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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 March 2012 (22.03.2012)

PCT

(10) International Publication Number
WO 2012/034237 A1

(51) International Patent Classification:
H04L 12/26 (2006.01) *G06Q 40/00* (2012.01)
G06Q 20/00 (2012.01) *H04L 12/16* (2006.01)
G06Q 30/00 (2012.01)

(21) International Application Number:
PCT/CA2011/050569

(22) International Filing Date:
16 September 2011 (16.09.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/383,583 16 September 2010 (16.09.2010) US

(72) Inventors; and
(71) Applicants : **CHRAPKO, Evan V** [CA/CA]; #128, 14-9977-178 Street, Edmonton, Alberta T5T 6J6 (CA). **CHAN, Leo M.** [CA/CA]; #30, 4755 Terwillegar Common, Edmonton, Alberta T6R 3V6 (CA).

(74) Agent: **SMART & BIGGAR**; P.O. Box 2999, Station D, 900 - 55 Metcalfe Street, Ottawa, Ontario K1P 5Y6 (CA).

(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, QA, RO, RS, RU, RW, 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))*

(54) Title: SYSTEMS AND METHODS FOR PROVIDING VIRTUAL CURRENCIES

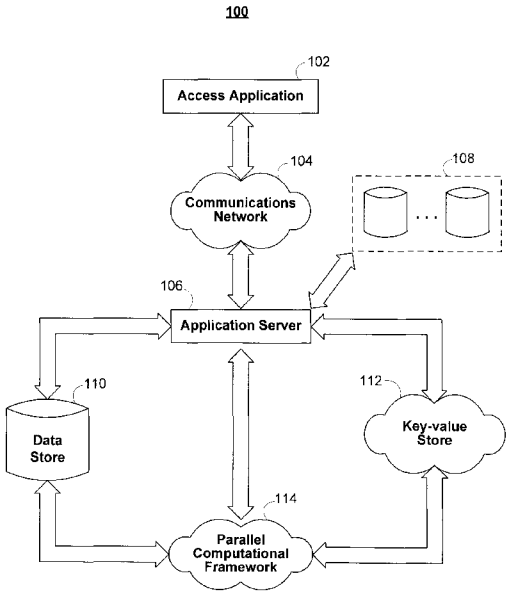

FIG. 1

(57) Abstract: Systems and methods for conducting reliable financial transactions, credit decisions, and security assessments are provided. A user may assign user connectivity values to other members of the community, or connectivity values may be automatically harvested or assigned from third parties or based on the frequency of interactions between members of the community. Connectivity values may represent such factors as alignment, reputation within the network community, or the degree of trust. Information about a financial transaction initiated by a first member of the community, a credit decision, and/or a security assessment may be automatically published to other qualifying members of the community based on connectivity values. The other qualifying members may then be given the opportunity to participate in the same financial transaction or access the same financial application in order to initiate their own financial transaction, or to take action based on information about the financial transaction, credit decision, and/or security assessment. These transactions may also be based on virtual and/or electronic currencies.

WO 2012/034237 A1

SYSTEMS AND METHODS FOR PROVIDING VIRTUAL CURRENCIES

Cross-reference to Related Applications

[0001] This application claims the benefit of U.S. Provisional Application No. 61/383,583, filed on September 16, 2010, which is hereby incorporated by reference herein in its entirety.

Background of the Invention

[0002] This invention relates generally to networks of individuals, entities, or both, and network communities and, more particularly, to systems and methods for determining trust scores or connectivity within or between individuals, entities, or both, or networks of individuals, entities, or both, and using these scores to facilitate financial transactions.

[0003] The connectivity, or relationships, of an individual or entity within a network community may be used to infer attributes of that individual or entity. For example, an individual or entity's connectivity within a network community may be used to determine the identity of the individual or entity (e.g., used to make decisions about identity claims and authentication), the trustworthiness or reputation of the individual, or any combination of the membership, status, and/or influence of that individual in a particular community or subset of a particular community.

[0004] An individual or entity's connectivity within a network community, however, is difficult to quantify. For example, network communities may include hundreds, thousands, millions, billions or more members. Each member may possess varying degrees of connectivity information about itself and possibly about other members of the community. Some of this information may be highly credible or objective, while other information may be less credible and subjective. In addition, connectivity information from community members may come in various forms and on various scales, making it difficult to meaningfully compare one member's "trustworthiness" or "competence" and connectivity information with another member's "trustworthiness" or "competence" and connectivity information. Also, many individuals may belong to multiple communities, further complicating the determination of a

quantifiable representation of trust and connectivity within a network community. Similarly, a particular individual may be associated with duplicate entries in one or more communities, due to, for example, errors in personal information such as name/information misspellings and/or outdated personal information. Even if a quantifiable representation of an individual's connectivity is determined, it is often difficult to use this representation in a meaningful way to make real-world decisions about the individual (e.g., whether or not to trust the individual). In some embodiments, virtual and/or electronic currency systems based on network connectivity and/or trust values may be used to facilitate transactions related to such decisions.

[0005] Further, it may be useful for these real-world decisions to be made prospectively (i.e., in advance of an anticipated event). Such prospective analysis may be difficult as an individual or entity's connectivity within a network community may change rapidly as the connections between the individual or entity and others in the network community may change quantitatively or qualitatively. This analysis becomes increasingly complex as if applied across multiple communities.

Summary of the Invention

[0006] In view of the foregoing, systems and methods are provided for determining the connectivity between nodes within a network community and inferring attributes, such as trustworthiness or competence, from the connectivity. Connectivity may be determined, at least in part, using various graph traversal and normalization techniques described in more detail below and in U.S. Provisional Patent Application Nos. 61/247,343, filed September 30, 2009, and 61/254,313, filed October 23, 2009, 61/294,949, filed January 14, 2010, 61/310,844, filed March 5, 2010, 61/329,899, filed April 30, 2010, and 61/383,583, filed September 16, 2010, and in International Patent Application Nos. CA2010001531, filed September 30, 2010, CA2010001658, filed October 22, 2010, CA2011050017, filed January 14, 2011, CA2011050125 filed March 3, 2011, and CA2011050260, each of which are hereby incorporated by reference herein in their entireties.

[0007] In an embodiment, a path counting approach may be used where processing circuitry is configured to count the number of paths between a first node n_1 and a second node n_2 within a network community. A connectivity rating R_{n1n2} may then be assigned to the nodes. The assigned connectivity rating may be proportional to the number of subpaths, or

relationships, connecting the two nodes, among other possible measures. Using the number of subpaths as a measure, a path with one or more intermediate nodes between the first node n_1 and the second node n_2 may be scaled by an appropriate number (e.g., the number of intermediate nodes) and this scaled number may be used to calculate the connectivity rating.

[0008] In some embodiments, weighted links are used in addition or as an alternative to the subpath counting approach. Processing circuitry may be configured to assign a relative user weight to each path connecting a first node n_1 and a second node n_2 within a network community. A user connectivity value may be assigned to each link. For example, a user or entity associated with node n_1 may assign user connectivity values for all outgoing paths from node n_1. In some embodiments, the connectivity values assigned by the user or entity may be indicative of that user or entity's trust in the user or entity associated with node n_2. The link values assigned by a particular user or entity may then be compared to each other to determine a relative user weight for each link.

[0009] The relative user weight for each link may be determined by first computing the average of all the user connectivity values assigned by that user or node (i.e., the out-link values). If t_i is the user connectivity value assigned to link i, then the relative user weight, w_i, assigned to that link may be given in accordance with:

$$w_i = 1 + (t_i - \bar{t}_i)^2 \tag{1}$$

In some embodiments, an alternative relative user weight, w_i', may be used based on the number of standard deviations, σ, the user connectivity value differs from the average value assigned by that user or node. For example, the alternative relative user weight may be given in accordance with:

$$w_i' = 1 - \frac{1}{2 + k^2} \text{ where } k = \left\{ \frac{0, \text{if } \sigma = 0}{\frac{t_i - \bar{t}_i}{\sigma}, \text{otherwise}} \right\} \tag{2}$$

[0010] To determine the overall weight of a path, in some embodiments, the weights of all the links along the path may be multiplied together. The overall path weight may then be given in accordance with:

$$w_{path} = \Pi(w_i) \tag{3}$$

or

$$w_{path} = \Pi(w_i') \tag{4}$$

The connectivity value for the path may then be defined as the minimum user connectivity value of all the links in the path multiplied by the overall path weight in accordance with:

$$t_{path} = w_{path} \times t_{\min} \tag{5}$$

[0011] In some embodiments, the connectivity or trust rating between two nodes may be based on connectivity statistics values for one of the nodes. The connectivity rating or trust rating a first node has for a second node may be based on a connectivity between the first node and the second node and one or more connectivity statistics associated with the first node.

[0012] In other embodiments, only "qualified" paths may be used to determine connectivity values. A qualified path may be a path whose path weight meets any suitable predefined or dynamic criteria. For example, a qualified path may be a path whose path weight is greater than or equal to some threshold value. As described in more detail below, any suitable threshold function may be used to define threshold values. The threshold function may be based, at least in some embodiments, on empirical data, desired path keep percentages, or both. In some embodiments, threshold values may depend on the length, *l*, of the path. For example, an illustrative threshold function specifying the minimum path weight for path *p* may be given in accordance with:

$$threshold(p) = \begin{Bmatrix} 0.5, \, if \; l = 1 \\ 0.428, \, if \; l = 2 \\ 0.289, \, if \; l = 3 \\ 0.220, \, if \; l = 4 \\ 0.216, \, if \; l = 5 \\ 0.192, \, if \; l = 6 \end{Bmatrix} \tag{6}$$

[0013] To determine path connectivity values, in some embodiments, a parallel computational framework or distributed computational framework (or both) may be used. For example, in one embodiment, a number of core processors implement an Apache Hadoop or Google MapReduce cluster. This cluster may perform some or all of the distributed computations in connection with determining new path link values and path weights. In some embodiments, the parallel computational framework or distributed computational framework may include a distributed graph storage/computation system. The distributed graph

storage/computation system may include a cluster registry, one or more node storage clusters, and one or more edge storage clusters. In some embodiments, the cluster registry, node storage cluster(s), and/or the edge storage cluster(s) each include a plurality of devices, computers, or processors. The distributed graph storage/computation system may be configured to store node and edge elements of one or more graphs representative of one or more network communities in a distributed fashion. In some embodiments, calculations and computations for determining connectivity information may be performed in a distributed fashion across the processors in the distributed graph storage/computation system.

[0014] The processing circuitry may identify a changed node within a network community. For example, a new outgoing link may be added, a link may be removed, or a user connectivity value may have been changed. In response to identifying a changed node, in some embodiments, the processing circuitry may re-compute link, path, weight, connectivity, and/or connectivity statistics values associated with some or all nodes in the implicated network community or communities.

[0015] In some embodiments, only values associated with affected nodes in the network community are recomputed after a changed node is identified. If there exists at least one changed node in the network community, the changed node or nodes may first undergo a prepare process. The prepare process may include a "map" phase and "reduce" phase. In the map phase of the prepare process, the prepare process may be divided into smaller sub-processes which are then distributed to a core in the parallel computational framework cluster. For example, in one embodiment, each node or link change (e.g., tail to out-link change and head to in-link change) may be mapped to a different core for parallel computation. In the reduce phase of the prepare process, each out-link's weight may be determined in accordance with equation (1). Each of the out-link weights may then be normalized by the sum of the out-link weights (or any other suitable value). The node table may then be updated for each changed node, its in-links, and its out-links.

[0016] After the changed nodes have been prepared, the paths originating from each changed node may be calculated. Once again, a "map" and "reduce" phase of this process may be defined. During this process, in some embodiments, a depth-first search may be performed of the node digraph or node tree. All affected ancestor nodes may then be identified and their paths recalculated.

[0017] In some embodiments, to improve performance, paths may be grouped by the last node in the path. For example, all paths ending with node n_1 may be grouped together, all paths ending with node n_2 may be grouped together, and so on. These path groups may then be stored separately (e.g., in different columns of a single database table). In some embodiments, the path groups may be stored in columns of a key-value store implementing an HBase cluster (or any other compressed, high performance database system, such as BigTable).

[0018] In some embodiments, one or more threshold functions may be defined. The threshold function or functions may be used to determine the maximum number of links in a path that will be analyzed in a connectivity determination or connectivity computation. Threshold factors may also be defined for minimum link weights, path weights, or both. Weights falling below a user-defined or system-defined threshold (or above a maximum threshold) may be ignored in a connectivity determination or connectivity computation, while only weights of sufficient magnitude may be considered.

[0019] In some embodiments, a user connectivity or trust value may represent the degree of trust between a first node and a second node. In one embodiment, node n_1 may assign a user connectivity value of l_1 to a link between it and node n_2. Node n_2 may also assign a user connectivity value of l_2 to a reverse link between it and node n_1. The values of l_1 and l_2 may be at least partially subjective indications of the trustworthiness of the individual or entity associated with the node connected by the link. For example, one or more of the individual's or entity's reputation within the network community (or some other community), the individual's or entity's alignment with the trusting party (e.g., political, social, or religious alignment), past dealings with the individual or entity, and the individual's or entity's character and integrity (or any other relevant considerations) may be used to determine a partially subjective user connectivity value indicative of trust. A user (or other individual authorized by the node) may then assign this value to an outgoing link connecting the node to the individual or entity. Objective measures (e.g., data from third-party ratings agencies or credit bureaus) may also be used, in some embodiments, to form composite user connectivity values indicative of trust. The subjective, objective, or both types of measures may be automatically harvested or manually inputted for analysis.

[0020] In other embodiments, the user connectivity or trust value may be calculated objectively. In one embodiment, the trust value of a first node for a second node may be

calculated based on the number of paths linking the two nodes, one or more path scores associated with the linking paths, connectivity statistics and/or other connectivity information associated with the first node.

[0021] In some embodiments, a decision-making algorithm may access the connectivity values in order to make automatic decisions (e.g., automatic network-based decisions, such as authentication or identity requests) on behalf of a user. Connectivity values may additionally or alternatively be outputted to external systems and processes located at third-parties. The external systems and processes may be configured to automatically initiate a transaction (or take some particular course of action) based, at least in part, on received connectivity values. For example, electronic or online advertising may be targeted to subgroups of members of a network community based, at least in part, on network connectivity values.

[0022] As another example, the decision-making algorithm may take the form of a financial application, such as a loan, lending, or donation application. Connectivity values may be used by financial institutions to make automatic credit-granting decisions. In some embodiments, connectivity values may be used in conjunction with third-party ratings agency information (e.g., credit bureau ratings information) in order to make credit-granting decisions. Connectivity values may also be used to advertise, promote, or publish information about charitable gifts, donations, or loans to other parties in a social networking environment or other network-based community. Decisions regarding loan amounts, interests rates, and/or loan repayment schedules may be automatically generated after a loan is approved and accepted by the financial application, the lender, or both the lender and financial application. In some embodiments, virtual and/or electronic currency systems based on network connectivity and/or trust values may be used to facilitate transactions related to such decisions.

[0023] In some embodiments, a decision-making algorithm may access connectivity values to make decisions prospectively (e.g., before an anticipated event like a request for credit). Such decisions may be made at the request of a user, or as part of an automated process (e.g., a credit bureau's periodic automated analysis of a database of customer information). This prospective analysis may allow for the initiation of a transaction (or taking of some particular action) in a fluid and/or dynamic manner.

[0024] In some embodiments, connectivity values may be used to present information to the user. This information may include, but is not limited to, static and/or interactive visualizations of connectivity values within a user's associated network community or communities. In some embodiments, this information may allow the user to explore or interact with an associated network community or communities, and encourage and/or discourage particular interactions within a user's associated network community or communities. In some embodiments, this information may explicitly present the user with the connectivity values. For example, a percentage may indicate how trustworthy another individual and/or entity is to a user. In some embodiments, the information may implicitly present the user with a representation of the connectivity values. For example, an avatar representing another individual and/or entity may change in appearance based on how trustworthy that individual and/or entity is to a user.

Brief Description of the Drawings

[0025] The above and other features of the present invention, its nature and various advantages will be more apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, and in which:

[0026] FIG. 1 is an illustrative block diagram of a network architecture used to support connectivity within a network community in accordance with one embodiment of the invention;

[0027] FIG. 2 is another illustrative block diagram of a network architecture used to support connectivity within a network community in accordance with one embodiment of the invention;

[0028] FIG. 3 is an illustrative diagram of a distributed storage/computation network in accordance with one embodiment of the invention;

[0029] FIGS. 4A-C show illustrative data tables for graph information storage in a distributed storage/computation network in accordance with one embodiment of the invention;

[0030] FIGS. 5A, 5B, and 5C show illustrative data tables for supporting connectivity determinations within a network community in accordance with one embodiment of the invention;

[0031] FIGS. 6A-6H show illustrative processes for supporting connectivity determinations within a network community in accordance with one embodiment of the invention;

[0032] FIG. 7 shows an illustrative process for querying all paths to a target node and computing a network connectivity value in accordance with one embodiment of the invention;

[0033] FIG. 8 shows an illustrative process for determining a connectivity or trust score of one node for another node based on connectivity statistics, in accordance with one embodiment of the invention;

[0034] FIG. 9 shows an illustrative process for supporting user sign-in profiles in accordance with one embodiment of the invention; and

[0035] FIG. 10 shows an illustrative process for facilitating financial transactions in accordance with one embodiment of the invention.

Detailed Description

[0036] Systems and methods for determining the connectivity between nodes in a network community are provided. As defined herein, a "node" may include any user terminal, network device, computer, mobile device, access point, robot, or any other electronic device capable of being uniquely identified within a network community. For example, nodes may include robots (or other machines) assigned unique serial numbers or network devices assigned unique network addresses. In some embodiments, a node may also represent an individual human being, entity (e.g., a legal entity, such as a public or private company, corporation, limited liability company (LLC), partnership, sole proprietorship, or charitable organization), concept (e.g., a social networking group), service, animal, city/town/village, parcel of land (which may be identified by land descriptions), or inanimate object (e.g., a car, aircraft, or tool). As also defined herein, a "network community" may include a collection of nodes and may represent any group of devices, individuals, or entities.

