# EXHIBIT A

US006035332A

# United States Patent [19]

Ingrassia, Jr. et al.

[11] Patent Number: 6,035,332

[45] Date of Patent: Mar. 7, 2000

[54] **METHOD FOR MONITORING USER INTERACTIONS WITH WEB PAGES FROM WEB SERVER USING DATA AND COMMAND LISTS FOR MAINTAINING INFORMATION VISITED AND ISSUED BY PARTICIPANTS**

[75] Inventors: **Michael I. Ingrassia, Jr.**, Edison; **James A. Shelton**, Holmdel; **Thomas M. Rowland**, Fair Haven, all of N.J.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **08/944,759**

[22] Filed: **Oct. 6, 1997**

[51] **Int. Cl.**[7] ..... **G06F 13/00**

[52] **U.S. Cl.** ..... **709/224**; 709/218; 709/223

[58] **Field of Search** ..... 395/761; 705/10; 709/224, 227, 218, 223

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,260,878 | 11/1993 | Luppy . | |
| 5,535,256 | 7/1996 | Maloney et al. . | |
| 5,544,649 | 8/1996 | David et al. . | |
| 5,715,453 | 2/1998 | Stewart . | |
| 5,737,619 | 4/1998 | Judson | 395/761 |
| 5,742,768 | 4/1998 | Gennaro et al. . | |
| 5,757,129 | 5/1998 | Schafnitzel et al. . | |
| 5,774,660 | 6/1998 | Brendel et al. . | |
| 5,774,670 | 6/1998 | Montulli | 395/200.57 |
| 5,784,058 | 7/1998 | LaStrange et al. . | |
| 5,793,972 | 8/1998 | Shane . | |
| 5,796,952 | 8/1998 | Davis et al. | 395/200.54 |
| 5,802,299 | 9/1998 | Logan et al. . | |
| 5,809,250 | 9/1998 | Kisor . | |
| 5,848,396 | 12/1998 | Gerace | 705/10 |

*Primary Examiner*—Le Hien Luu

*Attorney, Agent, or Firm*—Ying Tuo; Kenneth M. Berner

[57] **ABSTRACT**

Described is a mechanism for dependably tracking web page activities among a group of browsers. The web browsers retrieve web pages from an HTTP server, with each of the web pages embedding an applet. In response to web page activities (such as loading or unloading of a web page) performed at a browser, the respective applet reports the activities (together with the URL of the web page) to a synchronization server, which in turn stores them in a database.

**37 Claims, 22 Drawing Sheets**

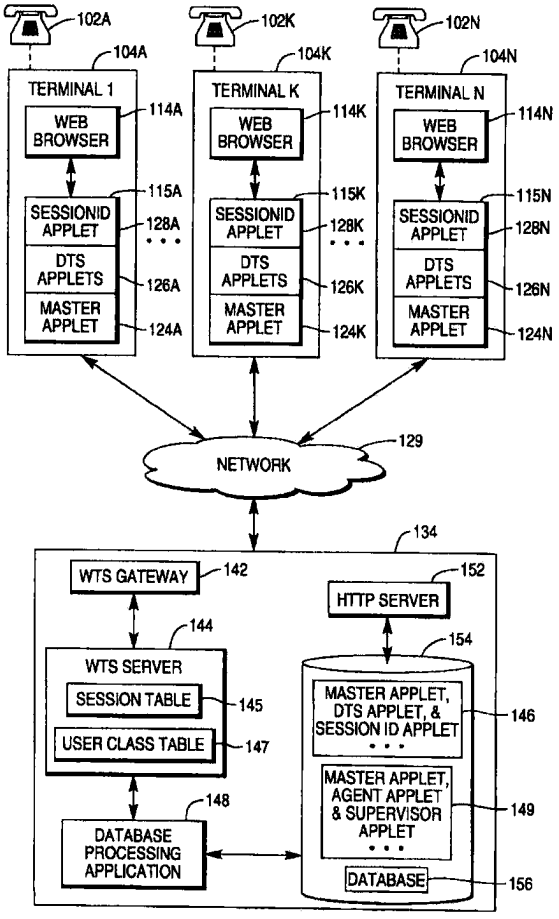
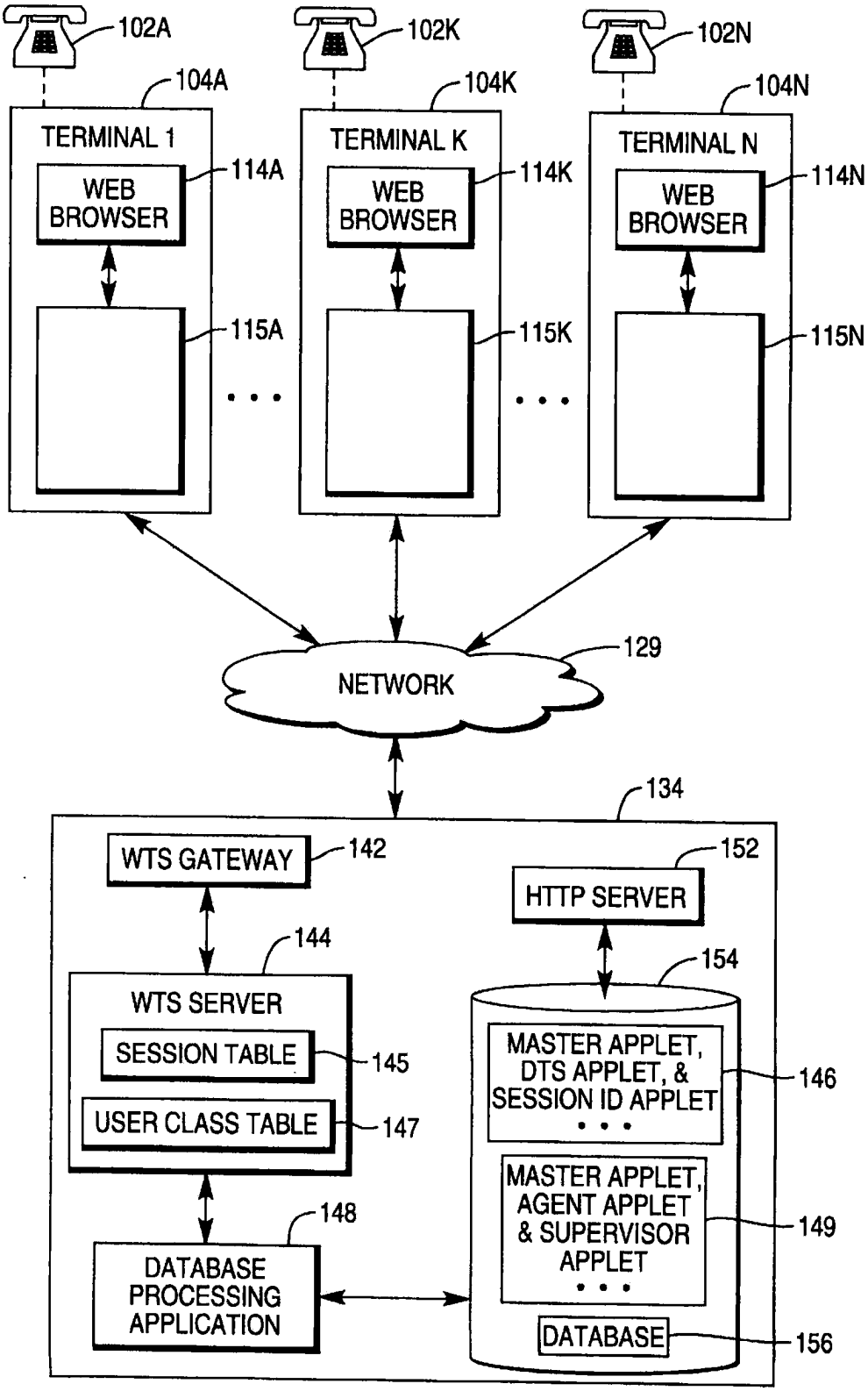


FIG. 1


102A
102K
102N
104A
104K
104N
TERMINAL 1
TERMINAL K
TERMINAL N
WEB BROWSER
WEB BROWSER
WEB BROWSER
114A
114K
114N
115A
115K
115N
NETWORK
129
134
WTS GATEWAY
142
HTTP SERVER
152
144
154
WTS SERVER
SESSION TABLE
145
USER CLASS TABLE
147
MASTER APPLET, DTS APPLET, & SESSION ID APPLET
146
MASTER APPLET, AGENT APPLET & SUPERVISOR APPLET
149
148
DATABASE PROCESSING APPLICATION
DATABASE
156

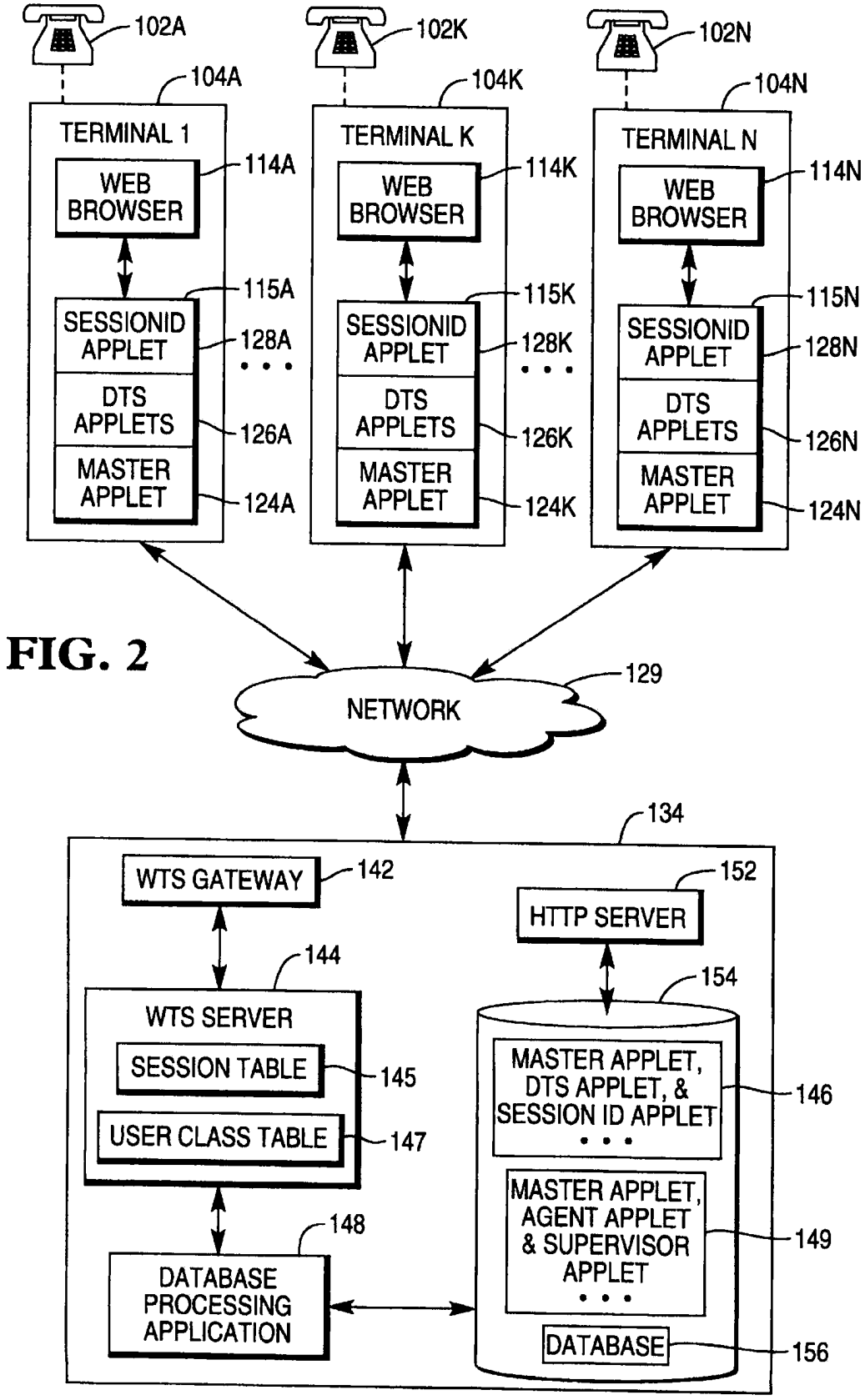

102A
102K
102N
104A
104K
104N
TERMINAL 1
TERMINAL K
TERMINAL N
WEB BROWSER
114A
114K
114N
SESSIONID APPLET
115A
115K
115N
128A
128K
128N
DTS APPLETS
126A
126K
126N
MASTER APPLET
124A
124K
124N
NETWORK
129
134
WTS GATEWAY
142
HTTP SERVER
152
144
154
WTS SERVER
SESSION TABLE
145
USER CLASS TABLE
147
MASTER APPLET, DTS APPLET, & SESSION ID APPLET
146
MASTER APPLET, AGENT APPLET & SUPERVISOR APPLET
149
148
DATABASE PROCESSING APPLICATION
DATABASE
156

FIG. 2

FIG. 3

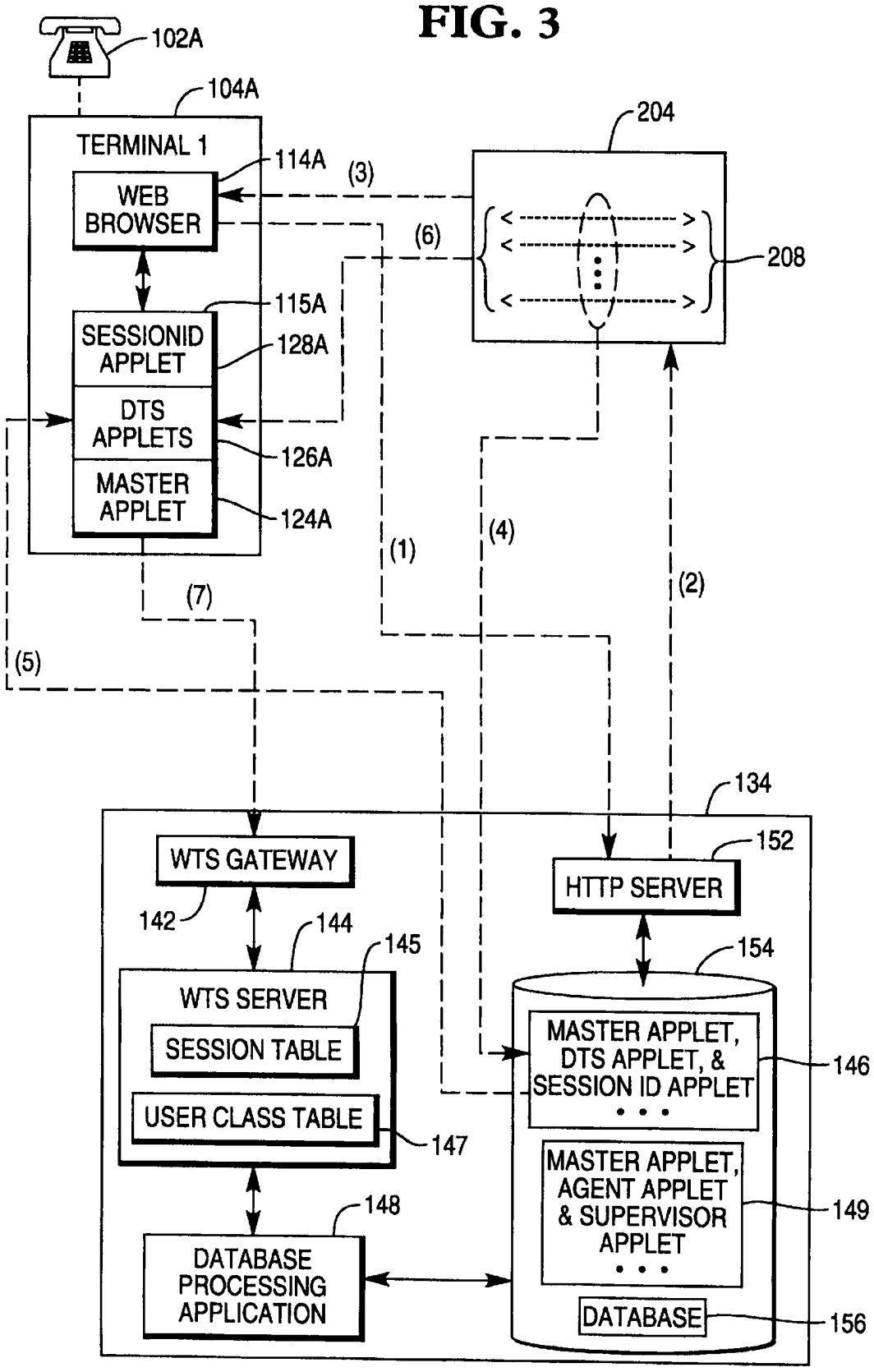

102A
104A
TERMINAL 1
114A
WEB BROWSER
115A
SESSIONID APPLET
128A
DTS APPLETS
126A
MASTER APPLET
124A
204
208
(1)
(2)
(3)
(4)
(5)
(6)
(7)
134
WTS GATEWAY
142
144
145
WTS SERVER
SESSION TABLE
USER CLASS TABLE
147
148
DATABASE PROCESSING APPLICATION
152
HTTP SERVER
154
MASTER APPLET, DTS APPLET, & SESSION ID APPLET
146
MASTER APPLET, AGENT APPLET & SUPERVISOR APPLET
149
DATABASE
156

FIG. 4

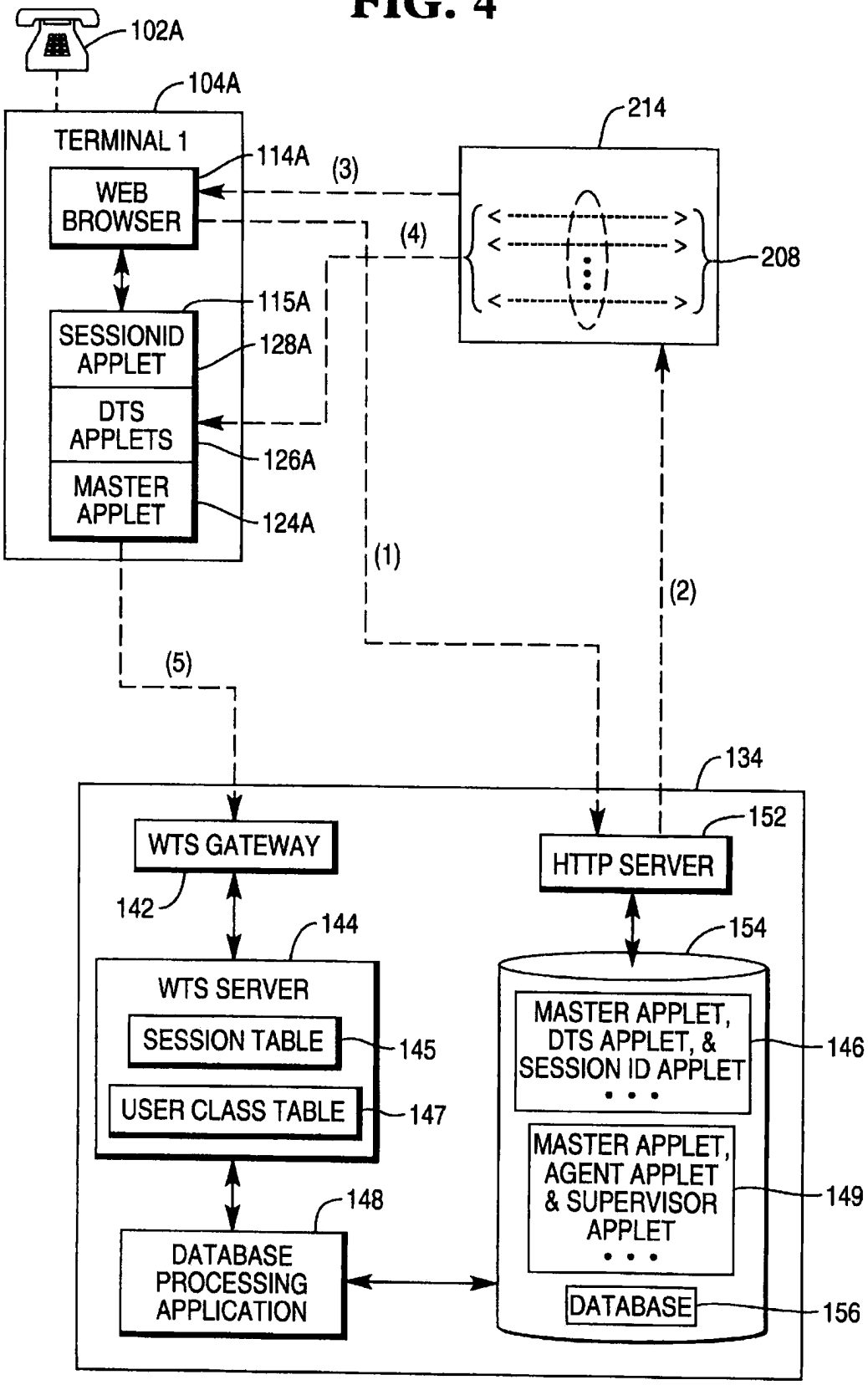

102A
104A
TERMINAL 1
114A
WEB BROWSER
(3)
214
(4)
208
115A
SESSIONID APPLET
128A
DTS APPLETS
126A
MASTER APPLET
124A
(1)
(2)
(5)
134
WTS GATEWAY
142
152
HTTP SERVER
144
154
WTS SERVER
SESSION TABLE
145
MASTER APPLET, DTS APPLET, & SESSION ID APPLET
146
USER CLASS TABLE
147
MASTER APPLET, AGENT APPLET & SUPERVISOR APPLET
149
148
DATABASE PROCESSING APPLICATION
DATABASE
156

