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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 November 2006 (02.11.2006)

PCT

(10) International Publication Number
WO 2006/115934 A2

(51) International Patent Classification:
H04J 3/16 (2006.01)

(21) International Application Number:
PCT/US2006/014742

(22) International Filing Date: 20 April 2006 (20.04.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/112,347 22 April 2005 (22.04.2005) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: **WEISBERG, Tomer**; One Microsoft Way, Redmond, WA 98052-6399 (US). **MANION, Todd, R.**; One Microsoft Way, Redmond, WA 98052-6399 (US). **CLASSEN, Andre, R.**; One Microsoft Way, Redmond, WA 98052-6399 (US). **ANIRUDH, Anirudh**; One Microsoft Way, Redmond, WA 98052-6399 (US). **TAO, Kevin, R.**; One Microsoft Way, Redmond, WA 98052-6399 (US). **PARKS, Upshur, Warren, III**; One Microsoft Way, Redmond, WA 98052-6399 (US). **RAO, Ravi**; One Microsoft Way, Redmond, WA 98052-6399 (US). **FLANNERY, Eliot, John**; One Microsoft Way, Redmond, WA 98052-6399 (US). **GUPTA, Rohit**; One Microsoft Way, Redmond, WA 98052-6399 (US). **THALER, David, G.**; One Microsoft Way, Redmond, WA 98052-6399 (US).

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

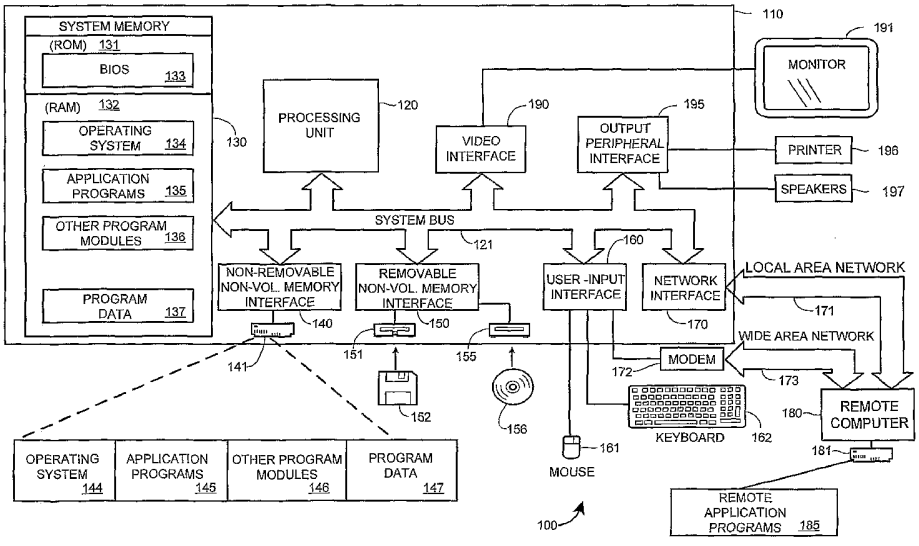
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: AN APPLICATION PROGRAMMING INTERFACE FOR INVITING PARTICIPANTS IN A SERVERLESS PEER TO PEER NETWORK



(57) Abstract: Methods and computer readable mediums are described that facilitate inviting user entities on a network. The method may include initiating a first application for sending an invitation by a first user entity on a first endpoint, selecting a second user entity to receive the invitation, and selecting an activity. The method may also include sending from the first endpoint to the second user entity the invitation to participate in the activity if the first endpoint determines the activity is supported by a computing application on the first endpoint.

WO 2006/115934 A2

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

AN APPLICATION PROGRAMMING INTERFACE FOR INVITING PARTICIPANTS IN A SERVERLESS PEER TO PEER NETWORK

Background

[0001] Server based communication services such as the Messenger service provided by MSN® communication services permit users to sign into a server-based network and then use the services of the network (e.g., e-mail, text messaging, etc.). A server may store a contact list for the user and the user can add and delete persons from the contact list. When the user signs in, a server or servers may notify persons in the user's contact list that the user is "online." Similarly, the server or servers may notify the user of persons in the user's contact list that are "online."

[0002] The MICROSOFT® Corporation also provides Peer-to-Peer Networking software for use with its WINDOWS® operating systems. With this system, users can create a network of peer computers and can communicate with one another without having to sign into a central server. For example, users can create a peer-to-peer group and then create a chat room in which all members of the group can post messages and see messages posted by others in the group. The system may also allow peers to invite others to participate in collaborative activities. The chat room is maintained using the peer computers and without the need for a central server.

Summary

[0003] Methods and computer readable mediums are described that facilitate inviting user entities on a network. The method may include initiating a first application for sending an invitation by a first user entity on a first endpoint, selecting a second user entity to receive the invitation, and selecting an activity. The method may also include sending from the first endpoint to the second user entity the invitation to participate in the activity if the first endpoint determines the activity is supported by a computing application on the first endpoint.

[0004] Additionally, the method may include that the first application for sending the invitation is the computing application, that the first application for sending the invitation is a messenger application, and initiating a name resolution mechanism to locate a second endpoint for the second user entity. The method may further include querying a presence store to locate a second endpoint for the second user entity, sending the invitation

synchronously or asynchronously, canceling an asynchronous invitation at the first endpoint, and sending the invitation in a secure or unsecure manner.

[0005] Furthermore, methods are described that facilitate receiving invitations on a network. The method may include initiating, at a receiving endpoint, a first application for receiving an invitation from a sending endpoint to participate in an activity, receiving the invitation at the receiving endpoint, and launching a computing application on the receiving endpoint to execute the activity if the receiving endpoint determines the activity is capable of being executed by the computing application at the receiving endpoint.

[0006] Additionally, the method may include the first application for receiving the invitation is the computing application, the first application for receiving the invitation is a messenger application, and the receiving endpoint receives the invitation by at least one of a generic invitation listener application, a messenger application, or the computing application at the receiving endpoint. The method may further include registering activity capabilities of the computing application at the receiving endpoint with a presence store, receiving the invitation synchronously or asynchronously, receiving the invitation in a secure or unsecure manner, and retrieving, by the sending endpoint, status information of the invitation sent to the receiving endpoint.

