(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 October 2008 (02.10.2008)

PCT

(10) International Publication Number
WO 2008/117116 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/IB2007/004511

(22) International Filing Date:
24 December 2007 (24.12.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0706074.2 28 March 2007 (28.03.2007) GB

(71) Applicant *(for all designated States except US)*: **SKYPE LIMITED** [IE/IE]; Arthur Cox Building, Earlsfort Centre, Earlsfort Terrace, Dublin 2 (IE).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **TUUBEL, Tauri** [EE/EE]; C.R. Jakobsoni 6-3a 10128, Tallin (EE). **RICE, Liz** [GB/GB]; 2 Wyndcroft Close, Enfield EN2 7BJ (GB). **JOHN, Stuart** [GB/GB]; 3 Bernay Gardens, Bolbeck Park, Milton Keynes, Buckinghamshire MK15 8QD (GB). **KONNUSSAAR, Teet** [EE/EE]; Terase 10-11, Tallinn 10125 (EE). **TOLAN, Jill** [US/GB]; 90A Culver Road, St Albans, Hertfordshire AL14 4ED (GB).

(74) Agents: **DRIVER, Virginia, Rozanne** et al.; Page White & Farrer, Bedford House, John Street, London WC1N 2BF (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

WO 2008/117116 A2

(54) Title: DETECTION OF COMMUNICATION STATES

(57) Abstract: A method of determining an overall presence state for a user of a communication system in which the user is connected to the communication system using a plurality of devices. The method comprising: each of the plurality of devices storing in a device memory a presence state for that device; detecting a change in the presence state in at least one of the plurality of devices; each of the plurality of devices transmitting a message via the communication system to the remainder of the plurality of devices, the message comprising the presence state; receiving the messages at the remainder of the plurality of devices; and executing a decision-making code sequence in a processor at each of the remainder of the plurality of devices to determine whether to synchronise the presence state of that device with the presence state from one of the messages based on the origin of an event causing the change in presence state at the at least one of the plurality of devices.

DETECTION OF COMMUNICATION STATES

This invention relates to the detection of communication states, particularly but not exclusively for use in packet-based communication systems.

Voice over internet protocol ("VoIP") communication systems allow the user of a device, such as a personal computer, to make telephone calls across a computer network such as the Internet. These systems are beneficial to the user as they are often of significantly lower cost than traditional telephony networks, such as fixed line or mobile networks. This may particularly be the case for long distance calls. To use a VoIP service, the user must install and execute client software on their device. The client software provides the VoIP connections as well as other functions such as registration and authentication. In addition to voice communication, the client may also provide video calling and instant messaging ("IM").

One type of VoIP communication system uses a peer-to-peer ("P2P") network topology built on proprietary protocols. An example of this type of communication system is the Skype™ system. To access the peer-to-peer network, the user must execute P2P client software provided by the operator of the P2P system on their user terminal, and register with the P2P system. When the user registers with the P2P system the client software is provided with a digital certificate from a central server. Once the client software has been provided with the certificate communication can subsequently be set up and routed between users of the P2P system without the further use of a central server. In particular, the users can establish their own communication routes through the P2P system based on exchange of one or more digital certificates (or user identity certificates, "UIC") to acquire access to the P2P system. The exchange of the digital certificates between users provides proof of the user's identities and that they are suitably authorised and authenticated in the P2P system. Therefore, the presentation of digital certificates provides trust in the identity of the user. It is therefore a characteristic of peer-to-peer communication that the communication is not routed using a central server but

directly from end-user to end-user. Further details on such a P2P system are disclosed in WO 2005/009019.

One of the advantages of VoIP communication systems, compared to traditional telephony services provided over the public switched telephone network ("PSTN"), is that presence information can be provided for the users. Presence information is an indication of the current status of a user of the system. More specifically, presence information is displayed in the user interface of the client for each of the contacts that the user has stored, and allows the user to view the current status of the contacts in the system. Example presence states that may be displayed include (but are not limited to) "online", "offline", "away", "not available" and "do not disturb".

The use of presence states provides a user with prior knowledge of the current state of a contact before attempting to communicate with the contact. For example, if the user is not online, and therefore unable to be contacted, then this is clear to the user before attempting to make a call. Similarly, if a contact is busy and unlikely to answer, then this is also communicated in advance via the presence state. This is a considerable advantage over PSTN telephony systems, which do not provide any prior information on the state of a user. The only option in PSTN telephony is to dial a number and wait and see if it is answered.

However, a problem exists in a scenario in which a user of the VoIP communication system uses several different devices to access the VoIP communication system. For example, the user can use a combination of a personal computer ("PC"), personal digital assistant ("PDA"), a mobile phone, a gaming device or other embedded device to connect to the VoIP communication system. The user may use several of these devices to connect to the VoIP communication system simultaneously. Furthermore, the user may also connect to the VoIP system from a number of different locations, and these devices may remain connected, even after the user has finished using them. For example, a user may connect to the VoIP communication system from a home PC, and then subsequently connect to the VoIP

communication system from an office PC without first disconnecting the home PC.

The problem arises because each of these devices may show a different presence for the user. Therefore, the presence for the user that is displayed to other users of the VoIP system depends largely on which one of the several devices has reported its presence, rather than reflecting the actual status of the user.

There is therefore a need for a technique to address the aforementioned problems with presence information over multiple devices.

According to one aspect of the present invention there is provided a method of determining an overall presence state for a user of a communication system in which the user is connected to the communication system using a plurality of devices, the method comprising: each of the plurality of devices storing in a device memory a presence state for that device; detecting a change in the presence state in at least one of said plurality of devices; each of said plurality of devices transmitting a message via the communication system to the remainder of said plurality of devices, said message comprising the presence state; receiving said messages at the remainder of said plurality of devices; and executing a decision-making code sequence in a processor at each of said remainder of said plurality of devices to determine whether to synchronise the presence state of that device with the presence state from one of said messages based on the origin of an event causing the change in presence state at said at least one of said plurality of devices.

According to another aspect of the present invention there is provided a system for determining an overall presence state for a user of a communication system in which the user is connected to the communication system using a plurality of devices, comprising: means for storing in a device memory in each of the plurality of devices a presence state for that device; means for detecting a change in the presence state in at least one of said plurality of devices; means for transmitting, from each of said plurality of

devices, a message via the communication system to the remainder of said plurality of devices, said message comprising the presence state; means for receiving said messages at the remainder of said plurality of devices; and means for executing a decision-making code sequence in a processor at each of said remainder of said plurality of devices to determine whether to synchronise the presence state of that device with the presence state from one of said messages based on the origin of an event causing the change in presence state at said at least one of said plurality of devices.

For a better understanding of the present invention and to show how the same may be put into effect, reference will now be made, by way of example, to the following drawings in which:

Figure 1 shows a VoIP system in which users operate a plurality of devices;
Figure 2 shows an example user interface for a VoIP client;
Figure 3 shows a detailed view of a user terminal;
Figure 4 shows an example scenario with three user devices having different presence states;
Figure 5 shows a flowchart to determine a single presence state for a user with multiple devices;
Figure 6A shows a flowchart for maintaining presence information at log-in;
Figure 6B shows a flowchart for maintaining presence information following a presence change;
Figure 6C shows a flowchart for maintaining presence information following a manual log-off;
Figure 7A shows a structure for a message containing presence information;
Figure 7B shows a flowchart for determining whether to synchronise a device presence state with another device;
Figure 8 shows a flowchart for determining the overall presence state to display for a user;
Figure 9 shows the example scenario of Figure 4 with the technique to manage multiple presence states; and
Figure 10 shows presence indicator icons.

Reference is first made to Figure 1, which illustrates a VoIP communication system 100 in which users communicate using multiple devices. In the embodiment shown in Figure 1, a P2P communication system is illustrated, although it will be understood that other forms of communication could also be used.

A first user of the P2P communication system (denoted "User A" 102) operates a plurality of user devices, indicated generally at 104. All of these user devices are connected to a network 106, such as the Internet. The user devices 104 can include, for example, a personal computer ("PC") (either desktop or laptop), personal digital assistant ("PDA"), a mobile phone, an embedded VoIP device (wireless or corded); a gaming device or any other suitable device able to connect to the network 106.

In the example shown in Figure 1, User A 102 has three devices. The first is a desktop PC 108. User A interacts with desktop PC 108 via a display screen and a keyboard and mouse. The desktop PC 108 is connected to the network 106 via a network interface 110 such as a modem, and the connection between the desktop PC 108 and the network interface 110 may be via a cable (wired) connection or a wireless connection.

The desktop PC 108 is running a client 112, provided by the operator of the peer-to-peer communication system. The client 112 is a software program executed on a local processor in the desktop PC 108. The desktop PC 108 is also connected to a handset 114, which comprises a speaker and microphone to enable the user to listen and speak in a voice call in the same manner as with traditional fixed-line telephony. The handset 114 does not necessarily have to be in the form of a traditional telephone handset, but can be in the form of a headphone or earphone with an integrated microphone, or as a separate loudspeaker and microphone independently connected to the desktop PC 108

The second example device of User A is a mobile phone 116 with an embedded client 118 that allows the mobile phone 116 to connect to the VoIP

system. The mobile phone 116 can be a wifi phone, which uses a wifi wireless local area network (IEEE 802.11) connection to connect to an access point ("AP") 120. The AP 120 connects to the network 106 via a network interface 122, such as a modem. In some embodiments, the AP 120 and network interface 122 may be integrated into a single device 124. In alternative embodiments, the mobile phone can be a cellular phone running an embedded or downloaded client application.

The wifi phone 116 acts as a stand-alone device for connecting to the VoIP and making calls. As well as embedded client software 118 running on a local processor in the device, the wifi phone 116 has a display, keyboard, microphone and speaker integrated into the device to enable calls to be made over the VoIP system.

The third example device of User A's is a laptop 126. In the example shown in Figure 1, the laptop 126 communicates using wifi with the AP 120, and connects to the network 106 via network interface 122. In alternative embodiments, the laptop 126 could connect to a network interface using a wired connection. Executed on a local processor of the laptop is a client 128, which is similar to the client 112 executed on the desktop PC 108, described above. The laptop 126 has an integrated display, and may also have an integrated microphone and speakers (although these could also be separate from the laptop 126).