FIG. 5

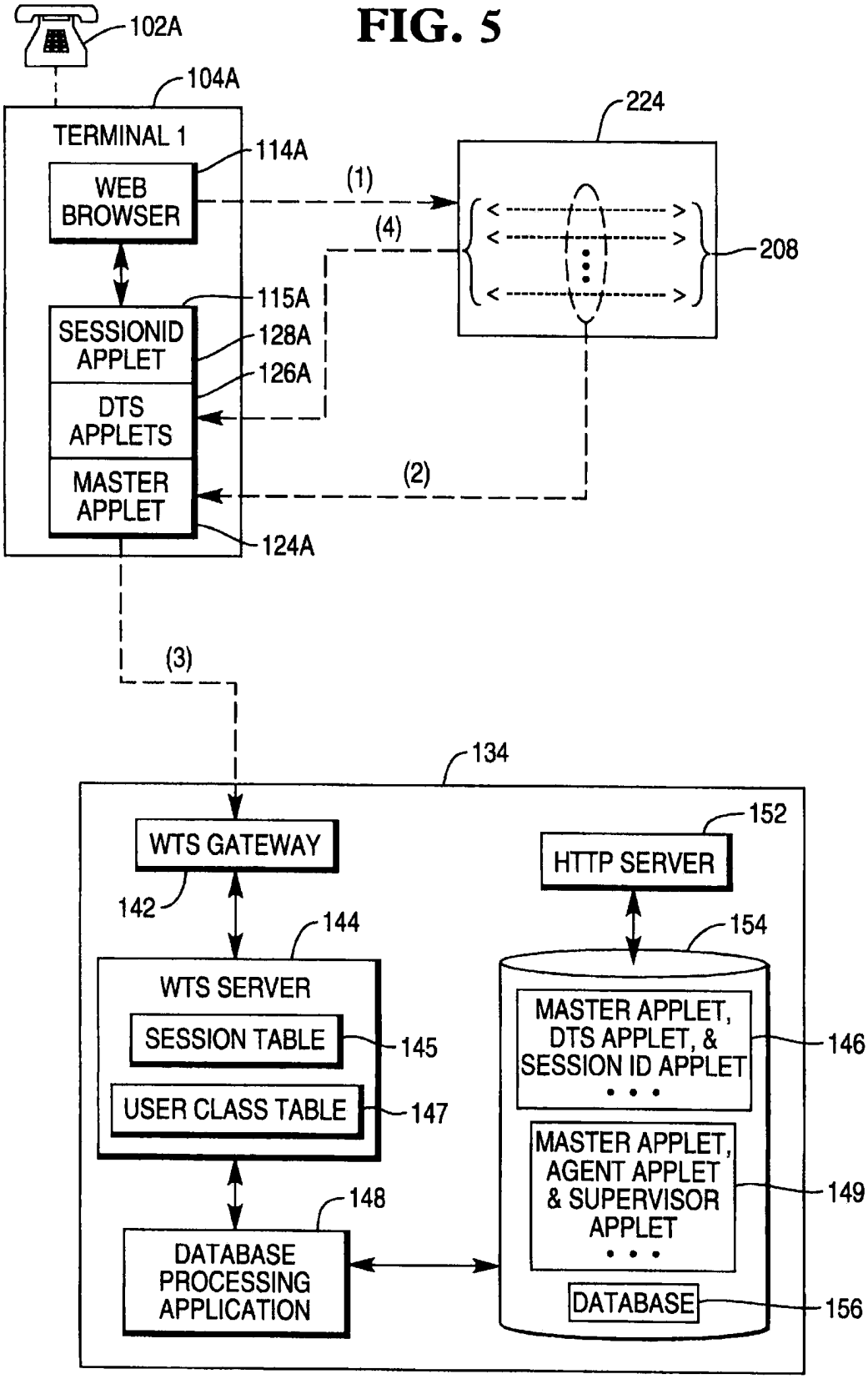

102A
104A
TERMINAL 1
114A
WEB BROWSER
(1)
224
(4)
208
115A
SESSIONID APPLET
128A
126A
DTS APPLETS
MASTER APPLET
(2)
124A
(3)
134
WTS GATEWAY
142
144
152
HTTP SERVER
154
WTS SERVER
SESSION TABLE
145
USER CLASS TABLE
147
MASTER APPLET, DTS APPLET, & SESSION ID APPLET
146
MASTER APPLET, AGENT APPLET & SUPERVISOR APPLET
149
148
DATABASE PROCESSING APPLICATION
DATABASE
156

FIG. 6

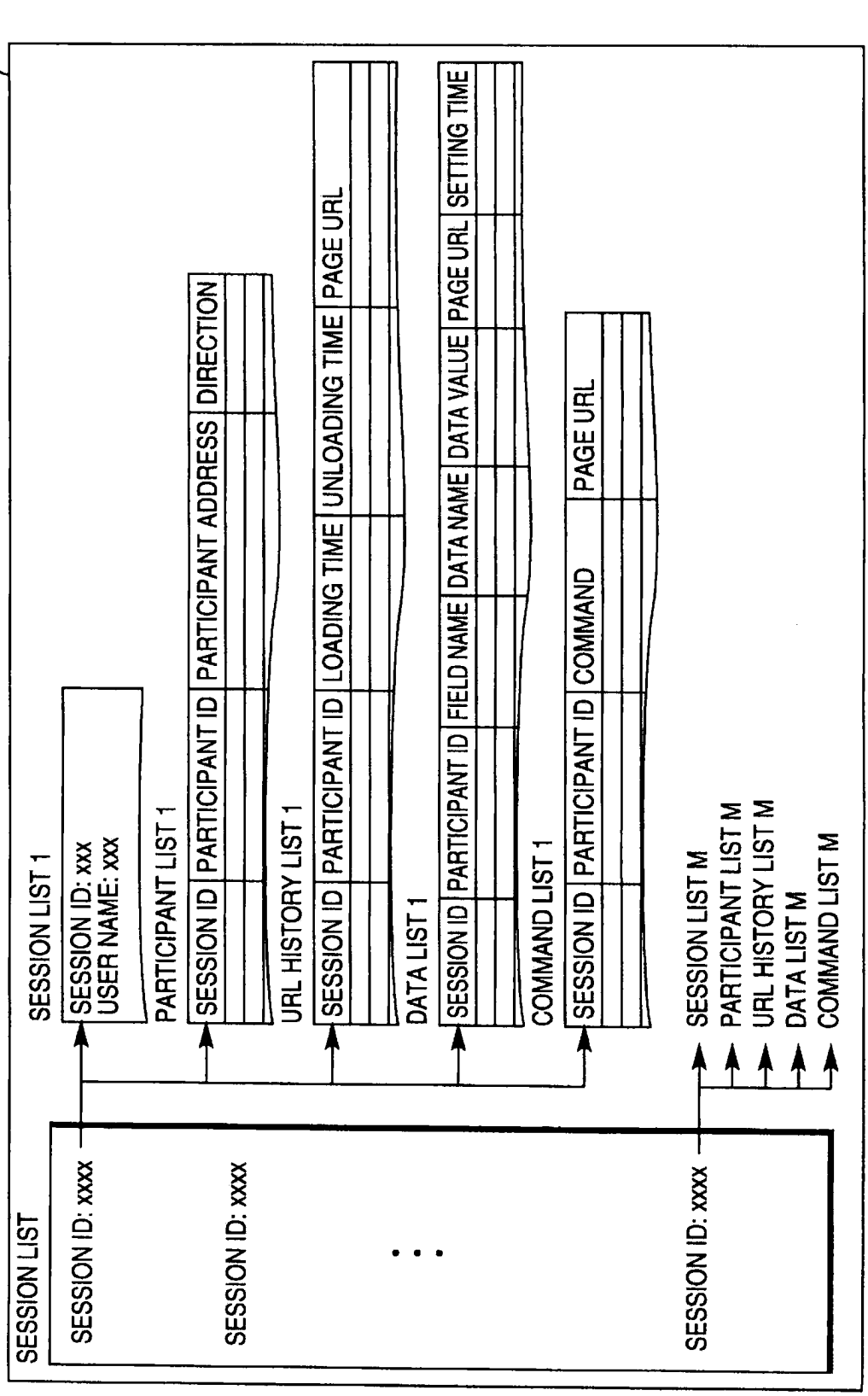

145
SESSION LIST
SESSION ID: xxxx
SESSION ID: xxxx
SESSION ID: xxxx
SESSION LIST 1
SESSION ID: xxx
USER NAME: xxx
PARTICIPANT LIST 1
SESSION ID
PARTICIPANT ID
PARTICIPANT ADDRESS
DIRECTION
URL HISTORY LIST 1
SESSION ID
PARTICIPANT ID
LOADING TIME
UNLOADING TIME
PAGE URL
DATA LIST 1
SESSION ID
PARTICIPANT ID
FIELD NAME
DATA NAME
DATA VALUE
PAGE URL
SETTING TIME
COMMAND LIST 1
SESSION ID
PARTICIPANT ID
COMMAND
PAGE URL
SESSION LIST M
PARTICIPANT LIST M
URL HISTORY LIST M
DATA LIST M
COMMAND LIST M

FIG. 7

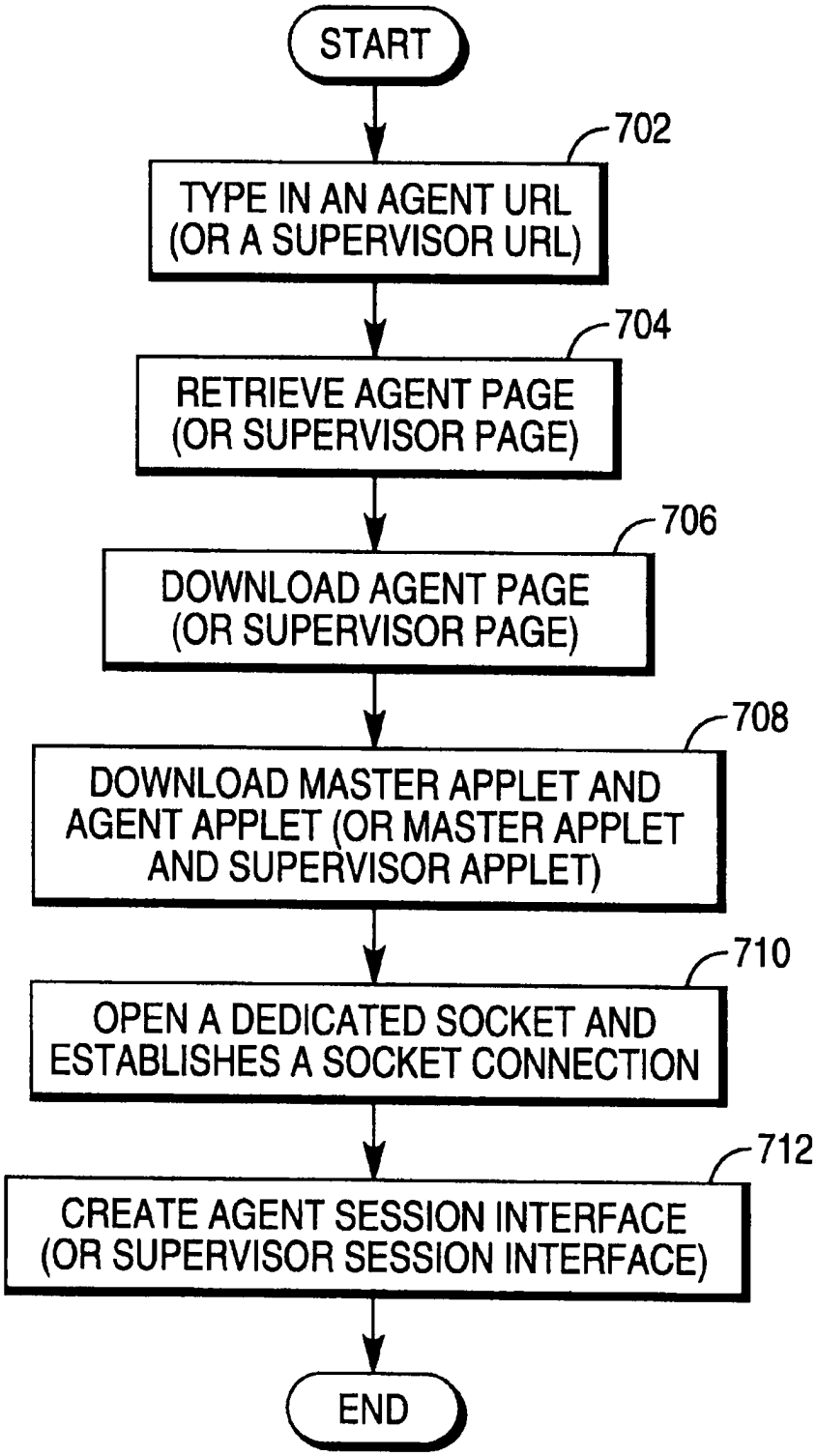

START
702
TYPE IN AN AGENT URL (OR A SUPERVISOR URL)
704
RETRIEVE AGENT PAGE (OR SUPERVISOR PAGE)
706
DOWNLOAD AGENT PAGE (OR SUPERVISOR PAGE)
708
DOWNLOAD MASTER APPLET AND AGENT APPLET (OR MASTER APPLET AND SUPERVISOR APPLET)
710
OPEN A DEDICATED SOCKET AND ESTABLISHES A SOCKET CONNECTION
712
CREATE AGENT SESSION INTERFACE (OR SUPERVISOR SESSION INTERFACE)
END

FIG. 8A

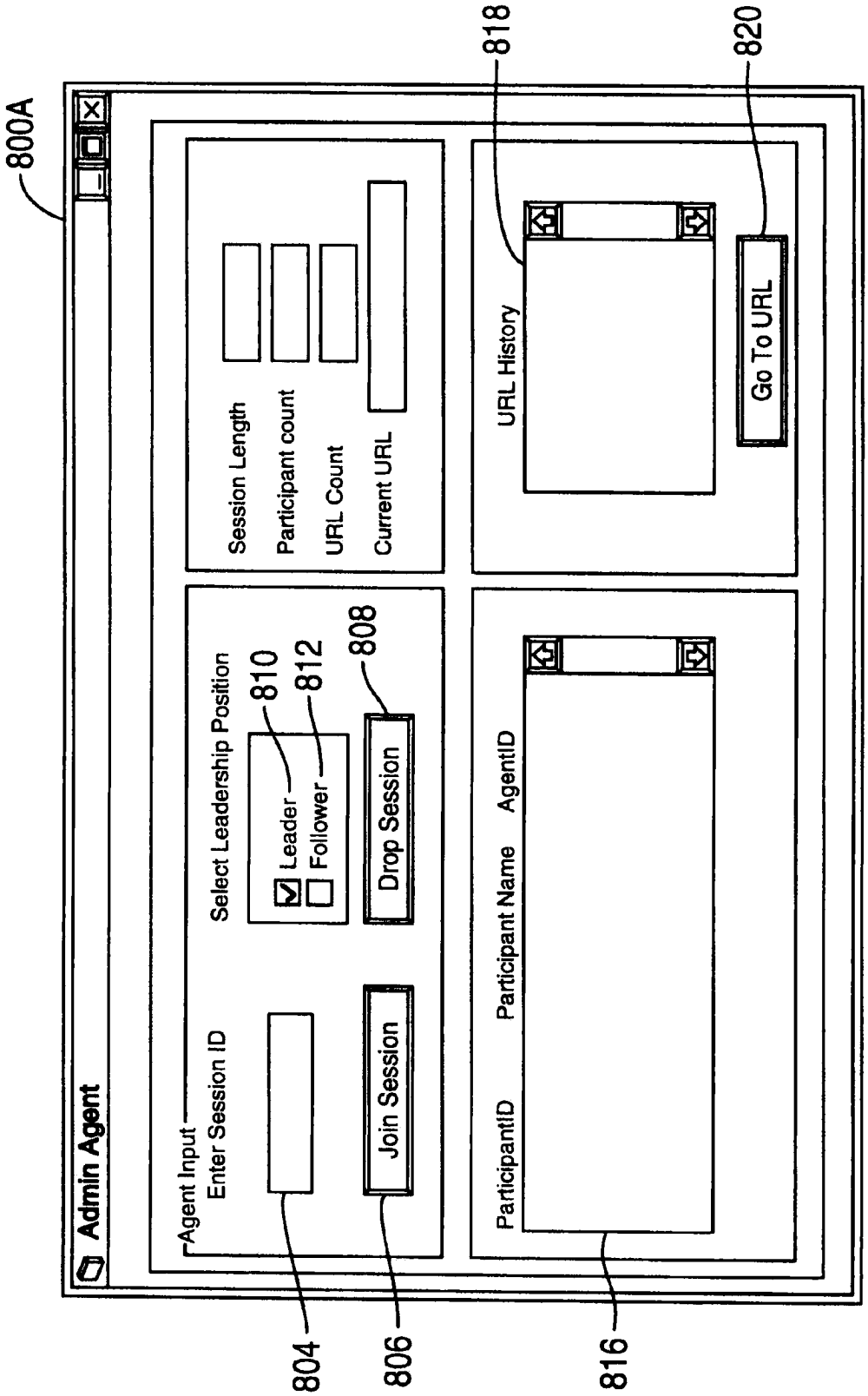
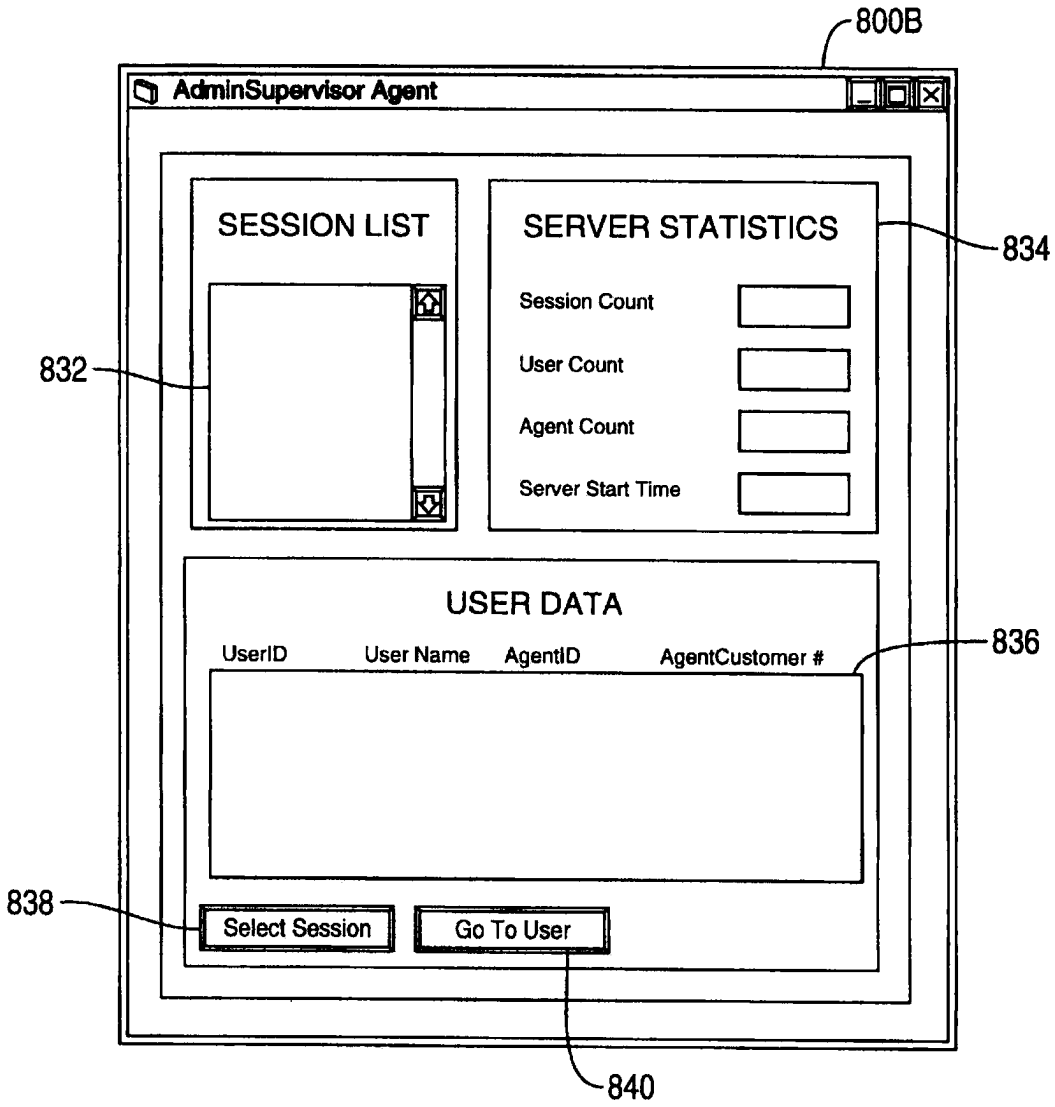

800A
818
820
Admin Agent
Session Length
Participant count
URL Count
Current URL
URL History
Go To URL
Select Leadership Position
Leader
Follower
810
812
808
Drop Session
Agent Input
Enter Session ID
Join Session
ParticipantID
Participant Name
AgentID
804
806
816

FIG. 8B


800B
AdminSupervisor Agent
SESSION LIST
SERVER STATISTICS
Session Count
User Count
Agent Count
Server Start Time
834
832
USER DATA
UserID
User Name
AgentID
AgentCustomer #
836
838
Select Session
Go To User
840

