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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 December 2011 (08.12.2011)

PCT

(10) International Publication Number
WO 2011/153167 A1

(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/US2011/038620

(22) International Filing Date:
31 May 2011 (31.05.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/794,676 4 June 2010 (04.06.2010) US

(71) Applicant *(for all designated States except US)*: **APPLE INC.** [US/US]; 1 Infinite Loop, Cupertino, CA 95014 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **DEVINE, Graeme** [US/US]; 1 Infinite Loop. MS 302-3KS, Cupertino, CA 95014 (US).

(74) Agents: **VINCENT, Lester, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 1279 Oakmead Parkway, Sunnyvale, CA 94085-4040 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, 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, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

WO 2011/153167 A1

(54) Title: METHODS FOR USING UNIQUE IDENTIFIERS TO IDENTIFY SYSTEMS IN COLLABORATIVE INTERACTION IN A MESH NETWORK

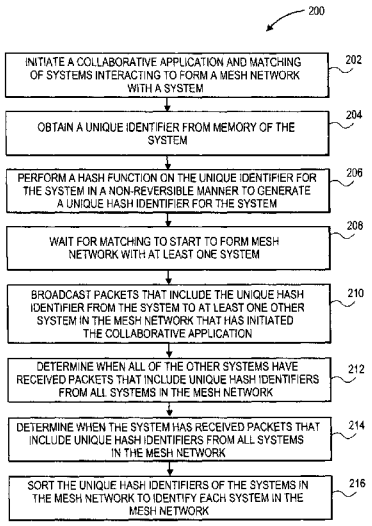

FIG. 2

(57) Abstract: Described herein are methods and systems for using unique identifiers to identify systems in collaborative interaction in a mesh network. For example, in at least certain embodiments, upon initiation of a collaborative application each system can broadcast packets that include a unique hash identifier for each system to other systems in the mesh network. Each system then can determine when the system has received packets that include the unique hash identifiers from all systems. Then, each system can sort the unique hash identifiers to identify each system.

METHODS FOR USING UNIQUE IDENTIFIERS TO IDENTIFY SYSTEMS IN COLLABORATIVE INTERACTION IN A MESH NETWORK

TECHNICAL FIELD

[0001] Embodiments of the present invention relate to methods for using unique identifiers to identify systems in collaborative interaction in a mesh network.

BACKGROUND

[0002] Various devices such as electronic devices, computing systems, portable devices, and handheld devices have collaborative applications such as software gaming applications. These devices can network with each other for a multi-player gaming experience.

[0003] One prior gaming environment allows players to interact with each other online. A server communicates over a data network with a number of client computers. The server receives information from the client computers to update the state of the multi-player game and distributes information back to the client computers regarding relevant game state for each of the client computers.

[0004] However, this prior approach has limitations in terms of connecting players and network delays. These online connections use the transmission control protocol (TCP) that provides reliable, ordered delivery of a stream of bytes from a game application on the client computer to the server and vice versa. The TCP controls segment size, flow control, the rate at which data is exchanged, and network traffic congestion. However, TCP packets may have timing, readiness, and internet issues that cause some client computers to not have relevant game information in a timely manner or at all.

SUMMARY

[0005] Described herein are methods for using unique identifiers to identify systems in collaborative interaction in, for example, a mesh network. For example, in at least certain embodiments, at least one system initiates a collaborative application (e.g., music creation, document creation, multi-player games) with other systems. In response, a data service provides a collaborative environment that matches systems and provides connection data to the systems to generate a mesh network. In one embodiment, each system in the mesh network or other network can perform a hash function on a unique identifier associated with each system to generate a unique hash identifier for each system. Then, each system can broadcast packets that include the unique hash identifier for each system and also status information to other systems in the mesh network. Each system can then determine when the system has heard from all systems and when the other systems have also heard from all other systems. Then, in one embodiment, each system can sort the unique hash identifiers to assign a relative reference value to each system in the mesh network and thus identify each system with the sorted unique hash identifier. In one embodiment, each system identifies other systems with these assigned relative reference values, which can be used for determining, for example, a player order in a multi-player gaming application or can be used for assigning each system to be a server or client in the mesh network.

[0006] In an embodiment, the unique identifier for each is obtained from memory of the system. Each system performs a hash function on the unique identifier for the system in a non-reversible manner to generate a unique hash identifier for the system and to protect the unique identifier. The unique identifier can be universally unique relative to all other systems of relatively unique within a type or subset of systems of a

particular type of hardware, or a particular combination of hardware and software, etc.

[0007] The present disclosure includes systems and devices that perform these methods, including data processing systems which perform these methods, and machine readable media which when executed on data processing systems cause the systems to perform these methods.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which:

[0009] Figure 1 illustrates a general network topology implemented in one embodiment that can include a group of "client" or "peer" computing systems 120-123, respectively, communicating with one another and with one or more services 109-114 over a network 130;

[0010] Figure 2 illustrates a flow diagram of one embodiment for a computer-implemented method 200 of using unique identifiers to identify systems in collaborative interaction in a mesh network;

[0011] Figure 3 illustrates a flow diagram of one embodiment for a computer-implemented method 300 of using unique identifiers to identify systems in collaborative interaction in a mesh network;

[0012] Figure 4 illustrates a flow diagram of one embodiment for a computer-implemented method 400 of determining a status of a cointoss;

[0013] Figure 5 illustrates an exemplary packet broadcast by a system in accordance with one embodiment;

[0014] Figure 6 illustrates a mesh network 600 having four systems in communication with each other in accordance with one embodiment;

[0015] Figure 7 illustrates a mesh network 700 having eight systems in communication with each other in accordance with one embodiment;

[0016] Figure 8 illustrates an exemplary bit string have an expected final status in accordance with one embodiment;

[0017] Figure 9 shows an embodiment of a wireless system which includes the capability for wireless communication;

[0018] Figure 10 is a block diagram illustrating an exemplary API architecture, which may be used in certain embodiments of the present disclosure; and

[0019] Figure 11 illustrates a Software Stack for an exemplary embodiment in which applications can make calls to Services A or B using several Service APIs and to Operating System (OS) using several OS APIs.

DETAILED DESCRIPTION

[0020] Described herein are methods for using unique identifiers to identify systems in collaborative interaction in a mesh network. For example, in at least certain embodiments, at least one system initiates a collaborative application (e.g., music creation, document creation, multi-player games). In response, a data service provides a collaborative environment that matches systems and provides connection data to the systems to generate a mesh network. Each system generates a hash identifier based on a unique identifier associated with the system. The hash identifiers are exchanged between systems within the mesh network. The hash identifiers can be sorted to identify systems in the mesh network.

[0021] As illustrated in Figure 1, a general network topology implemented in one embodiment can include a group of "client" or "peer"

computing systems 120-123, respectively, communicating with one another and with one or more services 109-114 over a network 130. Although illustrated as a single network cloud in Figure 1, the "network" 130 can be comprised of a variety of different components including public networks such as the Internet and private networks such as local Wi-Fi networks (e.g., 902.11*n* home wireless networks or wireless hotspots), local area Ethernet networks, cellular data networks, and WiMAX networks, to name a few. For example, system 120 may be connected to a home Wi-Fi network represented by network link 125, system 121 may be connected to a 3G network (e.g., Universal Mobile Telecommunications System ("UMTS"), High-Speed Uplink Packet Access ("HSUPA"), etc) represented by network link 126, system 122 may be connected to a WiMAX network represented by network link 127, and system 123 may be connected to a public Wi-Fi network represented by network link 128. Each of the local network links 125-128 over which the systems 120-123 are connected may be coupled to a public network such as the Internet, thereby enabling communication between the various systems 120-123 over the public network. However, if two systems are on the same local or private network (e.g., the same Wi-Fi network), then the two systems may communicate directly over that local/private network, bypassing the public network. It should be noted, of course, that the underlying principles of the present disclosure are not limited to any particular set of network types or network topologies and that the term mesh network is meant to include one or more networks of the same or different types.

[0022] Each of the systems 120-123 illustrated in Figure 1 can communicate with a data service 100 that may include a collaborative service 109 (e.g., game service, music creation service, document creation service), a connection data exchange (CDX) service 110, a matchmaker service 111, an invitation service 112, an account service 113, and an application service 114. In one embodiment, the collaborative

service 109 enables user to collaborate with collaborative applications. For example, the collaborative service 109 may be a game service that enables users to collaborate for multi-player gaming applications or other multiple user applications. The game service may include or access any of the services 110-114. The services 109-114 can be implemented as software executed across one or more physical computing systems such as servers. As shown in Figure 1, in one embodiment, the services may be implemented within the context of a larger data service 100 managed by the same entity (e.g., the same company) and accessible by each of the systems 120-123 over the network 130. The data service 100 can include a local area network (e.g., an Ethernet-based LAN) connecting various types of servers, a storage area networks ("SANs") and databases. In one embodiment, the databases store and manage data related to each of the user systems (e.g., client systems, computer systems, mobile systems) 120-123 and the users of those systems (e.g., user account data, system account data, user application data, etc.).

[0023] In one embodiment, a collaborative identifier module 130-133 is located on each system 120-123, respectively. The collaborative identifier module is associated with a collaborative software application that provides a collaborative environment in conjunction with the data service 100.

[0024] In one embodiment, the collaborative identifier module 130-133 is implemented on a game framework such as that described in co-pending applications U.S. patent application No. 61/321854, entitled "APPLICATION PROGRAMMING INTERFACE, SYSTEM AND METHOD FOR COLLABORATIVE ONLINE APPLICATIONS," Filed April 7, 2010 by Mike Lampell, attorney docket No. P9203; U.S. patent application No. 61/321842, entitled "APPARATUS AND METHOD FOR MATCHING USERS FOR ONLINE SESSIONS", Filed April 7, 2010 by Jeremy Werner, Phillip Smith, Andrew H. Vyrros, attorney docket No. P8549; U.S. patent

application No. 61/321832, entitled "APPARATUS AND METHOD FOR INVITING USERS TO ONLINE SESSIONS", Filed April 7, 2010 by Andrew H. Vyrros, Jeremy Werner, and Patrick Gates, attorney docket No. P8547; U.S. patent application No. 61/321841, entitled "APPARATUS AND METHOD FOR ESTABLISHING AND UTILIZING BACKUP COMMUNICATION CHANNELS", Filed April 7, 2010 by Jeff Tung, Barry A. Whitebook, Joe Abuan, Hyeonkuk Jeong, Andy Yang, and Roberto Garcia, attorney docket No. P9162; and U.S. patent application No. 61/321851, entitled "APPARATUS AND METHOD FOR EFFICIENTLY AND SECURELY EXCHANGING CONNECTION DATA", Filed April 7, 2010 by Joe Abuan, Jeff Tung, Robert Quattlebaum, Barry A. Whitebook, and Roberto Garcia attorney docket No. P9164 (hereinafter "Co-pending Applications"), which are assigned to the assignee of the present application and which are incorporated herein by reference in their entirety. It should be noted, however, that the game framework described in the co-pending applications is not required for complying with the underlying principles of the invention.

