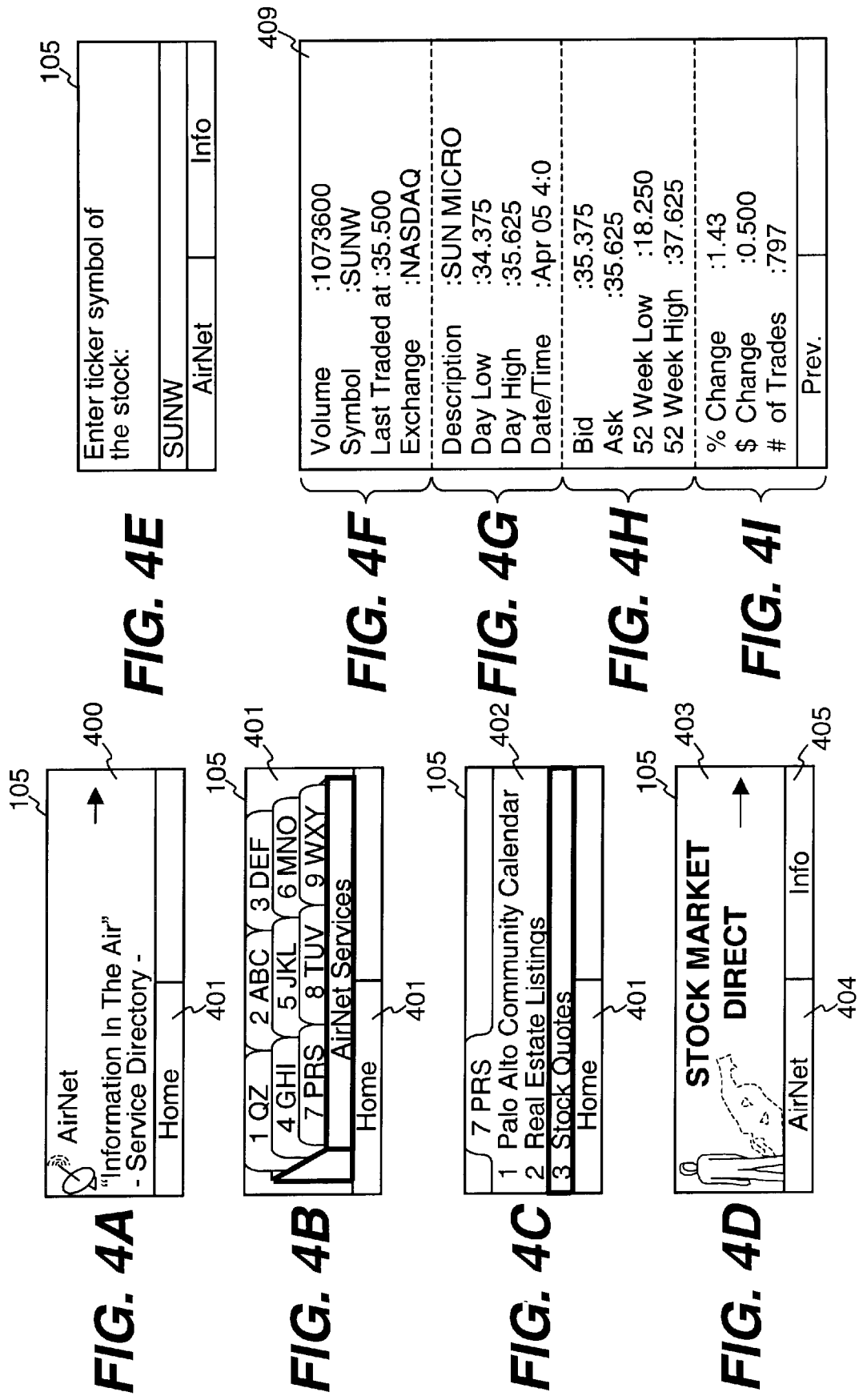


FIG. 3C


105
AirNet
"Information In The Air"
- Service Directory -
Home
400
401
FIG. 4A
105
1 QZ
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PRS
8 TUV
9 WXY
AirNet Services
Home
401
FIG. 4B
105
7 PRS
1 Palo Alto Community Calendar
2 Real Estate Listings
3 Stock Quotes
Home
401
402
FIG. 4C
105
STOCK MARKET
DIRECT
AirNet
Info
403
404
405
FIG. 4D
105
Enter ticker symbol of the stock:
SUNW
AirNet
Info
FIG. 4E
409
Volume :1073600
Symbol :SUNW
Last Traded at :35.500
Exchange :NASDAQ
FIG. 4F
Description :SUN MICRO
Day Low :34.375
Day High :35.625
Date/Time :Apr 05 4:0
FIG. 4G
Bid :35.375
Ask :35.625
52 Week Low :18.250
52 Week High :37.625
FIG. 4H
% Change :1.43
$ Change :0.500
of Trades :797
Prev.
FIG. 4I

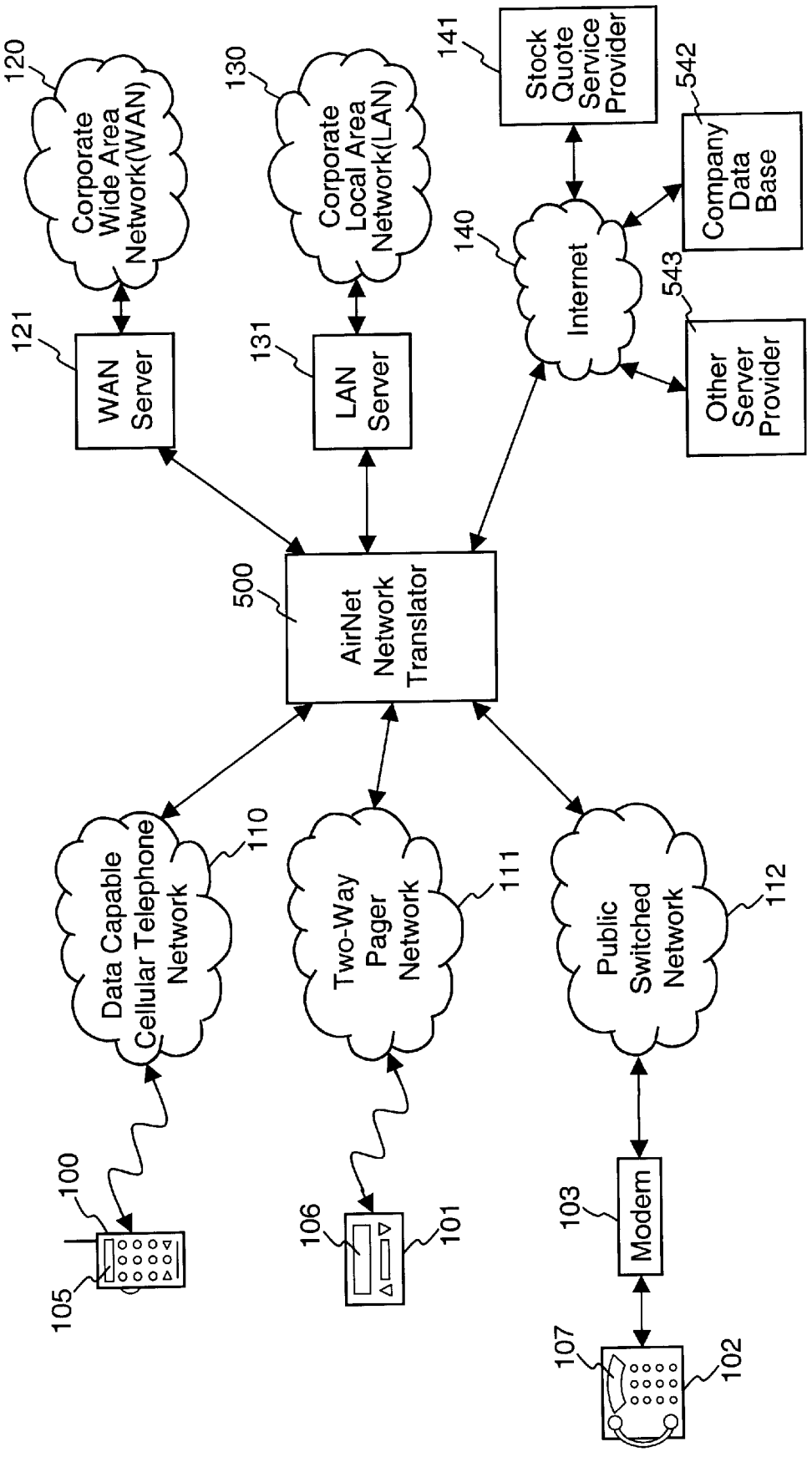

120
Corporate Wide Area Network(WAN)
130
Corporate Local Area Network(LAN)
141
Stock Quote Service Provider
542
Company Data Base
140
Internet
543
Other Server Provider
121
WAN Server
131
LAN Server
500
AirNet Network Translator
110
Data Capable Cellular Telephone Network
111
Two-Way Pager Network
112
Public Switched Network
100
105
106
101
103
Modem
107
102

FIG. 5

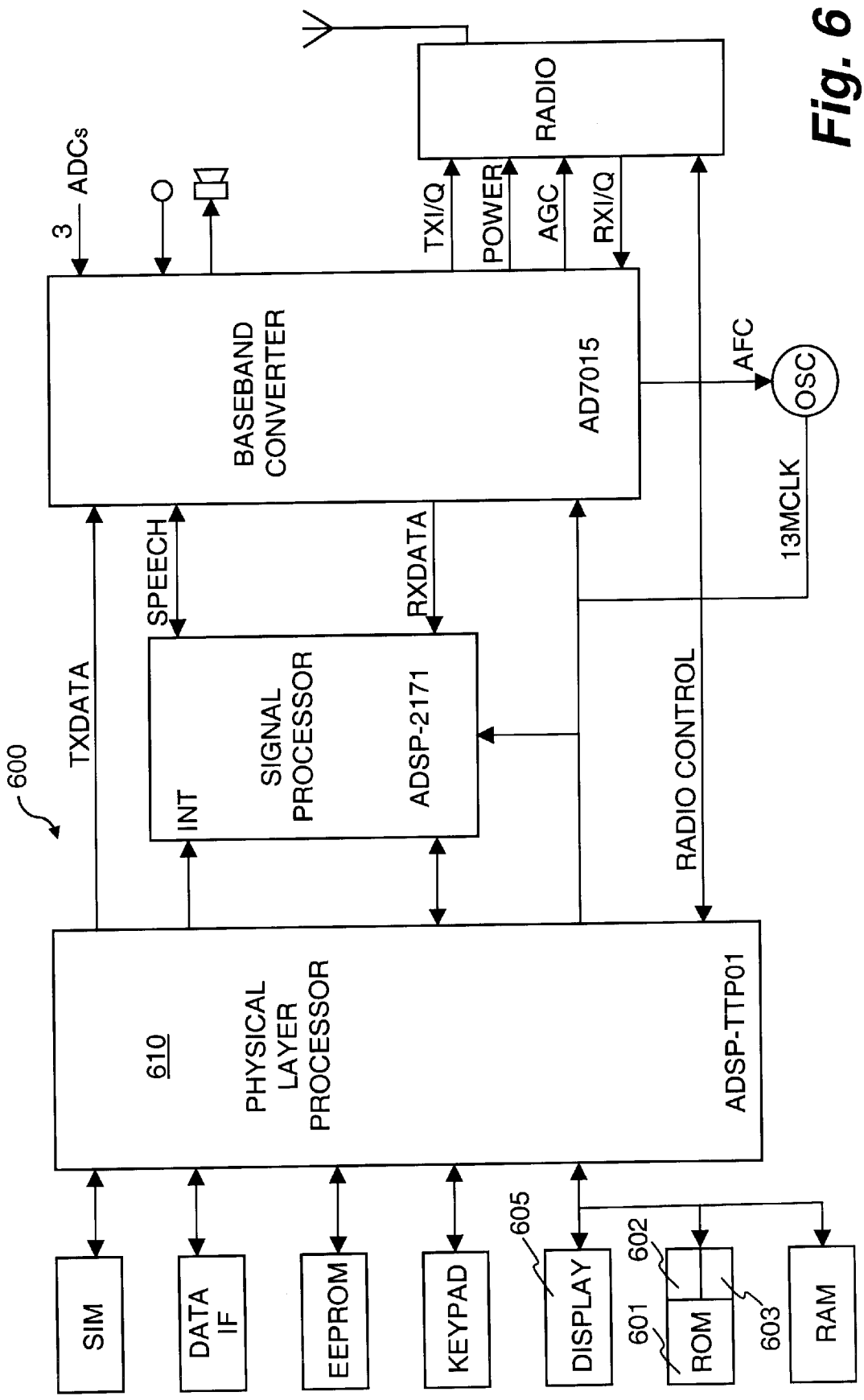

600
TXDATA
SPEECH
RXDATA
INT
SIGNAL PROCESSOR
ADSP-2171
610
PHYSICAL LAYER PROCESSOR
ADSP-TTP01
RADIO CONTROL
3 ADCs
BASEBAND CONVERTER
AD7015
TXI/Q
POWER
AGC
RXI/Q
RADIO
AFC
OSC
13MCLK
SIM
DATA IF
EEPROM
KEYPAD
605
DISPLAY
602
601
ROM
603
RAM

Fig. 6

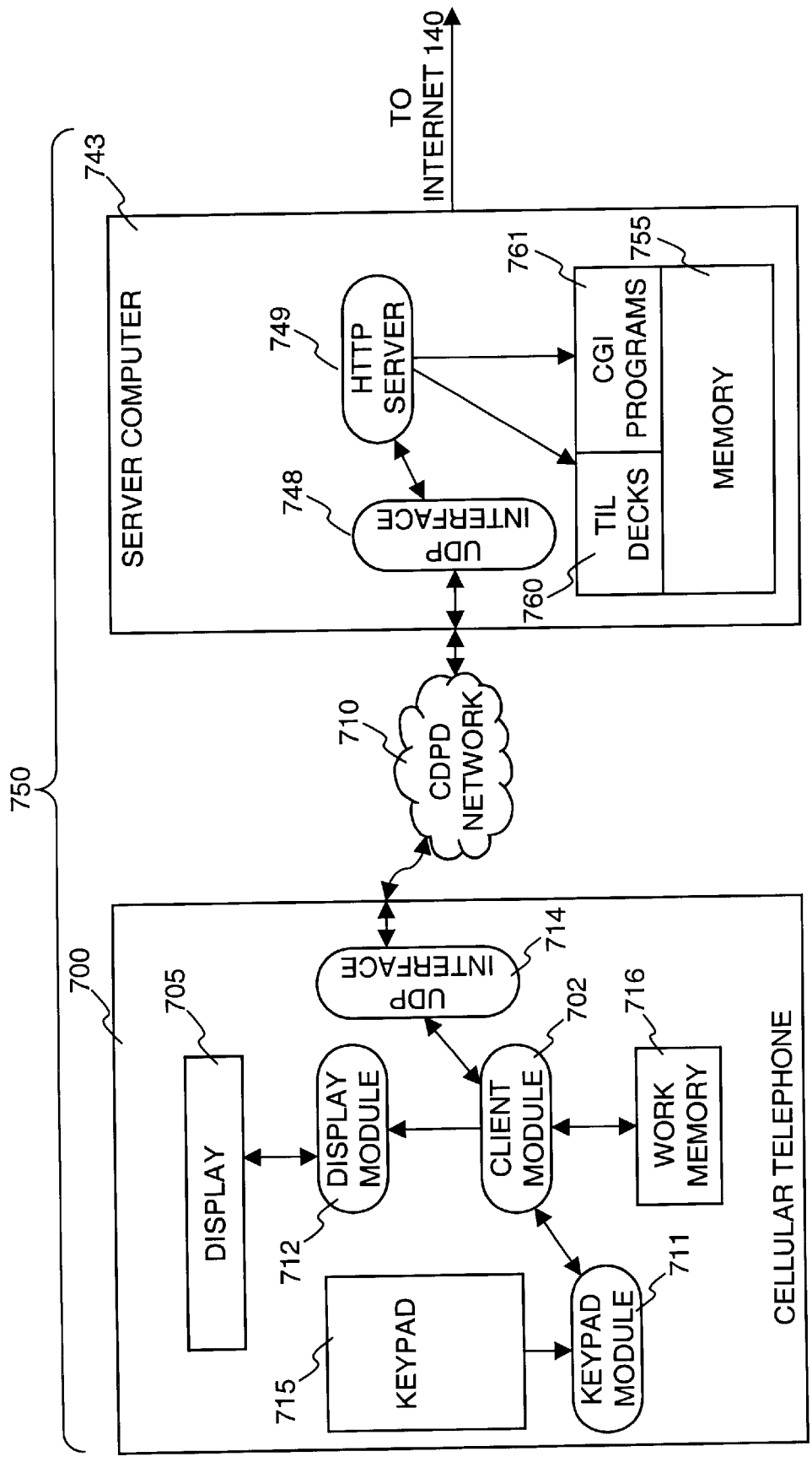

750
700
CELLULAR TELEPHONE
DISPLAY
705
DISPLAY MODULE
712
KEYPAD
715
KEYPAD MODULE
711
CLIENT MODULE
702
UDP INTERFACE
714
WORK MEMORY
716
CDPD NETWORK
710
743
SERVER COMPUTER
HTTP SERVER
749
UDP INTERFACE
748
CGI PROGRAMS
761
TIL DECKS
760
MEMORY
755
TO INTERNET 140

FIG. 7

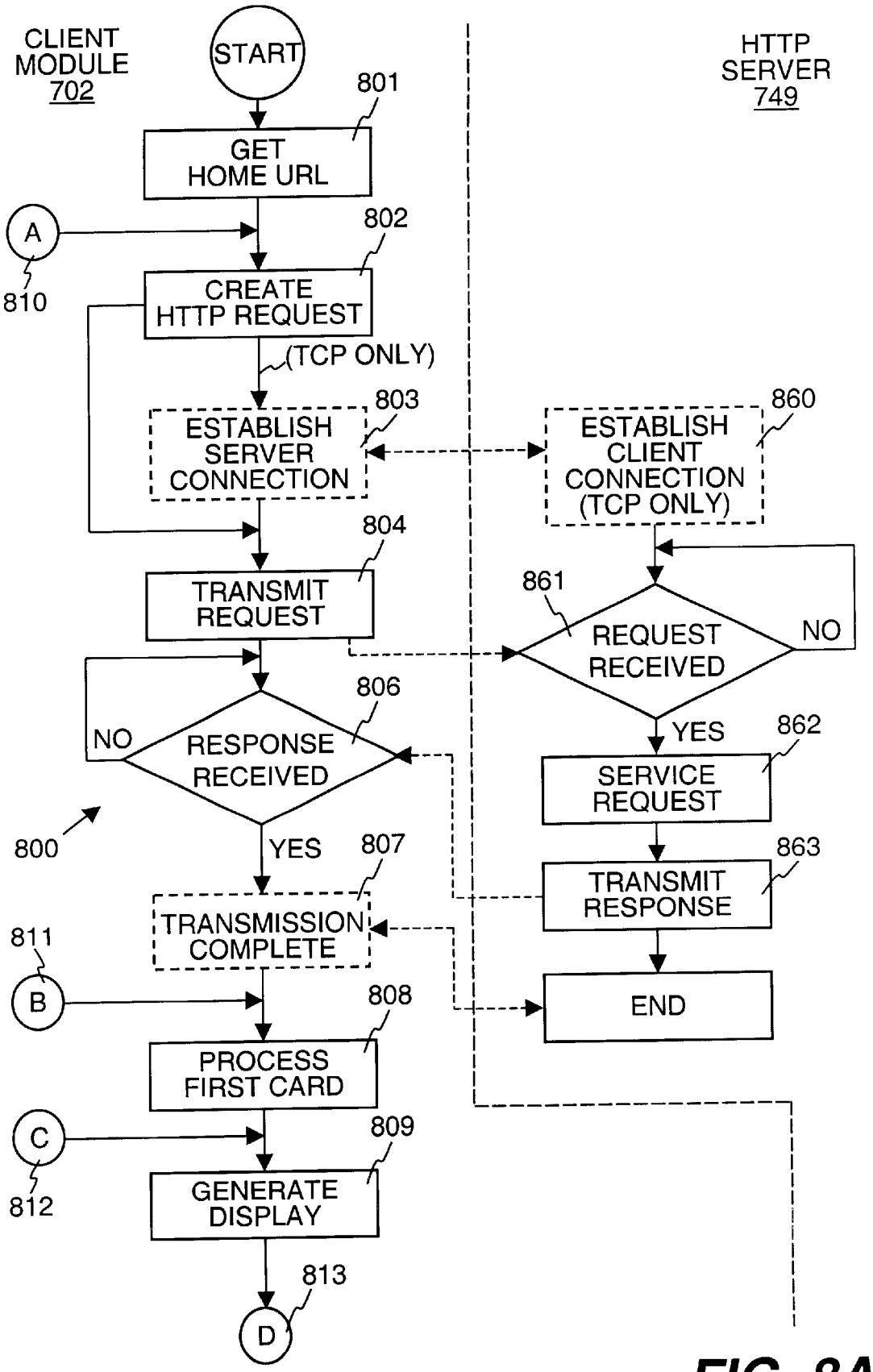

CLIENT MODULE 702
HTTP SERVER 749
START
801
GET HOME URL
A
810
802
CREATE HTTP REQUEST
(TCP ONLY)
803
ESTABLISH SERVER CONNECTION
860
ESTABLISH CLIENT CONNECTION (TCP ONLY)
804
TRANSMIT REQUEST
861
REQUEST RECEIVED
NO
YES
806
NO
RESPONSE RECEIVED
862
SERVICE REQUEST
800
YES
807
863
TRANSMIT RESPONSE
TRANSMISSION COMPLETE
811
B
808
END
PROCESS FIRST CARD
C
812
809
GENERATE DISPLAY
813
D

FIG. 8A

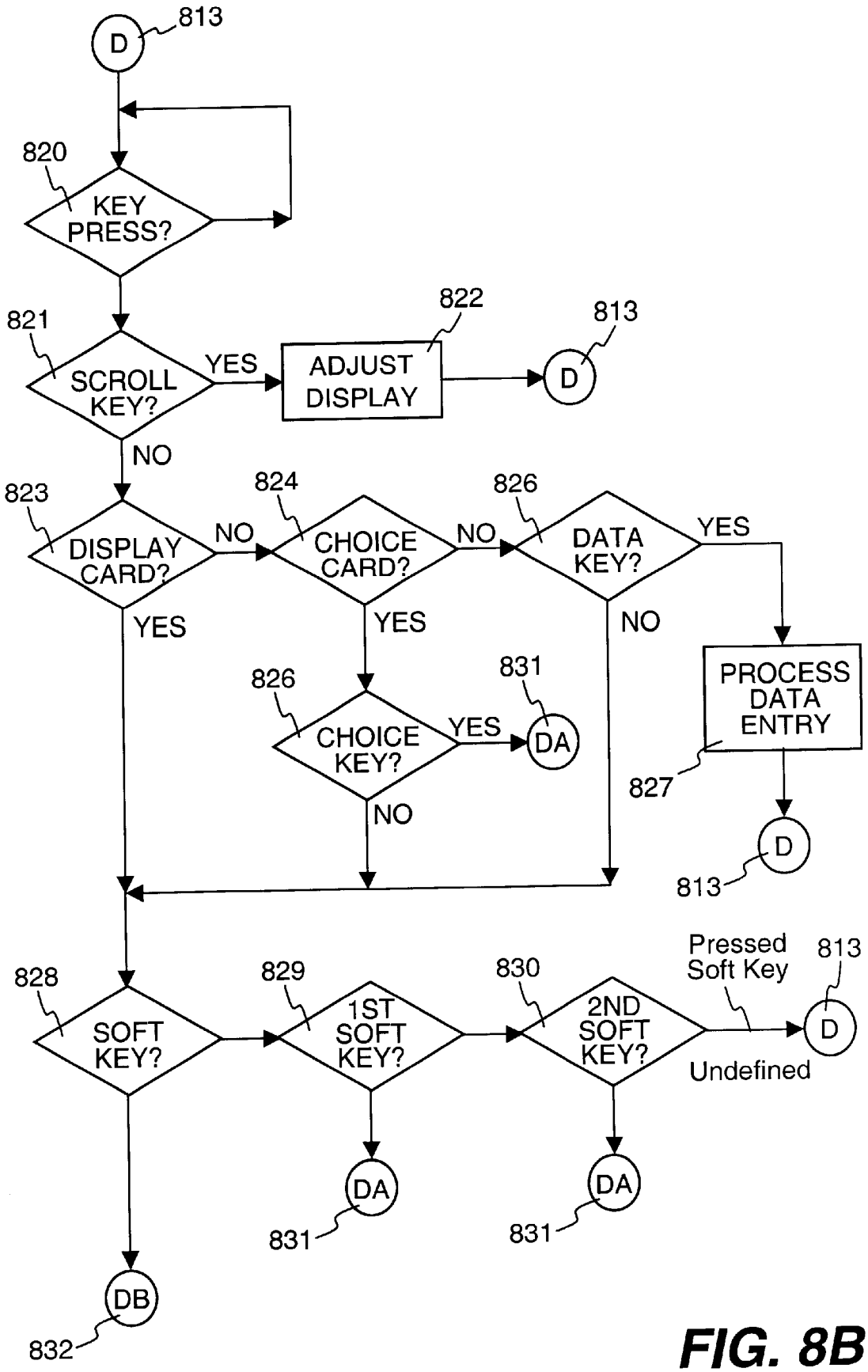

D
813
820
KEY PRESS?
821
SCROLL KEY?
YES
822
ADJUST DISPLAY
D
813
NO
823
DISPLAY CARD?
NO
824
CHOICE CARD?
NO
826
DATA KEY?
YES
YES
YES
NO
826
CHOICE KEY?
YES
831
DA
NO
PROCESS DATA ENTRY
827
D
813
828
SOFT KEY?
829
1ST SOFT KEY?
830
2ND SOFT KEY?
Pressed Soft Key
813
D
Undefined
DA
831
DA
831
DB
832

FIG. 8B

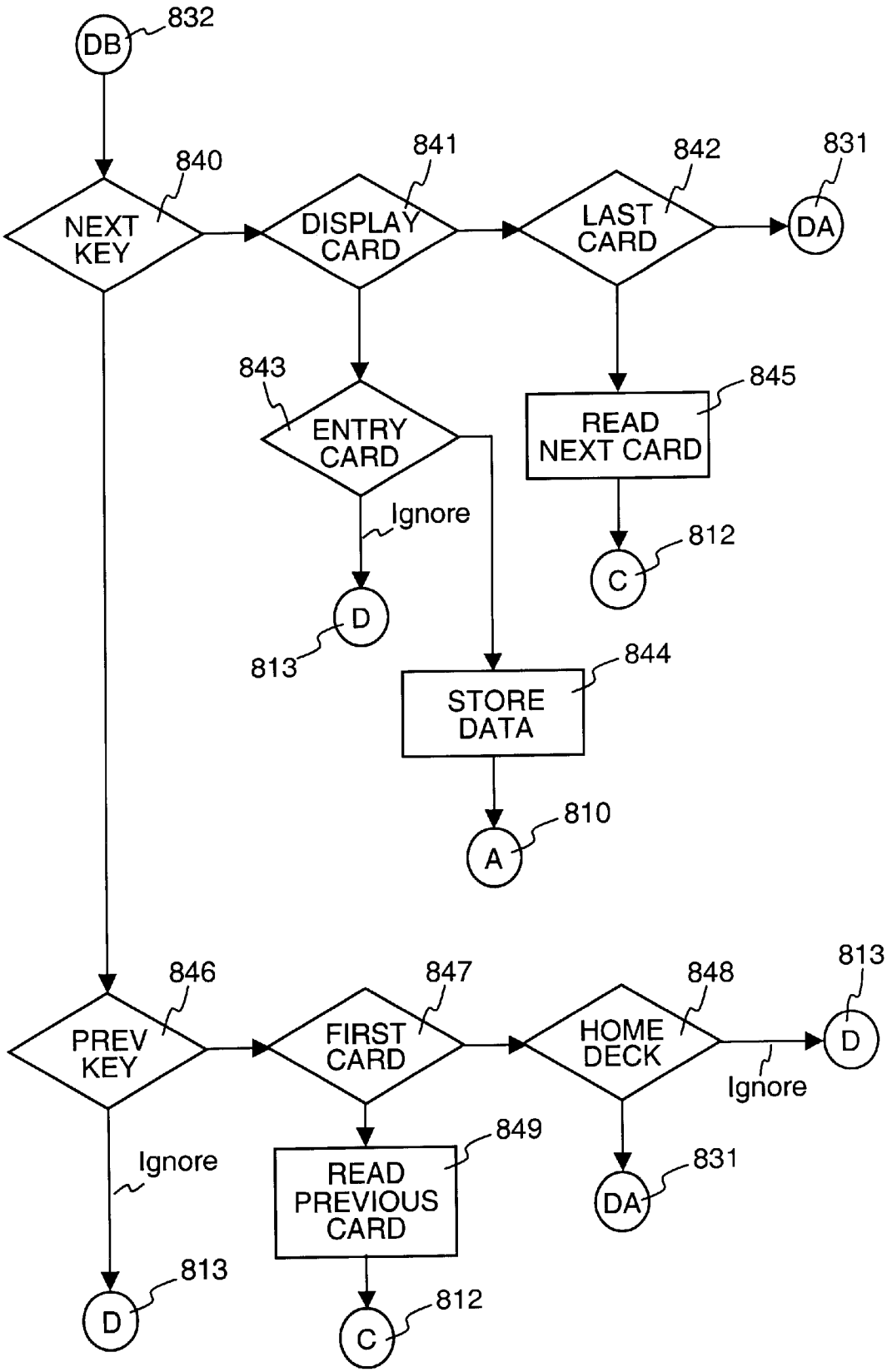

DB
832
NEXT KEY
840
DISPLAY CARD
841
LAST CARD
842
DA
831
ENTRY CARD
843
Ignore
D
813
READ NEXT CARD
845
C
812
STORE DATA
844
A
810
PREV KEY
846
FIRST CARD
847
HOME DECK
848
D
813
Ignore
Ignore
D
813
READ PREVIOUS CARD
849
C
812
DA
831