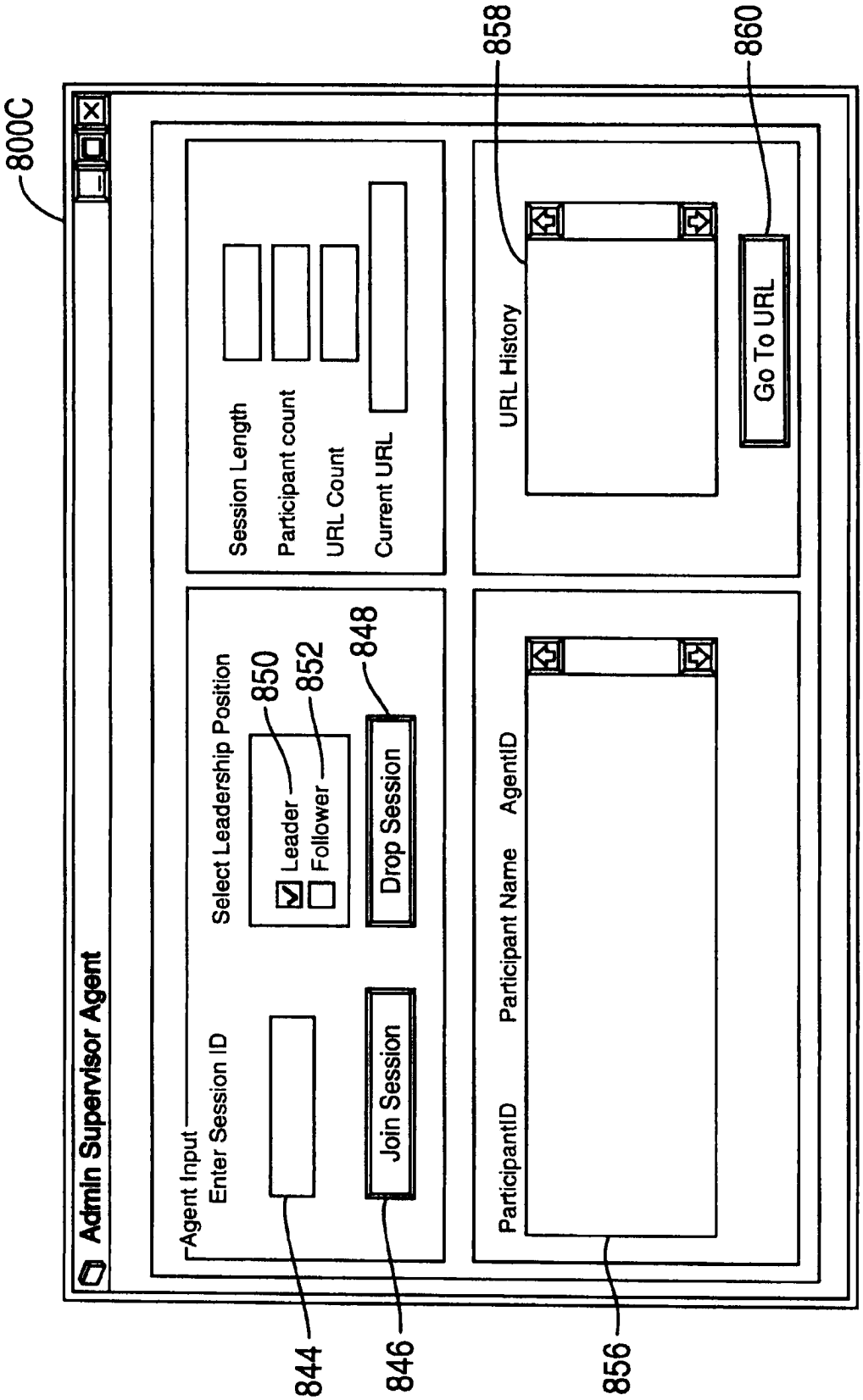
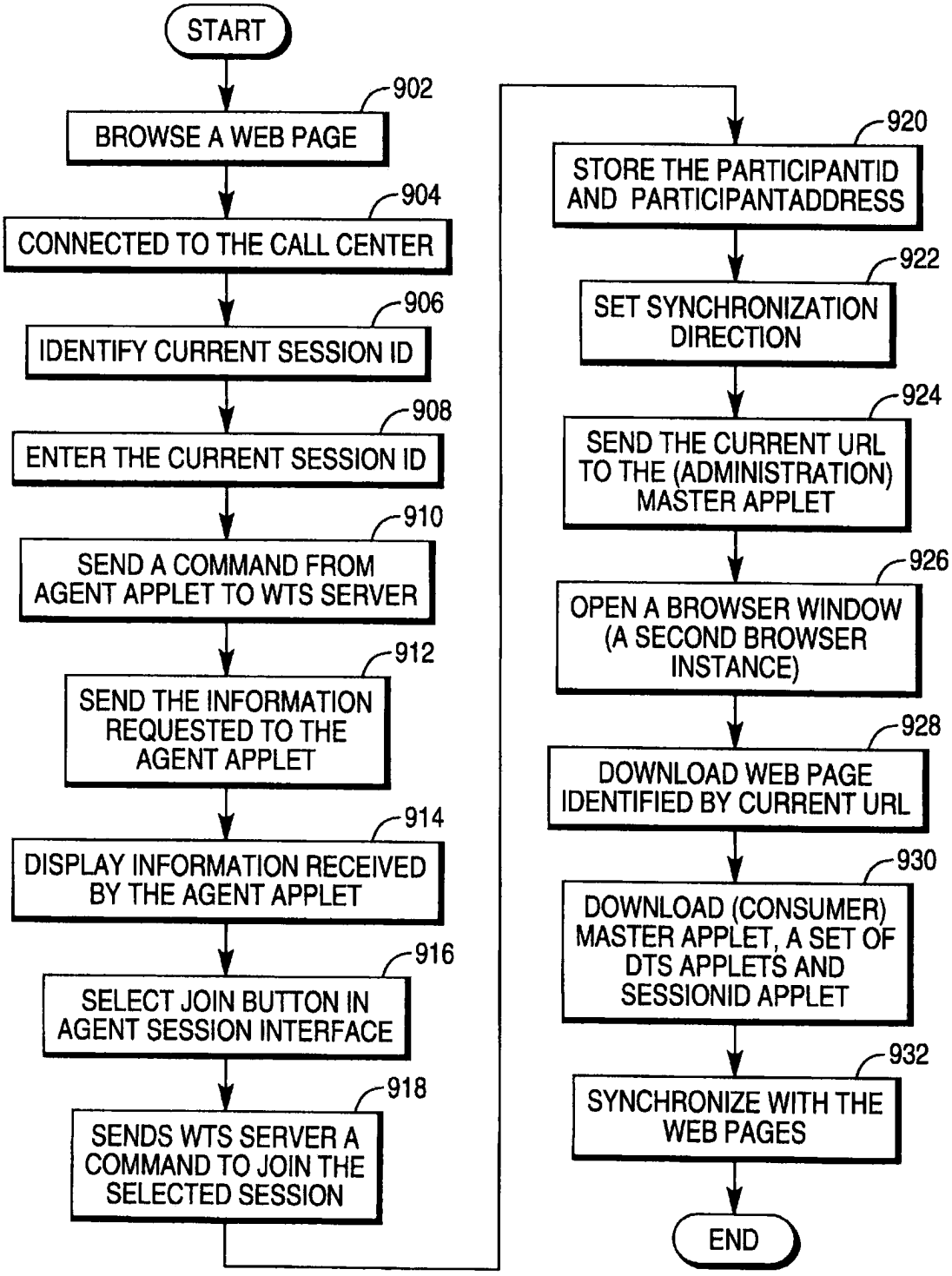

800C
858
860
Admin Supervisor Agent
Session Length
Participant count
URL Count
Current URL
URL History
Go To URL
Select Leadership Position
850
852
848
Leader
Follower
Drop Session
Agent Input
Enter Session ID
Join Session
ParticipantID
Participant Name
AgentID
844
846
856

FIG. 8C

# FIG. 9


START
902
BROWSE A WEB PAGE
904
CONNECTED TO THE CALL CENTER
906
IDENTIFY CURRENT SESSION ID
908
ENTER THE CURRENT SESSION ID
910
SEND A COMMAND FROM AGENT APPLET TO WTS SERVER
912
SEND THE INFORMATION REQUESTED TO THE AGENT APPLET
914
DISPLAY INFORMATION RECEIVED BY THE AGENT APPLET
916
SELECT JOIN BUTTON IN AGENT SESSION INTERFACE
918
SENDS WTS SERVER A COMMAND TO JOIN THE SELECTED SESSION
920
STORE THE PARTICIPANTID AND PARTICIPANTADDRESS
922
SET SYNCHRONIZATION DIRECTION
924
SEND THE CURRENT URL TO THE (ADMINISTRATION) MASTER APPLET
926
OPEN A BROWSER WINDOW (A SECOND BROWSER INSTANCE)
928
DOWNLOAD WEB PAGE IDENTIFIED BY CURRENT URL
930
DOWNLOAD (CONSUMER) MASTER APPLET, A SET OF DTS APPLETS AND SESSIONID APPLET
932
SYNCHRONIZE WITH THE WEB PAGES
END

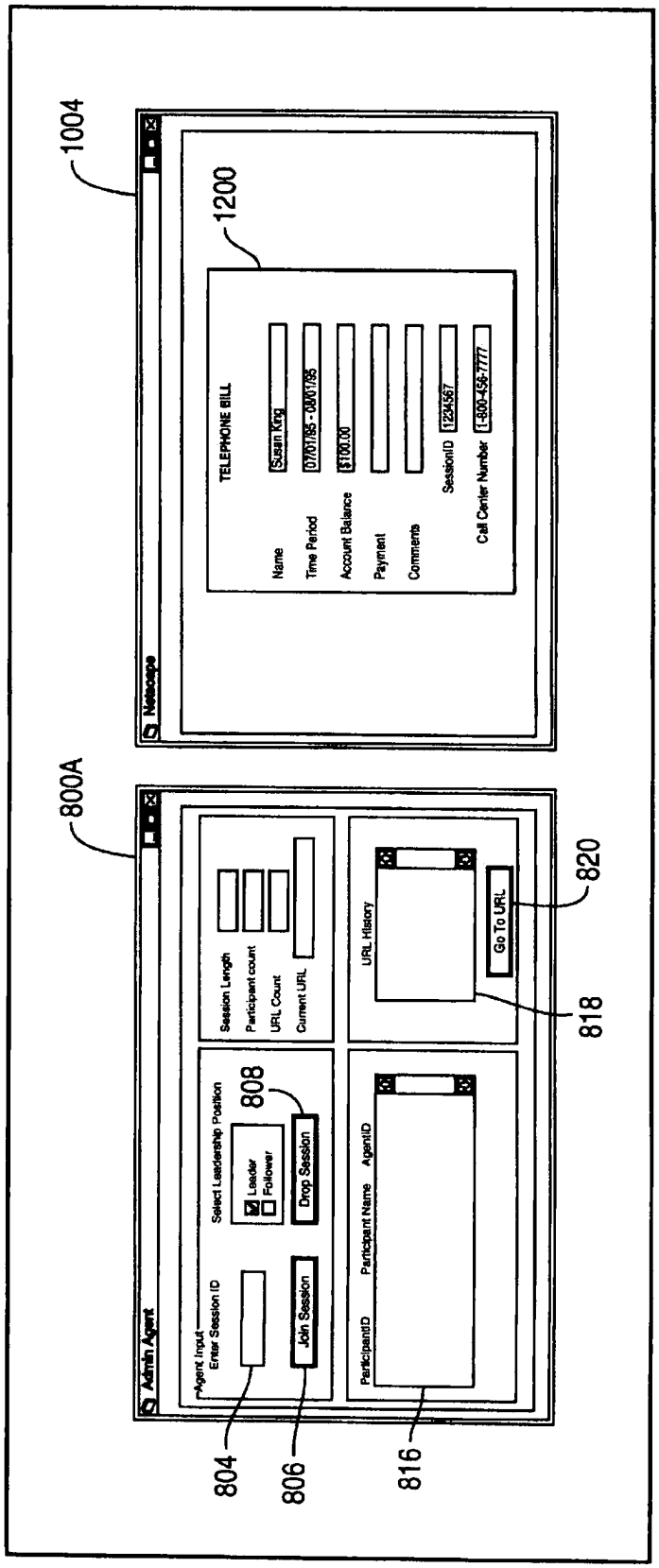
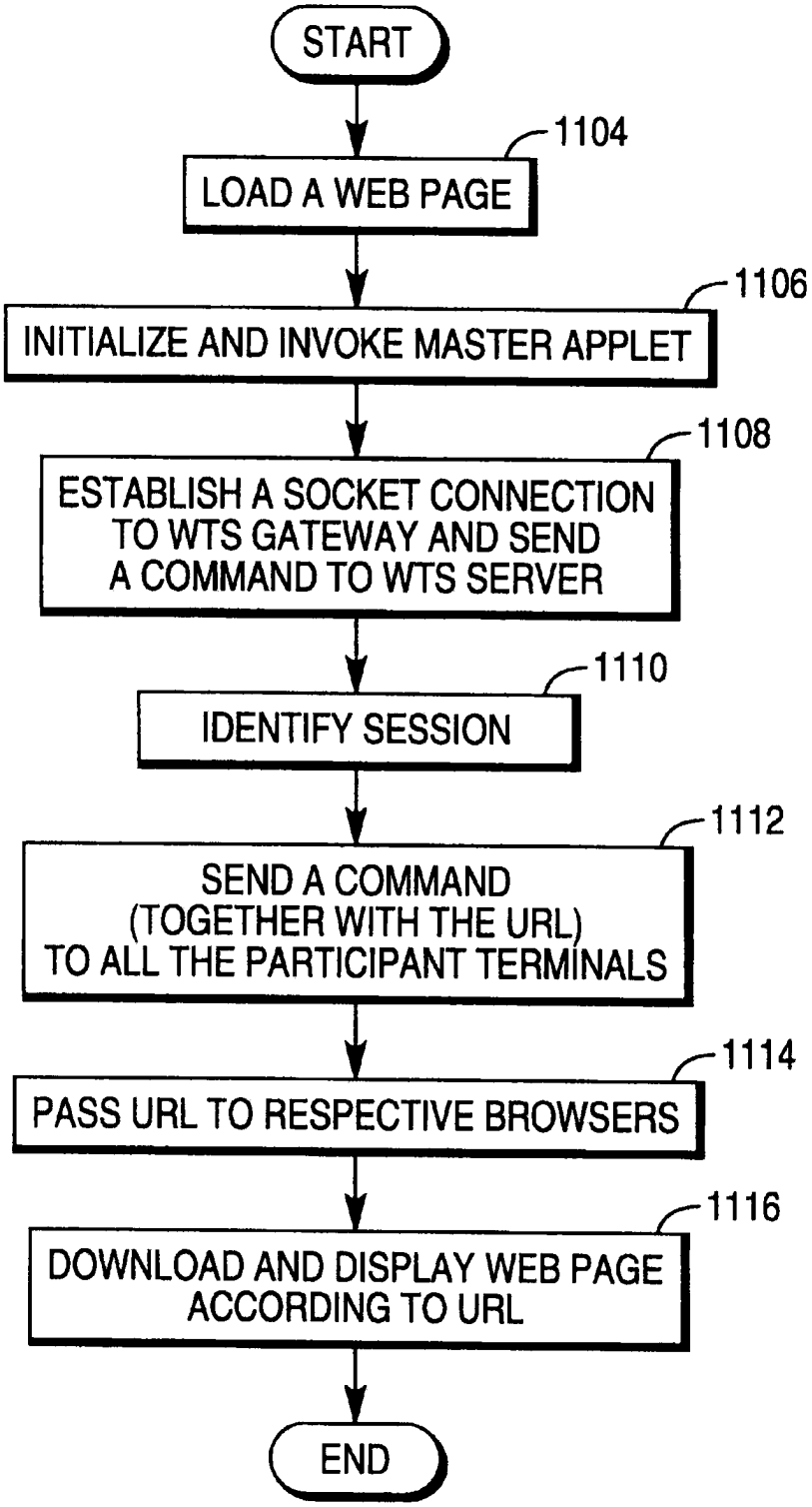

1004
1200
Netscape
TELEPHONE BILL
Name
Time Period
Account Balance
Payment
Comments
SessionID
Call Center Number
Susan King
07/01/85 - 08/01/85
$100.00
1234567
1-800-456-7777
800A
Admin Agent
Agent Input
Enter Session ID
Select Leadership Position
Leader
Follower
Drop Session
Join Session
Session Length
Participant count
URL Count
Current URL
URL History
Go To URL
ParticipantID
Participant Name
AgentID
808
820
818
804
806
816

FIG. 10

# FIG. 11


START
1104
LOAD A WEB PAGE
1106
INITIALIZE AND INVOKE MASTER APPLET
1108
ESTABLISH A SOCKET CONNECTION TO WTS GATEWAY AND SEND A COMMAND TO WTS SERVER
1110
IDENTIFY SESSION
1112
SEND A COMMAND (TOGETHER WITH THE URL) TO ALL THE PARTICIPANT TERMINALS
1114
PASS URL TO RESPECTIVE BROWSERS
1116
DOWNLOAD AND DISPLAY WEB PAGE ACCORDING TO URL
END

# FIG. 12A

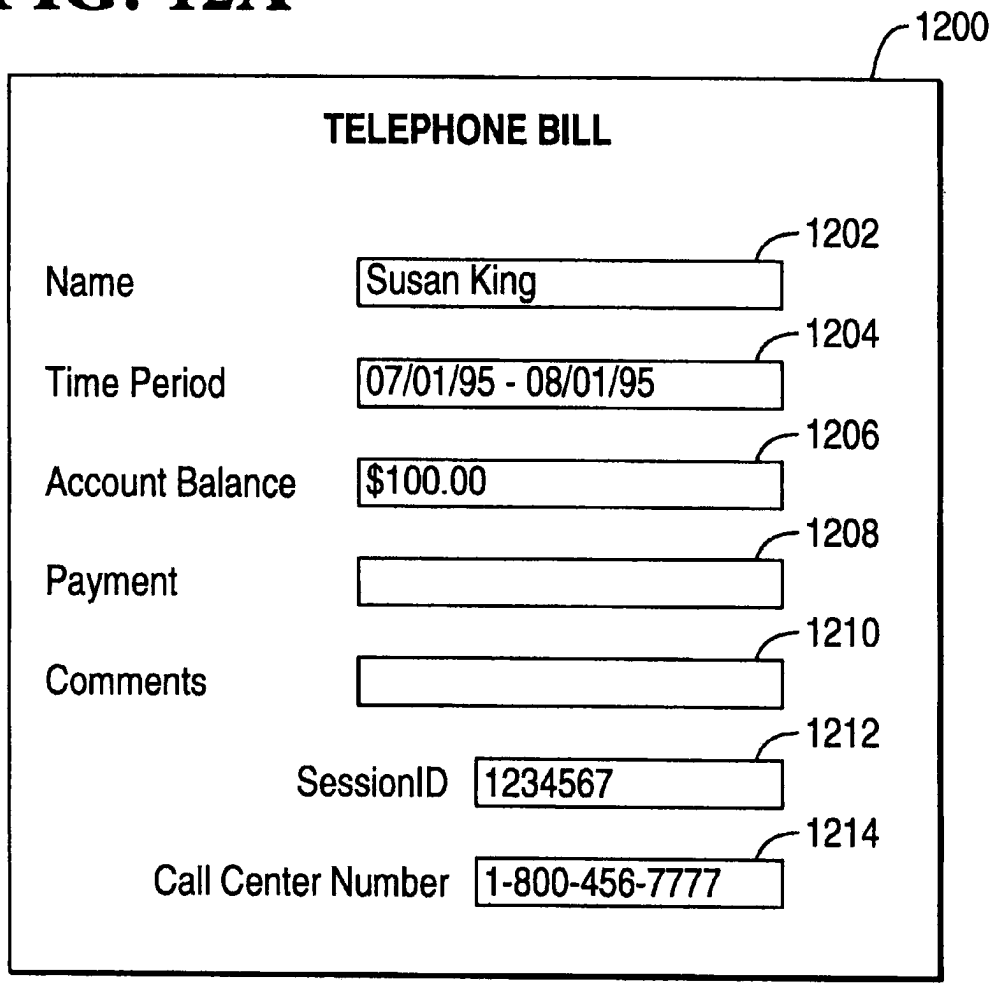

1200
TELEPHONE BILL
1202
Name
Susan King
1204
Time Period
07/01/95 - 08/01/95
1206
Account Balance
$100.00
1208
Payment
1210
Comments
1212
SessionID
1234567
1214
Call Center Number
1-800-456-7777