[0037] For example, all or some subset of the users of a social networking website or social networking service (or any other type of website or service, such as an online gaming community) may make up a single network community. Each user may be represented by a node in the network community. As another example, all the subscribers to a particular newsgroup or distribution list may make up a single network community, where each individual subscriber may be represented by a node in the network community. Any

particular node may belong in zero, one, or more than one network community, or a node may be banned from all, or a subset of, the community. To facilitate network community additions, deletions, and link changes, in some embodiments a network community may be represented by a directed graph, or digraph, weighted digraph, tree, or any other suitable data structure.

[0038] FIG. 1 shows illustrative network architecture 100 used to support the connectivity determinations within a network community. A user may utilize access application 102 to access application server 106 over communications network 104. For example, access application 102 may include a standard web browser, application server 106 may include a web server, and communication network 106 may include the Internet. Access application 102 may also include proprietary applications specifically developed for one or more platforms or devices. For example, access application 102 may include one or more instances of an Apple iOS, Android, WebOS, or any suitable application for use in accessing application 106 over communications network 104. Multiple users may access application service 106 via one or more instances of access application 102. For example, a plurality of mobile devices may each have an instance of access application 102 running locally on the devices. One or more users may use an instance of access application 102 to interact with application server 106.

[0039] Communication network 104 may include any wired or wireless network, such as the Internet, WiMax, wide area cellular, or local area wireless network. Communication network 104 may also include personal area networks, such as Bluetooth and infrared networks. Communications on communications network 104 may be encrypted or otherwise secured using any suitable security or encryption protocol.

[0040] Application server 106, which may include any network server or virtual server, such as a file or web server, may access data sources 108 locally or over any suitable network connection. Application server 106 may also include processing circuitry (e.g., one or more microprocessors), memory (e.g., RAM, ROM, and hybrid types of memory), storage devices (e.g., hard drives, optical drives, and tape drives). The processing circuitry included in application server 106 may execute a server process for supporting the network connectivity determinations of the present invention, while access application 102 executes a corresponding client process. The processing circuitry included in application server 106 may also perform any of the calculations and computations described herein in connection with

determining network connectivity. In some embodiments, a computer-readable medium with computer program logic recorded thereon is included within application server 106. The computer program logic may determine the connectivity between two or more nodes in a network community and it may or may not output such connectivity to a display screen or data store.

[0041] For example, application server 106 may access data sources 108 over the Internet, a secured private LAN, or any other communications network. Data sources 108 may include one or more third-party data sources, such as data from third-party social networking services, third-party ratings bureaus, and document issuers (e.g., driver's license and license plate issuers, such as the Department of Motor Vehicles). For example, data sources 108 may include user and relationship data (e.g., "friend" or “follower” data) from one or more of Facebook, MySpace, openSocial, Friendster, Bebo, hi5, Orkut, PerfSpot, Yahoo! 360, Gmail, Yahoo! Mail, Hotmail, or other email-based services and accounts, LinkedIn, Twitter, Google+, Really Simple Syndication readers, or any other social networking website or service. Data sources 108 may also include data stores and databases local to application server 106 containing relationship information about users accessing application server 106 via access application 102 (e.g., databases of addresses, legal records, transportation passenger lists, gambling patterns, political affiliations, vehicle license plate or identification numbers, universal product codes, news articles, business listings, and hospital or university affiliations).

[0042] Application server 106 may be in communication with one or more of data store 110, key-value store 112, and parallel computational framework 114. Data store 110, which may include any relational database management system (RDBMS), file server, or storage system, may store information relating to one or more network communities. For example, one or more of data tables 500 (FIG. 5A) may be stored on data store 110. Data store 110 may store identity information about users and entities in the network community, an identification of the nodes in the network community, user link and path weights, user configuration settings, system configuration settings, and/or any other suitable information. There may be one instance of data store 110 per network community, or data store 110 may store information relating to a plural number of network communities. For example, data store 110 may include one database per network community, or one database may store information about all available network communities (e.g., information about one network

community per database table). In some embodiments, the parallel computational framework 114 may include a distributed storage/computation network, described below in relation to FIG. 3.

[0043] Parallel computational framework 114, which may include any parallel or distributed computational framework or cluster, may be configured to divide computational jobs into smaller jobs to be performed simultaneously, in a distributed fashion, or both. For example, parallel computational framework 114 may support data-intensive distributed applications by implementing a map/reduce computational paradigm where the applications may be divided into a plurality of small fragments of work, each of which may be executed or re-executed on any core processor in a cluster of cores. A suitable example of parallel computational framework 114 includes an Apache Hadoop cluster.

[0044] Parallel computational framework 114 may interface with key-value store 112, which also may take the form of a cluster of cores. Key-value store 112 may hold sets of key-value pairs for use with the map/reduce computational paradigm implemented by parallel computational framework 114. For example, parallel computational framework 114 may express a large distributed computation as a sequence of distributed operations on data sets of key-value pairs. User-defined map/reduce jobs may be executed across a plurality of nodes in the cluster. The processing and computations described herein may be performed, at least in part, by any type of processor or combination of processors. For example, various types of quantum processors (e.g., solid-state quantum processors and light-based quantum processors), artificial neural networks, and the like may be used to perform massively parallel computing and processing.

[0045] In some embodiments, parallel computational framework 114 may support two distinct phases, a "map" phase and a "reduce" phase. The input to the computation may include a data set of key-value pairs stored at key-value store 112. In the map phase, parallel computational framework 114 may split, or divide, the input data set into a large number of fragments and assign each fragment to a map task. Parallel computational framework 114 may also distribute the map tasks across the cluster of nodes on which it operates. Each map task may consume key-value pairs from its assigned fragment and produce a set of intermediate key-value pairs. For each input key-value pair, the map task may invoke a user defined map function that transmutes the input into a different key-value pair. Following the map phase, parallel computational framework 114 may sort the intermediate data set by key

and produce a collection of tuples so that all the values associated with a particular key appear together. Parallel computational framework 114 may also partition the collection of tuples into a number of fragments equal to the number of reduce tasks.

[0046] In the reduce phase, each reduce task may consume the fragment of tuples assigned to it. For each such tuple, the reduce task may invoke a user-defined reduce function that transmutes the tuple into an output key-value pair. Parallel computational framework 114 may then distribute the many reduce tasks across the cluster of nodes and provide the appropriate fragment of intermediate data to each reduce task.

[0047] Tasks in each phase may be executed in a fault-tolerant manner, so that if one or more nodes fail during a computation the tasks assigned to such failed nodes may be redistributed across the remaining nodes. This behavior may allow for load balancing and for failed tasks to be re-executed with low runtime overhead.

[0048] Key-value store 112 may implement any distributed file system capable of storing large files reliably. For example key-value store 112 may implement Hadoop's own distributed file system (DFS) or a more scalable column-oriented distributed database, such as HBase. Such file systems or databases may include BigTable-like capabilities, such as support for an arbitrary number of table columns.

[0049] Although FIG. 1, in order to not over-complicate the drawing, only shows a single instance of access application 102, communications network 104, application server 106, data source 108, data store 110, key-value store 112, and parallel computational framework 114, in practice network architecture 100 may include multiple instances of one or more of the foregoing components. In addition, key-value store 112 and parallel computational framework 114 may also be removed, in some embodiments. As shown in network architecture 200 of FIG. 2, the parallel or distributed computations carried out by key-value store 112 and/or parallel computational framework 114 may be additionally or alternatively performed by a cluster of mobile devices 202 instead of stationary cores. In some embodiments, cluster of mobile devices 202, key-value store 112, and parallel computational framework 114 are all present in the network architecture. Certain application processes and computations may be performed by cluster of mobile devices 202 and certain other application processes and computations may be performed by key-value store 112 and parallel computational framework 114. In addition, in some embodiments, communication network 104 itself may perform some or all of the application processes and computations.

For example, specially-configured routers or satellites may include processing circuitry adapted to carry out some or all of the application processes and computations described herein.

[0050] Cluster of mobile devices 202 may include one or more mobile devices, such as PDAs, cellular telephones, mobile computers, or any other mobile computing device. Cluster of mobile devices 202 may also include any appliance (e.g., audio/video systems, microwaves, refrigerators, food processors) containing a microprocessor (e.g., with spare processing time), storage, or both. Application server 106 may instruct devices within cluster of mobile devices 202 to perform computation, storage, or both in a similar fashion as would have been distributed to multiple fixed cores by parallel computational framework 114 and the map/reduce computational paradigm. Each device in cluster of mobile devices 202 may perform a discrete computational job, storage job, or both. Application server 106 may combine the results of each distributed job and return a final result of the computation.

[0051] FIG. 3 is an illustrative diagram of a distributed storage/computation network 300 in accordance with one embodiment of the invention. The distributed network 300 may be used to store information about one or more network communities. In some embodiments, network community information may be stored in the distributed network 300 in the form of one or more graphs. The distributed network 300 may include a plurality of computers, processors, or devices, each of which may communicate with other computers in the network via a communications network such as a local area network, a wide area network, the Internet, any other suitable wired or wireless communications network, or any combination thereof. In some embodiments, the computers in the distributed network 300 may be grouped into one or more clusters, each with a unique cluster ID. In one embodiment, the computers in the distributed network 300 may be grouped into at least three clusters: a cluster registry 302, a node storage cluster 304, and an edge storage cluster 306. Each cluster may include one or more computers, processors, or devices, and in some embodiments, individual computers may be able to dynamically move between different clusters. For example, clusters may be scalable. Individual computers may also be able to leave or join the distributed network 300. For example, computers may be added to the distributed network 300 in order to increase storage and/or computing capacity. In some embodiments, each cluster may provide one or more services to one or more requesters, such as other computers or clusters in the distributed network 300, or a remote user or system.

[0052] In some embodiments, a cluster registry 302 may store information about all of the clusters in the distributed network 300 and/or all of the computers in the distributed network 300. In some embodiments, cluster registry may store information about any suitable subset of clusters in the distributed network 300 (*e.g.*, any suitable one or more of such clusters). In some embodiments, the distributed network 300 may include only one cluster registry, but in other embodiments, the distributed network 300 may include two or more cluster registries. The information stored in the cluster registry 302 may also be cached on one or more other computers in the distributed network 300. For example, in one embodiment, every other computer in the distributed network 300 may cache the information stored in the cluster registry.

[0053] The cluster registry 302 may provide various services to requesters. Requesters may include other clusters or computers in the distributed network 300, or remote/external users and systems. Illustrative services that the cluster registry 302 may provide may include any combination of the following:

[0054] **List all clusters** - the cluster registry 302 provides a list of all of the clusters in the distributed network 300.

[0055] **List all members of a cluster** – the cluster registry 302 provides a list of computers, processors, or devices in a given cluster. This service may require a cluster ID to identify the given cluster, and may return a list of the network addresses (e.g., IP addresses) of the computers, processors, or devices in the identified cluster.

[0056] **Create a cluster** – the cluster registry 302 creates a new cluster with a new, unique cluster ID. In some embodiments, the requester of this service may be able to specify the new cluster ID or the computers in the new cluster. In other embodiments, the cluster registry 302 may automatically assign the new cluster ID and/or automatically assign computers to the new cluster.

[0057] **Register/unregister a computer in a cluster** – since the cluster registry 302 keeps track of the particular computers in the different clusters, when a computer joins or leaves a cluster, it may notify the cluster registry 302, which then updates the computer/cluster registration information. In some embodiments, instead of waiting for a notification from the computer, the cluster registry 302 may periodically query the computers in the distributed network 300 to update computer registration information. Thus, if computers have unplanned outages or are disconnected from a cluster/the distributed network

300 without notifying the cluster registry 302, the cluster registry 302 is still able to maintain an accurate list of computers in the distributed network 300.

[0058] **Send notifications of changes to the registry** – when registry information changes, for example due to the creation of a new cluster or the registration/unregistration of a computer in a cluster, the cluster registry 302 may notify other computers that cache registry information in the distributed network 300 of the changes and/or update the registry information cached on those computers. The notification/update procedure may occur periodically or dynamically. For example, the cluster registry 302 may collect registry changes and provide notifications/updates every fraction of a second, second, fraction of a minute, or minute. In other embodiments, the cluster registry 302 may provide notifications/updates as soon as registry information is changed, to assure that the computers in the distributed network 300 cache the latest version of the registry information.

[0059] The cluster registry 302 may also be configured to provide other services. In some embodiments, the cluster registry may be implemented using an Apache Hadoop-derived ZooKeeper cluster.

[0060] Node storage cluster 304 and edge storage cluster 306 may store information about nodes and edges, respectively. In embodiments where the distributed network 300 includes multiple node storage clusters and/or multiple edge storage clusters, a particular node or edge in a graph representative of a network community (or information associated with the particular node or edge) may be stored on one particular node or edge storage cluster. In these embodiments, information about a particular node or edge may exist in a single storage cluster. Node/edge information may be stored in the form of data tables, described in more detail below with reference to FIGS. 4A-C.

[0061] In some embodiments, a database system that can be configured to run on computer clusters may be implemented on the storage clusters. For example, a storage cluster may use a PostgreSQL object-relational database management system. Each computer in a storage cluster may run both system software and database software in order to reduce network latency. The node storage cluster 304 and the edge storage cluster 306 may provide various services to requesters, which may include other clusters or computers in the distributed network 300, or remote/external users and systems. These services may be categorized as remote services, which may be implemented as remote procedure calls (RPCs) or Hypertext Transfer Protocol (HTTP) calls. In some embodiments, node storage cluster 304

may provide different remote services than edge storage cluster 306. In other embodiments, the requester may be the same computer the service is provided from, in which case the service is categorized as a local service. Local services may also vary according to type of storage cluster (node versus edge), or may be uniform across storage cluster type.

[0062] An example of a local service that may be uniform across storage cluster types is "**Pick a computer in a cluster**." This service allows a first computer to request the network address of a second computer in a cluster by providing a cluster ID. This local service may be used to distribute computational activity to all computers in a given cluster, so that processing load is distributed evenly across the computational resources available in a particular cluster. In some embodiments, the second computer may be selected via statistical techniques, round-robin techniques, any other suitable selection technique, or any combination thereof, and may take into account current computational/processing tasks. Selection of the second computer may be performed by consulting the cluster registry 302, or by consulting cached registry information on the first computer.

[0063] An example of a remote service that node storage cluster 304 may provide is "**Traverse node**". This service, when given a list of nodes, a direction, and an evaluation, traverses the nodes in the list with the direction and the evaluator. An example of pseudo-code for this service is described below:

```
public void traverseNodes(
   int depth, long[] nodeIds, Direction direction, String
   evaluatorClassName) {
      Evaluator eval =
      Evaluator.createEvaluator(evaluatorClassName);
      List<Integer> nodeIdsToTraverse = new
      ArrayList<Integer>();
      // Read nodes and evaluate each node
      List<Node> nodes = queryNodesFromDatabase(nodeIds);
      for (Node node : nodes) {
         if (eval.evaluateNode(depth, node))
            nodeIdsToTraverse.add(node.getLocalId());
      }
      // Get edge locators, depending on direction.
      // If direction is OUTGOING, query Outgoing_Edge;
      // otherwise query Incoming_Edge
      // The Map returned maps a cluster id to a set of edge
      local id's
      // within that cluster.
      Map<Integer, Set<Integer>> locators =
```

```
        queryEdgeLocatorsFromDatabase(direction,
        nodeIdsToTraverse);
      List<RemoteCall> calls = new ArrayList<RemoteCall>();
      for (Map.Entry<Integer, Set<Integer>> entry : locators) {
        int clusterId = entry.getKey();
        int[] edgeIds = convertToIntArray(entry.getValues());
        String machine = pickMachineForCluster(clusterId);
        calls.add(
          makeAsynchronousRemoteTraverseEdgesCall(
          machine, depth, edgeIds, direction,
          evaluatorClassName
          )
        );
      }
      waitForCallToFinish(calls);
}
```

[0064] An example of a remote service that edge storage cluster 306 may provide is "**Traverse edges**". This service traverses a set of given edges. An example of pseudo-code for this service is described below:

```
public void traverseEdges(int depth, int[] edgeIds, Direction
direction,
   String evaluatorClassName) {
   Evaluator eval =
   Evaluator.createEvaluator(evaluatorClassName);
   List<Integer> edgesToTraverse = new ArrayList<Integer>();
   // Read edges and evaluate each one
   List<Edge> edges = queryEdgesFromDatabase(edgeIds);
   for (Edge edge : edges) {
      if (eval.evaluateEdge(edge))
        edgesToTraverse.add(edge);
   }
   // Get node locators, depending on direction.
   // If direction is OUTGOING, get the head locators of the
   edges;
   // otherwise get the tail locators of the edges.
   // The Map returned maps a cluster id to a set of node local
   id's
   // within that cluster.
   Map<Integer, Set<Integer>> locators =
      queryNodeLocatorsFromDatabase(edgesToTraverse);
   List<RemoteCall> calls = new ArrayList<RemoteCall>();
   for (Map.Entry<Integer, Set<Integer>> entry : locators) {
      int clusterId = entry.getKey();
      int[] nodeIds = convertToIntArray(entry.getValues());
      String machine = pickMachineForCluster(clusterId);
```

```
        calls.add(
          makeAsynchronousRemoteTraverseNodesCall(
            machine, depth + 1, edgeIds, direction,
            evaluatorClassName
          );
        );
      }
```

[0065] Using the "**Traverse nodes**" and "**Traverse edges**" services described above, graphs representative of network communities stored on the distributed network 300 may be traversed. In one embodiment, given a start node, the cluster registry 302 (or cached registry information) is consulted to determine the particular cluster on which the start node is stored. A remote call to the "**Traverse nodes**" service may then be made, passing a depth of 0, the start node, a desired direction, such as "INCOMING" or "OUTGOING", and an evaluator class. From there, alternate calls to the "**Traverse edges**" service and the "**Traverse nodes**" service may be made until the traversal is complete. Completion of the traverse may be determined by the evaluator class. In certain embodiments, this traversal may not guarantee any kind of order, such as Depth First or Breadth First order, because the computation of the traversal may be distributed across the computers in one or more clusters, and may not result in visiting nodes sequentially.