Note that the devices 104 used by User A 102 may all be located in the same premises, or may be geographically separated. For example, desktop PC 108 can be User A's work computer (located in his office) and wifi phone 116 and laptop 126 can be located at User A's home.

An example of a user interface 200 for the clients (112, 118, 128) executed on each of the devices 104 of User A 102 is shown illustrated in Figure 2. Note that the precise look and layout of the user interface 200 may change depending on device, due to factors such as screen size and device

capabilities. Nevertheless, the information displayed in the user interface is generally the same for all devices.

The client user interface 200 displays the username 202 of User A 102 in the P2P system, and User A can manually set his own presence state for this device by using a drop down list by selecting icon 204.

The client user interface 200 comprises a tab 206 labelled "contacts", and when this tab is selected the contacts stored by the user in a contact list are displayed. In the example user interface in Figure 2, five contacts of other users of the P2P system (User B to F) are shown listed in contact list 208. Each of these contacts have authorised the user of the client 106 to view their contact details and online presence and mood message information. Each contact in the contact list has a presence status icon associated with it. For example, the presence status icon for User B 210 indicates that User B is "online", the presence icon for User C 212 indicates that User C is "not available", the presence icon for User D 214 indicates that User D's state is "do not disturb", the presence icon for User E 216 indicates User E is "away", and the presence icon for User F 218 indicates that User F is "offline". Further presence indications can also be included, as will be described in more detail hereinafter. Next to the names of the contacts in contact list 208 are mood messages 220 of the contacts.

Figure 3 illustrates a detailed view of a typical user device (such as desktop PC 108, wifi phone 116 or laptop 126) on which is executed a client (112, 118, 128). The user device comprises a central processing unit ("CPU") 302, to which is connected a display 304 such as a screen, an input device such as a keyboard 306, a pointing device 308 (such as a mouse, joystick or directional pad), a speaker 310 and a microphone 312. The CPU 302 is connected to a communication interface 313, which allows the device to connect to the network 106. The communication interface 313 can provide a wireless or wired connection.

Figure 3 also illustrates an operating system ("OS") 314 executed on the CPU 302. Running on top of the OS 314 is a software stack 316 for the client (112, 118, 128). The software stack shows a client user interface ("UI") layer 318, a client engine layer 320, and a protocol layer 322. Each layer is responsible for specific functions. Because each layer usually communicates with two other layers only, they are regarded as being arranged in a stack as shown in Figure 3. The operating system 314 manages the hardware resources of the device and handles data being transmitted to and from the network via the communication interface 313. The client protocol layer 322 of the client software communicates with the operating system 314 and manages the network connections over the P2P system. Processes requiring higher level processing are passed to the client engine layer 320, which handles the processing required for the user to make and receive calls over the P2P system. The client engine 320 also communicates with the user client user interface layer 318. The client engine 320 may be arranged to control the client user interface layer 318 to present information to the user via the user interface of the client (as shown in Figure 2) and to receive information from the user via the user interface.

The client engine layer 320 comprises two functional blocks that are used for managing presence information between multiple devices. The first of these is a synchronisation manager 324, and the second of these is a presence engine 326. These functional blocks will be described in more detail hereinafter.

Reference is once again made to Figure 1, and the process for User A to authenticate with the P2P system and initiate a call with another user (called User B) will now be described. In this example, User A is making a call using the desktop PC 108 to User B 132.

As mentioned, the clients of User A's devices are provided with a digital certificate ("UIC") when User A 102 registers with the P2P system, and communication can subsequently be set up and routed between users of the P2P without the further use of a central server. Furthermore, subsequent to

the initial registration with the P2P system, the User A must also provide a username and password in order to log-in to the P2P system and view their contact list and make calls. In the case of the desktop PC 108 in this example, this is performed by User A entering his username and password into the client 112 running on the desktop PC 108, and the username and password is then authenticated with an authentication server (not shown). Alternatively, these authentication details may be stored by the client, so that the user does not need to manually enter them every time the client is executed, but the stored details are still passed to the authentication server to be authenticated.

The contact list for the users (e.g. the contact list 208 for User A) is stored in a contact server 130 shown in Figure 1. When the clients (112) logs into the P2P network the contact server 130 is contacted, and the contact list is downloaded to the desktop PC 108. This allows the user to log into the P2P network from any device and still access the same contact list. The clients also periodically communicate with the contact server 130 in order to obtain any changes to the information on the contacts in the contact list, or to update the stored contact list with any new contacts that have been added. Presence information is not stored centrally in the contact server. Rather, the clients periodically request the presence information for each of the contacts in the contact list 208 directly over the P2P network.

Calls to the P2P users in the contact list may be initiated over the P2P system by selecting the contact listed in the client user interface 200 and clicking on a "call" button 222 (as shown in Figure 2) using the pointing device 308. Alternatively, the call may be initiated by typing in the P2P identity of a contact in the field 224. The call set-up is performed using proprietary protocols, and the route over the network 106 between the calling user and called user is determined by the peer-to-peer system without the use of central servers. In Figure 1, an illustrative route is shown between the caller, User A (102), and the called party, User B (132), via other peers (134, 136, 138, 140) of the P2P system. It will be understood that these peers are merely an illustrative example, and that the call may be routed via fewer or more peers.

Following authentication through the presentation of the digital certificates (to prove that the users are genuine subscribers of the P2P system – described in more detail in WO 2005/009019), the call can be made using VoIP. The client 112 performs the encoding and decoding of VoIP packets. VoIP packets from the desktop PC 108 are transmitted into the Internet 106 via the network interface 110, and routed by the P2P system to all the devices that User B 132 is logged in with. For example, if User B 132 is only logged into the P2P system with a single device, a desktop PC 142, then the call is routed to the desktop PC 142 of User B 132, via a network interface 144. A client 146 (similar to the client 112) running on the desktop PC 142 of User B 132 decodes the VoIP packets to produce an audio signal that can be heard by User B 132 using the handset 148. Conversely, when User B 132 talks into handset 148, the client 146 executed on desktop PC 142 encodes the audio signals into VoIP packets and transmits them across the Internet 106 to the desktop PC 108 of User A 102. The client 112 executed on desktop PC 108 decodes the VoIP packets from User B 132, and produces an audio signal that can be heard by User A 102 using handset 114.

The VoIP packets for the P2P call described above are passed across the Internet 106 only, and the public switched telephone network ("PSTN") is not involved. Furthermore, due to the P2P nature of the network, the actual voice calls between users of the P2P network can be made with no central servers being used (central servers are only required at initial registration and authentication, and to maintain a central contact list). This has the advantages that the network scales easily and maintains a high voice quality, and the call can be made free to the users.

In common with User A 102, User B 132 may also use a number of devices to log into the P2P system. For example, User B 132 may also use a corded telephone 150 with an embedded client 152, which is shown connected to the same network interface 144 as the desktop PC 142. Furthermore, in the example shown in Figure 1, User B 132 also uses a laptop 154, which is running a client 156 and is connected to the network 106 via a network interface 158. If some or all of these devices are logged into the P2P system,

then, when a call arrives from User A 102, all of the logged-in devices ring until one of them is answered by User B 132.

It will be appreciated that the precise devices and configuration shown in Figure 1 is merely an example scenario, and that the users may have more of fewer devices in a different configuration.

The problem with having multiple devices logged into the P2P system is that the client running in each of the devices may have a different presence status. Therefore, there is no single, unified presence state for a remote user who is viewing User A's presence to display. This problem is illustrated with reference to Figure 4. Figure 4 shows an example scenario for the three devices of User A 102 – the desktop PC 108, the wifi phone 116, and the laptop 126. Initially, at time t=0, all the devices are offline (i.e. not logged into the P2P system). Then at t=1 User A logs into the P2P system using the desktop PC 108, and hence the presence state in the client 112 of the desktop PC 108 is "online". At this point, this is the only device of User A 102 logged into the P2P system (and hence only one presence state), and therefore to a remote user (e.g. User B 132) there is no problem with viewing User A's presence.

At time t=2, User A 102 logs into the P2P system with the wifi phone 116, and the client 118 executed at the wifi phone also has a presence of "online". Therefore, at t=2, two devices are logged into the P2P system for User A, and both are showing an "online" status. At t=3, the desktop PC 108 has been idle for some time, and an automatic timer in the client 112 detects this and automatically puts the presence status to "away". At this point, there is now a conflict between the presence states of the devices, as the wifi phone 116 shows "online", and the desktop PC 108 shows "away". In this situation, it is uncertain which presence state should be displayed to a remote user.

For example, if User B 132 has User A 102 in his contact list, then the client at User B's device will periodically poll User A 102 to determine his presence state. If the presence state displayed is the presence state that was most

recently reported to User B, then it could be either "away" or "online" depending on which device of User A 102 responded first. Furthermore, the presence state may even alternate between polls, as a different device may report each time. This is clearly undesirable behaviour.

The situation becomes even more complex by t=5 in Figure 4, as User A has now manually set his wifi phone 116 to "do not disturb" ("DND") (at t=4), and has subsequently logged into the laptop 126. There are now three devices, each with a different presence state. This situation of different presence states on different devices will continue until User A either manually sets all of his devices to the same presence state, or logs out of all but one of this devices.

The above-described problem is solved by the use of the two functional blocks mentioned with regards to Figure 3 – the synchronisation manager 324 and the presence engine 326.

The synchronisation manager 324 is a functional block that exists in each of the clients running on each device (as indicated in Figure 3). The purpose of the synchronisation manager 324 is to determine whether there are other clients running on other devices that are logged into the P2P system with the same username. In other words, for a given client that a particular user has logged in with, the synchronisation manager 324 determines if there are other instances of clients also operating in the P2P system for that same user. Therefore, in the example in Figure 1, the synchronisation manager in the client 112 of the desktop PC 108 determines that there are another two instances active for User A (the wifi phone 116 and laptop 126). Similarly, the synchronisation manager in the client 118 of the wifi phone 116 determines that User A 102 is also logged in using the desktop PC 108 and the laptop 126. Likewise, the synchronisation manager in the client 128 of the laptop 126 determines that User A 102 is also logged in using the desktop PC 108 and the wifi phone 116.