FIG. 12B

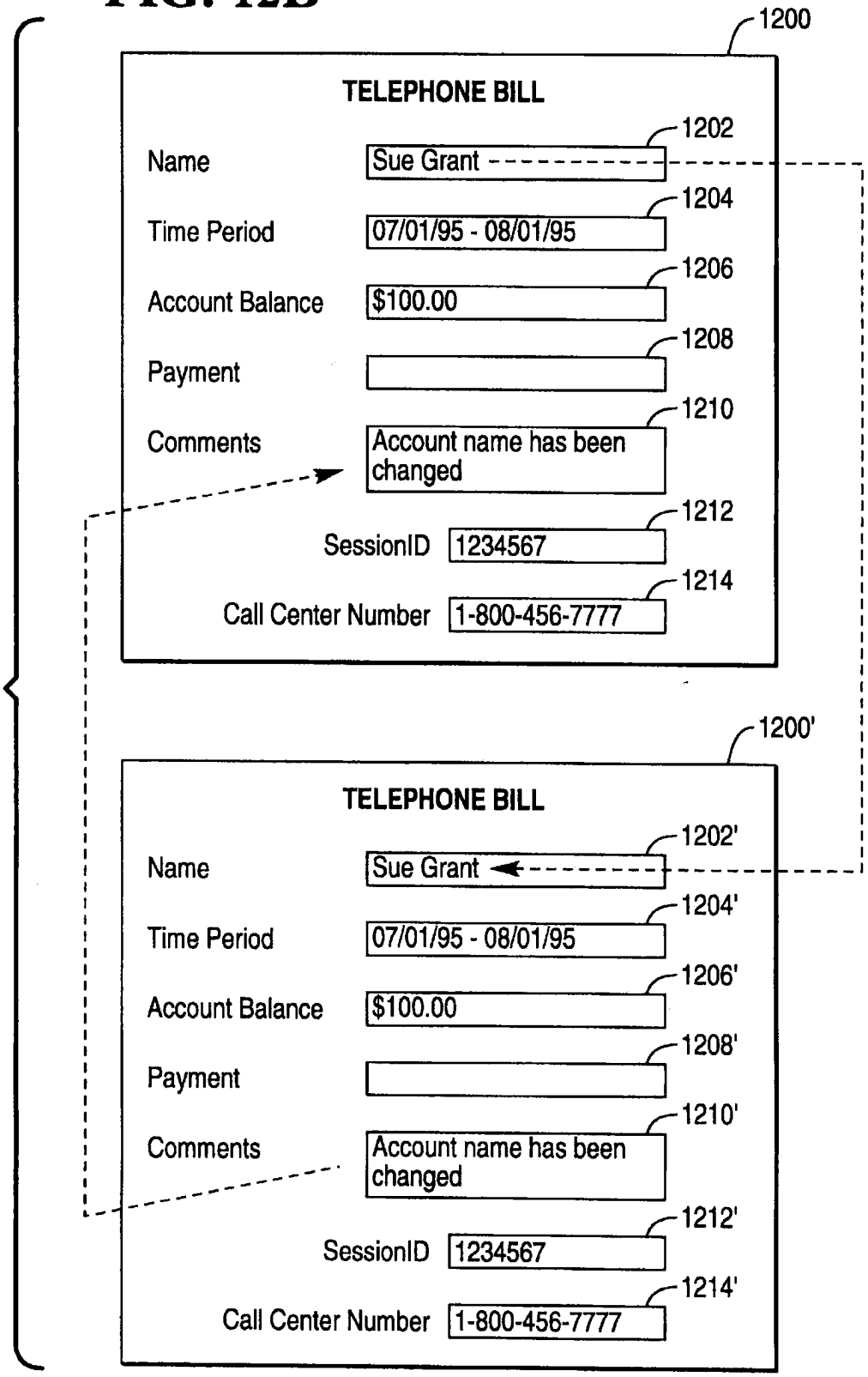
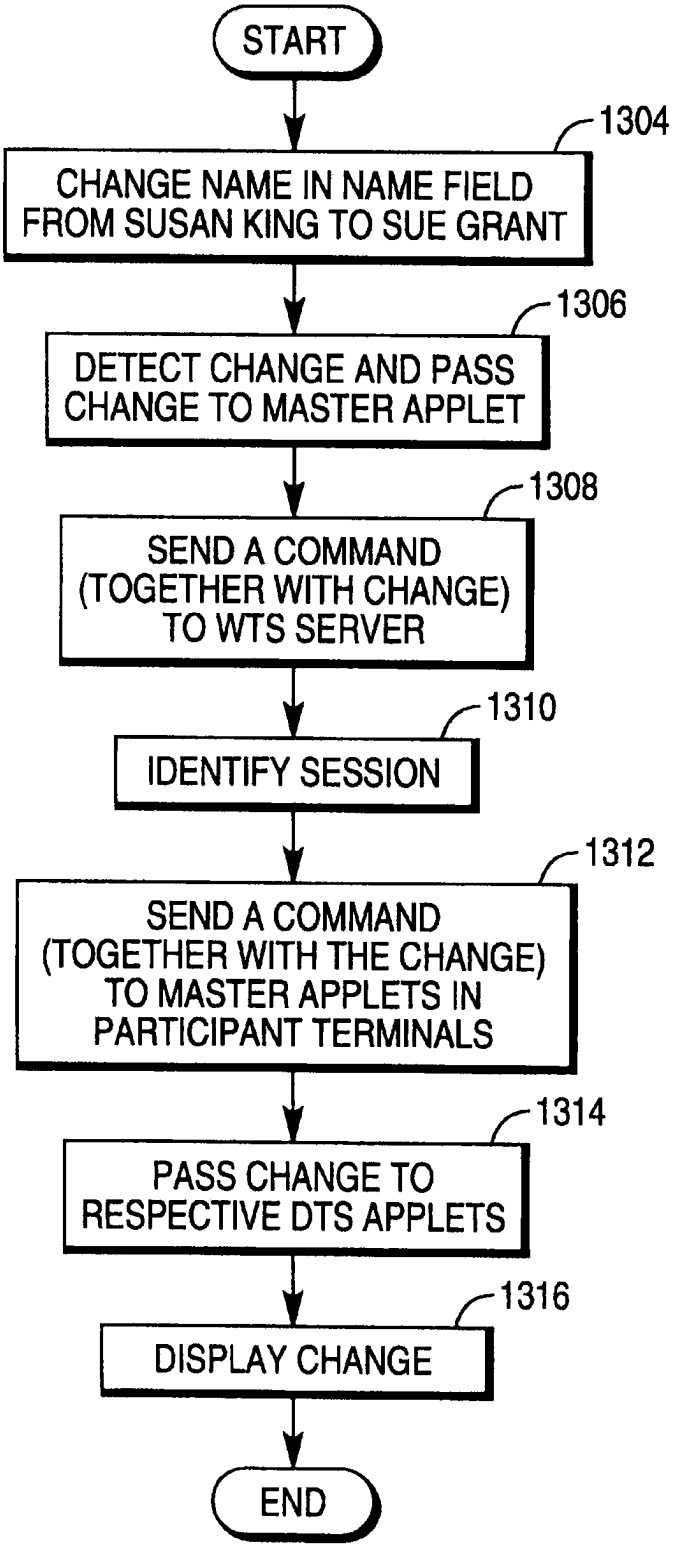
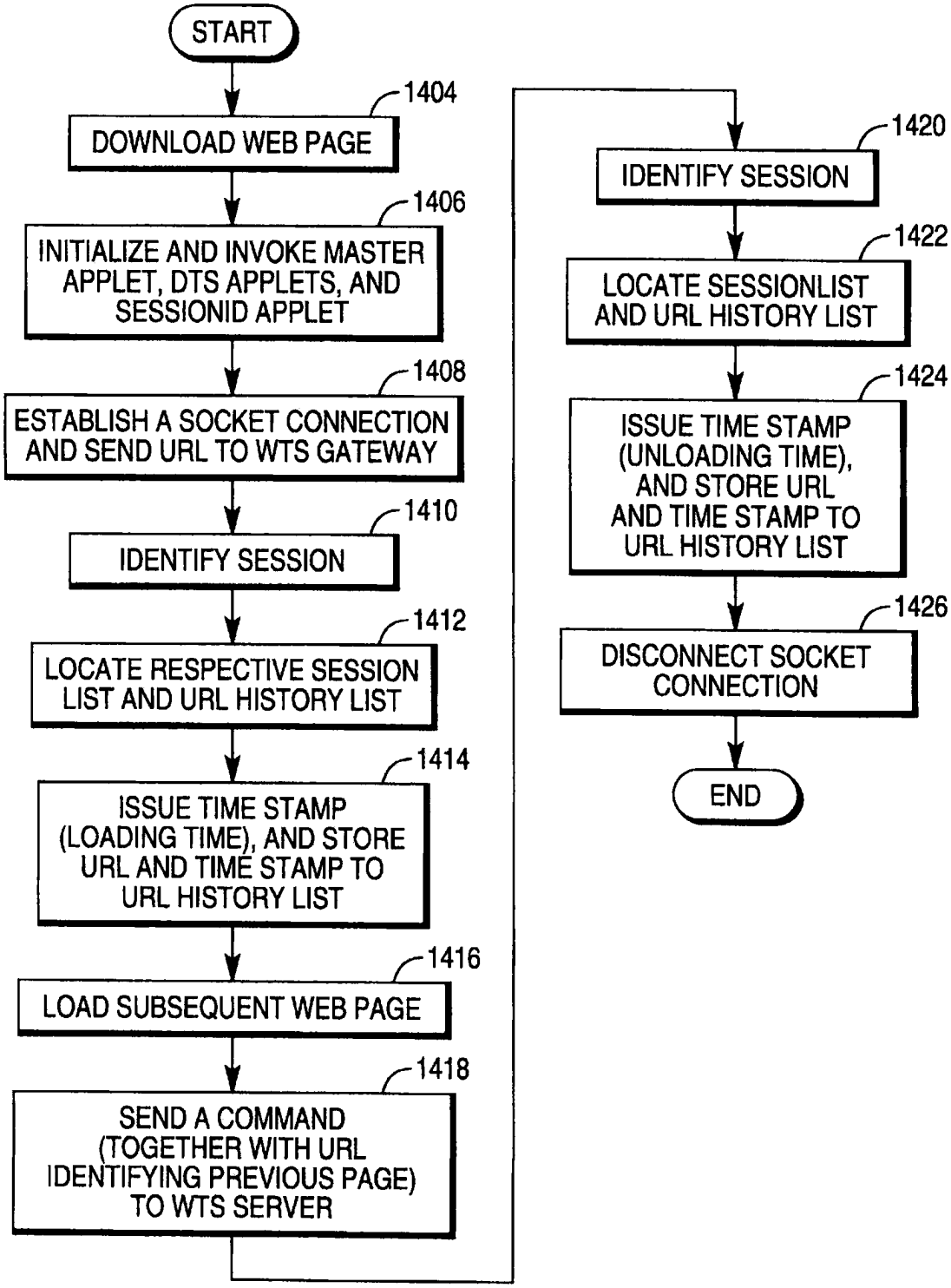

1200
TELEPHONE BILL
1202
Name
Sue Grant
1204
Time Period
07/01/95 - 08/01/95
1206
Account Balance
$100.00
1208
Payment
1210
Comments
Account name has been changed
1212
SessionID
1234567
1214
Call Center Number
1-800-456-7777
1200'
TELEPHONE BILL
1202'
Name
Sue Grant
1204'
Time Period
07/01/95 - 08/01/95
1206'
Account Balance
$100.00
1208'
Payment
1210'
Comments
Account name has been changed
1212'
SessionID
1234567
1214'
Call Center Number
1-800-456-7777

FIG. 13


START
1304
CHANGE NAME IN NAME FIELD FROM SUSAN KING TO SUE GRANT
1306
DETECT CHANGE AND PASS CHANGE TO MASTER APPLET
1308
SEND A COMMAND (TOGETHER WITH CHANGE) TO WTS SERVER
1310
IDENTIFY SESSION
1312
SEND A COMMAND (TOGETHER WITH THE CHANGE) TO MASTER APPLETS IN PARTICIPANT TERMINALS
1314
PASS CHANGE TO RESPECTIVE DTS APPLETS
1316
DISPLAY CHANGE
END

# FIG. 14


START
1404
DOWNLOAD WEB PAGE
1406
INITIALIZE AND INVOKE MASTER APPLET, DTS APPLETS, AND SESSIONID APPLET
1408
ESTABLISH A SOCKET CONNECTION AND SEND URL TO WTS GATEWAY
1410
IDENTIFY SESSION
1412
LOCATE RESPECTIVE SESSION LIST AND URL HISTORY LIST
1414
ISSUE TIME STAMP (LOADING TIME), AND STORE URL AND TIME STAMP TO URL HISTORY LIST
1416
LOAD SUBSEQUENT WEB PAGE
1418
SEND A COMMAND (TOGETHER WITH URL IDENTIFYING PREVIOUS PAGE) TO WTS SERVER
1420
IDENTIFY SESSION
1422
LOCATE SESSIONLIST AND URL HISTORY LIST
1424
ISSUE TIME STAMP (UNLOADING TIME), AND STORE URL AND TIME STAMP TO URL HISTORY LIST
1426
DISCONNECT SOCKET CONNECTION
END

# FIG. 15

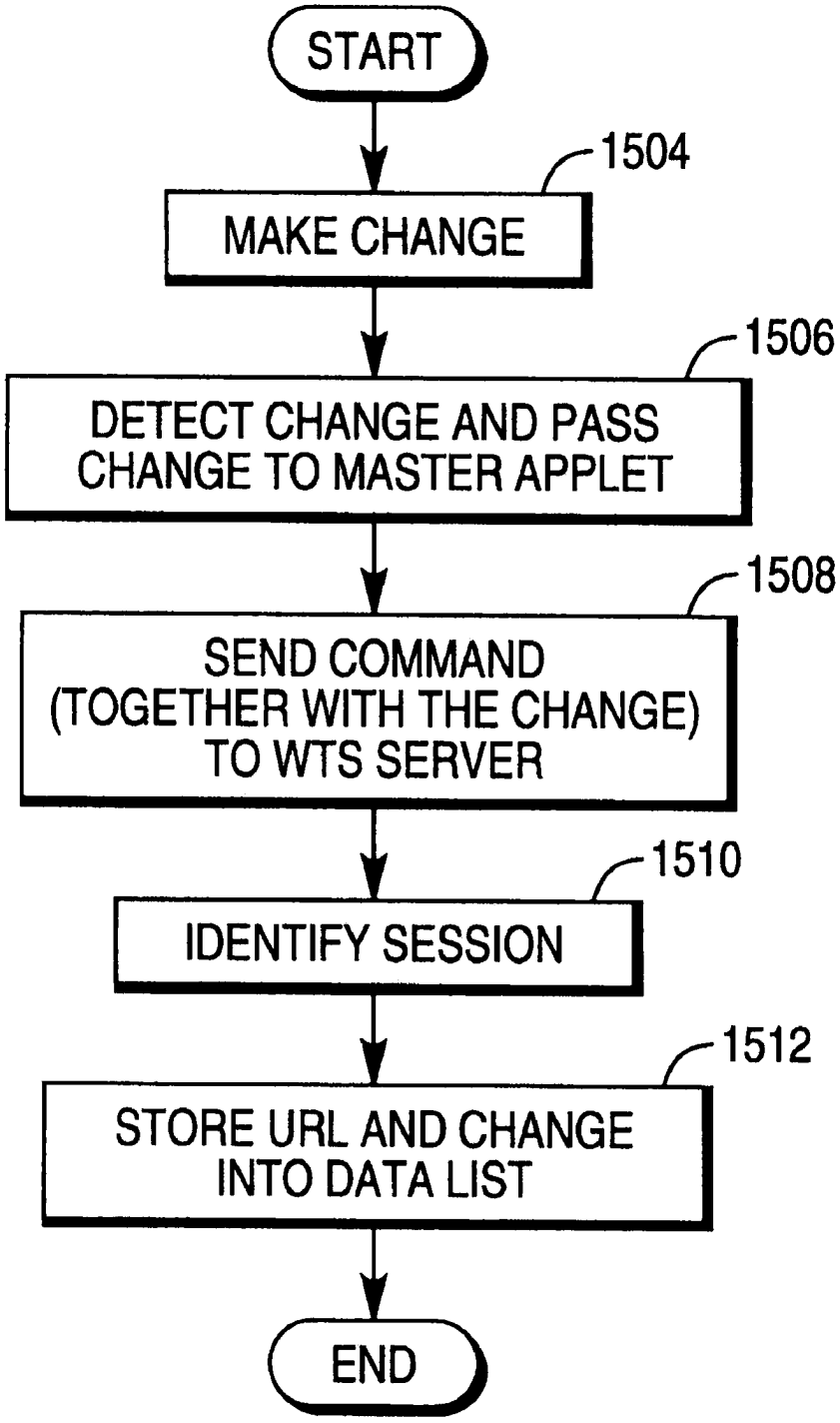

START
1504
MAKE CHANGE
1506
DETECT CHANGE AND PASS CHANGE TO MASTER APPLET
1508
SEND COMMAND (TOGETHER WITH THE CHANGE) TO WTS SERVER
1510
IDENTIFY SESSION
1512
STORE URL AND CHANGE INTO DATA LIST
END

FIG. 16

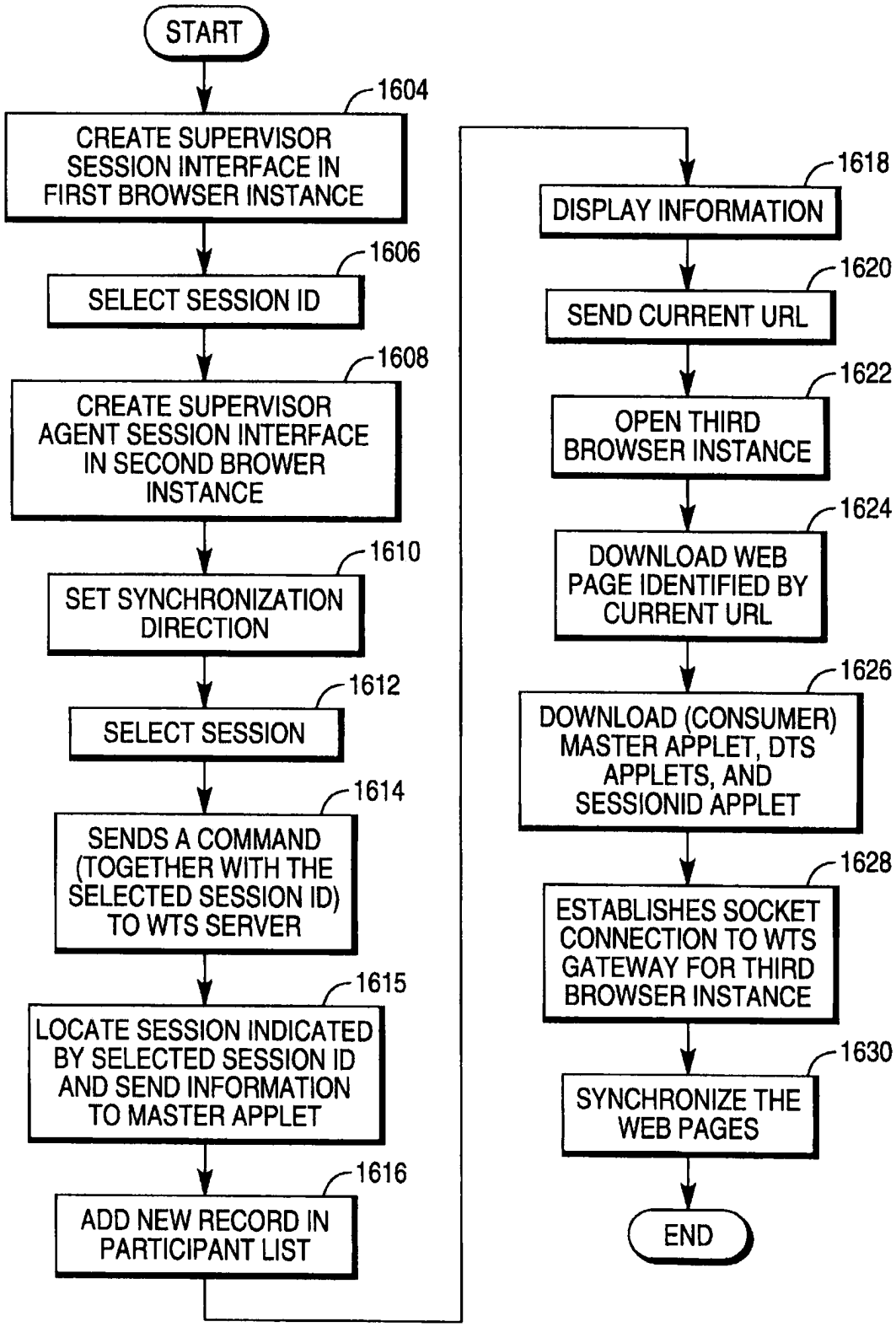

START
1604
CREATE SUPERVISOR SESSION INTERFACE IN FIRST BROWSER INSTANCE
1606
SELECT SESSION ID
1608
CREATE SUPERVISOR AGENT SESSION INTERFACE IN SECOND BROWER INSTANCE
1610
SET SYNCHRONIZATION DIRECTION
1612
SELECT SESSION
1614
SENDS A COMMAND (TOGETHER WITH THE SELECTED SESSION ID) TO WTS SERVER
1615
LOCATE SESSION INDICATED BY SELECTED SESSION ID AND SEND INFORMATION TO MASTER APPLET
1616
ADD NEW RECORD IN PARTICIPANT LIST
1618
DISPLAY INFORMATION
1620
SEND CURRENT URL
1622
OPEN THIRD BROWSER INSTANCE
1624
DOWNLOAD WEB PAGE IDENTIFIED BY CURRENT URL
1626
DOWNLOAD (CONSUMER) MASTER APPLET, DTS APPLETS, AND SESSIONID APPLET
1628
ESTABLISHES SOCKET CONNECTION TO WTS GATEWAY FOR THIRD BROWSER INSTANCE
1630
SYNCHRONIZE THE WEB PAGES
END