FIG. 8C

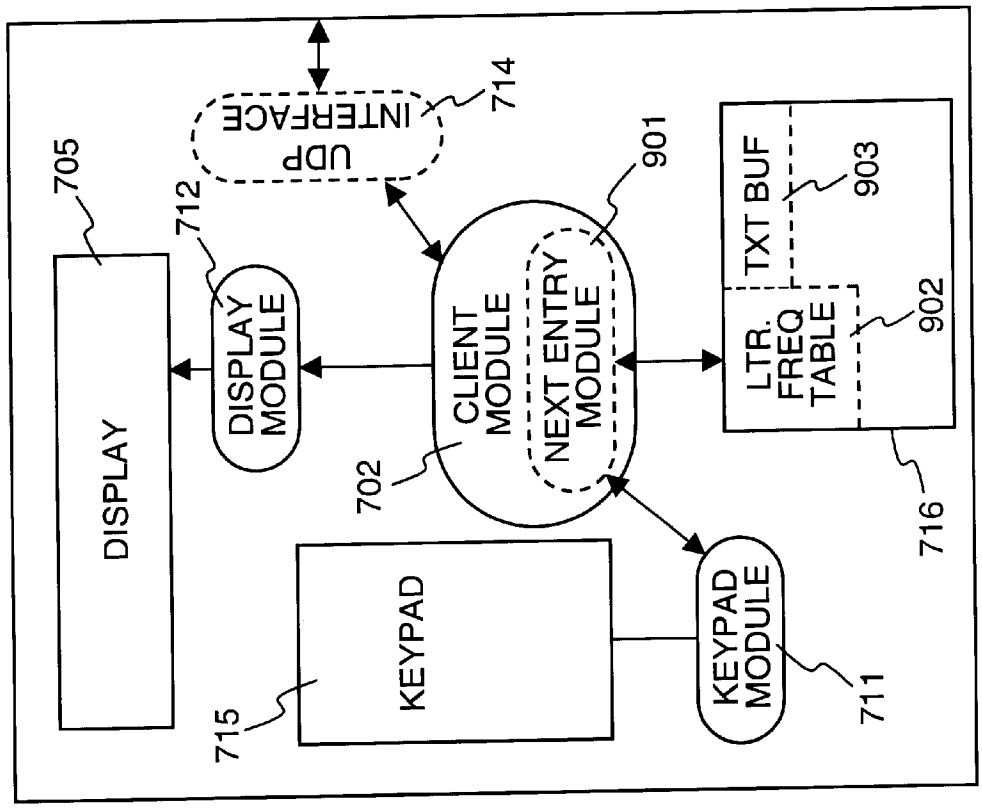

DISPLAY
705
DISPLAY MODULE
712
UDP INTERFACE
714
CLIENT MODULE
702
NEXT ENTRY MODULE
901
TXT BUF
903
LTR. FREQ TABLE
902
716
KEYPAD
715
KEYPAD MODULE
711

FIG. 9

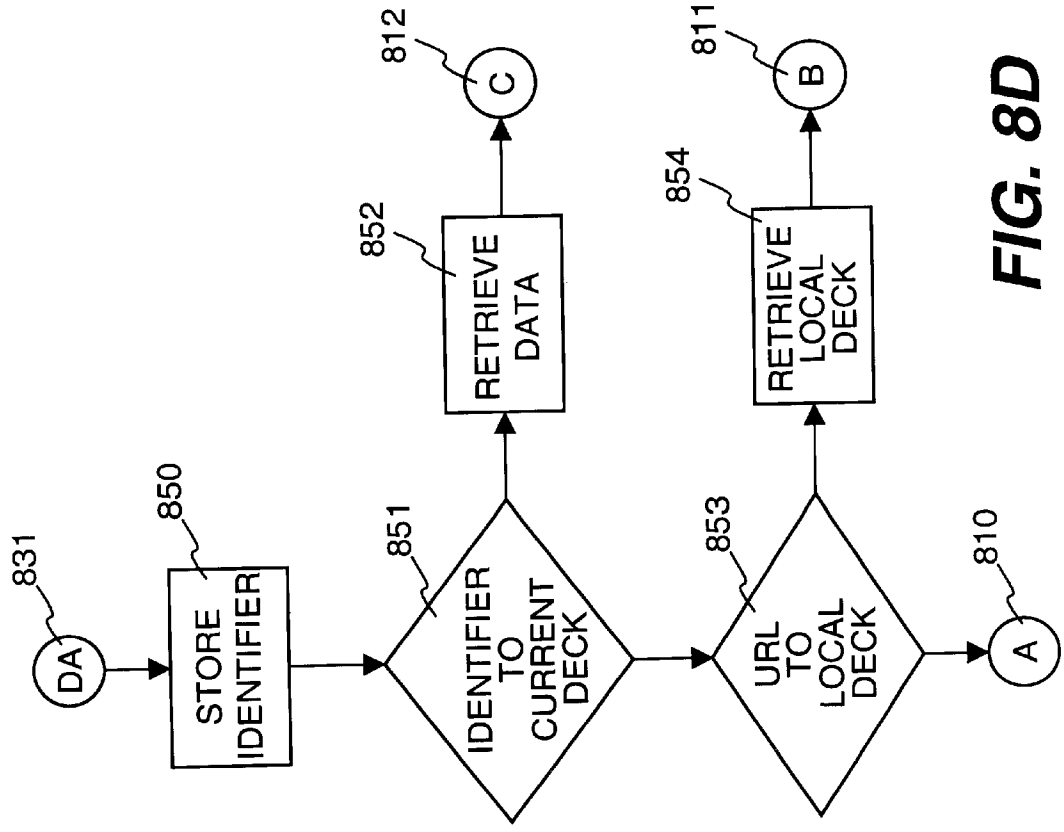

DA
831
STORE IDENTIFIER
850
IDENTIFIER TO CURRENT DECK
851
RETRIEVE DATA
852
C
812
URL TO LOCAL DECK
853
RETRIEVE LOCAL DECK
854
B
811
A
810

FIG. 8D

"-	-"	1	2	3	4	5	6	7	8	9
"a	a"	q	a	d	g	j	m	r	t	w
"a	b"	q	a	e	i	l	o	r	u	w
"a	c"	q	c	e	h	k	m	r	t	w
"a	d"	q	a	d	i	j	m	s	v	y
"a	e"	q	a	d	g	j	m	p	t	w
"a	f"	q	a	f	g	j	m	r	t	w
"a	g"	q	a	e	i	j	o	r	t	w
"a	h"	q	a	e	g	j	m	p	u	w
"a	i"	q	c	d	g	l	n	r	t	w
"a	j"	q	a	d	g	j	o	p	t	w
"a	k"	q	a	e	i	j	m	s	t	w
"a	l"	q	a	e	i	l	o	s	t	w
"a	m"	q	a	e	i	l	o	p	u	w
"a	n"	q	c	d	g	k	n	s	t	y
"a	o"	q	a	d	g	j	m	p	t	w
"a	p"	q	a	e	h	l	o	p	t	w
"a	q"	q	a	d	g	j	m	s	t	w
"a	r"	q	c	e	g	k	o	r	t	y
"a	s"	q	c	e	i	k	o	s	t	y
"a	t"	q	a	e	i	l	o	s	t	w
"a	u"	q	c	d	g	l	m	s	t	w
"a	v"	q	a	e	i	j	o	p	t	w
"a	w"	q	a	d	g	k	m	p	t	w
"a	x"	q	a	e	i	l	m	s	t	w
"a	y"	q	b	e	i	j	m	s	t	w
"a	z"	q	a	d	i	j	m	p	t	w
"a	"	q	a	d	g	j	m	p	t	w
"b	a"	q	c	d	g	k	n	s	t	y
"b	b"	q	a	d	i	j	m	s	t	y
"b	c"	q	a	d	g	j	o	r	t	w
"b	d"	q	a	d	g	j	m	s	t	w
"b	e"	q	c	e	g	l	n	s	t	x
"b	f"	q	a	d	g	j	m	p	t	w
"b	g"	q	a	d	g	j	m	p	t	w
"b	h"	q	a	d	g	j	m	p	t	w
"b	i"	q	c	d	g	l	n	s	t	w
"b	j"	q	a	e	g	j	m	p	t	w

FIG. 10A

"-	-"	1	2	3	4	5	6	7	8	9
"b	k"	q	a	d	g	j	m	p	t	w
"b	l"	q	a	e	i	j	o	p	t	y
"b	m"	q	a	d	g	j	m	r	t	w
"b	n"	q	a	d	g	j	m	p	t	w
"b	o"	q	a	d	i	j	o	s	u	x
"b	p"	q	a	d	g	j	m	s	t	w
"b	q"	q	a	d	g	j	m	p	t	w
"b	r"	q	a	e	i	j	o	p	u	w
"b	s"	q	c	d	i	j	o	s	t	w
"b	t"	q	a	d	g	j	m	p	t	w
"b	u"	q	a	d	i	l	m	s	t	y
"b	v"	q	a	d	i	j	m	p	t	w
"b	w"	q	a	e	g	j	m	p	t	w
"b	x"	q	a	d	g	j	m	p	t	w
"b	y"	q	a	d	g	j	m	p	t	w
"b	z"	q	a	d	g	j	m	p	t	w
"b	"	q	a	d	g	j	m	p	t	w
"c	a"	q	b	d	g	l	n	r	t	w
"c	b"	q	a	e	g	j	m	p	t	w
"c	c"	q	a	e	i	j	o	s	t	w
"c	d"	q	a	d	g	j	m	p	t	w
"c	e"	q	a	d	i	l	n	s	t	w
"c	f"	q	a	d	i	j	m	p	t	w
"c	g"	q	a	d	i	j	m	p	t	w
"c	h"	q	a	e	i	j	o	r	t	w
"c	i"	q	a	f	i	l	n	p	t	w
"c	j"	q	a	d	g	j	m	p	t	w
"c	k"	q	b	e	i	l	m	s	t	y
"c	l"	q	a	e	i	j	o	r	u	w
"c	m"	q	c	d	g	j	m	p	t	w
"c	n"	q	a	d	g	j	m	p	t	w
"c	o"	q	a	d	g	l	m	r	u	w
"c	p"	q	a	d	g	j	m	p	t	w
"c	q"	q	a	d	g	j	m	p	t	w
"c	r"	q	a	e	i	j	o	p	u	y
"c	s"	q	a	d	i	l	o	s	t	w
"c	t"	q	a	e	i	l	o	r	u	w

FIG. 10B

"-	-"	1	2	3	4	5	6	7	8	9
"c	u"	q	b	e	i	l	m	s	t	w
"c	v"	q	a	d	g	j	m	p	t	w
"c	w"	q	a	d	g	j	m	p	t	w
"c	x"	q	a	d	g	j	m	p	t	w
"c	y"	q	c	d	g	j	m	p	t	w
"c	z"	q	a	d	g	j	n	p	t	w
"c	"	q	a	d	g	j	m	p	t	w
"d	a"	q	b	d	g	j	n	r	t	y
"d	b"	q	a	d	g	j	m	p	t	w
"d	c"	q	a	d	g	j	m	p	t	w
"d	d"	q	a	e	i	l	m	r	t	w
"d	e"	q	a	d	g	l	m	r	v	x
"d	f"	q	a	d	i	j	m	p	t	w
"d	g"	q	a	e	g	j	m	p	t	w
"d	h"	q	a	e	g	j	m	p	t	w
"d	i"	z	a	f	g	l	n	s	t	w
"d	j"	q	a	d	g	j	m	p	t	w
"d	k"	q	a	d	g	j	m	p	t	w
"d	l"	q	a	e	i	j	m	p	t	y
"d	m"	q	a	d	i	j	m	s	t	w
"d	n"	q	a	e	g	j	m	p	t	w
"d	o"	q	c	e	i	j	n	p	t	w
"d	p"	q	a	d	g	j	m	p	t	w
"d	q"	q	a	d	g	j	m	p	u	w
"d	r"	q	a	e	i	j	o	p	t	w
"d	s"	q	a	e	i	j	m	p	t	w
"d	t"	q	a	d	g	j	m	r	t	w
"d	u"	q	c	e	g	l	m	r	t	w
"d	v"	q	a	d	i	j	m	p	t	w
"d	w"	q	a	d	i	j	o	p	t	w
"d	x"	q	a	d	g	j	m	p	t	w
"d	y"	q	a	d	g	j	m	p	t	w
"d	z"	q	a	d	g	j	m	p	t	w
"d	"	q	a	d	g	j	m	p	t	w
"e	a"	q	c	d	g	l	n	s	t	w
"e	b"	q	c	f	g	j	m	r	t	y
"e	c"	q	a	e	h	k	o	s	t	w

FIG. 10C

"-	-"	1	2	3	4	5	6	7	8	9
"e	d"	q	b	e	i	l	n	s	u	w
"e	e"	z	a	d	i	k	n	r	t	w
"e	f"	q	a	e	i	l	o	r	t	w
"e	g"	q	a	d	i	j	o	r	u	y
"e	h"	q	a	e	i	j	m	p	t	w
"e	i"	q	a	d	g	j	n	r	v	w
"e	j"	q	a	e	g	j	m	p	t	w
"e	k"	q	a	e	i	j	m	s	t	w
"e	l"	q	a	e	i	l	o	p	v	y
"e	m"	q	a	e	i	j	o	s	u	w
"e	n"	q	c	d	g	l	o	s	t	w
"e	o"	q	a	f	g	j	n	p	u	w
"e	p"	q	a	e	h	l	o	r	t	w
"e	q"	q	a	d	g	j	m	p	u	w
"e	r"	q	a	e	i	l	n	s	v	y
"e	s"	q	c	e	i	k	n	s	t	w
"e	t"	z	c	e	i	j	m	s	t	w
"e	u"	q	a	d	g	j	m	r	t	w
"e	v"	q	a	e	i	j	o	p	t	w
"e	w"	q	a	e	h	j	o	s	t	w
"e	x"	q	a	e	i	j	m	p	t	w
"e	y"	q	a	d	g	j	o	s	t	w
"e	z"	q	a	e	g	j	m	p	t	w
"e	"	q	a	d	g	j	m	p	t	w
"f	a"	q	c	d	i	k	m	s	v	x
"f	b"	q	a	d	g	j	m	p	t	w
"f	c"	q	c	d	g	j	m	p	t	w
"f	d"	q	a	d	g	j	m	p	t	w
"f	e"	q	a	e	g	l	m	r	t	w
"f	f"	q	a	e	i	j	o	s	t	w
"f	g"	q	a	d	g	j	m	p	t	w
"f	h"	q	a	d	g	j	m	p	t	w
"f	i"	q	c	e	g	l	n	r	t	x
"f	j"	q	a	d	g	j	m	p	t	w
"f	k"	q	a	d	g	j	m	p	t	w
"f	l"	q	a	e	i	j	o	p	t	y
"f	m"	q	a	d	g	j	m	p	t	w

FIG. 10D

"-	-"	1	2	3	4	5	6	7	8	9
"f	n"	q	a	d	g	j	m	p	t	w
"f	o"	q	c	d	g	l	n	r	u	w
"f	p"	q	a	d	g	j	m	p	t	w
"f	q"	q	a	d	g	j	m	p	t	w
"f	r"	q	a	e	i	j	o	p	t	y
"f	s"	q	a	d	g	j	m	p	t	w
"f	t"	q	a	e	g	k	m	p	t	w
"f	u"	q	a	d	g	l	n	r	t	w
"f	v"	q	a	d	g	j	m	p	t	w
"f	w"	q	a	d	g	j	m	p	t	w
"f	x"	q	a	d	g	j	m	p	t	w
"f	y"	q	a	d	i	j	m	p	t	w
"f	z"	q	a	d	g	j	m	p	t	w
"f	"	q	a	d	g	j	m	p	t	w
"g	a"	z	a	d	i	l	n	r	t	w
"g	b"	q	a	d	g	j	m	p	t	w
"g	c"	q	a	d	g	j	m	s	t	w
"g	d"	q	a	d	g	j	o	p	t	w
"g	e"	q	a	d	g	l	n	s	t	w
"g	f"	q	a	d	g	j	m	p	u	w
"g	g"	q	a	e	g	l	m	p	t	w
"g	h"	q	a	e	g	j	o	p	t	w
"g	i"	q	c	f	g	j	n	s	v	w
"g	j"	q	a	d	g	j	m	p	t	w
"g	k"	q	a	d	g	j	m	p	t	w
"g	l"	q	a	e	i	j	o	p	t	y
"g	m"	q	a	d	g	l	m	s	t	w
"g	n"	q	a	e	i	j	o	p	t	w
"g	o"	q	a	e	i	j	o	p	t	w
"g	p"	q	a	d	g	j	m	p	t	w
"g	q"	q	a	d	g	j	m	p	t	w
"g	r"	q	a	e	g	j	o	p	t	w
"g	s"	q	a	d	g	j	m	p	t	w
"g	t"	q	a	e	h	j	o	p	t	w
"g	u"	q	a	e	i	l	m	r	t	y
"g	v"	q	a	d	g	j	m	p	t	w
"g	w"	q	a	d	g	j	m	p	t	w

FIG. 10E

"-	-"	1	2	3	4	5	6	7	8	9
"g	x"	q	a	d	g	j	m	p	t	w
"g	y"	q	a	d	g	j	m	p	t	w
"g	z"	q	a	d	g	j	m	p	t	w
"g	"	q	a	d	g	j	m	p	t	w
"h	a"	q	b	d	i	l	n	s	t	w
"h	b"	q	a	d	g	j	m	p	t	w
"h	c"	q	a	d	g	j	o	p	t	w
"h	d"	q	a	d	g	j	m	r	t	w
"h	e"	q	c	d	i	l	n	r	t	y
"h	f"	q	a	d	g	j	m	p	t	w
"h	g"	q	a	d	g	j	m	p	t	w
"h	h"	q	a	d	g	j	m	p	t	w
"h	i"	q	c	e	g	l	n	s	t	w
"h	j"	q	a	d	g	j	m	p	t	w
"h	k"	q	a	d	g	j	m	p	t	w
"h	l"	q	a	d	g	j	m	p	t	y
"h	m"	q	a	e	g	j	m	p	t	w
"h	n"	q	a	d	i	j	o	p	t	w
"h	o"	q	b	d	i	l	n	r	u	w
"h	p"	q	a	d	g	j	m	p	u	w
"h	q"	q	a	d	g	j	m	p	t	w
"h	r"	q	a	e	i	j	o	p	t	w
"h	s"	q	a	d	g	j	m	p	t	w
"h	t"	q	a	d	g	l	m	s	t	w
"h	u"	q	a	d	g	l	m	r	t	w
"h	v"	q	a	d	g	j	m	p	t	w
"h	w"	q	a	d	g	j	m	p	t	w
"h	x"	q	a	d	g	j	m	p	t	w
"h	y"	q	a	d	g	j	m	p	t	w
"h	z"	q	a	d	g	j	m	p	t	w
"h	"	q	a	d	g	j	m	p	t	w
"i	a"	q	b	d	g	l	n	r	t	w
"i	b"	q	a	e	i	l	m	r	u	w
"i	c"	q	a	e	h	k	o	r	t	y
"i	d"	q	a	e	i	l	n	p	u	w
"i	e"	q	c	d	g	l	n	r	v	w
"i	f"	q	a	f	i	j	o	p	t	y

FIG. 10F

"-	-"	1	2	3	4	5	6	7	8	9
"i	g"	q	a	e	h	j	n	r	u	w
"i	h"	q	a	d	g	j	m	p	t	w
"i	i"	q	a	d	g	j	m	p	t	w
"i	j"	q	a	d	g	j	o	p	t	w
"i	k"	q	a	e	g	k	m	p	t	w
"i	l"	q	a	e	i	l	o	s	t	y
"i	m"	q	a	e	i	j	m	p	u	w
"i	n"	q	c	e	g	k	n	s	t	w
"i	o"	q	a	d	g	l	n	r	u	w
"i	p"	q	a	d	i	l	m	s	t	w
"i	q"	q	a	d	g	j	m	p	u	w
"i	r"	q	c	e	i	l	n	s	t	w
"i	s"	q	c	e	h	k	n	s	t	w
"i	t"	q	a	e	h	l	o	s	t	y
"i	u"	q	a	d	g	j	m	p	t	w
"i	v"	q	a	e	i	j	m	p	t	w
"i	w"	q	a	d	g	j	m	p	t	w
"i	x"	q	a	e	g	j	m	p	t	w
"i	y"	q	a	d	g	j	m	p	t	w
"i	z"	q	a	e	i	j	o	p	t	w
"i	"	q	a	d	g	j	m	p	t	w
"j	a"	q	c	d	g	j	n	p	t	w
"j	b"	q	a	d	g	j	m	p	t	w
"j	c"	q	a	d	g	j	m	p	t	w
"j	d"	q	a	d	g	j	m	p	t	w
"j	e"	q	c	d	g	j	m	r	t	w
"j	f"	q	a	d	g	j	m	p	t	w
"j	g"	q	a	d	g	j	m	p	t	w
"j	h"	q	a	d	g	j	m	p	t	w
"j	i"	q	a	d	g	j	m	p	t	w
"j	j"	q	a	d	g	j	m	p	t	w
"j	k"	q	a	d	g	j	m	p	t	w
"j	l"	q	a	d	g	j	m	p	t	w
"j	m"	q	a	d	g	j	m	p	t	w
"j	n"	q	a	d	g	j	m	p	t	w
"j	o"	q	b	e	i	j	m	r	t	w
"j	p"	q	a	d	g	j	m	p	t	w

FIG. 10G

"-	-"	1	2	3	4	5	6	7	8	9
"j	q"	q	a	d	g	j	m	p	t	w
"j	r"	q	a	d	g	j	m	p	t	w
"j	s"	q	a	d	g	j	m	p	t	w
"j	t"	q	a	d	g	j	m	p	t	w
"j	u"	q	a	d	g	l	m	s	t	w
"j	v"	q	a	d	g	j	m	p	t	w
"j	w"	q	a	d	g	j	m	p	t	w
"j	x"	q	a	d	g	j	m	p	t	w
"j	y"	q	a	d	g	j	m	p	t	w
"j	z"	q	a	d	g	j	m	p	t	w
"j	"	q	a	d	g	j	m	p	t	w
"k	a"	q	a	d	g	j	n	s	t	y
"k	b"	q	a	d	g	j	o	p	t	w
"k	c"	q	a	d	g	j	m	p	t	w
"k	d"	q	a	d	g	j	o	p	t	w
"k	e"	q	a	d	g	l	n	s	t	y
"k	f"	q	a	d	g	j	m	p	t	w
"k	g"	q	a	d	g	j	m	r	t	w
"k	h"	q	a	d	g	j	m	p	t	w
"k	i"	q	a	d	g	l	n	p	t	w
"k	j"	q	a	d	g	j	m	p	t	w
"k	k"	q	a	d	g	j	o	p	t	w
"k	l"	q	a	d	i	j	m	p	t	y
"k	m"	q	a	d	g	j	m	p	t	w
"k	n"	q	a	d	g	j	o	p	t	w
"k	o"	q	a	d	g	l	n	p	t	w
"k	p"	q	a	d	g	j	m	p	t	w
"k	q"	q	a	d	g	j	m	p	t	w
"k	r"	q	a	d	g	j	m	p	t	w
"k	s"	q	a	d	g	j	m	p	t	w
"k	t"	q	a	d	g	j	m	p	t	w
"k	u"	q	a	d	g	j	m	p	t	w
"k	v"	q	a	d	g	j	m	p	t	w
"k	w"	q	a	d	g	j	m	p	t	w
"k	x"	q	a	d	g	j	m	p	t	w
"k	y"	q	a	d	g	j	m	p	t	w
"k	z"	q	a	d	g	j	m	p	t	w

FIG. 10H

"-	-"	1	2	3	4	5	6	7	8	9
"k	"	q	a	d	g	j	m	p	t	w
"l	a"	q	b	d	i	j	n	r	t	y
"l	b"	q	a	d	g	j	o	p	t	w
"l	c"	q	a	d	g	j	o	p	t	w
"l	d"	q	a	e	i	j	n	s	t	w
"l	e"	q	a	d	g	l	m	s	t	x
"l	f"	q	a	d	g	j	m	p	t	w
"l	g"	q	a	d	g	j	m	p	t	w
"l	h"	q	a	d	g	j	m	p	t	w
"l	i"	z	c	e	g	k	n	s	t	w
"l	j"	q	a	d	g	j	m	p	t	w
"l	k"	q	a	e	i	j	m	s	t	w
"l	l"	q	c	e	i	j	o	s	u	y
"l	m"	q	a	d	g	j	o	s	t	w
"l	n"	q	a	e	g	j	m	p	t	w
"l	o"	q	c	d	g	j	o	s	t	w
"l	p"	q	a	d	h	j	m	s	t	w
"l	q"	q	a	d	g	j	m	p	t	w
"l	r"	q	a	e	g	j	m	p	t	w
"l	s"	q	a	e	i	j	o	p	t	w
"l	t"	q	a	e	i	j	o	s	t	w
"l	u"	q	c	d	g	l	m	s	t	w
"l	v"	q	a	e	i	j	m	p	t	w
"l	w"	q	a	e	g	j	m	p	t	w
"l	x"	q	a	d	g	j	m	p	t	w
"l	y"	z	a	d	i	j	n	s	t	w
"l	z"	q	a	d	g	j	m	p	t	w
"l	"	q	a	d	g	j	m	p	t	w
"m	a"	q	c	d	i	k	n	r	t	y
"m	b"	q	a	e	i	l	m	p	t	y
"m	c"	q	c	d	i	j	o	p	t	w
"m	d"	q	a	d	g	j	m	p	t	w
"m	e"	q	a	e	g	l	n	s	t	w
"m	f"	q	a	d	g	j	o	p	t	w
"m	g"	q	a	d	g	j	m	p	t	w
"m	h"	z	a	d	g	j	m	p	t	w
"m	i"	z	c	d	g	l	n	s	t	x

FIG. 10I

"-	-"	1	2	3	4	5	6	7	8	9
"m	j"	q	a	d	g	j	m	p	t	w
"m	k"	q	a	d	g	j	m	p	t	w
"m	l"	q	a	e	g	j	m	p	t	y
"m	m"	q	a	e	i	j	o	p	u	w
"m	n"	q	a	d	g	j	m	p	t	w
"m	o"	q	b	d	i	j	n	r	v	w
"m	p"	q	a	e	i	l	o	r	t	w
"m	q"	q	a	d	g	j	m	p	t	w
"m	r"	q	a	d	g	j	m	p	t	w
"m	s"	q	a	e	g	j	m	p	t	w
"m	t"	q	a	d	g	j	m	p	t	w
"m	u"	q	c	d	g	l	n	s	t	w
"m	v"	q	a	d	g	j	m	p	t	w
"m	w"	q	a	d	h	j	m	p	t	w
"m	x"	q	a	d	g	j	m	p	t	w
"m	y"	q	c	d	g	j	m	s	t	w
"m	z"	q	a	d	g	j	m	p	t	w
"m	"	q	a	d	g	j	m	p	t	w
"n	a"	q	b	d	g	l	m	r	t	w
"n	b"	q	a	d	g	j	m	p	t	w
"n	c"	q	a	e	i	l	o	r	t	y
"n	d"	q	a	e	i	l	o	s	t	y
"n	e"	q	c	e	i	l	n	s	t	w
"n	f"	q	a	e	i	j	o	s	u	w
"n	g"	q	a	e	i	l	m	s	u	w
"n	h"	q	a	d	g	j	m	p	t	w
"n	i"	z	c	e	g	l	n	s	t	x
"n	j"	q	a	d	g	j	m	p	u	w
"n	k"	q	a	e	i	j	n	s	t	w
"n	l"	q	a	e	i	j	o	p	t	y
"n	m"	q	a	e	g	j	m	p	t	w
"n	n"	q	a	e	i	j	o	p	u	y
"n	o"	q	b	d	g	l	n	r	t	w
"n	p"	q	a	d	g	j	o	p	u	w
"n	q"	q	a	d	g	j	m	p	u	w
"n	r"	q	a	d	g	j	m	p	t	w
"n	s"	q	a	e	i	l	m	p	t	w

FIG. 10J

"-	-"	1	2	3	4	5	6	7	8	9
"n	t"	q	a	e	i	l	o	s	u	y
"n	u"	q	a	f	g	j	m	s	t	x
"n	v"	q	a	e	i	j	o	p	t	w
"n	w"	q	a	e	g	j	m	p	t	w
"n	X"	q	a	d	g	j	m	p	t	w
"n	y"	q	b	d	h	j	m	p	t	w
"n	z"	q	a	d	g	j	m	p	t	w
"n	"	q	a	d	g	j	m	p	t	w
"o	a"	q	b	d	g	l	m	s	t	w
"o	b"	q	a	e	i	l	o	s	t	w
"o	c"	q	a	e	h	k	o	s	u	w
"o	d"	q	a	e	i	l	o	s	u	y
"o	e"	q	a	d	g	j	m	s	t	w
"o	f"	q	a	f	g	j	m	p	t	w
"o	g"	q	a	e	i	j	o	r	t	y
"o	h"	q	a	d	g	j	n	p	t	w
"o	i"	q	c	d	g	l	n	p	t	w
"o	j"	q	a	e	g	j	m	p	t	w
"o	k"	q	a	e	i	j	m	s	t	w
"o	l"	q	b	d	i	l	o	s	u	w
"o	m"	q	a	e	i	l	m	p	t	w
"o	n"	q	a	e	g	l	n	s	t	y
"o	o"	q	a	d	g	k	n	s	t	w
"o	p"	q	a	e	h	l	m	p	t	y
"o	q"	q	a	d	g	j	m	p	t	w
"o	r"	q	a	e	g	k	m	r	t	w
"o	s"	q	c	e	i	j	o	s	t	y
"o	t"	q	a	e	h	j	o	s	t	y
"o	u"	q	c	d	g	l	n	r	t	y
"o	v"	q	a	e	i	j	m	p	t	w
"o	w"	q	a	e	i	j	n	s	t	w
"o	x"	q	a	d	i	j	m	p	t	w
"o	y"	q	a	e	g	j	m	s	t	w
"o	z"	q	a	d	g	j	m	p	t	w
"o	"	q	a	d	g	j	m	p	t	w
"p	a"	q	c	d	g	l	n	r	t	y
"p	b"	q	a	e	g	j	m	p	t	w