[0025] The matchmaker service 111 can match two or more systems for a collaborative peer to peer (P2P) session based on a specified set of conditions. For example, users of two or more of the systems may be interested in playing a particular multi-player game. In such a case, the matchmaker service 111 may identify a group of systems to participate in the game based on variables such as each user's level of expertise, the age of each of the users, the timing of the match requests, the particular game for which a match is requested and game-specific variables associated with the game. By way of example, and not limitation, the matchmaker service 111 may attempt to match users with similar levels of expertise at playing a particular game. Additionally, adults may be matched with other adults and children may be matched with other children. Moreover, the matchmaker service 111 may prioritize user requests based on the order in which those requests are received. The

underlying principles of the present disclosure are not limited to any particular set of matching criteria or any particular type of P2P application.

[0026] In response to a match request, the matchmaker service 111 can coordinate with the CDX service 110 to ensure that all matched participants receive the necessary connection data for establishing P2P sessions in an efficient and secure manner.

[0027] In one embodiment, the invitation service 112 also identifies systems for participation in collaborative P2P sessions. However, in the case of the invitation service 112, at least one of the participants is specifically identified by another participant. For example, the user of system 120 may specifically request a collaborative session with the user of system 121. As with the matchmaker service 111, in response to an invitation request, the invitation service 112 can identify the set of participants and coordinate with the CDX service 110 to ensure that all participants receive the necessary connection data for establishing P2P sessions in an efficient and secure manner.

[0028] Figure 2 illustrates a flow diagram of one embodiment for a computer-implemented method 200 of using unique identifiers to identify systems in collaborative interaction in a mesh network. The unique identifiers can be universally unique or merely unique within a set of devices, such as devices having the same set of software or hardware (or a combination of the same software and hardware). Mesh networking is a type of networking in which each node in the network may act as an independent router, regardless of whether it is connected to another network or not. The computer-implemented method 200 is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine or a system), or a combination of both.

[0029] At block 202, the processing logic can initiate a collaborative application (e.g., music creation, document creation, multi-player games)

and matching of systems interacting collaboratively to form a mesh network. The matchmaker service 111 can match two or more systems for a collaborative session (e.g., peer to peer (P2P)). At block 204, the processing logic of a system can obtain a unique identifier from memory of the system. At block 206, the processing logic can perform a hash function on the unique identifier for the system in a non-reversible manner to generate a unique hash identifier for the system.

[0030] The unique hash identifier is generated in a non-reversible manner to protect the unique identifier of the system and prevent a spoofing attack. Even though the unique hash identifier is provided to other devices in a collaborative session, the unique identifier remains a secret (relative to those other devices) because the hash function used is non-reversible (e.g., the hash function cannot be reversed to reveal the unique identifier if the unique hash identifier is known). At block 208, the processing logic can wait for the matching to start to form a mesh network with at least one system. As discussed above, the matchmaking service 111 can match two or more systems for a collaborative session (e.g., music creation session, document creation session, multi-player gaming session).

[0031] At block 210, the processing logic can broadcast packets that include the unique hash identifier from the system to at least one other system in the mesh network or other network that has initiated the collaborative application. In one embodiment, each packet also contains heard from status information (e.g., cointoss data). The status information includes a field that indicates how many systems including the first system in the mesh network that the first system has heard from, a heard from status field for the first system, a heard from status field for the second system, a heard from status field for the third system, etc.

[0032] In an embodiment, the packets are sent and received using an unreliable connectionless protocol (e.g., user datagram protocol

(UDP)). UDP is a simple to implement protocol because it does not require tracking of every packet sent or received and it does not need to initiate or end a transmission. An establishment of a series of unique network identifiers to be shared amongst systems is guaranteed by using the unreliable protocol because the reliable protocol (TCP) is not guaranteed in a mobile environment and TCP queuing is undesirable when establishing this mesh. One of the unique aspects of this unreliable protocol is that by necessity it works well over unreliable networks that can actually be unreliable with only a certain percentage of packets actually arriving.

[0033] At block 212, the processing logic can determine when all of the other systems have received packets that include unique hash identifiers from all systems in the mesh network. At block 214, the processing logic can determine when the system has received packets that include unique hash identifiers from all systems in the mesh network. At block 216, the processing logic for each system or the data service 100 can sort the unique hash identifiers in order to identify each system in the mesh network. The unique hash identifiers may be assigned relative reference values to identify each system. In certain embodiments, the reference value for each system can be used to order players in a gaming application, assign preferences (e.g., colors) to each user or player sharing the collaborative application, or assign systems to be either a server or a client in the mesh network.

[0034] Figure 3 illustrates a flow diagram of one embodiment for a computer-implemented method 300 of using unique identifiers to identify systems in collaborative interaction in a mesh network. The computer-implemented method 300 is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine or a system), or a combination of both.

[0035] At block 302, the processing logic of a system may receive a user selection to initiate a collaborative application (e.g., multi-player gaming application) and matching of systems interacting collaboratively. The matchmaker service 111 can match two or more systems for a collaborative session (e.g., peer to peer (P2P)) based on a specified set of conditions. For example, users of two or more of the systems may be interested in playing a particular multi-player game. In such a case, the matchmaker service 111 may identify a group of systems to participate in the game based on variables such as each user's level of expertise, the age of each of the users, the timing of the match requests, the particular game for which a match is requested and game-specific variables associated with the game. In response to a match request, the matchmaker service 111 can coordinate with the CDX service 110 to ensure that all matched participants receive the necessary connection data (e.g., network addresses) for establishing P2P sessions in an efficient and secure manner to form a mesh network of systems that interact collaboratively.

[0036] At block 304, the processing logic for each system can obtain a unique identifier that uniquely identifies each system. The unique identifier may be obtained from memory of the system. At block 306, the processing logic for each system may perform a hash function on the unique identifier for each system to generate a unique hash identifier for each system. At block 308, the processing logic can wait for the matching to start to form a mesh network with at least one system. As discussed above, the matchmaking service 111 can match two or more systems for a collaborative peer to peer (P2P) session (e.g., multi-player gaming session). For example, the matchmaking service may match four players having similar skill levels to play a multi-player gaming application.

[0037] At block 310, each system broadcasts packets to determine a cointoss for the multi-player gaming application. For example, the

cointoss may determine player ordering or colors for each player. For a particular system, the packets include the unique hash identifier for the particular system. Each packet also contains heard from status information (e.g., cointoss data) as illustrated in Figure 5 and described in more detail below in conjunction with Figure 5. In an embodiment, the packet for a first system includes a unique hash identifier for the first system, a field that indicates how many systems including the first system in the mesh network that the first system has heard from, a heard from status field for the first system, a heard from status field for the second system, a heard from status field for the third system, etc.

[0038] The processing logic for each system can wait to receive packets broadcasted from other systems in the mesh network at block 312. Each system may continue to broadcast packets at block 310. At block 314, one or more systems can receive packets (e.g, cointoss packets) from other systems in the mesh network or other network. For example, the first system broadcasts packets to the other systems. These packets indicate that the first system is ready for the collaborative application (e.g., multi-player application). The other systems broadcast packets and these packets indicate to the first system that these systems are ready and also that these other systems know that the first system is ready. In one embodiment, the packets are sent and received using an unreliable connectionless protocol (e.g., UDP).

[0039] At block 316, the processing logic for each system that receives packets from other systems can update internal cointoss data for the particular system. At block 318, the processing logic for each system can determine whether the cointoss is complete. The flow diagram of Figure 4 describes this operation in more detail. In an embodiment, the first system determines whether the cointoss is complete by checking the heard from status information (e.g., cointoss data). This data may indicate that the first system has heard from all systems in the mesh network or

other network and additionally that each system has individually heard from all systems (and received the unique hash identifier from all systems) in the mesh network or other network. In this case, the cointoss is complete for the first system. If all systems determine that the cointoss is complete, then the unique hash identifiers can be sorted at block 320, in each of all of the systems, in order to assign a relative reference value to each system in the mesh network or other network. If the cointoss is not complete for a particular system at block 318, then the system broadcasts packets at block 310.

[0040] Each system may sort the unique hash identifiers using the same or similar sorting routine or method (e.g., numerical, alphabetic, alphanumerical, etc.) and thus generate the same result. The resulting relative reference values may indicate a player order, player colors, etc. for the multi-player gaming application. The relative reference values may also be used to configure the systems in the mesh network or other network. For example, one system may be elected as a server and the other systems are elected as clients. Alternatively, the systems may form P2P connections.

[0041] Figure 4 illustrates a flow diagram of one embodiment for a computer-implemented method 400 of determining a status of a cointoss. The computer-implemented method 400 is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine or a system), or a combination of both. Processing logic of each system in a mesh network or other network may perform the method 400.

[0042] At block 402, the processing logic for each system can determine a heard from status for the other systems to determine if the other systems have heard from all other systems in the mesh network or other network. The other systems may be checked at block 402. For example, in one embodiment, a first system checks a heard from status

for a second system. If the second system has an expected final status (e.g., 4444 for 4 systems), then the second system has heard from all other systems and the cointoss is complete for the second system. The flow returns to block 402 if the second system has not heard from all other systems. If the second system has heard from all other systems, then the processing logic determines whether the other systems (e.g., third and fourth systems) have also heard from all other systems at block 402. If so, then the processing logic for the first system determines whether the first system has heard from all other systems in the mesh network at block 404 and has the expected final status (e.g., 4444 for 4 systems) as illustrated as bit string 800 in Figure 8 in accordance with one embodiment. The bit string 800 includes the hash identifier 802 for the system and the number of systems that each system has heard from. If the other systems do not have the expected final status, then the flow returns to block 402 to determine if the other systems (e.g., third system, fourth system) have heard from all other systems and have the expected final status (e.g., 4444 for 4 systems). This flow of operations continues until all systems have heard from all other systems and have received the unique hash identifier from each of the other systems. Once this occurs and each system has heard back from all other systems, then the cointoss is complete for all systems. At block 406, sorting of the unique hash identifiers can occur as described at block 320 of method 300.

[0043] If the first system has not heard back from all other systems at block 404, then the flow returns to block 402. In an embodiment, packets are sent and received between systems across an unreliable network using an unreliable connectionless protocol (e.g., user datagram protocol (UDP)). One of the unique aspects of this unreliable protocol is that by necessity it works well over unreliable networks that can actually be unreliable with only a certain percentage of packets actually arriving.

[0044] Figure 5 illustrates an exemplary packet broadcast by a system in accordance with one embodiment. The packet may be associated with an unreliable connectionless protocol (e.g., user datagram protocol (UDP)). Each packet 500 may contain a header portion 502 and a data portion 504. The header portion may include source information (e.g., source port), destination information (e.g., destination port), length information, and checksum information. In an embodiment, the data portion 504 includes an identification field 510 having a unique hash identifier (e.g., 64 bit identifier) for the first system, a field 512 that indicates how many systems the first system has heard from (e.g., 1 if the first system has only heard from the first system), and a heard from status field 520 of the first system that indicates heard from status of the first system (e.g., 1344 for four systems in a mesh network with 1 representing the number of systems heard by the first system, 3 representing the number of systems heard by the second system, 4 representing the number of systems heard by the third system, and 4 representing the number of systems heard by the fourth system). The data portion 504 also includes a heard from status field 530 for the second system that indicates how many systems including the second system that the second system has heard from, a heard from status field 540 for the third system, a heard from status field 550 for the fourth system, etc.