FIG. 17

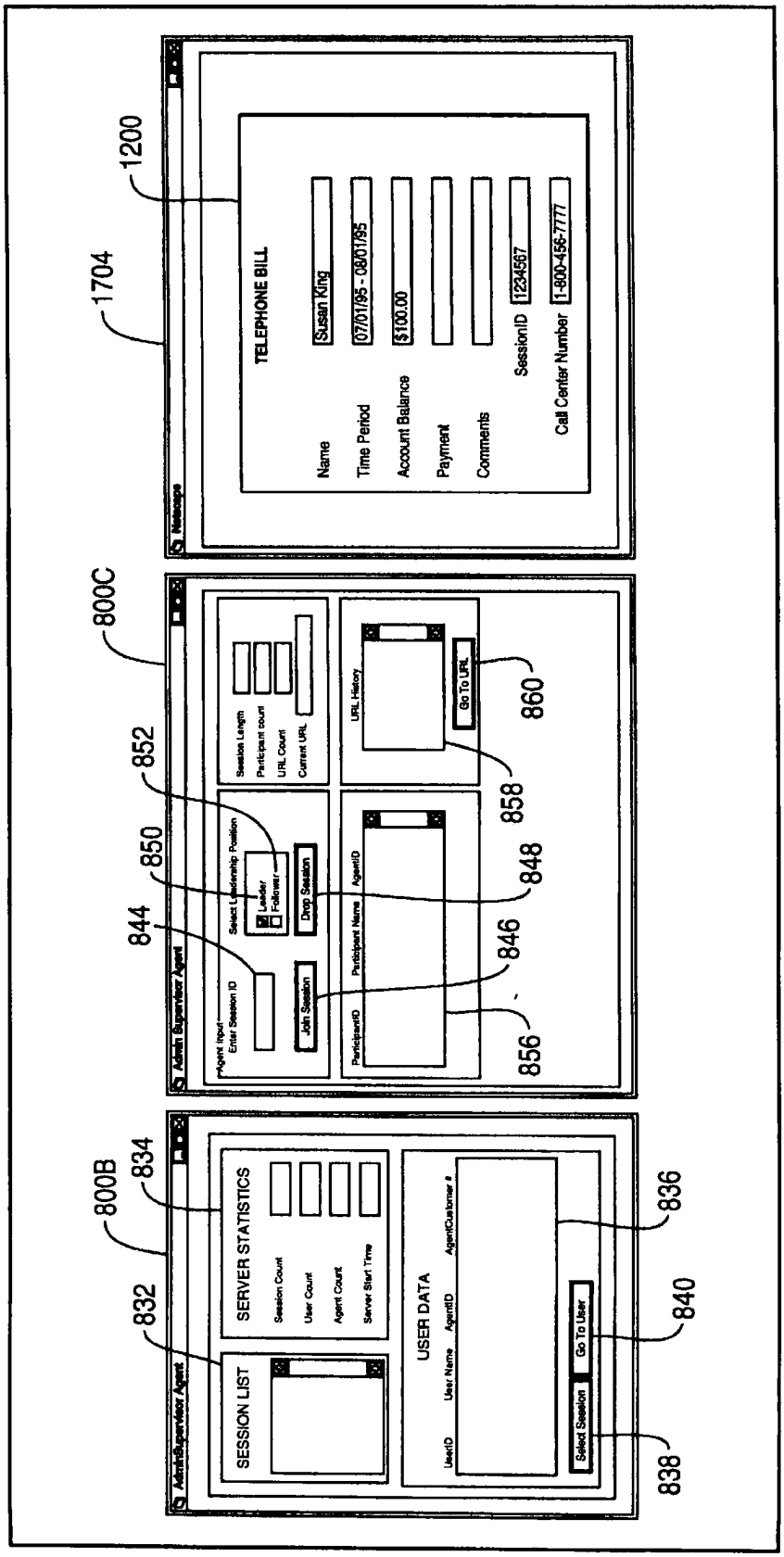
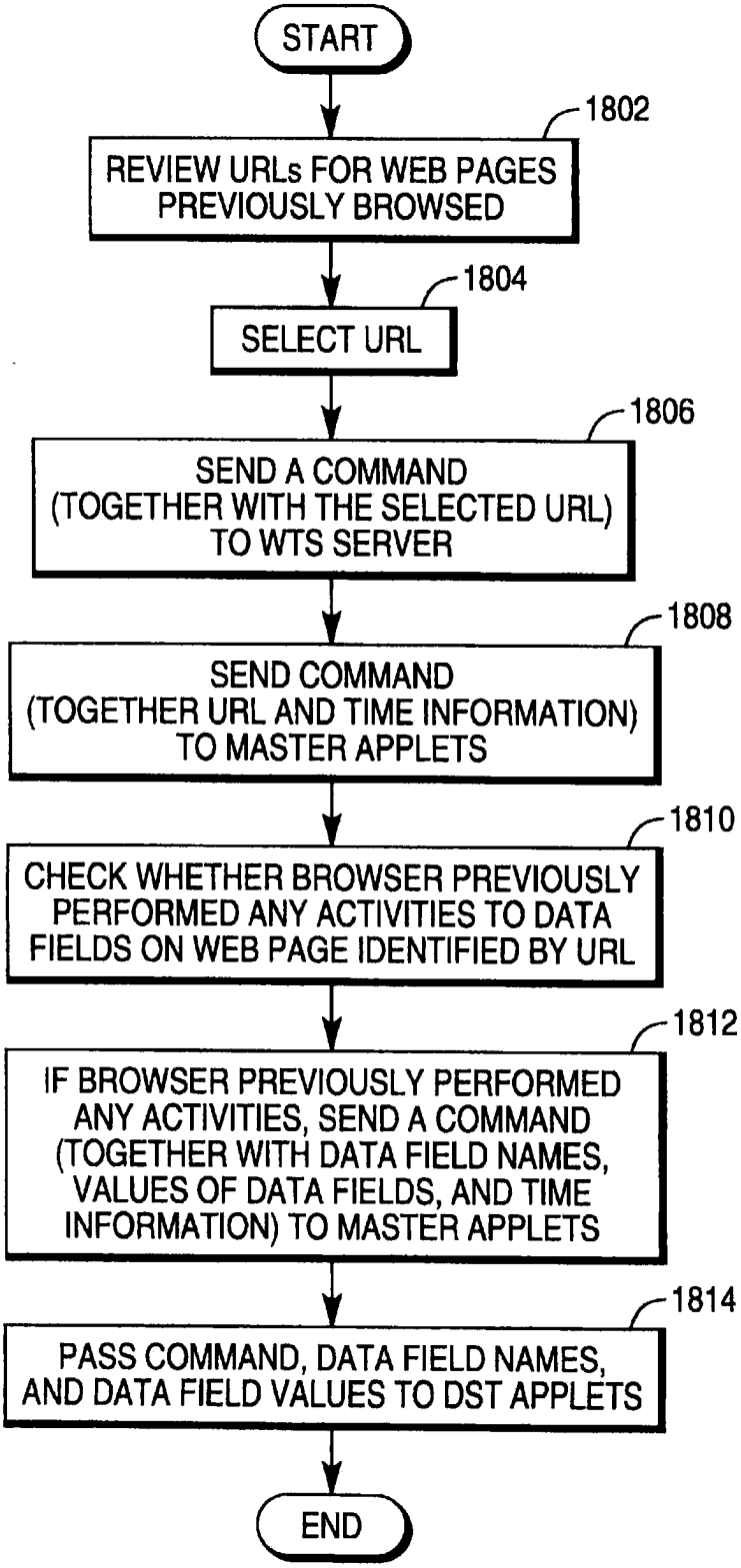
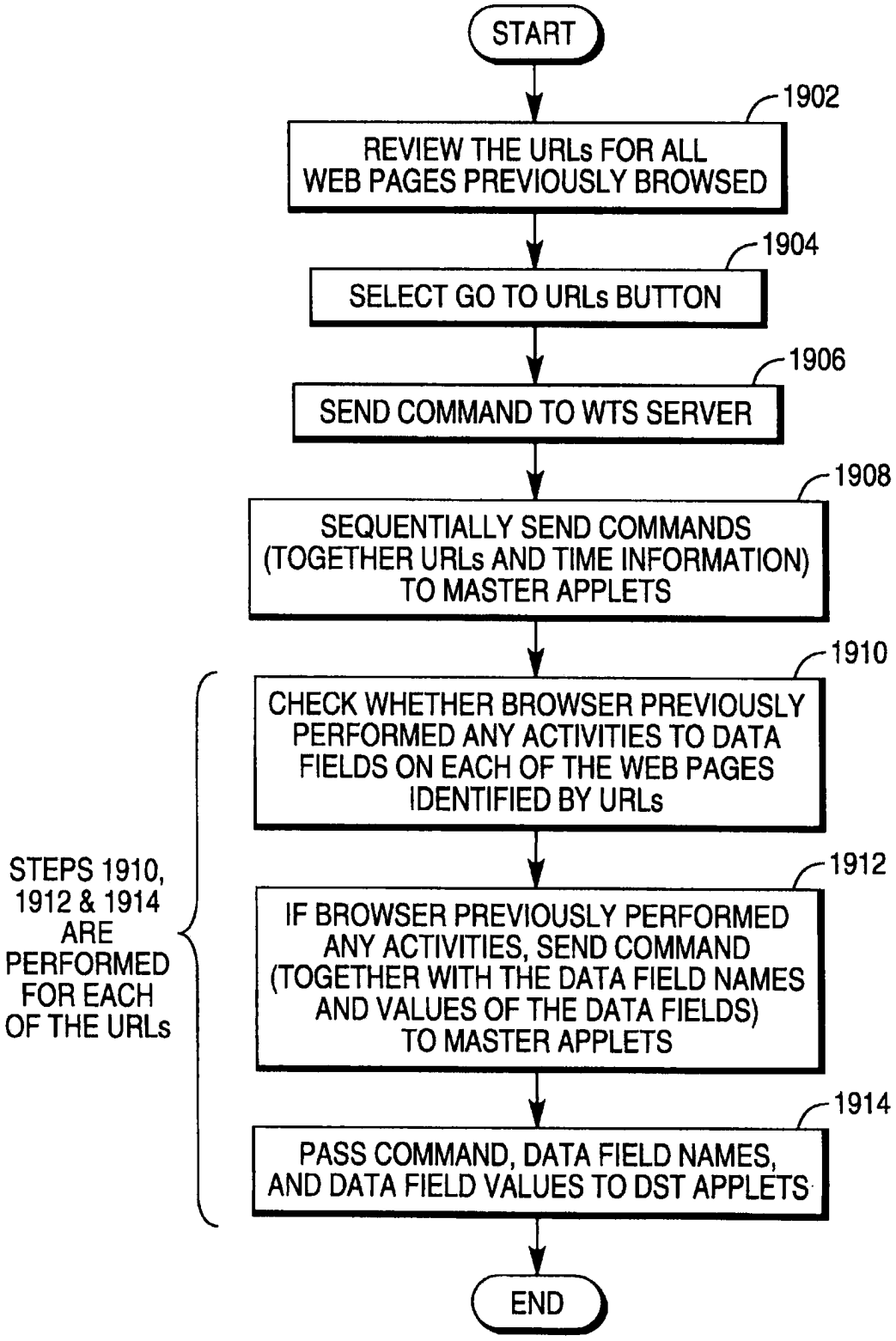

1200
1704
800C
800B
832
834
836
838
840
844
846
848
850
852
856
858
860
TELEPHONE BILL
Name
Time Period
Account Balance
Payment
Comments
Call Center Number
SessionID
Susan King
07/01/95 - 08/01/95
$100.00
1234567
1-800-456-7777
SERVER STATISTICS
SESSION LIST
USER DATA
Select Session
Go To User
Join Session
Drop Session
Go To URL

FIG. 18


START
1802
REVIEW URLs FOR WEB PAGES PREVIOUSLY BROWSED
1804
SELECT URL
1806
SEND A COMMAND (TOGETHER WITH THE SELECTED URL) TO WTS SERVER
1808
SEND COMMAND (TOGETHER URL AND TIME INFORMATION) TO MASTER APPLETS
1810
CHECK WHETHER BROWSER PREVIOUSLY PERFORMED ANY ACTIVITIES TO DATA FIELDS ON WEB PAGE IDENTIFIED BY URL
1812
IF BROWSER PREVIOUSLY PERFORMED ANY ACTIVITIES, SEND A COMMAND (TOGETHER WITH DATA FIELD NAMES, VALUES OF DATA FIELDS, AND TIME INFORMATION) TO MASTER APPLETS
1814
PASS COMMAND, DATA FIELD NAMES, AND DATA FIELD VALUES TO DST APPLETS
END

# FIG. 19


START
1902
REVIEW THE URLs FOR ALL WEB PAGES PREVIOUSLY BROWSED
1904
SELECT GO TO URLs BUTTON
1906
SEND COMMAND TO WTS SERVER
1908
SEQUENTIALLY SEND COMMANDS (TOGETHER URLs AND TIME INFORMATION) TO MASTER APPLETS
1910
CHECK WHETHER BROWSER PREVIOUSLY PERFORMED ANY ACTIVITIES TO DATA FIELDS ON EACH OF THE WEB PAGES IDENTIFIED BY URLs
1912
IF BROWSER PREVIOUSLY PERFORMED ANY ACTIVITIES, SEND COMMAND (TOGETHER WITH THE DATA FIELD NAMES AND VALUES OF THE DATA FIELDS) TO MASTER APPLETS
1914
PASS COMMAND, DATA FIELD NAMES, AND DATA FIELD VALUES TO DST APPLETS
STEPS 1910, 1912 & 1914 ARE PERFORMED FOR EACH OF THE URLs
END

# METHOD FOR MONITORING USER INTERACTIONS WITH WEB PAGES FROM WEB SERVER USING DATA AND COMMAND LISTS FOR MAINTAINING INFORMATION VISITED AND ISSUED BY PARTICIPANTS

## RELATED APPLICATIONS

This application relates to five other applications: (1) U.S. Ser. No. 08/944,757 filed Oct. 6, 1997, entitled "DEPENDABLE DATA ELEMENT SYNCHRONIZATION MECHANISM"; (2) U.S. Ser. No. 08/944,951 filed Oct. 6, 1997, entitled "DEPENDABLE DATA ELEMENT TRACKING MECHANISM"; (3) U.S. Ser. No. 08/944,121 filed Oct. 6, 1997, entitled "DEPENDABLE WEB PAGE SYNCHRONIZATION MECHANISM"; (4) U.S. Ser. No. 08/944,125 filed Oct. 6, 1997, entitled "MECHANISM FOR DEPENDABLY MANAGING WEB SYNCHRONIZATION AND TRACKING OPERATIONS AMONG MULTIPLE BROWSERS"; and (5) U.S. Ser. No. 08/944,124 filed Oct. 6, 1997, entitled "MECHANISM FOR DEPENDABLY ORGANIZING AND MANAGING INFORMATION FOR WEB SYNCHRONIZATION AND TRACKING AMONG MULTIPLE BROWSERS".

## BACKGROUND OF THE INVENTION

The present invention relates generally to a method and apparatus for coordinating access to Internet web sites by a group of web browsers that are being run at a group of user terminals.

It is known that users can retrieve information from web sites (network sites) via the Internet. The basic model for retrieving information from web sites is user initiated information searching. Specifically, a user interacts with (via a terminal) a web browser to send a request to a web site. In response to the request, the web server for the web site retrieves the information requested and sends the web browser the information arranged in so called web page (HTML) format. One of the unique features of this model is the feature of "hyper-text links" embedded in web pages that have been retrieved. This feature enables a user in searching for information to "navigate" from one web page to another. In order to provide services (or assistance) to users (or customers) via the Internet, it is desirable to provide a mechanism to track activities performed to the web pages among a group of browsers.

One method of tracking web pages navigated is to install a monitoring program at a web site. When a terminal sends requests to a web site, the monitoring program at the web side collects the URLs for the requested web pages and sends the URLs to a server. However, under this method, the monitoring program is not always able to monitor the requests from the terminal, because when the terminal retrieves web pages from its browser cache space or from a proxy server, the requests are fulfilled locally and are never sent to the web site. As a result, the URLs are not accurately tracked.

Another method of tracking web pages navigated is to install a monitoring program together with a browser at a terminal. The monitoring program constantly communicates with the web browser. When the browser sends requests out, the monitoring program collects the URLs for the web pages requested by the browser and sends the URLs to a server. However, this method requires designing and installing monitoring programs that are capable of communicating with the different browsers. At the current time, different



web browsers are manufactured by a variety of vendors, including: Netscape®, Microsoft®, Sun Microsystem, IBM, and others. For a programmer to design such a monitoring program, it requires him/her to know the details of a proprietary web browser, and it may require updating the monitoring program whenever a proprietary web browser is updated. In addition, a monitoring program designed for a web browser manufactured by one vendor is typically not portable to another web browser manufactured by another vendor because browser interface mechanisms are proprietary. Moreover, users may perceive it as intrusive to be required to install a specialized application capable of collecting and reporting the information about the web pages retrieved from all other web sites.

Therefore, there is a need for an improved method to provide more dependable web page tracking.

There is another need for an improved method to provide web page tracking without requiring knowledge of the details about the web navigation software.

There is yet another need for an improved method to design web page tracking software that is portable to different software environments.

The present invention meets these needs.

## SUMMARY OF THE INVENTION

In one aspect, the invention provides a method for tracking interactions with pages that have been loaded from a web server to a terminal, and for storing information about the interactions to a page tracking server. The method comprises the steps of:

(a) loading a first page from the web server, the first page being associated with a page locator for indicating a location of the first page in the web server, and the first page containing location information for indicating a location of a program;

(b) loading the program from the web server based on the location information, and executing the program;

(c) the program monitoring interactions with the page; and

(d) the program sending information about the interactions to the page tracking server.

In another aspect, the invention provides a method for tracking interactions with pages that have been loaded from a web server to a terminal, and for storing information about the interactions to a page tracking server. The method comprises the steps of:

(a) loading a page from the web server, the page being associated with a page locator for indicating a location of the page in the web server, and the page containing a program;

(b) executing the program;

(c) the program monitoring interactions with the page; and

(d) the program sending information about the interactions to the page tracking server.

The present invention also provides corresponding system for the respective aspects mentioned above.

## BRIEF DESCRIPTION OF THE DRAWINGS

The purpose and advantages of the present invention will be apparent to those skilled in the art from the following detailed description in conjunction with the appended drawing, in which:

FIG. **1** shows a system includes N terminals, a network, and a web site, in accordance with the present invention;

FIG. **2** shows a situation where each of the N terminals has downloaded its respective Master Applets, DTS Applets, and SessionID Applet, in accordance with the present invention;

FIG. **3** shows the process the (consumer) Master Applet, DTS Applets, and SessionID Applet being downloaded into a terminal, in accordance with the present invention;

FIG. **4** shows the process the (consumer) Master Applet, DTS Applets, and SessionID Applet being invoked, in response to loading a subsequent web page, to perform the operations in accordance with the present invention, when these Applets have been previously downloaded and cached in a terminal;

FIG. **5** shows the process of the (consumer) Master Applet, DTS Applets, and SessionID Applet being invoked, in response to loading a subsequent web page, to perform the operations in accordance with the present invention, when both these Applets and the web page have been previously downloaded and cached in a terminal;

FIG. **6** shows a session table in greater detail, in accordance with the present invention;

FIG. **7** shows how an agent (or supervisor) can create a session interface by downloading an agent page (or a supervisor page) from administration page repository **149**, in accordance with the present invention.

FIG. **8**A shows an agent session interface, in accordance with the present invention;

FIG. **8**B shows a browser supervisor session interface, in accordance with the current invention;

FIG. **8**C shows a supervisor agent session interface, in accordance with the present invention;

FIG. **9** shows a flowchart illustrating the operation of joining a session by an agent, in accordance with the present invention;

FIG. **10** shows a screen display containing two browse instances, in accordance with the present invention;

FIG. **11** shows a flowchart illustrating the operation of web page synchronization, in accordance with the present invention;

FIG. **12**A shows a web page containing five data fields, in accordance with the present invention. FIG. **12**B shows a web page that is similar to that of FIG. **12**A, except that the data in one of the five data fields is changed, in accordance with the present invention;

FIG. **13** shows a flowchart illustrating the operation of data synchronization, in accordance with the present invention;

FIG. **14** shows a flowchart illustrating the operation of web page tracking, in accordance with the present invention;

FIG. **15** shows a flowchart illustrating the operation of data tracking, in accordance with the present invention;

FIG. **16** shows a flowchart illustrating the operation of joining a session by a supervisor, in accordance with the present invention.

FIG. **17**, there shows three browser instances for a supervisor, in accordance with the present invention;

FIG. **18** shows a flowchart illustrating the operation of re-browsing a web page previously viewed in a session, in accordance with the present invention; and

FIG. **19** shows a flowchart illustrating the operation of re-browsing all web pages previously viewed in a session, in accordance with the present invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment(s) will be readily apparent to those skilled in the art, and the principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiment(s) shown, but is to be accorded with the broadest scope consistent with the principles and features disclosed herein.

Referring to FIG. **1**, there is shown an exemplary web page synchronization system **100**, in accordance with the present invention.

As shown in FIG. **1**, the system includes N terminals (**104**A, . . . , **104**K, . . . , and **104**N), a network **129** (the Internet, or a combination of the Internet and an Intranet), and a web site **134**. Each of the terminals has a telephone set (**102**A, . . . , **102**K, . . . , or **102**N) installed in its vicinity. Each of the terminals can be a PC computer, a workstation, a Java station, or even a web TV system.

Web site **134** includes a WTS (Web Tracking and Synching) gateway **142**, a WTS server **144** containing a session table **145** and a user class table **147**, a database processing application **148**, an HTTP (Hyper Text Transfer Protocol) server **152**, and a hard disk unit **154** for storing consumer page repository **146**, administration page repository **149**, and database **156**. All the components in web site **134** can be installed in one or more computer systems. Each of the computer systems includes a processing unit (which may include a plurality of processors), a memory device, and a disk unit (which may include a plurality of disk sets).

Each of the terminals (**104**A, . . . , **104**K, . . . , or **104**N) includes a processor unit (not shown) and a memory area (**115**A, . . . , **115**K, . . . , **115**N), and runs a Java enabled web browser (**114**A, . . . , **114**K, . . . , or **114**N). Each of the memory area (**115**A, . . . , **115**K, . . . , or **115**N) is maintained by its respective browser (**114**A, . . . , **114**K, . . . , or **114**N). Via network **129**, each of the browsers (**114**A, . . . , **114**K, . . . , or **114**N) is able to send requests to and receive web pages from HTTP server **152**, and to display the web pages received at its respective terminal. Each of the browsers (**114**A, . . . , **114**K, . . . , or **114**N) is able to run a Master Applet (**124**A, . . . , **124**K, . . . , or **124**N), a set of DTS (Data Tracking and Synching) Applets, a SessionID Applet, and an Agent Applet. As shown in FIG. **1**, these Applets are stored in consumer page repository **146** and can be downloaded from consumer page repository **146** and stored in the memory areas of the terminals (**104**A, . . . , **104**K, . . . , **104**N).

Referring to FIG. **2**, there is shown the situation where each of the terminals (**104**A, . . . , **104**K, . . . , or **104**N) has downloaded its respective Master Applets (**124**A, . . . , **124**K, . . . , or **124**N), DTS Applets (**126**A, . . . , **126**K, . . . , or **128**N), and SessionID Applet (**128**A, . . . , **128**K, . . . , **128**N), in accordance with the present invention.