In addition to determining whether there are any other instances of clients logged in with the same username, the synchronisation manager 324 can preferably also determine the capabilities of the other instances.

The synchronisation manager 324 determines the information about other instances by periodically polling the P2P system for information about the specific username in question.

Once the synchronisation manager 324 has discovered the information about the other instances, then communication can be established between the instances, in order to share information. In particular, information regarding the presence states can be distributed to all the instances of a client logged in with a particular username. Specifically, the synchronisation manager periodically sends presence information to all other instances that it has discovered, as will be described in more detail hereinafter.

It is the function of the presence engine 326 to monitor the presence state of a client and maintain an accurate record of what the presence of the device is, and where that presence originated from. Furthermore, the presence engine 326 receives updates from other instances, and decides how to react to them.

More specifically, the function of the presence engine 326 and synchronisation manager 324 can be summarised as a four step process, as illustrated by the flowchart in Figure 5. In step S502, the presence engine 326 maintains a record of the presence state of the client, which includes the reason or source of the change in presence. This can be a manual presence change, an automatic presence change, or a retrieved saved presence. This will be described in more detail with reference to Figures 6A-C, below.

The presence state information is distributed to all the instances that the synchronisation manager 324 has discovered in step S504. This is illustrated in more detail in Figure 7A.

In step S506, the presence engines 326 in all the other instances receive the information about the change in presence that was distributed in step S504. Responsive to receiving the information about the change in presence in another instance (from S504), each of the remaining presence engines 326 (each associated with a different device) analyses the information and determines whether to synchronise its own presence state with the new presence state that has changed in the other instance. This process is described in more detail with reference to Figure 7B, hereinbelow. In particular, in step S506, the source of the presence change is used to determine whether to synchronise the presence states.

Finally, in step S508, the presence engine 326 in a remote device (of another user of the P2P system, e.g. User B 132) must decide on a single, unified visible presence state that is to be displayed for the user with the multiple devices (e.g. User A 102). This process is described in more detail in Figure 8, hereinafter.

Reference is now made to Figures 6A to 6C, which illustrate in more detail the process of S502 in Figure 5. As mentioned, the purpose of this process is to maintain the presence and the reason for any changes.

The presence state information is maintained by storing four items of data. The presence engine 326 maintains two separate presence state variables. The first of these is called a "set availability" state. This stores the presence state that has been set by the user of the device. The second is called a "feedback availability" state. This stores a presence state that is actually displayed to the user of the device. This is useful in the case that the presence state that is displayed in the UI of the client is different to that set by the user. A simple example of this is when the user has selected a presence state, but the client is still attempting to connect to the P2P system, as in this case the "set availability" is the presence state selected by the user, but the "feedback availability" shows a status of "connecting" in the UI of the client. The third data item stored is a single bit that indicates whether the presence stored in the "set availability" was changed by the user during the current

session – i.e. whether it was set manually by the user, or retrieved from a saved presence state from a previous session. The fourth data item is a timestamp for the time that the presence state for the "set availability" was recorded.

The presence engine can use the above information to determine the origin or source of a presence state. For example, if the change is a result of the user manually changing the presence (e.g. by using drop-down list 204 in Figure 2), then this is said to be a manual presence change. If the change in presence is a result of the client deciding to change the presence itself (e.g. changing the presence to "away" or "not available" due to inactivity at the device), then this is said to be an automatic change.

Reference is first made to Figure 6A, which illustrates the process for setting the presence when a user logs into the P2P system using a device. In step S602, the user logs into the P2P system with a device (e.g. any one of the devices 104 of User A 102). In step S604, the presence engine 326 checks whether there is a previously saved presence state for this device.

Saved presence states are advantageous because a user may deliberately set a particular state (for example "do not disturb"), but may subsequently be disconnected from the network (e.g. does to loss of a wifi connection). The saving of any manually set presence state allows this state to be recovered when the user comes back online again. If the presence state was not saved, then the user would come back with the presence "online" and not "do not disturb", which could result in the user being contacted by other users, even though he had previously explicitly set his presence to "do not disturb".

If, in step S604, it is determined that there is not a previously saved manual presence state, then the user is logged onto the network with the default presence (e.g. "online"), and this presence state is stored in both the "set availability" and "feedback availability" variables in step S606.

If, in step S604, it is determined that there is a previously saved manual presence state, then this is retrieved from the presence store 600 in step S608, and the saved presence state is set as the new presence for this device. Specifically, the retrieved saved presence state is stored in both the "set availability" and "feedback availability" variables in step S610.

Reference is now made to Figure 6B, which illustrates a flowchart for the process for updating the presence information in the case that there is a change in the presence state of a device.

In step S612, the presence engine 326 for a particular client detects a change in the presence state for the device. For example, with reference to Figure 1, the presence engine in the client 112 in the desktop PC 108 may detect that the presence state on the desktop PC 108 has changed.

At step S614, the presence engine 326 determines if the source of the change was an automatic or manual change. If it was an automatic change, then, in step S616, the new presence is recorded in the "feedback availability" variable only (and not the "set availability"). The reason for this is that the user has not actively set this presence state (hence it is not stored in the "set availability"), but the new presence state is displayed to the user in the UI of the client (hence setting the "feedback availability"). The fact that the "set availability" and "feedback availability" are now set to different values can be exploited to determine the origin of the presence change as automatic, as described later.

If S614 determines that this was a manual change, then in step S618 the presence engine stores the manual change in presence state in both the "set availability" and "feedback availability" variables. In step S620, the manual change in presence state is also stored in the presence store 600. The presence store 600 is persistent storage, such that, if the user is disconnected or logged off from the P2P system, then the manually-set presence state can be retrieved (as in S608 of Figure 6A). In step S622, a bit is set to indicate that the value stored in the "set availability" variable has been changed manually by the user during the current session. The use of this bit will be

described in more detail hereinafter. Finally, in step S624, a timestamp for the manual change in presence is recorded.

Reference is now made to Figure 6C, in which is illustrated a flowchart for maintaining the presence after a manual sign-out from the system. In step S626, the user manually selects to sign-out from the system. In step S628, both the "set availability" and "feedback availability" variables record the new presence state (which is "offline").

In further embodiments, the process in Figure 6C can include further steps (not illustrated), whereby if the user manually signs-out of one device, he is promoted with a question as to whether he wishes to sign out of all the devices he has that are active. If he chooses to do this, then messages are sent to all the devices by the synchronisation manager, instructing the devices to log-off from the P2P system.

As a result of the information maintained through the flowcharts of Figures 6A to 6C, the origin or source of a presence state can be readily determined. For example, when the "feedback availability" and "set availability" variables differ, then the origin is an "automatic" change in presence. When the bit indicating a presence change during a session is not set, then this means that the "set availability" has not been changed during this session, thereby indicating that the origin is a "saved" presence state (i.e. carried over from a previous session) or a new log-in (this is because this bit is only set in step S622 of Figure 6B, and not in Figure 6A). Otherwise, if the above conditions are not met, the origin is a "manual" change in presence.

Reference is now made to Figure 7A, which illustrates an example of the content of the messages periodically transmitted by the synchronisation manager 324 to other instances from a device, as shown in step S504 of Figure 5. Note that this is merely an example, and the precise structure and content of the message can be different in different embodiments.

The first part 702 of the message contains the value of the presence stored in the "set availability" variable. The second part 704 of the message contains the value of the bit indicating whether there has been a presence change during this session. The third part 706 of the message includes the timestamp (if any) of a manual change in presence (set in S624).

The information contained in this message is sufficient for the other instances to determine how to react to a particular presence state change in the devices.

Note that the "feedback availability" does not need to be transmitted to the other instances. This is because the "feedback availability" and "set availability" will only differ as a result of an automatic change. An automatic change in presence is not synchronised across all devices (e.g. the desktop PC 108 changing to an "away" state due to being idle does not change the presence states of the wifi phone 116 and laptop 126). Therefore, the other instances do not need to be informed of automatic changes, and hence why the "feedback availability" does not need to be sent to the other instances.

Reference is now made to Figure 7B, which illustrates in more detail the process of S506 from Figure 5. This process is performed by the presence engine 326 of a client in response to receiving the presence information from other instances (as contained in the message in Figure 7A). For example, referring to Figure 1, if the presence is changed on the desktop PC 108, then this change will be reported to the other instances (the wifi phone 116 and the laptop 126) in a similar message to that shown in Figure 7A. The process described below with reference to Figure 7B is then performed in each of the instances receiving the presence information - i.e. the wifi phone 116 and the laptop 126.

In step S708, the presence engine 326 in a client receives presence information messages from the other instances, and compiles these for comparison. In step S710, the presence engine 326 reads the presence information for the first instance.

In step S712, the bit indicating a change during the current session (704) is read. If this bit is not set, then this indicates that the "set availability" comes from a saved presence or new log-in. These presence states are not synchronised between devices. In this case, in S714 the presence engine checks if there is information from other instances to read, and if so reads the information from the next instance in S716.

Returning again to S712, if the bit has been set, then this indicates a manual change in presence in this session. In this case, the timestamp of the manual change (706) is read in step S718. This is compared to the newest known manual change in step S720, which is stored in store 722 (which is obviously initialised to a null time value before the algorithm is run).

If the timestamp of the manual change is not the newest read so far, then in step S714 the presence engine checks if there is information from other instances to read, and if so reads the information from the next instance in S716. If, however, the timestamp of the manual change is the newest read so far, then in step S724 this newest value is written to the store (overwriting any previous value). Step S714 is then returned to, wherein the presence engine checks if there is information from other instances to read, and if so reads the information from the next instance in S716.

Once all information from all the instances has been read, then the process will return to step S714, but the presence engine will determine that there is no more information from other instances to consider. The process will then move to step S726, wherein the newest manual change from all the instances (stored in 722) is read, and compared to the timestamp of any manual change on the current device (i.e. the device on which the algorithm is being run).

If, in step S726, it is found that a manual change on another instance was made more recently than on the current device, then the presence of the current device is synchronised (in step S728) to the presence of the instance with the more recent manually-set presence.

Conversely, if in step S726 it is found that a manual change on the current device was made more recently than any made on the other instances, then the presence is not synchronised in step S730.

When a device synchronises with the presence state of another instance (as in S728), this is stored in the "set availability" and "feedback availability" variables of the current device.