[0045] Figure 6 illustrates a mesh network 600 having four systems in communication with each other in accordance with one embodiment. It will be understood that an embodiment of the invention can use a mesh network to connect at least some of the devices in a collaborative session and that other embodiments of the invention can use a non-mesh network to connect all of the devices in a collaborative session. One or more of the systems 610, 620, 630, and 640 initiates a collaborative application. The systems are matched together as discussed at block 302 of Figure 3. Once the systems have been matched and the connections formed between systems that now have network addresses for each other then

the systems can broadcast packets to each other as discussed at block 310. For simplicity only the packets 612, 614, and 616 from system 610 have been shown in Figure 6. In a similar manner, the other systems broadcast packets as well in order to determine the cointoss and assign reference values to each system. The packets include various fields as discussed in conjunction with Figure 5. A packet may have an expected final status (e.g., 4444 for 4 systems) as illustrated as bit string 800 in Figure 8. The bit string 800 includes the hash identifier 802 for the system and the number of systems that each system has heard from.In an embodiment, the users associated with systems 610, 620, 620, and 640 may want to play a four person gaming application. Each user may need to know which system is player 1, player 2, player 3, and player 4. This information is necessary for assigning colors, positions, and having the same guaranteed result on each system. Also, if one player wants to add random elements to the game or an artificial intelligence of some sort, then a player needs to be elected to manage these additions or changes and tell others about it. The cointoss can be used to determine player ordering. The players exchange a token that is unique (e.g., hash identifiers) and wait for responses. Networking issues may cause one or more players to be missing a vital bit of information. Thus, each system needs to hear from all of the other systems and each system needs to know that the other systems have also heard from all systems.

[0046] As discussed above, in an embodiment, the packets are sent and received using an unreliable connectionless protocol (e.g., user datagram protocol (UDP)). An establishment of a series of unique network identifiers to be shared amongst systems is guaranteed by using the unreliable protocol because the reliable protocol (TCP) is not guaranteed in a mobile environment and TCP queuing is undesirable when establishing this mesh. One of the unique aspects of this unreliable protocol is that by necessity it works well over unreliable networks that can

actually be unreliable with only a certain percentage of packets actually arriving.

[0047] In one embodiment, system 610 has a heard from status field of 1344, system 620 has a heard from status field of 3144, system 630 has a heard from status field of 4134, and system 640 has a heard from status field of 4314. These status fields would indicate that all systems having the same data in regards to each other. If system 610 hears from system 620, then the status field for system 610 updates to 2344.

[0048] Alternatively, system 610 may have a different status field for system 620 than system 620 actually has because the status field of system 620 has been recently updated and system 610 is not aware of this update.

[0049] Mesh networking allows for continuous connections and reconfiguration around broken or blocked paths by "hopping" from node to node until the destination is reached. Mesh networks are self-healing: the network can still operate when one node breaks down or a connection goes bad. As a result, the network may typically be very reliable, as there is often more than one path between a source and a destination in the network. A mesh network can be a local area network (LAN) that employs one of two connection arrangements, full mesh topology or partial mesh topology. In the full mesh topology, each (system or other device) is connected directly to each of the others. In the partial mesh topology, some nodes are connected to all the others, but some of the nodes are connected only to those other nodes with which they exchange the most data.

[0050] Figure 7 illustrates a mesh network 700 having eight systems in communication with each other in accordance with one embodiment. The systems 710, 720, 730, 740, 750, 760, 770, and 780 initiate a collaborative application and are matched together as discussed

at block 302 of Figure 3. Once the systems have been matched and the connections formed between systems then the systems can broadcast packets to each other as discussed at block 310. Exemplary bidirectional communication links 781-796 used in part for broadcasting packets are illustrated in Figure 7. The packets with the cointoss data are used to determine the cointoss and assign reference values to each system. In certain embodiments, the collaborative application is a multi-player gaming application and the assigned reference values determine player ordering. The reference values can also be used to configure the systems in the mesh network. One system can be elected a server and the other systems can be elected a client. For example, system 750 may be elected a server. Packets sent from system 770 to system 730 may be routed through system 750. If system 750 quits the collaborative application (e.g., multi-player gaming application), then another system can be elected to be the server. In an embodiment, after a collaborative application begins then additional players can not be added. A star or ring network of nodes can also be formed using the reference values.

[0051] In an alternative embodiment, the network 700 is a node based network that implements the cointoss to obtain cointoss data from other systems and to identify systems in the network 700. For example, system 740 can be a node that communicates with system 710, system 770, system 720, system 750, and system 780. System 750 can be a node that communicates with system 730, system 760, and system 780. If either of system 740 or 750 drops from the node network, then the node network automatically reforms using the cointoss data.

[0052] Attention is now directed towards embodiments of a system architecture that may be embodied within any portable or non-portable device including but not limited to a communication device (e.g. mobile phone, smart phone), a multi-media device (e.g., MP3 player, TV, radio), a portable or handheld computer (e.g., tablet, netbook, laptop), a desktop

computer, an All-In-One desktop, a peripheral device, or any other system or device adaptable to the inclusion of system architecture 900, including combinations of two or more of these types of devices. Figure 9 is a block diagram of one embodiment of system 900 that generally includes one or more computer-readable non-transitory storage mediums 901, processing system 904, Input/Output (I/O) subsystem 906, radio frequency (RF) circuitry 908 and audio circuitry 910. These components may be coupled by one or more communication buses or signal lines 903.

[0053] It should be apparent that the architecture shown in Figure 9 is only one example architecture of system 900, and that system 900 could have more or fewer components than shown, or a different configuration of components. The various components shown in Figure 9 can be implemented in hardware, software, firmware or any combination thereof, including one or more signal processing and/or application specific integrated circuits.

[0054] RF circuitry 908 is used to send and receive information over a wireless link or network to one or more other devices and includes well-known circuitry for performing this function. RF circuitry 908 and audio circuitry 910 are coupled to processing system 904 via peripherals interface 916. Interface 916 includes various known components for establishing and maintaining communication between peripherals and processing system 904. Audio circuitry 910 is coupled to audio speaker 950 and microphone 952 and includes known circuitry for processing voice signals received from interface 916 to enable a user to communicate in real-time with other users. In some embodiments, audio circuitry 910 includes a headphone jack (not shown).

[0055] Peripherals interface 916 couples the input and output peripherals of the system to processor 918 and computer-readable medium 901. One or more processors 918 communicate with one or more computer-readable mediums 901 via controller 920. Computer-readable

medium 901 can be any device or medium (e.g., storage device, non-transitory storage medium) that can store code and/or data for use by one or more processors 918. Medium 901 can include a memory hierarchy, including but not limited to cache, main memory and secondary memory. The memory hierarchy can be implemented using any combination of RAM (*e.g.*, SRAM, DRAM, DDRAM), ROM, FLASH, magnetic and/or optical storage devices, such as disk drives, magnetic tape, CDs (compact disks) and DVDs (digital video discs). Medium 901 may also include a transmission medium for carrying information-bearing signals indicative of computer instructions or data (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, including but not limited to the Internet (also referred to as the World Wide Web), intranet(s), Local Area Networks (LANs), Wide Local Area Networks (WLANs), Storage Area Networks (SANs), Metropolitan Area Networks (MAN) and the like.

[0056] One or more processors 918 run various software components stored in medium 901 to perform various functions for system 900. In some embodiments, the software components include operating system 922, communication module (or set of instructions) 924, touch processing module (or set of instructions) 926, graphics module (or set of instructions) 928, one or more applications (or set of instructions) 930, and collaborative identifier module [or set of instructions] 938. In an embodiment, a collaborative application is associated with a collaborative identifier module 938. Each of these modules and above noted applications correspond to a set of instructions for performing one or more functions described above and the methods described in this application (e.g., the computer-implemented methods and other information processing methods described herein). These modules (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise rearranged in various embodiments.

[0057] In some embodiments, medium 901 may store a subset of the modules and data structures identified above. Furthermore, medium 901 may store additional modules and data structures not described above.

[0058] Operating system 922 includes various procedures, sets of instructions, software components and/or drivers for controlling and managing general system tasks (*e.g.*, memory management, storage device control, power management, *etc.*) and facilitates communication between various hardware and software components.

[0059] Communication module 924 facilitates communication with other devices over one or more external ports 936 or via RF circuitry 908 and includes various software components for handling data received from RF circuitry 908 and/or external port 936.

[0060] Graphics module 928 includes various known software components for rendering, animating and displaying graphical objects on a display surface. In embodiments in which touch I/O device 912 is a touch sensitive display (e.g., touch screen), graphics module 928 includes components for rendering, displaying, and animating objects on the touch sensitive display.

[0061] One or more applications 930 can include any applications installed on system 900, including without limitation, a collaborative application, a browser, address book, contact list, email, instant messaging, word processing, keyboard emulation, widgets, JAVA-enabled applications, encryption, digital rights management, voice recognition, voice replication, location determination capability (such as that provided by the global positioning system (GPS)), a music player, *etc.*

[0062] Touch processing module 926 includes various software components for performing various tasks associated with touch I/O device

912 including but not limited to receiving and processing touch input received from I/O device 912 via touch I/O device controller 932.

[0063] System 900 may further include collaborative identifier module 938 (e.g., collaborative mesh network module) for performing the method/functions as described herein in connection with Figures 2-4. In one embodiment, the collaborative identifier module 938 may at least function to use unique identifiers associated with systems in a mesh network to identify the systems. For example, the collaborative identifier module may identify the systems for player ordering in a multi-player gaming application. The collaborative identifier module can perform a hash function on a unique identifier associated with each system to generate a unique hash identifier for each system. Then, each system broadcasts packets that include the unique hash identifier for each system and also status information to other systems in the mesh network. The collaborative identifier module then can determine when the system has heard from all systems and when the other systems have also heard from all other systems. Then, the collaborative identifier module of each system can sort the unique hash identifiers to assign a relative reference value to each system in the mesh network and thus identify each system. In one embodiment, the collaborative identifier module of each system identifies other systems with these assigned relative reference values, which can be used for determining a player order in a multi-player gaming application or can be used for assigning each system to be a server or client in the mesh network.

[0064] Module 938 may also interact with collaborative application 930 to provide the methods and functionality described herein. Module 938 may be embodied as hardware, software, firmware, or any combination thereof. Although module 938 is shown to reside within medium 901, all or portions of module 938 may be embodied within other

components within system 900 or may be wholly embodied as a separate component within system 900.

[0065] I/O subsystem 906 is coupled to touch I/O device 912 and one or more other I/O devices 914 for controlling or performing various functions. Touch I/O device 912 communicates with processing system 904 via touch I/O device controller 2032, which includes various components for processing user touch input (*e.g.*, scanning hardware). One or more other input controllers 2034 receives/sends electrical signals from/to other I/O devices 914. Other I/O devices 914 may include physical buttons, dials, slider switches, sticks, keyboards, touch pads, additional display screens, or any combination thereof.