Drawings

[0007] Fig. 1 is a block diagram of a computing system that may operate in accordance with the claims;

[0008] Fig. 2 is a block diagram of an exemplary system that may facilitate peer-to-peer, serverless collaboration and/or communications;

[0009] Fig. 3 is a flow diagram of an exemplary method related to sending an invitation to a contact that may facilitate collaborative activities; and

[0010] Fig. 4 is a flow diagram of an exemplary method related to receiving an invitation that may facilitate collaborative activities.

Description

[0011] Although the following text sets forth a detailed description of numerous different embodiments, it should be understood that the legal scope of the description is defined by the words of the claims set forth at the end of this patent. The detailed description is to be construed as exemplary only and does not describe every possible embodiment since describing every possible embodiment would be impractical, if not impossible. Numerous alternative embodiments could be implemented, using either current technology or technology developed after the filing date of this patent, which would still fall within the scope of the claims.

[0012] It should also be understood that, unless a term is expressly defined in this patent using the sentence "As used herein, the term '_____' is hereby defined to mean..." or a similar sentence, there is no intent to limit the meaning of that term, either expressly or by implication, beyond its plain or ordinary meaning, and such term should not be interpreted to be limited in scope based on any statement made in any section of this patent (other than the language of the claims). To the extent that any term recited in the claims at the end of this patent is referred to in this patent in a manner consistent with a single meaning, that is done for sake of clarity only so as to not confuse the reader, and it is not intended that such claim term by limited, by implication or otherwise, to that single meaning. Finally, unless a claim element is defined by reciting the word "means" and a function without the recital of any structure, it is not intended that the scope of any claim element be interpreted based on the application of 35 U.S.C. § 112, sixth paragraph.

[0013] Fig. 1 illustrates an example of a suitable computing system environment 100 on which a system for the steps of the claimed method and apparatus may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the method of apparatus of the claims. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0014] The steps of the claimed method and apparatus are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the methods or apparatus of the claims include, but are not limited to,

personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0015] The steps of the claimed method and apparatus may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The methods and apparatus may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0016] With reference to Fig. 1, an exemplary system for implementing the steps of the claimed method and apparatus includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0017] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage

or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0018] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Fig. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0019] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 1 illustrates a hard disk drive 140 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0020] The drives and their associated computer storage media discussed above and illustrated in Fig. 1, provide storage of computer readable instructions, data structures,

program modules and other data for the computer 110. In Fig. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 190.

[0021] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0022] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the

remote memory storage device. By way of example, and not limitation, Fig. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0023] Figure 2 is a block diagram of an example system 200 that may be used to implement example methods described herein. The system 200 may facilitate peer-to-peer, serverless collaboration and/or communications via a communication network 202, and may be implemented using a computing system such as the computing system 100 of Figure 1. The communication network may include, but is not limited to, a LAN and/or a WAN, or a communication medium supporting socket communications, or named-pipes, for example.

[0024] The system 200 may include a presence system 204 that monitors the presence of other entities on the communication network. An entity may be, for example, a particular person, a device (e.g., a printer, a copier, a computer, a scanner, etc.) a position in an organization (e.g., "manager," "customer service representative," etc.), an organization, etc. Presence on a network may generally refer to a current status of an entity with regard to their willingness or ability to communicate via the network, but may also refer to additional or alternative information regarding the entity such as a current activity of the entity. Presence on a network may be represented by presence information. Examples of presence information may include, but are not limited to, one or more of an indication that an entity is "online," an indication that an entity is "out to lunch," an indication that an entity is "away," an indication that an entity will "be right back," an indication that an entity is "idle," an indication that an entity is "busy," an indication that an entity is "on the phone," an indication that an entity is "watching a movie," an indication that an entity is "playing Halo®," an indication that an entity is "helping another customer," an indication of a device status (e.g., a printer with status "busy," "idle," etc). Presence information may include one or more enumerated strings and/or rich presence (e.g., custom strings generated by a user entity). For example, a user entity could define a custom presence state as, "I am out of the office. Will return tomorrow." Presence information obtained by the presence system 204 may be stored in a presence store 208.

[0025] The presence system 204 may facilitate a user entity to monitor (or "subscribe") to presence information of other entities. This may include the presence system 204 polling other computing systems periodically, for example. Additionally or alternatively, other computing systems corresponding to other user entities may transmit event indications to the

system 200 that notify the presence system 204 of events such as a change in presence state. For example, an event may occur when a user's presence changes from "offline" to "online," and the presence system 204 may detect this event. The presence system 204 could then notify other applications or software modules (e.g., such as the application 280) that the event occurred.

[0026] The presence system 204 may also monitor capabilities of other entities published on the network 202. Capabilities of an entity may include, for example, static capabilities such as whether a computing system of the entity is configured to execute a particular software application, whether a computing system of the entity has a particular hardware device, etc. Capabilities of an entity may also include, for example, dynamic capabilities such as real-time capabilities of an entity with respect to a game software application currently being executed on the entity's computing system, etc. An entity publishing capabilities on the network may refer to permitting other entities to be able to monitor the capabilities via the network 202. Capability information obtained by the presence system 204 may be stored in a capability store 212.

[0027] The presence system 204 may also monitor objects of other entities published on the network 202. Objects of an entity may include, for example, data objects such as files, structures, pictures, sounds, a description such as meta-data, a name-value pair, etc. An entity publishing objects on the network may refer to permitting other entities to be able to monitor the objects via the network 202. As just one example, publishing an object may permit an entity to provide other entities with information specific to an application being executed by a computing system of the entity and/or real-time information. With respect to a game application, for instance, a published object could include information regarding a player's current score, a weapon currently in possession of the player, etc. Objects information obtained by the presence system 204 may be stored in an objects store 216.