After the process in Figure 7B has been performed, each active instance has been informed of a change in presence on one device, and each individual instance has decided whether or not to synchronise its own presence in response to this change. However, there still exists the problem that the different instances can have different presences. Therefore, there is a need to calculate what the single presence state will be that is shown to the other users of the P2P system. The process for determining this presence is illustrated in Figure 8.

Figure 8 shows in more detail the process in step S508 of Figure 5. The process in Figure 8 is performed in the presence engine 326 of a remote user. For example, referring again to Figure 1, the three devices (112, 116, 126) of User A 102 can each have different presence states following the process in Figure 7B. Therefore, the client 146 of User B 132 (who has User A 102 in his contact list) must determine which of these presence states are to be displayed in the UI of the client 146 to User B.

In step S802 of Figure 8, the presence engine 326 of the remote user's client (e.g. client 146 of User B) reads the presence states of all the instances for his contact (e.g. by polling all the devices 104 of User A 102). The devices being polled report their "feedback availability" value to the remote user, and not their "set availability" – this is because the algorithm in Figure 8 needs to know of any automatically set presence states.

Once the "feedback availability" presence states for all the instances have been read, it is checked in S804 whether the presence states are all the same. This situation can frequently occur due to the user setting a manual presence state, which is synchronised over all the devices (see S728 in Figure 7B). If the presence states in each of the instances are all the same, then in step S806 the presence state for the user (that will be displayed in the remote user's client) is set to the presence state that all the instances have in common.

If the presence states in all the instances are not found to be the same in S804, then, in step S808, the presence states of the devices are compared to a priority list stored in table 800. In preferred embodiments, the list is ordered according to a "priority of availability". For example, seven presence states are shown below according to an example priority of availability:

1. Do not disturb
2. SkypeMe™
3. Online
4. Away
5. Not available
6. Invisible
7. Offline

Obviously, in alternative embodiments, the precise selection of the order of the presence states in the priority list could be different, and a different number of presence states can be included.

After the presence states in all the instances are compared to the priority table, then in step S810 the presence state that is highest in the list is selected as the visible presence state for the user. This presence state is then displayed in the client of the remote user.

Reference is now made to Figure 9, which illustrates the same example scenario as described above with reference to Figure 4, but in this case the

processes described in Figures 5 to 8 are utilised to manage the presence over the multiple devices. At time t=0, all the devices are offline (i.e. not logged into the P2P system). Therefore, the presence that is externally visible (to a remote user) for User A 102 (as shown on axis 902) is "offline". At time t=1 User A logs into the P2P system using the desktop PC 108, and hence the presence state in the client 112 of the desktop PC 108 is "online". At this point, this is the only device of User A 102 logged into the P2P system (and hence only one presence state), and therefore to a remote user (e.g. User B 132) the externally visible presence 902 is "online".

At time t=2, User A 102 logs into the P2P system with the wifi phone 116, and the client 118 executed at the wifi phone also has a presence of "online". The wifi phone 116 coming online does not cause a change in the status of the desktop PC 108, as this has the same presence state. The externally visible presence remains as "online" (see S804 and S806 of Figure 8).

At t=3, the desktop PC 108 has been idle for some time, and an automatic timer in the client 112 detects this and automatically puts the presence status to "away". The automatic presence change is not reported to the wifi phone 116 (as it is only recorded as "feedback availability"). Hence, the wifi phone does not synchronise its presence to the "away" status, as it is a result of an automatic change. The externally visible presence is decided according to Figure 8. In this case the desktop PC 108 has a presence of "away" and the wifi phone is "online". According to the priority of availability list shown above (which is used in this example) the highest rated presence on the devices is the "online" status of the wifi phone 116, and this is therefore maintained as the externally visible presence 902.

At t=4 User A 102 sets the status of the wifi phone 116 manually to DND. As this is a manual change, the message (as in Figure 7A) to the desktop PC 108 will reflect this change, and the desktop PC synchronises its presence to this manually changed setting as it is newer than any manual change on the desktop PC 108 (see Figure 7B). Therefore, the desktop PC presence now

shows DND. Consequently, the presence settings of all the active devices are the same, and the externally visible presence 902 is also DND.

At time t=5 in Figure 4, User A has logged into the laptop 126, and there are now three active devices. The new log-in results in the laptop presence initially being "online", which is reported to the wifi phone 116 and desktop PC 108. However, this new log-in does not override the presence at the desktop PC 108 and wifi phone 116, because the bit (704) is not set (hence not a manual change in S712 of Figure 7B). Therefore, the presence states at the desktop PC 108 and wifi phone 116 remain unchanged. Furthermore, the laptop 126 will also receive presence information from the wifi phone 116 and desktop PC 108, which indicates the manually set DND presence. This therefore results in the laptop 126 synchronising with the DND presence. The presence states for all devices is therefore DND, and the presence displayed at the remote user remains as DND.

At t=6, the laptop 126 has been idle, and automatically changes to "not available". As this is an automatic change, this is not synchronised on the wifi phone 116 and desktop PC 108. The externally visible presence 902 remains as DND, as this is a higher rated presence than "not available".

At time t=7 the wifi phone 116 goes offline. For example, User A 102 may have moved such that the wifi connection has been lost. The externally visible presence does not change (as DND is a higher priority than offline). However, when the wifi phone 116 regains a connection, and comes back online (e.g. if User A 102 moves into a region with wifi coverage) at t=8, the previously saved presence setting of DND is restored. The externally visible presence remains as DND, as all instances have this same presence state.

At t=9, the desktop PC 108 is manually changed from DND to "online" by User A 102. As this is a manual change, it is reported to all instances (S614). Both the wifi phone 116 and laptop 126 synchronise to this new manually set presence (S722), and the externally visible presence 902 displayed to a

remote user is changed to "online", as all the instances share this new presence setting (S806).

Therefore, it can be seen that the technique presented hereinbefore solves the problem of providing a single unified presence setting when a user is logged-in from multiple devices. In particular, by considering both the source of the presence and the priority of availability, the technique ensures that the presence that is displayed to remote users reflects as accurately as possible the user's intended behaviour.

In addition, in further embodiments, the presence state displayed to other users of the P2P system can also provide information regarding the type of device that the user is using. This functionality is illustrated with reference to Figure 10. The standard presence icon (in this case indicating "online") is shown at 1002. This is the same as the presence icons 204 and 210 in Figure 2. Similar icons are also shown for the other presence states (as described above with reference to Figure 2). However, these icons do not give any indication of the type of device that is being used. This problem is solved by the use of an icon such as that shown at 1004 of Figure 10. This is a device indicator icon, which can be used to indicate in the presence state that the user is using a particular type of device. The example device indicator icon illustrated in Figure 10 is the icon representing an "online" presence for a mobile device. Similar icons can also be used to indicate the other types of devices, and different presence states for these devices.

When device indicator presence icons are available, a decision process is needed to decide when they are to be displayed to other users of the P2P system, as described below. In the example below, the device indicator indicates the user of a mobile device, as in Figure 10. The device indicator could indicate different types of devices in different embodiments.

If a user is logged into the P2P system on a single mobile device, then a device indicator icon is used to display the user's presence on this type of device. For example, if User A 102 is logged-in using only the wifi phone 116,

then the presence status that is shown to other users is displayed using mobile device indicator presence icons as in 1004. Similarly, if a user is logged in with multiple devices, and all of these are mobile devices that support the same device indicators, then the presence can be displayed using these device indicator icons.

However, if a user is logged in using multiple devices and there is at least one device which is not a mobile device, then the device indicator presence icons should not necessarily be displayed. Whether or not the device indicator presence icons should be displayed depends upon which device is providing the presence that has the highest rating in the priority of availability table (see Figure 8). For example, if User A 102 is logged in using the desktop PC 108 and wifi phone 116, and both of the devices are "online", then the presence is shown with the standard (non-mobile) icon (1002 in Figure 10). However, if the non-mobile device (i.e. desktop PC 108) status drops automatically to "away" or "not available", then the externally visible presence comes from the wifi phone (which is still "online") as this has a higher rated priority of availability. As the externally visible presence comes from a mobile device (the wifi phone 116) the presence is displayed using the mobile device indicator icon 1004.

While this invention has been particularly shown and described with reference to preferred embodiments, it will be understood to those skilled in the art that various changes in form and detail may be made without departing from the scope of the invention as defined by the appendant claims.

CLAIMS:

1. A method of determining an overall presence state for a user of a communication system in which the user is connected to the communication system using a plurality of devices, the method comprising:

each of the plurality of devices storing in a device memory a presence state for that device;

detecting a change in the presence state in at least one of said plurality of devices;

each of said plurality of devices transmitting a message via the communication system to the remainder of said plurality of devices, said message comprising the presence state;

receiving said messages at the remainder of said plurality of devices; and

executing a decision-making code sequence in a processor at each of said remainder of said plurality of devices to determine whether to synchronise the presence state of that device with the presence state from one of said messages based on the origin of an event causing the change in presence state at said at least one of said plurality of devices.

2. A method according to claim 1, wherein the message further comprises the origin of the event causing the change in presence state at said at least one of said plurality of devices.

3. A method according to claim 1 or 2, wherein the method further comprises a terminal of at least one further user of the communication system collating a list of presence states from the plurality of devices and comparing the presence state in the list to a predetermined ranking and selecting and the highest ranked presence state as the overall presence state for said user.

4. A method according to claim 3, further comprising storing the overall presence state in a memory of the terminal of the at least one further user.

5. A method according to claim 3 or 4, further comprising the step of displaying the overall presence of the user in a contact list shown in a user interface of a client program executed on the terminal of at least one further user of the communications system.

6. A method according to any preceding claim, wherein the origin of the event causing the change in presence state at said at least one of said plurality of devices is one of an automatic change in presence state, a manual change in presence state, a manual log-off from the communication system, a new log-in to the communication system, or a retrieval of a saved presence state.

7. A method according to claim 6, wherein the decision-making code sequence is arranged to synchronise the presence state of that device with the presence state from said message if the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state that is more recent than a manual change on that device.

8. A method according to claim 7, wherein, in the case that the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state, the method further comprises the step of storing a bit indicating the manual change in presence state in the device memory.