In FIG. **2**, each of the (consumer) Master Applet (**124**A, . . . , **124**K, . . . , or **124**N) is primarily responsible for: (1) in response to loading each web page at its respective browser, opening a dedicated socket, and establishing a socket connection to WTS gateway **142** via network **129** for its respective browser (**114**A, . . . , **114**K, . . . , **114**N), (2) communicating with WTS server **144** via the socket connection, from which WTS server **144** is able identify the origin (i.e. which browser, which web page, etc.) of the commands and information that are being delivered through, (3) monitoring the activities of its respective browser, (4) sending the information about its respective browser's

activities to WTS server **144**, (5) receiving and processing the information about other browsers' activities, (6) via the socket connection, providing a single communication path to WTS server **144** for DTS Applets (**126A**, . . . , **126K**, . . . , or **126N**), SessionID Applets (**128A**, . . . , **128K**, . . . , or **128N**), or any other consumer Applets embedded on the same page with the Master Applet, (7) sending commands to WTS server **144** to request services, for itself and for DTS Applets (**126A**, . . . , **126K**, . . . , or **126N**), SessionID Applets (**128A**, . . . , **128K**, . . . , or **128N**) , or any other consumer Applets embedded on the same page with the Master Applet, and (8) sending user class information together with the commands, to indicate that its respective browser is a consumer user.

Each set of DTS Applets (**126A**, . . . , **126K**, . . . , or **126N**) contains one or more individual DTS Applets, which are primarily responsible for: (1) displaying and monitoring the data activities (data inputs or data updates of data fields) on web pages that are being displayed by its respective browser, (2) sending the data activities to WTS server **144** via its respective Master Applet, (3) receiving the data activities from other browsers via its respective Master Applet, and (4) processing the data activities from other browsers for the web pages that are being displayed by its respective browser.

Each of the SessionID Applets (**128A**, . . . , **128K**, . . . , or **128N**) is responsible for retrieving, and for displaying on a web page the current SessionID.

As shown in administration page repository **149**, Agent Applet (or Supervisor Applet) is responsible for creating a session interface, joining, monitoring, and controlling a session through the session interface. The(administration) Master Applet is primarily responsible for: (1) opening a dedicated socket, and establishing a socket connection to WTS gateway **142** via network **129** for the session interface created by Agent Applet, Supervisor Applet, or any other administration Applets embedded on the same web page with the Master Applet, (2) communicating with WTS server **144** via the socket connection, from which WTS server **144** is able identify the origin (i.e. from which session interface) of the commands and information that are being delivered through, (3) via the socket connection, providing a single communication path to WTS server **144** for Agent Applet, Supervisor Applet, or any other administration Applets embedded on the same web page with the Master Applet, and (4) sending user class information together with the commands, to indicate that its respective browser is an administration user.

WTS gateway **142** is responsible for maintaining all socket connections between Master Applets and WTS server **144**. The connections between Master Applets and WTS gateway **142** take place using standard sockets. The connection between WTS gateway **142** and WTS server **144** takes place using RMI (Remote Method Invocation).

WTS server **144** is responsible for: (1) managing and tracking the activities of all browsers participating in active sessions, exemplary activities including: loading of, interacting with, and unloading of web pages, (2) recording the information about the activities, (3) managing the synchronization of the activities for all browsers participating in the active sessions, (4) creating a session when a consumer user (via a browser) sends a request to web site **134** for the first time, (5) defining the session length intervals, (6) purging sessions that have been inactive for more than the specified session length intervals, (7) adding and deleting participants to a session, and (8) providing services to all commands from consumer Applets, such as: (consumer) Master Applet, DTS Applets, SessionID Applets, and administration Applets, such as: (administration) Master Applet, Agent Applets, and Supervisor Applet.

Consumer page repository **146** stores the web pages and Applets for consumers. Consumer Applets can be selectively embedded into consumer web pages. Exemplary consumer Applets include: (consumer) Master Applet, DTS Applets, SessionID Applet, etc.

Administration page repository **149** stores the web pages and Applets for call center administration users, including: administrator, supervisor, agent, etc. Administration Applets can be selectively embedded into administration web pages. Exemplary administration Applets include: (administration) Master Applet, Agent Applet, Supervisor Applet, etc.

To better describe the present invention, the Applets stored in (or downloaded from) repository **146** can be referred to as consumer Applets, and the Applets stored in (or downloaded from) repository **149** can be referred to as administration Applets. For example, the Master Applet stored in (or downloaded from) repository **146** can be referred to as consumer Master Applet, and the Master Applet stored in (or downloaded from) repository **149** can be referred to as administration Master Applets. HTTP server **152** contains a security application that allows consumer users to get access only to the web pages stored in consumer page repository **146**, and allows administration users (such as administrator, supervisor, agent, etc.) to get access to the web pages stored in both consumer page repository **146** and administration page repository **149**.

Session table **145** is responsible for maintaining the information for all active sessions.

Class table **147** is responsible for keeping records of user classes assigned to different users. Listed are exemplary user classes: administrator, supervisor, agent, and consumer.

Based on user classes (administrator, supervisor, agent, and consumer), WTS server **144** provides the following services:

(1) creating a session (consumer);

(2) storing data received from a session participant (supervisor, agent, and consumer);

(3) listing active sessions (administrator and supervisor);

(4) listing the information associated with active sessions (administrator, and supervisor);

(5) listing current users (administrator);

(6) joining a session (supervisor and agent);

(7) terminating a session (supervisor);

(8) monitoring a session (supervisor and agent);

(9) configuring a session parameters (administrator); and

(10) sending commands and information to a consumer Master Applet or an administration Master Applet in a participating browser (supervisor, agent, and consumer).

Database **156** is responsible for storing data collected in session table **145**.

HTTP server **152** is responsible for processing the requests issued by one of the web browsers, retrieving the web pages from either consumer page repository **146** or administration page repository **149**, and sending the web pages to the browsers that have generated these requests.

Database processing application **148** is responsible for writing the data collected in session table **145** into database **156**.

Referring to FIG. **3**, there is shown the process of how the (consumer) Master Applet, DTS Applets, and SessionID

Applet being downloaded into terminal **104A** from HTTP server **152** in response to loading an initial web page, and then being invoked to perform the operations in accordance with the present invention.

As shown in FIG. **3**, a (consumer) Master Applet, a set of DTS Applets, and a SessionID Applet are embedded into web page **204** by using a set of applet tags **208**. Web page **204** is associated with a specific URL indicating the location of web page **204** in HTTP server **152**.

As indicated by dotted line (**1**), web browser **114A** sends a request including the URL of web page **204** to HTTP server **152** via network **129**. As indicated by dotted line (**2**), in response to the request, HTTP server **152** retrieves web page **204** from consumer page repository **146** and sends it to web browser **114A** via network **129**. Web page **204** contains a set of applet tags **208**, which indicate the location of Master Applet, DTS Applets, and SessionID Applet in HTTP server **152**. As indicated by dotted line (**3**), web browser **114A** loads web page **204**. As indicated by dotted line (**4**), since Master Applet, DTS Applets, and SessionID Applet have not been downloaded, web browser **114A** sends requests via network **129**, to download these Applets based on applet tags **208**. As indicated by dotted line (**5**), HTTP server **152** sends Master Applet, DTS Applets, and SessionID Applet to browser **114A** via network **129**. As indicated by dotted line (**6**), browser **114A** stores Master Applet **124A**, DTS Applets **126A**, and SessionID Applet **128A** into memory area **115A** of terminal **104A**, and initializes and invokes these Applets. After being invoked, these Applets are running together with web browser **114A**, to monitor and process the activities for which they are assigned to be responsible. As indicated by line (**7**), Master Applet **124A** opens a dedicated socket and establishes a socket connection to WTS gateway **142** for browser **114A** and web page **204**. Via the socket connection, Master Applet **126** sends WTS server **144** a command, together with an ID unique to browser **114A**. In response to the command from Master Applet **126**, WTS server **144** creates a session for browser **114A** based on the unique ID, and issues a time stamp (loading time) indicating the time at which the command was received, and stores the URL and time stamp of web page **204** into the session created for browser **114**. As will see in the description in connection with FIG. **6** following, the URL, command, and loading time are stored in a URL history list and a command list created for the session.

Referring to FIG. **4**, there is shown the process of the (consumer) Master Applet **124A**, DTS Applets **126A**, and SessionID Applet **128A** being invoked, in response to loading a subsequent web page **214** (subsequent to web page **204**), to perform the operations in accordance with the present invention, when these Applets have been previously downloaded and cached in terminal **104A**.

As indicated by dotted line (**1**), to download web page **214**, web browser **114A** sends a request including the URL of web page **214** to HTTP server **152** via network **129**. Before loading web page **214**, the following events occur: (a) browser **114A** instructs Master Applet **124A** to run a stop routine, (b) via the socket connection established for browser **114A** and web page **204**, Master Applet **124** sends a command to inform WTS server **144** that web page **204** has been unloaded, and disconnects the socket connection established for browser **114A** and web page **204**, (c) WTS server **144** issues a time stamp (unloading time) indicating the time the command was received, and (d) records the URL and the time stamp of web page **204** into the session created for browser **114A**. As will be seen in the description in connection with FIG. **6**, following, URL, command, and unloading time are stored in a URL history list and a command list created for the session. As indicated by dotted line (**2**), HTTP server **152** retrieves web page **214** from consumer page repository **146** and sends it to browser **114A**. Like web page **204**, web page **214** contains a set of applet tags **208** for indicating the location of Master Applet **124A**, DTS Applets **126A**, and SessionID Applet **128A**. As indicated by dotted line (**3**), web browser **114A** loads web page **214**. As indicated by dotted line (**4**), in response to the loading of web page **214**, web browser **114A** locates Master Applet **124A**, DTS Applets **126A**, and SessionID Applet **128A** (based on the indication of applet tags **208**) that are cached by browser **114A** in memory area **115A**, and initializes these Applets and then invokes them. As indicated by line (**5**), Master Applet **124A** opens a dedicated socket and establishes a socket connection to WTS gateway **142** for browser **114A** and web page **214**. Via the socket connection established for browser **114A** and web page **214**, Master Applet **126A** sends a command, together with the ID unique to browser **114A** and the URL of web page **214**, to inform WTS server **144** that web page **214** has been loaded. WTS server **144** issues a time stamp (loading time) indicating the time the command was received and stores the URL and time stamp of web page into the session created for browser **114A**. As will be seen in the description in connection with FIG. **6**, following, URL, command, and loading time are stored in a URL history list and a command list created for the session.

Referring to FIG. **5**, there is shown the process of the (consumer) Master Applet **124A**, DTS Applets **126A**, and SessionID Applet **128A** being invoked, in response to loading a subsequent web page **224** (subsequent to web page **214**), to perform the operations in accordance with the present invention, when both these Applets and web page **224** have been previously downloaded and cached by browser **114A** in terminal **104A**.

As indicated by dotted line (**1**), web browser **114A** loads web page **224** cached in memory area **115A** maintained by browser **114A**. Like web pages **204** and **214**, web page **224** contains a set of applet tags **208** indicating the location of Master Applet **124A**, DTS Applets **126A**, and SessionID Applet **128A**. Before loading web page **224**, the following events occur: (a) browser **114A** instructs Master Applet **124A** to run a stop routine, (b) via the socket connection established for browser **114A** and web page **214A**, Master Applet **124A** sends a command to inform WTS server **144** that web page **214** has been unloaded, and disconnects the socket connection established for browser **114A** and web page **214**, (c) WTS server **144** issues a time stamp (unloading time) indicating the time the command was received, and (d) WTS server **144** records the URL and time stamp of web page **214** into the session created for browser **114A**. As will be seen in the description in connection with FIG. **6**, following, the URL, command, and unloading time are stored in a URL history list and a command list created for the session. As indicated by dotted line (**2**), in response to the loading of web page **224**, browser **114A** locates Master Applet **124A**, DTS Applets **126A**, SessionID Applet **128A** that have been cached by browser **114A** in memory area **115A** in terminal **104A**, and initializes and invokes these Applets. As indicated by line (**3**), Master Applet **124A** opens a dedicated socket and establishes a socket connection to WTS gateway **142** for browser **114A** and web page **224**. Via the socket connection established for browser **114A** and web page **224**, Master Applet **126A** sends a command, together with the ID unique to browser **114A** and the URL of web page **224**, to inform WTS server **144** that web page **224** has been loaded. WTS server **144** issues a time stamp

(loading time) indicating the time the command was received and stores the URL and time stamp into the session created for browser **114**. As will be seen in the description in connection with FIG. **6**, following, the URL, command, and loading time are stored in a URL history list and a command list created for the session.

In the example shown in FIG. **5**, it should be appreciated that even through no request arrives at HTTP server **144** when web page **224** is loaded from cached memory in terminal **104A**, Master Applet **124A** still sends browsing activities to WTS server **144**.

It should be noted that the processes shown in FIGS. **3–5** of loading and invoking Master Applet, DTS Applets, and SessionID Applet for terminal **104A** can also be used for terminals **104K**, . . . , **104N**.

In FIGS. **3–5**, Master Applet, DTS Applets, and SessionID Applet are all embedded into web pages **204**, **214**, and **224**. However, it should be noted that not all the Applets are required to be embedded into a web page. Depending on the desired functions to be performed, respective Applets can be selectively embedded into a web page by selectively setting applet tags in the web page. For example, if data synchronization and tracking of individual elements are not needed, the applet tags for linking DTS Applets can be eliminated from the web page. By the same token, if additional functions are needed, additional applet tags can be added into the web page to link additional Applets.

Referring to FIG. **6**, there is shown session table **145** (see FIG. **1**) in greater detail, in accordance with the present invention.

While browsers at their respective terminals are browsing through the web pages in web site **134**, WTS server **144** collects and analyzes the information about the interactions between all browsers and the web pages that have been downloaded to the browsers from web site **134**. One difficulty in collecting and analyzing such information is that browsing individual web pages in web site **134** is a stateless process. More specifically, web site **134** receives a sequence of requests from different browsers, and sends the respective web pages to the respective browsers in response to the sequence requests. Since in processing the requests from an individual browser, web site **134** does maintain a constant connection to the same browser to keep an one-to-one relationship, web site **134** has no control over, or maintain data on, the sequences of the requests from the browsers.

To meaningfully collect and analyze the information about the interactions between the browsers and web pages, a session is defined as a collection of web page interactions that occur over a given period of time from a specific browser. A session is created when a browser first hits web site **134**, and a session window (or session length interval) is defined for the session. If activities from a specific browser (identified by an ID unique to the browser, issued by a respective Master Applet) does not occur within the session window, the session is terminated and cleaned up by WTS server **144**. A session window is refreshed (reset to time zero) each time the information about the associated browser is sent to WTS server **144**. For example, if a session window is defined as 15 minutes, as long as the associated terminal has some activity every 15 minutes, the session will remain open. After 15 minutes of inactivity, the session is terminated and purged. A subsequent request from the same terminal will cause a new session to be created. After a session has been created for a terminal, one or more other terminals can join the session.

As shown in FIG. **6**, session table **145** includes M Session IDs created for M sessions respectively. Each of the session ID is associated with: (1) a session list for maintaining information about a session, (2) a participant list for maintaining information about all participant browsers in a session (note: when a session is first created, it only contains one participant), (3) a URL history list for maintaining information about all web pages visited by all participants in a session, (4) a data list for maintaining information about the data fields on the web pages visited by all participants in a session, and (5) a command list for maintaining information about all commands issued to WTS server **144** by the various participants in a session.

Typical items in a session list are: (1) SessionID for identifying a session, (2) UserName for indicating the actual name for whom the session is created, (3) StartTime for indicating the time of starting the session, (4) StopTime for indicating the time of stopping the session, and (5) SessionNotes for recording the notes of the session.

Typical fields contained in a participant list are: (1) SessionID for linking the participant list to a session, (2) ParticipantID for identifying a participant, (3) ParticipantAddresses for indicating a participant's IP address, (4) Class for indicating the user class of the participant (customer, agent, supervisor, administrator, etc.) and (5) Direction for indicating the synchronization direction for the participant browser.

Typical fields contained in a URL history list are: (1) SessionID for linking the URL history list to a session, (2) PageURL for indicating the URL of a web page visited, (3) ParticipantID for identifying a participant who visited the web page, (4) LoadingTime for indicating the loading time of the web page, and (4) UnloadingTime for indicating the unloading time of the web page.

Typical fields contained in a data list are: (1) SessionID for linking the data list to a session, (2) WasRelayed for indicating if this data field has been broadcasted, (2) FieldName for indicating the actual name of the data field, (3) DataName for indicating the name of the data field displayed on a web page, (4) DataValue for indicating the value of the data field, (5) TimeStamp for indicating the time at which this data field is updated, (6) URL for indicating the web page on which the data field was displayed, and (7) ParticipantID for indicating the participant browser who updated this data field.

Typical fields contained in a command list are: (1) SessionID for linking the data list to a session, (2) Command for indicating the specific command executed (loading a page, unloading a page, changing a data field, etc.), (3) URL for indicating the web page to which the command operated, (4) FieldPoint for indicating the data field to which the command operated, and (5) TimeStamp for indicating the time at which command was executed.

Before a session is purged from session table **145**, database processing application **147** stores the associated session list, URL history list, and command list to database **156**. The data contained in these three lists can be late used by data warehouse integration applications.

Referring to FIG. **7**, there is shown an operation for creating a session interface for an agent (or a supervisor) by downloading an agent page (or a supervisor page) from administration page repository **149**, in accordance with the present invention. In the example shown in FIG. **7**, it is assumed that administration user class (either agent user class or supervisor user class ) is assigned to terminal **104N**, so that the security application in HTTP server **152** grants the access to the web page stored in both consumer page repository **146** and administration page repository **149**.

At step **702**, an agent at terminal **144N** types in an agent URL at terminal **104N**, and browser **114N** sends the URL to

HTTP server **152**, to retrieve an agent page, in which an (administration) Master Applet and an Agent Applet are embedded. For a supervisor, he/she types in a supervisor URL at terminal **104N**, and browser **114N** sends the URL to HTTP server **152**, to retrieve a supervisor page, in which an (administration) Master Applet and a Supervisor Applet are embedded.

At step **704**, HTTP server **152** retrieves the agent page (or a supervisor page) from administration page repository **149** and sends it to browser **114N**.

At step **706**, browser **114N** downloads the agent page, in which a Master Applet (administration Master Applet) and an Agent Applet are embedded; or downloads the supervisor page, in which a Master Applet (administration Master Applet) and a Supervisor Applet are embedded.

At step **708**, browser **114N** downloads the Master Applet and Agent Applet from HTTP server **152**, initializes and invokes these Applets; or downloads the Master Applet and Supervisor Applet from HTTP server **152**, initializes and invokes these Applets.

At step **710**, Master Applet opens a dedicated socket, establishes a socket connection to WTS gateway **142**, and sends an ID unique to browser **114N** to WTS server **144**. WTS server **144** is able to identify browser **114N** based on the unique ID.

At step **712**, Agent Applet creates an agent session interface **800A** shown in FIG. **8A** for the agent user; or Supervisor Applet creates a supervisor session interface **800B** shown in FIG. **8B** for the supervisor agent.

Referring to FIG. **8A**, there is shown an agent session interface **800A** created for an agent at step **712**, in accordance with the present invention.

As shown in FIG. **8A**, the session interface contains a text box **804** for entering a session ID, a Join session button **806** for joining a session identified by the session ID, a drop button **808** for leaving a session, a leader check box **810** (selecting of which designates a browser as a leading browser in synchronization), a follower check box **812** (selecting of which designates a browser as a following browser in synchronization), a scrollable list box **816** for displaying the information contained in the participant list associated with a selected session, a scrollable list box **818** for displaying the information in an identified URL history list, and a text box **820** for displaying the information in an identified data list. If both the leader and follower check boxes **810** and **812** are selected in the agent session interface, browser **114A** acts as both leading and following browser in synchronization.

Referring to FIG. **8B**, there is shown a browser supervisor session interface **800B** created for a supervisor at step **712**, in accordance with the current invention.

As shown in FIG. **8B**, the session interface contains a scrollable list box **832** for displaying session IDs of all active sessions in session table **145** and for selecting one of the session IDs, a text box **834** for displaying relevant statistics of WTS server **144**, a multi column scrollable list box **836** for displaying details about the session selected in scrollable list box **832**, a select session button **838** for selecting a session from scrollable list box **832**. By using the information in scrollable list box **832**, a supervisor agent can monitor all active sessions. By using the information in multi column scrollable list box **836**, a supervisor can monitor operational status of a session selected from scrollable list box **832**, including: (1) whether this session is being helped by an agent, (2) user name, and (3) agent ID. By selecting select session button **838**, a supervisor can create an agent session interface as shown in FIG. **8C**.

Referring to FIG. **8C**, there is shown a supervisor agent session interface **800C**, in accordance with the present invention.

Referring to FIG. **9**, there is shown a flowchart illustrating the operation of joining a session by an agent, in accordance with the present invention.

In the example shown in FIG. **9**, it is assumed that: (1) a consumer at terminal **104A** is browsing web pages from consumer page repository **146** via browser **114A**, (2) session list **1** shown in FIG. **6** has been created for browser **114A**, (3) an agent class has been assigned to browser **114N**, (4) agent session interface **800A** shown in FIG. **8A** has been displayed on terminal **104N**; (5) a (administration) Master Applet and Agent Applet have been previously downloaded into browser **114N**, (6) a dedicated socket connection has been established for session interface **800A** displayed at terminal **104N** by the (administration) Master Applet, and (7) the agent at terminal **104A** is on duty at a call center.

As shown in FIG. **9**, at step **902**, the consumer is browsing a web page at terminal **104A**. On the web page, SessionID Applet **128A** displays the current session ID. A call center telephone number the consumer can call is also displayed on the web page.

At step **904**, the consumer is connected to the call center by dialing the telephone number via telephone **102A** (see FIG. **1**), and the call is directed by the call center to the agent.

At step **906**, the consumer tells, via telephone **102A** (see FIG. **1**), the agent the current session ID displayed. It should be noted that, instead of using the telephone, the agent can be informed of the current session ID by alternative methods. For example, the consumer can enter his/her telephone number into a special web page that contains the caller ID of the consumer along with the current session ID. This information can be stored into a special lookup table that can be used by the agent to lookup the current session ID.

At step **908**, at terminal **104N**, the agent types the current session ID into text box **804** (see FIG. **8A**).