[0028] The presence system 204 may also provide (or "publish") presence information associated with a user entity (i.e., the entity associated with the system 200) to other entities on the network 202. The presence information associated with the user entity may be stored in the presence store 208 or some other storage. Similarly, the presence system 204 may also provide (or "publish") information regarding capabilities of the user entity to other entities on the network 202. The capability information associated with the user entity may be stored in a capability store 208. Further, the presence system 204 may also provide (or "publish") information regarding objects of the user entity to other entities on the network. The object

information associated with the user entity may be stored in an object store 216, or some other storage. Similarly, the presence system 204 may facilitate an ability for the user entity to monitor (or "subscribe-to") presence information. As such, when presence information is monitored by the user entity, subsequent changes may elicit events for which the user entity is notified. For example, if the monitored presence information changes from "away" to "playing Halo®," an event may trigger thereby notifying the user entity of a change.

[0029] The presence system 204 may interface with a contact store 240 that stores information regarding other entities. The contact store 240 may store information for an entity such as one or more of a secure identifier, a human readable alias, an indicator of whether presence information for this entity is to be monitored, and an indicator of whether to allow this entity to obtain presence information regarding the user entity. An entity as represented in the contact store 240 may be referred to as a contact.

[0030] Each user entity may have one or more communication endpoints with which it is associated. Generally, different communication endpoints associated with a user entity may include different communication termination points associated with the entity, such as different computing systems. As an example, endpoints for a particular entity may include a desktop computer at work, a desktop computer at home, a personal digital assistant (PDA), etc. Optionally, different communication endpoints associated with a user entity may also include different software applications being executed by a single computing system. Endpoint information may include a peer name, a machine name, or a device type, to name a few.

[0031] The presence system 204 may also interface with a communication system 260, which is coupled to the communication network 202. The communication system 260 may establish connections between the system 200 and other peer computing systems associated with other entities. Establishing a connection may include, for example, one or more of determining an endpoint associated with an entity, resolving an address of the endpoint, authenticating communications, encrypting and decrypting communications, etc. In one implementation, the communication system 260 may include a Peer Name Resolution Protocol (PNRP), or similar. The PNRP may resolve a name (e.g., a peer name) for a contact to derive its IP address without reliance upon a Domain Name System (DNS), commonly used by server computers. In another implementation, the communication system 260 may interface with an authentication system 270 that is itself coupled to the contact store 240. In attempting to establish a connection with another computing system, the communication

system 260 may receive from the other computing system an indication of an identifier associated with an entity. The authentication system 270 may then check whether information about the entity with the security identifier presented is stored in the contact store 240. If the identifier is not found in the contact store 240, the connection may be refused.

[0032] A connection may be secured. Establishing a connection and communicating over a connection may include, for example, one or more of utilizing a secure channel, utilizing secure socket layer (SSL) techniques, utilizing transport layer security (TLS) techniques, utilizing public/private key pairs, utilizing authentication techniques (e.g., X.509 certificates, encrypted signatures utilizing a pretty good privacy (PGP) program, etc.), utilizing a peer name resolution protocol (PNRP), transmission control protocol (TCP), internet protocol (IP), internet protocol version six (IPv6), etc. Resolving an address of an endpoint may include, for example, resolving a PNRP identifier to an IP address and a port.

[0033] A software application 280, or some other software module, may communicate with the presence system 204 to obtain presence information, capabilities information, and/or objects information associated with other user entities on the communication network 202. For example, the presence system 204 may provide a set of application programming interfaces (APIs) that permit software applications to request and receive information regarding presence, capabilities, and/or objects associated with other user entities. The presence system 204 may retrieve the requested information from the presence store 208, capabilities store 212, and/or the objects store 216. Additionally or alternatively, the presence system 204 could obtain requested information from the other user entities via the communication system 260 and the communication network 202. Generally speaking, "availability" may refer to presence information, capabilities and objects. A user entity has the ability to publish all, some, or none of this information.

[0034] An Invitation API 282 may permit an application 280 used by a user entity to invite other contacts or user entities of a serverless network to participate in a collaborative activity. The Invitation API 282 may employ the use of the Presence System 204 to determine contacts from the contact store 240. Additionally, the Presence System 204 may retrieve capability information located in the capability store 212. If a particular contact in the contact store 240 has no associated capability information, the Invitation API 282 or, alternatively, the Application 280 may employ a Capabilities API 284 to determine the capabilities of another user entity's endpoint. A capability may be a collaborative activity including any activity which is supported by an application on multiple endpoints. For

example, a contact may support the activity of Halo® game play because the contact's endpoint has the Halo® gaming application installed. The Invitation API may allow the contact owner to determine other contacts on a serverless network that may support the activity of Halo® game-play.

[0035] When the Invitation API locates a contact that supports the activity, an invitation may be sent by the user entity (invitor) to that contact (invitee), thereby allowing the invitee to respond to the invitation. A response may include standard invitee options of "accept," "reject," or "ignore." Other response options may include standard or custom text such as "busy now, please invite me later," or "please try again after 9 pm." However, if the invitee is not present at the endpoint, or simply chooses not to acknowledge the invitation with "accept," "reject," or "ignore," then the invitation may time-out. Upon expiration of a time-out period, the invitation may disappear from view at the invitee's endpoint, thereby minimizing clutter on the invitee's desktop. Optionally, an invitee may configure alert settings so that, for example, invitation dialog boxes do not interrupt an invitee while giving a presentation. As such, any incoming invitations may automatically receive an indication that the invitee is unavailable. The invitee may further customize such indications, for example, "Presentation ongoing, please try after 4 pm."