9. A method according to claim 8, wherein the message further comprises said bit.

10. A method according to claim 9, wherein said decision-making code sequence is arranged to read said bit to determine whether the presence from said message is a manual change in presence state.

11. A method according to any of claims 7 to 10, wherein, in the case that the origin of the event causing the change in presence state at said at least

one of said plurality of devices is a manual change in presence state, the method further comprises the step of storing a timestamp recording the time of the manual change in presence state in the device memory.

12. A method according to claim 11, wherein the message further comprises said timestamp.

13. A method according to claim 12, wherein said decision-making code sequence is arranged to read said timestamp to determine whether the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state that is more recent than a manual change on that device.

14. A method according to any of claims 6 to 13, wherein the decision-making code sequence is arranged to not synchronise the presence state of that device with the presence state from said message if the origin of the event causing the change in presence state at said at least one of said plurality of devices is an automatic change in presence state, a manual log-off from the communication system, a new log-in to the communication system or the retrieval of a saved presence state.

15. A method according to any of claims 6 to 14, wherein the automatic change in presence state is caused by the at least one of said plurality of devices entering an idle state.

16. A method according to any of claims 6 to 15, wherein the manual change in presence state is caused by the user selecting a specific presence state in a user interface of a client program executed on said at least one of said plurality of devices.

17. A method according to any of claims 6 to 16, wherein the retrieval of a saved presence state is caused by the user logging into the communication system using said at least one of said plurality of devices, and said at least one of said plurality of devices has a saved presence stored in the device

memory corresponding to a manual presence setting previously selected by the user.

18. A method according to claim 5, wherein the step of displaying further comprises the steps of receiving information from a device associated with the highest ranked presence state regarding at least one characteristic of said device associated with the highest ranked presence state, and displaying a presence indicator that coveys information regarding said at least one characteristic.

19. A method according to any preceding claim, wherein the communication system is a voice over internet protocol communication system

20. A method according to claim 14, wherein the voice over internet protocol communication system is a peer-to-peer communication system.

21. A system for determining an overall presence state for a user of a communication system in which the user is connected to the communication system using a plurality of devices, comprising:

means for storing in a device memory in each of the plurality of devices a presence state for that device;

means for detecting a change in the presence state in at least one of said plurality of devices;

means for transmitting, from each of said plurality of devices, a message via the communication system to the remainder of said plurality of devices, said message comprising the presence state;

means for receiving said messages at the remainder of said plurality of devices; and

means for executing a decision-making code sequence in a processor at each of said remainder of said plurality of devices to determine whether to synchronise the presence state of that device with the presence state from one of said messages based on the origin of an event causing the change in presence state at said at least one of said plurality of devices.

22. A system according to claim 21, wherein the message further comprises the origin of the event causing the change in presence state at said at least one of said plurality of devices.

23. A system according to claim 21 or 22, wherein the system further comprises a terminal of at least one further user of the communication system comprising means for collating a list of presence states from the plurality of devices and means for comparing the presence state in the list to a predetermined ranking and selecting and the highest ranked presence state as the overall presence state for said user.

24. A system according to claim 23, further comprising means for storing the overall presence state in a memory of the terminal of the at least one further user.

25. A system according to claim 23 or 24, further comprising means for displaying the overall presence of the user in a contact list shown in a user interface of a client program executed on the terminal of at least one further user of the communications system.

26. A system according to any preceding claim, wherein the origin of the event causing the change in presence state at said at least one of said plurality of devices is one of an automatic change in presence state, a manual change in presence state, a manual log-off from the communication system, a new log-in to the communication system, or a retrieval of a saved presence state.

27. A system according to claim 26, wherein the decision-making code sequence is arranged to synchronise the presence state of that device with the presence state from said message if the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state that is more recent than a manual change on that device.

28. A system according to claim 27, further comprising means for storing a bit indicating the manual change in presence state in the device memory in the case that the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state.

29. A system according to claim 28, wherein the message further comprises said bit.

30. A system according to claim 29, wherein said decision-making code sequence is arranged to read said bit to determine whether the presence from said message is a manual change in presence state.

31. A system according to any of claims 27 to 30, further comprising means for storing a timestamp recording the time of the manual change in presence state in the device memory in the case that the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state.

32. A system according to claim 31, wherein the message further comprises said timestamp.

33. A system according to claim 32, wherein said decision-making code sequence is arranged to read said timestamp to determine whether the origin of the event causing the change in presence state at said at least one of said plurality of devices is a manual change in presence state that is more recent than a manual change on that device.

34. A system according to any of claims 26 to 33, wherein the decision-making code sequence is arranged to not synchronise the presence state of that device with the presence state from said message if the origin of the event causing the change in presence state at said at least one of said plurality of devices is an automatic change in presence state, a manual log-off

from the communication system, a new log-in to the communication system or the retrieval of a saved presence state.

35. A system according to any of claims 26 to 34, wherein the automatic change in presence state is caused by the at least one of said plurality of devices entering an idle state.

36. A system according to any of claims 26 to 35, wherein the manual change in presence state is caused by the user selecting a specific presence state in a user interface of a client program executed on said at least one of said plurality of devices.

37. A system according to any of claims 26 to 36, wherein the retrieval of a saved presence state is caused by the user logging into the communication system using said at least one of said plurality of devices, and said at least one of said plurality of devices has a saved presence stored in the device memory corresponding to a manual presence setting previously selected by the user.

38. A system according to claim 25, wherein the means for displaying further comprises means for receiving information from a device associated with the highest ranked presence state regarding at least one characteristic of said device associated with the highest ranked presence state, and means for displaying a presence indicator that coveys information regarding said at least one characteristic.

39. A system according to any preceding claim, wherein the communication system is a voice over internet protocol communication system

40. A system according to claim 14, wherein the voice over internet protocol communication system is a peer-to-peer communication system.

41. A computer program product comprising program code means which when executed by a computer implement the steps according to the method of any of claims 1 to 20.

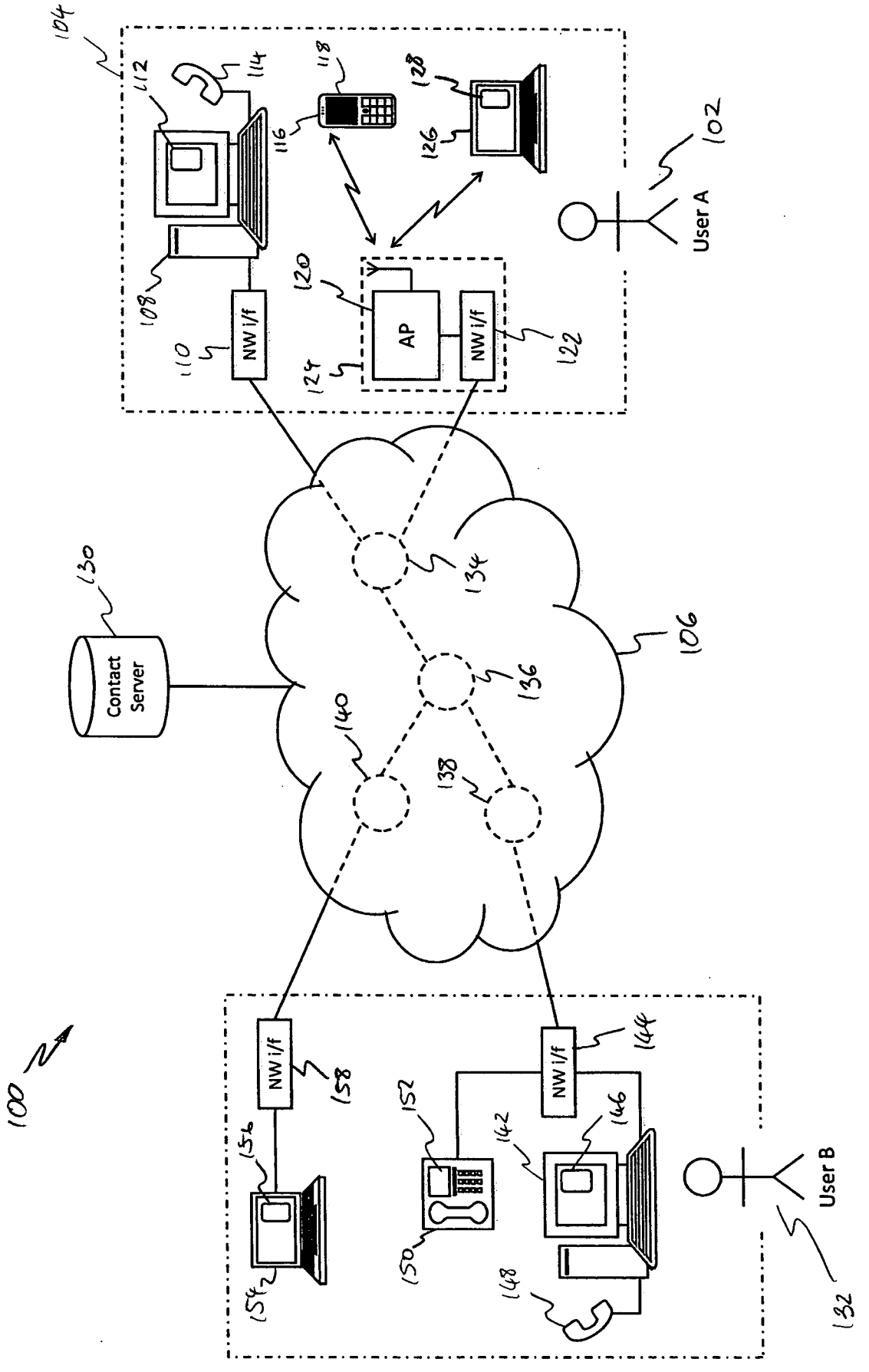
100
104
112
114
116
118
128
126
108
110
NW i/f
120
AP
NW i/f
122
124
User A
102
130
Contact Server
134
136
138
140
106
158
156
154
152
150
142
146
144
148
User B
132