[0066] If embodied as a touch screen, touch I/O device 912 displays visual output to the user in a GUI. The visual output may include text, graphics, video, and any combination thereof. Some or all of the visual output may correspond to user-interface objects. Touch I/O device 912 forms a touch-sensitive surface that accepts touch input from the user. Touch I/O device 912 and touch screen controller 932 (along with any associated modules and/or sets of instructions in medium 901) detects and tracks touches or near touches (and any movement or release of the touch) on touch I/O device 912 and converts the detected touch input into interaction with graphical objects, such as one or more user-interface objects. In the case in which device 912 is embodied as a touch screen, the user can directly interact with graphical objects that are displayed on the touch screen. Alternatively, in the case in which device 912 is embodied as a touch device other than a touch screen (e.g., a touch pad), the user may indirectly interact with graphical objects that are displayed on a separate display screen embodied as I/O device 914.

[0067] Touch I/O device 912 may be analogous to the multi-touch sensitive surface described in the following U.S. Patents: 6,323,846 (Westerman et al.), 6,570,557 (Westerman et al.), and/or 6,677,932

(Westerman), and/or U.S. Patent Publication 2002/0015024A1, each of which is hereby incorporated by reference in its entirety.

[0068] Embodiments in which touch I/O device 912 is a touch screen, the touch screen may use LCD (liquid crystal display) technology, LPD (light emitting polymer display) technology, OLED (organic LED), or OEL (organic electro luminescence), although other display technologies may be used in other embodiments.

[0069] Feedback may be provided by touch I/O device 912 based on the user's touch input as well as a state or states of what is being displayed and/or of the computing system. Feedback may be transmitted optically (e.g., light signal or displayed image), mechanically (e.g., haptic feedback, touch feedback, force feedback, or the like), electrically (e.g., electrical stimulation), olfactory, acoustically (e.g., beep or the like), or the like or any combination thereof and in a variable or non-variable manner.

[0070] System 900 also includes power system 944 for powering the various hardware components and may include a power management system, one or more power sources, a recharging system, a power failure detection circuit, a power converter or inverter, a power status indicator and any other components typically associated with the generation, management and distribution of power in portable devices.

[0071] In some embodiments, peripherals interface 916, one or more processors 918, and memory controller 920 may be implemented on a single chip, such as processing system 904. In some other embodiments, they may be implemented on separate chips.

[0072] In certain embodiments of the present disclosure, the system 900 can be used to implement at least some of the methods discussed in the present disclosure.

[0073] Some portions of the detailed descriptions are presented in terms of algorithms which include operations on data stored within a

computer memory. An algorithm is generally a self-consistent sequence of operations leading to a desired result. The operations typically require or involve physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0074] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, can refer to the action and processes of a data processing system, or similar electronic device, that manipulates and transforms data represented as physical (electronic) quantities within the system's registers and memories into other data similarly represented as physical quantities within the system's memories or registers or other such information storage, transmission or display devices.

[0075] The present disclosure can relate to an apparatus for performing one or more of the operations described herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a machine (e.g. computer) readable non-transitory storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), erasable programmable ROMs (EPROMs), electrically erasable

programmable ROMs (EEPROMs), flash memory, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a bus.

[0076] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, machines store and communicate (internally and with other devices over a network) code and data using machine-readable media, such as machine storage media (e.g., magnetic disks; optical disks; random access memory; read only memory; flash memory devices; phase-change memory).

[0077] One or more Application Programming Interfaces (APIs) may be used in some embodiments. An API is an interface implemented by a program code component or hardware component (hereinafter "API-implementing component") that allows a different program code component or hardware component (hereinafter "API-calling component") to access and use one or more functions, methods, procedures, data structures, classes, and/or other services provided by the API-implementing component. An API can define one or more parameters that are passed between the API-calling component and the API-implementing component.

[0078] An API allows a developer of an API-calling component (which may be a third party developer) to leverage specified features provided by an API-implementing component. There may be one API-calling component or there may be more than one such component. An API can be a source code interface that a computer system or program library provides in order to support requests for services from an application. An operating system (OS) can have multiple APIs to allow applications running on the OS to call one or more of those APIs, and a service (such as a program library) can have multiple APIs to allow an application that uses the service to call one or more of those APIs. An API

can be specified in terms of a programming language that can be interpreted or compiled when an application is built.

[0079] In some embodiments the API-implementing component may provide more than one API, each providing a different view of or with different aspects that access different aspects of the functionality implemented by the API-implementing component. For example, one API of an API-implementing component can provide a first set of functions and can be exposed to third party developers, and another API of the API-implementing component can be hidden (not exposed) and provide a subset of the first set of functions and also provide another set of functions, such as testing or debugging functions which are not in the first set of functions. In other embodiments the API-implementing component may itself call one or more other components via an underlying API and thus be both an API-calling component and an API-implementing component.

[0080] An API defines the language and parameters that API-calling components use when accessing and using specified features of the API-implementing component. For example, an API-calling component accesses the specified features of the API-implementing component through one or more API calls or invocations (embodied for example by function or method calls) exposed by the API and passes data and control information using parameters via the API calls or invocations. The API-implementing component may return a value through the API in response to an API call from an API-calling component. While the API defines the syntax and result of an API call (e.g., how to invoke the API call and what the API call does), the API may not reveal how the API call accomplishes the function specified by the API call. Various API calls are transferred via the one or more application programming interfaces between the calling (API-calling component) and an API-implementing component. Transferring the API calls may include issuing, initiating, invoking, calling,

receiving, returning, or responding to the function calls or messages; in other words, transferring can describe actions by either of the API-calling component or the API-implementing component. The function calls or other invocations of the API may send or receive one or more parameters through a parameter list or other structure. A parameter can be a constant, key, data structure, object, object class, variable, data type, pointer, array, list or a pointer to a function or method or another way to reference a data or other item to be passed via the API.

[0081] Furthermore, data types or classes may be provided by the API and implemented by the API-implementing component. Thus, the API-calling component may declare variables, use pointers to, use or instantiate constant values of such types or classes by using definitions provided in the API.

[0082] Generally, an API can be used to access a service or data provided by the API-implementing component or to initiate performance of an operation or computation provided by the API-implementing component. By way of example, the API-implementing component and the API-calling component may each be any one of an operating system, a library, a device driver, an API, an application program, or other module (it should be understood that the API-implementing component and the API-calling component may be the same or different type of module from each other). API-implementing components may in some cases be embodied at least in part in firmware, microcode, or other hardware logic. In some embodiments, an API may allow a client program (e.g., collaborative application) to use the services provided by a Software Development Kit (SDK) library. In other embodiments an application or other client program may use an API provided by an Application Framework. In these embodiments the application or client program may incorporate calls to functions or methods provided by the SDK and provided by the API or use data types or objects defined in the SDK and

provided by the API. An Application Framework may in these embodiments provide a main event loop for a program that responds to various events defined by the Framework. The API allows the application to specify the events and the responses to the events using the Application Framework. In some implementations, an API call can report to an application the capabilities or state of a hardware device, including those related to aspects such as input capabilities and state, output capabilities and state, processing capability, power state, storage capacity and state, communications capability, etc., and the API may be implemented in part by firmware, microcode, or other low level logic that executes in part on the hardware component.

[0083] The API-calling component may be a local component (i.e., on the same data processing system as the API-implementing component) or a remote component (i.e., on a different data processing system from the API-implementing component) that communicates with the API-implementing component through the API over a network. It should be understood that an API-implementing component may also act as an API-calling component (i.e., it may make API calls to an API exposed by a different API-implementing component) and an API-calling component may also act as an API-implementing component by implementing an API that is exposed to a different API-calling component.

[0084] The API may allow multiple API-calling components written in different programming languages to communicate with the API-implementing component (thus the API may include features for translating calls and returns between the API-implementing component and the API-calling component); however the API may be implemented in terms of a specific programming language. An API-calling component can, in one embedment, call APIs from different providers such as a set of APIs from an OS provider and another set of APIs from a plug-in provider

and another set of APIs from another provider (e.g. the provider of a software library) or creator of the another set of APIs.

[0085] Figure 10 is a block diagram illustrating an exemplary API architecture, which may be used in certain embodiments of the present disclosure. As shown in Figure 10, the API architecture 3200 includes the API-implementing component 1010 (e.g., an operating system, a library, a device driver, an API, an application program, software or other module) that implements the API 1020. The API 1020 specifies one or more functions, methods, classes, objects, protocols, data structures, formats and/or other features of the API-implementing component that may be used by the API-calling component 1030. The API 1020 can specify at least one calling convention that specifies how a function in the API-implementing component receives parameters from the API-calling component and how the function returns a result to the API-calling component. The API-calling component 1030 (e.g., an operating system, a library, a device driver, an API, an application program, software or other module) makes API calls through the API 1020 to access and use the features of the API-implementing component 1010 that are specified by the API 1020. The API-implementing component 1010 may return a value through the API 1020 to the API-calling component 1030 in response to an API call.

[0086] It will be appreciated that the API-implementing component 1010 may include additional functions, methods, classes, data structures, and/or other features that are not specified through the API 1020 and are not available to the API-calling component 1030. It should be understood that the API-calling component 1030 may be on the same system as the API-implementing component 1010 or may be located remotely and accesses the API-implementing component 1010 using the API 1020 over a network. While Figure 10 illustrates a single API-calling component 1030 interacting with the API 1020, it should be understood that other API-

calling components, which may be written in different languages (or the same language) than the API-calling component 1030, may use the API 1020.

[0087] The API-implementing component 1010, the API 1020, and the API-calling component 1030 may be stored in a machine-readable medium (e.g., computer-readable medium), which includes any mechanism for storing information in a form readable by a machine (e.g., a computer or other data processing system). For example, a machine-readable medium includes magnetic disks, optical disks, random access memory; read only memory, flash memory devices, etc.

[0088] In Figure 11 ("Software Stack"), an exemplary embodiment, applications can make calls to Services A or B using several Service APIs and to Operating System (OS) using several OS APIs. Services A and B can make calls to OS using several OS APIs.

[0089] Note that the Service 2 has two APIs, one of which (Service 2 API 1) receives calls from and returns values to Application 1 and the other (Service 2 API 2) receives calls from and returns values to Application 2. Service 1 (which can be, for example, a software library) makes calls to and receives returned values from OS API 1, and Service 2 (which can be, for example, a software library) makes calls to and receives returned values from both OS API 1 and OS API 2. Application 2 makes calls to and receives returned values from OS API 2.

[0090] In the foregoing specification, the disclosure has been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope of the disclosure as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A computer-implemented method comprising:

initiating a collaborative application on a system;

obtaining a unique identifier from memory of the system;

performing a hash function on the unique identifier for the system in a non-reversible manner to generate a unique hash identifier for the system;

broadcasting packets that include the unique hash identifier from the system to at least one other system in a network that has initiated the collaborative application; and

determining when the system has received packets that include unique hash identifiers from all systems in the network that have initiated the collaborative application.

2. The computer-implemented method of claim 1, further comprising:

sorting the unique hash identifiers of the systems in the network to assign a relative reference value for each system in the network.

3. The computer-implemented method of claim 1, further comprising:

determining when each system has received the unique hash identifiers from all systems in the network and wherein the network is a mesh network.