FIG. 10K

"-	-"	1	2	3	4	5	6	7	8	9
"p	c"	q	a	d	g	j	m	s	t	w
"p	d"	q	a	d	g	j	m	p	t	w
"p	e"	q	c	d	g	j	n	r	t	w
"p	f"	q	a	d	g	j	m	r	t	w
"p	g"	q	a	d	g	j	m	p	t	w
"p	h"	q	a	e	i	j	o	s	t	y
"p	i"	q	c	d	g	l	n	r	t	x
"p	j"	q	a	d	g	j	m	p	t	w
"p	k"	q	a	d	g	j	m	p	t	w
"p	l"	q	a	e	i	j	o	p	u	y
"p	m"	q	a	e	g	j	m	p	t	w
"p	n"	q	a	d	g	j	m	p	t	w
"p	o"	q	c	d	i	l	n	r	t	w
"p	p"	q	a	e	i	l	o	p	t	y
"p	q"	q	a	d	g	j	m	p	t	w
"p	r"	q	a	e	i	j	o	p	t	y
"p	s"	q	a	d	g	j	m	p	t	w
"p	t"	q	c	e	i	j	o	s	t	w
"p	u"	q	b	d	g	j	m	r	t	w
"p	v"	q	a	d	g	j	m	p	t	w
"p	w"	q	a	d	g	j	m	p	t	w
"p	x"	q	a	d	g	j	m	p	t	w
"p	y"	q	a	d	g	j	m	r	t	w
"p	z"	q	a	d	g	j	m	p	t	w
"p	"	q	a	d	g	j	m	p	t	w
"q	a"	q	a	d	g	j	m	p	t	w
"q	b"	q	a	d	g	j	m	p	t	w
"q	c"	q	a	d	g	j	m	p	t	w
"q	d"	q	a	d	g	j	m	p	t	w
"q	e"	q	a	d	g	j	m	p	t	w
"q	f"	q	a	d	g	j	m	p	t	w
"q	g"	q	a	d	g	j	m	p	t	w
"q	h"	q	a	d	g	j	m	p	t	w
"q	i"	q	a	d	g	j	m	p	t	w
"q	j"	q	a	d	g	j	m	p	t	w
"q	k"	q	a	d	g	j	m	p	t	w
"q	l"	q	a	d	g	j	m	p	t	w

FIG. 10L

"-	-"	1	2	3	4	5	6	7	8	9
"q	m"	q	a	d	g	j	m	p	t	w
"q	n"	q	a	d	g	j	m	p	t	w
"q	o"	q	a	d	g	j	m	p	t	w
"q	p"	q	a	d	g	j	m	p	t	w
"q	q"	q	a	d	g	j	m	p	t	w
"q	r"	q	a	d	g	j	m	p	t	w
"q	s"	q	a	d	g	j	m	p	t	w
"q	t"	q	a	d	g	j	m	p	t	w
"q	u"	q	a	e	i	j	o	p	t	w
"q	v"	q	a	d	g	j	m	p	t	w
"q	w"	q	a	d	g	j	m	p	t	w
"q	x"	q	a	d	g	j	m	p	t	w
"q	y"	q	a	d	g	j	m	p	t	w
"q	z"	q	a	d	g	j	m	p	t	w
"q	"	q	a	d	g	j	m	p	t	w
"r	a"	q	c	d	g	l	n	p	t	w
"r	b"	q	a	d	g	j	o	p	t	w
"r	c"	q	a	e	h	l	m	p	u	w
"r	d"	q	a	e	i	j	n	s	v	w
"r	e"	q	a	e	g	l	n	s	v	w
"r	f"	q	a	e	g	j	o	p	t	w
"r	g"	q	a	e	i	j	m	s	u	w
"r	h"	q	a	e	g	j	o	p	t	w
"r	i"	z	c	e	g	l	n	p	t	w
"r	j"	q	a	d	g	j	m	p	t	w
"r	k"	q	a	e	i	l	m	s	t	y
"r	l"	q	a	d	i	j	o	p	t	y
"r	m"	q	a	e	i	j	m	s	t	w
"r	n"	q	a	e	i	j	o	s	t	w
"r	o"	q	b	d	g	l	m	p	u	w
"r	p"	q	a	d	g	j	o	r	t	w
"r	q"	q	a	d	g	j	m	p	t	w
"r	r"	q	a	e	i	j	o	p	t	y
"r	s"	q	a	e	i	j	o	p	t	w
"r	t"	z	a	e	h	j	n	s	u	w
"r	u"	q	c	e	g	l	n	s	t	w
"r	v"	q	a	e	i	j	m	p	t	w

FIG. 10M

"-	-"	1	2	3	4	5	6	7	8	9
"r	w"	q	a	d	i	j	o	p	t	w
"r	x"	q	a	d	g	j	m	p	t	w
"r	y"	q	b	d	i	j	m	p	t	w
"r	z"	q	a	d	g	j	m	p	t	w
"r	"	q	a	d	g	j	m	p	t	w
"s	a"	q	b	d	g	l	m	p	v	y
"s	b"	q	a	d	g	j	m	p	t	w
"s	c"	q	a	d	h	l	o	r	u	w
"s	d"	q	a	d	g	j	n	p	t	w
"s	e"	q	c	e	g	l	n	r	t	y
"s	f"	q	a	e	g	j	m	p	u	w
"s	g"	q	a	d	g	j	m	r	t	w
"s	h"	q	a	e	i	l	o	r	t	w
"s	i"	z	c	d	g	l	n	s	t	x
"s	j"	q	a	d	g	j	m	p	t	w
"s	k"	q	a	e	i	j	m	p	t	w
"s	l"	q	a	e	i	j	o	p	t	y
"s	m"	q	a	e	i	j	o	s	t	w
"s	n"	q	a	e	g	j	m	p	t	w
"s	o"	q	c	f	g	l	m	r	u	w
"s	p"	q	a	e	g	l	o	r	t	w
"s	q"	q	a	d	g	l	m	p	u	w
"s	r"	q	c	d	g	j	m	p	t	w
"s	s"	q	a	e	i	j	o	p	u	w
"s	t"	q	a	e	i	l	o	r	u	y
"s	u"	q	b	e	g	l	m	p	t	w
"s	v"	q	a	d	g	j	m	p	t	w
"s	w"	q	a	e	i	j	o	p	t	w
"s	x"	q	a	d	g	j	m	p	t	w
"s	y"	q	a	d	g	j	m	s	t	w
"s	z"	q	a	d	g	j	m	p	t	w
"s	"	q	a	d	g	j	m	p	t	w
"t	a"	q	c	f	i	l	n	r	t	y
"t	b"	q	a	d	g	j	m	p	t	w
"t	c"	z	a	d	h	j	o	p	t	w
"t	d"	q	a	d	g	j	m	p	t	w
"t	e"	q	c	d	g	l	m	r	u	x

FIG. 10N

"-	-"	1	2	3	4	5	6	7	8	9
"t	f"	q	a	d	i	j	o	p	t	w
"t	g"	q	a	e	g	j	o	p	t	w
"t	h"	q	a	e	i	l	o	r	u	w
"t	i"	z	c	e	g	l	o	s	v	w
"t	j"	q	a	d	g	j	m	p	t	w
"t	k"	q	a	e	g	j	m	p	t	w
"t	l"	q	a	e	i	j	m	p	t	y
"t	m"	q	a	e	g	l	o	p	t	w
"t	n"	q	a	e	g	j	m	p	t	w
"t	o"	q	c	d	g	l	m	r	u	w
"t	p"	q	c	d	g	j	m	p	u	w
"t	q"	q	a	d	g	j	m	p	t	w
"t	r"	q	a	e	i	j	o	p	u	y
"t	s"	q	c	e	i	j	o	p	u	w
"t	t"	q	a	e	i	l	o	p	t	y
"t	u"	q	a	f	i	j	n	r	t	w
"t	v"	q	a	d	g	j	m	p	t	w
"t	w"	q	a	e	i	j	o	p	t	w
"t	x"	q	a	d	g	j	m	p	t	w
"t	y"	q	a	d	g	l	m	p	t	w
"t	z"	q	a	d	g	j	m	p	t	w
"t	"	q	a	d	g	j	m	p	t	w
"u	a"	q	a	d	g	l	n	r	t	w
"u	b"	q	a	e	i	j	m	s	t	w
"u	c"	q	c	e	h	k	m	p	t	w
"u	d"	q	a	e	i	j	o	p	t	y
"u	e"	q	a	d	g	j	n	s	t	w
"u	f"	q	a	f	g	j	m	p	t	w
"u	g"	q	a	e	h	j	m	s	t	w
"u	h"	q	a	d	g	j	m	p	t	w
"u	i"	q	c	d	g	l	n	r	t	w
"u	j"	q	a	d	g	j	m	p	t	w
"u	k"	q	a	d	g	j	m	p	t	w
"u	l"	q	a	d	g	l	m	p	t	y
"u	m"	q	b	e	i	j	m	p	t	w
"u	n"	q	c	d	i	l	n	s	t	w
"u	o"	q	a	d	g	j	m	p	t	w

FIG. 10O

"-	-"	1	2	3	4	5	6	7	8	9
"u	p"	q	a	d	i	l	o	p	t	y
"u	q"	q	a	d	g	j	m	p	t	w
"u	r"	q	c	e	i	l	n	s	t	y
"u	s"	q	a	e	i	l	m	s	t	y
"u	t"	q	a	e	i	l	o	s	t	w
"u	u"	q	c	d	g	j	m	p	t	w
"u	v"	q	a	d	g	j	m	p	t	w
"u	w"	q	a	d	g	j	m	p	t	w
"u	x"	q	a	d	g	j	m	p	t	w
"u	y"	q	a	e	g	j	m	s	t	w
"u	z"	q	a	d	g	j	m	p	t	w
"u	"	q	a	d	g	j	m	p	t	w
"v	a"	q	c	d	i	l	n	r	t	w
"v	b"	q	a	d	g	j	m	p	t	w
"v	c"	q	a	d	g	j	m	s	t	w
"v	d"	q	a	d	g	j	m	p	t	w
"v	e"	q	a	d	g	l	n	r	t	w
"v	f"	q	a	d	g	j	m	p	t	w
"v	g"	q	a	d	g	j	m	p	t	w
"v	h"	q	a	d	g	j	m	p	t	w
"v	i"	q	c	e	g	j	n	s	t	w
"v	j"	q	a	d	g	j	m	p	t	w
"v	k"	q	a	d	g	j	m	p	t	w
"v	l"	q	a	d	g	j	m	s	t	w
"v	m"	q	a	d	g	j	m	p	t	w
"v	n"	q	a	d	g	j	m	p	t	w
"v	o"	q	a	d	i	l	o	r	t	w
"v	p"	q	a	d	g	j	m	p	t	w
"v	q"	q	a	d	g	j	m	p	t	w
"v	r"	q	a	d	g	j	m	p	t	w
"v	s"	q	a	d	g	j	m	p	t	w
"v	t"	q	a	d	g	j	m	p	t	w
"v	u"	q	a	d	g	j	m	p	t	w
"v	v"	q	a	d	g	j	m	p	t	w
"v	w"	q	a	d	g	j	m	p	t	w
"v	x"	q	a	d	g	j	m	p	t	w
"v	y"	q	a	d	g	j	m	p	t	w

FIG. 10P

"-	-"	1	2	3	4	5	6	7	8	9
"v	z"	q	a	d	g	j	m	p	t	w
"v	"	q	a	d	g	j	m	p	t	w
"w	a"	q	b	d	i	l	n	r	v	y
"w	b"	q	a	d	g	j	m	p	t	w
"w	c"	q	a	e	g	j	m	p	t	w
"w	d"	q	a	f	g	j	m	p	t	w
"w	e"	q	b	e	i	l	n	r	v	w
"w	f"	q	a	d	g	j	m	p	t	w
"w	g"	q	a	d	g	j	m	p	t	w
"w	h"	q	a	e	i	j	o	p	t	y
"w	i"	q	c	d	g	l	n	s	t	w
"w	j"	q	a	d	g	j	m	p	t	w
"w	k"	q	a	d	g	j	m	p	t	w
"w	l"	q	a	d	g	j	m	p	t	w
"w	m"	q	a	d	g	j	m	p	t	w
"w	n"	q	a	d	g	l	m	s	t	w
"w	o"	q	a	d	g	j	n	r	u	w
"w	p"	q	a	d	h	j	m	p	t	w
"w	q"	q	a	d	g	j	m	p	t	w
"w	r"	q	a	d	i	j	o	p	t	w
"w	s"	q	a	e	g	j	m	p	t	w
"w	t"	q	a	d	g	j	m	p	t	w
"w	u"	q	a	d	g	j	m	p	t	w
"w	v"	q	a	d	g	j	m	p	t	w
"w	w"	q	a	d	g	j	m	p	t	w
"w	x"	q	a	d	g	j	m	p	t	w
"w	y"	q	a	d	g	j	m	p	t	w
"w	z"	q	a	d	g	j	m	p	t	w
"w	"	q	a	d	g	j	m	p	t	w
"x	a"	q	c	d	g	j	m	p	t	w
"x	b"	q	a	d	g	j	m	p	t	w
"x	c"	q	a	e	h	j	m	p	t	w
"x	d"	q	a	d	g	j	m	p	t	w
"x	e"	q	c	d	g	l	m	s	t	x
"x	f"	q	a	d	g	j	m	p	t	w
"x	g"	q	a	d	g	j	m	p	t	w
"x	h"	q	a	d	g	j	m	p	t	w

FIG. 10Q

"-	-"	1	2	3	4	5	6	7	8	9
"x	i"	q	c	d	g	j	n	s	t	w
"x	j"	q	a	d	g	j	m	p	t	w
"x	k"	q	a	d	g	j	m	p	t	w
"x	l"	q	a	d	i	j	m	p	t	w
"x	m"	q	a	d	g	j	m	p	t	w
"x	n"	q	a	d	g	j	m	p	t	w
"x	o"	q	a	d	g	j	m	p	t	w
"x	p"	q	a	e	i	l	o	p	t	w
"x	q"	q	a	d	g	j	m	p	t	w
"x	r"	q	a	d	g	j	m	p	t	w
"x	s"	q	a	e	i	j	m	p	t	w
"x	t"	q	a	e	g	j	m	r	u	w
"x	u"	q	a	d	g	j	m	p	t	w
"x	v"	q	a	d	g	j	m	p	t	w
"x	w"	q	a	d	g	j	m	p	t	w
"x	x"	q	a	d	g	j	m	p	t	w
"x	y"	z	a	d	g	j	m	p	t	w
"x	z"	q	a	d	g	j	m	p	t	w
"x	"	q	a	d	g	j	m	p	t	w
"y	a"	q	a	d	g	j	m	p	t	w
"y	b"	q	a	e	i	j	o	p	t	w
"y	c"	q	a	d	g	l	o	p	t	w
"y	d"	q	a	d	g	j	m	p	t	w
"y	e"	q	a	e	g	j	m	s	t	w
"y	f"	q	a	d	g	j	m	p	t	w
"y	g"	q	a	d	g	j	m	p	t	w
"y	h"	q	a	d	g	j	o	p	t	w
"y	i"	q	c	d	g	j	n	p	t	w
"y	j"	q	a	d	g	j	m	p	t	w
"y	k"	q	a	d	g	j	m	p	t	w
"y	l"	q	a	e	g	j	m	p	t	w
"y	m"	q	a	e	g	j	o	p	t	w
"y	n"	q	c	d	g	j	m	p	t	x
"y	o"	q	a	d	g	j	n	r	u	w
"y	p"	q	a	e	i	j	m	p	t	w
"y	q"	q	a	d	g	j	m	p	t	w
"y	r"	q	a	d	i	j	m	p	t	w

FIG. 10R

"-	-"	1	2	3	4	5	6	7	8	9
"y	s"	q	c	e	i	j	m	p	t	w
"y	t"	q	a	e	h	j	m	p	t	w
"y	u"	q	a	d	g	k	m	r	t	w
"y	v"	q	a	d	g	j	m	p	t	w
"y	w"	q	a	d	h	j	m	p	t	w
"y	x"	q	a	d	g	j	m	p	t	w
"y	y"	q	a	d	g	j	m	p	t	w
"y	z"	q	a	e	g	j	m	p	t	w
"y	"	q	a	d	g	j	m	p	t	w
"z	a"	q	a	d	g	j	m	p	t	w
"z	b"	q	a	d	g	j	m	p	t	w
"z	c"	q	a	d	g	j	m	p	t	w
"z	d"	q	a	d	g	j	m	p	t	w
"z	e"	q	a	d	g	j	n	r	t	w
"z	f"	q	a	d	g	j	m	p	t	w
"z	g"	q	a	d	g	j	m	p	t	w
"z	h"	q	a	d	g	j	m	p	t	w
"z	i"	q	a	d	g	j	n	p	t	w
"z	j"	q	a	d	g	j	m	p	t	w
"z	k"	q	a	d	g	j	m	p	t	w
"z	l"	q	a	d	g	j	m	p	t	w
"z	m"	q	a	d	g	j	m	p	t	w
"z	n"	q	a	e	g	j	m	p	t	w
"z	o"	q	a	d	g	j	n	p	t	w
"z	p"	q	a	d	g	j	m	p	t	w
"z	q"	q	a	d	g	j	m	p	t	w
"z	r"	q	a	d	g	j	m	p	t	w
"z	s"	q	a	d	g	j	m	p	t	w
"z	t"	q	a	d	g	j	m	p	t	w
"z	u"	q	a	d	g	j	m	p	t	w
"z	v"	q	a	d	g	j	m	p	t	w
"z	w"	q	a	d	g	j	m	p	t	w
"z	x"	q	a	d	g	j	m	p	t	w
"z	y"	q	a	d	g	j	m	p	t	w
"z	z"	q	a	d	g	j	m	p	t	w
"z	"	q	a	d	g	j	m	p	t	w
"	a"	q	c	d	i	l	n	r	t	w

FIG. 10S

"_	_"	1	2	3	4	5	6	7	8	9
"	b"	q	a	e	i	l	o	r	u	y
"	c"	q	a	e	h	l	o	r	u	y
"	d"	q	a	e	i	j	o	r	u	w
"	e"	q	a	d	i	l	n	s	v	x
"	f"	q	a	e	i	l	o	r	u	w
"	g"	q	a	e	i	l	o	r	u	w
"	h"	q	a	e	i	j	o	p	t	w
"	i"	q	c	f	g	j	n	s	t	w
"	j"	q	a	e	g	j	o	p	u	w
"	k"	q	b	e	i	j	n	p	t	w
"	l"	q	a	e	i	l	o	p	u	y
"	m"	q	a	e	i	k	o	s	u	y
"	n"	q	a	e	i	j	o	p	u	w
"	o"	z	b	f	g	k	n	r	u	w
"	p"	q	a	e	h	l	o	r	u	w
"	q"	z	a	d	g	j	m	p	u	w
"	r"	q	a	e	i	j	o	s	u	w
"	s"	q	c	e	h	l	o	p	t	y
"	t"	q	a	e	h	j	o	r	u	w
"	u"	q	a	d	i	k	n	s	t	w
"	v"	q	a	e	i	l	o	p	t	w
"	w"	q	a	e	i	j	o	r	t	w
"	x"	q	a	d	g	j	m	p	t	y
"	y"	q	a	e	i	j	o	p	u	w
"	z"	q	a	e	i	j	o	p	t	y
"	"	q	a	f	i	l	o	s	t	w

FIG. 10T

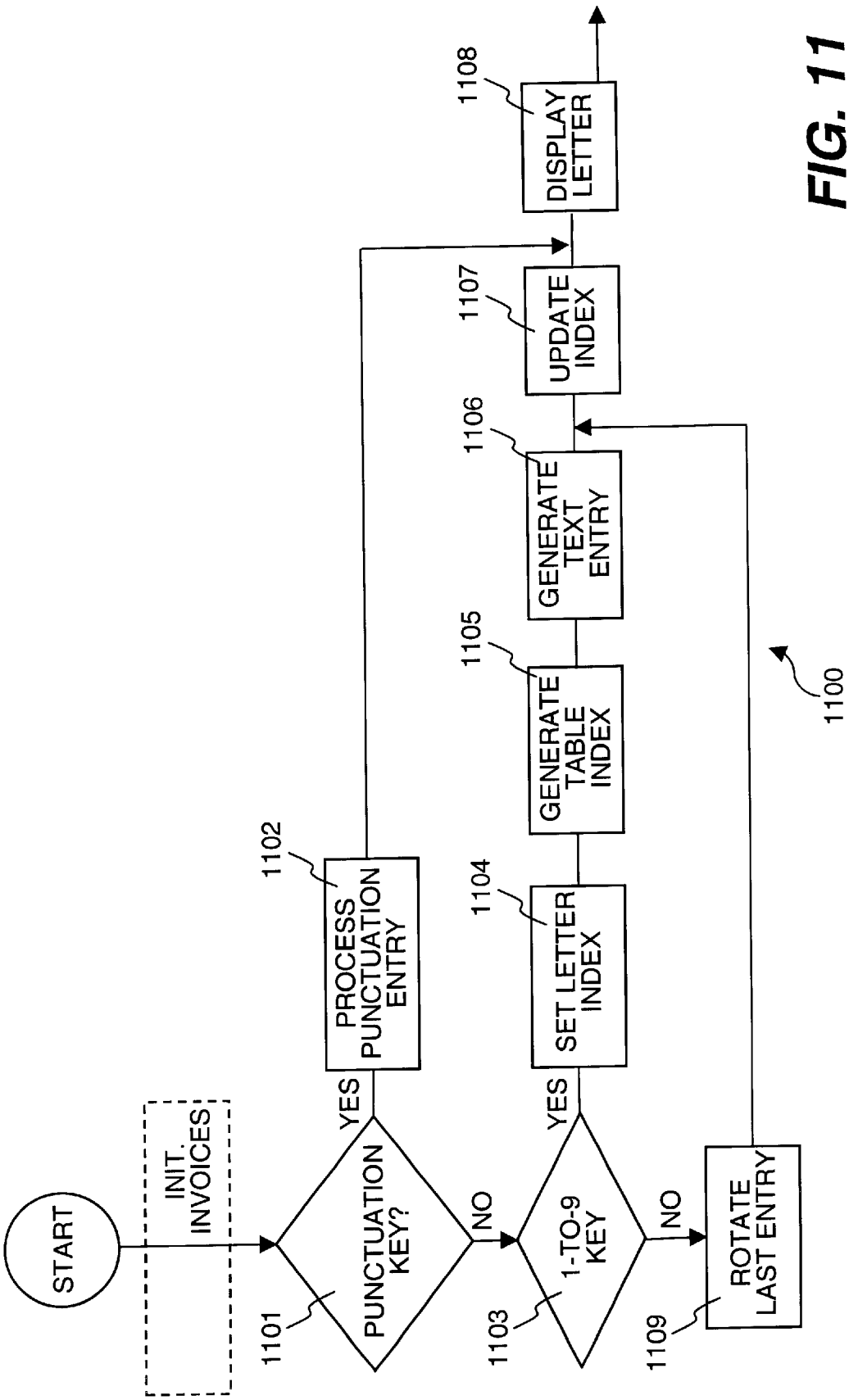

START
INIT. INVOICES
PUNCTUATION KEY?
1101
YES
NO
PROCESS PUNCTUATION ENTRY
1102
1-TO-9 KEY
1103
YES
NO
SET LETTER INDEX
1104
GENERATE TABLE INDEX
1105
GENERATE TEXT ENTRY
1106
UPDATE INDEX
1107
DISPLAY LETTER
1108
ROTATE LAST ENTRY
1109
1100

FIG. 11

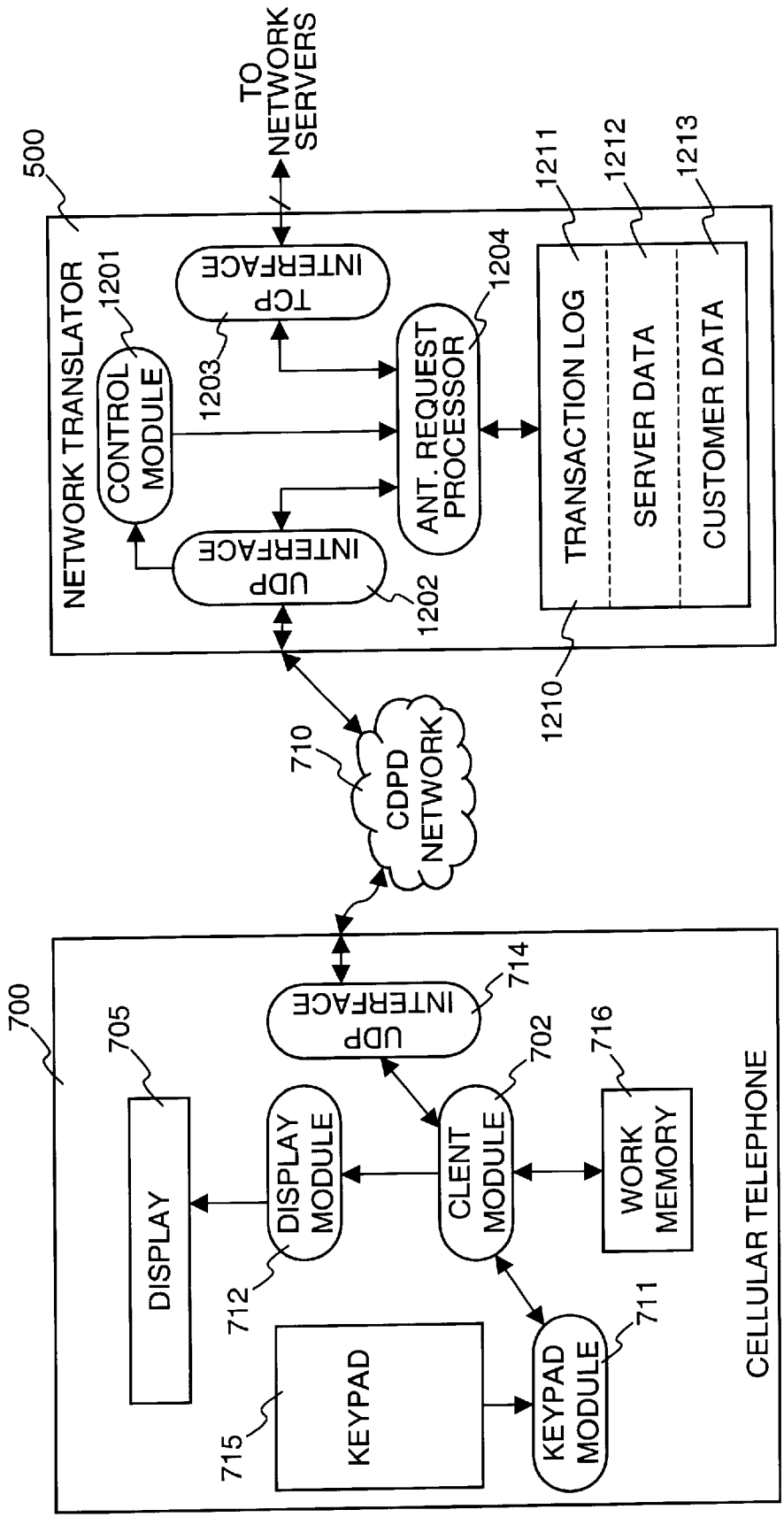

TO NETWORK SERVERS
500
NETWORK TRANSLATOR
1201
CONTROL MODULE
1203
TCP INTERFACE
1204
ANT. REQUEST PROCESSOR
1211
1212
1213
TRANSACTION LOG
SERVER DATA
CUSTOMER DATA
UDP INTERFACE
1202
1210
710
CDPD NETWORK
700
714
UDP INTERFACE
705
DISPLAY
DISPLAY MODULE
712
CLENT MODULE
702
716
WORK MEMORY
KEYPAD
715
KEYPAD MODULE
711
CELLULAR TELEPHONE