FIGURE 1

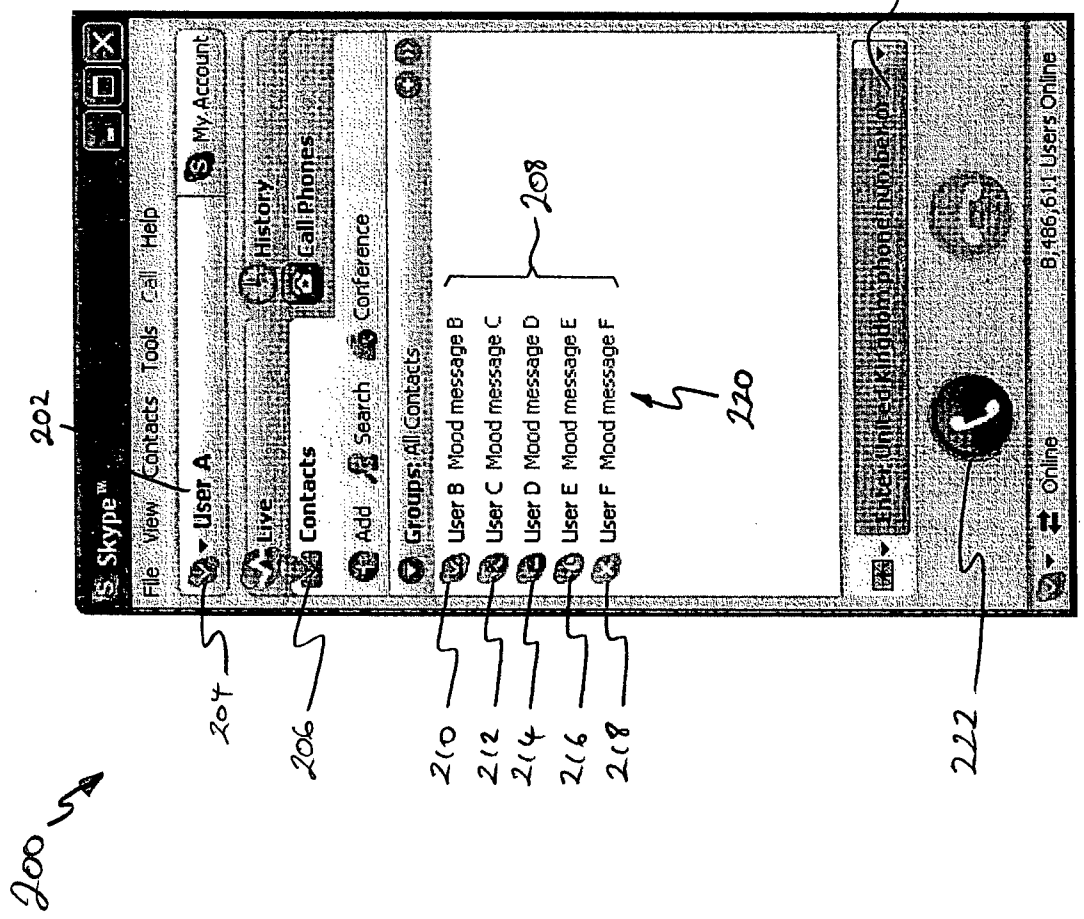
200
202
Skype™
File View Contacts Tools Call Help
User A
My Account
Live
History
Contacts
Call Phones
Add
Search
Conference
Groups: All Contacts
User B Mood message B
User C Mood message C
User D Mood message D
User E Mood message E
User F Mood message F
Enter United Kingdom phone number
Online
8,486,611 Users Online
204
206
208
210
212
214
216
218
220
222
224

FIGURE 2

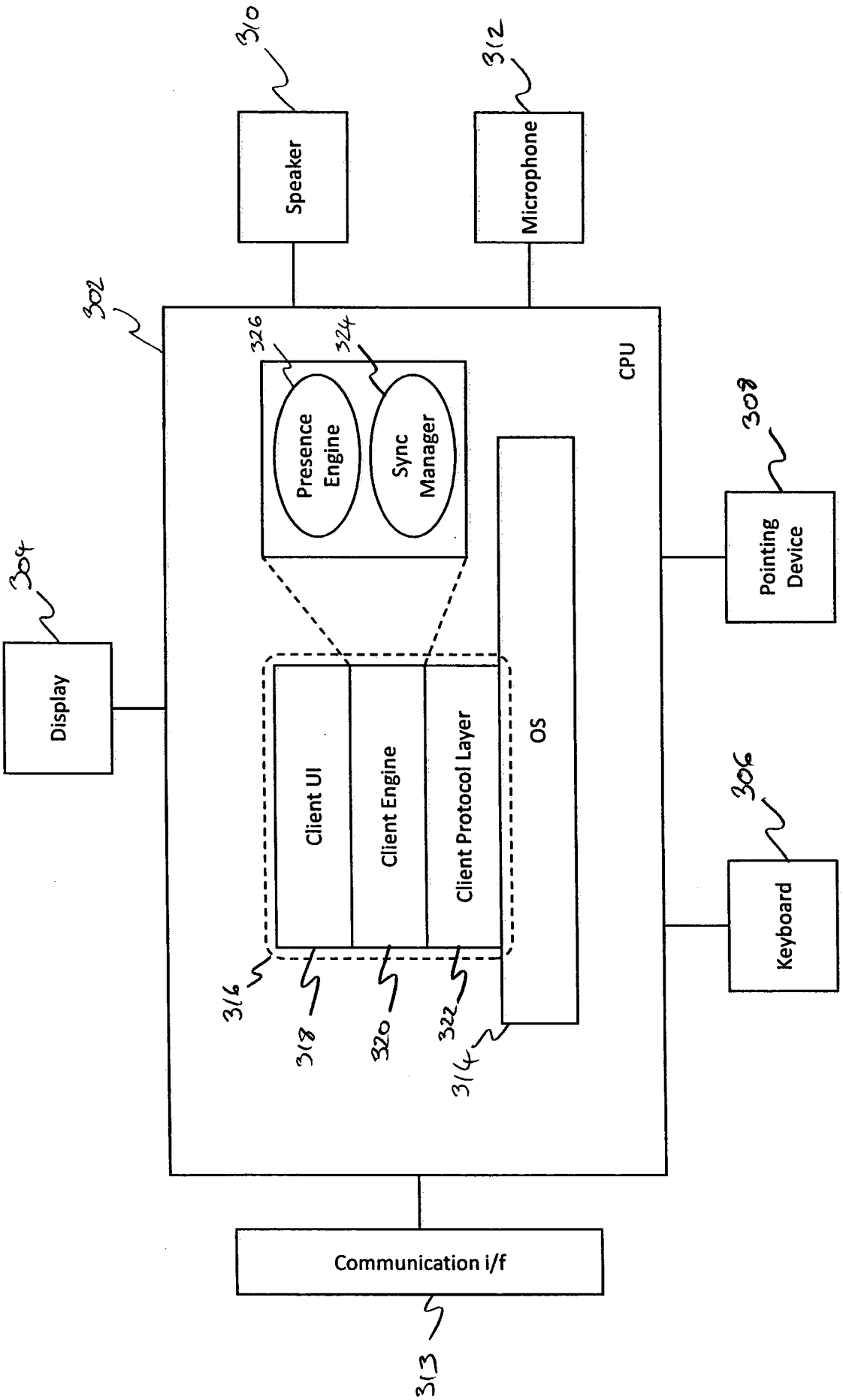
310
Speaker
312
Microphone
302
CPU
326
324
Presence Engine
Sync Manager
308
Pointing Device
304
Display
Client UI
Client Engine
Client Protocol Layer
OS
306
Keyboard
316
318
320
322
314
Communication i/f
313

FIGURE 3

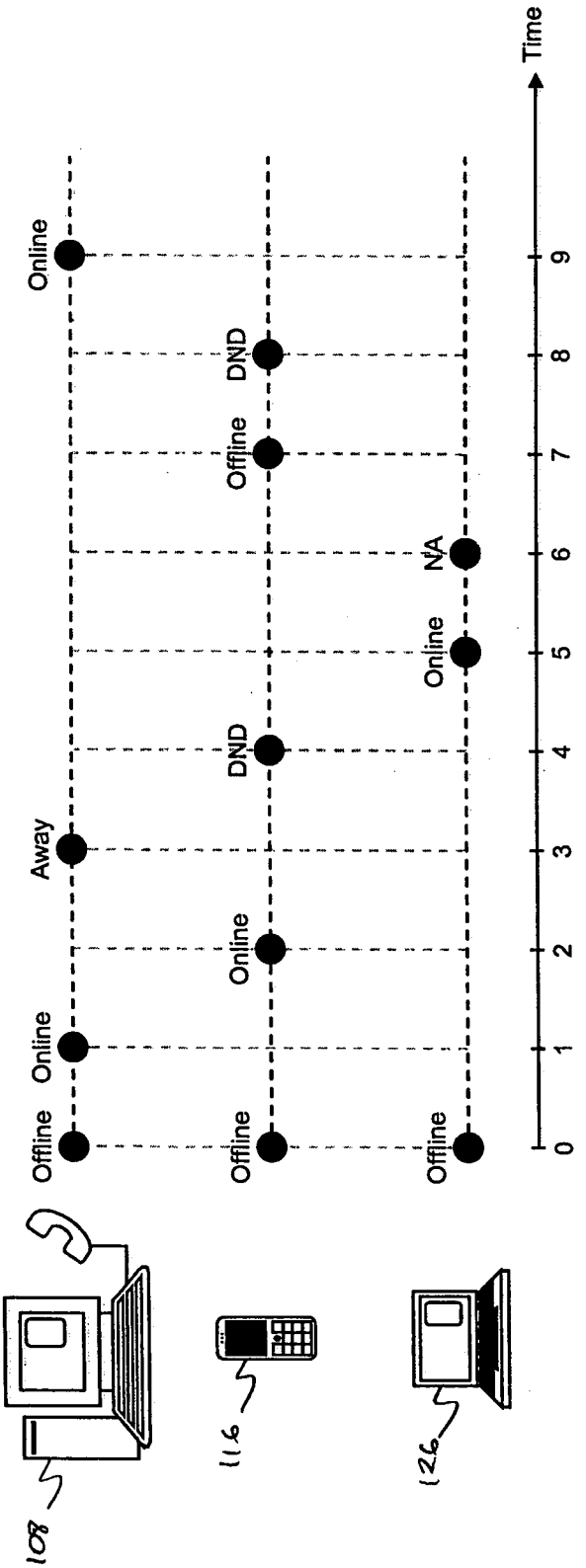
Time
0
1
2
3
4
5
6
7
8
9
Offline
Online
Away
DND
Offline
Online
DND
Offline
NA
Online
Offline
108
116
126