4. The computer-implemented method of claim 2, wherein the unique hash identifier is generated in a non-reversible manner to protect the unique identifier of the system.

5. A machine readable non-transitory storage medium containing executable computer program instructions which when executed by a

computing system cause said system to perform a method, the method comprising:

initiating a collaborative application on a system;

obtaining a unique identifier from memory of the system;

performing a hash function on the unique identifier for the system in a non-reversible manner to generate a unique hash identifier for the system;

broadcasting packets that include the unique hash identifier from the system to at least one other system in a network that has initiated the collaborative application; and

determining when the system has received packets that include unique hash identifiers from all systems in the network that have initiated the collaborative application.

6. The medium of claim 5, the method further comprises:

sorting the unique hash identifiers of the systems in the network to assign a relative reference value for each system in the network.

7. The medium of claim 5, the method further comprises:

determining when each system has received the unique hash identifiers from all systems in the network and wherein the network is a mesh network.

8. A computer-implemented method comprising:

initiating a multi-player gaming application on a system;

performing a hash function on a unique identifier for the system to generate a unique hash identifier for the system;

broadcasting packets that include the unique hash identifier from the system to at least one other system in a mesh network that has initiated the multi-player gaming application; and

determining when the system has received packets that include unique hash identifiers from all systems in the mesh network that have initiated the multi-player gaming application.

9. The computer-implemented method of claim 8, further comprising:

determining when each system has received the unique hash identifiers from all systems in the mesh network.

10. The computer-implemented method of claim 9, further comprising:

sorting the unique hash identifiers to assign a relative reference value to each system in the mesh network.

11. The computer-implemented method of claim 10, wherein the relative reference values in combination generate a player order for each player associated with a system in the mesh network.

12. The computer-implemented method of claim 10, wherein one of the relative reference values is used to assign a system to be a server in the mesh network.

13. The computer-implemented method of claim 12, wherein the other relative reference values are used to assign the other systems to be clients in the mesh network.

14. A machine readable non-transitory storage medium containing executable computer program instructions which when executed by a computing system cause said system to perform a method, the method comprising:

initiating a multi-player gaming application on a system;

performing a hash function on a unique identifier for the system to generate a unique hash identifier for the system;

broadcasting packets that include the unique hash identifier from the system to at least one other system in a mesh network that has initiated the multi-player gaming application; and

determining when the system has received packets that include unique hash identifiers from all systems in the mesh network that have initiated the multi-player gaming application.

15. The medium of claim 14, the method further comprises:

determining when each system has received the unique hash identifiers from all systems in the mesh network; and

sorting the unique hash identifiers to assign a relative reference value to each system in the mesh network.

16. The medium of claim 15, wherein the relative reference values in combination generate a player order for each player associated with a system in the mesh network.

17. The medium of claim 14, wherein one of the relative reference values is used to assign a system to be a server in the mesh network and the other relative reference values are used to assign the other systems to be clients in the mesh network.

18. A computer-implemented method comprising:

initiating a multi-player gaming application on a plurality of systems in a mesh network;

performing a hash function on a unique identifier for each system to generate a unique hash identifier for each system;

broadcasting packets that include the unique hash identifier and status information from each system in the mesh network; and

determining when each system has received packets that include the unique hash identifiers and status information from all systems in the mesh network.

19. The computer-implemented method of claim 18, further comprising:

sorting the unique hash identifiers to assign a relative reference value to each system in the mesh network.

20. The computer-implemented method of claim 19, wherein the relative reference values in combination generate a player order for each player associated with a system in the mesh network.

21. The computer-implemented method of claim 5, wherein each packet is broadcast and received with an unreliable connectionless protocol.

22. A machine readable non-transitory storage medium containing executable computer program instructions which when executed by a computing system cause said system to perform a method, the method comprising:

initiating a multi-player gaming application on a plurality of systems in a mesh network;

performing a hash function on a unique identifier for each system to generate a unique hash identifier for each system;

broadcasting packets that include the unique hash identifier and status information from each system in the mesh network; and

determining when each system has received packets that include the unique hash identifiers and status information from all systems in the mesh network.

23. The medium of claim 22, the method further comprises:

sorting the unique hash identifiers to assign a relative reference value to each system in the mesh network.

24. The medium of claim 23, wherein the relative reference values in combination generate a player order for each player associated with a system in the mesh network.

25. The medium of claim 24, wherein each packet is broadcast and received with an unreliable connectionless protocol.

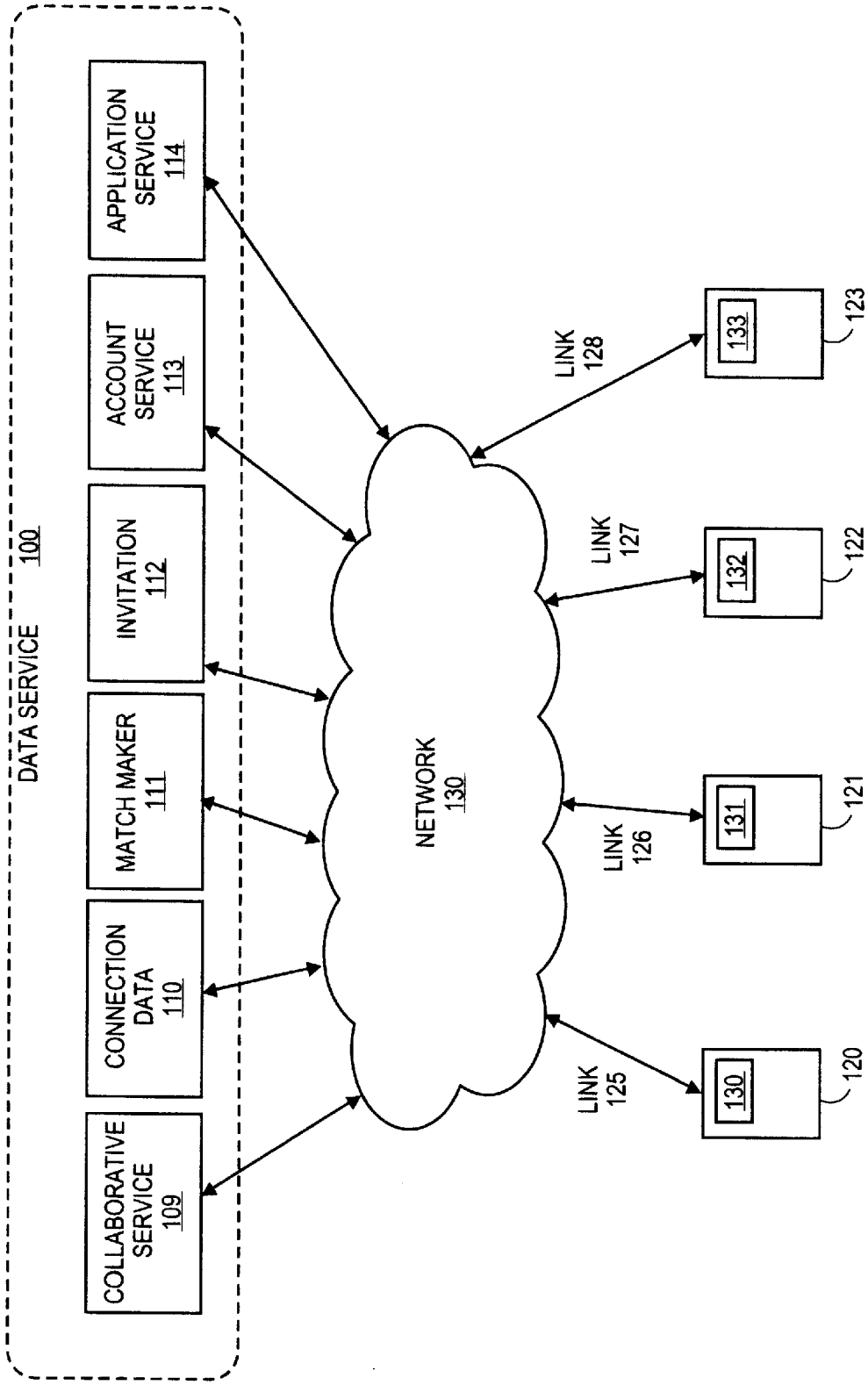
DATA SERVICE 100
APPLICATION SERVICE 114
ACCOUNT SERVICE 113
INVITATION 112
MATCH MAKER 111
CONNECTION DATA 110
COLLABORATIVE SERVICE 109
NETWORK 130
LINK 128
LINK 127
LINK 126
LINK 125
133
123
132
122
131
121
130
120