[0036] Figure 3 may be an illustration of a method 300 in accordance with the claims. The method may begin at either block 302 or block 350, depending on whether the application 280 used by the user entity is a messenger application (starting at block 302) or a computing application (starting at block 350). Assuming the user entity at an endpoint wishes to send an invitation to a contact for collaborative participation in an activity, but no associated computing application for the activity is currently running, then the user entity may begin the invitation process at block 302. The user entity may access the messenger application 280 at block 304 which may provide the user entity with a list of contacts from the contact store 240. The messenger application 280 may also access the presence system 204 to obtain presence information from the presence store 208 and provide the user entity with presence information for the contacts for which the user has selected. At block 306 the user entity may select a contact and at block 308 the user entity may select a desired activity. The contact selected is not limited to other user entities, but may include one or more of the invitor contact's alternate points of presence. The user may select activities including, but not limited to, playing Halo®, internet telephony, or a collaborative whiteboard. Despite a user entity selecting a particular activity, the Capabilities API 284 may allow the user entity to

verify endpoint capabilities at block 310. If the user entity's endpoint is a computer without a telephony application installed, or if the computer has no sound card, the method may proceed to block 312 and terminate 395. Additionally, or alternatively, the method may send a message to the contact selected at block 306 to notify that contact of the failed invitation attempt, and possible reasons for that failure.

[0037] Alternatively, if the user entity's endpoint supports the activity, the method may determine if the messenger, or some other application, sends the invitation at block 314. The method may include a default setting in which the messenger always performs the invitation, or the particular application supporting the activity performs the invitation. The method may also prompt the user entity to choose one of the aforementioned options. If the messenger performs the invitation, then, via the Invite API's, the method may launch the corresponding application which supports the activity at block 316. On the other hand, the application supporting the activity may perform the invitation by calling the Invite API 282 which may also set environment variables to the application and launch such application at block 318. The Invite API 282 may also permit the invitation to hold application specific data and/or pass parameters to the application. Such data and parameters may be useful for handshaking of remote applications. Environment variables may include a list of invitees, or other information pertaining to the activity, the application supporting the activity, and the contacts invited to participate in the activity. Once those variables are set at block 318 they may be read by the application 280 at block 320. The application may then use a function call from the Invitation API 282 at block 322 to send the invitation. If, however, a contact has multiple points of presence, the Invitation API 282 may send the invitation to all endpoints for which the invitee is associated.

[0038] Notice that, in the alternative, if the computing application 352 is to perform the invitation in lieu of the messenger, control passes from the computing application 352 to block 354 where the user entity may select a contact with which to perform a collaborative activity.

[0039] An invitation function call from the Invitation API 282 may require sending parameters derived from the calling application, whether the calling application is the messenger or the application supporting the activity. For example, the sending parameters may include the receiving name or names, the receiving contact endpoint (e.g., an IP address and port), activity information, and a function call handle, to name a few. The Communication System 260 may determine whether a particular endpoint for a contact name

is known at block 324. If unknown, block 326 may use the Communication System 260 to determine a corresponding IP address and port for the contact, and then establish a connection with that endpoint at block 328. The Communication System 260 may employ PNRP, or similar, to resolve such endpoint credentials without a server. The activity information may also include a globally unique identifier (GUID) to identify an appropriate application to the invitee (recipient) that may support the desired activity. The activity information may also include an invitation message (e.g., "Let's talk about the TPS report now.") and other information to support various requirements of any particular activity. The handle may allow the function call to receive event information after the invitation is sent to an invitee. Such a handle is particularly useful for asynchronous function calls, however the function calls employed by the Invitation API 282 may also be synchronous, in which case a response is promptly received, or the function call times out. Additionally, another handle may identify the invitation instance so that a cancellation function call may correctly identify a specific invitation to cancel. Generally speaking, upon the user making the function call to send an invitation, an invitation session is created having a corresponding GUID. This GUID may be used to identify responses received as a result of this particular invitation. The Invitation API may serialize the invitation in XML (or other) format before sending to the invitee via the Communication System 260.

[0040] Figure 4 may be an illustration of a method 400 in accordance with the claims. The exemplary method for receiving an invitation may include one of three starting blocks 402, 426, or 456. A starting block is determined by virtue of an invitee's endpoint status as either running an application 280 of type messenger (block 402), running an application 280 supporting the invitation activity (block 426), or not running either of these (block 456). In any of these three cases, the method in Figure 4 uses function calls from the Invitation API 282 to receive the invitation. Optionally, an invitee may choose to receive invitations from a specific subset of contacts, all contacts, or no contacts at all. In the case of an invitee running a generic user interface or application invitation listener, upon receipt of an invitation at block 404, the Invitation API 282 determines if the endpoint supports the requested activity 406 through a query to the capability store 212. The Capabilities API 284 may make such a determination by checking for a matching GUID in the capabilities store 212 shown in Figure 2. If not supported (e.g., no appropriate matching GUID found), the Invitation API 282 of the listener sends the calling function (invitor) a decline message at block 408. Alternatively, the Invitation API 282 may permit the invitor, upon determining that the requested activity is

not supported by the invitee endpoint, to provide information regarding where the invitee may obtain an application which may support that activity. For example, if the invitor learns that the invitee does not support playing a game of chess, the invitation may include a URL, or other useful information, to allow the invitee to acquire or install an appropriate application supporting the activity. The invitee may choose to install the appropriate application, which may further have a GUID. Protective measures may further guard against newer applications overwriting GUID's of existing applications.

[0041] However, if the activity is supported by the listener endpoint, then the invitee may acknowledge the invitation, perhaps via a dialog box, by accepting, rejecting, or ignoring 410. Again, a decline message may be sent at block 408 in which the message may further describe potential reasons for the decline (e.g., activity is not supported at this endpoint, user did not allow the activity, etc.). The response may also be sent automatically by the invitee through a default setting (e.g., always reject invitations from Bob to play Halo®). For example, the invitee may accept all invitations by default, reject all invitations by default, or ignore all invitations by default. Additionally, the invitee may have a default setting of accept, reject or ignore based on the specific identity of the inviter. For example, the invitee may choose to accept all invitations by default if the inviter is a buddy, reject all invitations by default if the inviter is unknown (not in the invitee's contact store), or ignore an invitation by default if the inviter is an in-law. Similar to block 318 in Figure 3, environment variables are set and the application supporting the activity is launched at block 412. The application reads the environment variables at block 414, sends back acceptance notification 416 and starts its collaborative activity at block 418. Additionally, or alternatively, the Capabilities API 284 may be called at block 414 to obtain relevant information, discussed in further detail below.