[0066] In the pseudo-code shown above for the "**Traverse nodes**" and "**Traverse edges**" services, one or more objects that inherit from an "**Evaluator**" abstract class may be used. An example of pseudo-code for the "**Evaluator**" abstract class is described below:

```
abstract class Evaluator {
  private static Map<String, Evaluator> evaluators = new
  HashMap<String, Evaluator>();
  public static Evaluator createEvaluator(String name) {
    Evaluator result = evaluators.get(name);
    if (result == null) {
      result = Class.forName(name).newInstance();
      // the evaluator should listen on the network for a
      query
      // from the remote caller.
      listenForQueries(result);
      // Let the remote caller know of this evaluator's
      existence,
      // so the remote caller can query it later.
      registerEvaluatorWithRemoteCaller();
      evaluators.put(name, result);
    }
```

```
      return result;
    }
    public abstract void evaluateEdge(Edge edge);
    public abstract void evaluateNode(int depth, Node node);
}
```

[0067] For example, pseudo-code for an evaluator that counts nodes and edges is described below:

```
class NodeAndEdgesCounter extends Evaluator {
    private int nodes;
    private int edges;
    @Override
    public void evaluateEdge(Edge edge) {
      edges++;
    }
    @Override
    public void evaluateNode(int depth, Node node) {
      nodes++;
    }
    @RemoteCall
    public int getNodeCount() { return nodes; }
    @RemoteCall
    public int getEdgeCount() { return edges ; }
}
```

The @RemoteCall annotation in the example pseudo-code above indicates that those particular methods ("**getNodeCount()**" and "**getEdgeCount()**") may be called remotely.

[0068] FIG. 4A-C shows illustrative data tables for graph information storage in a distributed storage/computation network, such as distributed network 300, in accordance with one embodiment of the invention. FIG. 4A shows common data tables 400 that may be stored on each computer and/or cluster in the distributed network 300. For example, a particular computer may store cluster information table 402, which includes information about the cluster the particular computer is in. The cluster information table 402 may include a unique identifier or ID assigned to the cluster, along with the particular type of cluster (e.g., node storage or edge storage) it is. A particular computer may also store a registry cache table 404, which may be a cache of the data stored in the cluster registry 302. The registry cache table 404 may store the unique ID for each cluster in the distributed network 300, as well as an identifier (such as a network/IP address) for each computer in each cluster.

[0069] Data tables 410, shown in FIG. 4B, may store information about nodes in a network community, and may be stored on computers in node storage cluster 304. In some

embodiments, each node storage cluster is responsible for a subset of the nodes in a network community, and stores the information for that subset of nodes. For example, a node table 412 stored on computers in node storage cluster 304 may contain information only for nodes that node storage cluster 304 is responsible for. Node table 412 may store an identifier for a node stored in a particular node storage cluster, and each node may have a different node table. The identifier may be a unique, local identifier used to identify the node within the cluster. In other embodiments, the node table 412 may also store application specific data. For example, for the purposes of calculating a trust score, node table 412 may store a mean trust value and/or a trust value standard deviation for the node.

[0070] In some embodiments, computers in node storage cluster 304 may also store an outgoing edge storage table 414. Outgoing edge storage table 414 may store all outgoing edges for all nodes in a given cluster. For example, an outgoing edge storage table stored on computers in node storage cluster 304 may only store outgoing edge information for nodes that storage cluster 304 is responsible for. For each outgoing edge, the storage table 414 may store a node identifier that identifies the particular node within node storage cluster 304 that is associated with the outgoing edge. The storage table 414 may also store a cluster identifier for the edge that identifies the particular cluster the edge is stored in, as well as an edge identifier that identifies the edge in that particular cluster.

[0071] In some embodiments, computers in node storage cluster 304 may also store an incoming edge storage table 416. Incoming edge storage table 416 may store all incoming edges for all nodes in a given cluster, and in some embodiments may be structurally similar to outgoing edge storage table 414. For example, an incoming edge storage table stored on computers in node storage cluster 304 may only store incoming edge information for nodes that storage cluster 304 is responsible for. For each incoming edge, the storage table 416 may store a node identifier that identifies the particular node within node storage cluster 304 that is associated with the incoming edge. The storage table 416 may also store a cluster identifier for the edge that identifies the particular cluster the edge is stored in, as well as an edge identifier that identifies the edge in that particular cluster.

[0072] Data table 420, shown in FIG. 4C, may store information about edges in a network community, and may be stored on computers in edge storage cluster 306. In some embodiments, each edge storage cluster is responsible for a subset of the edges in a network community, and stores the information for that subset of edges. For example, an edge table

422 stored on computers in edge storage cluster 306 may only contain information for edges that edge storage cluster 306 is responsible for. Edge table 422 may store an identifier for an edge stored in a particular edge storage cluster, and each edge may have a different edge table 422. The identifier may be a unique, local identifier used to identify the edge within the cluster. The edge table 422 may also store information about the nodes that a particular edge connects. In some embodiments, an edge may be a directed edge that links a head node and a tail node. In these embodiments, the edge table 422 may store a cluster identifier and a node identifier for each of the head node and the tail node. The cluster identifier identifies the particular node storage cluster in the distributed system 300 that the head or tail node is stored in, and the node identifier identifies the particular head or tail node within that node storage cluster. In other embodiments, other data tables may be stored in cluster registry 302, node storage cluster 304, or edge storage cluster 306.

[0073] FIG. 5A shows illustrative data tables 500 used to support the connectivity determinations of the present invention. One or more of tables 500 may be stored in, for example, a relational database in data store 110 (FIG. 1). Table 502 may store an identification of all the nodes registered in the network community. A unique identifier may be assigned to each node and stored in table 502. In addition, a string name may be associated with each node and stored in table 502. As described above, in some embodiments, nodes may represent individuals or entities, in which case the string name may include the individual or person's first and/or last name, nickname, handle, or entity name.

[0074] Table 504 may store user connectivity values. User connectivity values may be positive, indicating some degree of trust between two or more parties, or may be negative, indicating some degree of distrust between two or more parties. In some embodiments, user connectivity values may be assigned automatically by the system (e.g., by application server 106 (FIG. 1)). For example, application server 106 (FIG. 1) may monitor all electronic interaction (e.g., electronic communication, electronic transactions, or both) between members of a network community. In some embodiments, a default user connectivity value (e.g., the link value 1) may be assigned initially to all links in the network community. After electronic interaction is identified between two or more nodes in the network community, user connectivity values may be adjusted upwards or downwards depending on the type of interaction between the nodes, the content of the interaction, and/or the result of the interaction. For example, each simple email exchange between two nodes may automatically

increase or decrease the user connectivity values connecting those two nodes by a fixed amount. In some embodiments, the content of the emails in the email exchange may be processed by, for example, application server 106 (FIG. 1) to determine the direction of the user connectivity value change as well as its magnitude. For example, an email exchange regarding a transaction executed in a timely fashion may increase the user connectivity value, whereas an email exchange regarding a missed deadline may decrease the user connectivity value. The content of the email exchange or other interaction may be processed by using heuristic and/or data/text mining techniques to parse the content of the interaction. For example, a language parser may be used to identify keywords in the email exchange. In some embodiments, individual emails and/or the email exchange may be processed to identify keywords that are associated with successful/favorable transactions and/or keywords that are associated with unsuccessful/unfavorable transactions, and the difference between the frequency/type of the keywords may affect the user connectivity value. In certain embodiments, natural language parsers may be used to extract semantic meaning from structured text in addition to keyword detection.

[0075] More complicated interactions (e.g., product or service sales or inquires) between two nodes may increase or decrease the user connectivity values connecting those two nodes by some larger fixed amount. In some embodiments, user connectivity values between two nodes may always be increased unless a user or node indicates that the interaction was unfavorable, not successfully completed, or otherwise adverse. For example, a transaction may not have been timely executed or an email exchange may have been particularly displeasing. Adverse interactions may automatically decrease user connectivity values while all other interactions may increase user connectivity values (or have no effect). In some embodiments, the magnitude of the user connectivity value change may be based on the content of the interactions. For example, a failed transaction involving a small monetary value may cause the user connectivity value to decrease less than a failed transaction involving a larger monetary value. In addition, user connectivity values may be automatically harvested using outside sources. For example, third-party data sources (such as ratings agencies and credit bureaus) may be automatically queried for connectivity information. This connectivity information may include completely objective information, completely subjective information, composite information that is partially objective and partially subjective, any other suitable connectivity information, or any combination of the foregoing.

[0076] In some embodiments, user connectivity values may be manually assigned by members of the network community. These values may represent, for example, the degree or level of trust between two users or nodes or one node's assessment of another node's competence in some endeavor. As described above, user connectivity values may include a subjective component and an objective component in some embodiments. The subjective component may include a trustworthiness "score" indicative of how trustworthy a first user or node finds a second user, node, community, or subcommunity. This score or value may be entirely subjective and based on interactions between the two users, nodes, or communities. A composite user connectivity value including subjective and objective components may also be used. For example, third-party information may be consulted to form an objective component based on, for example, the number of consumer complaints, credit score, socio-economic factors (e.g., age, income, political or religions affiliations, and criminal history), or number of citations/hits in the media or in search engine searches. Third-party information may be accessed using communications network 104 (FIG. 1). For example, a third-party credit bureau's database may be polled or a personal biography and background information, including criminal history information, may be accessed from a third-party database or data source (e.g., as part of data sources 108 (FIG. 1) or a separate data source) or input directly by a node, user, or system administrator. In some embodiments, the third-party data source(s) or system(s) may also include third-party user connectivity values and transaction histories, related to user interactions with the third-party system(s). In these embodiments, the user connectivity value or composite user connectivity value may also include one or more components based on the third-party user connectivity values and transaction histories.

[0077] In other embodiments, the user connectivity or trust value may be calculated objectively. In one embodiment, the trust value of a first node for a second node may be calculated based on the number of paths linking the two nodes, one or more path scores associated with the linking paths, connectivity statistics associated with the first node, and/or other connectivity information associated with the first node.

[0078] Table 504 may store an identification of a link head, link tail, and user connectivity value for the link. Links may or may not be bidirectional. For example, a user connectivity value from node n_1 to node n_2 may be different (and completely separate) than a link from node n_2 to node n_1. Especially in the trust context described above, each user can

assign his or her own user connectivity value to a link (i.e., two users need not trust each other an equal amount in some embodiments).

[0079] Table 506 may store an audit log of table 504. Table 506 may be analyzed to determine which nodes or links have changed in the network community. In some embodiments, a database trigger is used to automatically insert an audit record into table 506 whenever a change of the data in table 504 is detected. For example, a new link may be created, a link may be removed, and/or a user connectivity value may be changed. This audit log may allow for decisions related to connectivity values to be made prospectively (i.e., before an anticipated event). Such decisions may be made at the request of a user, or as part of an automated process, such as the processes described below with respect to FIG. 10. This prospective analysis may allow for the initiation of a transaction (or taking of some particular action) in a fluid and/or dynamic manner. After such a change is detected, the trigger may automatically create a new row in table 506. Table 506 may store an identification of the changed node, identification of the changed link head, changed link tail, and/or the user connectivity value to be assigned to the changed link. Table 506 may also store a timestamp indicative of the time of the change and/or an operation code. In some embodiments, operation codes may include "insert," "update," and/or "delete" operations, corresponding to whether a link was inserted, a user connectivity value was changed, or a link was deleted, respectively. Other operation codes may be used in other embodiments.

[0080] FIG. 5B shows illustrative data structure 510 used to support the connectivity determinations of the present invention. In some embodiments, data structure 510 may be stored using key-value store 112 (FIG. 1), while tables 500 are stored in data store 110 (FIG. 1). As described above, key-value store 112 (FIG. 1) may implement an HBase storage system and include BigTable support. Like a traditional relational database management system, the data shown in FIG. 5B may be stored in tables. However, the BigTable support may allow for an arbitrary number of columns in each table, whereas traditional relational database management systems may require a fixed number of columns.

[0081] Data structure 510 may include node table 512. In the example shown in FIG. 5B, node table 512 includes several columns. Node table 512 may include row identifier column 514, which may store 64-bit, 128-bit, 256-bit, 512-bit, or 1024-bit integers and may be used to uniquely identify each row (e.g., each node) in node table 512. Column 516 may include a list of all the incoming links for the current node. Column 518 may include a list of

all the outgoing links for the current node. Node table 512 may also include one or more "bucket" columns 520 and 522. These columns may store a list of paths that connect, for example, a source node to the current node, the current node to a target node, or both. As described above, grouping paths by the last node in the path (e.g., the target node), the first node in the path (e.g., the source node), or both, may facilitate connectivity computations. As shown in FIG. 5B, in some embodiments, to facilitate scanning, bucket column names may include the target node identifier appended to the end of the "bucket:" column name.

[0082] FIG. 5C shows illustrative database schema 530 used to facilitate financial transactions. Table 532 includes information related to users' sign-in profiles. For example, a user may have accounts for multiple email, social networking services, other online or network services, or any combination of the foregoing. Each of these accounts may be included in a separate sign-in profile associated with the user. As such, a single user may be associated with one or more sign-in profiles. In some embodiments, instead of including a distinct sign-in system specific to the connectivity system, a user may sign in to one of these existing accounts or services identified in a sign-in profile, and then the connectivity system may ask the existing service to vouch for or verify the identity of the user. Table 532 may include a string identification of the service or provider associated with the profile, a unique identifier associated with the profile, an email or username field, and a nickname, handle, or real name field.

[0083] For example, a user may wish to log into the connectivity system (or some loan or financial transaction system that uses the connectivity system) using access application 102 (FIG. 1). Application server 106 (FIG. 1) may then ask the user which service (of a list of available external services) to use for authentication. Application server 106 (FIG. 1) may then redirect the user to the external service's sign-in mechanism. The external service may then redirect the user back to the connectivity system (for example, a web page hosted by application server 106 (FIG. 1)). Application server 106 (FIG. 1) may then lookup the sign-in profile (e.g., in table 532) in order to identify the user.

[0084] Table 534 may include an indication of a person or node in the network community. For example, the person associated with table 534 may be an officer in a financial institution, a lender, a borrower, or a donor. Officer table 536 may include a unique identifier representing the financial institution associated with the officer and identified in organization table 538. Donation table 540 and loan table 542 may include any suitable

information related to donations or loans, respectively, available on the network. Donation table 540 may include such information as a unique identifier associated with a donation, a unique identifier associated with the donor, a unique identifier associated with the financial application, whether or not a tax receipt is needed, whether or not a tax receipt has been issued, the tax receipt number, the tax receipt date, and a status indicator. The status indicator may include "0" if the donation is still waiting for a check as a source of funding for the donation, a "1" if the donation is still waiting for an external payment system as a source of funding for the donation, "2" if the donation has been canceled by the user, the financial application, the officer, or financial institution, "3" if the donation is currently active, "4" is the donation has been completed, "5" if the donor has defaulted, "6" is the donation is associated with a refund amount.

[0085] Similarly, loan table 542 may include a unique identifier associated with a loan, a unique identifier associated with the financial application, a unique identifier associated with the lender, the principal of the loan, the balance of the loan (e.g., the remaining principal on the loan), and a status indication. The status indicator may be the same as the status indicators described above with respect to the donation table. Financial application table 544 may identify the loans, donations, or other types of financial applications available in the network. Financial application table 544 may include a unique identifier for the application, a string description associated with the application (which may also include attribute flags and other metadata associated with the financial application and used in determining publication groups, as described in more detail with regard to FIG. 10 below), a unique borrower identifier, a currency type indication, the principal requested or available, the principal raised, the interest rate associated with the loan or donation, the payment period, the number of payment periods per year, and the number of compounding periods per year. Some fields in financial application table 544 may only apply to loan type applications or donation type applications.

[0086] In some embodiments, the description field in financial application table 544 may include "LIKE" and "DISLIKE" flags identifying affinity groups, blogs, newsgroups, and other information used to determine what nodes or users may be interested or not interested in a particular financial application. These flags may be used in determining publication groups, as described in more detail below. For example, a mortgage type financial application may include a "LIKE" flag for users or nodes interested in securing real property (e.g., users or

nodes belonging to a real estate affinity group or real estate blog or newsgroup). As another example, a donation type financial application to support same-sex marriage may include a "LIKE" flag for users or nodes subscribed to the Human Rights Campaign or American Civil Liberties Union affinity group and a "DISLIKE" flag for users or nodes belonging to "Yes on Prop 8" or defense of marriage affinity group. Other attribute flags may also be defined in financial application table 544. These flags may be created by the sponsor or creator of the financial application and may be customized by users initiating financial transactions, in some embodiments.

[0087] Repayment schedule table 546 may be associated with each loan in loan table 542. Repayment schedule table 546 may include a unique identifier associated with the loan to which the repayment schedule relates, the current payment number, the due date for the net payment, the total amount due, and the total amount paid. Repayment schedule table 546 may be automatically generated, in some embodiments, whenever a new loan is created or initiated by a user and approved.

[0088] In a typical usage scenario, a user may be notified when certain users in the user's network have initiated a new financial transaction using a financial application identified in financial application table 544. For example, in some embodiments, users are notified whenever any other user initiates a financial transaction. In other embodiments, users are only notified about financial transactions made by other users meeting some threshold path weight or threshold user connectivity value with the to-be-notified user. For example, a message may be sent to second user that a first user has loaned $10,000 to "Save the Pandas" and that the specific financial application is the "Wildlife Sanctuary Project." This message may appear in email, as a pop-up message, or displayed as a link on the user's homepage, profile page, or initial log-in page.

[0089] The notified user may also decide to initiate a financial transaction using the same financial application. The user may then decide whether to fund the transaction using a check or using an external payment system (such as PayPal). Before the funding is received, the transaction may be marked as "waiting" for either a check or external payment system. For example, the status indicators in donation table 540 or loan table 542 may be set to "0" or "1". A repayment schedule may then be generated. For example, repayment schedule table 546 may be populated.

[0090] After funding has been received, the transaction may be marked as "active" and repayments may begin (depending on the transaction type). Repayments may be made, in some embodiments, by mailing a check, direct deposit, using an external payment system, or using any other suitable mechanism.