FIG. 1

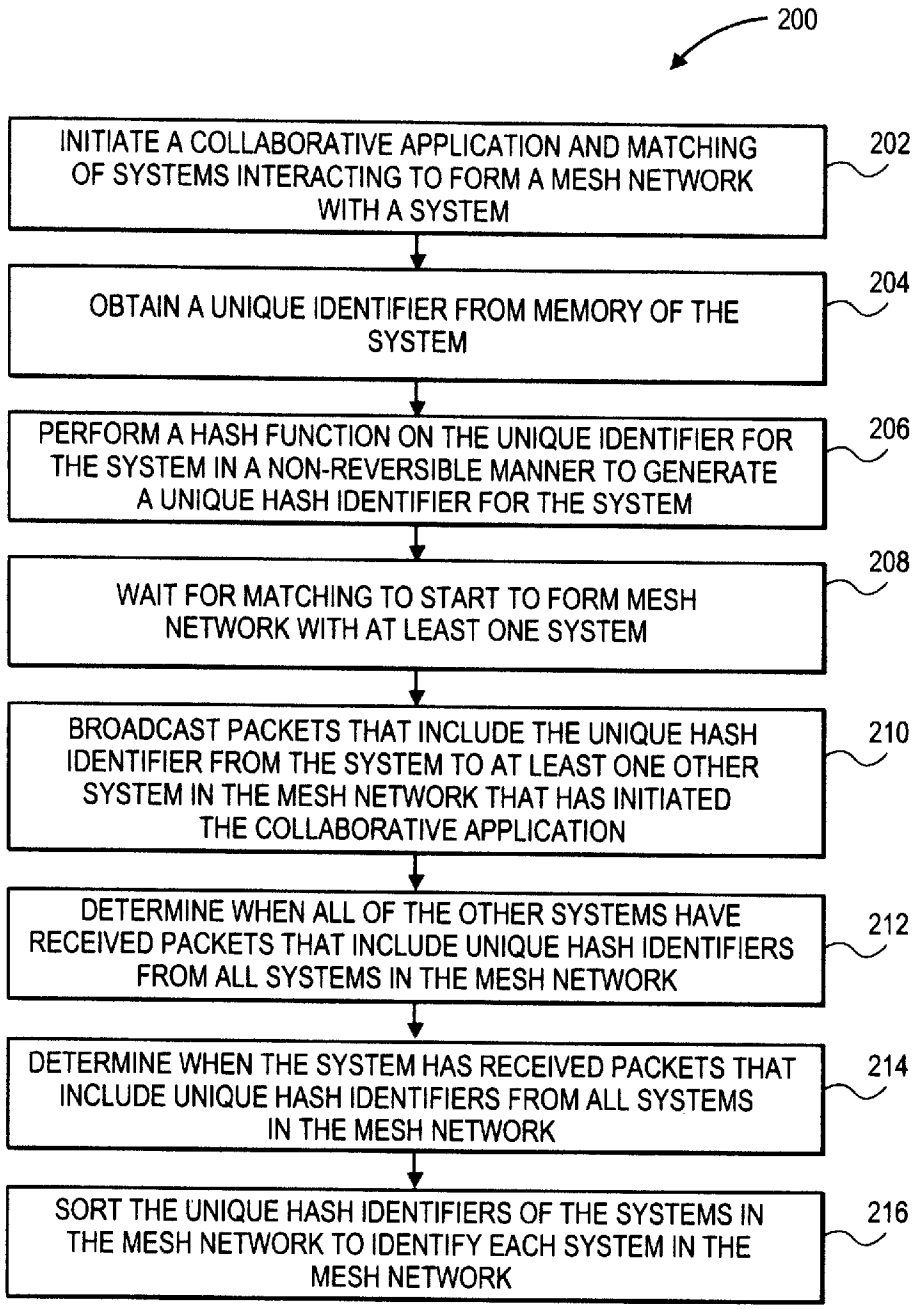
200
INITIATE A COLLABORATIVE APPLICATION AND MATCHING OF SYSTEMS INTERACTING TO FORM A MESH NETWORK WITH A SYSTEM
202
OBTAIN A UNIQUE IDENTIFIER FROM MEMORY OF THE SYSTEM
204
PERFORM A HASH FUNCTION ON THE UNIQUE IDENTIFIER FOR THE SYSTEM IN A NON-REVERSIBLE MANNER TO GENERATE A UNIQUE HASH IDENTIFIER FOR THE SYSTEM
206
WAIT FOR MATCHING TO START TO FORM MESH NETWORK WITH AT LEAST ONE SYSTEM
208
BROADCAST PACKETS THAT INCLUDE THE UNIQUE HASH IDENTIFIER FROM THE SYSTEM TO AT LEAST ONE OTHER SYSTEM IN THE MESH NETWORK THAT HAS INITIATED THE COLLABORATIVE APPLICATION
210
DETERMINE WHEN ALL OF THE OTHER SYSTEMS HAVE RECEIVED PACKETS THAT INCLUDE UNIQUE HASH IDENTIFIERS FROM ALL SYSTEMS IN THE MESH NETWORK
212
DETERMINE WHEN THE SYSTEM HAS RECEIVED PACKETS THAT INCLUDE UNIQUE HASH IDENTIFIERS FROM ALL SYSTEMS IN THE MESH NETWORK
214
SORT THE UNIQUE HASH IDENTIFIERS OF THE SYSTEMS IN THE MESH NETWORK TO IDENTIFY EACH SYSTEM IN THE MESH NETWORK
216

FIG. 2

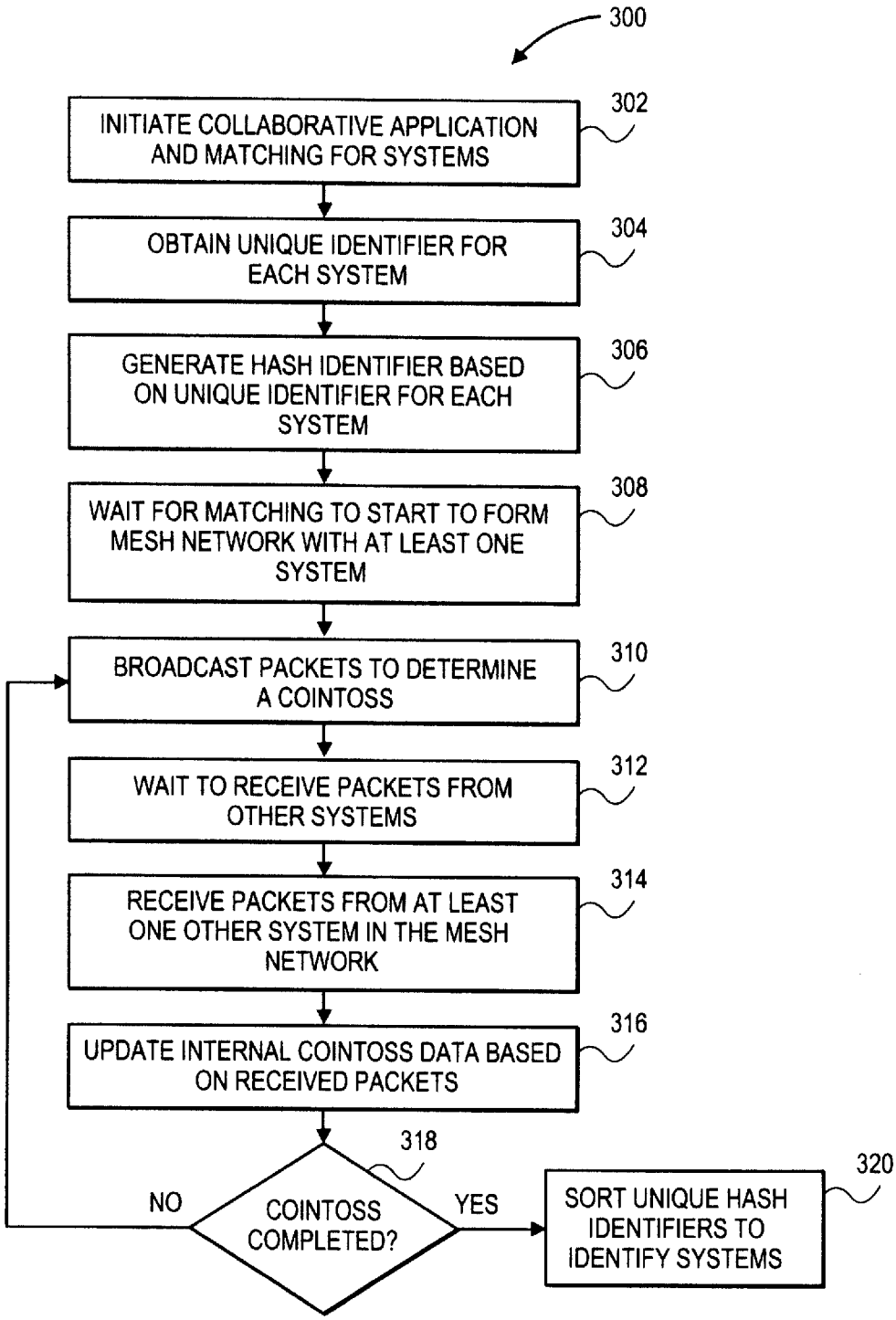
300
INITIATE COLLABORATIVE APPLICATION AND MATCHING FOR SYSTEMS
302
OBTAIN UNIQUE IDENTIFIER FOR EACH SYSTEM
304
GENERATE HASH IDENTIFIER BASED ON UNIQUE IDENTIFIER FOR EACH SYSTEM
306
WAIT FOR MATCHING TO START TO FORM MESH NETWORK WITH AT LEAST ONE SYSTEM
308
BROADCAST PACKETS TO DETERMINE A COINTOSS
310
WAIT TO RECEIVE PACKETS FROM OTHER SYSTEMS
312
RECEIVE PACKETS FROM AT LEAST ONE OTHER SYSTEM IN THE MESH NETWORK
314
UPDATE INTERNAL COINTOSS DATA BASED ON RECEIVED PACKETS
316
318
COINTOSS COMPLETED?
NO
YES
SORT UNIQUE HASH IDENTIFIERS TO IDENTIFY SYSTEMS
320

FIG. 3

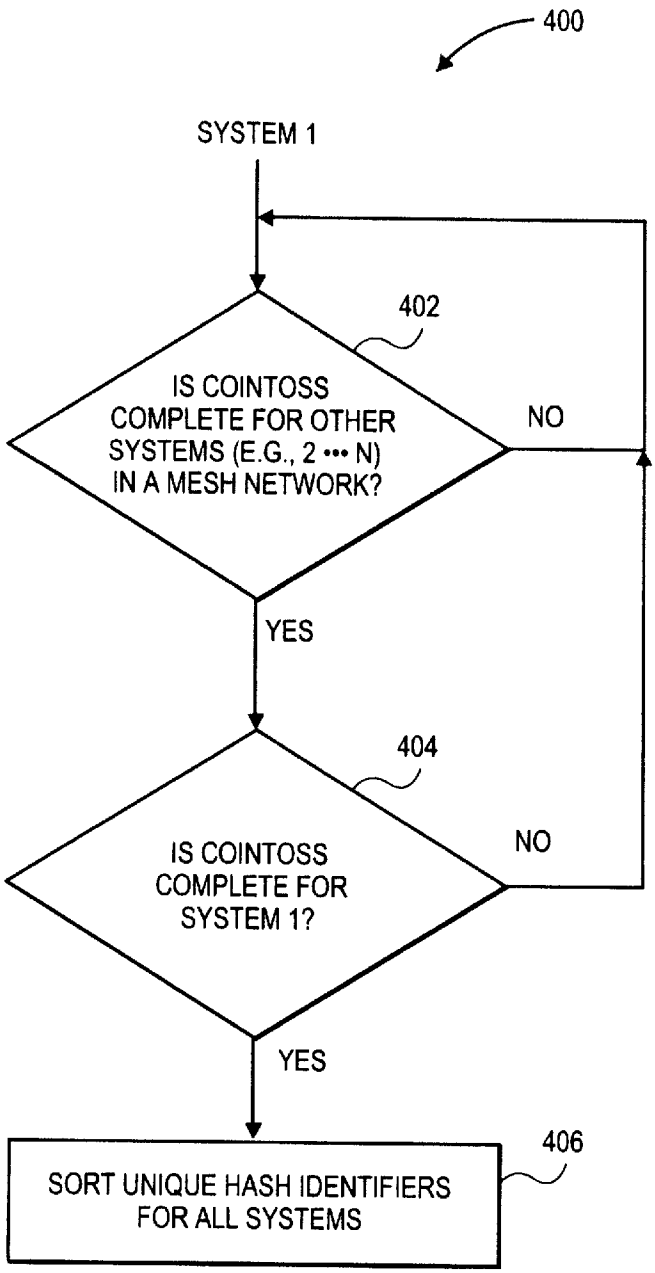
400
SYSTEM 1
402
IS COINTOSS COMPLETE FOR OTHER SYSTEMS (E.G., 2 ••• N) IN A MESH NETWORK?
NO
YES
404
IS COINTOSS COMPLETE FOR SYSTEM 1?
NO
YES
406
SORT UNIQUE HASH IDENTIFIERS FOR ALL SYSTEMS