[0042] If the invitee's endpoint is running the messenger application, then the messenger application may already have appropriate environment variables. As such, blocks 426 through 428 accomplish receipt of the invitation and initiation of the collaborative activity. Similarly, blocks 456 through 466 may execute if the invitee already happens to have the application running that supports the collaborative activity. In such a case, the method does not need to query the invitee's endpoint for the application supporting the collaborative activity because it is already running, but the invitee may still choose to accept, reject, or ignore at block 460.

[0043] Returning briefly to block 310 of Figure 3 and blocks 406 and 430 of Figure 4, the Capabilities API 284 determines whether the endpoint supports a requested activity. Before the Invitation API 282 may launch an activity, or a specific application that supports the activity, the capabilities supported at that endpoint must be known. As such, the Capabilities API 284 may include functions for registration, unregistration, enumeration of capabilities, and retrieval of information regarding the capabilities. A peer registration function may allow a user entity to add an entry to an operating system registry containing information relating to the capabilities supported by a particular application. Additionally, this information may include a GUID to identify the application, application configuration, and a description of the capability. The peer registration function may further allow the user entity to specify which other user entities or contacts may access the registered application. Alternatively, the user entity may unregister capabilities with an unregister function which removes the capability information from the registry.

[0044] A user entity may call, via the application 280, a capability query function to learn about application capabilities when the GUID is known. However, the user entity may first call a capability enumeration function to obtain a list of all capabilities currently registered at an endpoint, including the endpoint of the user entity. The user entity may then select one capability from the enumerated list and call the capability query function for more specific capability information supported by that particular application.

[0045] The following exemplary scenario illustrates Invitation API functionality. A user entity named Toby has friends named Bob and Alice in his contacts list. Bob and Alice have each allowed publication of their presence information, and Toby notices that Bob is online and has the game Halo ® installed. Toby selects Bob from his contact list and sends an invitation to Bob containing a request to play Halo®. The Invitation API 282 determines that no endpoint connection is currently available and, consequently, employs the Communication Module 260 which may use PNRP to resolve an endpoint from Bob's contact information in Toby's contact store. When such endpoint is determined, the Communication Module 260 may optionally authenticate a connection via the Authentication System 270 between Bob and Toby and then send the invitation. Assuming Bob has his Messenger application running, it receives the invitation and the Capabilities API 284 is called to verify that Halo® is supported on Bob's computer. If it is supported, the Invitation API 282 presents Toby with an option (via a pop-up user interface) to "accept," "reject," or "ignore" the invitation. Bob

accepts, the Halo ® application is launched, and an "accept" message is returned to the invitor (Toby).

[0046] Although the forgoing text sets forth a detailed description of numerous different embodiments, it should be understood that the scope of the patent is defined by the words of the claims set forth at the end of this patent. The detailed description is to be construed as exemplary only and does not describe every possible embodiment because describing every possible embodiment would be impractical, if not impossible. Numerous alternative embodiments could be implemented, using either current technology or technology developed after the filing date of this patent, which would still fall within the scope of the claims.

[0047] Thus, many modifications and variations may be made in the techniques and structures described and illustrated herein without departing from the spirit and scope of the present claims. Accordingly, it should be understood that the methods and apparatus described herein are illustrative only and are not limiting upon the scope of the claims.

CLAIMS:

1. A method of inviting user entities on a network, comprising:

initiating a first application for sending an invitation by a first user entity on a first endpoint;

selecting a second user entity to receive the invitation;

selecting an activity;

sending from the first endpoint to the second user entity the invitation to participate in the activity if the first endpoint determines the activity is supported by a computing application on the first endpoint.

2. The method of claim 1 wherein the first application for sending the invitation is the computing application.

3. The method of claim 1 wherein the first application for sending the invitation is a messenger application.

4. The method of claim 1 further comprising initiating a name resolution mechanism to locate a second endpoint for the second user entity.

5. The method of claim 1 further comprising querying a presence store to locate a second endpoint for the second user entity.

6. The method of claim 1, further comprising sending the invitation synchronously or asynchronously.

7. The method of claim 6, further comprising canceling an asynchronous invitation at the first endpoint.

8. The method of claim 1, further comprising sending the invitation in a secure or unsecure manner.

9. A method of invitation receipt by user entities on a network, comprising:

initiating, at a receiving endpoint, a first application for receiving an invitation from a sending endpoint to participate in an activity;

receiving the invitation at the receiving endpoint;

launching a computing application on the receiving endpoint to execute the activity if the receiving endpoint determines the activity is capable of being executed by the computing application at the receiving endpoint.

10. The method of claim 9 wherein the first application for receiving the invitation is the computing application.

11. The method of claim 9 wherein the first application for receiving the invitation is a messenger application.

12. The method of claim 9 wherein the receiving endpoint receives the invitation by at least one of a generic invitation listener application, a messenger application, or the computing application at the receiving endpoint.

13. The method of claim 9 further comprising registering activity capabilities of the computing application at the receiving endpoint with a presence store.

14. The method of claim 9, further comprising receiving the invitation synchronously or asynchronously.

15. The method of claim 9, further comprising receiving the invitation in a secure or unsecure manner.

16. The method of claim 9, further comprising retrieving, by the sending endpoint, status information of the invitation sent to the receiving endpoint.

17. A computer readable medium having computer executable instructions for inviting user entities on a network, comprising:

computer executable instructions for initiating a first application for sending an invitation by a first user entity on a first endpoint;

computer executable instructions for selecting a second user entity to receive the invitation;

computer executable instructions for selecting an activity; and

computer executable instructions for sending from the first endpoint to the second user entity the invitation to participate in the activity if the first endpoint determines the activity is supported by a computing application on the first endpoint.

18. The computer readable medium of claim 17, wherein the first application for sending the invitation is the computing application.

19. The computer readable medium of claim 17, wherein the first application for sending the invitation is a messenger application.