[0091] Although FIG. 5C shows one illustrative arrangement for schema 530, any other suitable schema may also be used. For example, more or fewer tables than those shown in FIG. 5C may be defined, each including more or fewer fields. In additiona, although a relational database management system may be used in some embodiments to save and access information in accordance with schema 530, any other storage or access mechanism may be used in other embodiments.

[0092] FIGS. 6A-6H show illustrative processes for determining the connectivity of nodes within a network community. FIG. 6A shows process 600 for updating a connectivity graph (or any other suitable data structure) associated with a network community. As described above, in some embodiments, each network community is associated with its own connectivity graph, digraph, tree, or other suitable data structure. In other embodiments, a plurality of network communities may share one or more connectivity graphs (or other data structure).

[0093] In some embodiments, the processes described with respect to FIGS. 4A-4C and 6A-6H may be executed to make decisions prospectively (i.e., before an anticipated event). Such decisions may be made at the request of a user, or as part of an automated process, such as the processes described below with respect to FIGS. 7-10. This prospective analysis may allow for the initiation of a transaction (or taking of some particular action) in a fluid and/or dynamic manner.

[0094] In some embodiments, the processes described with respect to FIGS. 4A-4C and 6A-6H may be executed to provide information to a user. Such presentations may be made at the request of a user, or as part of an automated presentation. This information may include, but is not limited to, static and/or interactive visualizations of connectivity values within a user's associated network community or communities. In some embodiments, this information may be integrated into explorations of or interactions within a user's associated network community or communities. Providing this information to a user may allow the user to better understand what other individuals and/or entities they may trust within a network

community, and/or may encourage and/or discourage particular interactions within a user's associated network community or communities.

[0095] At step 602, a determination is made whether at least one node has changed in the network community. As described above, an audit record may be inserted into table 506 (FIG. 5) after a node has changed. By analyzing table 506 (FIG. 5), a determination may be made (e.g., by application server 106 of FIG. 1) that a new link has been added, an existing link has been removed, or a user connectivity value has changed. If, at step 604, it is determined that a node has changed, then process 600 may continue to step 610 (shown in FIG. 6B) to process the changed links, step 612 (shown in FIG. 6C) to save the nodes with changed links, step 614 (shown in FIG. 6D) to create path set input files, step 616 (shown in FIG. 6E) to remove paths with changed nodes, one or more iterations of step 618 (shown in FIG. 6F) to grow paths by one link at a time, step 620 (shown in FIG. 6G) to save the paths that have grown by one or more links, and step 622 (shown in FIG. 6H) to join paths that go through changed nodes. It should be noted that more than one step or task shown in FIGS. 6B, 6C, 6D, 6E, 6F, 6G, and 6H may be performed in parallel using, for example, a cluster of cores. For example, multiple steps or tasks shown in FIG. 6B may be executed in parallel or in a distributed fashion, then multiple steps or tasks shown in FIG. 6C may be executed in parallel or in a distributed fashion, then multiple steps or tasks shown in FIG. 6D may be executed in parallel or in a distributed fashion, then multiple steps or tasks shown in FIG. 6E may be executed in parallel or in a distributed fashion, and so on. In this way, overall latency associated with process 600 may be reduced.

[0096] As described above, step 618 may be executed one or more times. This step may be operative to grow paths by a single link. Each iteration of step 618 may take as input the results of a previous iteration of step 618 so that paths may grow by more than one link, if desired. In the example of FIG. 6A, three iterations of step 618 are shown. Thus, process 600 may generate paths with lengths less than or equal to three. In other embodiments, more or fewer iterations of step 618 may allow process 600 to generate paths with more or fewer links.

[0097] If a node change is not detected at step 604, then process 600 enters a sleep mode at step 606. For example, in some embodiments, an application thread or process may continuously check to determine if at least one node or link has changed in the network community. In other embodiments, the application thread or process may periodically check for changed links and nodes every *n* seconds, where *n* is any positive number. After the paths

are calculated that go through a changed node at step 616 or after a period of sleep at step 606, process 600 may determine whether or not to loop at step 608. For example, if all changed nodes have been updated, then process 600 may stop at step 618. If, however, there are more changed nodes or links to process, then process 600 may loop at step 608 and return to step 604.

[0098] In practice, one or more steps shown in process 600 may be combined with other steps, performed in any suitable order, performed in parallel (e.g., simultaneously or substantially simultaneously), or removed.

[0099] FIGS. 6B-6H each include processes with a "map" phase and "reduce" phase. As described above, these phases may form part of a map/reduce computational paradigm carried out by parallel computational framework 114 (FIG. 1), key-value store 112 (FIG. 1), or both. As shown in FIG. 6B, in order to process link changes, map phase 626 may include determining if there are any more link changes at step 628, retrieving the next link change at step 630, mapping the tail to out-link change at step 632, and mapping the head to in-link change at step 634.

[0100] If there are no more link changes at step 628, then, in reduce phase 636, a determination may be made at step 638 that there are more nodes with mapped link changes to process. If so, then the next node and its link changes may be retrieved at step 640. The most recent link changes may be preserved at step 642 while any intermediate link changes are replaced by more recent changes. For example, the timestamp stored in table 506 (FIG. 5) may be used to determine the time of every link or node change. At step 644, the average out-link user connectivity value may be calculated. For example, if node n_1 has eight out-links with assigned user connectivity values, these eight user connectivity values may be averaged at step 644. At step 646, each out-link's weight may be calculated in accordance with equation (1) or (2) above. At step 648, an output file may be created or appended with the out-links changed and corresponding changed node identifier. For example, one or more (out-links changed, node identifier) records may be written to the output file. Although the term "file" is sometimes used herein, the output need not be in a literal file or even file format. For example, any output stream, whether or not it is recorded, may be used. In some embodiments, some or all of the output file may be passed directly to a calling application, process, or function from a returning application, process, or function in the form of a stream

or object return value. If there are no more nodes and link changes to process at step 638, the process may stop at step 650.

[0101] As shown in FIG. 6C, in order to save nodes with changed links, map phase 652 may include determining if there are any more changed nodes at step 654, retrieving the next changed node at step 656, and mapping "null" to the node at step 658.

[0102] If there are no more changed nodes at step 654, then, in reduce phase 660, a determination may be made at step 662 that there are more nodes to process. If so, then the next node may be retrieved at step 664. At step 666, the in-links and out-links associated with the node may be written to a key-value store (e.g., key-value store 112 of FIG. 1). As described above, the key-value store may implement an HBase cluster (or any other compressed, high performance database system, such as BigTable). If there are no more nodes to process at step 662, the process may stop at step 668.

[0103] As shown in FIG. 6D, in order to create path set input files, map phase 670 may include determining if there are any more (out-links changed, node identifier) records in the output file created or appended at step 648 (FIG. 6B). If so, the next record may be retrieved at step 674. At step 676, a determination may be made if an out-link has changed. If so, then at step 678 a "null" value may be mapped to the node. Otherwise, map phase 670 may return to step 672 to determine if there are any more (out-links changed, node identifier) records in the output file.

[0104] If there are no more changed records at step 672, then, in reduce phase 680, a determination may be made at step 682 that there are more node to process. If so, then the next node may be retrieved at step 684. At step 686, new records may be written to the output file. In some embodiments, the records written at step 686 may include records of the form (node identifier, empty path set for the node identifier). If there are no more nodes to process at step 682, the process may stop at step 688.

[0105] As shown in FIG. 6E, in order to remove paths with changed nodes, map phase 690 may include determining if there are any more (node identifier, path set) records in the output file at step 692 and retrieving the next such record at step 694. At step 696, for every "in" bucket identifier, the "in" bucket identifier may be mapped to a record of the form (out bucket type, node identifier, set of "out" bucket identifiers) (or any other suitable form). At step 698, for every "out" bucket identifier, the "out" bucket identifier may be mapped to a record of the form (in bucket type, node identifier, set of "in" bucket identifiers) (or any other

suitable form). At step 700, the node's "out" buckets may be deleted, and the process may return to step 692 to determine if there are more records to process.

[0106] If there are no more records at step 692, then, in reduce phase 702, a determination may be made at step 704 that there are more node identifiers with their mapped (bucket type, changed node identifier, bucket identifiers) records to process. If so, then at step 706, if the bucket type is "out", out-buckets with the given bucket identifiers may be searched and paths with the changed node identifier may be removed. At step 708, if the bucket type is "in", in-buckets with the given bucket identifiers may be searched and paths with the changed node identifier may be removed. If there are no more records to process at step 704, the process may stop at step 710.

[0107] As shown in FIG. 6F, in order to grow paths by one link, map phase 712 may include determining if there are any more (node identifier, path set) records in the output file at step 714. If so, then at step 716, if the path set is empty, for each out-link of the node, a link head identifier may be mapped to the link. At step 718, if the path set is not empty, then for each path *n* in the path set, and for each out-link of a node, a new path may be created by appending (out-link, map link head identifier) to the new path.

[0108] If there are no more records at step 714, then, in reduce phase 720, a determination may be made at step 722 that there are more node identifiers with mapped paths to process. If so, then at step 724, new records of the form (node identifier, mapped paths) (or any other suitable form) may be written to the output file. If there are no more records to process at step 722, the process may stop at step 726.

[0109] The process shown in FIG. 6F may be executed one or more times, with the result of growing path lengths by one link for each execution. As shown in FIG. 6A, in some embodiments, three iterations of the process shown in FIG. 6F are used to grow paths by three links. In other embodiments, more or fewer iterations are used.

[0110] As shown in FIG. 6G, in order to save the new paths, map phase 728 may include determining if there are any more (node identifier, path set) records in the output file at step 730. If so, then at step 732, for each path in the path set, the path tail identifier may be mapped to the path. At step 734, for each path in the path set, the path head identifier may be mapped to the path.

[0111] If there are no more records at step 730, then, in reduce phase 736, a determination may be made at step 738 that there are more node identifiers with mapped paths

to process. If so, then at step 740, if the path tail identifier equals the node identifier, then that path may be added to the node's "out" bucket for the path head identifier. At step 742, if the path head identifier equals the node identifier, then that path may be added to the node's "in" bucket for the path tail identifier. At step 744, the node may be saved. If there are no more records to process at step 738, the process may stop at step 746.

[0112] As shown in FIG. 6H, in order to join paths that go through changed nodes, map phase 748 may include determining if there are any more (node identifier, path set) records in the output file at step 750. If so, then at step 752, all paths in "in" buckets may be joined with all paths in "out" buckets. At step 754, for each qualified joined path with length less than or equal to three (or the number of iterations of the process shown in FIG. 6F), the path tail identifier may be mapped to the path, and the path head identifier may also be mapped to the path.

[0113] If there are no more records at step 750, then, in reduce phase 756, a determination may be made at step 758 that there are more node identifiers with mapped paths to process. If so, then at step 760, if the path tail identifier equals the node identifier, then that path may be added to the node's "out" bucket for the path head identifier. At step 762, if the path head identifier equals the node identifier, then that path may be added to the node's "in" bucket for the path tail identifier. At step 764, the node may be saved. If there are no more records to process at step 758, the process may stop at step 766.

[0114] FIG. 7 shows illustrative process 780 for supporting a user query for all paths from a first node to a target node. For example, a first node (representing, for example, a first individual or entity) may wish to know how connected the first node is to some second node (representing, for example, a second individual or entity) in the network community. In the context of trust described above (and where the user connectivity values represent, for example, at least partially subjective user trust values), this query may return an indication of how much the first node may trust the second node. In general, the more paths connecting the two nodes may yield a greater (or lesser if, for example, adverse ratings are used) network connectivity value (or network trust amount).

[0115] At step 782, for each source node "out" bucket, the corresponding "in" bucket of target nodes may be located. For example, column 520 of node table 512 (both of FIG. 5B) may be accessed at step 782. Paths from the source node's "out" bucket may then be joined with paths in the target node's "in" bucket at step 784. Joined paths with paths in the source

node's "out" bucket may then be returned for the target node's identifier. Process 780 may stop at step 788.

[0116] Having returned all paths between the source and target node (of length less than or equal to three, or any other suitable value depending on the number of iterations of the process shown in FIG. 6F), a network connectivity value may be computed. The path weights assigned to the paths returned at step 786 may then be summed. The path weights may be normalized by dividing each path weight by the computed sum of the path weights. A network connectivity value may then be computed. For example, each path's user connectivity value may be multiplied by its normalized path weight. The network connectivity value may then be computed in some embodiments in accordance with:

$$t_{network} = \sum t_{path} \times w_{path} \tag{7}$$

where t_{path} is the user connectivity value for a path (given in accordance with equation (5)) and w_{path} is the normalized weight for that path. The network connectivity value may then be held or output by processing circuitry of application server 106 (FIG. 1), stored on data store 110 (FIG. 1), or both. In addition, a decision-making algorithm may access the network connectivity value in order to make automatic decisions (e.g., automatic network-based decisions, such as authentication or identity requests) on behalf of the user. Network connectivity values may additionally or alternatively be outputted to external systems and processes located at third-parties. The external systems and processes may be configured to automatically initiate a transaction (or take some particular course of action) based, at least in part, on the received network connectivity values. For example, some locales or organizations may require identity references in order to apply for a document (e.g., a passport, driver's license, group or club membership card, etc.). The identity reference or references may vouch that an individual actually exists and/or is the individual the applicant is claiming to be. Network connectivity values may be queried by the document issuer (e.g., a local government agency, such as the Department of Motor Vehicles or a private organization) and used as one (or the sole) metric in order to verify the identity of the applicant, the identity of an identity reference, or both. In some embodiments, network connectivity values may be used as an added assurance of the identity of an applicant or reference in conjunction with more traditional forms of identification (e.g., document verification and knowledge-based identity techniques). If the document issuer (or some other party trusted by the document issuer) has a set of strong paths from the applicant or reference, this may indicate a higher degree of

confidence in the identity of the applicant or reference. Such an indication may be outputted to the third-party system or process.

[0117] As another example, credit-granting decisions may be made by third parties based, at least in part, on network connectivity values. One or more queries for a network connectivity value may be automatically executed by the credit-granting institution (e.g., a bank, private financial institution, department store) as part of the credit application process. For example, a query for a network connectivity value between the applicant and the credit-granting institution itself (or its directors, board members, etc.) and between the applicant and one or more trusted nodes may be automatically executed as part of the credit application process. The one or more network connectivity values returned to the credit-granting institution may then be used as an input to a proprietary credit-granting decision algorithm. In this way, a credit-granting decision may be based on a more traditional component (e.g., occupation, income, repayment delinquencies, and credit score) and a network connectivity component. Each component may be assigned a weight and a weighted sum or weighted average may be computed. The weighted sum or average may then be used directly to make an automatic credit-granting decision for the applicant. The weights assigned to each component of the weighted sum or average may be based on such factors as the applicant's credit history with the financial institution, the amount of credit requested, the degree of confidence in the trusted nodes, any other suitable factor, or any combination of the foregoing factors. In some embodiments, the credit-granting or other decisions made by third parties may be made based entirely on network connectivity values.

[0118] In practice, one or more steps shown in process 780 may be combined with other steps, performed in any suitable order, performed in parallel (e.g., simultaneously or substantially simultaneously), or removed. In addition, as described above, various threshold functions may be used in order to reduce computational complexity. For example, one or more threshold functions defining the maximum and/or minimum number of links to traverse may be defined. Paths containing more than the maximum number of links or less than the minimum number of links specified by the threshold function(s) may not be considered in the network connectivity determination. In addition, various maximum and/or minimum threshold functions relating to link and path weights may be defined. Links or paths above a maximum threshold weight or below a minimum threshold weight specified by the threshold function(s) may not be considered in the network connectivity determination.

[0119] Although process 780 describes a single user query for all paths from a first node to a target node, in actual implementations groups of nodes may initiate a single query for all the paths from each node in the group to a particular target node. For example, multiple members of a network community may all initiate a group query to a target node. Process 780 may return an individual network connectivity value for each querying node in the group or a single composite network connectivity value taking into account all the nodes in the querying group. For example, the individual network connectivity values may be averaged to form a composite value or some weighted average may be used. The weights assigned to each individual network connectivity value may be based on seniority in the community (e.g., how long each node has been a member in the community), rank, or social stature. In addition, in some embodiments, a user may initiate a request for network connectivity values for multiple target nodes in a single query. For example, node n_1 may wish to determine network connectivity values between it and multiple other nodes. For example, the multiple other nodes may represent several candidates for initiating a particular transaction with node n_1. By querying for all the network connectivity values in a single query, the computations may be distributed in a parallel fashion to multiple cores so that some or all of the results are computed substantially simultaneously.

[0120] In addition, queries may be initiated in a number of ways. For example, a user (represented by a source node) may identify another user (represented by a target node) in order to automatically initiate process 780. A user may identify the target node in any suitable way, for example, by selecting the target node from a visual display, graph, or tree, by inputting or selecting a username, handle, network address, email address, telephone number, geographic coordinates, or unique identifier associated with the target node, or by speaking a predetermined command (e.g., "query node 1" or "query node group 1, 5, 9" where 1, 5, and 9 represent unique node identifiers). After an identification of the target node or nodes is received, process 720 may be automatically executed. The results of the process (e.g., the individual or composite network connectivity values) may then be automatically sent to one or more third-party services or processes as described above.

[0121] In an embodiment, a user may utilize access application 102 to generate a user query that is sent to access application server 106 over communications network 104 (see also, FIG. 1) and automatically initiate process 780. For example, a user may access an Apple iOS, Android, or WebOs application or any suitable application for use in accessing

application 106 over communications network 104. The application may display a searchable list of relationship data related to that user (e.g., "friend" or "follower" data) from one or more of Facebook, MySpace, open Social, Friendster, Bebop, hi5, Rout, PerfSpot, Yahoo! 360, LinkedIn, Twitter, Google+, Really Simple Syndication readers or any other social networking website or information service. In some embodiments, a user may search for relationship data that is not readily listed – i.e., search Facebook, Twitter, or any suitable database of information for target nodes that are not displayed in the searchable list of relationship data. A user may select a target node as described above (e.g., select an item from a list of usernames representing a "friend" or "follower") to request a measure of how connected the user is to the target node. Using the processes described with respect to FIGs. 4A-C, 5A-5C, and 6A-6H, this query may return an indication of how much the user may trust the target node. The returned indication may be displayed to the user using any suitable indicator. In some embodiments, indicator may be a percentage that indicates how trustworthy the target node is to the user.