FIGURE 4

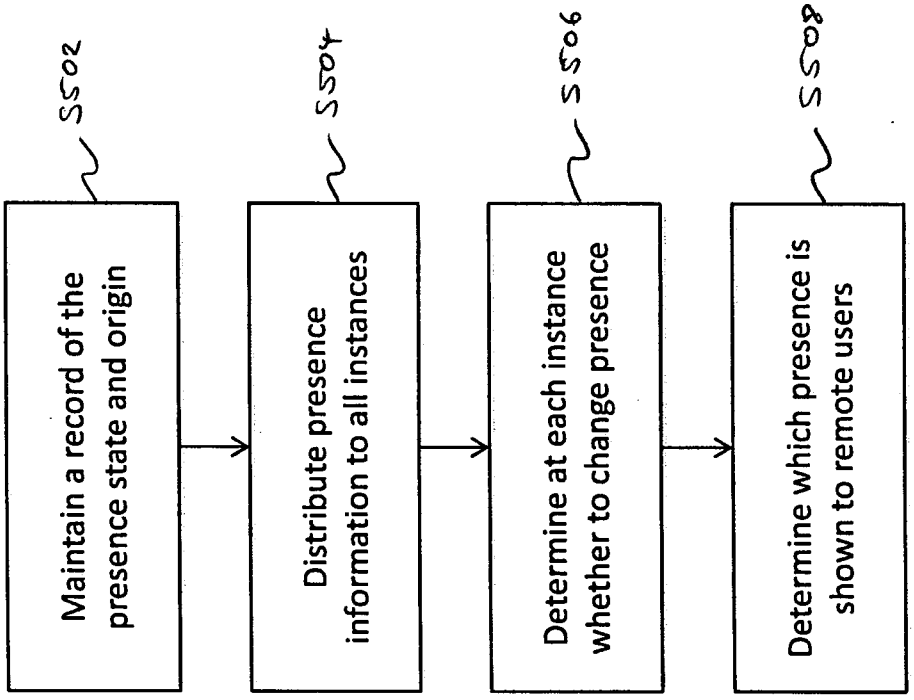
S502
Maintain a record of the presence state and origin
S504
Distribute presence information to all instances
S506
Determine at each instance whether to change presence
S508
Determine which presence is shown to remote users

FIGURE 5

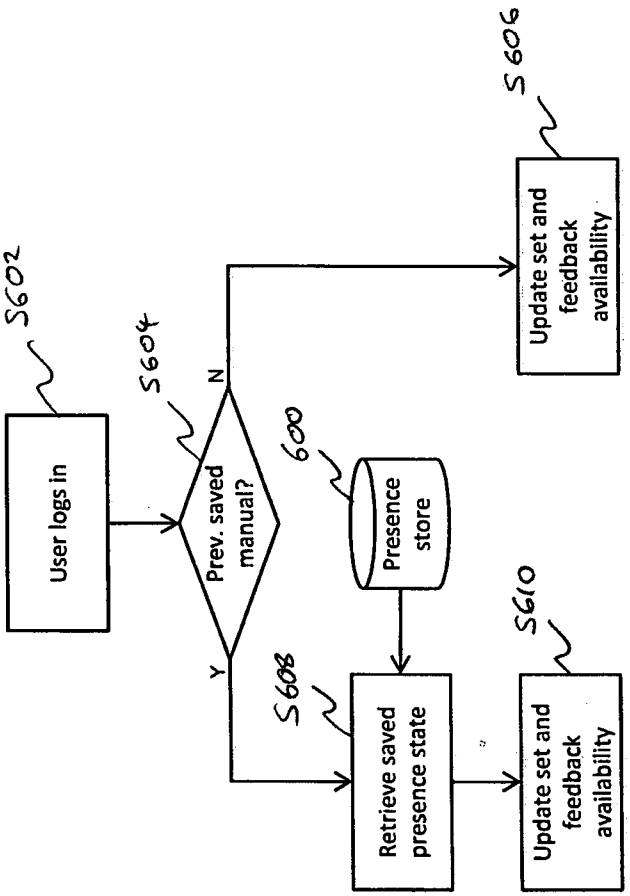
S602
User logs in
S604
Prev. saved manual?
N
Y
S606
Update set and feedback availability
600
Presence store
S608
Retrieve saved presence state
S610
Update set and feedback availability

FIGURE 6A

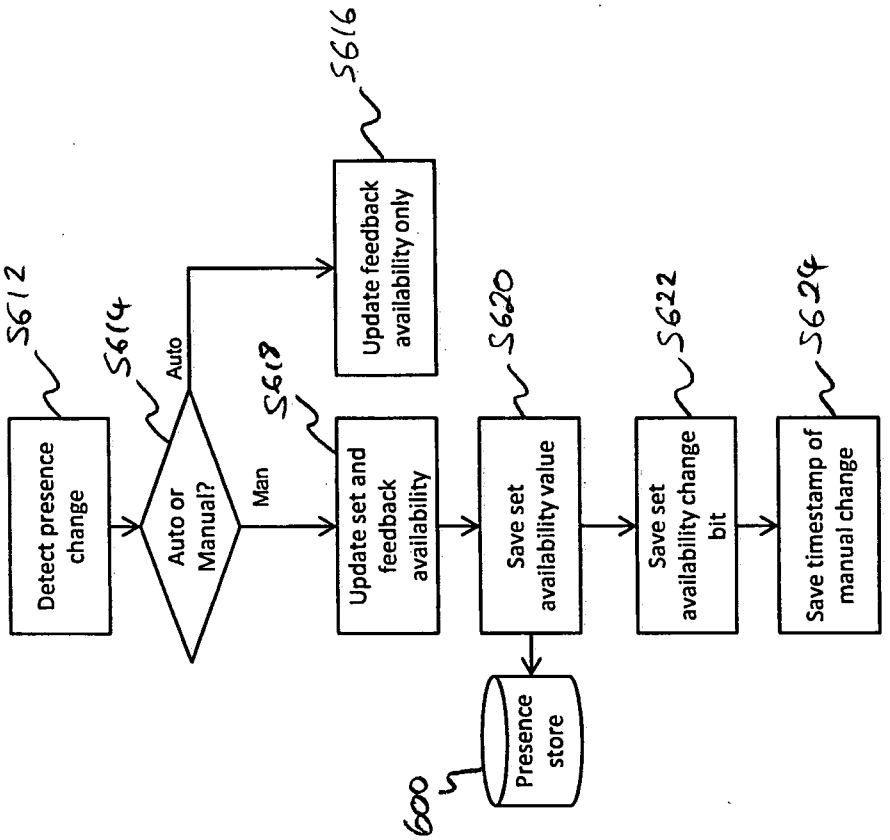
S612
Detect presence change
S614
Auto or Manual?
Auto
Man
S616
Update feedback availability only
S618
Update set and feedback availability
S620
Save set availability value
S622
Save set availability change bit
S624
Save timestamp of manual change
600
Presence store

FIGURE 6B

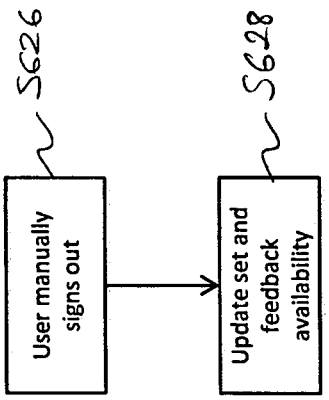
S626
User manually signs out
S628
Update set and feedback availability

FIGURE 6C

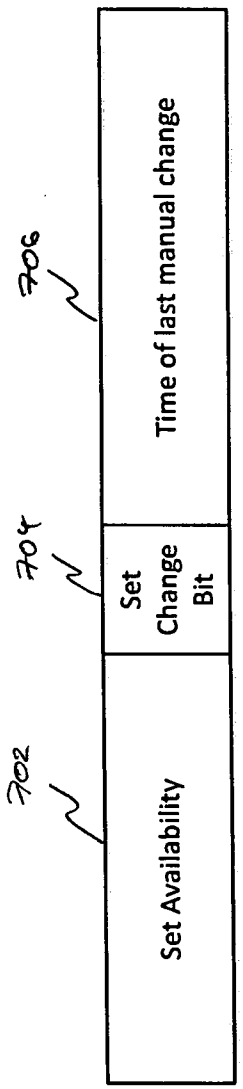
702
Set Availability
704
Set Change Bit
706
Time of last manual change