20. The computer readable medium of claim 17, further comprising initiating a name resolution mechanism to locate a second endpoint for the second user entity.

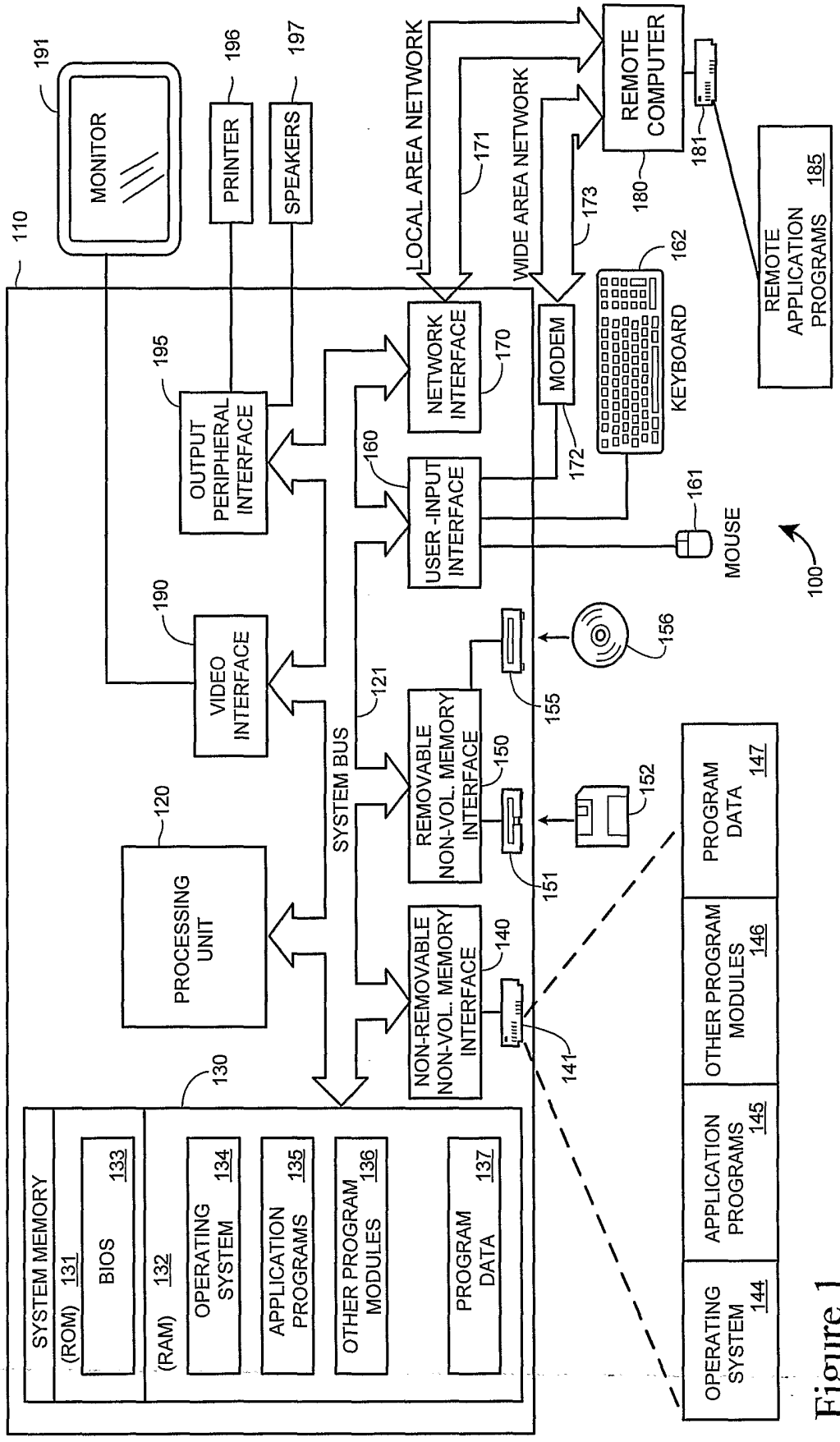

100
110
120
121
130
131
132
133
134
135
136
137
140
141
144
145
146
147
150
151
152
155
156
160
161
162
170
171
172
173
180
181
185
190
191
195
196
197
SYSTEM MEMORY
(ROM)
BIOS
(RAM)
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA
PROCESSING UNIT
SYSTEM BUS
VIDEO INTERFACE
OUTPUT PERIPHERAL INTERFACE
NON-REMOVABLE NON-VOL. MEMORY INTERFACE
REMOVABLE NON-VOL. MEMORY INTERFACE
USER -INPUT INTERFACE
NETWORK INTERFACE
MODEM
LOCAL AREA NETWORK
WIDE AREA NETWORK
REMOTE COMPUTER
REMOTE APPLICATION PROGRAMS
MONITOR
PRINTER
SPEAKERS
KEYBOARD
MOUSE
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA

Figure 1

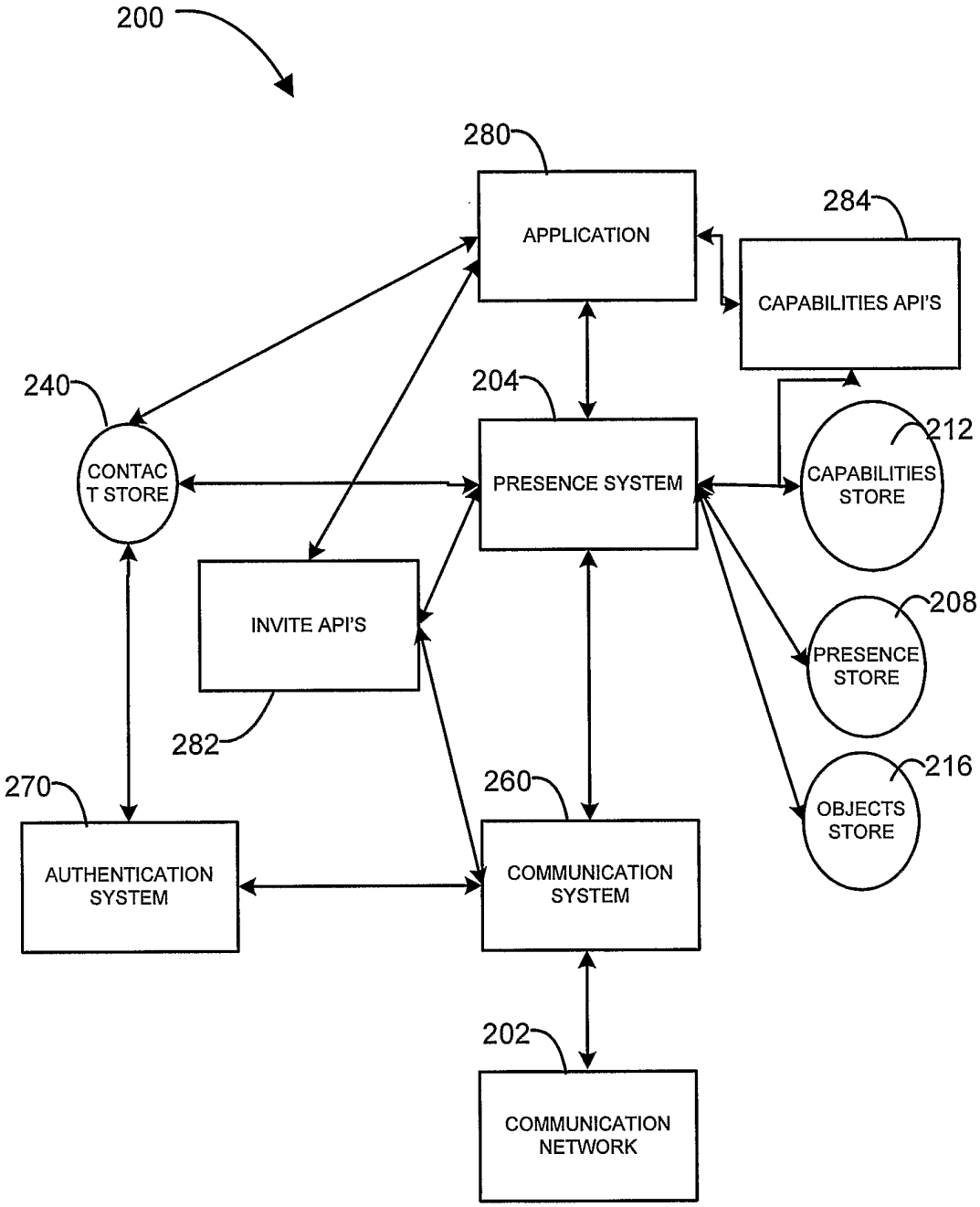
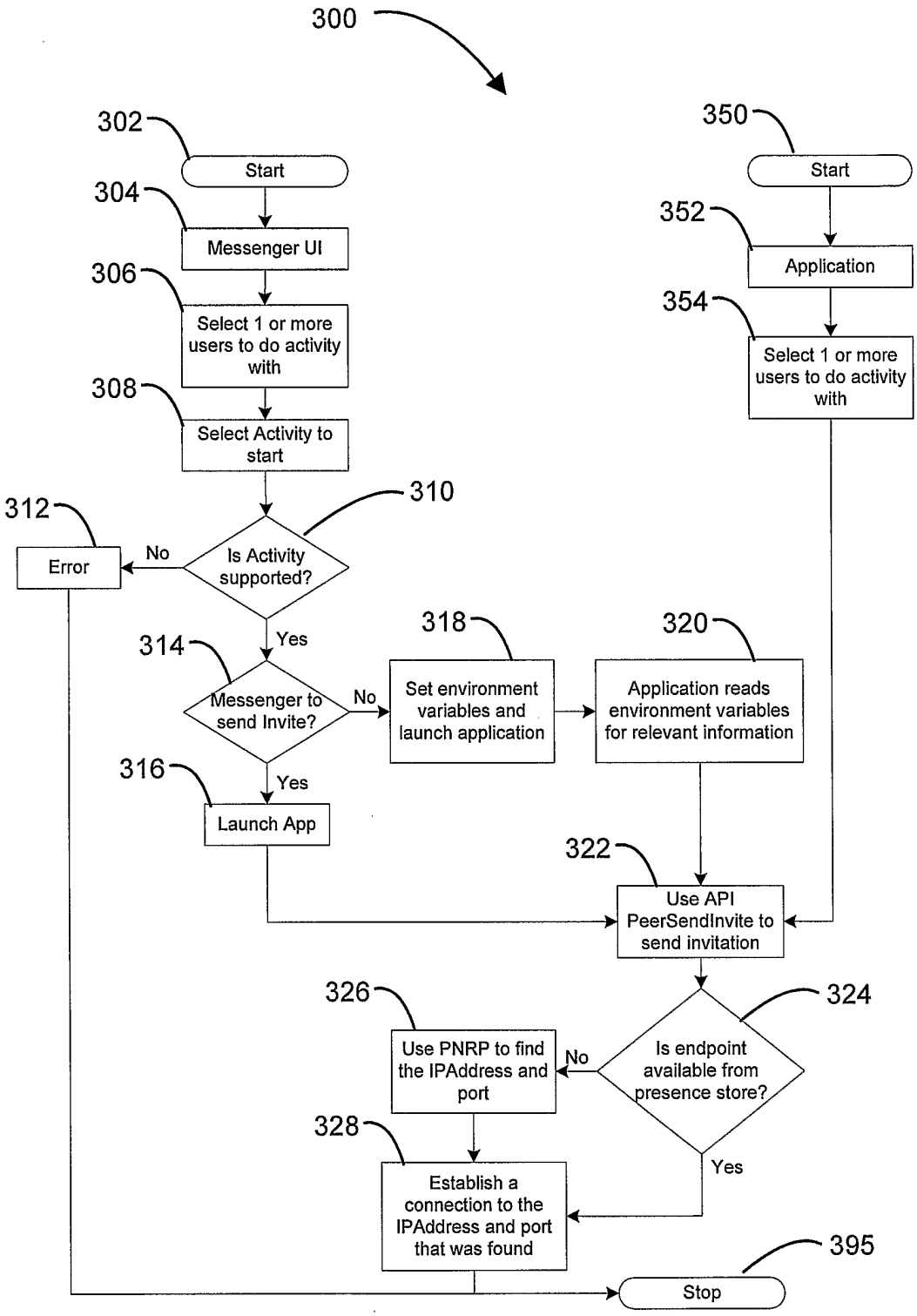
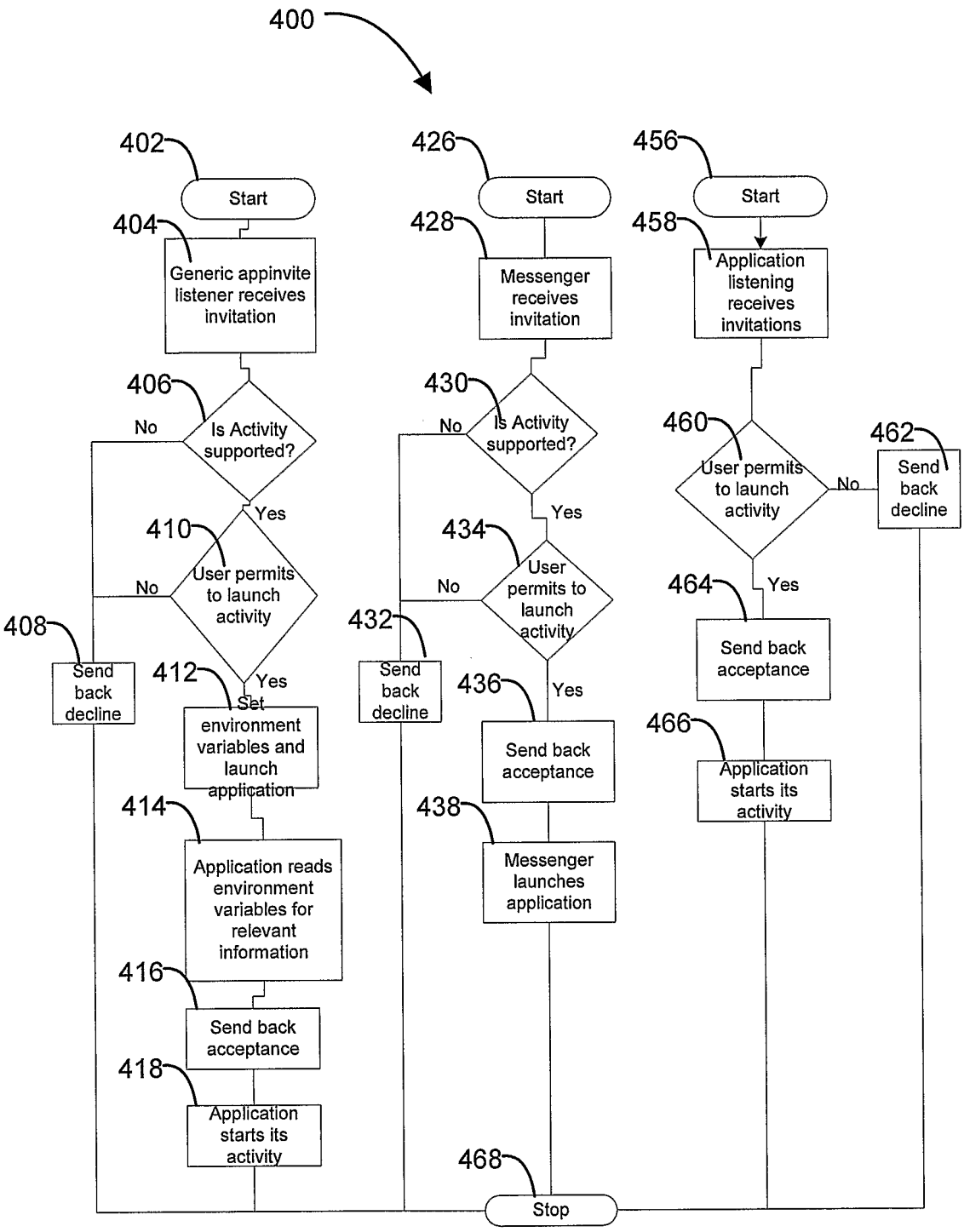