[0122] In some embodiments, a user may utilize access application 102 to provide manual assignments of at least partially subjective indications of how trustworthy the target node is. For example, the user may specify that he or she trusts a selected target node (e.g., a selected "friend" or "follower") to a particular degree. The particular degree may be in the form of a percentage that represents the user's perception of how trustworthy the target node is. The user may provide this indication before, after, or during process 780 described above. The indication provided by the user (e.g., the at least partially subjective indications of trustworthiness) may then be automatically sent to one or more third-party services or processes as described above. In some embodiments, the indications provided by the user may cause a node and/or link to change in a network community. This change may cause a determination to be made that at least one node and/or link has changed in the network community, which in turn triggers various processes as described with respect to FIGs. 4A-C, 5A-5C, and 6A-6H. In some embodiments, a user may utilize access application 102 to interact with or explore a network community. For example, a user may be presented with an interactive visualization that includes one or more implicit or explicit representations of connectivity values between the user and other individuals and/or entities within the network community. This interactive visualization may allow the user to better understand what other individuals and/or entities they may trust within a network community, and/or may encourage

and/or discourage particular interactions within a user's associated network community or communities.

[0123] In some embodiments, a path counting approach may be used in addition to or in place of the weighted link approach described above. Processing circuitry (e.g., of application server 106 (FIG. 1)) may be configured to count the number of paths between a first node n_1 and a second node n_2 within a network community. A connectivity rating R_{n1n2} may then be assigned to the nodes. The assigned connectivity rating may be proportional to the number of paths, or relationships, connecting the two nodes. A path with one or more intermediate nodes between the first node n_1 and the second node n_2 may be scaled by an appropriate number (e.g., the number of intermediate nodes) and this scaled number may be used to calculate the connectivity rating.

[0124] In certain embodiments, the connectivity statistics of one or more nodes may be used to determine the connectivity score or rating between one node and another node. FIG. 8 shows illustrative process 800 for determining a connectivity or trust score of a node *a* for another node *b* based on connectivity statistics, in accordance with one embodiment of the invention. In step 802, a path score is determined for each path between node *a* and node *b*. In some embodiments, path scores may vary as a function of the path length. For example, the path score of a particular path may be calculated in accordance with:

$$PathScore(path) = \frac{1}{Length(path)^2} \tag{8}$$

where Length(path) is the length of a particular path between *a* and *b*, for example in terms of the number of nodes the path passes through. While in equation (8), the path score is shown to vary inversely according to the square of the length of the path, in other embodiments, the exponent may vary, and/or the path score function may vary according to some other function of path length. In some embodiments, the path score may also be based on one or more specific ratings or weights associated with one or more edges along the path, where an edge is a path between two adjacent nodes. For example, the path score may vary either directly or inversely proportional to the sum or the product of one or more ratings or weights associated with each edge along the path. In some embodiments, only the ratings or weights associated with specific edges may be included, and in other embodiments, ratings or weights associated with all of the edges in a particular path may be considered. For example, in some embodiment, the path score of a particular path may be calculated in accordance with:

$$PathScore(path) = \frac{\prod_{edge \in path} w_{edge}}{Length(path)^2} \quad (8a)$$

where $0 \le w_{edge} \le 1$.

[0125] In some embodiments, the path score may vary as one or more functions of the weights associated with one or more edges along the path. For example, in some embodiment, the path score of a particular path may be calculated in accordance with:

$$PathScore(path) = g(path) * \prod_{edge \in path} f(w_{edge}) \quad (9)$$

where $f(w\ )$ is defined in accordance with:

$$f(w\ ) = \begin{cases} 4, & if\ w < 0.2 \\ 2, & if\ 0.2 \le w < 0.4 \\ 1, & if\ 0.4 \le w < 0.8 \\ 2\ & if\ 0.8 \le w < 1.0 \\ 4, & if\ w = 1.0 \end{cases} \quad (10)$$

and $g(path)$ is defined in accordance with:

$$g(path) = \begin{cases} -1, & \exists w_{edge} < .6 \\ 1, & otherwise \end{cases} \quad (11)$$

[0126] After path scores for one or more of the paths linking nodes *a* and *b* have been calculated in step 802, the calculated path scores may be aggregated in step 804 to result in a connectivity value between the two nodes. The connectivity value between nodes *a* and *b* may be given in accordance with:

$$Connectivity(a,b) = \sum_{p \in Paths(a,b)} PathScore(p) \quad (12)$$

where Paths(a,b) represent one or more of the paths between nodes *a* and *b* and PathScore(p) represents the path score of one of the paths in Paths(a,b) (i.e., one of the paths between nodes *a* and *b*). While in equation (12), the connectivity between nodes *a* and *b* is a summation of path scores associated with one or more paths between *a* and *b*, in other embodiments, the connectivity may be a product or any other function of the path scores associated with one or more paths between *a* and *b*.

[0127] In step 806, the connectivity statistics for node *a* may be determined. First, a node sample may be selected for node *a*. In one embodiment, the node sample may include nodes that meet some network parameter with respect to node *a*. For example, every individual node with a network distance less than or equal to some number *x* from node *a* (i.e., a path exists from the node to node *a* with length less than or equal to *x*) may be included in the node sample. For example, in certain embodiments, every individual node with a network distance less than or equal to 3 from node *a* may be included in the node sample. In some embodiments, the node sample may include a certain number of individual nodes randomly selected from the population. In some embodiments, the node sample may include a certain number of individual nodes visited via a random exploration of the network, starting from node *a*, in a manner similar to a graph traversal. In some embodiments, the node sample may include a certain number of nodes that are directly connected to a. For example, in certain embodiments, the node sample may include every node with a network distance of 1 from node a. In other embodiments, any other suitable method for selecting individual nodes in the network may be used to create the node sample.

[0128] Once the sample has been selected, a mean path score μ_a, in accordance with:

$$\mu_a = \frac{1}{|S|}\sum_{y \in S} Connectivity(a, y) \tag{13}$$

and a standard deviation σ_a, in accordance with:

$$\sigma_a = \sqrt{\frac{1}{|S|}\sum_{y \in S}\left(Connectivity(a, y) - \mu_a\right)^2} \tag{14}$$

may be calculated for node *a*, where *S* is the number of individual nodes in the sample and Connectivity(a,y) is the connectivity (as described above in equation (12) between node *a* and a node *y* in the sample S.

[0129] Once the connectivity statistics have been determined for node *a*, the trust or connectivity score (not to be confused with the connectivity described above in equation (12)) of node *a* for node *b* may be determined in step 808, based on the connectivity statistics of node *a* and the connectivity between node *a* and node *b*. In one embodiment, the trust or connectivity score may be determined as a function of the area under the normal curve for μ_a and σ_a. For example, let *n* be the number of standard deviations the connectivity between node *a* and node *b* is away from the mean path score μ_a:

$$n = \frac{Connectivity(a,b) - \mu_a}{\sigma_a} \tag{15}$$

The trust or connectivity score between node *a* and node *b* TrustScore(*a*,*b*) may then be determined as a function of the area under the normal curve, in accordance with:

$$TrustScore(a,b) = 0.5 + \frac{erf\left(\frac{n}{\sqrt{2}}\right)}{2} \tag{16}$$

In some embodiments, the trust score may be used as is, be multiplied by 100 and presented as a percentage, or be multiplied by 1000 and presented as a number. The process 800 may then stop at step 810.

[0130] FIG. 9 shows illustrative process 900 for logging into the connectivity system. At step 902, a user request to login may be received. For example, application server 106 (FIG. 1) may receive a login attempt from access application 102 (FIG. 1). At step 904, one or more external login mechanisms may be accessed. For example, the user may be redirected to a login mechanism associated with an email or social networking service, like Facebook, Hotmail, Gmail, Twitter, or the like. After the external login mechanism is accessed, the user may be redirected to the application server at step 906. For example, the user may be redirected back to the page associated with application server 106 (FIG. 1). At step 908, a determination is made whether the external login mechanism was completed successfully. For example, the external login mechanism may return a token, timestamp, username, handle, email address, unique identifier, cryptographic hash (e.g., of a username or unique identifier associated with the user), any other identity information, or any combination of the foregoing in the URL to the redirected application server page. The information may be verified using any known authentication protocol. If the external login mechanism was successful, then at step 910 application server 106 (FIG. 1) may lookup a corresponding sign-in profile in order to identify the user. For example, the provider of the external login mechanism may pass its name as a string along with a unique identifier to application server 106 (FIG. 1). Application server 106 (FIG. 1) may then look this information up in table 532 (FIG. 5C). If a corresponding sign-in profile record is located, this profile may be used to identify the user.

[0131] In practice, one or more steps shown in process 900 may be combined with other steps, performed in any suitable order, performed in parallel (e.g., simultaneously or substantially simultaneously), or removed.

[0132] FIG. 10 shows illustrative process 1000 for facilitating a financial transaction. Although the described embodiments sometimes refer to a loan or donation financial application or transaction, the present invention may be used to facilitate any type of financial transaction. For example, financial transactions may include purchases, sales, donations of cash, donations of property, loans, mortgages, liens, credit applications, credit-granting decisions, or any other type of financial transaction involving the change in status of finances or change in legal status between two or more individuals, nodes, users, institutions, organizations, pieces of property, tangible assets, or things. At step 1002, a first user may initiate a new financial transaction. For example, the user may access a loan or donation application at step 1002. The application may include a series of electronic forms (e.g., web pages) to be filled out by the user and submitted for approval. At step 1004, a determination is made whether the transaction is a public or private transaction. In some embodiments, users may designate specific transactions as public or private. In some embodiments, the financial application itself may also determine whether a transaction is public or private. For example, charitable contributions may always be designated as public transactions whereas personal loans may always be designated as private transactions.

[0133] At step 1006, a publication group is determined. For example, all users or nodes meeting or exceeding a minimum threshold connectivity value and/or not exceeding a maximum threshold connectivity value with the first user may be added to the publication group. As another example, all nodes or users meeting or exceeding some minimum threshold path weight and/or not exceeding a maximum threshold path weight to the first user may be added to the publication group. In some embodiments, the first user is given an opportunity to select the publication group or groups to which the user wants transaction information to be published. For example, the user may specify custom connectivity value maximum/minimum thresholds, custom path weight maximum/minimum thresholds, or both. This threshold value (or values) may then be used to determine the appropriate publication group. The user may also be given an opportunity to view a listing of publication group members, add additional members, and remove existing members, if desired.

[0134] In some embodiments, publication groups may be further refined using additional information known about other nodes or users in the network. For example, a first user may initiate a donation transaction for a wildlife refuge. In determining the appropriate publication group, nodes with high connectivity values with a known wildlife affinity or

support group may be automatically added to the publication group, whether or not they meet the path weight or connectivity threshold values. Application server 106 (FIG. 1) may automatically compare attribute flags and other metadata associated with the financial application (for example, stored in the description field in financial application table 544 (FIG. 5C)) with attributes known about other nodes or users in the network and use the results of this comparison in adding additional members to, or removing otherwise qualifying members from, publication groups. For example, "LIKE" and "DISLIKE" flags (as described above with regard to FIG. 5C) may be read from financial application table 544 (FIG. 5C) and used to refine publication group membership using information other than (or in addition to) connectivity values and path weights. Users matching a "LIKE" flag may be automatically added to the publication group whether or not they meet one or more threshold values in some embodiments. In other embodiments, users or nodes must both match any defined "LIKE" flag and meet applicable threshold values in order to be added to a publication group. Similarly, users matching a "DISLIKE" flag may be automatically removed from the publication group even if they meet one or more threshold values in some embodiments.

[0135] At step 1008, transaction information may be published to the selected publication group or groups. Publication may take a variety of forms, including email messages, text messages, voicemails, listings on a homepage, listings on a profile page, listings on a shared-access or community page, postings to a discussion forum, notification messages, other suitable notifications, or any combination of the foregoing. The type of notifications may be dependent on the active sign-in profile, in some embodiments. For example, if the active sign-in profile is for an email account provider, at least some of the notifications may take the form of email messages. If the active sign-in profile is for a social networking service provider, at least some of the notifications may take the form of provider notifications, wall postings, profile page postings, or the like.

[0136] At step 1010, a determination is made whether a second user (e.g., a member of the publication group) has accessed the same financial application. In some embodiments, the second user may access the same financial application directly from the publication. For example, a published notification may include a link (e.g., hyperlink) to the financial application. The second user may directly access the financial application by activating the link (e.g., by clicking or selecting the link). In some embodiments, at least some of the information from the first user's financial transaction is automatically carried over to the

second user's transaction, allowing the second user to efficiently execute a partly or wholly-identical transaction as the first user. For example, if the transaction is a donation, the donation amount (or more generically the principal) from the first user's transaction may be pre-populated in the electronic forms associated with the second user's transaction. In that way, users may be encouraged to donate (or borrow) the same amount as the first user. In some embodiments, users are not allowed to change pre-populated information (e.g., so as to encourage a minimum level of charitable giving). In other embodiments, pre-populated information may be changed by the user. If at step 1010 the second user does access the same financial application, a new financial transaction may be processed on behalf of the second user at step 1012. If applicable, a repayment schedule may also be automatically generated at step 1014. For example table 546 may be automatically populated, if the financial transaction is a loan.

[0137] In processing financial transactions, connectivity values may be used to determine eligibility of the lender, borrower, or both (in the case of a loan transaction). For example, eligible borrowers may need to meet a threshold connectivity value with the lender, the lending institution, one or more officers or directors of the lending institution, or any combination of the foregoing. In addition, as described above, third-party processes may make automatic transaction decisions based, at least in part, the connectivity values. For example, in some embodiments, at least three threshold network connectivity values may be defined, N_1, N_2, and N_3, where $N_1 > N_2 > N_3$. Potential borrowers may be automatically approved for the financial transaction if they meet the threshold network connectivity value N_1. If borrowers fail to meet the threshold network connectivity value N_1, but meet threshold network connectivity value N_2, then a composite score based on the actual network connectivity value and a third-party ratings agency (such as a credit ratings bureau score) may be used to determine the approval status for the financial transaction. If potential borrowers do not meet threshold network connectivity value N_2, but meet threshold network connectivity value N_3, these potential borrowers may be referred for manual processing. If potential borrowers do not meet threshold network connectivity value N_3, these potential borrowers may be automatically denied participation in the financial transaction. The values of N_1, N_2, and N_3 may be specified by the lending institution, an officer of the lending institution, or the financial application.

[0138] In practice, one or more steps shown in process 1000 may be combined with other steps, performed in any suitable order, performed in parallel (e.g., simultaneously or substantially simultaneously), or removed. In some embodiments, process 1000 may be used to facilitate other transactions, such as identity assessments, security risk assessments, or any other transaction that can take advantage of user connectivity values.

[0139] In some embodiments, virtual and/or electronic currency systems based on network connectivity and/or trust values may be used to facilitate transactions. These virtual currency systems may provide means for facilitating transactions between a node in a network and another node within the same network, in a different network, or not in a network at all. In some embodiments, a single virtual currency system may be associated with a particular network, or may be associated with a plurality of networks. For example, different networks may each be associated with the same or different virtual currency systems. The virtual/electronic currencies in these virtual currency systems may be in the form of points, or any other suitable markers or units. A node may exchange these virtual/electronic currency units for goods, services, or other currencies, with other nodes within the same network, in a different network, or not in a network at all. In an embodiment, a node in a virtual currency system may be implemented substantially similar to node 302 in a distributed storage/computation network such as distributed storage/computation network 300 (FIG. 3).

[0140] In some embodiments, each virtual currency system may be linked to other currency systems via one or more exchange rates. These other currency systems may be implemented on any suitable combination of software and hardware, such as the hardware used to implement distributed storage/computation network 300 (FIG. 3). For example, a unit of a virtual currency may be valued at some fraction or multiple of a different virtual currency. In some embodiments, a unit of a virtual currency may be valued at some fraction or multiple of a different currency system, such as a currency backed and/or issued by a government, nation, or other political, business, or other entity. The exchange rate between a particular virtual currency and some other currency may be static or dynamic. For example, the exchange rate between a virtual currency and another currency may be set at a static value, periodically reset to different static values, or varied continuously. The exchange rate may be determined based upon the values or one or more parameters, such as the network connectivity of one or more nodes in the network, the exchange rate(s) between other currencies, or any other parameter.

[0141] In certain embodiments, one or more nodes within a network with a virtual currency system may issue a virtual currency. For example, a node in a network may generate and/or distribute virtual currency units for facilitating transactions within the network. In some embodiments, an entity at least partially responsible for the administration of a network with a virtual currency system may issue, generate, and/or distribute virtual currency units. Optionally, each node in a network with a virtual currency system may be able to generate virtual currency units. Different nodes may generate different types of virtual currency units, or different nodes may generate the same type of virtual currency units.

[0142] Virtual currencies in a network community may be generated and/or distributed based on network connectivity/trust values between different nodes within the network community. In some embodiments, a node may be provided with virtual currency units at a rate commensurate with one or more network connectivity/trust values associated with that node. In some embodiments, a network connectivity/trust values may be calculated or computed according to the connectivity value calculation or computation described with respect to equations (1) through (5), the network connectivity value calculation or computation described with respect to equation (7), the connectivity value calculation or computation between two nodes described with respect to equations (8) through (12), the calculation or computation of connectivity statistics described with respect to equations (13) through (16), or any suitable combination of these calculations or computations. In some embodiments, these calculations and computations for determining connectivity information may be performed in a distributed fashion across the processors in distributed graph storage/computation system 300 (FIG. 3). In some embodiments, the processing circuitry included in application server 106 may perform any of the calculations and computations in connection with determining network connectivity.

[0143] For example, a node with a large composite network connectivity value may be provided with or accumulate virtual currency units at a greater rate than a node with a small composite network connectivity value. As another example, a node with larger incoming network connectivity/trust values may be provided with or accumulate virtual currency units at a greater rate than a node with smaller incoming network connectivity/trust values. Optionally, the rates at which virtual currency is provided/accumulated may also vary based on outgoing network connectivity/trust values. In some embodiments, the amount of virtual currency accumulated or currently possessed by a node may be directly associated to its

current network connectivity/trust values. For example, if a node gains (or loses/spends) virtual currency units, one or more of its network connectivity/trust values may increase (or decrease). In some embodiments, the network connectivity/trust values of a node may be able to increase as a result of gaining virtual currency units, but may not be able to decrease as a result of losing or spending virtual currency units. Similarly, in other embodiments, the network connectivity/trust values of a node may be able to decrease as a result of losing/spending virtual currency units, but may not be able to increase as a result of gaining virtual currency units.