FIG. 12

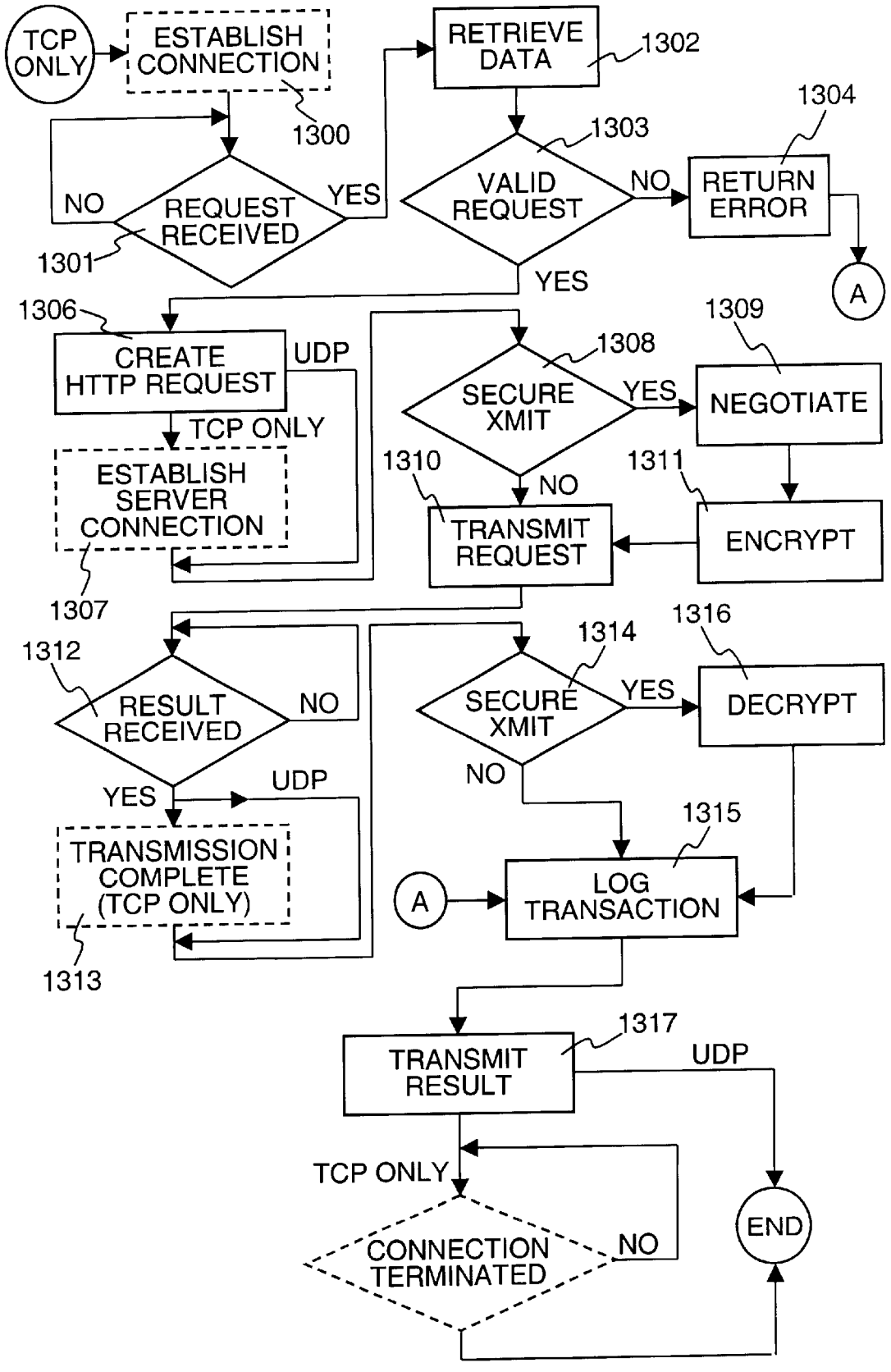

TCP ONLY
ESTABLISH CONNECTION
1300
RETRIEVE DATA
1302
1304
1303
VALID REQUEST
NO
RETURN ERROR
A
NO
REQUEST RECEIVED
YES
1301
YES
1306
CREATE HTTP REQUEST
UDP
TCP ONLY
ESTABLISH SERVER CONNECTION
1307
1309
1308
SECURE XMIT
YES
NEGOTIATE
1310
NO
1311
TRANSMIT REQUEST
ENCRYPT
1312
RESULT RECEIVED
NO
1314
1316
SECURE XMIT
YES
DECRYPT
YES
UDP
NO
TRANSMISSION COMPLETE (TCP ONLY)
1313
1315
A
LOG TRANSACTION
1317
TRANSMIT RESULT
UDP
TCP ONLY
CONNECTION TERMINATED
NO
END

FIG. 13

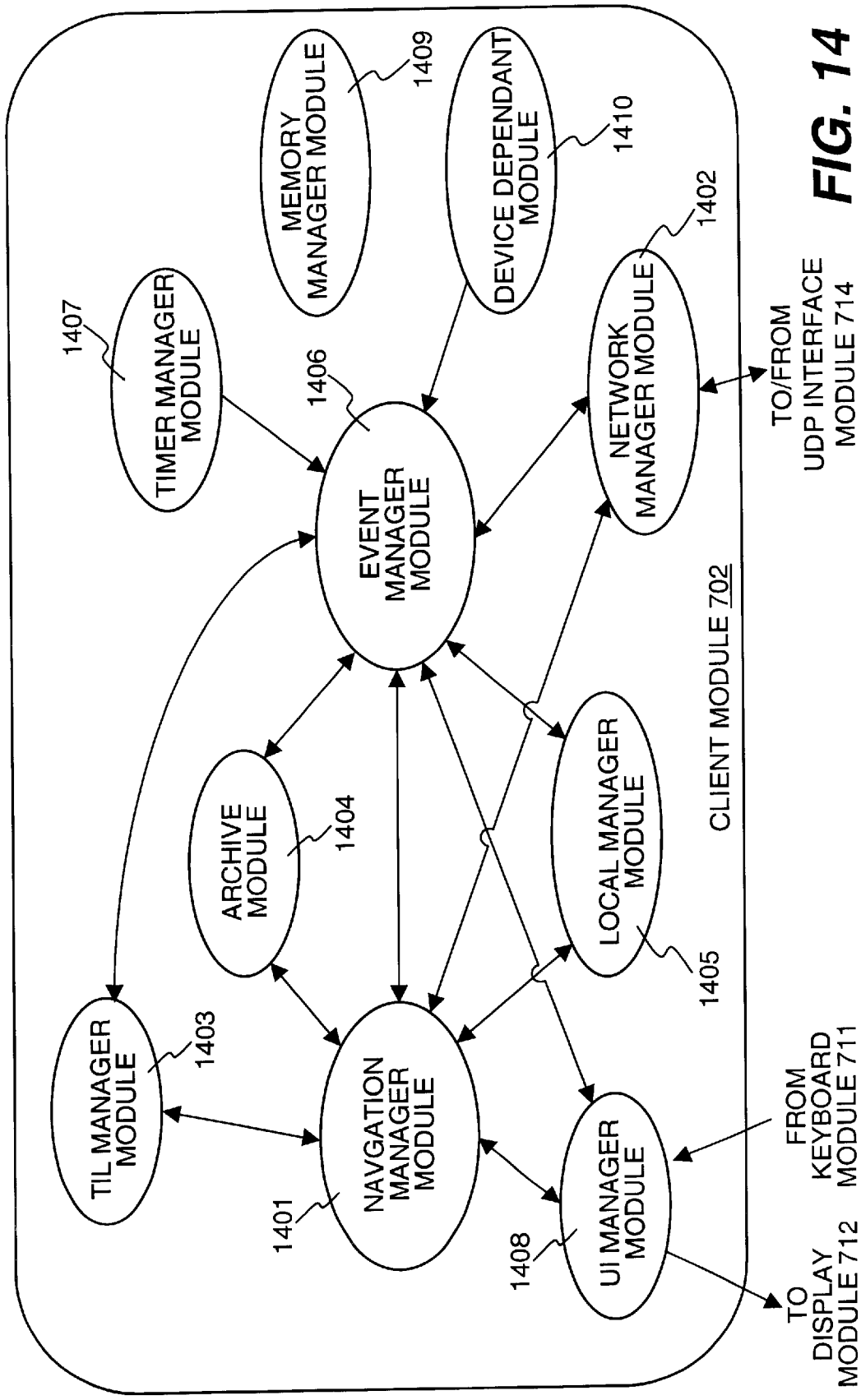

CLIENT MODULE 702
TIL MANAGER MODULE
1403
NAVGATION MANAGER MODULE
1401
ARCHIVE MODULE
1404
EVENT MANAGER MODULE
1406
TIMER MANAGER MODULE
1407
MEMORY MANAGER MODULE
1409
DEVICE DEPENDANT MODULE
1410
NETWORK MANAGER MODULE
1402
LOCAL MANAGER MODULE
1405
UI MANAGER MODULE
1408
TO/FROM UDP INTERFACE MODULE 714
FROM KEYBOARD MODULE 711
TO DISPLAY MODULE 712

FIG. 14

METHOD AND ARCHITECTURE FOR AN INTERACTIVE TWO-WAY DATA COMMUNICATION NETWORK

CROSS REFERENCE TO MICROFICHE APPENDIX

The present application is a continuation application of U.S. patent application ("Parent Application"), Ser. No. 08/978,701, filed on, Nov. 6, 1997, entitled "A Method and Architecture for an Interactive Two-way Data Communication Network," assigned to Openwave Systems Inc., which is also the Assignee of the present application. The Parent Application is a continuation application of U.S. patent application, Ser. No. 08/570,210, filed on, Dec. 11, 1995, now U.S. Pat. No. 5,809,415 entitled "A Method and Architecture for an Interactive Two-way Data Communication Network," also assigned to Openwave Systems Inc.

Appendix A, which is a part of the present disclosure, is a microfiche appendix consisting of six sheets of microfiche having a total of 369 frames. Microfiche Appendix A is a listing of one embodiment of the client module of this invention, which is described more completely below, and a server, as described more completely below, to communicate and interact with the client module of this invention.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to data communications, and in particular to two-way data communication devices including a cellular telephone, a two-way pager, and a telephone that permit a user to interface with and interact with a server on a computer network.

2. Description of Related Art

For at least the last five years, the wireless communication industry has tried to merge computing with wireless communications. This industry wide effort has held the promise of bringing software intelligence to telecommunication devices including mobile wireless communications devices such as cellular telephones and two-way pagers as well as standard telephones.

After years of research and development, and hundreds of millions of dollars' investment by some of the largest companies in the field such as Motorola, AT&T, Sony, Matsushita, Phillips and IBM, the results have been nothing but disappointing. Typically, the intelligent communication devices resulting from these efforts include both the hardware necessary for a computer module and the hardware for a wireless communications module. Examples of such products are Simon from IBM and Bell South, MagicLink from Sony, and Envoy from Motorola.

Fundamental design and cost problems arising directly from the approach taken by the designers of these intelligent communication devices have limited widespread market acceptance of these devices. The combination of a wireless communication module with a computing module leads to a device that is too bulky, too expensive, and too inflexible to address the market requirements.

The combination of the two modules is too large and too heavy to fit in a user's pocket. Pocket size is a key requirement of the mobile communication market which remains unmet by these devices

In addition, the cost of these devices is close to the sum of the cost of the computer module and of the communications module, which is around a one thousand dollar end-user price. Market research indicates that the market for intelligent wireless communications devices is at prices around $300. Even with a 20% compound cost decline, it would take five years for the combination units to meet today's customers' price requirements. It is therefore unlikely that devices designed by combining a computer and a wireless module, no matter how miniaturized and cost reduced, can satisfy the cost requirement of the market during this decade.

To succeed in the market place, intelligent wireless communication devices must be able to support a wide variety of applications specific to each market segment. Typically, these applications must be added to the device by the end-user after purchase. Thus, the device must provide a method for loading the initial application and for subsequent updating of the application.

The price sensitivity for intelligent communication devices and the size limitations means that an intelligent communication device cannot support the amount of core memory (RAM), a hard disk or non-erasable memory, or a traditional floppy disk drive, commonly found on computers. These limitations close the traditional routes for delivering new applications or updates to intelligent communications devices.

As a result, the current crop of intelligent communication devices run only the few applications which were burned into their ROMs at the factory or which are contained in a ROM card plugged into a slot designed for this purpose. This scheme lacks the flexibility needed to run the thousands of applications required to address the fragmented requirements of the market and provides no simple method for updating the applications after the device has been sold.

Two other communication oriented attempts at bringing intelligence to telephones are Short Messaging Service (SMS) and Analog Display Service Interface (ADSI). SMS specifies how messages are delivered to and from a cellular telephone and how the cellular telephone should store the messages. SMS also defines some simple processing which the cellular telephone can perform on the message, such as calling a telephone number embedded in the message.

SMS's architecture is similar to that of paging networks with the difference that devices implementing the SMS architecture operate over the control channel of the cellular telephone network. SMS is deployed primarily in Europe over the GSM network.

SMS messages are not delivered in real time. The time delays can range from 30 seconds up to 10 minutes, which makes SMS unsuitable for real time applications. The main purpose of SMS is the delivery of messages. SMS does not specify an application protocol or cellular telephone application module which further restricts its usefulness in running applications on cellular telephones. After a few years of deployment in Europe, SMS implementations have been limited to notification services such as two-way paging and voice mail notification.

SMS as a medium is unsuited to building applications which allows the retrieval, manipulation, to and storage of information. This is the reason why the industry giants have not turned to SMS in their quest to add intelligence to

cellular telephones, but have consistently attempted to combine a computer module with a wireless communications module.

ADSI was designed as an extension to Interactive Voice Response Systems. ADSI allows a smart telephone with a small screen to display prompts to assist users in choosing among various options. By using visual prompts instead of cumbersome voice prompts, ADSI is thought to make the use of interactive voice services easier and faster.

ADSI allows data to be sent from the service provider to the telephone in the form of screens. ADSI also allows the telephone to respond through touch tone signaling with a special coding to describe the full alphanumeric character set. With ADSI, a telephone is primarily a passive device services send text screens to the telephone, and the telephone sends back short strings indicating the choices the user made from the text screen.

ADSI makes no provisions for performance of processing in the telephone. As a result, ADSI generates a high traffic load on the telephone network since each user input is sent back to the service for processing. This makes ADSI unsuitable for wireless networks where bandwidth is at a premium and "air efficiency" is one of the most sought after qualities. The lack of processing capability in the telephone and the high bandwidth requirements of ADSI have prevented it from being considered by the industry for implementing intelligent wireless devices.

Up to now, intelligent communication devices have combined a computing module with a wireless communications module. However, to gain widespread acceptance, a two-way data communication device with processing capability and the ability to run a wide variety of differing user applications is needed. In addition, such a device should be comparable in size, cost, and weight to a cellular telephone.

SUMMARY OF THE INVENTION

According to the principles of this invention, the prior art limitations of combining a computer module with a wireless communication module have been overcome. In particular, a two-way data communication device of this invention, such as a cellular telephone, two-way pager, or telephone includes a client module that communicates with a server computer over a two-way data communication network. The principles of this invention can be used with a wide variety of two-way data communication networks. For example, two-way data communication networks for cellular telephones that may be used include a cellular digital packet data network as well as TDMA, CDMA, and GSM circuit switched data networks; and the AMPS analog cellular network with a modem. Similarly, for two-way pagers, two-way data communication networks include PACT, the new AT&T endorsed two way paging standard, or other priority two-way paging networks with data transport capability. The two-way data communication network for a telephone is the public switched telephone network.

Using the two-way communication device that includes the client module, a user can provide information to the server computer, retrieve information from the server computer, provide data to an application on the server computer which uses the data and provides information to the two-way communication device, or sends the information to another location. The functionally provided to the user of the two-way communication device is limited only by the applications available on a server computer that is accessible to the user over the two-way data communication network.

This invention allows for the first time two-way communications devices such as cellular telephones, two-way pagers, and telephones to become open application platforms which in turn empowers software developers to deliver value-added applications and services to any two-way communication device that incorporates the principles of this invention. This is a radical shift from the current situation where telephones and two-way pagers are closed, proprietary systems. Consequently, an even playing field is created for the market to invent new uses for two-way communication devices and for two-way communication networks. Any entity from corporations to individuals can make new applications available to the installed base of two-way data communication devices that include this invention without physical modification or addition to the two-way communication device. Years after purchase, a two-way communication device incorporating this invention will run all the applications which were developed since its purchase.

Further, all these applications are available without the end user having to add anything or make any modification to the two-way communication device. Also, the applications are independent of the two-way data communication network. The applications do not depend on any feature of the two-way data communication network. Thus, the applications are unaffected by a change in the two-way data communication network.

Also, the applications on the server computer are independent of the two-way data communication device with which the server computer is interacting. An application on the server computer can communicate with any two-way data communication device that includes the client module of this invention and a network interface module to transmit data over, and receive data from the two-way data communication network. These two features mean that an investment in developing an application is insulated from either advances in two-way data communication devices, or advances in two-way data communication network technology.

As indicated above, the two-way data communication device of this invention utilizes a client module to transmit a message including a resource locator selected by the user over the two-way data communication network to a server on a server computer on the computer network. For example, the computer network can be a corporate wide area network, a corporate local area network, the Internet, or any combination of computer networks.

The server processes the message, i.e, executes the application addressed by the resource locator and transmits a response over the two-way data communication network to the two-way data communication device, which stores the response in a memory. The client module interprets the response and generates a user interface using information in the response. In one embodiment, the user interface includes at least one user data input option that is associated with a resource locator. In another embodiment, the user interface is a display.

The resource locator associated with the at least one user data input option can address any one of a wide variety of objects. In one embodiment, the resource locator associated with the at least one user data input option addresses an object on the server computer that transmitted the response. In another embodiment, the resource locator addresses an object on another server computer coupled to the two-way data communication network. In yet another embodiment the resource locator addresses an object stored in the two-way communication device.

When the user selects the at least one user data input option, the client module interprets the selection and if required, appends any input data to the resource allocator associated with the at least one user data input option. The client module transmits a message including the resource locator with any appended input data to the server computer.

Alternatively, the resource locator with any appended data can be addressed to another server computer, or can address an object stored in the two-way communication device. If the resource locator addresses an object on a server computer, the client module provides the message to the network interface module which in turn transmits the message over the two-way data communication network.

Thus, in this embodiment, the message originally transmitted to the two-way data communication device included all the information necessary for the client module to generate the user interface, to associate the user selection and any data entered with a particular resource locator, and to transmit the appropriate resource locator in a subsequent message. The client module includes an interpreter that processed the information in the message. Since the message included all the information needed by the client module, the server computer that transmitted the message retained no state information concerning the message. Consequently, the server computer is defined as a stateless server computer.

An important aspect of this invention is that the message includes all information necessary for the client module to generate the user interface and a particular user interface can be independent from other user interfaces. Unlike prior art systems that gave the user a predetermined menu from which to select items, or limited the user to an E-mail like format, according to the principles of this invention, the user interfaces and possible interactions available to the user are determined only by the applications that developers make available. The possible interactions and user interfaces for one application can be totally different and independent from the possible interactions and user interfaces of another application. Thus, a cellular telephone, two-way pager, and a telephone all truly become an open platform.

These features of the invention are a significant departure from prior art systems. Typically, in the prior art, use of a particular application on a particular platform required that the application be compatible with the operating system on that platform. Further, each time a new version of the application was released, the user was required to take steps to update the application on the user's platform. Further, if the user of the platform did not modify the operating system as new versions of the operating system were released, at some point in time, the platform would no longer be capable of processing a new version of an application that required a current version of the operating system.

This invention eliminates these problems. As explained above, the client module in the two-way data communication device functions an interpreter. The application on the server computer provides all information necessary for the interpreter to generate a user interface on the two-way data communication device, and in response to user selections or data input using the user interface, to route messages to an appropriate server, i.e, either the server that sent the original information or another server.

Thus, the client module only interprets this information and interacts appropriately with the hardware of the two-way data communication device. Consequently, to update an application requires only changes on the server computer and not changes in each two-way data communication device that communicates with that server computer. This invention eliminates the usual requirement for distribution of application software, and application software updates to the end user of the two-way data communication device.

In one embodiment, a two-way data communication system for communication between a server computer and a two-way data communication device selected from a group consisting of a cellular telephone, a two-way pager, and a telephone, includes a two-way data communication network, a server computer coupled to the two-way data communication network, and a two-way data communication device coupled to the two-way data communication network. The server computer includes a two-way data communication interface module coupled to the two-way data communication network, and a server coupled to the two-way data communication interface module. The server receives a message including a resource locator from the two-way data communication network. The resource locator includes an address of the server computer and of an application on that server computer. The server processes the message using the resource locator. In this embodiment, the server transmits a response to the message over the two-way data communication network.

The two-way data communication device, selected from the group consisting of a cellular telephone, a two-way pager, and a telephone, includes a network interface module coupled to the two-way data communication network, and a client module coupled to the network interface module. The client module transmitted the message including the resource locator to the server over the two-way data communication network. The client module also processes the response to the message from the server. The response includes information for a user interaction over the two-way data communication network.

The client module of this invention is lightweight, and thus requires only lightweight resources in a two-way data communication device. Consequently, the client module can use existing resources in such a device and therefore does not add to the cost of the two-way data communication device.

In one embodiment, the interpreter within the client module includes a plurality of managers including a user interface manager coupled to a display of the two-way data communication device where the user interface manager handles interactions with the display. The user interface manager also is coupled to a keypad of the two-way data communication device and handle interactions with the keypad. Herein, a keypad can be a telephone keypad, the keys found on a two-way pager, or other data input interface of a two-way communication device.

In one embodiment, the response generated by the server computer includes a plurality of resource locators and at least one of the plurality of resource locators includes an address to another server coupled to the communication network.

According to the principles of this invention, a method for using a two-way data communication device, selected from a group consisting of a cellular telephone, a two-way pager, and a telephone, to communicate with a server computer includes:

generating a message by a client module in response to data entered by the user of a two-way data communication device coupled to a two-way data communication network,

wherein the client module executes on a microcontroller of the two-way data communication device; and the message includes a re source locator;

transmitting the message over the two-way data communication network to a server computer wherein the server computer is identified by the resource locator;

executing an application on the server computer identified by the resource locator to generate a response to the message; and

transmitting the response to a location identified by the application.

As indicated above the location can be the two-way communication device, another server computer, or some other device coupled to the server computer.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates one embodiment of the airnet network of this invention that includes the two-way data communication devices of this invention.

FIGS. **2**A to **2**H are illustrations of a series of screen displays of the two-way data communication device of this invention that illustrate one application of the principles of this invention.

FIGS. **3**A to **3**F are illustrations of a series of screen displays of the two-way data communication device of this invention that illustrate a second application of the principles of this invention.

FIGS. **4**A to **4**I are illustrations of a series of screen displays of the two-way data communication device of this invention that illustrate yet another application of the principles of this invention.

FIG. **5** illustrates another embodiment of the airnet network of this invention that includes the two-way data communication devices of this invention and an airnet network translator.

FIG. **6** is a block diagram of a mobile wireless communication device that includes the client and support modules of this invention.

FIG. **7** is a more detailed diagram of the mobile wireless communication device and a server computer within the airnet network architecture of this invention.

FIGS. **8**A to **8**D are a process flow diagram showing the process performed by the client in the mobile wireless communication device and the server on the server computer of FIG. **7**.

FIG. **9** is a diagram of a mobile wireless communication device of this invention that includes a novel predictive text entry system that is a part of this invention.

FIGS. **10**A to **10**T are one embodiment of a letter frequency table.

FIG. **11** is a process flow diagram for one embodiment of a data entry process that includes the novel predictive data entry process of this invention,

FIG. **12** is a more detailed diagram of the mobile wireless communication device and the airnet network translator within the airnet network architecture of the another embodiment of this is invention.

FIG. **13** is a process flow diagram showing the various processes performed by the airnet network translator of FIG. **12**.

FIG. **14** is a diagram illustrating the various module managers included in one embodiment of the client module of this invention.

Herein, objects with the same reference numeral are the same object. Also, the first number of a reference numeral indicates the Figure where the object first appeared.

DETAILED DESCRIPTION

According to the principles of this invention, a novel airnet network **150**, i.e., a two-way data communication network, interconnects any one, any combination, or all of two-way data communication devices **100**, **101**, or **102**, that each include this invention, with a wide variety of computer networks **120**, **130**, and **140**, for example. As explained more completely below, each two-way data communication device **100**, **101**, and **102** can be configured to transmit data to and receive data from any desired combination of computers on computer networks **120**, **130**, and **140**. Airnet network **150** is the two-way data communication path from the two-way data communication device to the particular computer that is accessed by the user of that two-way data communication device.

Each wireless communication device **100** that includes this invention can communicate over airnet network **150** with any server computer **121**, **131**, and **141** on airnet network **150** that includes at least one application that communicates and interacts with the processes of this invention that are included within device **100**. Thus, device **100** can access information on the computer network and provide information to the computer network. Similarly, a two-way pager **101**, and a telephone **102** with a modem **103**, that each include this invention, can communicate over airnet network **150** with any of server computers **121**, **131**, and **141** that includes at least one application that communicates and interacts with the processes of this invention that are included within devices **101** and **102**.

As explained more completely below, an application on a server computer can be accessed by any two-way data communication device that can communicate with that server computer. The application is independent of the particular type of two-way data communication device that is used to access the application and independent of the particular two-way data communication network used. This means that a user can access an application from anywhere so long as the user has a two-way data communication device that can communicate with the server computer.

In one embodiment, a process on wireless communication device **100** is configured as a client process and the applications on server computers **121**, **131** and **141** on airnet network **150**, that communicate with the client process, are server processes. This architecture allows some of the processing burden to be moved away from cellular telephone **100**, across airnet network **150**, to a server module on any computer on airnet network **150**.

Specifically, a wireless communication device **100** e.g., a cellular telephone, with a telephone like keypad, communicates via a data capable cellular telephone network **110**, e.g., a cellular digital packet data telephone network, with an application on a server computer on a computer network that has an interface to data capable cellular telephone network **110**. For example, the computer network can be a corporate wide area network **120**, a corporate local area network **130**, or perhaps the Internet **140**.

Similarly, a two-way pager **101** communicates via a two-way pager network **111** with an application on a server computer on a computer network that has an interface to two-way pager network **111**. Again, for example, the computer network can be a corporate wide area network **120**, a corporate local area network **130**, or perhaps the Internet **140**. Finally, a telephone **102** communicates via a modem **103** and public switched telephone network **112** with an application on a server computer on a computer network that has an interface to public switched telephone network **112**.

As with the other two-way data communication devices, the computer network can be, for example, a corporate wide area network **120**, a corporate local area network **130**, or perhaps the Internet **140**.

In each of two-way data communication devices **100**, **101**, and **102**, the client process is stored as a client module in the device and the execution of the client module on a microcontroller in the device is sometimes referred to as the client process. The client process performs important processing functions locally. This allows the communication between the client process, hereinafter sometimes referred to as simply client, and the server process, hereinafter sometimes referred to as server, to be minimized and the server computing requirements to grow slowly as the number of clients, i.e., users, grows.

The client module is small, e.g., under 64 KByte, and requires only low processing power congruent with the memory chips and built-in microcontrollers in two-way data communication devices such as cellular telephone **100**, two-way pager **101**, and telephone **102**. Thus, unlike the prior art attempts at an intelligent telephone, the cost, size, and battery life of either cellular telephones, two-way pagers, or telephones that incorporate this invention are not adversely affected.

While client/server architectures have been used extensively in computer networks, a client/sever architecture implements using two-way communication data devices such as cellular telephone **100**, two-way pager **101**, or telephone **102** yields new and unexpected results. This invention allows for the first time a wide variety of two-way data communication devices including but not limited to cellular telephones, two-way pagers, and telephones to become open application platforms which in turn empowers software developers to deliver value added applications and service to any two-way data communication device which incorporates the principles to invention.

This is a radical shift from the current situation where cellular telephones, two-way pagers, and telephones are closed, proprietary systems. Consequently, an even playing field is created for the market to invent new uses for cellular telephones and two-way pager networks, and for telephones on the public switched network.

Any entity from corporations to individuals can make new applications available to the installed base of data ready cellular telephones, two-way pager, and telephones, that include this invention without physical modification or addition to the devices. Years after purchase, a two-way data communication device with this invention can run all the applications which were developed since its purchase. Further, all these applications are available without the user having to add anything or make any modification to the two-way data communication device. These features of the invention are a significant departure from prior art systems. Typically, in the prior art, use of a particular application on a particular platform requited that the application be compatible with the operating system on that platform. Further, each time a new version of the application was released, the user was required to take steps to update the application on the user's platform. Further, if the user of the platform did not modify the operating system as new versions of the operating system were released, at some point in time, the platform would no longer be capable of processing a new version of an application that required a current version of the operating system.

Also, small devices, such as cellular telephones or pagers, usually do not have card slots, floppy or hard disk drives, or other means commonly found on computers to add or update applications. This limitation has led prior art attempts at intelligent communication devices to design closed systems with fixed functionality. Such devices can neither adapt nor be adapted to the fast changing requirements of the market place and so have not met with market success.

This invention eliminates these problems. The client process in the two-way data communication device functions an interpreter. The application on the server computer provides all information necessary for the interpreter to generate a user interface on the two-way data communication device, and in response to user selections or data input using the user interface, to route messages to an appropriate server, i.e, either the server that sent the original information or another server.

Thus, the client process only interprets this information and interacts appropriately with the hardware of the two-way data communication device. Consequently, to update an application requires only changes on the server computer and not changes in each two-way data communication device that communicates with that server computer. This invention eliminates the usual requirement for distribution of application software, and application software updates to the end user of the two-way data communication device.

For example, if initially, two-way pager **101** receives a response to a message from an application on server computer **121** on corporate wide area network **120**, the interpreter in two-way pager **101** generates a user interface on display screen **106** using information in the message. As described more completely below, options presented in the user interface can allow the user to access information, or provide information to any one, any combination of, or all of networks **120**, **130**, and **140**.

Specifically, in the response to the message from two-way pager **101**, the application initially accessed on server computer **121** included resource locators for applications on each of networks **120**, **130**, **140**, typically common gateway interface programs, accessible to the user of pager **101** as well as information required to generate the user interface. Consequently, when the user makes a particular selection or enters data, the interpreter accesses the appropriate resource locator and appends any necessary data to the resource locator. The client transmits a message including the resource locator to the appropriate server.

As shown by this example, the applications on networks **120**, **130**, **140** send to the two-way data communication device all information necessary to generate a user interface, and to process all user input. Consequently, only an application must be changed to update the information provided to the two-way data communication device.

In addition, since all the information needed by the client to generate a user interface and all information necessary for the client process to respond to any input data is included in the message, the computer server does not retain any state information concerning the information transmitted to the client process. Consequently, the computer server is stateless.

Each two-way data communication device **100**, **101**, and **102** that utilizes airnet network **150**, includes a data communication capability, a display screen, preferably a multiline display screen, and storage capability for the processes of this invention in an on-board memory, and for the message being processed. Nearly every data capable cellular telephone, e.g., a telephone that utilizes a cellular digital packet data network, includes excess on-board memory capacity and a multi-line display screen. These hardware

resources are often available, but unused in a data capable cellular telephone because of the indivisibility of memory chip packages. The inclusion of the processes of this invention in such cellular telephones therefore has very little effect on the cost, size, and power consumption of the cellular telephone. Similarly, the inclusion of the processes of this invention in two-way pagers and telephones, that include a microcontroller and memory, has very little affect on the cost, size, and power consumption of these devices.

Thus, unlike prior art approaches that attempted to combine a computer module and a wireless communication module in a single package, this embodiment of the invention preferably utilizes the memory and processing power that currently exists in the cellular telephone **100**, two-way pager **101**, telephone **102** or other wireless or landline two-way data communication devices. This approach limits the cost of the resulting device and overcomes many of the problems of the prior art devices, e.g., the size and weight of the two-way data communication device is not changed, and, as explained above, updating user applications is removed from cellular telephone **100**, two-way pager **101**, and telephone **102**.

In particular, unlike devices produced by previous industry attempts at combining computing modules and a wireless cellular module, two-way data communication devices which incorporate this invention are size and cost competitive with voice-only telephones and can, for the first time, satisfy the market cost and size requirements for an intelligent cellular telephone, for example.