200
280
APPLICATION
284
CAPABILITIES API'S
240
CONTAC T STORE
204
PRESENCE SYSTEM
212
CAPABILITIES STORE
INVITE API'S
208
PRESENCE STORE
282
270
260
216
OBJECTS STORE
AUTHENTICATION SYSTEM
COMMUNICATION SYSTEM
202
COMMUNICATION NETWORK

Figure 2


300
302
Start
304
Messenger UI
306
Select 1 or more users to do activity with
308
Select Activity to start
310
Is Activity supported?
No
312
Error
Yes
314
Messenger to send Invite?
No
Yes
316
Launch App
318
Set environment variables and launch application
320
Application reads environment variables for relevant information
322
Use API PeerSendInvite to send invitation
324
Is endpoint available from presence store?
No
326
Use PNRP to find the IPAddress and port
Yes
328
Establish a connection to the IPAddress and port that was found
350
Start
352
Application
354
Select 1 or more users to do activity with
395
Stop


400
402
Start
404
Generic appinvite listener receives invitation
406
Is Activity supported?
No
Yes
410
User permits to launch activity
No
Yes
408
Send back decline
412
Set environment variables and launch application
414
Application reads environment variables for relevant information
416
Send back acceptance
418
Application starts its activity
426
Start
428
Messenger receives invitation
430
Is Activity supported?
No
Yes
434
User permits to launch activity
No
Yes
432
Send back decline
436
Send back acceptance
438
Messenger launches application
456
Start
458
Application listening receives invitations
460
User permits to launch activity
No
462
Send back decline
Yes
464
Send back acceptance
466
Application starts its activity
468
Stop