At step **910**, in response to a loss of focus or a pressing of the Enter key, through the socket connection established for agent session interface **800A** displayed on terminal **104N**, the (administration) Master Applet at terminal **104N** sends a command to WTS server **144**, to retrieve the information in participant list **1**, URL history list **1**, and data list **1** (see FIG. **6**) for the Agent Applet.

At step **912**, WTS server **144** sends the information requested to the Agent Applet (via the Master Applet).

At step **914**, the Agent Applet at terminal **104N** displays some information from participant list **1** and URL history list **1** in (participant) scrollable list box **816** and (URL history) scrollable list box **818**, respectively.

At step **916**, the agent selects join button **806** in agent session interface **800A** displayed on terminal **104N**.

At step **918**, in response to the selection at step **916**, through the socket connection which has been established for agent session interface displayed on terminal **104N**, the (administration) Master Applet sends WTS server **144** a command to join the selected session. Based on the identification associated with the socket connection, WTS server is able to generate a ParticipantID for browser **114N** and to find the ParticipantAddress for terminal **104N**.

At step **920**, WTS server **144** stores the ParticipantID and ParticipantAddress into participant list **1**. At this step, participant list **1** includes two participant records (two rows) containing the PaticipantIDs for browsers **114A** and **114N** respectively.

At step **922**, at terminal **104N**, the agent selects: leading check box **810** or following check box **812**, or both of them.

By only selecting leader check box **810**, the activities at terminal **104**N are synchronized at terminal **104**A, but not other way around. By only selecting follower check box **812**, the activities at terminal **104**A are synchronized at terminal **104**N, but not other way around. By selecting both leader and follower check boxes **810** and **812**, the activities at terminals **104**A and **104**N are synchronized with each other (bi-directional synchronization). In response to the selection(s), through the socket connection which has been established for agent session interface **800**A, the (administration) Master Applet sends WTS server **144** a command designating the synchronization direction. WTS server **144** stores the synchronization direction information into the Direction fields of the two records in participation list **1**. In this example, it is assumed that the bi-directional synchronization has been selected for terminals **104**A and **104**N.

At step **924**, WTS server **144** sends the (administration) Master Applet the URL of the web page being currently browsed at terminal **104**A.

At step **926**, the Agent Applet at terminal **104**N opens a browser window **1004** (a second browser instance) as shown in FIG. **10**.

At step **928**, browser **114**N downloads the web page identified by the URL from consumer page repository **146**, and displays it in browser window **1004**. A (consumer) Master Applet, a set of DTS Applets, and a SessionID Applet are embedded in the web page downloaded.

At step **930**, browser **114**N downloads (consumer) Master Applet **124**N, set of DTS Applets **126**N, and SessionID Applet **128**N.

At step **932**, the web pages displayed in second browser window **1004** at terminal **104**N are being synchronized with the web pages being displayed at terminal **104**A.

After step **932**, if the agent (the first agent) at terminal **104**A needs assistance from another agent (the second agent) at terminal **104**K, the first agent can call the second agent and tell him/her the current session ID. The second agent can then join the current session using an agent session interface as shown in FIG. **8**A displayed at terminal **104**K.

Referring to FIG. **10**, there is shown a screen display containing two browser instances (**800**A and **1004**) at terminal **104**N, in accordance with the present invention.

As shown in FIG. **10**, at terminal **104**N, the first browser instance provides an agent session interface **800**A to control and monitor the current session, and the (administration) Master Applet for agent session interface **800**A establishes and maintains a socket connection for agent session interface **800**A. The second browser instance provides a browser window **1004** to display the web pages being synchronized. (Consumer) Master Applet **124**N establishes and maintains a socket connection for each web page displayed in browser window **1004**.

Referring to FIG. **11**, there is shown a flowchart illustrating the operation of web page synchronization, in accordance with the present invention.

In the example shown in FIG. **11**, it is assumed that: (1) a consumer at terminal **104**A is browsing web pages from consumer page repository **146** via browser **114**A, (2) a session has been created for browser **114**A, (3) session list **1** and participant list **1** shown in FIG. **6** have been created for the session, (4) bi-directional synchronization has been selected for terminal **104**A and all participant terminals, and (5) the (consumer) Master Applet, DTS Applets, and SessionID Applet have been downloaded into browser **104**A and all participant browsers.

As shown in FIG. **11**, at step **1104**, browser **114**A loads a web page either from consumer page repository **146** or from memory area **115**A in terminal **104**A. If Master Applet **124**A, DTS Applets **126**A, and SessionID Applet **128**A had not been download to browser **114**A, browser **114**A would download these Applets from consumer page repository **146**. However, in this example, these Applets are assumed to be downloaded.

At step **1106**, in response to the loading of the web page, browser **114**A initializes and invokes Master Applet **124**A, DTS Applets **126**A, and SessionID Applet **128**A.

At step **1108**, Master Applet **124**A: (1) opens a dedicated socket, and establishes a socket connection to WTS gateway **142** for browser **114**A and the web page loaded, and (2) via the socket connection, sends WTS server **144** a command together with an ID unique to browser **114**A and the URL of the web page loaded. Based on the unique ID, WTS server is able to identify the session created for browser **114**A.

At step **1110**, WTS server **144** identifies the session for browser **114**A.

At step **1112**, WTS server **144** locates all IP addresses assigned to participant terminals in participant list **1** (shown in FIG. **6**), and sends a command, together with the URL, to all the participant terminals (except that WTS server **144** does not sent the URL to terminal **104**A, because the URL is originated from terminal **104**A).

At step **1114**, upon receiving the command, the (consumer) Master Applets in the participant terminals initialize themselves and pass the URL to their respective browsers.

At step **1116**, the respective browsers in the participant terminals download and display the web page according to the URL.

It should be noted that, like terminal **104**A, each of the participant terminals (at which agent session interface is displayed) can lead the page synchronization using the operation shown in FIG. **11**.

Referring to FIG. **12**A, there is shown a web page containing five data fields, specifically: name **1202**, time period **1204**, account balance **1206**, payment **1208**, comments **1210**, a text box **1212** for displaying the current session ID, and a text box for displaying the call center number the consumer can call, in accordance with the present invention.

Referring to FIG. **12**B, there is shown a web page that is similar to that of FIG. **12**A, except that the data in the field of name **202** is changed from Susan King to Sue Grant and the changes are synchronized at a participant terminal, in accordance with the present invention.

Referring to FIG. **13**, there is shown a flowchart illustrating the operation of data synchronization, in accordance with the present invention.

In the example shown in FIG. **13**, it is assumed that: (1) a customer at terminal **104**A is browsing web pages via browser **114**A, (2) a session has been created for terminal **104**A, (3) session list **1** and participant list **1** shown in FIG. **6** has been created for the session, (4) terminal **104**N is one of the participants, (5) web page **1200** containing five data fields shown in FIG. **12**A is displayed on terminals **104**A and all participant terminals, (6) a bi-directional synchronization has been selected for terminal **104**A and all participant terminals, (7) the (consumer) Master Applet, DTS Applets, and SessionID Applet have been downloaded to browser **114**A and the browsers at all participant terminals, (8) the DTS Applets contains five individual Applets: DTS $Applet_1$, DTS $Applet_2$, DTS $Applet_3$, DTS $Applets_4$, and DTS $Applet_5$, (9) these five individual DTS Applets are respectively responsible for monitoring and processing the events occurred on the five data fields of web page **1200** shown in

FIG. 12A, (10) (consumer) Master Applet 124A has established a dedicated socket connection to WTS gateway 142 for web page 12A displayed at terminal 104A, and (11) the customer at terminal 104A wants to make changes to name field 1202 from Susan King to Sue Grant.

As shown in FIG. 13, at step 1304, the customer changes the name in name field 1202 from Susan King to Sue Grant.

At step 1306, in response to a loss of focus on name field 1202 or pressing the Enter key, DTS Applet$_1$ detects the change and passes the change to Master Applet 124A.

At step 1308, via the dedicated socket connection, Master Applet 124A sends WTS server 144 a command together with the change of name field 1202. Since this change is passed to WTS server 144 via the dedicated socket connection established for web page 1200, WTS server 144 is able to recognize the origin of the command, web page 1200, and the name field upon which the change was made.

At step 1310, WTS server 144 identifies the session created for browser 114A.

At step 1312, WTS server 144 locates the IP addresses assigned to participant browsers in participant list 1 and sends a command (together with the change of name field 1202) to the Master Applets in all participant terminals (except that WTS server 144 does not send the command and change to browser 114A, since this change originated from browser 114A).

At step 1314, upon receiving the command, the (consumer) Master Applets (including Master Applets 124N) pass the change of name field 1200 to their respective DTS Applets, including the DTS Applet$_1$ at browser 114N.

At step 1316, the DTS Applet$_1$, display the update "Susan Grant" into the name fields on respective web page 1200 displayed on the respective terminals, including terminal 104N.

It should be noted that the operation shown in FIG. 13 can be used to perform data synchronization for the other four data fields on web page 1200 shown in FIG. 12A.

It should also be noted that the data field synchronization can also be performed at terminal 104N. For example, as shown in FIG. 12B, when the agent at terminal 104N enters comments of "Account's name had been changed" to comments field 1210' on web page 1200', this updates will be displayed in comments field 1210 at terminal 104A, by using the operation shown in FIG. 13.

Referring to FIG. 14, there is shown a flowchart illustrating the operation of web page tracking, in accordance with the present invention.

In the example shown in FIG. 14, it is assumed that: (1) a customer at terminal 104A is browsing web pages via browser 114A, (2) a session has been created for terminal 104A and all participant terminals, (3) session list 1, participant list 1, and URL history list 1 shown in FIG. 6 have been created for the session, (4) bi-directional synchronization has been selected for terminal 104A and all participant terminals, and (5) the (consumer) Master Applet, DTS Applets, and SessionID Applet have been downloaded into terminals 104A and all participant terminals.

As shown in FIG. 14, at step 1404, browser 114A downloads a web page from either the consumer page repository 146 or memory area 115A of terminal 104A. If Master Applet 124A, DTS Applets 126A, and SessionID Applet 128A had not been download to terminal 104A, browser 114A would download them from HTTP server 152. However, in this example, these Applets have been downloaded.

At step 1406, web browser 114A initializes and invokes Master Applet 124A, DTS Applets 126A, and SessionID Applet 128A.

At step 1408, Master Applet 124A opens a dedicated socket and establishes a socket connection to WTS gateway 142 for web browser 1 14A and the web page loaded. Master Applet 124A then sends WTS server 144 a command, together with: (1) an ID unique to browser 114A, and (2) the URL of the web page loaded. When commands and URL are delivered through this socket connection, WTS server 144 is able to recognize the origin of the commands and URL.

At step 1410, WTS server 144 identifies the session ID for browser 114A.

At step 1412, WTS server 144 locates the session list 1 and URL history list 1.

At step 1414, WTS server 144 issues a time stamp (loading time) for indicating the time at which the command was received, and stores the URL and time stamp to URL history list 1.

At step 1416, browser 114A sends WTS server 144 a request to load a subsequent web page.

At step 1418, before loading the subsequent web page, via the socket connection, Master Applet 124A sends WTS server 144 a command, together with the URL, to inform WTS server 144 that the current web page has been unloaded.

At step 1420, WTS server 144 identifies the session for terminal 104A.

At step 1422, WTS server 144 locates the session list 1 and URL history list 1.

At step 1424, WTS server 144 issues a time stamp (unloading time) for indicating the time at which the command was received, and stores the URL and time stamp to URL history list 1.

At step 1426, Master Applet 124A disconnects the socket connection for the web page that has been unloaded.

Referring to FIG. 15, there is shown a flowchart illustrating the operation of data tracking, in accordance with the present invention.

In the example shown in FIG. 15, it is assumed that: (1) a customer at terminal 104A is browsing web pages via browser 114A, (2) a session has been created for terminal 104A, (3) session list 1 and participant list 1 shown in FIG. 6 has been created for the session, (4) terminal 104N is one of the participants, (5) web page 1200 containing five data fields shown in FIG. 12A is displayed on terminals 104A and all participant terminals, (6) a bi-directional synchronization has been selected for terminal 104A and all participant terminals, (7) the (consumer) Master Applet, DTS Applets, and SessionID Applet are downloaded to terminal 104A and all participant terminals, (8) the DTS Applets contains five individual Applets: DTS Applet$_1$, DTS Applet$_2$, DTS Applet$_3$, DTS Applets$_4$, and DTS Applet$_5$, (9) these five individual DTS Applets are respectively responsible for displaying, monitoring and processing the events occurred on the five data fields of web page 1200 shown in FIG. 12A, (10) Master Applet 124A has established a dedicated socket connection to WTS server 144 for web page 1200 displayed on terminal 104A, and (11) the customer at terminal 104A wants to make changes to name field 1202 from Susan King to Sue Grant.

As shown in FIG. 15, at step 1504, the customer changes the name in name field 1502 from Susan King to Sue Grant.

At step 1506, in response to a loss of focus on name field 1202 or pressing the Enter key, DTS Applet$_1$, detects the change and passes the change to Master Applet 124A.

At step 1508, via the dedicated socket connection, Master Applet 124A sends WTS server 144 a command together with the change of name field 1202. Since this change is passed to WTS server 144 via the dedicated socket connec-

tion established for web page **1200**, WTS server **144** is able to recognize the origin of the command, web page **1200**, and the name field upon which the change was made.

At step **1510**, WTS server **144** identifies the session created for terminal **104**A.

At step **1512**, WTS server **144** stores the URL and update of name field **1202** into data list **1**.

It should be noted that the operation shown in FIG. **15** can be used to perform data tracking for the other four data fields on web page **1200**, and to perform data tracking for all participant terminals.

Referring to FIG. **16**, there is shown a flowchart illustrating the operation of joining a session by a supervisor, in accordance with the present invention. In the example shown in FIG. **16**, it is assumed that the supervisor is on duty at terminal **104**K in a call center.

As shown in FIG. **16**, at step **1604**, the supervisor performs the steps shown in FIG. **7**, where the supervisor downloads a supervisor page on which a (administration) Master Applet (referred as Master-Applet$_1$) and a Supervisor Applet are imbedded. The Supervisor Applet displays a supervisor session interface (as shown in FIG. **8**B) in a first browser instance (see **800**B in FIG. **17**) on terminal **104**K. Master-Applet$_1$ maintains a dedicated socket connection to WTS gateway **142** for the first browser instance (see **800**B shown in FIG. **17**).

At step **1606**, from the first browser instance (see **800**B shown in FIG. **17**), the supervisor selects a session ID (listed in text box **832**) and then select session button **838**.

At step **1608**, in response to the selection of the select session button **838**, browser **114**K downloads a supervisor agent page, on which a (administration) Master Applet (referred as Master-Applet$_2$) and an Agent Applet are embedded. The Agent Applet creates a supervisor agent session interface **800**C (as shown in FIG. **8**C) and displays it in a second browser instance (see **800**C in FIG. **17**) on terminal **104**K. Master-Applet$_2$ opens and maintains a dedicated socket connection to WTS gateway **142** for the second browser instance (see supervisor agent session interface **800**C shown in FIG. **17**).

At step **1610**, there can be two possible scenarios. A first scenario is that: an agent has joined the selected session to help the consumer, and the supervisor wants to join the selected session as a participant. Under this scenario, the supervisor simply selects join button **846**, and leader and follower buttons **850** and **852** are both selected automatically. A second scenario is that: no agent has joined the session and the supervisor wants to join the session. Under this scenario, the supervisor joins the session just like an agent, by first selecting leader button **850** and/or follower button **852**, and then join session button **846**. In this example, it is assumed that a consumer at terminal **104**A and an agent at terminal **104**N have joined the session, and the supervisor wants to join the selected session.

At step **1612**, the supervisor selects join session button **846**.

At step **1614**, via the socket connection for the second browser instance (see supervisor agent session interface **800**C shown in FIG. **17**), the Master-Applet$_2$ sends WTS server **144** a command together with the selected session ID for the selected session.

At step **1615**, WTS server **144** locates the session indicated by the selected session ID, and sends information stored in participant list **1** and URL history list **1** (see FIG. **6**) to Master-Applet$_2$.

At step **1616**, WTS server **144** stores ParticipantID and ParticipantAddress into participant list **1** for browser **114**K.

At this step, participant list **1** includes three participant records (three rows) for browsers **114**A, **114**K, and **114**N respectively.

At step **1618**, Agent Applet at terminal **104**K displays the information stored in participant list **1**, and URL history list **1** in participant text box **856**, and URL history text box **858** (see FIG. **8**C) respectively.

At step **1620**, WTS server **144** sends Master-Applet$_2$ the URL of the web page being currently displayed at terminals **104**A and **104**N.

At step **1622**, the Agent Applet at terminal **104**K opens a third browser instance (see **1704** in FIG. **17**).

At step **1624**, browser **114**K downloads the web page identified by the URL from consumer page repository **146** (or loads the web page from memory area **115**K in terminal **104**K if it is cached there), and displays it in the third browser instance (see **1704** in FIG. **17**).

At step **1626**, browser **114**K downloads (consumer) Master Applet **124**K, DTS Applets **126**K, and SessionID Applet **128** from consumer page repository **146** according to the applet tags in the current web page (assuming that these Applets have not previously downloaded).

At step **1628**, Master Applet **124**K opens a dedicated socket and establishes a socket connection to WTS gateway **142** for the third browser instance **1704** shown in FIG. **17**.

After step **1630**, the web pages displayed in third browser instance **1704** at terminal **104**K are being synchronized with the web pages being displayed at terminals **104**A and **104**N.

Referring to FIG. **17**, there is shown three browser instances (**800**B, **800**C, and **1704**) for the supervisor in response to the steps shown in FIG. **16**, in accordance with the present invention.

Referring to FIG. **18**, there is shown a flowchart illustrating the operation of re-browsing a web page previously reviewed in a session, in accordance with the present invention.

In the example shown in FIG. **18**, it is assumed that: (1) a consumer at terminal **104**A is browsing web pages from consumer page repository **146** via browser **114**A, (2) a session list **1** shown in FIG. **6** has been created for browser **114**A, (3) an agent (or a supervisor) is on duty at terminal **104**N in a call center, and agent class (or supervisor class) has been assigned to browser **104**N, (4) at browser **114**N, the first and second browser instances for the agent as shown in FIG. **10** (or the first, second and third browser instances for the supervisor as shown in FIG. **17**) have been displayed, (5) via their respective socket connections established by their Master Applets, the first and second browser instances for the agent as shown in FIG. **10** (or the first, second and third browser instances for the supervisor as shown in FIG. **17**) have been connected to WTS gateway **142**, (6) Master Applets (**124**A and **124**N), DTS Applets (**126**A and **126**N) and SessionID Applets (**128**A and **128**N) have been downloaded into terminals **104**A and **104**N respectively, (7) the agent (or supervisor) has selected and joined the session created for browser **114**A, (8) at browser **114**N, the second browser instance for the agent as shown in FIG. **10** (or the third browser instance for the supervisor as shown in FIG. **17**) is being synchronized with browser **114**A, and (9) bi-direction synchronization has been selected for browsers **114**A and **114**N.

At step **1802**, for an agent user, via scrollable list box **818** on agent session interface shown in FIG. **10**, he/she reviews the URLs for all the web pages previously browsed by browser **114**A in the selected session. For a supervisor, via scrollable list box **858** on supervisor session interface shown in FIG. **17**, he/she reviews the URLs for all the web pages previously browsed by browser **114**A in the selected session.

At step **1804**, to display an individual web page previously browsed by browser **114A**, the agent (or supervisor) selects a URL from scrollable list box **818** (or scrollable list box **858**) and double-clicks on it.

At step **1806**, for the agent, the (agent) Master Applet or (the supervisor Master Applet) sends WTS server **144** a command together with the selected URL, via its respective socket connection.

At **1808**, WTS server **144** sends a command together with the URL and the time information (loading and unloading) to Master Applets **124A** and **124N**, so that Master Applets **124A** and **124N** can inform their respective browsers **114A** and **114N** to load and display the web page based on the URL.

At **1810**, WTS server **144** checks whether browser **114A** previously performed any activities to data fields on the web page identified by the URL, based on the information stored in URL history list **1** and data list 1. As shown in FIG. **6**, URL history 1 contains the information about: (a) participant ID of browser **114A**, (b) the URL, and (c) the loading and unloading time of the web page identified by the URL. Data list 1 contains the information about: (a) data field names for data fields, (b) value of the data fields, and (c) the times at which values of the data fields were changed.

At step **1812**, if browser **114** previously performed any activities to the data fields on the web page identified by the URL, WTS server **144** sends a command (together with the data field names, values of the data fields, and time information) to Master Applet **124A** (at browser **114A**) and Master Applet **124N** (at browser **114N**).

At step **1814**, at browser **114A**, Master Applet **124A** passes the command, data field names, and data field values to DST Applets **126A**, so that DTS Applets **124A** can display the data field values into respective data fields on the web page being displayed. At browser **114N**, Master Applet **124N** passes the command, data field names, and data field values to DST Applets **126N**, so that DTS Applets **124N** can display the data field values into respective data fields on the web page being displayed.

Since the loading time and unloading time of the URL and the setting time for a data field are recorded in URL history list **1** and data list 1, if desired, the web page identified by the URL and the activities performed to the data fields can be duplicated (loading the web page, setting data fields on the web page, and unloading the web page) according to the time information.

Referring to FIG. **19**, there is shown a flowchart illustrating the operation of re-browsing all web pages previously reviewed in a session, in accordance with the present invention.

In the example shown in FIG. **19**, it is assumed that: (1) a consumer at terminal **104A** is browsing web pages from consumer page repository **146** via browser **114A**, (2) a session list **1** shown in FIG. **6** has been created for browser **114A**, (3) an agent (or a supervisor) is on duty at terminal **104N** in a call center, and agent class (or supervisor class) has been assigned to browser **104N**, (4) at browser **114N**, the first and second browser instances for the agent as shown in FIG. **10** (or the first, second and third browser instances for the supervisor as shown in FIG. **17**) have been displayed, (5) via their respective socket connections established by their respective Master Applets, the first and second browser instances for the agent as shown in FIG. **10** (or the first, second and third browser instances for the supervisor as shown in FIG. **17**) have been connected to WTS gateway **142**, (6) Master Applets (**124A** and **124N**), DTS Applets (**126A** and **126N**) and SessionID Applets (**128A** and **128N**) have been downloaded into terminals **104A** and **104N** respectively, (7) the agent (or supervisor) has selected and joined the session created for browser **114A**, (8) at browser **114N**, the second browser instance for the agent as shown in FIG. **10** (or the third browser instance for the supervisor as shown in FIG. **17**) is being synchronized with browser **114A**, and (9) bi-direction synchronization has been selected for browsers **114A** and **114N**.