[0144] In some embodiments, nodes and/or entities associated with a network may generate and/or distribute virtual currency instruments with static and/or dynamic virtual currency values. For example, a node may be able to generate a virtual currency instrument for use in facilitating transactions for goods and/or services, where the value of the virtual currency instrument is linked to the current (or future, or past) value of one or more network connectivity/trust values associated with the node.

[0145] Each equation presented above should be construed as a class of equations of a similar kind, with the actual equation presented being one representative example of the class. For example, the equations presented above include all mathematically equivalent versions of those equations, reductions, simplifications, normalizations, and other equations of the same degree.

[0146] The above described embodiments of the invention are presented for purposes of illustration and not of limitation. The following claims give additional embodiments of the present invention.

What is claimed is:

1. A method for supporting a virtual currency system in a network of a plurality of nodes, comprising:

determining at least one network connectivity value associated with a first node in the network; and

providing a first virtual currency for use at least within the network, wherein the first virtual currency is provided based on the at least one network connectivity value.

2. The method of claim 1, wherein the value of a unit of the first virtual currency is based at least in part on the at least one network connectivity value.

3. The method of claim 1, wherein the value of a unit of the first virtual currency is based at least in part on the value of a unit of a different currency.

4. The method of claim 1, further comprising providing a first virtual currency instrument, wherein the first virtual currency instrument is issued by the first node, the value of the first virtual currency instrument is expressed in units of the first virtual currency, and the value of the first virtual currency instrument is determined based at least in part on the at least one network connectivity value.

5. The method of claim 4, further comprising changing the value of the first virtual currency instrument in response to a change in the at least one network connectivity value.

6. The method of claim 1, wherein providing the first virtual currency comprises providing a quantity of the first virtual currency to the first node, and wherein the quantity is determined based on the at least one network connectivity value.

7. The method of claim 1, wherein the value of the at least one network connectivity value is based at least in part on a quantity of the first virtual currency accumulated by the first node.

8. The method of claim 1, further comprising computing the at least one network connectivity value according to the equation

$$t_{network} = \sum t_{path} \times w_{path} ,$$

wherein t_{path} is a user connectivity value for a path with at least one intermediate node between the first node in the network and a second node in the network, and wherein w_{path} is the normalized weight for said path.

9. The method of claim 1, further comprising computing the at least one network connectivity value according to the equation

$$Connectivity(a,b) = \sum_{p \in Paths(a,b)} PathScore(path) ,$$

wherein Paths(a,b) is at least one path between the first node *a* in the network and a second node *b* in the network, and wherein PathScore(path) represents a path score of one of the paths in Paths(a,b).

10. The method of claim 9, further comprising computing Pathscore(path) according to the equation

$$PathScore(path) = g(path) * \prod_{edge \in path} f(w_{edge}) ,$$

wherein w_{edge} is the weight of an edge in one of the parths in Paths(a,b), $f(w\ )$ is defined according to the function

$$f(w\ ) = \begin{Bmatrix} 4, & if\ w < 0.2 \\ 2, & if\ 0.2 \le w < 0.4 \\ 1, & if\ 0.4 \le w < 0.8 \\ 2 & if\ 0.8 \le w < 1.0 \\ 4, & if\ w = 1.0 \end{Bmatrix},$$

and $g(path)$ is defined according to the function

$$g(path) = \begin{Bmatrix} -1, & \exists w_{edge} < .6 \\ 1, & otherwise \end{Bmatrix} .$$

11. A system for supporting a virtual currency in a network of a plurality of nodes, wherein the system comprises a distributed computation network configured to:

determine at least one network connectivity value associated with a first node in the network; and

provide a first virtual currency for use at least within the network, wherein the first virtual currency is provided based on the at least one network connectivity value.

12. The system of claim 11, wherein the value of a unit of the first virtual currency is based at least in part on the at least one network connectivity value.

13. The system of claim 11, wherein the value of a unit of the first virtual currency is based at least in part on the value of a unit of a different currency.

14. The system of claim 11, wherein the distributed computing network is further configured to provide a first virtual currency instrument, wherein the first virtual currency instrument is issued by the first node, the value of the first virtual currency instrument is expressed in units of the first virtual currency, and the value of the first virtual currency instrument is determined based at least in part on the at least one network connectivity value.

15. The system of claim 14, wherein the distributed computing network is further configured to change the value of the first virtual currency instrument in response to a change in the at least one network connectivity value.

16. The system of claim 11, wherein the distributed computing network provides the first virtual currency by providing a quantity of the first virtual currency to the first node, and wherein the quantity is determined based on the at least one network connectivity value.

17. The system of claim 11, wherein the value of the at least one network connectivity value is based at least in part on a quantity of the first virtual currency accumulated by the first node.

18. The system of claim 11, wherein the distributed computing network is further configured to compute the at least one network connectivity value according to the equation

$$t_{network} = \sum t_{path} \times w_{path} ,$$

wherein t_{path} is a user connectivity value for a path with at least one intermediate node between the first node in the network and a second node in the network, and wherein w_{path} is the normalized weight for said path.

19. The system of claim 11, wherein the distributed computing network is further configured to compute the at least one network connectivity value according to the equation

$$Connectivity(a,b) = \sum_{p \in Paths(a,b)} PathScore(path) ,$$

wherein Paths(a,b) is at least one path between the first nodes *a* in the network and a second node *b* in the network, and wherein PathScore(path) represents a path score of one of the paths in Paths(a,b).

20. The system of claim 19, wherein the distributed computing network is further configured to compute Pathscore(path) according to the equation

$$PathScore(path) = g(path) * \prod_{edge \in path} f(w_{edge}) ,$$

wherein w_{edge} is the weight of an edge in one of the parths in Paths(a,b), $f(w\ )$ is defined according to the function

$$f(w\ ) = \begin{Bmatrix} 4, & if\ w < 0.2 \\ 2, & if\ 0.2 \le w < 0.4 \\ 1, & if\ 0.4 \le w < 0.8 \\ 2 & if\ 0.8 \le w < 1.0 \\ 4, & if\ w = 1.0 \end{Bmatrix},$$

and $g(path)$ is defined according to the function

$$g(path) = \begin{Bmatrix} -1, & \exists w_{edge} < .6 \\ 1, & otherwise \end{Bmatrix} .$$

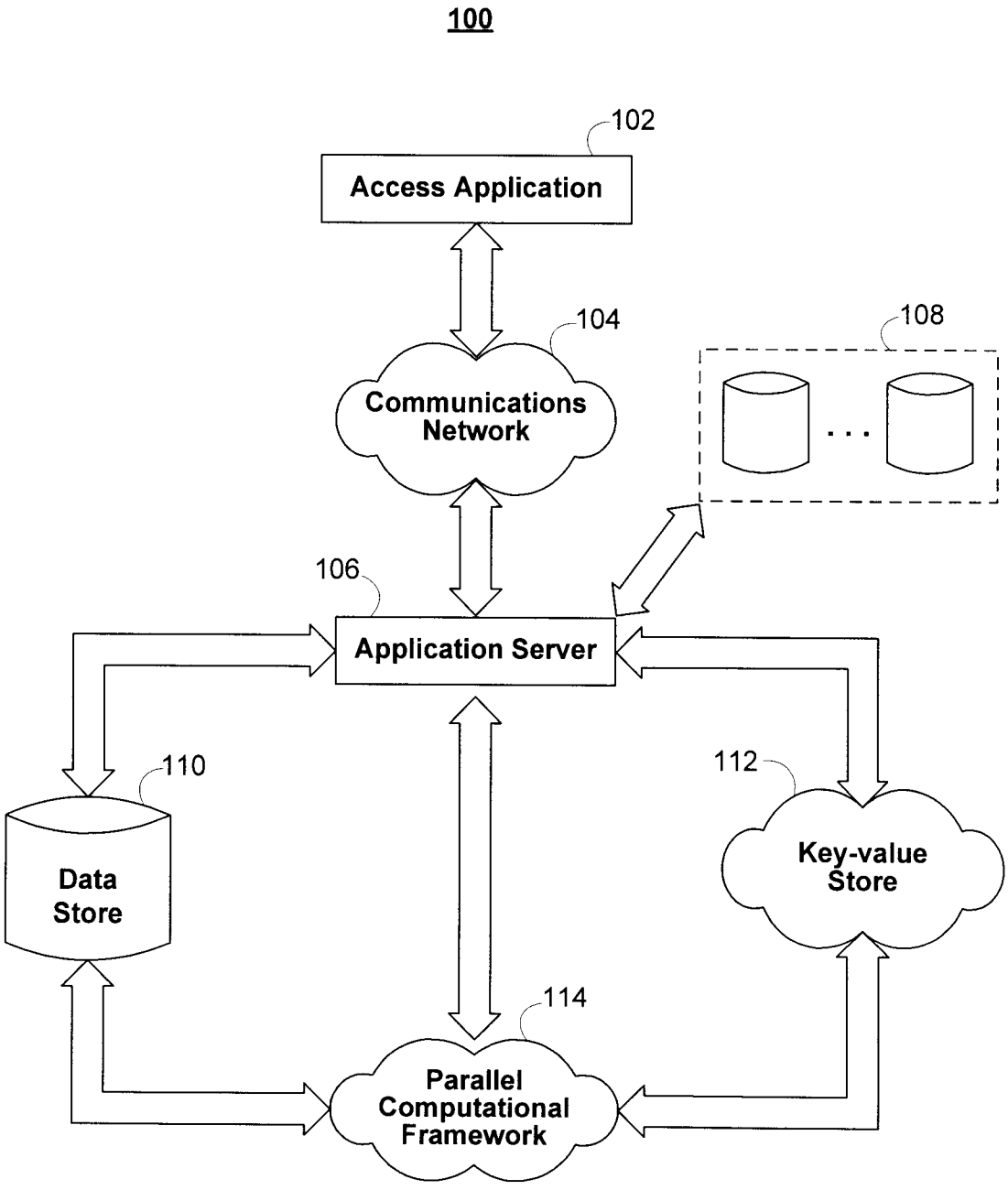
100
102
Access Application
104
Communications Network
108
106
Application Server
110
Data Store
112
Key-value Store
114
Parallel Computational Framework

FIG. 1

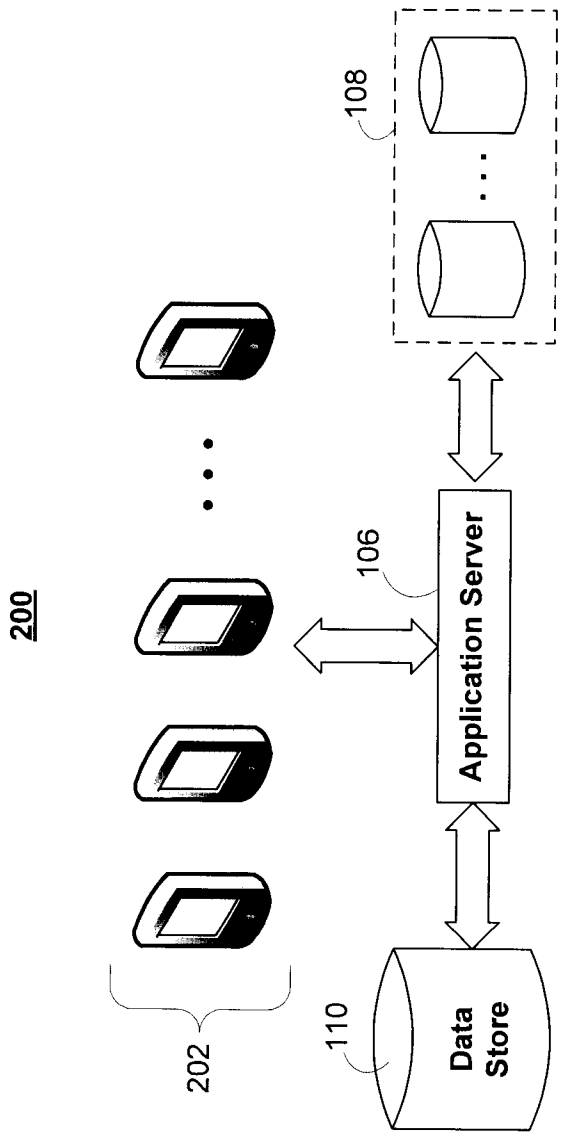

FIG. 2

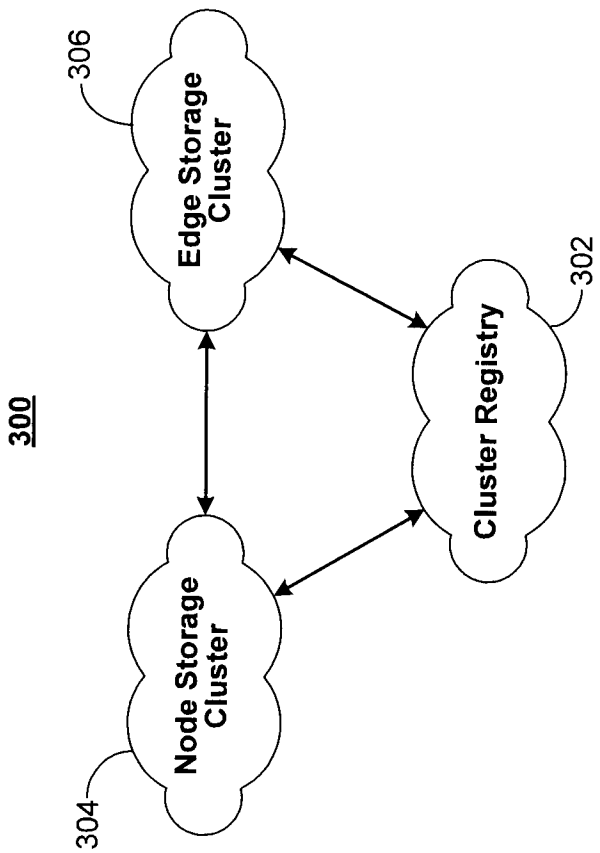
300
306
Edge Storage Cluster
302
Cluster Registry
304
Node Storage Cluster

FIG. 3

400

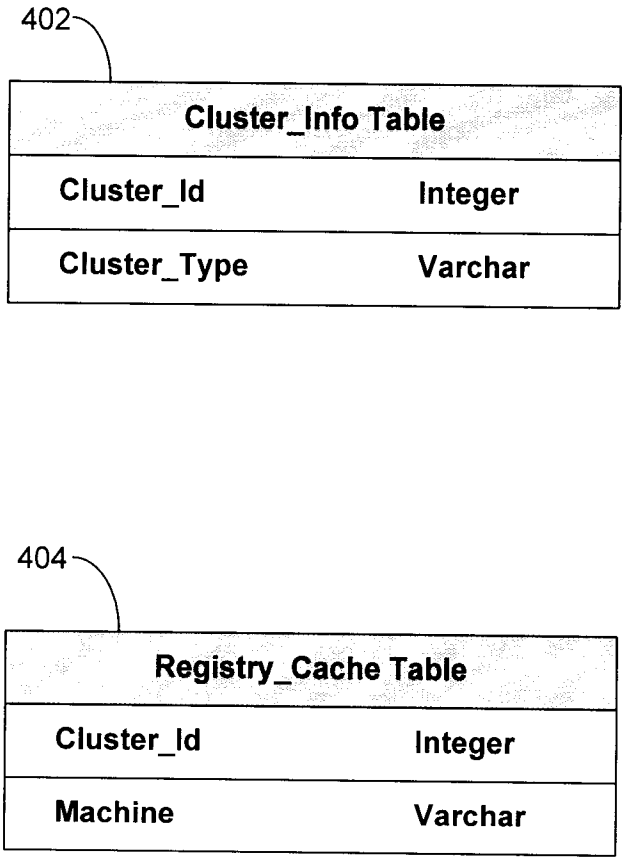
402
Cluster_Info Table
Cluster_Id
Integer
Cluster_Type
Varchar
404
Registry_Cache Table
Cluster_Id
Integer
Machine
Varchar

FIG. 4A

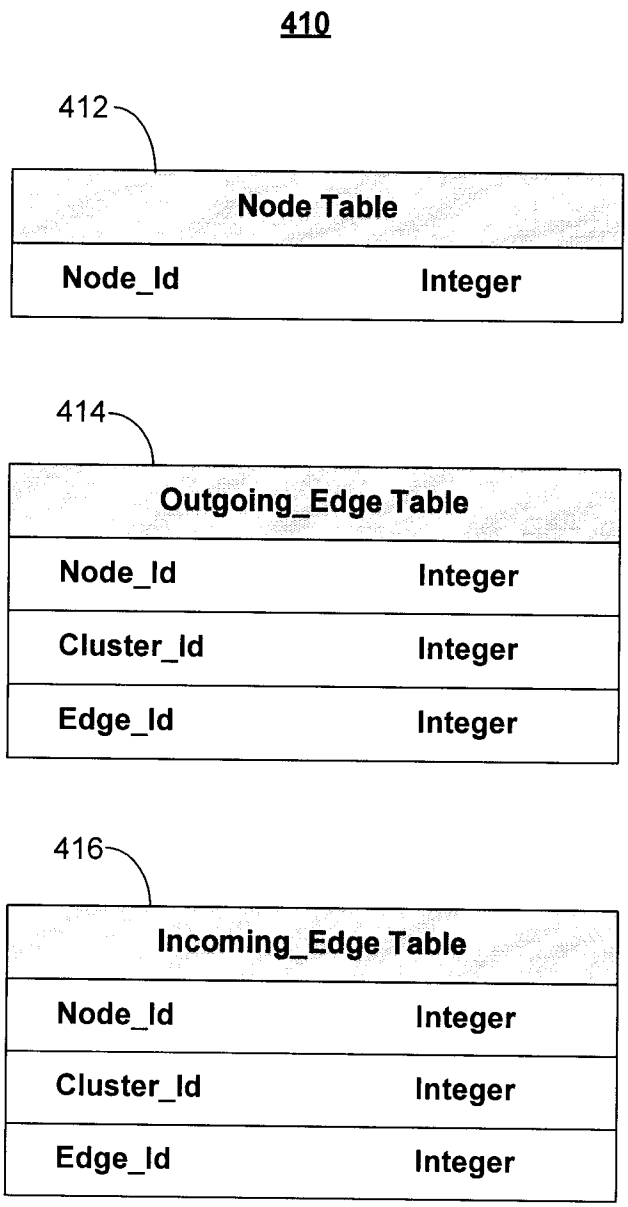
410
412
Node Table
Node_Id
Integer
414
Outgoing_Edge Table
Node_Id
Integer
Cluster_Id
Integer
Edge_Id
Integer
416
Incoming_Edge Table
Node_Id
Integer
Cluster_Id
Integer
Edge_Id
Integer