The incremental cost of supporting interactive applications on cellular telephone **100**, two-way pager **101**, and telephone **102** is reduced to at most a slightly larger screen that is required to display the application to the user. This is a fraction of the cost of adding a complete computer module to a cellular telephone, for example.

The incremental power consumption required to support this invention is also very small, as the incremental memory and screen required are small consumers of power compared to the cellular radio itself. Intelligent two-way data communication devices built according to the principles of this invention are not expected to have a significantly lower battery life than standard cellular telephones, or two-way pagers, for example.

The configuration and processes of the client process in two-way data communication devices **100**, **101**, and **102** are similar when the differences in the devices and the two-way data communication network over which the devices communicate are considered. Consequently, in the following description, the operation of data-ready cellular telephone **100** is considered. The same or similar operations can be performed on two-way data communication devices **101**, and **102**. The main difference is that some device dependent features within the client module must be changed to accommodate the particular hardware used in the two-way communication device. However, the client module is architecture described more completely below limits the number of changes that must be made.

As indicated above, in response to user actions, wireless communication device **100** transmits a message, typically a data request, to a server computer **121** on computer network **120** and receives a response to the message. Alternatively, the user action can result in directions to server computer **121** on computer network **120** to transmit the response to the message to another location or to another user. Also, wireless communication device **100** can receive a message from any one of the computers coupled to airnet network **150**.

An important aspect of this invention is that the client module interpreter in wireless communication device **100** generates a user interface by which the user can both initiate and receive messages from a variety of applications. The interactions take place in real-time and are not limited by the client module interpreter. The uses of wireless communication device **100** are limited only by the availability of applications on server computers.

The applications available are determined by application developers. Prior to considering one implementation of the invention in further detail, several illustrative examples of applications that can be implemented according to the principles of this invention are described. These applications are illustrative only and are not intended to limit the invention to the particular applications and features described.

In one use, the user configures cellular telephone **100** to access server computer **121** on XYZ corporate wide area network **120**. In response to the access by the user, server computer **121** transmits a card deck to cellular telephone **100** over data capable cellular telephone network **110**. As explained more completely below, a card deck includes one or more cards, and each card is interpreted by the client module to generate a user interface screen.

In the embodiment illustrated in FIG. **2**A, the initial card deck transmitted to cellular telephone **100** includes an introductory display card and a choice card. FIG. **2**A is an example of introductory screen display **200** that is generated on display screen **105** by the client process in cellular telephone **100** by interpreting the display card. As used herein, a display screen is the physical display apparatus in a two-way communication device. A screen display is the image presented on the display screen.

In this embodiment, display screen **105** is a pixel display that displays graphics. In another embodiment, display screen **105** displays only text and so the graphics would not appear on display screen **105**. Screen display **200**, and other screen displays described more completely below, include a horizontal arrow, i.e., a multi-card deck indicator, to communicate to the user that the current deck includes another card. The inclusion of screen indicators, such as the multi-card deck indicator, to communicate with the user is optional. The functionality of this invention is independent of such screen indicators.

When the user presses a predetermined key, or key sequence, the client process in cellular telephone **100** interprets the next card in the card deck, i.e., the choice card, and in turn generates a menu **201** (FIG. **2**B) of items that can be accessed by the user. In this embodiment, each of the menu items is available on server computer **121** to the user who, in this example, is a representative of XYZ corporation visiting ABC Designs.

As explained more completely below, each of the menu items is associated with a resource locator that includes an address of the particular object associated with that menu item, typically an address to a common gateway interface program on server computer **121**. In general, a resource locator includes an address and may include appended data. The address can be to a local object within the two-way data communication device or to a remote object on a server computer. As is known to those skilled in the art, the common gateway interface is an Internet standard that is used to dynamically generate information, e.g., cards. In view of this disclosure, other techniques to generate dynamic cards could be used.

Initially, the highlighting of the first line of menu **201** is not present. When a key on the keypad of cellular telephone

100 is pressed, the menu item corresponding to that key is highlighted on screen 105. Thus, menu 201 shows the first item highlighted to indicate that the one key was pressed by the user. However, highlighting a selected item is a feature that is specific to this example, and in general is not required to implement the invention. Other methods can be used to indicate the user's choice on display screen 105 such as an arrow pointing at the choice, if such an indication is desired.

After the one key is pressed, the user presses a predetermined key, e.g., an enter key, to verify the selection. Alternatively, in another embodiment, the verification of the selection is not required. In both embodiments, the resource locator for the selection is transmitted to server computer 121 by the client process in cellular telephone 100 over data capable cellular telephone network 110. In response to the selection, server computer 121 processes the message containing the selection, and in this embodiment, transmits another card deck to cellular telephone 100.

The client process in cellular telephone 100 interprets the first card in the deck received from server computer 121, which is a choice card, and generates a screen display 202, that includes a second menu as illustrated in FIG. 2C, on display screen 105. Initially, none of the items in the second menu are highlighted.

Notice that screen display 202 includes a header, that describes the selection made by the user on screen display 201, in addition to the second menu of choices available to the user. A multi-display screen card indicator 203, e.g., in this embodiment, a hand icon with a finger pointing down, shows that the screen associated with the current choice card includes additional items that are not shown on display screen 105. Herein, a screen can be larger than the number of lines available on display screen 105 and so the user must scroll the screen display to view the complete screen.

Thus, to view the additional items, the user presses a first screen scroll key, e.g., a next key, on cellular telephone 100. In this embodiment, when the first screen scroll key is pressed, each line of the display is rolled up one line. The resulting display has an icon with a finger pointing up (not shown) if the menu requires only two screen displays. If the menu requires more than two screen displays, the second screen display of the menu would have two icons, one with a finger pointing up, and another with a finger pointing down. To scroll between the various lines in the second menu, the user uses the first screen scroll key, and a second screen scroll key.

If the user displays the last line of a card, e.g., the last line in the second menu, and presses the first screen scroll key nothing happens. In this embodiment, the user must make a choice before the next card is available.

Screen display 202 also includes representations of two soft keys, a home key 204, and an info key 205. In this example, these soft keys are defined only for the card used to generate screen display 202. When the user presses a predetermined key sequence, the home key is highlighted to indicate the selection. In this embodiment, when the home key is selected, the user is returned to screen display 200. In another embodiment, the user could be returned, for example, to a home screen display that is displayed each time the user activates cellular telephone 100 for use on airnet network 150.

The home key is associated with a pointer, that in one embodiment is a resource locator, and the card addressed by the pointer is displayed by the client process when the home key is selected by the user.. Specifically, if the pointer is to a card in the current deck, the client process simply displays that card. If the pointer is to other than a card in the current deck, the client process in cellular telephone 100 retrieves the deck containing the card at the location identified by the pointer. The location could be, for example, either a memory in cellular telephone 100, or a memory in computer 121.

Similarly, when the user presses another predetermined key sequence, the info key is highlighted to indicate the selection. In this embodiment, when the info key is selected, a help screen is displayed for the user that describes the possible selections. The particular contents of the help screen are determined by the provider of the service. Specifically, a pointer is associated with the info key and when the info key is depressed by the user, the information stored at the location identified by the pointer is retrieved and interpreted by the client process in cellular telephone 100.

Returning to the menu in FIG. 2C, since the user wants to determine the status of an order, the user pushes the two key on the keypad of cellular telephone 100. In response to the key press, the second choice in the menu is highlighted as shown in FIG. 2C. In response to verification of the key press, e.g., the user presses a predetermined key sequence, cellular telephone 100 transmits a check open order request to computer 121, i.e, the client process transmits a message that includes a resource locator associated with the menu item selected by pressing the two key.

In response to the check open order request, computer 121 transmits yet another card deck to cellular telephone 100. The client process in cellular telephone 100 interprets this deck, that is an entry card, and in turn generates a purchase order number entry screen display 206 (FIG. 2D) on display screen 105. Notice that screen display 206 has a previous soft key 207 and a fax soft key 208. Again, each of these soft keys has an associated pointer and the information stored at the location identified by the pointer is retrieved and interpreted by the client process when the user selects the soft key.

In this example, the user does not select a soft key, but rather the user enters the purchase order number as shown in FIG. 2E using the keypad of cellular telephone 100. The user enters only the various numbers. The client process formats the number and inserts the dashes as shown in FIG. 2E.

After the purchase order is entered, the user presses a predetermined key sequence to indicate to the client process that entry of the purchase order number is complete. Notice that the user is entering data and not simply selecting a menu item. The user is utilizing cellular telephone 100 as if cellular telephone 100 was a computer connected to network 120, but, as explained more completely below, cellular telephone 100 is similar to a standard digital data capable cellular telephone that communicates over data capable cellular telephone network 110. Specifically, cellular telephone 100 is not a combination of a computer module and a wireless communication module as in prior art attempts to create an intelligent telephone.

In addition, the user enters data using only the standard cellular telephone keypad. Thus, cellular telephone 100 eliminates the need for a computer keyboard or for a sophisticated touch screen that recognizes motion of a pointing object. This is important to maintaining the size, weight, and power requirements of cellular telephone 100 similar to those of a voice-only cellular telephone. In one embodiment, to facilitate data entry, as explained more completely below, cellular telephone 100 includes a text prediction process that reduces the number of key strokes required to enter text data. In this embodiment, the text prediction process is turned on or off for each entry card.

In response to entry of the purchase order number, the client process transmits a request to server computer **121** for the particular purchase order. Specifically, the client process appends the entered data to a resource locator and transmits a message containing the resource locator to server computer **121**. Server computer **121**, in response to the message, retrieves the appropriate purchase order and transmits the purchase order as a card deck to the client process in cellular telephone **100** over airnet network **150**.

The client process interprets the card deck and generates a screen display **209** (FIG. **2**F). Initially, fax key **208** is not highlighted in screen display **209**.

Notice that screen display **209** includes multi-display screen card indicator **203** to show the user that the purchase order screen contains more information that can be displayed at one time on display screen **105**.

After the user reviews the purchase order, the user presses the key sequence for fax key **208** and in response, fax key **208** is highlighted as illustrated in FIG. **2**F.

In response to selection of fax key **208**, the client process retrieves the card deck at the location identified by the pointer associated with fax key **208**. If the location is on server computer **121**, the client process transmits a message including a resource locator to server computer **121** and in response to the message, server computer **121** transmits back yet another card deck. If the location is on a server computer other than server computer **121**, the client process transmits a message including a resource locator to that server computer and in response to the message, that server computer transmits back yet another card deck. If the location identified by the pointer is within cellular telephone **100**, the client process simply retrieves the deck. In either case, fax form **210** (FIG. **2**G), that is an entry card, is displayed on display screen **105** by cellular telephone **100**. This example demonstrates the information accessed by the client process can be located in any number of locations. The resource locator associated with the fax key identifies the appropriate location.

When fax form **210** is displayed, the user enters the facsimile machine telephone number at ABC Designs, as shown in FIG. **2**H, using the cellular telephone keypad. In this embodiment, the telephone number is automatically formatted by the client process. After the telephone number is entered, the client process appends the telephone number to a resource locator and transmits the information to server computer **121**.

When server computer **121** receives the information, server computer **121** executes a common gateway interface application (CGI) pointed to by the resource locator. The CGI application grabs the necessary information and transmits the information via e-mail to a fax gateway.

The fax gateway, upon receipt of the e-mail, converts the information to a fax and sends the information to the specified telephone number. Thus, cellular telephone **100** requires neither a printer connection nor a print driver, but yet can print using the facsimile machine at ABC Designs.

As illustrated in this example, cellular telephone **100** transmitted a request for a particular purchase order, and scheduled transmission of data responsive to the request to a local machine capable of printing the data. Thus, the processes of this invention, as described more completely below, in cellular telephone **100** in combination with data capable cellular telephone network **110** and server computer **121** permit cellular telephone **100** to effectively utilize an application on server computer **121** on network **120** even though cellular telephone **100** utilizes only a microcontroller found in telephone **100** and does not required a separate computer module as in the prior art.

In addition, the client process using the information transmitted from server computer **121**, i.e, the cards, generates a wide-variety of user interfaces as illustrated in FIGS. **2**A to **2**H. The particular configuration of the various user interfaces is defined by the cards transmitted in a card deck. Consequently, the user interface is not fixed to one particular format such as an E-mail type format, but rather the format is variable and can be redefined by each card that is interpreted by the client process. Also, in general, the user interface for one application on a server computer is independent from the user interface for another application on that server computer.

Specifically, the application accessed on server computer **121** generates the card deck and so in turn defines each of the various user interfaces. Each user interface permits the user to identify a particular selection. Each particular selection could result in generation of a different user interface with different selections. Thus, the user interfaces are limited only by the applications accessible to the two-way data communication device.

As shown below, a wide variety of applications can be provided on a server computer. Despite the robustness of the client module in interpreting a wide variety of application, typically, the client process is lightweight and thus requires only lightweight resources, e.g., 60 Kbytes of read-only memory (ROM) for the client module, 10 Kbytes of random access memory (RAM), and less than one million instructions per second (MIPS) of processing power. Since the client process needs only these lightweight resources in a two-way data communication device, the client can use existing resources in such a device and therefore does not add to the cost of the two-way data communication device such as data capable cellular telephone **100**.

In another embodiment, the user can configure cellular telephone **100** to access server computer **131** on corporate local area network **130**. In response to the access by the user, computer **131** transmits a home card (not shown) to cellular telephone **100** which in turn generates a home screen display on display screen **105**.

When the user selects personal information on the home screen display or on a subsequent screen display associated with the home card, a message including a resource locator for a personal information deck is transmitted from cellular telephone **100** to computer **131**. In response to the message, computer **131** transmits a card deck that includes a display card and a choice card to cellular telephone **100**. In these examples, the card deck is described as including one of three cards, a display card, a choice card, and an entry card. However, these examples are illustrative only, and are not intended to limit the invention to those particular embodiments of cards. In view of this disclosure, those skilled in the art will be able to form combinations of these types of cards and define other types of cards, if such cards are appropriate for the particular application.

The client process in cellular telephone **100** interprets the display card that includes image and text data and generates screen display **300** on display screen **105** (FIG. **3**A). Screen display **300** includes a home key **301**, and an info key **302**. When the user selects home key **301**, the user is returned to the home screen. Info key **302** functions in a manner similar to that described above for info key **205**.

When the user presses a predetermined key, the client process interprets the choice card and a second screen display **304** (FIG. **3**B) is driven on display screen **105**.

Screen display **304** is a menu of the personal information that is stored on server computer **131** for use by the user of cellular telephone **100**. Multi-display screen card indicator **203**, e.g., the hand with a finger pointing down, illustrates to the user that the list has additional items that appear on the next screen display. Screen display **304** also indicates the number of E-mail messages, faxes, and voice messages waiting for the user.

The user scrolls the screen display line by line until screen display **305** is on display screen **105**. Initially, the fourth item in the menu is not highlighted. In this example, the user presses the four key on the keypad of cellular telephone **100** to view the user's schedule. In response to the key press, the client module in cellular telephone **100** transmits a message, including a resource locator associated with the menu item selected by pressing the four key, to server computer **131** using data capable cellular telephone network **110** and corporate local area network **130**.

In response to the message, server computer **131** executes the application identified in the resource locator. Upon completion of the execution, server computer **131** transmits, over corporate local area network **130** and data capable cellular telephone network **110** to cellular telephone **100**, a card deck that includes a choice card that describes the user's schedule for that day.

In this embodiment, when server computer **131** completes the transmission, server computer **131** has completed the response to the message and has transmitted all necessary information to cellular telephone **100**. Therefore, server computer **131** does not retain any state information concerning the transmitted information and so is referred to as a stateless server computer **131**. In this embodiment, the client process can only request a card deck. However, as demonstrated herein, card decks and the two-way interactive data communication system of this invention provide the user with a new level of capability.

When cellular telephone **100** receives the card deck, the client process in cellular telephone **100** interprets the choice card and drives screen display **306** (FIG. **3**D) on display screen **105**. Initially, the first item in the menu of screen display **306** is not highlighted. When the user depresses the one key on the keypad of cellular telephone **100**, cellular telephone **100** highlights the first item in the menu. Cellular telephone **100** generates screen display **308** (FIG. **3**E) upon the user subsequently depressing a predetermined key. Screen display **308** includes a schedule key **309**, that when selected returns the user to screen display **306** (FIG. **3**D). Screen display **308** also includes a more detailed description of the 10:00 a.m. meeting.

While screen display **308** is active, if the user depresses a predetermined key, the user is presented with the options in screen display **310** (FIG. **3**F). Initially, item two in screen display **310** is not highlighted.

In this example, the user depresses key two on the keypad of cellular telephone **100** and so cellular telephone **100** sends a message including a resource locator to server computer **131** to send an E-mail message to Bill Smith confirming the meeting at 10:00 a.m. When server computer **131** executes the application addressed by the resource locator, an E-mail message is sent.

In another example, the user of cellular telephone **100** connects to Internet service provider computer **141** on Internet **140** using data capable cellular telephone network **110**. Upon connection of cellular telephone **100**, service provider **141** transmits to cellular telephone **100** a card deck to generate FIGS. **4**A to **4**C.

The client process in cellular telephone **100** interprets the first card in the card deck from computer **141** and generates screen display **400** (FIG. **4**A). When the user presses a predetermined key, cellular telephone **100** displays screen display **401** (FIG. **4**B). Screen display **401** provides the user with a series of choices that group services alphabetically.

When the user depresses the seven key on the keypad of cellular telephone **100**, cellular telephone **100** displays a list of the services that have letters P, R, or S as the first letter in the service name. In this embodiment, screen displays **401** and **402** are a single card, e.g., a single screen. Each of the various services associated with a key has an index and when a particular choice is made by the user, the choice defines an index. The client process then displays all of the services with the index that corresponds to the index defined by the user's choice.

In screen display **402**, the user is given a series of choices of services that are available to the user under tab seven. Initially, item three in screen display **402** is not highlighted. In this example, the user depresses the three key on the keypad of cellular telephone **100** to select the stock quotes and item three in screen display **402** is highlighted.

In response to this selection, cellular telephone **100** transmits a request for a stock quote, i.e, a message including a resource locator, over cellular telephone network **100** and internet **140** to service provider **141**. In response to the request, service provider computer **141** executes the application addressed by the resource locator. The application retrieves a card deck that, in turn is transmitted to cellular telephone **100**. The card deck includes a display card and an entry card.

Upon receiving the card deck, the client process in cellular telephone **100** interprets the display card and generates screen display **403** (FIG. **4**D). When the user depresses a predetermined key, entry screen display **406** (FIG. **4**E) is generated on display screen **105** of cellular telephone **100**.

Initially, the box with letters SUNW in screen display **406** is empty. The letters SUNW are entered in the box by the user to indicate the ticker symbol of the stock for which the user wants information. After the user has entered the stock ticker symbol, the user presses the predetermined key to indicate that the entry is complete.

In response to the entry by the user, the client module appends the stock ticker symbol to the resource locator and transmits the resource locator to service provider computer **141** which, in turn, executes an application addressed by the resource locator to retrieve the latest stock market information for the stock ticker symbol. Service provider **141** uses the retrieved information to generate a card deck that contains the information and then transmits the card deck to cellular telephone **100**.

The client process in cellular telephone **100** interprets the first card in the deck and generates screen display **409** (FIG. **4**F). For convenience, the FIGS. **4**F to **4**I are grouped together and separated by a dotted line. However, at any given time, in this embodiment, display screen **105** can display any four adjacent lines and so the grouping of lines in FIGS. **4**F to **4**I is for convenience only to demonstrate the level of information that can be retrieved and displayed by the client process. The use of a four line display screen is illustrative only. The client process of this invention can work with any size display screen, even a one line display screen. However, a multi-line display screen is preferred.

In the Figures discussed above, the display screen is a pixel display and so can display images. In another

embodiment, the display screen only displays text and is smaller in size. For such an embodiment, the various entries are abbreviated and only text is displayed, but the general operation is identical to that just described. Also, the various computer networks can be interlinked so that a user with access to one computer network can obtain information on another computer network. Moreover, the embodiments described above are merely illustrative. One important aspect of this invention is that cellular telephone **100** can interact with any type of server application that is configured to communicate with and interact with the client process in cellular telephone **100**. Thus, the user is no longer limited to only a few services offered by a telephone network provider.

In FIG. **1**, the cellular telephone user must address, i.e., connect to, each computer of interest to access the different services. Consequently, each computer requires the information necessary to communicate with cellular telephone **100**. In another embodiment, not illustrated, cellular telephone **100** contacts a single central computer over data capable cellular telephone network **110**. This computer is connected to each of the other networks illustrated in FIG. **1**. Consequently, the user of cellular telephone **100** sends a message including a resource locator to the central computer, the central computer processes the message and retrieves the information addressed by the resource locator from the appropriate network shown in FIG. **1**. After the requested information is retrieved, the central computer generates a card deck and transmits the card deck to cellular telephone **100**. In this embodiment, only one computer must be configured to communicate with cellular telephone **100**. However, that same computer must be configured to communicate with all other computer networks that are of interest to the user of cellular telephone **100**.

Hence, according to the principles of this invention, the client process on a two-way data communication device can initiate an interaction with a particular server computer. The server computer transmits (i) information to the client process to generate a user interface, and (ii) a resource locator for each possible selection by the user from the user interface. The resource locators can address applications on the server computer, applications on over server computers, or an application on the server computer that in turn accesses other server computers. Consequently, the user of a two-way data communication device is limited only by the applications provided on the server computers.

Further, the user can be provided new and/or updated capabilities by modifying the applications on the server computers. There is no requirement that the client process be changed for a new or updated application. The client process must only interpret the information received from an application and transmit a message for additional information. These operations are unaffected by a new or updated application. Consequently, as noted above, this invention does not require distribution of application updates or new applications to the end user of the two-way data communication device.

FIG. **5** is an illustration of another embodiment of airnet network **150**. In this embodiment, the messages from a two-way data communication device, e.g., devices **100**, **101**, and **102** are directed to an airnet network translator **500**. Airnet network translator **500** and a particular two-way data communication device, e.g., any one of devices **100**, **101**, and **102** communicate using the protocol for point-to-point communication on the particular network linking airnet network translator **500** and that two-way data communication device. For example, if data capable cellular telephone network **110** is a cellular digital packet data network, either the transmission control protocol (TCP) or the user datagram protocol (UDP) can be used.

Airnet network translator **500** transfers data between the two-way data communication device and the selected computer network after translator **500** validates the communication path, as explained more completely below, and encrypts the message transferred to the computer network if necessary. In addition, airnet network translator **500** collects transaction and billing information concerning the communication between the two-way data communication device and the designated computer network. Specifically, airnet network translator **500** provides access control for paying services and a logging mechanism for billing. Airnet network translator **500** can also provide a directory service to users.

FIG. **6** is a block diagram of a typical GSM digital cellular telephone. Each of the hardware components in cellular telephone **600** is known to those skilled in the art and so the hardware components are not described in detail herein. The compiled and linked processes of this invention are stored in ROM **601** as a client module **602** and support modules **603**. Upon activation of a predetermined key sequence utilizing the keypad, physical layer processor **610**, that is sometimes referred to herein as a microcontroller, initiates a client process using client module **602** in ROM **601**.

In this embodiment, client module **602** includes a plurality of manager modules, as explained more completely below. The particular manager modules utilized is determined by the characteristics of the particular cellular telephone **100** in which client module **602** is implemented. Client module **602** must include manager modules to interface with modules that control the particular hardware in cellular telephone **100**, a manager module to interface with the particular cellular telephone network protocol used by cellular telephone **100**, and a manager module to interpret the card decks received. Therefore, the particular manager modules described herein are only illustrative of the principles of this invention and are not intended to limit the invention to the specific modules described more completely below.

In this embodiment, the client process controls the operations of a plurality of cellular telephone dependent support processes that are stored in ROM **601** such as a display module, a keypad module, and a network and terminal control module, that were referred to above collectively as support modules **603**. The combination of the client process, display process, keypad process, and network and terminal control process are considered foreground tasks by the microkernel in cellular telephone **600**. Also, herein module and process are used interchangeably, but those skilled in the art will appreciate that the module is the computer software as stored in a memory, preferably, a ROM, of cellular telephone **600** and the corresponding process is the execution of the module by the microcontroller in cellular telephone **600**. Again, note that this invention does not require a separate processor and instead can utilize the processing power that already exists in cellular telephone **600**, because as described above, the client process of this invention is so lightweight.

The user interface for cellular telephone **600** determines the version of the user interface manager module that is stored in ROM **601**. In one embodiment, the parameters used to define the user interface level are the display resolution, the pixel access of the display, and the support of soft keys. One definition of the user interface levels is given in Table 1.

TABLE 1

USER INTERFACE LEVEL DEFINITIONS	
Level 1	Text only; 1 or more lines; 12 to 15 characters per line; and no soft keys.
Level 2	Text only; 4 or more lines; 20 to 25 characters per line; and soft keys.
Level 3	Pixel access; 150 by 75 pixels or larger; and soft keys.

The user interface manager module presents data to the display module which in turn drives display screen **605**; and captures data entered by the user on display screen **605**. In response to this information, the client process prepares a message for transmission by a network manager module.

To more completely explain the operations performed over airnet network **150**, FIG. **7** is a block diagram that illustrates the various components in one embodiment of this invention of cellular telephone **700**. Those skilled in the art will appreciate that cellular telephone **700** includes circuitry and software similar to that illustrated in cellular telephone **600** for voice and data operations supported by cellular telephone **700** in addition to the modules for operation on airnet network **750**. Similarly, server computer **743** includes other software and hardware that is known to those skilled in the art and so is not illustrated in FIG. **7** for clarity.

In this embodiment, client module **702** in digital cellular telephone **700**, that is executing on the microcontroller of telephone **700**, communicates with server computer **743** over cellular digital packet data (CDPD) network **710**. Cellular digital packet data network **710** is used to illustrate one embodiment of this invention on one two-way data communication network. The principles of this invention can be used with a wide variety of two-way data communication networks. For example other two-way data communication networks for cellular telephones that may be used include TDMA, CDMA, and GSM circuit switched data networks; and the AMPS analog cellular network with a modem. Similarly, for two-way pagers, two-way data communication networks include PACT, or other priority two-way paging networks with data transport capability.

Prior to considering the operation of this configuration of airnet network **750** in more detail,, another aspect of this invention is required. Specifically, a technique is required for conveying instructions from digital cellular telephone **700** to a server application on server computer **743**, and conversely.

A telephone interaction description language (PIDL) is defined for use by service developers. A terminal interaction language (TIL) is a distillation of the telephone interaction description language and describes the same interaction to digital cellular telephone **700** as the telephone interaction description language describes to computer **743**.

With the exceptions described more completely below, a process in the terminal interaction language is a compressed version of the same process written in the telephone interaction description language. The terminal interaction language allows easy parsing on the two-way data communication device, which in turn makes the client smaller than a client for the telephone interaction description language that is readable by humans, but is not optimized for parsing by a machine.

The compression from the telephone interaction description language to the terminal interaction description language is done typically at run time because some cards are computed cards and so cannot be precompiled. A wide variety of techniques can be used to convert the telephone interaction description language to terminal interaction language. The important aspect is that, if bandwidth across the cellular telephone network is limited, a compressed form of the telephone interaction description language is used.

Preferably, each data type is compressed to facilitate optimal transfer over the two-way data communication network. For example, the verbs in the telephone interaction description language are compressed using a binary tokenization. Graphics are compressed using run length limited compression and text is compressed using any one of the well-known techniques for text compression. While compression of the telephone interaction description language is not required to implement this invention, compression makes the invention more efficient by utilizing the bandwidth of the network more effectively.

Instructions in the telephone interaction description language and in the terminal interaction language are grouped into a deck and a card. Each deck includes one or more cards. A card includes the information, i.e., a set of telephone interaction description language, required to generate a screen. As indicated above, a screen can be larger than the number of lines in a display screen. Other equivalent terms for a card include a page and an atomic interaction. Thus, a card deck is simply a group of screens. The number of cards in a card deck is selected to facilitate efficient use of the resources in the two-way data communication device and in the airnet network.

For simplicity, in this embodiment, each card is a single operation. Herein, an operation is defined as a related set of actions such that the user does not encounter an unanticipated delay in moving from one action to the next, i.e, the user does not have to wait for client module **702** to retrieve another card deck from computer **743**. Also, a deck may include definitions of soft keys that stay in force while the deck is active, i.e, being executed by the cellular telephone microcontroller.

Computer **743** may contain stored static telephone interaction description language decks. Computer **743** also generates telephone interaction description language decks in response to data from, or choices made by, the user of cellular telephone **700**.

In the embodiment shown in FIG. **7**, computer **743** converts a telephone interaction description language deck to a terminal interaction language deck, that in turn is transmitted to cellular telephone **700**. The terminal interaction language is designed so that decks can be stored unaltered in memory **716** of cellular telephone **700** and referenced directly with little or no parsing. While telephone interaction description language decks on computer **743** may contain references to images, a terminal interaction language deck contains the images at the end of the deck. Thus, if a particular two-way data communication device does not support display of images, the images are easily stripped from the terminal interaction language deck before the deck is transmitted to that particular two-way data communication device.

As indicated above, each interaction with the user of cellular telephone **700** is described by a deck or a series of decks. Logically, the user retrieves a terminal interaction language deck stored in a memory **716** of cellular telephone **700** after receipt from computer **743** over CDPD network **710**. The user reviews the information displayed by cards in the deck and makes choices and/or enters requested infor-

mation and then requests another deck, as described above with respect to FIGS. **2**A to **2**H, for example.

When the user receives a deck, the first card of information is displayed on display screen **705**. Typically, as shown above, the first card is text, an image, or a combination of an image and text. After the user has reviewed the first card, the user hits a NEXT key to view the next card in the deck. Similarly, a user can return to a previous card in the deck by using a PREV key. Thus, using the NEXT and PREV keys, the user can navigate back and forth through the deck. Within a card, the user uses a scroll key or keys to move the portion of the card displayed up and down. This description of a particular method used to navigate through a deck and within a card is not intended to limit the invention to this particular method. In view of this disclosure, those skilled in the art will be able to use a wide variety of ways to navigate through a deck and within a card.