At step **1902**, for an agent user, via scrollable list box **818** on agent session interface shown in FIG. **10**, he/she reviews the URLs for all the web pages previously browsed by browser **114A** in the selected session. For a supervisor, via scrollable list box **858** on supervisor session interface shown in FIG. **17**, he/she reviews the URLs for all the web pages previously browsed by browser **114A** in the selected session.

At step **1904**, to display all web pages previously browsed by browser **114A**, the agent (or supervisor) selects Go to URLs button **820** in the agent session interface as shown in FIG. **10** (or Go to URLs button **860** in the supervisor session interface as shown in FIG. **17**.

At step **1906**, the (agent) Master Applet, or the (supervisor) Master Applet, sends a command to WTS server **144**.

At **1908**, WTS server **144** sequentially sends commands, together the URLs and time information, to Master Applets **124A** and **124N**, so that Master Applets **124A** and **124N** can inform their respective browsers **114A** and **114N** to load and display the web pages based on the URLs.

At **1910**, for each one of URLs that are sent together with the commands, WTS server **144** checks whether browser **114A** previously performed any activities to data fields on a web page identified by a respective URL, based on the information stored in URL history list **1** and data list 1.

At step **1912**, if browser **114** previously performed any activities to the data fields on the web page identified by a respective URL, WTS server **144** sends a command (together with the data field names and values of the data fields) to Master Applets **124A** (at browser **114A**) and Master Applet **124N** (at browser **114N**).

At step **1914**, at browser **114A**, Master Applet **124A** passes the command, data field names, and data field values to DST Applets **126A**, so that DTS Applets **124A** can display the data field values into respective data fields on the web page being currently displayed. At browser **114N**, Master Applet **124N** passes the command, data field names, and data field values to DST Applets **126N**, so that DTS Applets **124N** can display the data field values into respective data fields on the web page being currently displayed.

Since the loading time and unloading time of the URLs and the setting time for data fields are recorded in URL history list **1** and data list 1, if desired, all the web pages identified by the URLs and the activities performed to the data fields can be duplicated (loading the web page, setting data fields on the web page, and unloading the web page) according to the timing information.

It should be noted that, in the above-described embodiments, all the Applets (Master Applets, DTS Applets, SessionID Applets, and Agent Applet) embedded into web pages are written using Java. However, some or all of these Applets can be written using a browser script language, such as Java Script. More specifically, the codes for these Applets can be selectively written into web pages using the browser script language, instead of using applet tags to link these Applets. When a web browser downloads a web page containing the Applets written in browser script language, it stores these Applets into the memory area of the terminal on which the web browser is running, and then initializes and invokes them.

The present invention has the following advantages:

Dependable web page tracking and synchronizing—It tracks and synchronizes all user activities, even if web pages come from cached pages stored in browser cache or proxy servers.

Ease of use—It eliminates the current manual process of multiple users separately re-creating the web navigation.

Ease of execution (from users' point of view)—It does not require additional software to support the present invention. No software needs to be installed, configured, or run by a user.

Portability—It works across different operating systems at both client and server sides. On the client side, the requirement is that there be a web browser that supports Java Applets. On the server side, the requirement is that there be a Java Virtual Machine (JVM) on the same server that provides the HTTP service. Since there are JVMs practically for very operating system, the server components of the present invention have the potential to run on all the operating systems.

Compatibility—It works together with any HTTP servers from different vendors because the server components of the present invention requires no processing by HTTP servers, and thus are independent from HTTP servers.

Flexibility—Web page synchronization can be used independently in conjunction with web page tracking. Web page synchronization does not require persistent storage of any of the data tracked.

User privacy—It ensures a reasonable level of user privacy, since tracking and synchronization is limited to pages served by a web site that the information provider has control over.

Multiple HTTP server supported—It can handle the situation where a company has multiple physical servers running its web site, since the separation of the WTS gateway and server components enables a gateway to be placed on each HTTP server—each communicating with a common WTS server.

While the invention has been illustrated and described in detail in the drawing and foregoing description, it should be understood that the invention may be implemented through alternative embodiments within the spirit of the present invention. Thus, the scope of the invention is not intended to be limited to the illustration and description in this specification, but is to be defined by the appended claims.

What is claimed is:

**1**. A method for tracking interactions with pages that have been loaded from a web server to a terminal during a user session, and for storing information about the interactions to a sage tracking server, comprising the steps of:

loading a first page from the web server, the first page being associated with a page locator for indicating a location of the first page in the web server, and the first page containing location information for indicating a location of a program;

loading the program from the web server based on the location information, and executing the program;

the program monitoring interactions with the page; and

the program sending information about the interactions to the page tracking server during the session,

creating a session table using sent information about the interactions,

creating a sessionID for the session table wherein each sessionID is associated with a session list for maintaining information about a session, a participant list for maintaining information about all participant browsers in a session, a URL history list for maintaining information about all web pages visited by all participants in a session, a data list for maintaining information about the data fields on the web pages visited by all participants in a session, and a command list for maintaining information about all commands issues to the server by the various participants in a session,

wherein the data list includes data fields for a Session ID for linking the data list to a session, a WasRelayed for indicating if this data field has been broadcasted, a FieldName for indicating the actual name of the data field, a DataName for indicating the name of the data field displayed on a web page, a DataValue for indicating the value of the data field, a TimeStamp for indicating the time at which this data field is updated, a URL for indicating the web page on which the data field was displayed, and a ParticipantID for indicating the participant browser who updated this data field.

**2**. The method of claim **1**, further comprising the steps of:

loading a second page from the web server, the second page being associated with a page locator for indicating a location of the second page in the web server, the second page containing location information for indicating a location of the program loaded in said loading step; and

executing the program based on the location information in the second page;

the program monitoring interactions with another page; and

the program sending the information about the interactions with the second page to the page tracking server during the session.

**3**. The method of claim **2**, further comprising the step of:

storing the information about the interactions with the second page in the page tracking server during the session.

**4**. The method of claim **2**, comprising storing the sent information from the web servers.

**5**. The method of claim **2**, comprising collecting and analyzing the stored information about the interactions.

**6**. The method of claim **1**, further comprising the step of:

storing the information about the interactions with the first page in the page tracking server.

**7**. The method of claim **1**, the page locator being URL, and the location information being a tag.

**8**. The method of claim **1**, the web server being an HTTP server.

**9**. The method of claim **1**, comprising opening a dedicated socket between the terminal and a gateway.

**10**. The method of claim **1**, comprising informing the page tracking server that the first page has been loaded on the terminal.

**11**. The method of claim **1**, wherein a session is defined as a collection web page interactions that occur over a given period of time from a specific browser.

**12**. The method of claim **1**, further comprising creating a session table using sent information about the interactions.

**13**. The method of claim **12**, comprising creating a sessionID for the session table wherein each sessionID is associated with a session list for maintaining information about a session, a participant list for maintaining information about all participant browsers in a session, a URL history list for maintaining information about all web pages visited by all participants in a session, a data list for maintaining information about the data fields on the web

pages visited by all participants in a session, and a command list for maintaining information about all commands issues to the server by the various participants in a session.

**14**. The method of claim **13**, wherein the session list includes a sessionID for identifying a session, a user name for indicating the actual name for whom the session is created, a start time for indicating the time of starting the session, a stop time for indicating the time of stopping the session, and session notes for recording the notes of the session.

**15**. The method of claim **14**, wherein the participant list includes a sessionID for linking the participant list to a session, a participantID for identifying a participant, ParticipantAddresses for indicating a participant's IP address, a class for indicating a user class of the participant, and a direction for indicating the synchronization direction for the participant browser.

**16**. The method of claim **13**, wherein the URL history list includes a SessionID for linking the URL history list to a session, a PageURL for indicating the URL of a web page visited, a ParticipantID for identifying a participant who visited the web page, a loading time for indicating the loading time of the web page, and an unloading time for indicating the unloading time of the web page.

**17**. The method of claim **13**, wherein the command list includes data fields for a Session ID for linking the data list to session, a Command for indicating the specific command executed, a URL for indicating the web page to which the command operated, a FieldPoint for indicating the data field to which the command operated, and a TimeStamp for indicating the time at which command was executed.

**18**. The method of claim **1**, further comprising creating a session interface for an administrator to monitor the information about the interactions.

**19**. The method of claim **18**, comprising joining the administrator to a session to monitor the information about the interactions.

**20**. The method of claim **1**, further comprising creating a browser supervisor session interface to monitor the information about the interactions.

**21**. A method for tracking interactions with pages that have been loaded from a web server to a terminal during a user session, and for storing information about the interactions to a page tracking server, comprising the steps of:

loading a first page from the web server, the first page being associated with a page locator for indicating a location of the first page in the web server, and the first page containing location information for indicating a location of a program;

loading the program from the web server based on the location information, and executing the program;

the program monitoring interactions with the page; and

the program sending information about the interactions to the page tracking server during the session,

joining the administrator to a session to monitor the information about the interactions,

wherein the session interface includes a text box for entering a SessionID, a join session button for joining a session identified by the Session ID, a drop button for leaving a session, a leader check box, a follower check box, a scrollable list box for displaying the information contained in the participant list associated with a selected session, a scrollable list box for displaying the information in an identified URL history list, and a text box for displaying the information in an identified data list.

**22**. The method of claim **21**, further comprising the steps of:

loading a second page from the web server, the second page being associated with a page locator for indicating a location of the second page in the web server, the second page containing location information for indicating a location of the program loaded in said loading step; and

executing the program based on the location information in the second page;

the program monitoring interactions with another page; and

the program sending the information about the interactions with the second page to the page tracking server during the session.

**23**. The method of claim **21**, further comprising the step of:

storing the information about the interactions with the first page in the page tracking server.

**24**. The method of claim **21**, the page locator being URL, and the location information being a tag.

**25**. The method of claim **21**, the web server being an HTTP server.

**26**. The method of claim **21**, comprising opening a dedicated socket between the terminal and a gateway.

**27**. The method of claim **21**, comprising informing the page tracking server that the first page has been loaded on the terminal.

**28**. The method of claim **21**, wherein a session is defined as a collection of web page interactions that occur over a given period of time from a specific browser.

**29**. A method for tracking interactions with pages that have been loaded from a web server to a terminal during a user session, and for storing information about the interactions to a page tracking server, comprising the steps of:

loading a first page from the web server, the first page being associated with a page locator for indicating a location of the first page in the web server, and the first page containing location information for indicating a location of a program;

loading the program from the web server based on the location information, and executing the program;

the program monitoring interactions with the page; and

the program sending information about the interactions to the page tracking server during the session,

creating a browser supervisor session interface to monitor the information about the interactions,

wherein the supervisor session interface includes a scrollable list box for displaying session IDs of all active sessions in the session table and for selecting one of the session IDs, a text box for displaying relevant statistics of the server, a multi-column scrollable list box for displaying details about the session selected in the scrollable list box, a select session button for selecting a session from the scrollable list box.

**30**. The method of claim **29**, comprising monitoring operational status of a session using the scrollable list box.

**31**. The method of claim **29**, further comprising the steps of:

loading a second page from the web server, the second page being associated with a page locator for indicating a location of the second page in the web server, the second page containing location information for indicating a location of the program loaded in said loading step; and

executing the program based on the location information in the second page;

the program monitoring interactions with another page; and

the program sending the information about the interactions with the second page to the page tracking server during the session.

**32**. The method of claim **29**, further comprising the step of:

storing the information about the interactions with the first page in the page tracking server.

**33**. The method of claim **29**, the page locator being URL, and the location information being a tag.

**34**. The method of claim **29**, the web server being an HTTP server.

**35**. The method of claim **29**, comprising opening a dedicated socket between the terminal and a gateway.

**36**. The method of claim **29**, comprising informing the page tracking server that the first page has been loaded on the terminal.

**37**. The method of claim **29**, wherein a session is defined as collection of web page interactions that occur over a given period of time from a specific browser.

* * * * *

# UNITED STATES PATENT AND TRADEMARK OFFICE
# CERTIFICATE OF CORRECTION

**PATENT NO. : 6,035,332**

**DATED : March 7, 2000**

**INVENTOR(S) : Michael I. Ingrassia, Jr., et al.**

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 21, line 50, delete "sage" and substitute –page--.

Signed and Sealed this

Thirteenth Day of March, 2001

*Attest:*

*Attesting Officer*

NICHOLAS P. GODICI

*Acting Director of the United States Patent and Trademark Office*

# EXHIBIT B

#14/Amdt 3 Cs
T. McBeth-Brown 2152
11/17/00

ATTORNEY DOCKET NO.
036560.5892

PATENT APPLICATION
08/997,348



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | |
|---|---|
| In re Application of: | Peter Sung-An Huang, et al. |
| Serial No.: | 08/997,348 |
| Filing Date: | December 23, 1997 |
| Group Art Unit: | 2756 |
| Examiner: | Mehmet B. Geckil |
| Title: | GRAPHIC USER INTERFACE SYSTEM FOR A TELECOMMUNICATIONS SWITCH MANAGEMENT SYSTEM |

RECEIVED
NOV 15 2000
TC 2700 MAIL ROOM

OIPE JC38
NOV 0 9 2000
PATENT & TRADEMARK OFFICE

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on the date shown below.

Name

06 NOV 00

Date of Signature

Assistant Commissioner
for Patents
Washington, D.C. 20231

Dear Sir:

RESPONSE TO EXAMINER'S ACTION

In response to the Official Action mailed August 4, 2000, Applicants respectfully request the Examiner to reconsider the rejection of the claims in view of the following Amendments thereto and the comments as set forth below. Please amend the Application as follows.

IN THE CLAIMS

For the convenience of the Examiner, all pending claims are shown below whether or not an amendment has been made. Please amend the claims as follows.

1. (Three Times Amended) A graphic user interface system which provides displays for enabling one or more users to access a telecommunications switching system, comprising:

a remote computer having a browser for providing communications, the remote computer generating an access request by a user for access to the telecommunications switching system;

a server accessible by said browser and operable to receive the access request from the remote computer over a first communications link, the server having an application program which enables said remote computer to communicate with the telecommunications switching system, the server being capable of providing a copy of said application program to said remote computer, said remote computer running the application program separate from the server; and

a system manager building block in communication via a second communications link with said remote computer running said application program, the second communication link being separate from the first communication link, said system manager building block also being in communication with a system management interface building block for communicating with the telecommunications switching system, said system manager building block containing information related to authorized users for determining whether to permit the users to access the telecommunications switching system, the remote computer operable to access the telecommunications switching system <u>for monitoring of the telecommunications switching system</u> through and as authorized by the system manager building block using the application program.

2. The graphic user interface system of claim 1, wherein said server further comprises a server page for storing said application program.

3. The graphic user interface system of claim 1, wherein said server is a hypertext markup language (HTML) server.

4. The graphic user interface system of claim 1, wherein said system manager building block further comprises a runtime library for containing said information related to authorized users.

5. The graphic user interface system of claim 1, wherein said system manager building block is configured to maintain records related to each instance a user accesses the telecommunications switching system.

6. The graphic user interface system of claim 1, wherein said server provides said application program to said remote computer in the form of a Java Applet.

7. The graphic user interface system of claim 1, wherein said application program is configured to run on said remote computer in a stand-alone capacity without additional communication with said server, in order to communicate with said system manager building block to access said telecommunications switching system.

8. The graphic user interface system of claim 1, wherein said system manager building block and said system management interface building block conform to a common object request broker architecture (CORBA).

9. (Previously Amended) The graphic user interface system of claim 1, wherein said remote computer is in communication with said server via a first public communications link, and wherein said remote computer is in communication with said system manager building block via a second public communications link.

10. (Previously Amended) The graphic user interface system of claim 9, wherein said first and second public communications links are internet communications links.

11. (Previously Amended) The graphic user interface system of claim 9, wherein said first and second public communications links use an internet inter-ORB protocol (IIOP) communications protocol.

12. (Previously Amended) The graphic user interface system of claim 9, wherein said first and second public communications links use a hypertext transport protocol (HTTP) communications protocol.

13. The graphic user interface system of claim 1, wherein said system manager building block further comprises said server, and wherein said remote computer is in communication with said server and with said system manager building block via a single communications link.

14. The graphic user interface system of claim 13, wherein said single communications link is an internet communications link.

15. The graphic user interface system of claim 13, wherein said single communications link uses an internet inter-ORB protocol (IIOP) communications protocol.

16. The graphic user interface system of claim 13, wherein said single communications link uses a hypertext transport protocol (HTTP) communications protocol.

17. (Twice Amended) A method for accessing a telecommunications switching system, comprising the steps of:

initializing a browser running on a remote computer;

accessing a server with said browser, said server having an application program for communicating with said telecommunications switching system;

providing a copy of the application program from said server to said remote computer;

initializing said application program on said remote computer and accessing said telecommunications switching system with said application program from said remote computer;

determining whether to permit access to said telecommunications switching system by said remote computer; and

communicating with a system manager building block of the telecommunications switching system from said remote computer to provide [communication with] monitoring of the telecommunications switching system.

18. The method of claim 17, wherein said step of accessing a server further comprises the step of accessing a server page in the server, wherein the server page contains said application program for communicating between said remote computer and said telecommunications switching system.

19. The method of claim 17, wherein the server comprises a hypertext markup language (HTML) server.

20. The method of claim 17, wherein said step of determining whether to permit access to said telecommunications switching system further comprises the step of comparing information provided by a user at said remote computer with information related to authorized users contained in a runtime library in said system manager building block.

21. The method of claim 17, wherein said step of determining whether to permit access to said telecommunications switching system further comprises the steps of:

comparing identification numbers provided by a user at said remote computer with identification numbers of authorized users; and

comparing passwords provided by said user with passwords of authorized users.

22. The method of claim 17, further comprising the step of maintaining records related to each instance a user accesses the telecommunications switching system.

23. The method of claim 17, further comprising the step of establishing a socket connection between said remote computer and a system security manager client building block.

24. The method of claim 17, wherein said application program is provided to the remote computer in the form of a Java Applet.

25. The method of claim 17, wherein said step of initializing said application program on said remote computer comprises running said application program in a stand-alone capacity without further communicating with the server.

26. The method of claim 17, wherein said system manager building block conforms to a common object request broker architecture (CORBA).

27. (Amended) A graphic user interface system which provides displays for enabling one or more users to remotely access a telecommunications switching system, comprising:

a remote computer having an internet browser for accessing a first internet communications link;

a hypertext markup language (HTML) remote server accessible by said internet browser and having a server page, said server page having a Java application program which enables said remote computer to communicate with the telecommunications switching system, said remote server being in communication with said remote computer via said first internet communications link, and being capable of providing a copy of said Java application program to said remote computer in the form of a Java Applet so that said remote computer can run said copy of said Java application program in a stand-alone capacity without additional communication with said remote server; and

a system manager building block having a CORBA architecture and having a runtime library, said system manager building block being in communication via a second internet communications link with said remote computer running said copy of said Java application program, said system manager building block also being in communication with a system management interface building block having a CORBA architecture for communicating with the telecommunications switching system, said runtime library containing information related to identification numbers and passwords of authorized users for determining whether to permit the users to access the telecommunications switching system for monitoring thereof.

28. The graphic user interface launcher of claim 27, wherein said system manager building block is configured to maintain records related to each instance a user accesses the telecommunications switching system.

29. The graphic user interface launcher of claim 27, wherein said first and second internet communications links use an internet inter-ORB protocol (IIOP) communications protocol.

30. The graphic user interface launcher of claim 27, wherein said first and second internet communications links use a hypertext transport protocol (HTTP) communications protocol.

REMARKS

This Application has been carefully reviewed in light of the Official Action mailed August 4, 2000. In order to advance prosecution of the present Application, Claims 1, 17, and 27 have been amended in order to clarify and further describe various inventive concepts. Applicants respectfully request reconsideration and favorable action in this case.

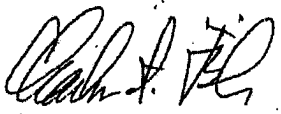
Claims 1-30 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Ingrassia, Jr., et al. Independent Claims 1, 17, and 27 recite in general the ability to access a telecommunications switching system for monitoring purposes over a second communication link from a remote computer independently from a server upon receiving an application program from the server over a first communication link separate from the second communication link. By contrast, the Ingrassia, Jr., et al. patent provides only a terminal to server occupied web site connection with no capability to access a telecommunications switching system as required by the claimed invention. The Ingrassia, Jr., et al. patent merely discloses the ability for a consumer to connect to a call center by dialing a telephone number but has no capability to monitor a telecommunications switching system. Moreover, the Ingrassia, Jr., et al. patent has a single communication link between its terminal and its web site through its network. The dotted lines identified by the Examiner as communication links in Figure 3 of the Ingrassia, Jr., et al. patent merely show process steps through the single communication link and not separate communication links. Thus, the Ingrassia, Jr., et al. patent cannot access a telecommunications switching system for monitoring thereof through a dual communication link function provided for by the claimed invention. Therefore, Applicants respectfully submit that Claims 1-30 are patentably distinct from the Ingrassia, Jr., et al. patent.

Applicants have now made an earnest attempt to place this case in condition for allowance. For the foregoing reasons, and for other reasons clearly apparent, Applicants respectfully request full allowance of Claims 1-30.

The Commissioner is hereby authorized to charge any fees or credit any overpayments to Deposit Account No. 02-0384 of BAKER BOTTS L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicants



Charles S. Fish
Reg. No. 35,870

2001 Ross Avenue
Dallas, Texas 75201-2980
(214) 953-6507
November 6, 2000