FIG. 4B

420

422

Edge Table	
Edge_Id	Integer
Tail_Cluster_Id	Integer
Tail_Node_Id	Integer
Head_Cluster_Id	Integer
Head_Node_Id	Integer

FIG. 4C

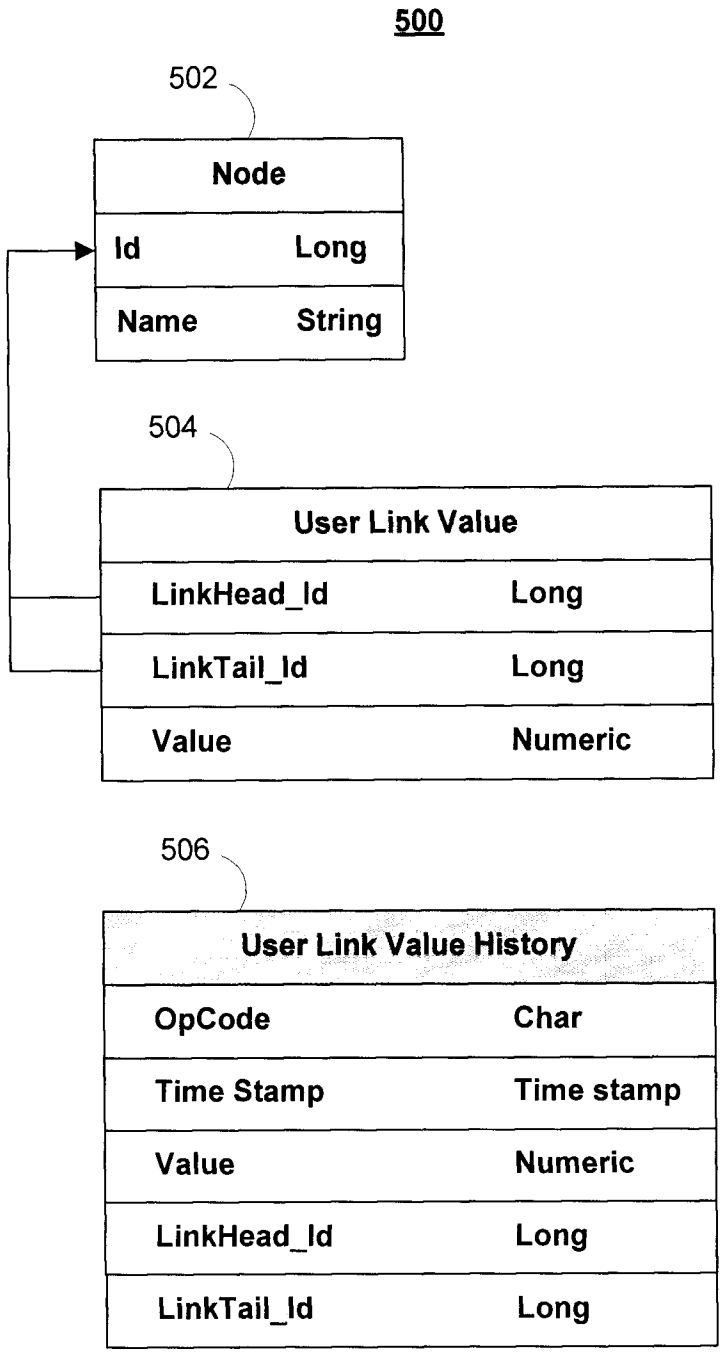
500
502
Node
Id Long
Name String
504
User Link Value
LinkHead_Id Long
LinkTail_Id Long
Value Numeric
506
User Link Value History
OpCode Char
Time Stamp Time stamp
Value Numeric
LinkHead_Id Long
LinkTail_Id Long

FIG. 5A

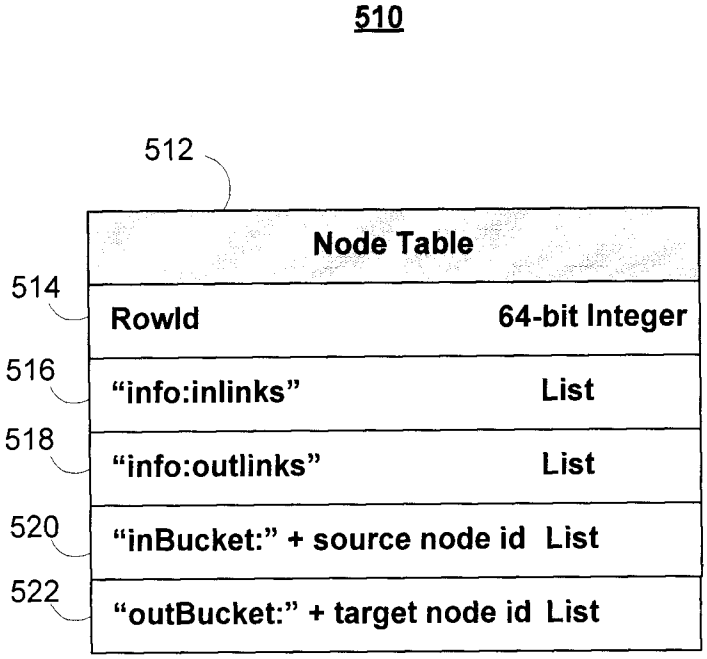
510
512
Node Table
514
RowId
64-bit Integer
516
"info:inlinks"
List
518
"info:outlinks"
List
520
"inBucket:" + source node id
List
522
"outBucket:" + target node id
List

FIG. 5B

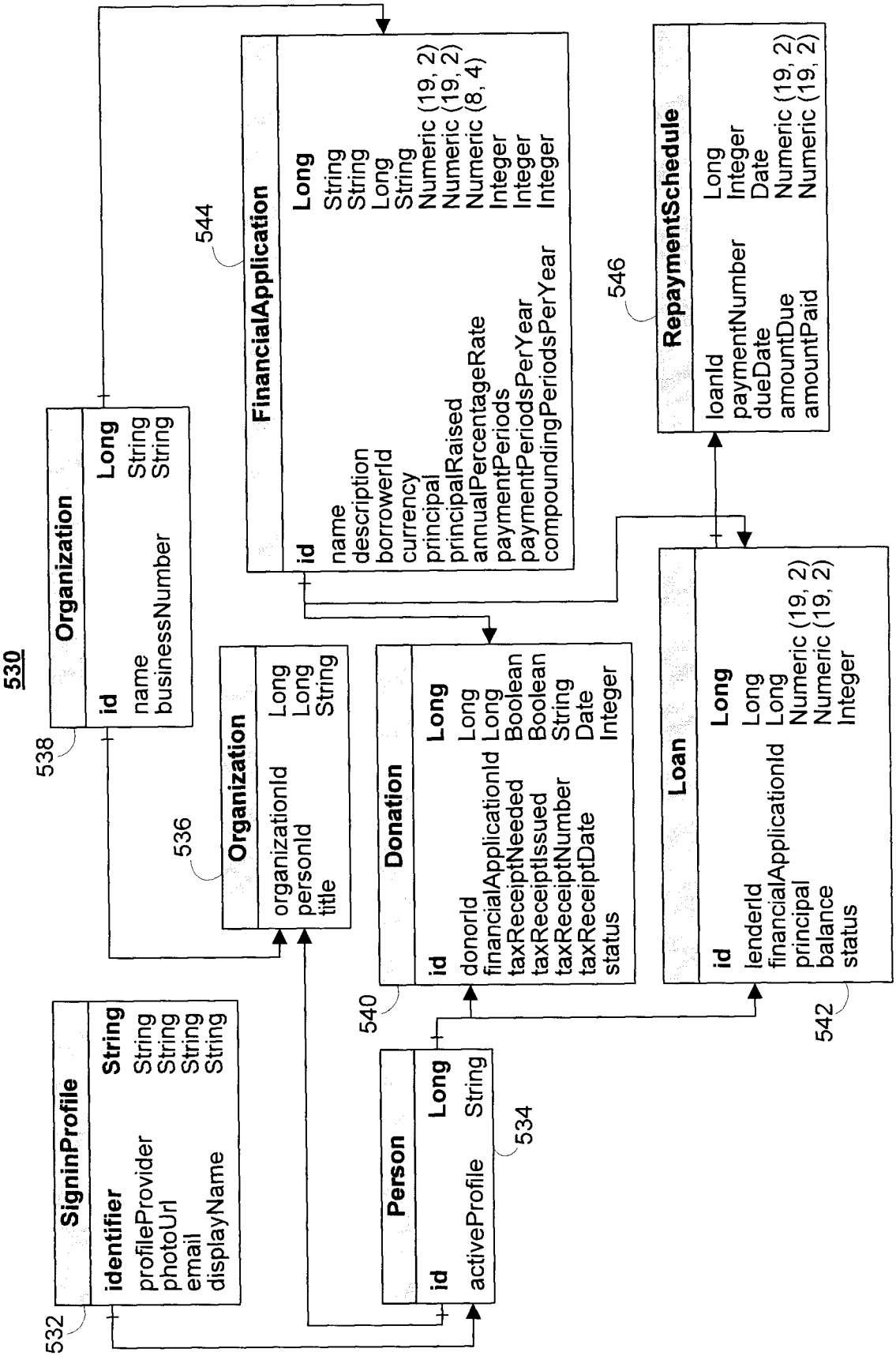
530
SigninProfile
identifier String
profileProvider String
photoUrl String
email String
displayName String
532
Organization
id Long
name String
businessNumber String
538
FinancialApplication
id Long
name String
description String
borrowerId Long
currency String
principal Numeric (19, 2)
principalRaised Numeric (19, 2)
annualPercentageRate Numeric (8, 4)
paymentPeriods Integer
paymentPeriodsPerYear Integer
compoundingPeriodsPerYear Integer
544
RepaymentSchedule
loanId Long
paymentNumber Integer
dueDate Date
amountDue Numeric (19, 2)
amountPaid Numeric (19, 2)
546
Organization
organizationId Long
personId Long
title String
536
Donation
id Long
donorId Long
financialApplicationId Long
taxReceiptNeeded Boolean
taxReceiptIssued Boolean
taxReceiptNumber String
taxReceiptDate Date
status Integer
540
Loan
id Long
lenderId Long
financialApplicationId Long
principal Numeric (19, 2)
balance Numeric (19, 2)
status Integer
542
Person
id Long
activeProfile String
534

FIG. 5C

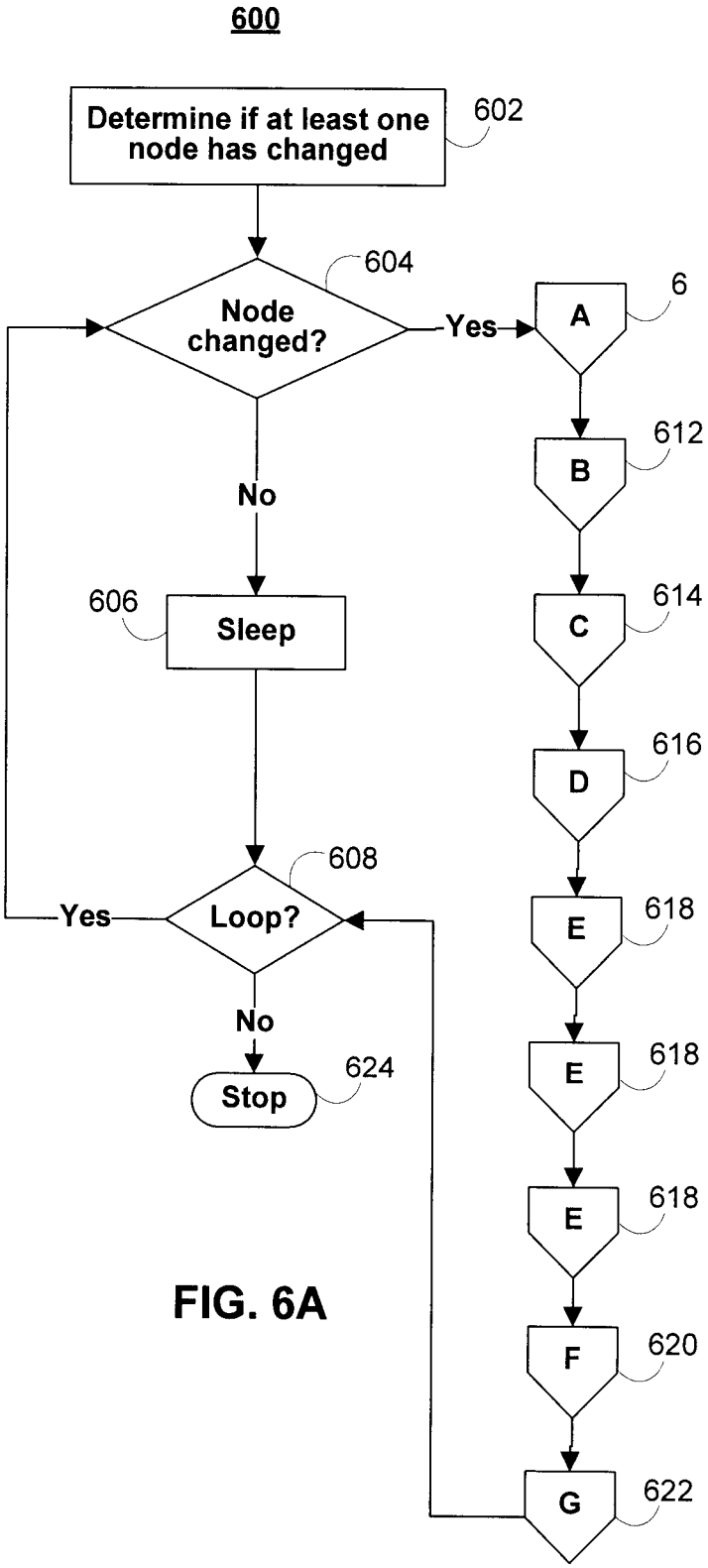
600
Determine if at least one node has changed
602
Node changed?
604
Yes
No
A
6
B
612
C
614
D
616
E
618
E
618
E
618
F
620
G
622
606
Sleep
Loop?
608
Yes
No
Stop
624

FIG. 6A

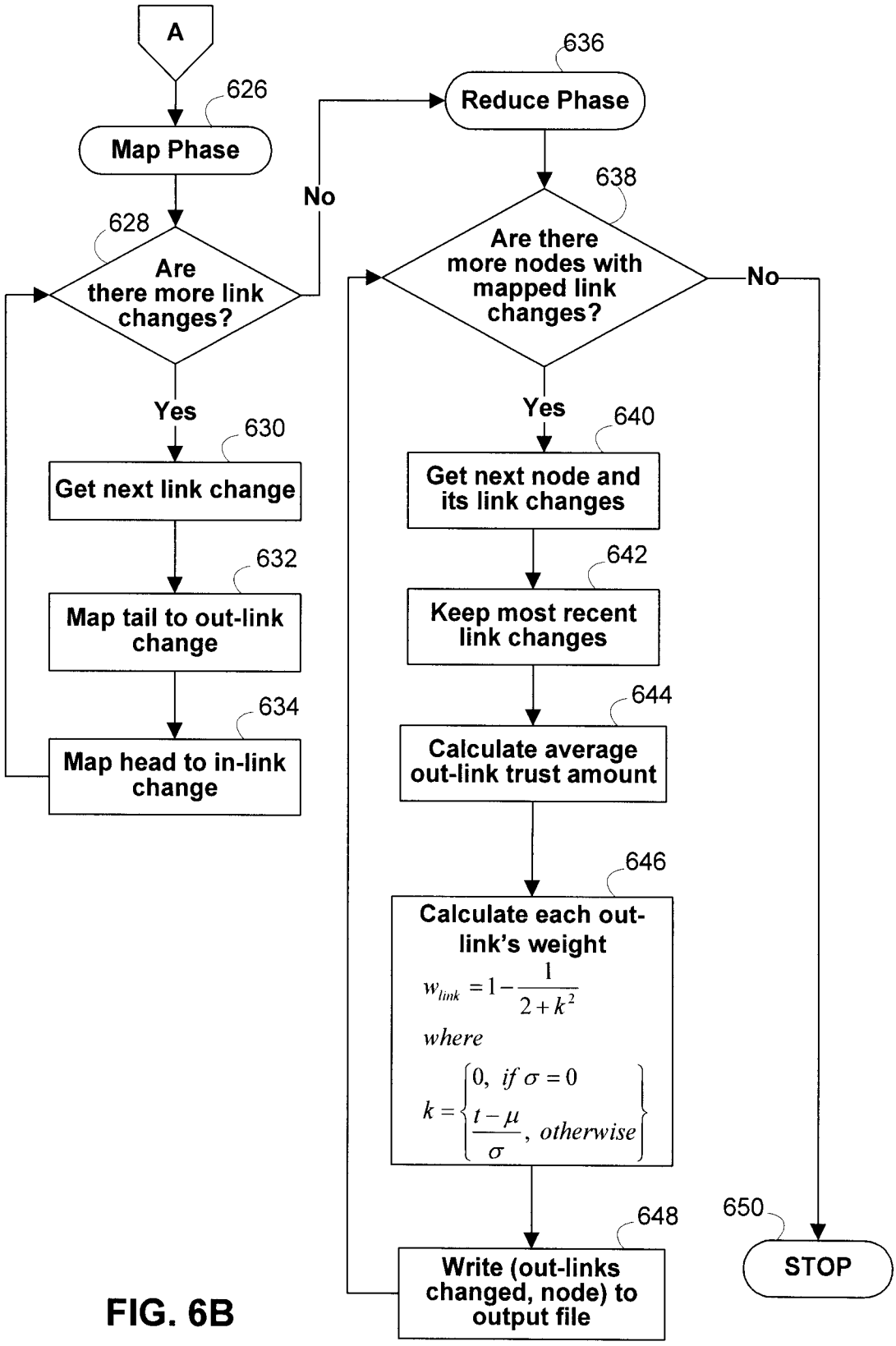
A
626
Map Phase
628
Are there more link changes?
No
Yes
630
Get next link change
632
Map tail to out-link change
634
Map head to in-link change
636
Reduce Phase
638
Are there more nodes with mapped link changes?
No
Yes
640
Get next node and its link changes
642
Keep most recent link changes
644
Calculate average out-link trust amount
646
Calculate each out-link's weight
$w_{link} = 1 - \frac{1}{2 + k^2}$
where
$k = \begin{cases} 0, & if\ \sigma = 0 \\ \frac{t - \mu}{\sigma}, & otherwise \end{cases}$
648
Write (out-links changed, node) to output file
650
STOP