Cards, in this embodiment, are one of three types, a display card, a choice card, and an entry card. Independent of the type of card, the card can contain text and images. In addition, the invention is not limited to these three particular types of cards. The definition of the three particular types of cards is used to facilitate a description of the invention and to assist the developer's in organizing applications.

A display card gives information to the user to read. The display content can include any one of, or any combination of text, an image, and a soft key. The soft key is in effect only while the display card is active.

A choice card displays a list of choices for the user. The choices are automatically presented in a format specified on the choice card. See Appendix I, which is a part of the present disclosure and is incorporated herein by reference in its entirety. As explained above, the user makes a choice by depressing the key corresponding to the choice.

An entry card is used to obtain input data from the user. An entry card displays one or more entry lines. Typically, each entry line includes a display followed by an entry line. The entry line, in this embodiment, can be for either numeric or text data.

In this embodiment, choice and entry cards prevent the user from moving to the next card until the user has entered the requested information. When the user reaches the last card in a deck and hits the NEXT key, a request for a new deck is initiated. The deck requested is determined by either the deck that the user has completed, or by the choices made by the user. Also, when the deck is completed, the choices and/or data entered by the user typically are transmitted along with the request for the new deck to computer **743**.

Appendix I is one embodiment of a syntax for the telephone interaction description language and the terminal interaction language of this invention. In one embodiment, the telephone interaction description language is described using a subset of the standard generalized markup language. Only a subset of the standard generalized markup language is utilized so that telephone interaction description language parsers also can be written easily using simple tools like lex and yacc.

Returning to operation over airnet network **750**, cellular telephone **700** includes a display module **712**, a keyboard module **711**, a client module **702**, and a UDP interface module **714**. In this embodiment, module **702** is stored in a non-volatile memory (not shown) of telephone **700** and is executed by the microcontroller (not shown) in telephone **700**. Modules **711**, **712**, and **714** operate under the control of client module **702**. Client module **702** includes instructions that direct the microcontroller in cellular telephone **700** to perform the operations described more completely below with respect to FIGS. **8**A to **8**D. The operations include sending uniform resource locator (URL) requests to HyperText Transfer Protocol (HTTP) server **749**, parsing and displaying a TIL deck or decks returned by HTTP server **749**, and generating new URLs based on the user's key presses. For a description of HTTP server software and platforms that can run the HTTP server software, see, for example, Ian S. Graham, *The HTML Sourcebook,* John Wiley & Sons, Inc., New York, Chapt. 8, (1995), which is incorporated herein by reference.

User datagram protocol (UDP) interface module **714** couples CDPD network **710** to client module **702**, and allows client module **702** to communicate using UDP over CDPD network **710**. The user datagram protocol is well known to those skilled in the art and is documented extensively. UDP interface module **714** supports transmission of simple stand-alone messages between the connection partners.

Display module **712** is a display driver that couples client module **702** to display screen **705** and so allows client module **702** to specify the information presented on display screen **705**. The user interface manager module within client module **702** converts the display information in a card to instructions for display module **704** which in turn provides signals that drive the hardware that controls the operation of display screen **705**. For example, if the TIL deck includes an image, the user interface manager module determines whether the active card calls for display of the image. If the active card directs the user interface manager module to display the image, the user interface manager module passes the image in memory **716** to display module **712**, which in turn displays the image on display screen **705**.

Keyboard module **705** couples keypad **715** to client module **702**, and stores data representing keys pressed by the user on physical keypad **715** in memory **716**. Keyboard module **705** notifies client module **702** when the user has pressed a key.

When client module **702** is notified of a key press, the user interface manager module within client module **702** passes information about the key press to display module **712** that in turn displays the appropriate character on display screen **705**, if an entry card is active. If the user interface manager module determines that a choice card is active, and the key press corresponds to one of the choices, the user interface manager module sends instructions to display module **712** that result in the choice being identified for the user, e.g., highlighted as described above.

In addition to HTTP server **749**, host computer **743** includes a UDP interface module **748**, CGI programs **761** stored in a memory **755** of host computer **743**, and TIL decks **760** stored in memory **755**.

HTTP server **749** uses UDP interface module **748** to send data to and receive data from CDPD network **710**. TIL decks **760** are TIL decks that can be accessed by HTTP server **749**. Static files containing PIDL decks are converted to TIL decks only once on HTTP server **749**. CGI programs **761** are common gateway interface programs that produce PIDL decks that are used by HTTP server **749** to produce TIL decks that in turn are transmitted via UDP interface modules **748** and **714** and cellular telephone network **710** to client module **702**. In this embodiment, the services available over airnet network **750** are applications accessible by HTTP server **749** on Internet **140** for which a service developer has written a PIDL deck, or a CGI script that in turn generates a PIDL deck, and is stored on computer **743**.

The architecture in FIG. **7** demonstrates some important aspects of this invention. First, the applications, the PIDL decks and CGI scripts in this embodiment, are independent of the particular two-way data communication network. For HTTP server **749** to communicate over a different two-way data communication network that does not support UDP, only UDP interface module **748** must be changed. The applications are unaffected by such a change.

Second, the applications on HTTP server **749** are independent of the two-way data communication device with which HTTP server **749** is interacting. An application on HTTP server **749** can communicate with any two-way data communication device that includes the appropriate client and a module to transmit and receive data over the two-way data communication network. These two facts mean that an investment in developing an application is insulated from either advances in two-way data communication devices, or advances in two-way data communication network technology.

FIGS. **8**A to **8**D are a process flow diagram for one embodiment of this invention. Initially, when the user initiates communication over airnet network **750**, client module **702** initializes a work space in memory **716** of cellular telephone **700** and then, in get home URL process **801**, stores a URL in the work space. According to the principles of this invention, in one embodiment, each cellular telephone that utilizes the airnet network has a home URL stored in a non-volatile memory that is used to retrieve a home card deck for the cellular telephone. In another embodiment, the cellular telephone obtains the home URL from server **749**. Thus, in get home URL process **801**, client module **702** obtains the home URL. Herein, a URL is an example of a specific embodiment of a resource locator.

For example, in get home URL process **801**, client module **702** obtains a home URL, such as

http://www.libris.com/airnet/home.cgi

and stores the home URL in the work space. The portion of the home URL, http:/www.libris.com, identifies a particular HTTP server, i.e, server **749**, on the world-wide web. The portion of the URL, /airnet/home.cgi, specifies a particular common gateway interface program within CGI programs **761**. The use of a URL pointing to a server on the world-wide web is illustrative only is not intended to limit the invention to applications on the world-wide web. In general, cellular telephone **700** obtains an identifier, i.e,.a resource locator, of a home application on a home server that is executed by the server when the cellular telephone initially becomes active on airnet network **750**, and stores the resource locator in the work space.

Next in create HTTP request process **802**, client module **702** converts the URL in the work space to a HTTP request. For example, for the above URL, create HTTP request process **802** generates a method field, such as

GET /airnet/home.cgi HTTP/1.0

The GET method is part of HTTP. Thus, the format for the GET method is known to those skilled in the art. Also, this particular form of the method is used because a specific server connection is established by cellular telephone **700** and so identification of the server is unnecessary. Nevertheless, briefly, this command instructs server **749** to execute application home.cgi and execution of application home.cgi in turn results in generation of a home deck and a subsequent transmission of the home deck to cellular telephone **700**. HTTP/1.0 specifies the HTTP version used by client module **702** in cellular telephone **700**.

In addition to the method field, client module **702** in process **802** could also generate appropriate HTTP request fields to pass information to server **749** about the capabilities of client module **702**. The request fields can include information such as lists of the MIME content-types acceptable to the client; lists of data encoding types acceptable to the client; user authentication and encryption scheme information for the server; the length in bytes of the message being sent to the server; and the Internet mail address of the user accessing the server. This list of information is illustrative only and is not intended to limit the invention to the particular request fields described herein. Any request field defined by HTTP can be utilized by client module **702**. However, in this embodiment, the defaults are utilized and so no HTTP request fields are generated.

Typical HTTP methods that can be generated in HTTP request process **802** are a GET method for requesting either a TIL deck from server **749**, or execution of a common gateway interface program on server **749**; and a GET method request to a common gateway interface program with data, e.g., a query string appended to the URL. In either case, a URL is transmitted to server **749** within the particular message. After create HTTP request process **802** is complete, client process transfers to transmit request process **804**.

However, if the transmission control protocol is used instead of UDP, client module **702** would access a TCP module in establish server connection process **803** that replaced UDP module **714**. Since, in this embodiment, UDP is used, establish connection process **803** is enclosed by a dashed line in FIG. **8**A to indicate that this process is unnecessary when using UDP.

In establish server connection process **803**, a virtual connection would be made over CDPD network **710** between TCP interface module **714** and a TCP interface module in HTTP server **749** so that data could be transmitted between cellular telephone **700** and computer **743** using TCP, e.g., buffers to support data exchange are defined. The establishment of a TCP connection is well-known and so is not described further.

In FIG. **8**A, a dashed line connects establish server connection process **803** with establish client connection process **860**, that is also dashed, that is performed by HTTP server **749**. This indicates that both client module **702** and server **749** are required to complete process **803**.

When the TCP virtual connection is established, client module **702** transfers processing from establish server connection process **803** to transmit request process **804**. Similarly, server **749** transfers to request received check **861**, in which server **749** waits until a request is received. Establish client connection process **860** is not needed for UDP and so HTTP server **749** initiates processing in request received check process **861**. Process **860** is enclosed within a dashed line box to indicate that the process is used only for TCP.

In transmit request process **804**, the HTTP request is sent from the work area in telephone **700** to HTTP server **749**. Again, a dashed line connects process **804** of client module **702** to request received check **861** that is performed by HTTP server **749** to indicate that the check is dependent upon information from client module **702**. When the transmission of the request is complete, client module **702** transfers to response received check **806**.

Upon receipt and storage of the HTTP request, request received check **861** transfers to service request process **862** in which HTTP server **749** initiates service of the received request. In service request process **862**, if the HTTP request only seeks transfer of a static deck, HTTP server **749** retrieves the requested static deck from TIL decks **760**.

Conversely, if the request requires server **749** to obtain data from the Internet or to append data to a particular file, server **749** launches the common gateway interface application addressed in the request, and passes the data in the HTTP request to this application for further processing.

For example, if the user of cellular telephone **700** requested a fax as in FIG. **2**F, the HTTP request identifies a common gateway interface application in CGI programs **761** that accepts as input data the telephone number and grabs the information to be faxed. The CGI application generates an e-mail transmission to the fax gateway. Similarly, for a stock quote, server **749**, in response to the HTTP request, launches a common gateway interface application that sends out a stock query over Internet **140** to a stock quote service provider using the ticker tape symbol passed as input data by server **749** to the common gateway interface application. When the response to the stock query is received, the common gateway interface application builds a PIDL deck that includes the data in the response to the stock query.

Upon completion of servicing the request, HTTP server **749** converts the PIDL deck to a TIL deck and returns the TIL deck to client module **702** using UDP in transfer response process **863**, that is connected by a dotted line to response received check **806** in client module **702**. As the TIL deck is transferred, client module **702** stores the deck in memory **716**.

After the TIL deck is transferred, HTTP server **749** closes the process for responding to the message from cellular telephone **700**. All the information needed by client module **702** to generate a user interface on display screen **705** and for responding to any selection or data entry presented in the user interface is included in the TIL deck. Consequently, client module **702** only has to interpret the TIL deck and interpret the user input to transmit the next message to HTTP server **749**. The state for the HTTP server is defined in the next message. Consequently, HTTP server **749** is stateless because HTTP server **749** does not retain state information concerning a response to a message after the message is transmitted.

However, in another embodiment (not shown), a server could retain state information concerning each interaction with a client module. For example, if the server transmitted a choice card to the client module, the server would retain state information indicating that a choice was pending from the client module. In this embodiment, when the user makes a choice, e.g., depresses key two to indicate choice two, the choice is transmitted to the server which in turn accesses the. URL associated with choice two. If this URL addresses another application, the server executes that application. Thus, in this embodiment, the server retains state information concerning each interaction with a client module. In view of this disclosure, those skilled in the art can implement the principles of this invention utilizing a server that retains state information when such a client/server combination is advantageous.

Returning to the present embodiment, when the TIL deck is received, client module **702** leaves response received check process **806** and transfers to process first card **808**. However, if TCP is used instead of UDP, client module **702** upon leaving check **806** would close the virtual TCP connection in transmission completed process **807**. Upon closing the virtual TCP connection, processing would transfer to process first card **808**. Again, transmission complete process **807** is enclosed within a dashed line box to indicate that process **807** is used only with TCP.

In process first card **808**, client module **702** parses the TIL deck and interprets the first card. Processing transfers from process first card **808** to generate display process **809**.

In generate display process **809**, client module **702** passes the data to be displayed in the first card to display module **712**. Display module **712**, in response to the data, drives the text and images in the data on display screen **705**. Generate display process **809** transfers processing to key press check **820** through node **813**. In FIGS. **8**A to **8**D, any circular node with the same alphanumeric character and reference numeral is the same node. The circular nodes are used to establish connections between the various processes in the method of FIGS. **8**A to **8**D without cluttering the figures with a number of connection lines.

Client module **702** waits in key press check **820** for the user to press a key on keypad **715** of cellular telephone **700**. In this embodiment, cellular telephone **700** is assumed to have the capability to support two soft keys, a scroll-up key, a scroll-down key, a previous key, a next key, and keys zero to **9** that are configured in the standard telephone keypad configuration. In view of the following disclosure, if one or more of these keys are not present, one of skill in the art can alter the method for the particular configuration of the cellular telephone keypad, or other two-way data communication device keypad. For example, if the cellular telephone included a home key, the key press processing described more completely below would include a check that detected when the home key was pressed and would in turn transfer to get home URL process **801**.

Briefly, the processes in FIGS. **8**B to **8**C, identify the key pressed by the user, identify the action required, and then transfer to a process that implements the action required. Specifically, when a key on the keypad is pressed, keypad module **711** stores an identifier for the key in work memory **716** and notifies client module **702** of the key press. Upon receipt of the notification from keypad module **711**, client module **702** reads the storage location in work memory **716** to determine the key pressed and transfers processing from key press check **820** to scroll key check **821**. In scroll key check **821**, client module **702** determines whether the user pressed either of the scroll keys. If a scroll key was pressed, processing transfers to adjust display process **822** and otherwise to display card check **823**.

In adjust display process **822**, client module **702** determines which of the scroll-up or scroll-down keys was pressed. Client module **702** then sends information to display module **712** so that the current display is either scrolled-up one line or scrolled- down one line. If the scroll key would move the display beyond a boundary of the current card, the scroll key press is ignored in adjust display process **822**.

In response to the information from client module **702**, display module **712** adjusts the screen display on display screen **705**. Client module **702** transfers processing from adjust display process **822** to key press check **820** through node **813**.

If a scroll key was not pressed, processing is passed through scroll key check **821** to display card check **823**. Client module **702** takes action that depends on the particular type of card that is currently being displayed on display screen **705**. If the current card is a display card, client module **702** passes through display card check **823** to soft key check **828**, and otherwise transfers to choice card check **824**.

Assuming for the moment that the current card is not a display card, choice card check **824** determines whether the current card is a choice card. If the current card is a choice card, client module **702** passes through choice card check **824** to choice key check **826**, and otherwise transfers to data key check **826**.

Assuming for the moment that the current card is neither a display card nor a choice card, the current card must be an entry card, because in this embodiment only three card types are defined. Thus, client module **702** does not check for an entry card. Rather, data key check **826** determines whether a valid data key was pressed. In this embodiment, the data keys are keys zero to nine on the key pad, and the # key. In other embodiments, other combinations of keys could be defined as data keys. If the pressed key was one of the data keys, data key check **826** transfers to process data entry **827** and otherwise transfers to soft key check **828**.

In process data entry **827**, client module **702** knows whether the predictive text entry process is turned-on, because one of the parameters on the entry card specifies whether to use the predictive text entry process, as described in Appendix I, which is incorporated herein by reference in its entirety.

If the predictive text entry process is not turned-on, client module **702** in process data entry **827** enters the pressed key value in a text entry buffer in work memory **716** at the appropriate location. Also, client module **702** sends information to display module **712** so the value of the pressed key is displayed in the appropriate location on display screen **705** by display module **712**.

If the predictive text entry process is turned-on, client module **702** uses the novel predictive text entry process in process data entry **827**, as described more completely below with respect to FIGS. **9**, **10**A to **10**T, and **11**, to determine the letter to select from the set of letters associated with the pressed key. After the predictive text entry process determines the appropriate letter, a value representing the letter is stored at the appropriate location in the text buffer in work memory **716**. Also, client module **702** sends information to display module **712** so that the letter is displayed in the appropriate location on display screen **705**. Upon completion of process data entry **827**, client module **702** transfers processing through node **813** to key press check **820**.

The previous description assumed that the current card was an entry card, but if the current card is a choice card, choice card check **824** transferred to choice key check **826**. In generate display process **804** for the choice card, each of the choices are labeled according to information on the choice card and some or all of the choices are displayed on display screen **705**. Thus, choice key check **826** determines whether the pressed key corresponds to one of the choices. If the pressed key is one of the choices, client module **702**, in one embodiment, sends information to display module **712** to indicate the selected choice. Client module **702** also transfers from choice key check **826** through node **831** to store identifier process **850** (FIG. **8**D), that is described more completely below. Conversely, if the pressed key is not one of the choices, choice key check **826** transfers to soft key check **828**.

Soft keys can be specified both for a deck as a whole and per card, i.e., a physical key on the keypad is specified as a soft key as described more completely in Appendix I. Each soft key specification includes an identifier that defines the action to be taken when the soft key is pressed.

When a soft key is specified for a deck, the soft key remains in effect for the entire deck. However, when a soft key is specified for a card, the card soft key specification temporarily overrides the corresponding deck soft key specification, i.e., the deck soft key specification for the same physical key as the card soft key specification, while the card is visible, i.e., displayed on display screen **705**. This override is done independently for the two soft keys.

Thus, soft key check **828** transfers processing to first soft key check **829** if the key pressed is one of the two possible physical soft keys. Conversely, soft key check **828** transfers processing to next key check **840** (FIG. **8**C), if neither of the two possible physical soft keys is pressed by the user.

In first soft key check **829**, client module **702** determines whether the pressed key corresponds to the first soft key. If the pressed key is the first soft key, check **829** passes the active identifier for the first soft key to store identifier process **850** through node **831**. Conversely, if the pressed key is not the first soft key, processing transfers from check **829** to second soft key check **830**.

If the pressed key is the second soft key, check **830** passes the active identifier for the second soft key to store identifier process **850** through node **831**. Conversely, if the pressed key is not the second soft key, e.g., a physical key that can be defined as a soft key was pressed but neither the current deck nor the current card defines a soft key for that physical key, processing transfers from check **830** to key press check **820** through node **813**.

When pressing transfers to next key check **840**, client module **702** determines whether the pressed key was the next key. If the next key was pressed, processing transfers to display card check **841** and otherwise to previous key check **846**.

If a display card is the current card, the next key is used to move to another card in a deck, or alternatively to another deck. Thus, display card check **841** transfers processing to last card check **842** when a display card is the current card, and otherwise to entry card check **843**.

Last card check **842** determines whether the current card is the last card in the deck. If the current display card is not the last card in the deck, last card check **842** transfers processing to read next card process **845**, which in turn reads the next card in the deck and transfers through node **812** to generate display process **809**.

If the current display card is the last card in the deck, the deck includes an identifier that specifies the location to transfer to from the last card. This identifier can be a URL to another deck, to a common gateway interface program, or an address for a card within the current deck, for example. Thus, last card check **842** transfers through node **831** to store identifier process **850** when the current display card is the last card in the deck.

If the current card is not a display card but is an entry card, display card check **841** transfers to entry card check **843**. In this embodiment, the next key is the predetermined key used to indicate that all the data for an entry on an entry card has been entered. Thus, if the current card is an entry card, entry card check **843** transfers processing to store data process **844**.

Store data process **844** stores the data entered in at an appropriate location in memory that is specified in the current entry card. Typically, the data is combined as an argument with a URL and stored. Upon completion, store data process **844** transfers through node **810** to create HTTP request process **802** (FIG. **8**A).

When the next key is pressed, if the current card is neither a display card nor an entry card, the current card is a choice card. However, as indicated above, in this embodiment client module **702** requires that the user make a choice and does not allow use of the next key. Consequently, if the current card is not an entry card, entry card check **843** transfers processing through node **813** to key press check **820**.

The previous discussion assumed that the next key was pressed and so next key check **840** transferred processing to display card check **841**. However, if the next key was not pressed, next key check **840** transfers processing to previous

key check **846**. If the previous key was pressed, check **846** transfers to first card check **847** and otherwise returns processing to key press check **820**.

First card check **847** determines whether the current card is the first card of a deck. If the current card is not the first card, processing transfers from first card check **847** to read previous card **849**, which in turn reads the previous card and transfers to generate display process **809** through node **813**. Conversely, if the current card is the first card, processing transfers to home deck check **848**.

If the current card is the first card in the home deck, there is not a previous card and so home deck check transfers processing to key press check **820** through node **813** and so the previous key press is ignored. If the current deck is not the home deck, home deck check **848** retrieves the identifier for the previous deck and transfers through node **831** to store identifier process **850**.

Store identifier process **850** is reached through node **831** from several different points. The operations in store identifier process **850** are the same irrespective of the particular process that transfers to process **850**. In each instance, an identifier is passed to store identifier process **850** and process **850** saves the identifier in working memory **716**. The identifier can be, for example, a pointer to another location in the current card, an address of another card in the current deck, a URL to a deck stored in working memory **716**, a URL to a TIL deck in TIL decks **760** on computer **743**, or perhaps, a URL to a common gateway interface program in CGI programs **761** on computer **743**. Thus, process **800** checks the stored identifier to determine the action required.

Specifically, in identifier to current deck check **851**, client module **702** determines whether the identifier is to a card in the current deck. If the identifier points to the current deck, check **851** transfers processing to retrieve data process **852** and otherwise to URL to local deck check **853**.

In retrieve data process **852**, client module **702** retrieves the information stored at the location indicated by the identifier from working memory **716** and processes the information. Retrieve data process **852** transfers through node **812** to generate display **809** (FIG. **8**A) that was described above.

URL to local deck check **853** determines whether the identifier is a URL to a deck that is stored in working memory **716**, e.g., cached. If the deck is stored locally, check **853** transfers to retrieve local deck **854** which in turn moves the local deck into the storage location for the current deck. Retrieve local deck **854** transfers processing through node **811** to process first card **808** (FIG. **8**A), that was described above.

If the identifier is neither to a location in the current deck, nor to a local deck, the identifier is a URL to an object on computer **743**. Thus, in this case, check **853** returns processing to create HTTP request **802** through node **810**.

Process **800** continues so long as the user continues to enter and process the information provided. In this embodiment, process **800** is terminated, for example, either by the user powering-off cellular telephone **700**, selecting a choice or entry card that discontinues operations of client module **702**, or remaining inactive for a time longer than a time-out period so that client module **702** shuts itself down.

To further illustrate the operations in process **800**, consider the following example which is returned to client module **702** as a TIL deck in response to a HTTP request generated by process **802**. For readability, Table 2 presents the deck in PIDL. In this example, all of the choices are for applications on the same server. However, in another embodiment, each URL could address any desired combination of servers.

TABLE 2

EXAMPLE OF PIDL CHOICE DECK

```
<PIDL>
<CHOICE>
<CE URL=http://www.libris.com/airnet/nnn>News
<CE URL=http://www.libris.com/airnet/www>Weather
<CE URL=http://www.libris.com/airnet/sss>Sports
</CHOICE>
</PIDL>
```

In process first card **808**, client module **702** interprets the information in Table 2 and transfers to generate display process **809**. In generate display process **809**, client module **702** sends information to display module **712** so that the user is presented with a list of three choices on display screen **705**, i.e, a user interface for the choice card is generated:

1. News
2. Weather
3. Sports

Generate display process **809** (FIG. **8**A) transfers to key press check **820** (FIG. **8**B). When the user **35** presses the two key on keypad **715**, key press check **820** transfers through check **821** to display card check **823**.

Since the current card is a choice card, check **823** transfers processing to choice card check **824**, which in turn transfers to choice key check **826**. Since the two key was pressed and that key is a choice key, check **826** transfers processing to store identifier process **850** (FIG. **8**D). In process **850**, client module **702** stores the URL corresponding to two, i.e,

URL=http://www.libris.com/airnet/www

in working memory **716**.

Since this URL is to an object on computer **743**, processing transfers through checks **851** and **853** to create HTTP request process **802**, which in turn generates the request. When the HTTP request is transmitted to server **749**, as described above with respect to process **804**, server **749** in service request process **862** retrieves deck www from TIL decks **760**. An example of the deck is given in Table 3. Again for readability, the deck in present herein in PIDL.

TABLE 3

EXAMPLE OF A SECOND PIDL CHOICE DECK

```
<PIDL>
<CHOICE>
<CE URL=http://www.libris.com/airnet/www-1>World
<CE URL=http://www.libris.com/airnet
  /www-2>National
<CE URL=http://www.libris.com/airnet/www-3>State
<CE URL=http://www.libris.com/airnet/www-4>Local
</CHOICE>
</PIDL>
```

The deck in Table 3 is transmitted to cellular telephone **700** and stored in memory **716**, as described above with respect to process **806**. The choice card is processed in process **808** and displayed in process **809**. As a result of process **809**, the user is presented with a list of choices:

1. World
2. National
3. State
4. Local.

When the user makes another selection, the same sequence of processes as described above for the first choice card is executed by client module **702**, and another URL is stored that points to a program on server **749** that retrieves the desired weather information and generates a deck with that information. This deck is transferred to cellular telephone **700** and displayed.

As described above, if the current card is an entry card and a key is pressed, client process **702** reaches data key press check **826** (FIG. **8**B). If the pressed key is a valid data key, check **826** transfers to process data entry **827**.

In one embodiment, process data entry **827** uses a novel predictive text entry process for text entry. Recall that on a typical telephone keypad, the keys are labeled with both a number and two or three letters. For example, the two key is also labeled abc. This leads to some ambiguity when using the telephone keypad to enter text. Is the user attempting to enter an a, b, or c when the two key is pressed?

In one prior art method, two keystrokes were required to enter each letter of text. The first keystroke identified the first key and the second key stroke identified the specific letter desired on the first key. For example, to enter the letter s, the user would first press the seven key that is labeled with letters p, r, and s. Next, the user would press the three key to select the letter s. While this method may work well for short sequences that consist of only three or four letters, the method does not work well for English text. For example, if the user has already entered th and then presses the three key that is labeled with letters d, e, and f, almost always the desired next letter is the letter e. Therefore, making the user press the two key is an extra and unnecessary step.

Client module **702** of this invention utilizes a novel predictive text entry process to reduce the number of key strokes required to enter text using a telephone keypad, or any similar keypad. Using this process, in most cases a single key stroke suffices to enter a single letter.

While this embodiment of the invention is described in terms of a telephone keypad, the principles of the invention are not limited to only a telephone keypad. In general, the process described more completely below, can be extended to any keypad where a single key is used to enter two or more letters. Further, the process is not limited to only letters, but rather is applicable to any keypad where a single key is used to represent two or more characters. In view of the following disclosure, those skilled in the art can use the principles of the predictive text entry process in a wide variety of applications.

The system for predictive text entry includes a predictive text entry module **901** that in this embodiment is included in client module **702**, keyboard module **711**, and a letter frequency table **902** that is loaded into memory **716**, when client module **702** is activated. Predictive text entry module **901** is used in process data entry **827** when specified by the current entry card. Predictive text entry module **901** performs routine buffer management processes, that are known to one of skill in the art and so are not described further to avoid detracting from the process.

Predictive text entry module **901** stores a letter entry for each letter entered in a text buffer **903** in memory **716**. In this embodiment, letters Q and Z are assigned to the one key and the zero key is used to enter a space, period, and comma, i.e., the zero key provides punctuation. However, these assignments are illustrative only, and are not intended to limit the invention to this particular embodiment.

The first letter entered is placed at the left end of the buffer and each additional letter is placed in the left most unused space in buffer **903**. Thus, the last letter entered in text buffer **903** is the right most character. Letter frequency table **902**, sometimes referred to as a table of predictive letter entries, is a look-up table where each entry in the look-table is addressed by three indices. The first two indices represent the two most recently entered letters in text buffer **903** and the third index represents the key that was pressed. Each predictive letter entry stored in letter frequency table **902** defines which of the letters associated with the pressed key to use given the previous two letters. For example, since the is a commonly occurring string, the entry in table **902** addressed by (t, h, **3**) returns e, or more concisely the predictive letter entry **2** is returned to indicate that the second letter of the group of letters d, e, and f associated with the three key is the predicted letter. Of course, letter frequency table **902** could be altered to return more than a single letter.

In this embodiment, letter frequency table **902** was empirically generated using a collection of e-mail. Appendix II is a computer program listing that was used to generate letter frequency table **902** that is illustrated in FIGS. **10**A to **10**T. Briefly, the computer program implements a process that sequentially steps through the data provided and (i) for each possible single letter determines the most likely letter that follows for each key on the keypad; and (ii) for each possible combination of two letters determines the most likely letter that follows for each key on the keypad. In this embodiment, the most likely letter is the letter having the greatest frequency after the single letter. Similarly, the most likely letter is the letter having the greatest frequency after the combination of two letters. If there is a tie in the frequency, the first letter associated with a key is selected. Of course, other measures of likelihood could be used to generate the entries in table **902**.

Thus, in FIGS. **10**A to **10**T, the first of the ten columns, i.e., the left most column, is the two letter sequence and the first row, i.e., the top row is the keys on the key pad used to enter text. A combination of an entry in the first column and a key in the top row is used to select the predicted text entry. Thus, using the example of th, this two key sequence appears in the first column of FIG. **10**O. When the three key is pressed, the letter in the row with th as the first entry and in the column with three as the first entry, i.e., e, is retrieved. Alternatively, if the four key is pressed, letter i is retrieved from the table.

In this embodiment, table **902** is a buffer of two bit numbers. Each two bit number has a value in the range of zero to three, and the two bit number represents a predicted letter for the pressed key. Thus, for a two key labeled with letters A, B and C, a zero represents A; a one represents B; and a two represents C. In general, the number of bits used is determined by the key that represents the maximum number of characters. In this embodiment, the maximum number of characters represented by a key is three. The number of storage bits required is an integer S where S is the smallest number such that 2**S is greater than or equal to the maximum number of characters represented by a key.