FIG. 4

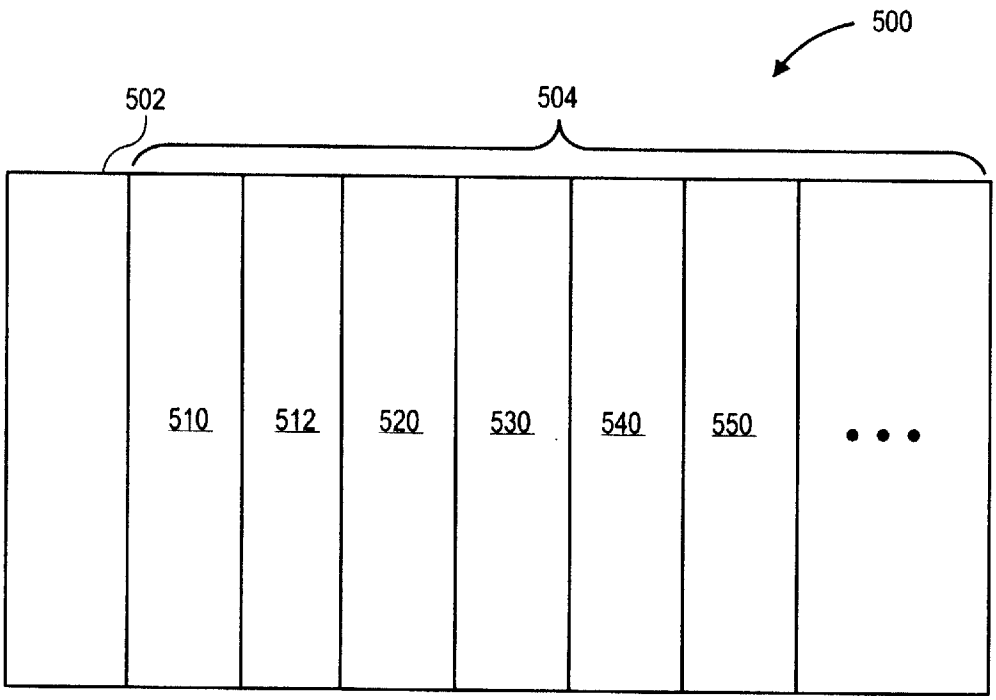

FIG. 5

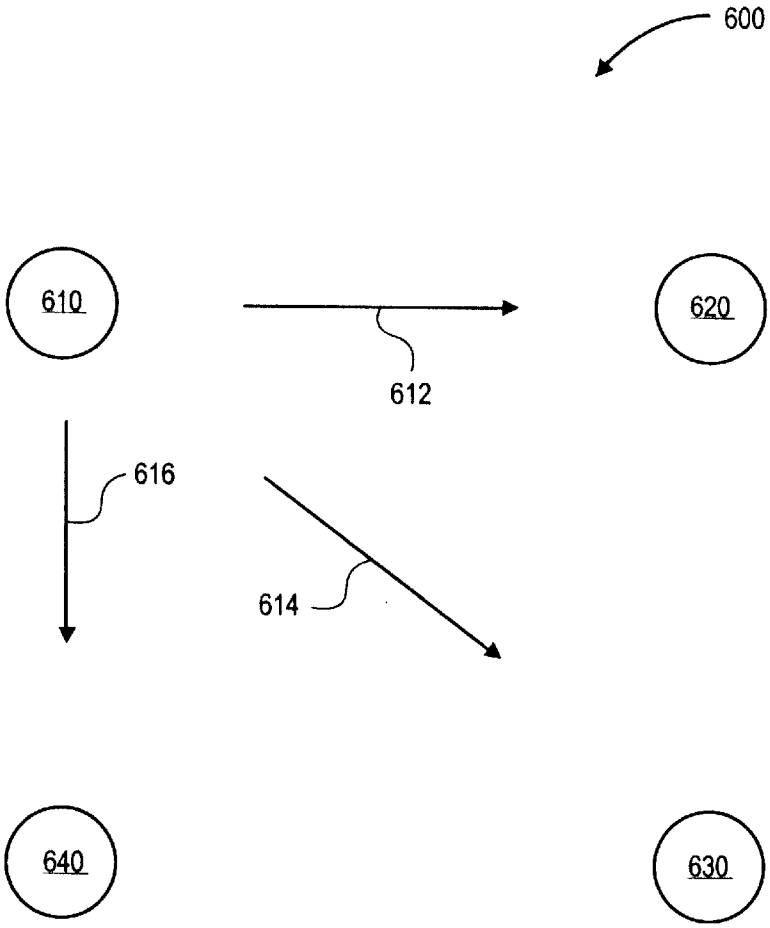

FIG. 6

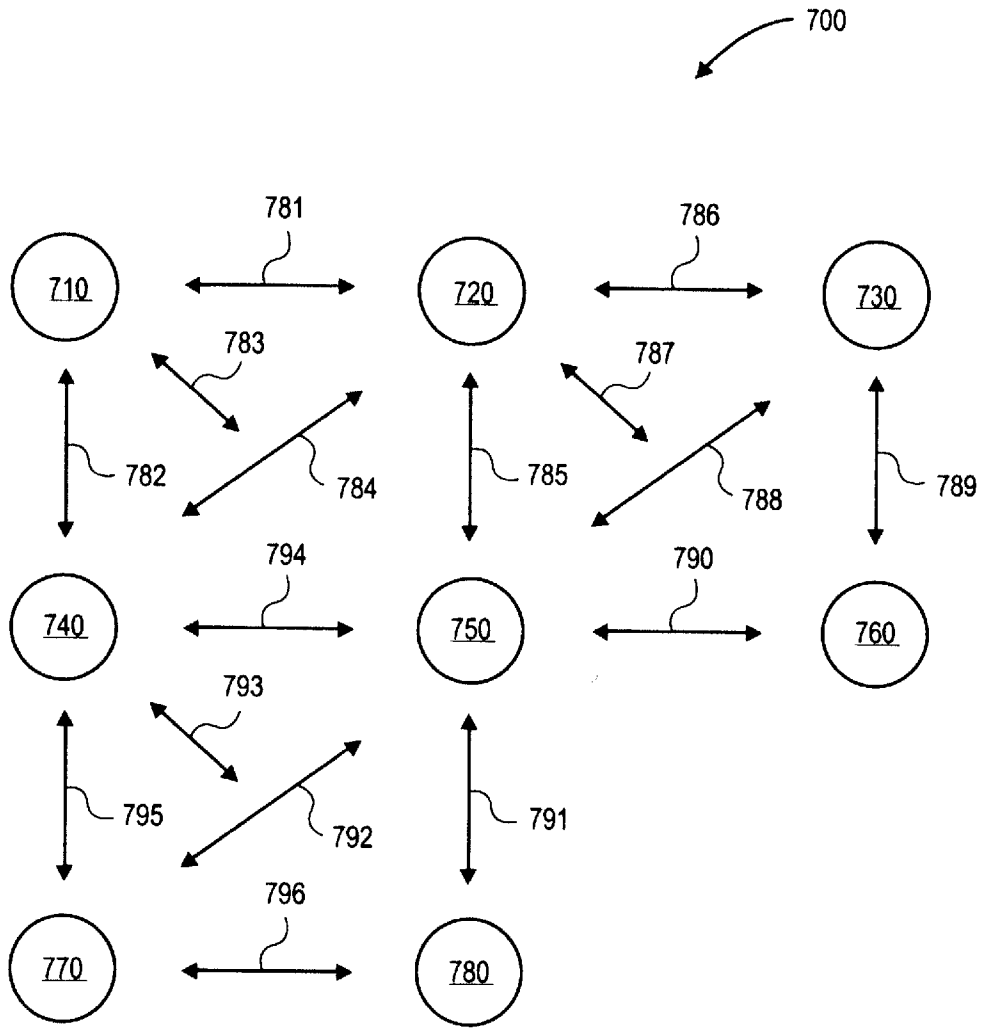
700
781
786
710
720
730
783
787
782
784
785
788
789
794
790
740
750
760
793
795
792
791
796
770
780