FIGURE 7A

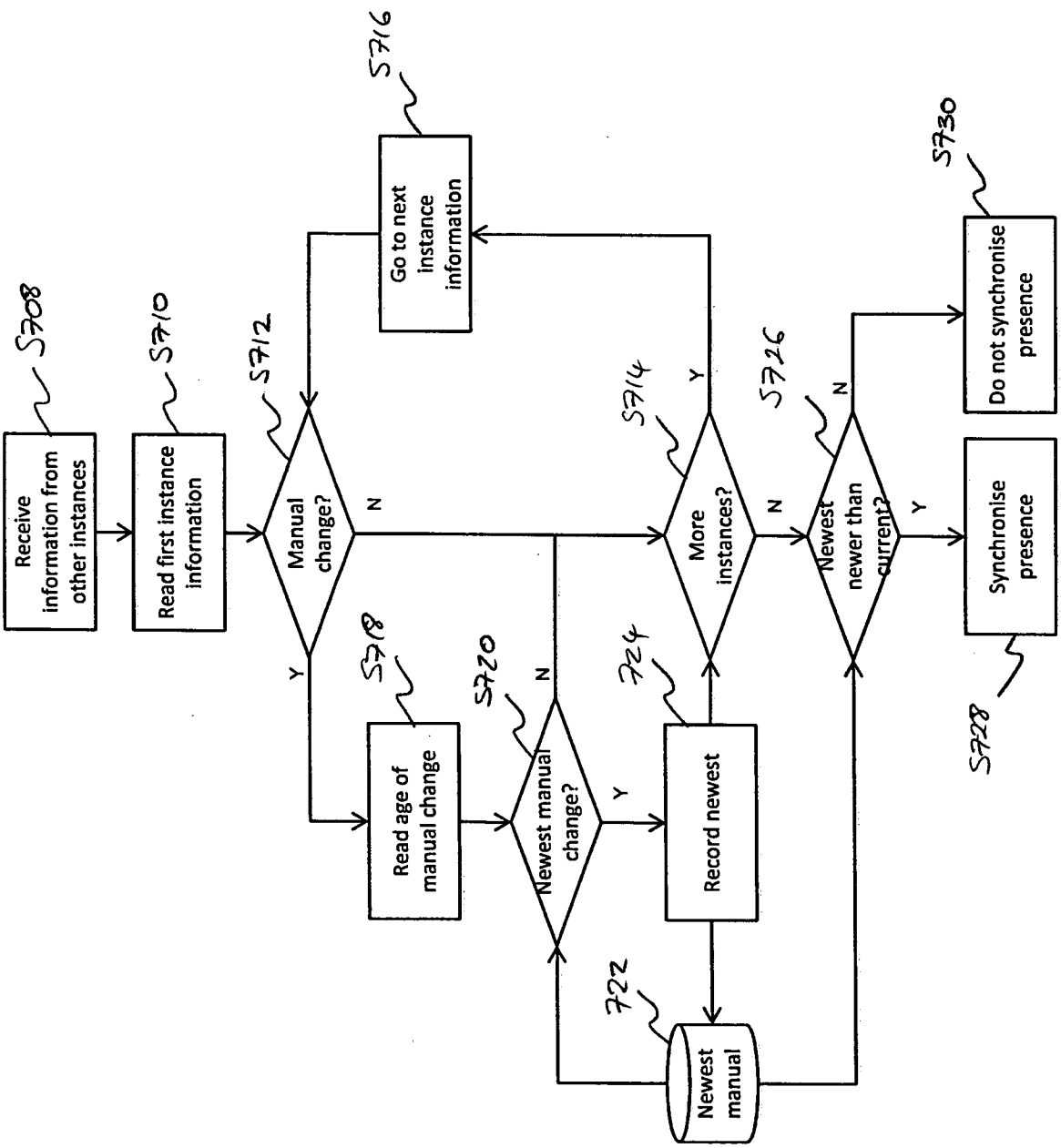
S708
Receive information from other instances
S710
Read first instance information
S712
Manual change?
Y
N
S716
Go to next instance information
S714
More instances?
Y
N
S726
Newest newer than current?
N
Y
S730
Do not synchronise presence
Synchronise presence
S728
S718
Read age of manual change
S720
Newest manual change?
N
Y
724
Record newest
722
Newest manual

FIGURE 7B

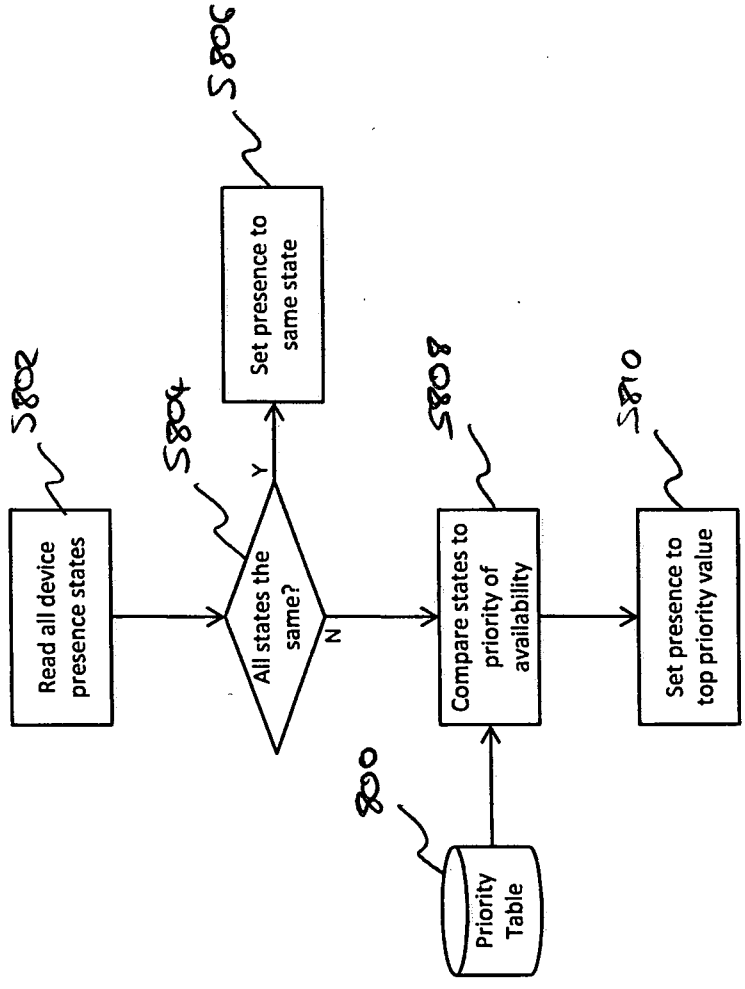
S802
Read all device presence states
S804
All states the same?
Y
N
S806
Set presence to same state
S808
Compare states to priority of availability
S810
Set presence to top priority value
800
Priority Table

FIGURE 8

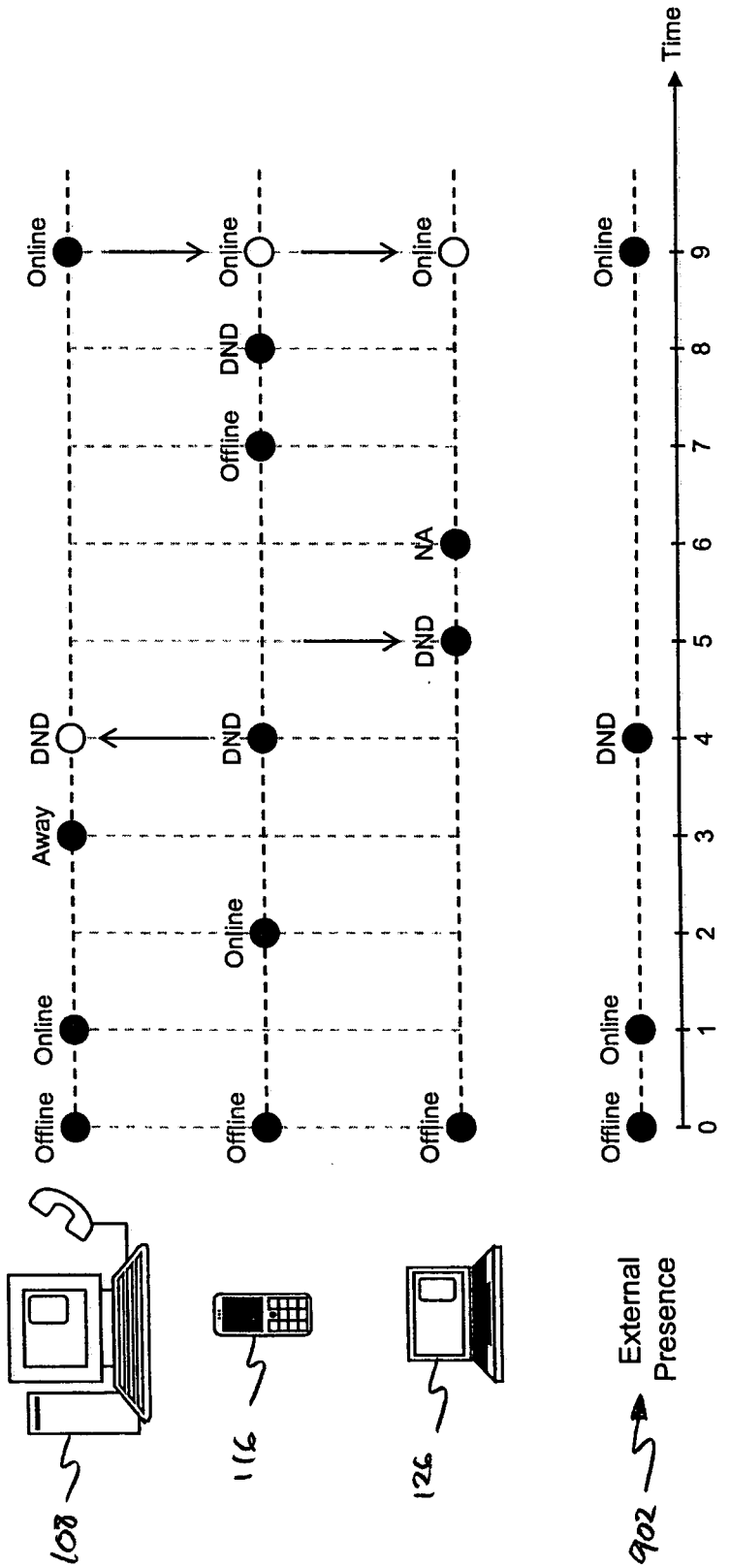
Time
0
1
2
3
4
5
6
7
8
9
Offline
Online
Away
DND
Offline
Online
DND
Offline
Online
DND
Offline
Online
DND
NA
DND
Offline
Online
DND
108
116
126
902
External Presence

FIGURE 9

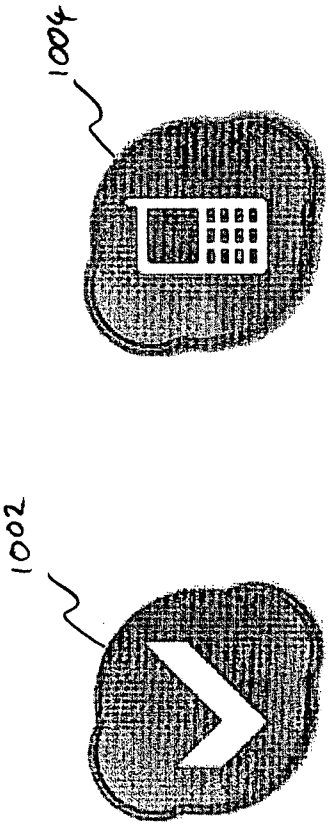

FIGURE 10