In this embodiment, three indices i**0**, i**1**, and i**2** are used generate a table index that in turn is used to access a particular predictive letter entry in table **902** of two bit numbers. Each letter is represented as a number, i.e., a letter entry, with letter A being zero, letter B being a one, letter C being a two, and so forth with letter Z being twenty-five. A space element is assigned a space element value of twenty-six. Thus, in this embodiment, there are twenty-seven possible characters.

Upon the initial entry to process **1100** (FIG. **11**), letter indices i**0**, i**1**, and i**2** were set to twenty-six in the initial

processing of the entry card to indicate that the text buffer is empty. Also, as explained more completely below, as each letter of text is entered, letter indices i**0** and i**1** are updated and stored in memory **716**.

However, in another embodiment, an initialize indices process is the first operation in predictive text entry process **1100**. In this embodiment, for the first letter entered, letter indices i**0** and i**1** are set to twenty six; for the second letter entered, letter index i**0** is set to twenty six and letter index i**1** is set to the value of the letter in text buffer **903**; and for all letters entered after the first two, the value associated with next to the last letter in text buffer **903** is assigned to letter index i**0** and the value associated with the last letter in text buffer **903** is assigned to letter index i**1**.

Punctuation key check **1101** determines whether the zero key was pressed, i.e., the key selected to represent punctuation.

If the zero key was pressed, processing transfers from check **1101** to process punctuation entry **1102**. Process punctuation entry **1102** sets index i**2** to twenty-six, and sends the space element value to display letter process **1108**. Display letter process **1108** transfers the space element value to display module **712** which in turn drives a space in the text entry on display screen **705**. This completes the operation of process data entry for a zero key press and so processing returns to key press check **820**.

If the zero key was not pressed, processing transfers through punctuation key check **1101** in data entry process **1100** to key one-to-nine check **1103**, i.e., to a data entry key check. If the pressed key was any one of keys one to nine, check **1103** transfers to set letter index process **1104** and otherwise to rotate last entry process **1109**.

In set letter index process **1104**, one is subtracted from the numeric value of the pressed key and the resulting value is assigned to index i**2**. Set index process **1104** transfers to generate table index process **1105**.

Generate table index process **1105** combines indices i**0**, i**1** and i**2** to create a table index. In this embodiment, table index TABLE_INDEX is defined as:

$$TABLE\_INDEX=(((i0*27)+i1)*9)+i2$$

Upon completion of generate table index process **1105**, generate text entry process **1106**, retrieves the two bit value in the table at the location pointed to by table index TABLE_INDEX and converts the two bit value to a letter represented by the two bit value.

Generate text entry process **1106** transfers to update index process **1107**, which in turn stores the value of letter index i**1** as letter index i**0**; stores the value of the retrieved letter in letter index i**1**; and stores the predicted letter in text buffer **903**. While this step assumes that letter indices i**0**, and i**1** are stored and accessed each time in process **827**, alternatively, the last two letters in text buffer **903** can be retrieved and assigned to indices i**0** and i**1**, respectively, as described above.

Update index process **1107** transfers to display letter process **1108**. Display letter process **1108** sends information to display module **712** which in turn generates the predicted letter on display screen **705**.

If the pressed key is not one of keys one to nine, i.e, is not a data entry key, processing transfers from check **1103** to rotate last entry **1109**. Recall that data key check **826** determined whether the pressed key was one of the zero to nine keys, or the # key. Thus, since checks **1101** and **1103** determined that keys zero to nine were not pressed, the only key press remaining is the # key, i.e., the rotate entry key, which indicates the user wants a letter different than the one entered last in text buffer **903**. In rotate last entry **1109**, the last character, i.e., the right most character, in text buffer **903** is replaced by the next character in the set of characters assigned to the last key pressed before the # key was pressed. Again, the use of the # key is illustrative only and is not intended to limit the invention to the use of that particular key to rotate an entry.

For example, if the last character in the text buffer **903** was a t and the # key is pressed, process **1109** changes the t to u. If the # key is pressed again, the u is changed to a v. Alternatively, if the last character in text buffer **903** was a u and the # key is pressed, process **1109** changes the u to a v. If the last character in text buffer **903** was a v and the # key is pressed, process **1109** changes the v to a t. If index i**1** is stored, as the last character in text buffer **903** is rotated, index i**1** is updated.

Text entry in cellular telephone **700** in different languages or contexts can be supported by using different letter frequency tables. For example, for plumbers, the prediction table can be based on text about plumbing procedures. For Frenchmen, the prediction table can be based on French text. Also, multiple letter frequency tables could be stored in cellular telephone **700**, or selectively transmitted to cellular telephone **700**, and a particular letter frequency table would be selected on an entry card.

In addition, an entry in the table can be more that a single letter, and thus save even more key strokes. For example, if the text buffer contains sche then typing a 3 could return dule rather than just d. Further, this novel method of text entry can be utilized with other than a cellular telephone. The method is applicable to any device that has several characters assigned to a single key on a keypad.

In the above embodiment, the English alphabet and a space element were used as the character set. Thus, the number **27** used in defining the table index is just the number N of characters in the set. Similarly, the number **9** used in defining the table index is just the number M of keys in the keypad that represent two or more different characters. Hence, predictive text entry method of this invention is not limited to text and is directly applicable to any keypad where each key represents a plurality of different characters.

In the embodiment of FIGS. **7**, **8**, and **9**, client module **702** and server module **749** communicate over CDPD network **710**. However, this architecture is illustrative only of the principles of the invention and is not intended to limit the invention to the particular architecture described. Client module **702** and server module **749** can use a wide variety of two-way data communication links to exchange resource locators, e.g., URLs, and TIL decks. For example, the communications link could be a switched voice circuit in which the client module and server module communicate using modems. Alternatively, the communications link could be any other packet switched network, so long as there is some way for client module **702** to get requests to server module **749** and for server module **749** to send data back to client module **702**. Further, a special purpose server could be used in place of HTTP server **749**. For example, the principles of this invention can be used over various data transport mechanisms including circuit switched data and packet switched data. These data transport mechanisms are being defined and implemented for most of the cellular network standards including GSM, TDMA, and CDMA.

In the configuration of airnet network **750** (FIG. **7**), client module **702** communicated directly with a server computer **743**. In another embodiment, as illustrated in FIG. **5**, the two-way data communication device first communicates with an airnet network translator **500** that in turn commu-

nicates with the appropriate server. In this embodiment, the operation of two-way data communication devices **100**, **101**, and **102** is similar to that described above for cellular telephone **700**, except the method field in the request generated in process **802** has a different form. For example, using the same information as before, the method field in this embodiment is:

GET http://www.libris.com/airnet/home.cgi?&cost=1 ANTP/1.0

The method field includes the full address of the server, the expected cost of the service, and the version of the protocol used for communicating with airnet network translator **500**. The two-way data communication device transmits the HTTP request including the complete URL to airnet network translator **500**.

FIG. **12** is a more detailed block diagram that illustrates the structures in one embodiment of airnet network translator **500**, according to the principles of this invention. In this embodiment, airnet network translator **500** is a computer running under the UNIX operating system with an interface to CDPD network **710**. Such computers are well known to those skilled in the art. Thus, herein only the structures and processes that must be added to such a computer are described.

Airnet network translator **500** supports internet protocol (IP) connections over CDPD network **710** and with each computer network with which translator **500** can interact. In this embodiment, each of the modules in network translator **500** are processes that are executed by the processor in the computer. Control module **1201** is a daemon that listens for transmissions over an IP connection from CDPD network **710**. When control module **1201** accepts a transmission, control module **1201** spawns an ANT request processor **1204**, which in this embodiment is a process, as indicated above. While in FIG. **12**, only one ANT request processor **1204** is shown, there is an ANT request processor spawned for each transmission that control module **1201** accepts and the ANT request processor remains active until the communication is terminated.

FIG. **13** is a process flow diagram that illustrates the operation of ANT request processor **1204**. This process flow diagram considers transmissions that utilize both TCP/IP and UDP/IP. However, the processes that are specific only to TCP/IP are enclosed in dashed-line boxes. Upon being spawned for a TCP/IP, in establish connection process **1300**, ANT request processor **1204** establishes a TCP connection using a TCP module in the server with the client module over CDPD network **710**. After the connection is established processing transfers from process **1300** to request received check **1301**.

If UDP is being used, upon being spawned ANT request processor **1204** initiates processing in request received check **1301**. In check **1301**, ANT request processor **1204** determines whether the request from cellular telephone **700** (FIG. **12**) has been received and stored in memory **1210**. Memory **1210** represents both RAM and non-volatile memory in this embodiment. When the request has been received and stored, processing transfers from check **1301** to retrieve data process **1302**.

In retrieve data process **1302**, ANT request processor **1204** retrieves information concerning the source of the URL, i.e., client module **702** of cellular telephone **700** from customer database **1213**, and the destination specified in the URL, i.e., the designated server, from server database **1212**. Both databases **1212** and **1213** are stored in memory **1210**. A customer record in database **1213** includes, for example, a carrier address, e.g., an IP number, an airnet network translator account number, billing information, and server subscriptions. A server record in database **1212** includes a server IP address, name, category, and class of service. Class of service refers to the pricing of the service, e.g., basic services, premium services, or pay-per-view services. Other pricing schemes can be supported in other implementations. When the information is retrieved for the server and service specified in the URL, and for the customer, processing transfers to valid request check **1303**.

In valid request check **1303**, ANT request processor **1204** determines, for example, whether client module **702**, i.e., the customer, is authorized to access airnet network translator **500**; whether client module **702** is authorized to access the server specified in the URL; whether the specified server is available through translator **500**; and whether the specified server supports the requested service. Thus, valid request check **1303**, validates the client, the server, and the client/server pair. Also, since an estimated cost is included in the request, the status and credit limits on the customer's account could be checked to determine whether the estimated cost is acceptable. If all of the checks are true, processing transfers to create HTTP request process **1306**. Conversely, if any one of the checks is untrue, valid request check **1303** passes information concerning the error to return error process **1304**.

Return error process **1304** launches a CGI program stored in memory **1210** based on the information received and passes appropriate information to the CGI program. The CGI program builds an appropriate PIDL deck describing the error and converts the PIDL deck to a TIL deck, as described above. When the TIL deck describing the error is complete, return error process **1304** transfers processing to log transaction process **1315** that is described more completely below.

If all the checks in valid request check **1303** are true, create HTTP request **1306** converts the request in memory **1211** to a request specific to the server specified, which in this embodiment is a HTTP request. For example, for the above request, create HTTP request process **1306** generates a method field, such as

GET /airnet/home.cgi?&client=xyz&cost=1 HTTP/1.0

In this embodiment, the method field includes the same information as in the embodiment described above, and in addition, the method field includes a client identification and the estimated cost.

After create HTTP request process **1306** is complete, ANT request processor **1204** accesses TCP module **1203** in establish server connection process **1307** for TCP/IP and transfers to secure transmission check **1308** for UDP/IP. In establish connection process **1307**, a connection is made between the server designated in the client request and the TCP interface module (not shown) so that data can be transmitted between airnet network translator **500** and the server. When the TCP connection to the server is established, ANT request processor **1204** transfers processing from establish server connection process **1307** to secure transmission check **1308**.

In secure transmission check **1308**, ANT request processor **1204** determines whether the HTTP request from the client requested a server that utilizes a protocol that supports encryption. If such a server was requested, processing transfers to negotiate process **1309** and otherwise to transmit request process **1310**.

In negotiate process **1309**, ANT request processor **1204** negotiates an encryption technique with the server. Upon completion of the negotiation, processing transfers from process **1309** to encryption process **1311**. In encryption

process **1311**, the HTTP request is encrypted using the negotiated encryption technique, and then processing transfers to transmit request process **1310**.

In transmit request process **1310**, the HTTP request is sent from memory **1210** to the HTTP server. When the transmission is complete, ANT request processor **1204** goes to result received check **1312**.

As described above, upon receipt of the request, the HTTP server services the request. Upon completion of servicing the request, the HTTP server returns either a PIDL deck or a TIL deck to airnet network translator **500**. The deck is stored in memory **1210**. If the server does not convert the PIDL deck to a TIL deck, the translation is done by airnet network translator **500**.

When the deck is received and stored, ANT request processor **1204** transitions from check **1312** to transmission completed process **1313** for TCP/IP and to secure transmission check **1314** for UDP/IP. ANT request processor **1204** closes the TCP circuit with the server in transmission completed process **1313**. Upon closing the server TCP connection, processing transfers to secure transmission check **1314**.

If the server utilized encryption, the deck stored in memory **1210** is encrypted. Thus, secure transmission check **1314** transfers processing to decryption process **1316** if encryption was used and otherwise to log transaction **1315**.

In decryption process **1316**, the encrypted deck is decoded and stored in memory **1210**. Also, after the decoding, if the deck must be converted to a TIL deck, the translation is performed. Decryption process **1316** transfer to log transaction process **1315**.

In log transaction process **1315**, ANT request processor **1204** writes a description of the transaction to transaction log **1211** in memory **1210**. In this embodiment, each transaction record includes a customer identification, a server identification, time required for the transaction, cost of the transaction, and a completion code. In one embodiment, for security purposes, each cellular telephone is assigned to only one customer and only one account.

After the transaction is logged, processing transfers to transmit result **1317**. In transmit result **1317**, ANT request processor **1204** returns the deck to client **702**. After the deck is transmitted, ANT request processor **1204** is terminated.

In one embodiment, if an airnet network translator is fully loaded and another transmission comes in, the translator returns the address of another airnet network translator and refuses the transmission. The cellular telephone transmits the message to the other airnet network translator. In yet another embodiment, all incoming transmissions are directed to a router. A plurality of airnet network translators are connected to the router. The router monitors the status of each translator. Each incoming transmission is routed to the least busy translator, which in turn responds to the transmission and performs the necessary operations for continuing communications with the client module.

In the above description of client module **702**, module **702** interacted with components within the cellular telephone to perform the various operations specified by the user. To insulate client module **702** from the exigencies of various cellular telephones to the extent possible, a general architecture for client module **702** is described more completely below. This general architecture is designed to have specific manager modules that interact with the modules described above within the cellular telephone and to provide standard information to the remaining manager modules within client module **702**. The manager modules with client module **702** form an interpreter that interprets TIL decks to generate a user interface; interprets data input by the user; and interprets the TIL decks so that the data input by the user is combined with an appropriate resource locator and either a message is sent to an appropriate server, or another local TIL deck is interpreted by client module **702**. While this embodiment is for a cellular telephone, the manager modules are generic and so are applicable to any client module in a two-way data communication device.

This approach limits the modifications that must be made to client module **702** to implement the principles of this invention in a wide variety of two-way data communication devices over a wide variety of two-way data communication networks. Also, in the above embodiment, client module **702** supported communications and interactions over the cellular telephone network. However, client module **702** can also support local services on cellular telephone **700**. Typical local services includes local messages, an address book, and preconfigured e-mail replies, or any combination of such services.

In this embodiment, client module **702** includes a plurality of manager modules including a navigation manager module **1401**, a network manager module **1402**, a TIL manager module **1403**, an archive manager module **1404**, a local manager module **1405**, an event manager module **1406**, a timer manager module **1407**, a user interface manager module **1408**, a memory manager module **1409**, and a device dependent module **1410**.

Navigation manager module **1401** handles card and deck navigation as well as managing any caches. Navigation manager module **1401** owns and manages a history list and as well as a pushed card list. In addition, navigation manager module **1401** functions as the main line of client module **702**; does all event distribution; and supports local services.

For local services, like local message store, there are two basic approaches that can be used. First, local services are implemented in a CGI-like manner. Each local service has an entry point which is called with an argument list. A TIL deck is returned via the event manager. From that point on, the TIL deck is processed in the standard manner. This approach limits local services to the same constraints as remote services. A less restrictive approach is to allow the local service to field events instead of the standard event loop. The local service would construct TIL cards on-the-fly and feed them to user interface manager **1406**. Note that the local service would need to cooperate with the standard event loop with regard to the history, the pushed card list, and any other state that is normally managed by the event loop. Table 4 is a listing of processes for the architecture for navigation manager module **1401**.

TABLE 4

ARCHITECTURE FOR NAVIGATION MANAGER MODULE 1401

```
ProcessEvents (void) ;
PushLocation (void * location, Boolean forStack) ;
void * PopLocation (Boolean forStack) ;
void * CurrentLocation( ) ;
struct LOCAL_SERVICE {
    char name[50] ;
    FUNC HandleEvent(Event * pevent) ;
    FUNC StartLocalService(void) ;
    FUNC StopLocalService(void) ;
    } ;
static LOCAL_SERVICE localServices[ ] = { . . . } ;
STATUS HandleEvent(Event * pevent) ;
STATUS StartLocalService( ) ;
STATUS StopLocalService( ) ;
```

Routine ProcessEvents is the main entry point for event processing in client module **702**. Typical events include key

presses on the keypad, choice selection for a choice card, text entry for an entry card, network events, and history events. Routine ProcessEvents can be called at any time to process an event or events. Routine ProcessEvents does not return until all events on a queue generated by event manager module **1406** are processed. If a local service is running, events are distributed to the local service before being processed by routine ProcessEvents.

The remaining routines in Table 4 are called internally to navigation manager module **1401** and by local services. Routine PushLocation pushes a location on the history list and issues a request for that location. The forstack flag indicates a stack push of local cards.

Routine *PopLocation pops a location on the history stack and issues a request for the top location of the history stack. In routine *PopLocation the forStack flag indicates that all cards since the last stack push should be popped.

Routine *CurrentLocation returns the current location the current URL being displayed.

As shown in Table 4, each local service provides a number of functions. If a local service is running, function HandleEvent, the local service's event handler, is called before any processing by navigation manager module **1401**. If the event is handled by the local service, the event is not processed any further.

Function StartLocalService is the local services start function. Function StartLocalService is called before any events are distributed to the local function. Similarly, function StopLocalService is the stop function for the particular local service. Function StopLocalService is called when no more events are distributed to the local service.

Network manager module **1402** insulates the rest of client module **702** from the specific networking protocol used over the cellular telephone network. Network manager module **1402** delivers requests to the server specified in the URL via the cellular telephone network interface; segments responses from the server for lower latency; delivers responses from local services to navigation module **1401** via event module **1406**; handles request/response cycle (e.g. cancellation, retry strategy) with the server over the cellular telephone network; can receive asynchronous messages from the server; performs memory management of TIL decks; performs caching of TIL decks; handles all negotiations concerning protocols and server scaling with the server; handles any encryption for information exchanged between cellular telephone **700** and the server.

In some cellular telephone, the maximum message size is fixed. However, for UDP and TCP messages, a more direct interface is used that bypasses this limitation of message passing. It is important to avoid copying network data from memory buffer to memory buffer as such copying increases the memory "high water mark" as well as decreases performance. Since different cellular telephones have different interfaces for delivering network data, network manager module **1402** manages the network data. In this way, network data is only copied from the network buffer for long-term storage.

When a message or reply arrives, network manager module **1402** uses event manager module **1406** to report that fact. However, access to the data by other manager modules in client module **702** is through a protocol that allows storage of data in a variety of fashions on different telephones. Any transparent, short-term caching of TIL data is handled by network manager module **1402**. Table 5 is one architecture for network manager module **1402**.

TABLE 5

SPECIFICATION FOR NETWORK MANAGER MODULE 1402

```
typedef short TID;
void NM_Init(void) ;
void NM_Terminate(void) ;
TID NM_SendRequest (void *requestData, int length,
    Boolean ignoreCache) ;
NM_CancelRequest (TID TRANSACTIONId) ;
NM_DataType(TID TRANSACTIONId) ;
NM_GetData(TID TRANSACTIONId, void *data, int
    *length, Boolean *complete) ;
void *NM_HoldData (TID TRANSACTIONId) ;
NM_ReleaseData(TID TRANSACTIONId) ;
TID NM_StartData(int data Type, char *requestData,
    int length) ;
STATUS NM_EndData(TID TRANSACTIONId) ;
STATUS NM_SetDataLength (TID TRANSACTIONId, int
    length) ;
STATUS NM_GrowDataLength (TID TRANSACTIONId, Int
    grow) ;
int NM_GetDataLength(TID TRANSACTIONId) ;
void *NM_GetDataPointer (TID TRANSACTIONId) ;
STATUS NM_DeliverData (TID TRANSACTIONId) ;
```

Network manager module **1402** identifies each network data transaction by a 16-bit transaction identification code TID. Network manager module **1402** increments transaction identification code TID by one for each new transaction. Transaction identification code TID rolls over after 0xffff.

Routine NM_Init initializes network manager module **1402** and so is called before any other calls in network manager module **1402**. Routine NM_Terminate closes processing of network manager module **1402** and so is called after all other calls in network manager module **1402**.

Network manager module **1402** uses routine TID NM_SendRequest as the standard process of sending a request to the server. Pointer *requestData in the call to routine TID MN_SendRequest is defined by the server protocol. Similarly, the state, e.g., the Boolean value, of variable ignoreCache is used to indicate whether any cached replies should be ignored. After sending the request, this routine returns a server transaction identification code TRANSACTIONId. A local service can also send a request to the server.

When the user instructs client module **702** to cancel a request, network manager module **1402** calls a routine NM_CancelRequest with cellular telephone transaction identification code TID and server transaction identification code TRANSACTIONId. Routine NM_CancelRequest issues a command to the server to cancel the specified request.

When data are received from the network, the data can be either a response to a request sent by routine TID MN_SendRequest, or by a local service. Thus, in response to receiving data from the server, network manager module **1402** generates an event that includes server transaction identification code TRANSACTIONId and the type of data DATAType. For replies to requests sent by routine TID MN_SendRequest, server transaction identification code TRANSACTIONId is the same as the one returned by the matching call to routine TID MN_SendRequest and data type DATAType indicates that the data is a response. For local service originated messages, server transaction ID is new, and data type DATAType depends on whether the data is an e-mail, pushed TIL, or another type.

After the network event is received by event manager module **1406**, and navigation manager module **1401** distributes control of the event to network manager module **1402**, network manager module **1402** users the server transaction

identification code TRANSACTIONId and the remaining routines in Table 5 to process the data.

Routine NM_DataType is used to return the particular data type dataTYPE, e.g, reply, MIME, server push, etc. Routine NM_GetData sets a pointer to the data identified by server transaction identification code TRANSACTIONId, retrieves the length of the data, and determines whether all the data has been received. The interface provided by this routine allows the first part of a data stream, e.g. the first card of a TIL deck, to be processed by client module **702** before the rest of the deck is received.

Routine NM_HoldData is called before calling NM_routine NM_GetData to hold the data and thus insure that the data remains valid during processing by client module **702**. If the data is not held, the data can be deleted or moved with the internal buffers of network manager module **1402**. If the data is held, routine NM_ReleaseData is called after network data has been processed to release the data.

Routines TID NM_StartData, NM_EndData, NM_SetDataLength, NM_GrowDataLength, NM_GetDataLength, NM_GetDataPointer, and NM_DeliverData are used internally by network manager module **1402**, and by local services to deliver data. By allowing local services to use these routines, the same buffers can be used to store both network and locally generated data thereby reducing the amount of memory required to support client module **702**.

Routine TID NM_StartData creates a new data transaction and triggers a data delivery event. Routine NM_EndData is called when all data for the given server transaction identification code TRANSACTIONId has been transmitted. Routine NM_SetDataLength sets the data segment to a given length and may cause the location of the data to change. Routine NM_GrowDataLength grows the data segment by a given length and also may cause the location of the data to change. Routine NM_GetDataLength returns the length of the data segment. Routine NM_GetDataPointer returns a,pointer to the data. This routine is preferably called before writing into the data buffer. Also, this routine is preferably called whenever the data's location may have changed. Routine NM_DeliverData can be called when at least one card has been stored to reduce latency while the other cards are being generated.

TIL manager module **1403** insulates the rest of client module **702** from changes to the TIL specification. The interface provided by TIL manager module **1403** has the following characteristics: removes the need for parsing by the rest of client module **702**; uses cursors to avoid generating data structures on-the-fly; does not need an entire deck to operate; and handles TIL versioning.

Each TIL deck contains a major and a minor version number. The minor version number is incremented when TIL changes in a way that does not break existing TIL manager modules. The major version number is incremented for non-compatible versions of TIL.

Each TIL deck has the same hierarchy. One embodiment of this hierarchy is presented in Table 6. In Table 6, indentation is used to represent the relationships of the various hierarchical levels.

TABLE 6

TIL DECK HIERARCHY

```
deck
    options
    softkeys
        options
    card
        options
        softkeys
            options
        formatted text
            formatted lines
        entries
            options
formatted line
```

The interface presented in Table 7 for TIL manager module **1403** is designed with the assumption that TIL is a direct tokenization of PIDL as described in Appendix I. However, the interface does not have any dependencies on that tokenization and can support other PIDL encoding techniques. Given the above assumption, the opaque pointers described below are actual pointers into the TIL deck itself. A rudimentary object typing scheme based on where in the deck the opaque pointer points can be used to implement the generic functions described below. If this object typing is not feasible due to details of TIL encoding, the generic functions can be replaced with specific functions.

TABLE 7

ARCHITECTURE FOR TIL MANAGER MODULE 1403

```
typedef char *opaque;
typedef opaque Deck;
typedef opaque Card;
typedef opaque Text;
typedef opaque Entry;
typedef opaque Option;
typedef opaque SoftKey;
typedef opaque Object;
/* Generic functions */
FirstOption(Object obj, Option *o) ;
   /* obj is a card, softkey, entry, or deck */
GetSoftkey(Object obj, Option *o) ;
   /* obj is a card or deck */
GetText(Object obj, Option *o) ;
   /* obj is a card or entry */
/* Deck functions */
SetDeck(Deck d, int length) ;
   /* tells module which deck to use */
DeckGetCard(Card *c, int num) ;
-or-
DeckGetCard(Deck d, Card *c, int num) ;
/* Card functions */
int CardType(Card c) ;
CardFirstEntry(Card c, Entry *e) ;
CardLookupSoftkey(Card c, int num, Softkey *s) ;
CardIsLast(Card c) ;
/* Option cursor functions */
OptionNext(Option *o) ;
char *OptionKey(Option o) ;
char *OptionValue(Option o)
/* Entry cursor functions */
/* Text (and image) cursor functions */
TextNextToken(Text *t, int *type, int *subtype,
    int *length, char *data) ;
```

Archive manager module **1404** stores and retrieves long-lived information. This information includes: data related to the server's location and/or required to support server scaling; data related to encryption; TIL caching (transparent to user); TIL storage (specified by user); and message storage and retrieval (see local manager module). Archive manager

module **1404** should support a variety of nonvolatile memory schemes that are provided by the two-way data communication devices.

Local manager module **1405** is an interface to local device resources, such as local messages, address book entries, and preconfigured e-mail replies. Local manager module **1405** should also define an abstract interface to navigation manager module **1401** for use by archive manager module **1404**.

Table 8 is an architecture for an interface within local manager module **1405** to access to an address book stored on cellular telephone **700**. The name of a routine in Table 8 is descriptive of the operations performed by the routine.

TABLE 8

ARCHITECTURE FOR ADDRESS BOOK ACCESS

```
int NumAddresses();
char *AddressName(int num);
char *AddressGetEMail(int num);
    // returns e-mail address
char *AddressGetPhone(int num);
    // returns phone number
char * AddressGetFax(int num);
    // returns fax number
SetAddress(int num, char *name, char *email,
        char *phone, char *fax);
DeleteAddress(int num);
InsertAddress(int before);
```

Table 9 is an architecture for an interface within local manager module **1405** to access predetermined replies stored on cellular telephone **700**. The name of a routine in Table 9 is descriptive of the operations performed by the routine.

TABLE 9

ARCHITECTURE FOR PREDETERMINED REPLY ACCESS

```
int NumReplies();
char * GetReply(int num);
DeleteReply(int num);
SetReply(int num, char *text);
InsertReply(int before);
```

Table 10 is an architecture for an interface within local manager module **1405** to access messages stored locally on cellular telephone **700**. The name of a routine in Table 10 is descriptive of the operations performed by the routine.

TABLE 10

ARCHITECTURE FOR LOCALLY STORED MESSAGE ACCESS

```
int NumMessages();
void *FirstMessage();
void *NextMessage();
int MessageType (void *msg);
    // e.g. e-mail, TIL, etc.
void *MessageContent(void *msg);
void *SaveMessage(int type, void *content, int
                contentLength);
DeleteMessage(void *msg);
```

Event manager module **1406** handles the distribution of events. In this embodiment, events include low-level events like key presses and higher level navigation and user interface events. There are typically only a small number of events at any one time. The main event loop in the two-way data communication device dependent module keeps calling EM_GetNextEvent( ) until no events are left in the queue. Note that processing one event can cause another event to be pushed onto the queue. The main event loop is not restarted until another event is pushed onto the queue due to a user key press or a network event.

In this embodiment, the event types include:

1) keypad events, i.e., pressing of a key;
2) choice events relating to a current choice card, e.g., the user selecting choice three;
3) text entry events relating to a current entry card, e.g., the user keying in "Hello";
4) network events, e.g., response arrived, request arrived, transaction terminated, network status; and
5) history events, e.g., pop, pop to marker.

Table 11 is an architecture for event manager module **1406**. As in the other tables herein, the name of a routine in Table 11 is descriptive of the operations performed by the routine and in addition a brief description is given in the comment field.

TABLE 11

ARCHITECTURE FOR EVENT MANAGER MODULE 1406

```
struct Event {
    int type;
    void *data;
        /* e.g. keycode, choice num, entry
            text, status code, other data */
}
EM_QueueEvent(int type, void * data);
    /* Adds event at end of queue*/
EM_GetNextEvent(Event * event);
    /*Pops next event*/
EM_PeekNextEvent (Event event);
    /*Peeks at next event*/
```

Timer manager module **1407** allows timer events to support timeouts, animation, and other time-domain features. Timeouts are delivered via event manager module **1406**.

Table 12 is an architecture for timer manager module **1407**. As in the other tables herein, the name of a routine in Table 12 is descriptive of the operations performed by the routine.