FIG. 7

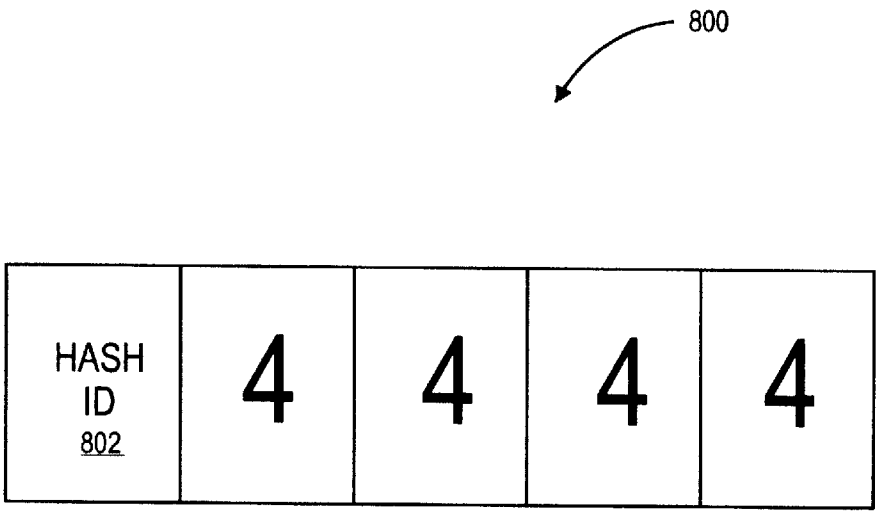

FIG. 8

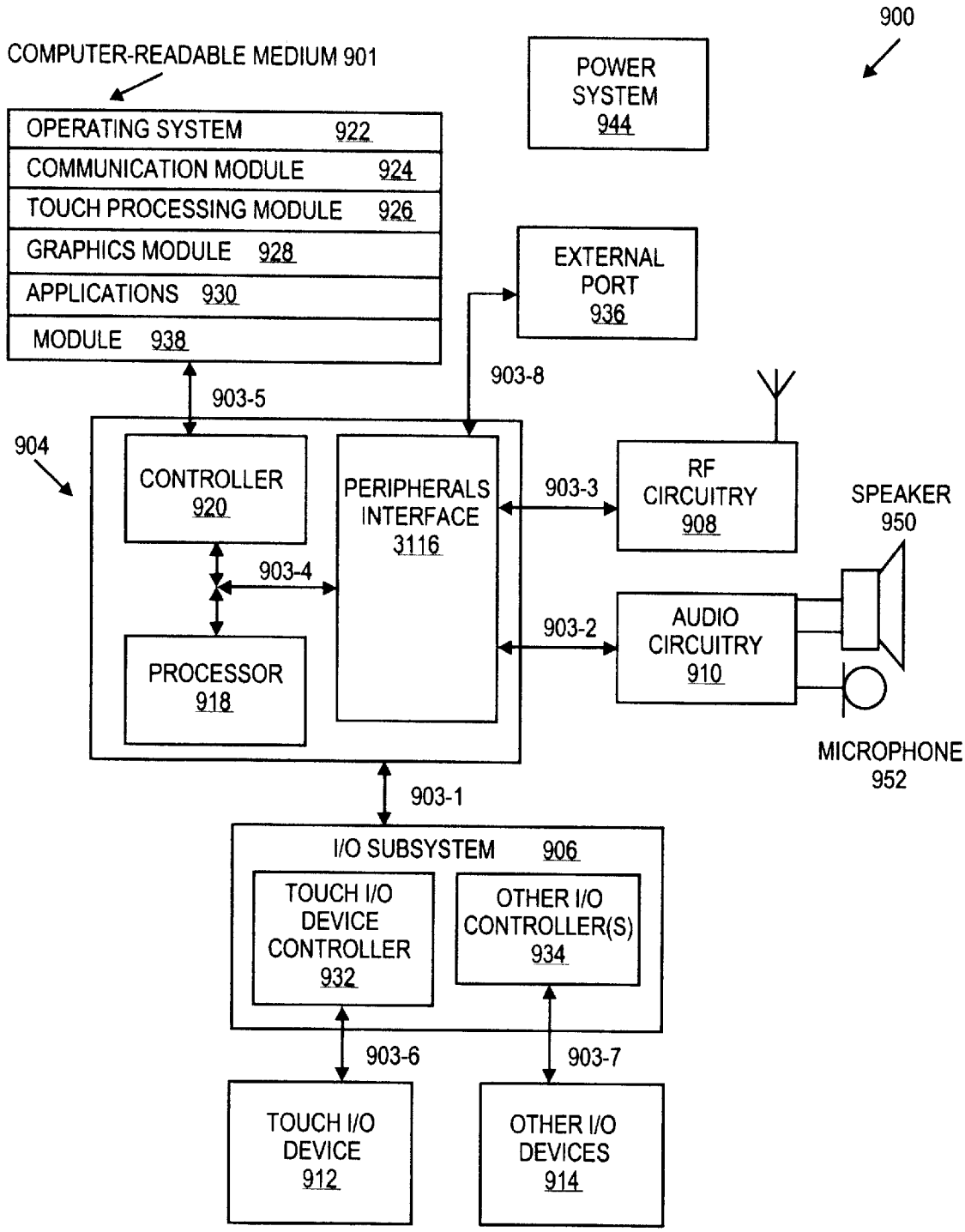
900
COMPUTER-READABLE MEDIUM 901
OPERATING SYSTEM 922
COMMUNICATION MODULE 924
TOUCH PROCESSING MODULE 926
GRAPHICS MODULE 928
APPLICATIONS 930
MODULE 938
POWER SYSTEM 944
EXTERNAL PORT 936
903-8
903-5
904
CONTROLLER 920
PERIPHERALS INTERFACE 3116
903-3
RF CIRCUITRY 908
SPEAKER 950
903-4
903-2
AUDIO CIRCUITRY 910
PROCESSOR 918
MICROPHONE 952
903-1
I/O SUBSYSTEM 906
TOUCH I/O DEVICE CONTROLLER 932
OTHER I/O CONTROLLER(S) 934
903-6
903-7
TOUCH I/O DEVICE 912
OTHER I/O DEVICES 914

FIG. 9

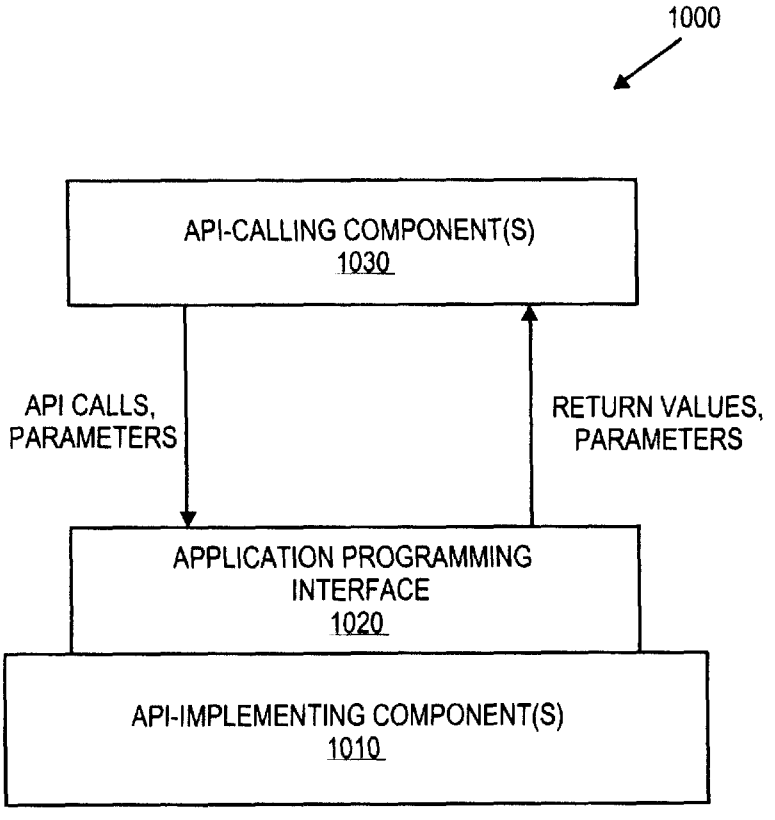
1000
API-CALLING COMPONENT(S)
1030
API CALLS,
PARAMETERS
RETURN VALUES,
PARAMETERS
APPLICATION PROGRAMMING
INTERFACE
1020
API-IMPLEMENTING COMPONENT(S)
1010

FIG. 10

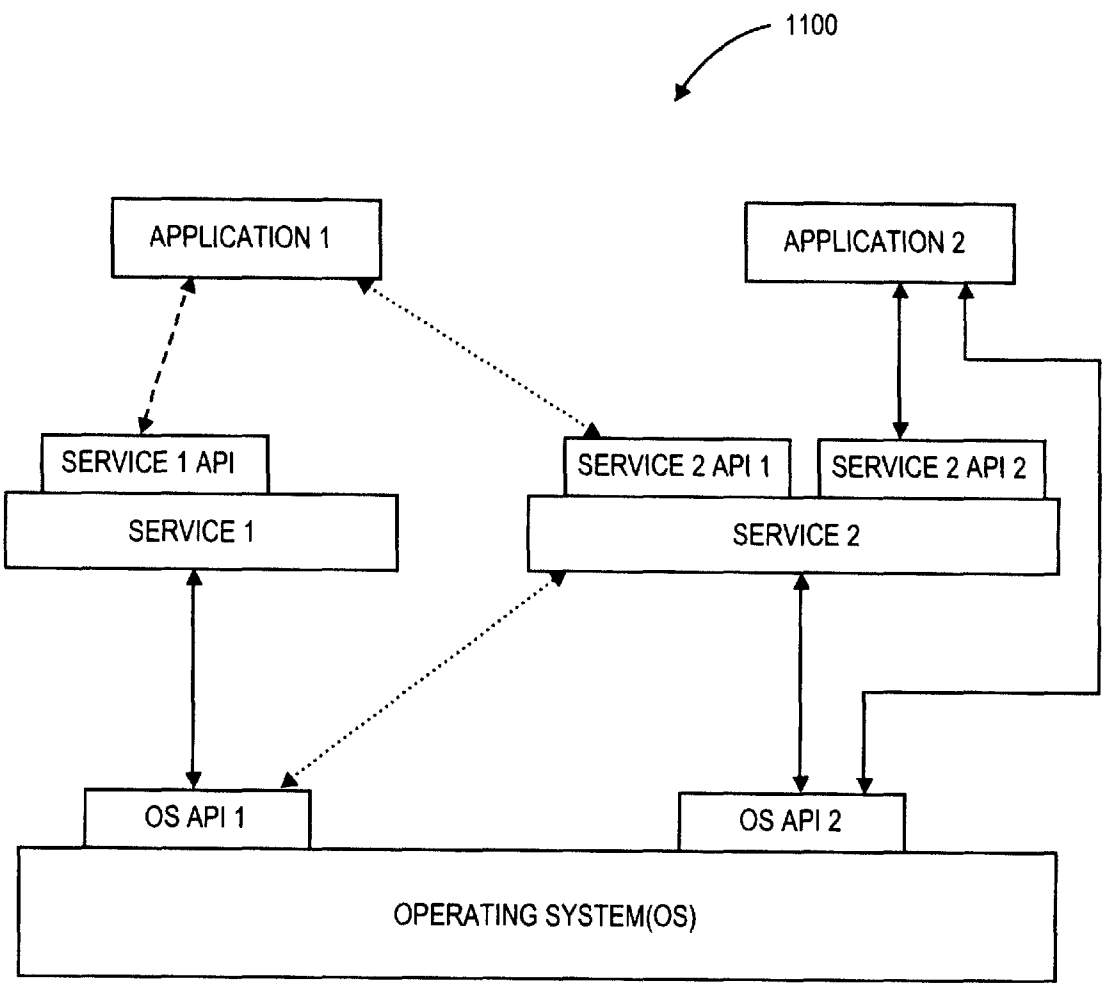
1100
APPLICATION 1
APPLICATION 2
SERVICE 1 API
SERVICE 1
SERVICE 2 API 1
SERVICE 2 API 2
SERVICE 2
OS API 1
OS API 2
OPERATING SYSTEM(OS)

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2011/038620

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, COMPENDEX, INSPEC, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Daal, K: "Link.m - punchball - Open-source iPhone arcade game revision r3", Google Project Hosting , 3 December 2009 (2009-12-03), pages 1-5, XP000002655627, Retrieved from the Internet: URL:http://code.google.com/p/punchball/source/browse/trunk/Classes/Link.m [retrieved on 2011-08-03] the whole document ----- -/--	1-25

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 August 2011	16/08/2011

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Eraso Helguera, J

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2011/038620

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Apple Inc: "TankViewController.m", iOS developer library , 8 June 2009 (2009-06-08), XP000002655628, Retrieved from the Internet: URL:http://developer.apple.com/LIBRARY/IOS/samplecode/GKTank/GKTank.zip [retrieved on 2011-08-03] the whole document	1-25
A	Apple, Inc: "Game Kit Framework Reference", , 26 May 2009 (2009-05-26), XP000002655629, Retrieved from the Internet: URL:http://pooh.poly.asu.edu/Cst194/Resources/Docs/system.framework.GameKit_Collection.pdf [retrieved on 2011-08-03] the whole document	1-25
A	US 2008/198850 A1 (COOPER ERIC [CA] ET AL) 21 August 2008 (2008-08-21) paragraph [0028]	1-25
A	KNUTSSON B ET AL: "Peer-to-peer support for massively multiplayer games", INFOCOM 2004. TWENTY-THIRD ANNUALJOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, IEEE, PISCATAWAY, NJ, USA, vol. 1, 7 March 2004 (2004-03-07), pages 96-107, XP010740794, DOI: 10.1109/INFCOM.2004.1354485 ISBN: 978-0-7803-8355-5 the whole document	1-25
A	JIANG ET AL: "An approach to achieve scalability through a structured peer-to-peer network for massively multiplayer online role playing games", COMPUTER COMMUNICATIONS, ELSEVIER SCIENCE PUBLISHERS BV, AMSTERDAM, NL, vol. 30, no. 16, 13 October 2007 (2007-10-13), pages 3075-3084, XP022297346, ISSN: 0140-3664, DOI: 10.1016/J.COMCOM.2007.05.052 page 3076, right-hand column	1-25

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2011/038620

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008198850 A1	21-08-2008	CN 101309301 A	19-11-2008
		DE 102008010145 A1	28-08-2008
		GB 2446951 A	27-08-2008
		JP 2008206160 A	04-09-2008
		KR 20080077915 A	26-08-2008