TABLE 12

ARCHITECTURE FOR TIMER MANAGER MODULE 1407

```
TimerInit();
int TimerSet(int milliseconds, int code, void
                *clientData);
    /*Returns a timer identification timerId to
        be used for cancellations*/
TimerCancel(int timerId);
TimerCancelAll()
```

User interface manager module **1408** handles interactions with the keypad and the display. Each of the three types of user interfaces defined in Table 1 above requires a different version of user interface manager module **1408**. For most cellular telephones, only one card at a time is used. However, some cellular telephones can display multiple cards at once and so would require a different version of user interface manager module **1408** from the version that handled display of only one card at a time.

In this embodiment, user interface manager module provides a user interface for the three types of cards display, choice, and entry; provides hooks for custom user interfaces for the address list and e-mail reply entry; only cares about the user interface aspects of cards and provides no navigation, argument, or option processing; handles all text

and graphic layout including word wrapping; handles scrolling of text; operates from PIDL data structures; generates keyboard events, some of which may be generated by soft keys; and generates high-level events, e.g. next card, choice entry **3**, text entry "IBM".

Table 13 is an architecture for processing cards by user interface manager module **1408**. As in the other tables herein, the name of a routine in Table 13 is descriptive of the operations performed by the routine.

TABLE 13

ARCHITECTURE FOR CARD PROCESSING BY UI MANAGER MODULE 1408

```
void UI_StartCard(Card c);
  /* called to begin display and processing of
       a given card*/
void UI_EndCard(Card c);
  /*called when a card is no longer to be
       displayed*/
Boolean UI_HandleEvent(Event *pevent) ;
  /*returns true if the event is handled, false
       if not*/
```

Table 14 is an architecture for the user interface implementation by user interface manager module **1408**. As in the other tables herein, the name of a routine in Table 14 is descriptive of the operations performed by the routine.

TABLE 14

ARCHITECTURE FOR UI IMPLEMENTATION BY UI MANAGER MODULE 1408

```
UI_LayoutCard(Card c, Boolean draw, Proc)
                              callback)
  /* relies on global data; needs to be able
       to: draw as it goes; and note the
       special function of the currentLine
       (e.g. none, choice, softkey)*/
int numLines, firstVisible, lastVisible,
                                  currentLine;
char currentEntry[80];
int currentChoice;
void *currentSoftkey;
Card currentCard; and
  . . . other info as needed for in-line
       scrolling
```

The callback routine is notified of the special function of each line as the line is laid out. Thus, routine UI_LayoutCard can be used to scroll to a particular choice. If the current line is too wide to display all at once, horizontal scrolling is used to display the complete line, one display width at a time.

Memory manager module **1409** is optional, and is used in two-way data communication devices that do not support dynamic memory allocation. In these devices, all memory allocation and releases must go through memory manager module **1409**. Also, by allocating memory in advance via memory manager module **1409**, client module **702** does not run out of memory due to some other process on the device using up memory.

Microfiche Appendix A is a computer source code listing in the C++ computer language of one embodiment of a client module within a cellular telephone, and one embodiment of an airnet network translator that was used with an Internet server to communicate with client module. The Internet server was a UNIX computer running the Mosaic HTTP server. The source code was used to generate executable code by compiling the source code on a computer running the Sun Microsystems Operating System Solaris 2.4 using Sun Microsystems compiler SunPro C and C#, and the Sun Microsystems SDK make utility. All of these products are available from Sun Microsystems of Mountain View, Calif.

This application is related to copending and commonly filed U.S. patent application Ser. No. 09/332,436 entitled "A PREDICTIVE DATA ENTRY METHOD FOR A KEYPAD" of Alain Rossmann, which is incorporated herein by reference in its entirety.

Various embodiments of a novel interactive two-way data communication system, a two-way data communication device, an airnet network architecture, and a predictive text entry system have been described herein. These embodiments are illustrative only of the principles of the invention and are not intended to limit the invention to the specific embodiments described. In view of this disclosure, those skilled in the art will be able to use the principles of this invention in a wide variety of applications to obtain the advantages of this invention, as described above.

APPENDIX I

A Description of PIDL and TIL

The main structure of PIDL is described by an abstract syntax. This appendix describes the elements of the language and their semantics. In the syntax description of each element, an element is defined in an enhanced BNF.

a :: = b	the element a is defined as b
a :: = b :: = c	the element a is defined as b or c
b c	the element b followed by element c, the intervening space is just for clarity
a|b|c	element a or element b or element c
{a}	the element a is optional
a :: = bc	the element a is defined as b or c
{a}*	the element a may appear zero or more times in a row
{a}+	the element a may appear one or more times in a row
abc	the characters abc literally
ol(a)	an option list with zero or more topions of the element a, see Options below

In general, the element blank-space can optionally appear between any two other elements. To keep the diagram clear, it has been omitted except where required. Where a blank-space is illegal or treated specially, it is noted.

The PIDL Elements

deck ::=deck-header {softkey}* {card}+deck-footer

deck-header ::=<PIDL ol(deck-options)>

deck-options ::=o-args |o-cost|o-ttl

deck-footer ::=</PIDL>

A deck consists of one or more cards. There must be at least one card. A deck may also have a number of softkeys defined that stay in force for the whole deck. See soft keys below for the syntax and full description.

o-cost ::=cost=value

o-ttl ::=ttl=integer

Additional arguments to be passed on the next deck request are given in o-args. See Arguments below for syntax and full description.

The cost of retrieving this page (exclusive of telephone system charges) is represented in o-cost. If no o-cost

is given, the deck cost is included with the user's standard service contract.

Decks can be cached by the cellular telephone for a period of time. The o-ttl entry indicates the number of seconds that the deck can be cached from time of reception. If no o-ttl entry is given, the deck can only be cached for short periods of time, for example, to implement a back function similar to that of most Web browsers. If the value of o-ttl is zero, the deck must not be cached.

Card Elements

card ::=display-card|choice card|entry-card

A card is one of three types of card in this embodiment. These are described in the sections below.

card-options ::=o-name|o-next|o-prev

o-name ::=name=identifier

o-next ::=next=destination

o-prev ::=prev=destination

All cards can have these options. The optional o-name option gives a name to the card. If a card has a name, the card can be referred to by a destination.

The o-next and o-prev give destinations for the NEXT and PREV keys. If omitted, the defaults are the next and previous sequential card in the deck. If o-prev is omitted from the first card, the PREV key returns to the deck last visited. If o-next is omitted from the last card, the NEXT key returns to the first card of the current deck. However, this default behavior is only a fail-safe: the last card in a deck should always have either an o-next option, or be a choice card where each choice entry indicates a new destination.

Display Card

display-card display-header display-content display-footer

display-header ::=<DISPLAY option-list>

display-options ::=card-options

display-content ::={softkey}* formatted text

display-footer ::=</DISPLAY>

Display cards give information for the user to read. See Formatted Text below for a full description of the format of information that can be displayed.

Softkeys can be described for this card only, see Softkeys below.

Choice Card

choice-card ::=choice-header display-content {entries} choice-footer

choice-header ::=<CHOICE ol{choice-options}>

choice-options ::=card-options|o-method|o-key | o-default

o-method ::=method=method-type

method-type ::=number|list|alpha|group

entries ::={choice-entry}+

::=group-entry {choice-entry}+)+

choice-footer ::=</CHOICE>

Choices let user pick one from a list. The initial display content is shown to the user, followed by the choices. Each choice can have one line of formatted text (which may be wrapped or scrolled by the phone if too long).

How the choices are displayed and chosen is based on the o-method option. Note that this option is a hint only, and can be disregarded by the phone. The number method is the default and indicates that the choices are numbered sequentially from one and are chosen by pressing the appropriate digit on the keypad. If there are more than nine options, the phone may choose some other method of selection. The list method indicates that the list should be unnumbered and that the user should scroll through the list and hit some designated enter key to choose an entry. The alpha method is like list, only it is an indication that the text of the entries should be used to aid selection if at all possible. In this case, the entries are assumed to be alphabetically sorted. The group method is described in more detail below.

The o-key option indicates, if present, the key of an argument to be added to the argument list. See Arguments below for more information. The value of the argument comes from the choice entry; see below. The o-default option indicates the default value if the user just hit ENTER. See o-default under Entry Card below for more information.

choice-entry ::=<CE ol(entry-options)>formatted-line

entry-options ::=action-options|o-value

Each choice has text displayed to the user. If the action-options are given, the indicated action is performed if the choice is made. If the o-value option is present, it supplies the value to the argument identified with the o-key option in the choice header. If no o-value is given, the text of the entry is used (without any formatting) as the argument value.

group-entry ::=<GE ol(group-options)>

group-options ::=label=value

If the group method is used, the choices are divided into a number of groups. Each group is headed by a group-entry, which, via the label option, gives a short name to the group. The phone can then give the user a hierarchial interface for choosing among a large number of choices. The text of the label should be limited to eight characters and may be truncated by the phone.

Entry Card

entry-card ::=entry-header display-content entry-footer

entry-header ::=<ENTRY ol(entry-options)>

entry-options ::=card-options|o-format|o-key | o-default

entry-footer ::=</ENTRY>

Entries let the user enter a value. The display content is shown to the user, followed by an entry line. The user's entry is controlled by the format. The o-key option indicates the argument that is being set by this entry. The value of the argument are the user's entry.

o-format ::=format=value{; format-hint}

format-hint ::=value

This option specifies the format for user input entries. The string consists of format control characters and static text which is displayed in the input area. Most of the format control characters control what data is expected to be keyed in by the user. They are displayed as blanks until the user types into them.

The format codes are:

A entry of any alphabetic character

9 entry of any numeric character

X entry of any alphabetic or numeric character

*f allow entry of any number of characters; the next character, f, is one of A, 9 or X and specifies what kind of characters can be entered.

\c display the next character, c, in the entry field; allows display of the formatting characters in the entry field.

Format hints indicate what kind of value is expected. If a format hint is not understood, it is ignored. Currently defined format hints are:

text text is expected to be text, use special input techniques; generally follows *A or *X

mail-reply like text, but expected text is for an e-mail message or page; may affect input algorithm

address-list entry is a list of e-mail addresses

o-default ::=default=value

The o-default option supplies a value that is used if the user simply hits NEXT. If no default value is given, then the user must supply a value.

Formatted Text

formatted-text ::={{flow-image}{line-format}text-line)*

formatted-line ::=text-line

text-line ::={text|image|text-format|alignment-format}*

Formatted text is what is shown to the user in most cards. Formatted lines are used for choice entries.

text-format ::=<B>|<I>|<BL>

::=B>/B<|>|<BL>

The format codes control Bold, Italic and Blinking. The slash versions cancel the formatting. Unlike HTML, these needn't be strictly nested and over application and over cancellation are tolerated. Formatted-text and formatted-line elements start in plain mode (no bold, italic, or blinking).

alignment-format ::=<CENTER>|<BOLD>|<TAB>

The alignment codes specify how parts of a line are to be laid out. The text following the alignment code is either centered or right justified on the same line as the other text. The text or image following the code is considered to be all text up to the next alignment code or line break. All lines start implicitly aligned left. Note that these do not include an implicit line break so that one can have both left and right justified text on a single line. If there is too much text and not enough room on the line then, if in wrap mode, the non-fitting text is moved to the next line and aligned the same way. If in line mode, the line may end up running together with two spaces between the left, center, and right justified segments.

The tab code is used to create aligned columns. Rather than tab to specific character positions, the tab code separates the text for each column. The width of the column is determined by the maximal width of the text (or images) in each line. The extent of the columns is from the first line with tab codes through the last contiguous line with tab codes. Some lines may have fewer tab codes than others, in which case they are assumed to have no text for the extra columns.

line-format ::=<WRAP>|<LINE>

::=

Multiple lines of text are separated by the
 code. If a line is too long to fit on the screen and, if in wrap mode, the line is word wrapped onto multiple lines. If in line mode, the line is left as one line and is scrolled horizontally. Formatted-text and formatted-line elements start in wrap mode and may be changed with either the <WRAP> or <LINE> codes. These codes are an implicit line break.

Images

image ::=<IMAGE ol(image-options)>

::=<INLINE ol(image-options)>

inline-data </INLINE>

flow-image ::=<IMAGE ol(flow-image-options)>

::=<INLINE ol(flow-image-options)>

inline-data </INLINE>

image-options ::=o-source

flow-image-options ::=image-options|o-flow

inline-data ::=ASCII85 encoding of image data

Images are treated as large words and, by default, are simply displayed as part of the text. Flow-Images have a flow option that causes them to be treated differently. The image data is stored in a separate data stream as identified by the source option.

Inline images are treated identically, only the data is part of the current data stream. ASCII85 is a standard way of encoding binary data in printable ASCII, whereby each four bytes of data is encoded in five characters. Note that TIL only uses inline images, and uses a different encoding.

o-source ::=src=location

This option specifies the location of the source for images.

o-flow ::=flow={left|right}

This option controls the alignment of flow-images. The option specifies that the image is flush left or flush right with the screen. Subsequent lines of text flow in the remaining right or left hand space.

Softkeys

softkey ::=<SOFTKEY ol(softkey-options)>

softkey-options ::=o-label|o-button|action-options

Softkeys supply definitions for two buttons known as SOFT-L and SOFT-R. They do not show up in the normal text and graphics area displayed to the user, but on a separate line for soft key labels. (Note: in some implementations, where screen real estate is scarce, this label line may get used for normal text and graphics display when there are no softkeys defined on the current card).

When the softkey is pressed, the indicated action takes place.

o-button ::=button=side

side ::=left|right

o-label ::=label=value

The button option specifies which physical key the softkey applies to. The label option is the text that is displayed on screen for that key. The phone may truncate the label. It is suggested that labels be fewer than eight characters.

Softkeys can be specified both for the deck as a whole and per card. When specified for the deck (after the deck-header, but before the first card) they remain in effect for the entire deck. When specified for a card (at the beginning of the formatted-text for the card), they temporarily override any deck softkeys while the card is visible. Note that the override is done independently for the two keys (a card can override one softkey, but not the other). To override a deck softkey with no softkey (in effect, to remove a softkey for the duration of a card) use a softkey with no label and no action.

Syntax: Options

Many of the syntactic elements of PIDL have option lists associated with them. Options refine the operation of the elements they are part of. Unless otherwise noted, options do not nest, even when the same option is given in two nested elements. Options that are not defined for an element are ignored, even if valid for an enclosing element.

ol(valid-option) ::={blank-space valid-option}* {blank-spaces}

option-list ::=ol(option)

option ::=key =value
key ::=identifier
value ::=plain-text
::="{text}"
An option list contains zero or more options. Each option is separated by blank-space (required!) and optionally followed by blank-space. In the syntax diagrams, option lists are shown as: ol (valid-option) where valid-option is replaced with an element that defines the possible options in this context. ol is a generic syntactic description of option-lists.
Each option is a key and a value. They may be given in any order within the list of options. The key is always an alpha-numeric name that is case insensitive. The value, if it is composed of only alphanumeric characters, may appear directly after the equals sign. Otherwise, the value must be quoted. In quotes, blank-space is treated literally and is considered part of the value.
In the syntax diagrams, the possible values for various options are specified without quotes. However, quotes are always acceptable around an option value.
Unlike almost all other syntactic elements, blank space is not permitted between the key and the equals sign or between the equals sign and the value.
Many options have a more restricted set of possible values than represented by the above syntax. See the individual options for details.

Destinations

destination ::=location {; animation}
::=card-loc {; animation}
::=stack-operation {; animation}
location ::=full-loc|partial-loc
|relative-loc
full-loc ::=: service-id / deck-path
::=: service-host / deck-path
partial-loc ::=/ deck-path
relative-loc {../}* deck-path
card-loc ::=# identifier
deck-path ::=plain-text {/ plain-text}*
service-id ::=% plain-text
service-host ::=plain-text
Destinations are used in some options to indicate the next, or previous deck or card to show. A deck is specified either with a full location (service-id and deck-path), just a deck-path (in which case the service is the same as the current deck's service), or a relative deck-path. In the later case, the last component of the current deck-path is removed, (and one additional component for each ../ in the relative deck-path), and the deck-path appended.
A particular card can be a destination and is specified by a card-loc element.
stack-operation ::=+card-loc
::=–
In addition to the normal history list of where a user has been that is kept by a phone, the phone also keeps a short stack of locations. Using a plus sign form causes the current location (deck and card, and location in the history list, and animation used) to be pushed on the stack before going to the new card. Using the minus sign form causes a return to the location on the top of the stack, and the history list to be pruned back to the saved point. If no animation is given, the inverse animation is used. The stack is popped.
animation ::=slideN|slides
::=slideW|slideE
::=slideSW|slideNE
::=slideSE|slideNW
::=flipV
::=flipH
::=fade
::=none
The optional animation argument indicates what form of screen animation, if available, is to be used when going to the destination. The animation is remembered with the destination in the history and destination stack. If the user moves to a destination via a 'go' or 'next' operation, then the animation is performed. If the user moves to a destination via a 'prev' or 'pop' operation, the reverse animation associated with the current location is performed.

Actions

Choice entries and soft keys can specify actions to be performed when the user selects the choice or softkey.
action-options ::=o-args|o-call |o-page
o-go ::=go=destination
o-call ::=call=value
The go operation indicates that the destination should be moved.

Argument Processing

Each time a deck is requested, arguments may be passed along with the request. These arguments may be used by the service end to compute a deck specific for the user rather than just return a pre-written deck.
Arguments are built-up as the user traverses the deck. Each argument is a key-value pair. While arguments superficially look like options, these two entities are quite distinct: Options are a part of PIDL and affect the operation of the phone. Arguments are information gathered by the phone and returned to the service. Neither PIDL nor the phone understands the arguments beyond their basic syntactic structure.
Arguments come from three places: Choice cards, Entry cards, and the args option. Each of these specifies a key-value pair that is added to a buffer of arguments to be sent. In the case of the args option, multiple arguments may be specified. When an argument key-value pair is added to the argument buffer, if the key is already present in the buffer, its value is replaced.
o-args ::=args=arg-list
argument-list ::=arg-key-value{{&|; }
key-value}*
arg-key-value ::=arg-key=arg-value
arg-key ::=identifier
arg-value ::=plain-text
Note: The entire o-args element is actually the value of an option. If it has more than one arg-key-value it will need to be in quotes. Since the ampersand (&) and semi-colon (;) are used as key-value pair separators, these characters cannot be part of argument values.
o-key ::=key=arg-key
o-value ::=value=arg-value
These options are used in choice and entry cards to specify the key and value for the arguments those cards set.

Basic Elements

alpha ::=any alphabetic character
numeric ::=any digit
alpha-numeric ::=alpha|numeric

hex ::=numeric|any letter A through F, either case

blank-space ::={space|tab|new-line}+

space ::=the space character

tab ::=the tab character

new-line ::=the carriage return character
::=the line feed character
::=the sequence carriage return, line feed

word ::={alpha-numeric}+

identifier ::=alpha {alpha-numeric}*

integer ::={+|–}{numeric}+

text ::=any 7-bit ASCII character except<, >, ', or &
::=> |< |" |&
|
::=any ISO-Latin-1 named entity
::=&# hex hex;
In text, runs of blank-space are treated as single spaces and my be used as point for word wrapping.

plain-text

safe ::={alpha|numeric|safe}*
::=$|–|_|@|.|& |!
|*|,

TIL Encoding

Except where noted, TIL is identical to PIDL in structure. To translate PIDL to TIL several steps are conceptually needed (these may be done in one pass by a translator):

1. Escape characters with the high bit set.
2. Compress or remove all blank space where possible.
3. Tokenize comment elements with a single byte with the high bit set.
4. Inline images.

Fundamentally, TIL is just PIDL with certain common character sequences replaced by single bytes with the high-bit set. The first two steps above support this. Additionally, images are further compacted by including them inline in a dense format.

The tokenizing follows the encoding given in the table below. Note that for purposes of element separation, the tokens that represent option key identifiers (with the equal sign) can be considered to include all preceding blank space. Similarly, the tokens that represent option values can be considered to include all following blank space.

<PILD>	90	args=	C0	alpha	E0
</PIDL>	91	button=	C1	center	E1
<DISPLAY>	92	call=	C2	fade	E2
</DISPLAY>	93	cost=	C3	flipH	E3
<CHOICE>	94	default=	C4	flipV	E4
</CHOICE>	95	flow=	C5	group	E5
<ENTRY>	96	format=	C6	inline	E6
</ENTRY>	97	go=	C7	left	E7
<CE	A0	key=	C8	list	E8
<GE	A1	label=	C9	none	E9
<IMAGE	A2	method=	CA	number	EA
<INLINE	A3	name=	CB	right	EB
<SOFTKEY	A4	next=	CC	slideE	EC
<B>	B0	page=	CD	slideN	ED
</B>	B1	prev=	CE	slideNE	EE
<I>	B2	src=	CF	slideNW	EF
</I>	B3	ttl=	D0	slideS	F0
<BL>	B4	value=	D1	slideSE	F1
</BL>	B5			slideSW	F2
<CENTER>	B6			slideW	F3
<RIGHT>	B7				

-continued

<WRAP>	B8
<LINE>	B9
 	BA

APPENDIX II

A Computer Program to Generate a Letter Frequency Tables for Use in the Predictive Data Entry Process Unpublished © 1995 Libris, Inc.

```
/*  This program opens a text file selected by the
    user, generates the frequency table for that file,
    and then writes the frequency table to another
    file also selected by the user.
*/
#include <stdio.h>
#include <string.h>
#include <console.h>
#include <assert.h>
typedef unsigned char byte;
typedef byte triplet[3];
typedef byte tristorage[27][27][27];
IncrementTrigram(triplet t, tristorage trigrams)
{
  byte * pb;
  assert(t[O] < 27);
  assert(t[1] < 27);
  assert(t[2] < 27);
  pb = &trigrams[t[O]][t[1]][t[2]];
  if (*pb < 255) *pb = *pb + 1;
  return *pb;
}
StoreTrigramValue(triplet t, tristorage trigrams, byte
value)
{
  assert(t[O] < 27);
  assert(t[1] < 27);
  assert(t[2] < 27);
  trigrams[t[O]][t[1]][t[2]] = value;
}
byte FetchTrigramvalue(triplet t, tristorage trigrams)
{
  assert(t[O] < 27);
  assert(t[1] < 27);
  assert(t[2] < 27);
  return trigrams[t[O]][t[1]][t[2]];
}
byte DumpTrigram(triplet t, tristorage trigrams)
{
  byte value;
  assert(t[O] < 27);
  assert(t[1] < 27);
  assert(t[2] < 27);
  value = FetchTrigramValue(t, trigrams);
  if value != 0)
  {
    printf("%c%c%c = ", t[O] + 'a', t[1] = 'a', t[2] +
'a');
    if (value == 255) printf("***");
    else      printf("%3d", value);
}
return value;
}
int IdFromChar(short c)
{
  c = tolower(c);
  if (c < 'a' || c > 'z') return 26;
  return c – 'a';
}
AddChar(tristorage trigrams, triplet t, byte b)
{
  byte value;
  unsigning short r;
```

-continued

```
    assert(b <= 26);
    if (b == 26) { t[0]= t[1] = t[2] = 26; return; }
    t[0] = t[1];
    t[1] = t[2];
    t[2] = b;
    value = FetchTrigramValue(t, trigrams);
    of (value == 255) return;
#if 0
    if (value > 64) {
      r = Random( );
      if (value > 192 && r & OxE000) return;
      else if (value > 128 && r & OxC000) return;
      else if (value > 64 && r & Ox8000) return;
    }
#endif
    StoreTrigramValue(t, trigrams, value + 1);
}
DumpTrigrams(tristorage trigrams)
{
    int i, j, k;
    int x;
    triplet t;
    x = O;
    for (i = 0; i < 26; ++i)
    for (j = 0; j < 26; ++j)
      for (k = 0; k < 26; ++k)
      {
        byte value;
        t[0] = i;
        t[1] = j;
        t[2] = k;
        value = DumpTrigram(t, trigrams);
        if (value == 0) continue;
        if (++ x ==6) {
          printf("/n"); x = 0;
        }
        else
          printf("   ");
      }
}
OSErr BuildTrigram(short refNum, tristorage trigrams)
{
    OSErr err;
    triplet t;
    t[0] = t[1] = t[2] = 26;
    while (true)
    {
      long count = 80;
      char buf[80];
      int i;
      err = FSRead(refNum, &count, buf);
      if (count == 0) return err;
      for (i = 0; i < count; ++i) {
        AddChar(trigrams, t, IdFromChar(buf[i]));
      }
      if (err) return err;
    }
    return 0;
}
Handle OpenTrigrams(void)
{
    OSErr err;
    OSType type;
    StandardFileReply reply
    short refNum;
    short id;
    Handle h;
    Str63   name;
    tristorage *trigrams;
    type = 'TEXT';
    StandardGetFile(nil, 1, &type, &reply);
    if (!reply.sfGood)   return nil;
    err = FSpOpenDF(&reply.sfFile, fsCurPerm, &refNum);
    if (err) return nil;
    memcpy(name, reply.sfFile.name, sizeof(name));
    h = NewHandle(sizeof(tristorage));
    HLock(h);
    trigrams = (tristorage *)(*h);
    memset(*trigrams, 0, sizeof(tristorage));
```

-continued

```
    BuildTrigram(refNum, *trigrams);
    FSClose(refNum);
    DumpTrigrams(*trigrams);
    HUnlock(h);
    type = 'rsrc';
    StandardGetFile(nil, 1, &type, &reply);
    if (!reply.sfGood) return;
    refNum = FSpOpenResFile(&reply.sfFile, fsCurPerm);
    if (refNum == -1) return;
    UseResFile(refNum);
    id = Unique1ID('smrt');
    //id = 128;
    AddResource(h, 'smrt', id, name);
    UpdateResFile(refNum);
    FSClose(refNum);
    return h;
}
main( )
{
    OSErr   err;
    Handle   h;
    cshow(stdout);
    TEInit( );
    InitDialogs(OL);
    InitCursor( );
    h = OpenTrigrams( );
```

I claim:

1. A method for making available a newer version of an application to a mobile device with no changes in the mobile device, the method comprising:

receiving a message over a wireless network from a first server, the message pertaining to the newer version of the application offered from a second server;

displaying, on a screen of the mobile device, a graphic user interface in response to the message;

generating a response message in response to an interaction by a user with the graphic user interface; and

providing access to the newer version of the application from the second server after the response message is successfully transported by the first server to the second server.

2. The method of claim **1**, wherein the first server is the second server.

3. The method of claim **1**, wherein the first server, remotely located with respect to the second server, coupled to the second server over a landline data network.

4. The method of claim **1**, wherein the wireless network complies with a first protocol and the landline data network complies with a second protocol.

5. The method of claim **4**, wherein the first protocol and the second protocol are compliant to each other.

6. The method and claim **4**, wherein the first protocol and the second protocol are based on the Internet Protocol (IP) but function differently.

7. The method of claim **6**, wherein the first protocol and the second protocol are respectively User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).

8. The method of claim **6**, wherein the first server functions as a protocol conversion to facilitate data communication between the mobile device and the second server.

9. The method of claim **4**, wherein the landline data network is one or more of a corporate wide area network, a corporate local area network and the Internet.

10. The method of claim **3**, wherein the first server maintains an authority to control access by the mobile device to the second server.

11. The method of claim **10**, wherein the response message is authenticated by the first server upon being received in the first server.

12. The method of claim 1, wherein the message includes information for the mobile device to generate the graphic user interface.

13. The method of claim 12, wherein the displaying of the graphic user interface in response to the message comprises:

extracting the parameters from the message received from the first server; and

interacting appropriately with hardware of the mobile device with respect to the extracted information to formulate the graphic user interface for display.

14. The method of claim 1, wherein the new application is resident on either the first server or the second server.

15. A method for making available a newer version of an application to a mobile device with no changes in the mobile device, the method comprising:

sending from a first server a message over a wireless network to the mobile device, the message pertaining to the newer version of the application being offered in a second server and including parameters that are to be extracted by the mobile device to generate a graphic user interface for display on the mobile device in response to the message;

receiving in the first server a response from the mobile device, the response including an interaction by a user of the mobile device with the graphic user interface; and

providing access to the newer version of the application by the mobile device when the response message is successfully transported by the first server to the second server.

16. The method of claim 15, wherein the first server is the second server.

17. The method of claim 15, wherein the first server, remotely located with respect to the second server, coupled to the second server over a landline data network.

18. The method of claim 17, wherein the wireless network complies with a first protocol and the landline data network complies with a second protocol.

19. The method of claim 18, wherein the landline data network is one or more of a corporate wide area network, a corporate local area network and the Internet.

20. The method of claim 17, wherein the first protocol and the second protocol are compliant to each other.

21. The method of claim 20, wherein the first protocol and the second protocol are based on the Internet Protocol (IP) but function differently.

22. The method of claim 21, wherein the first protocol and the second protocol are respectively User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).

23. The method of claim 21, wherein the first server functions as a protocol conversion to facilitate data communication between the mobile device and the second server.

24. The method of claim 17, wherein the first server maintains an authority to control access by the mobile device to the second server.

25. The method of claim 24, wherein the response message is authenticated by the first server upon being received in the first server.

26. The method of claim 15, wherein the message includes information for the mobile device to generate the graphic user interface.

27. A computer product for making available a newer version of an application to a mobile device with no changes in the mobile device, the computer product to be executed in the mobile device, the computer product comprising:

program code for receiving a message over a wireless network from a first server, the message pertaining to the newer version of the application offered from a second server;

program code for displaying, on a screen of the mobile device, a graphic user interface in response to the message;

program code for generating a response message in response to an interaction by a user with the graphic user interface; and

program code for providing access to the newer version of the application from the second server after the response message is successfully transported by the first server to the second server.

28. A computer product for making available a newer version of an application to a mobile device with no changes in the mobile device, the computer product to be executed in a first server, the computer product comprising:

program code for sending from the first server a message over a wireless network to the mobile device, the message pertaining to the newer version of the application being offered in a second server and including parameters that are to be extracted by the mobile device to generate a graphic user interface for display on the mobile device in response to the message;

program code for receiving in the first server a response message from the mobile device, the response including an interaction by a user of the mobile device with the graphic user interface; and

program code for providing access to the newer version of the application by the mobile device when the response message is successfully transported by the first server to the second server.

* * * * *