FIG. 6B

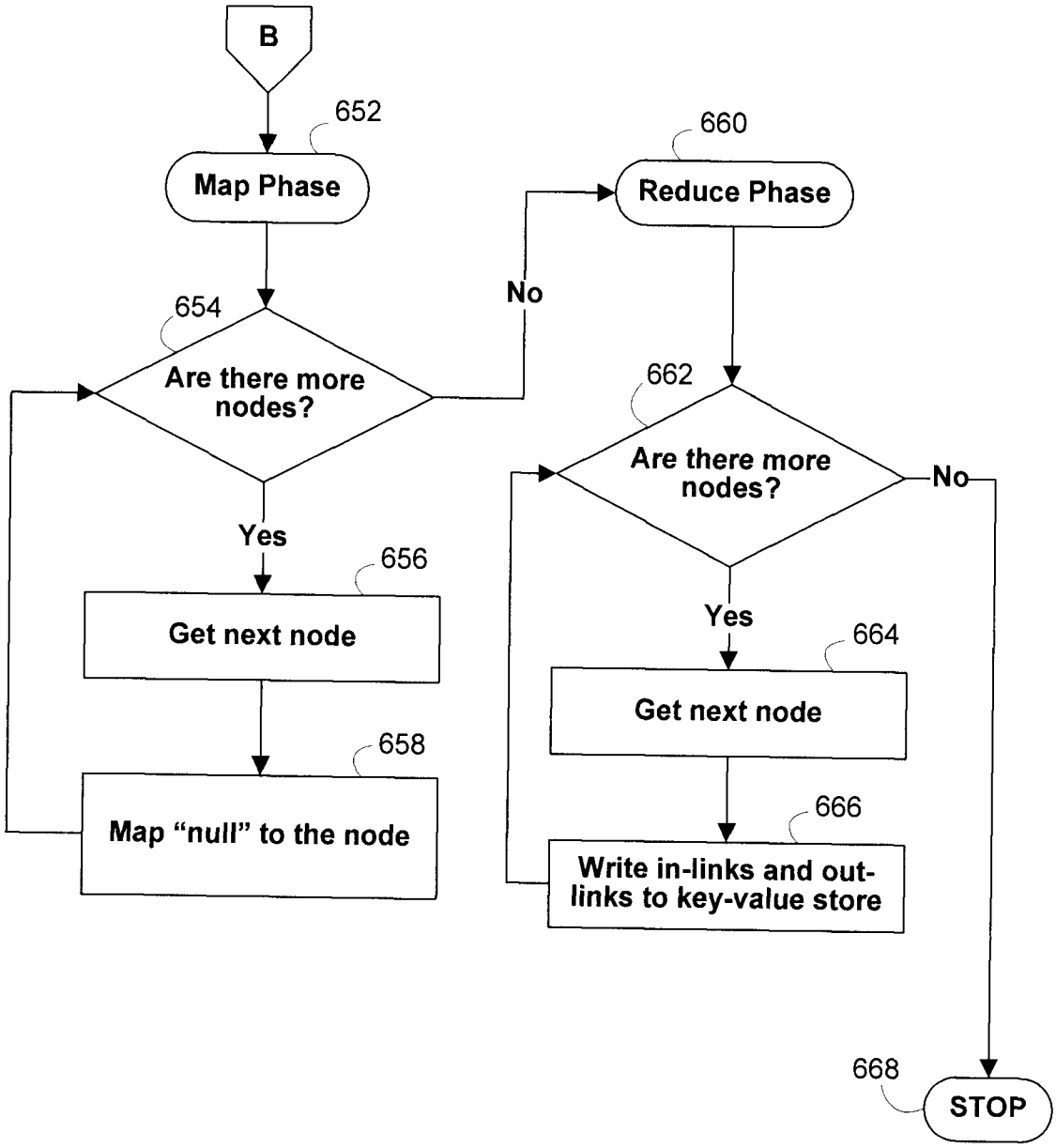
B
652
Map Phase
654
Are there more nodes?
No
Yes
656
Get next node
658
Map "null" to the node
660
Reduce Phase
662
Are there more nodes?
No
Yes
664
Get next node
666
Write in-links and out-links to key-value store
668
STOP

FIG. 6C

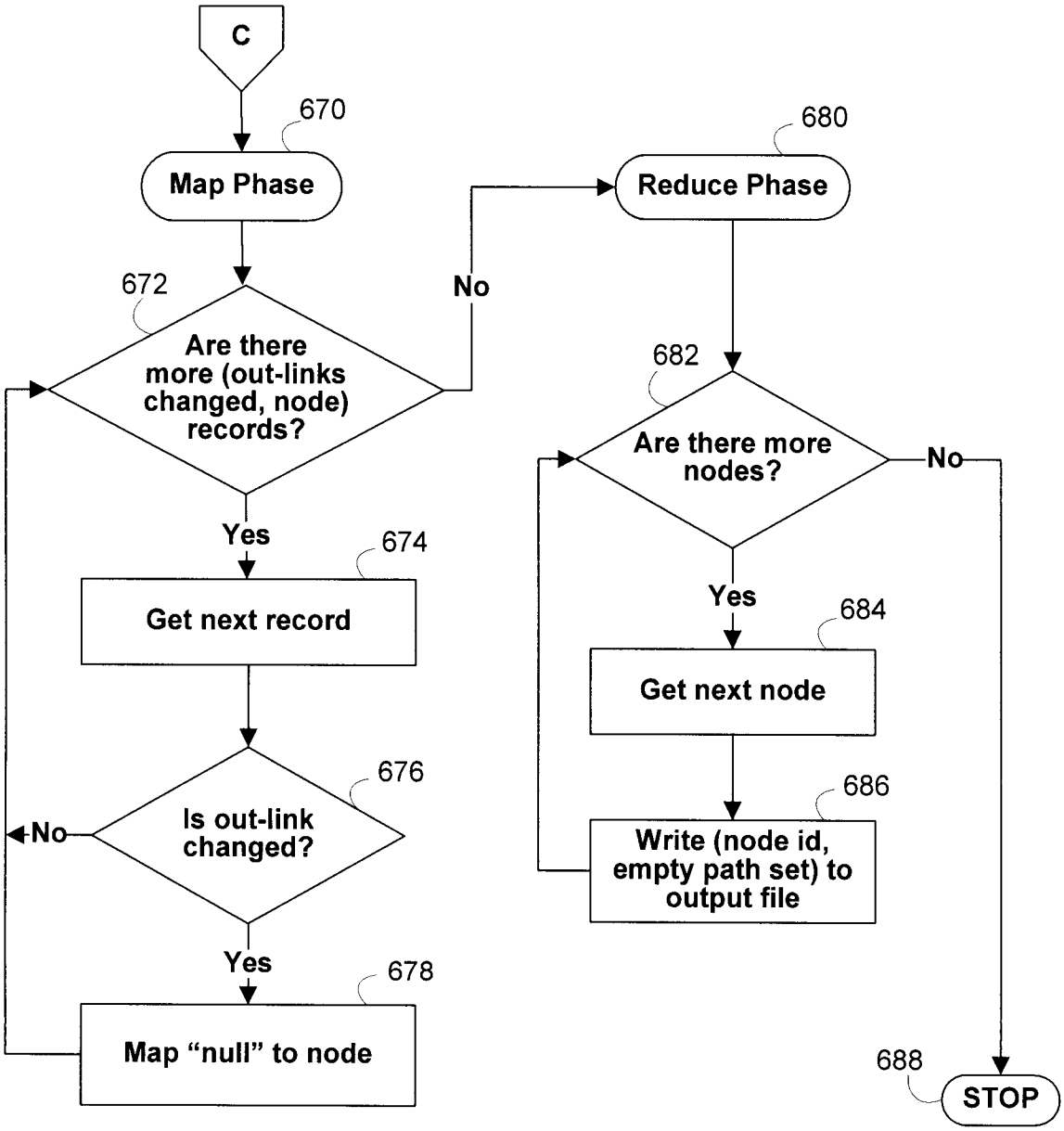
C
670
Map Phase
672
Are there more (out-links changed, node) records?
No
Yes
674
Get next record
676
Is out-link changed?
No
Yes
678
Map "null" to node
680
Reduce Phase
682
Are there more nodes?
No
Yes
684
Get next node
686
Write (node id, empty path set) to output file
688
STOP

FIG. 6D

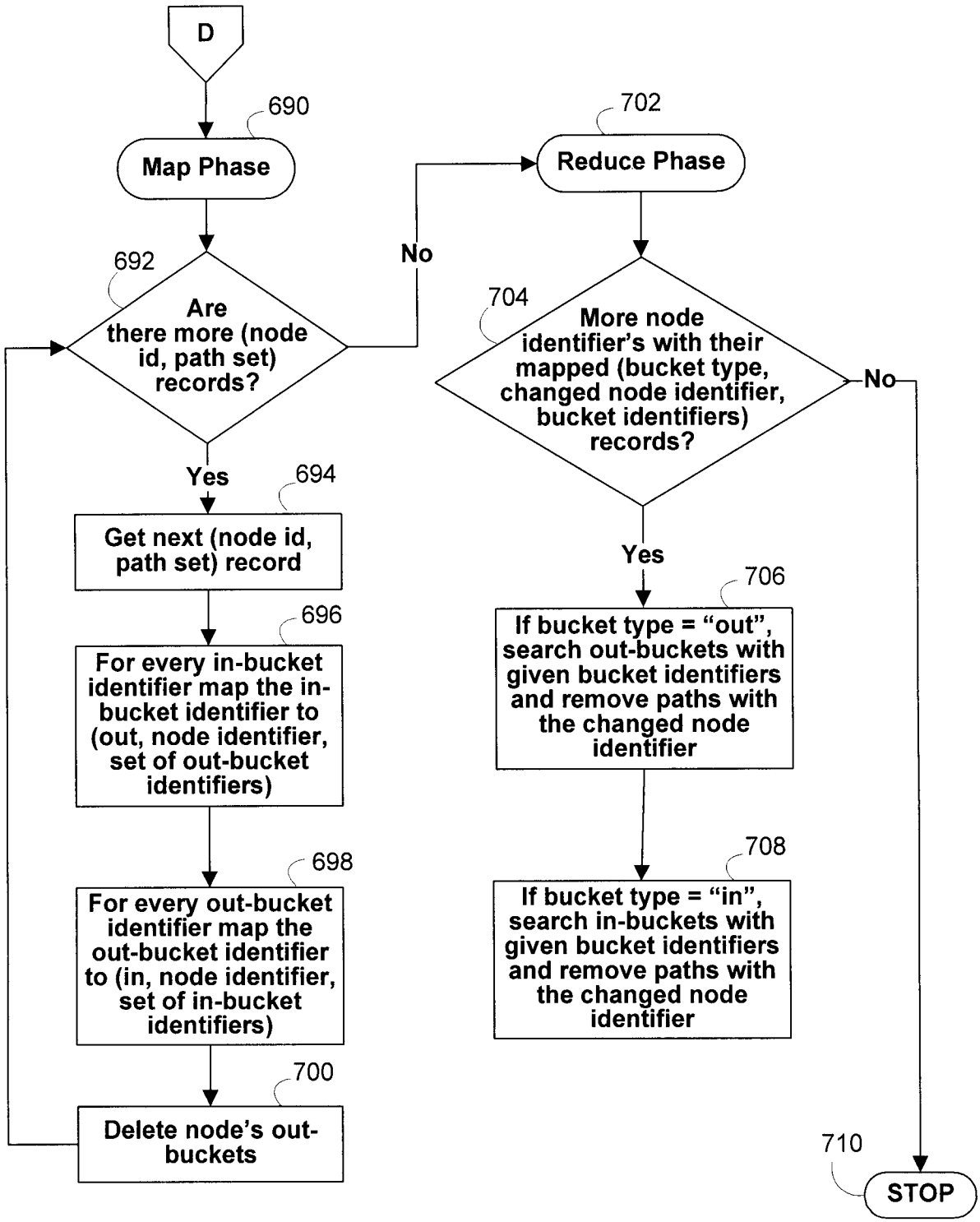
D
690
Map Phase
692
Are there more (node id, path set) records?
No
Yes
694
Get next (node id, path set) record
696
For every in-bucket identifier map the in-bucket identifier to (out, node identifier, set of out-bucket identifiers)
698
For every out-bucket identifier map the out-bucket identifier to (in, node identifier, set of in-bucket identifiers)
700
Delete node's out-buckets
702
Reduce Phase
704
More node identifier's with their mapped (bucket type, changed node identifier, bucket identifiers) records?
No
Yes
706
If bucket type = "out", search out-buckets with given bucket identifiers and remove paths with the changed node identifier
708
If bucket type = "in", search in-buckets with given bucket identifiers and remove paths with the changed node identifier
710
STOP

FIG. 6E

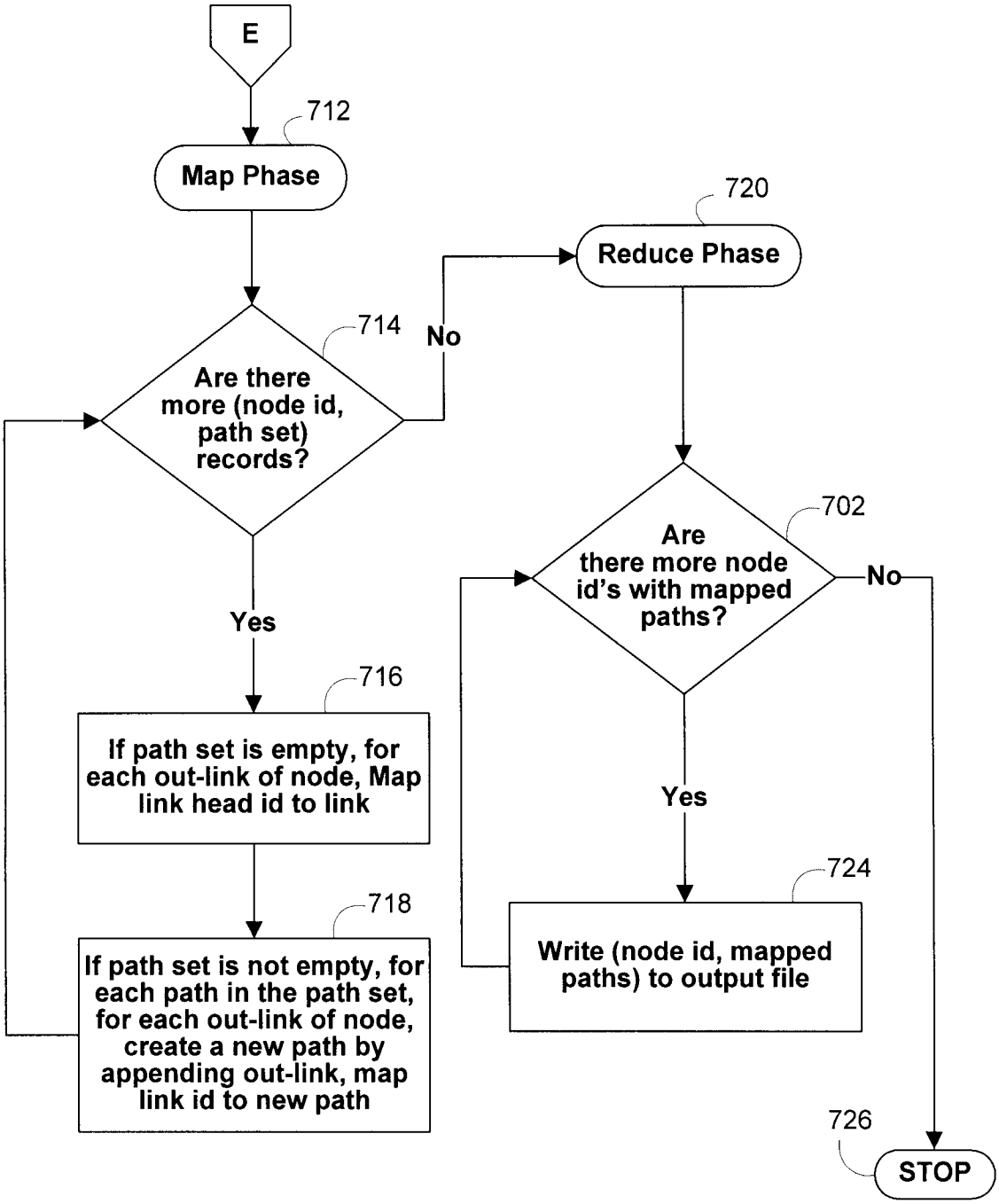
E
712
Map Phase
720
Reduce Phase
714
No
Are there more (node id, path set) records?
Yes
716
If path set is empty, for each out-link of node, Map link head id to link
718
If path set is not empty, for each path in the path set, for each out-link of node, create a new path by appending out-link, map link id to new path
702
Are there more node id's with mapped paths?
No
Yes
724
Write (node id, mapped paths) to output file
726
STOP

FIG. 6F

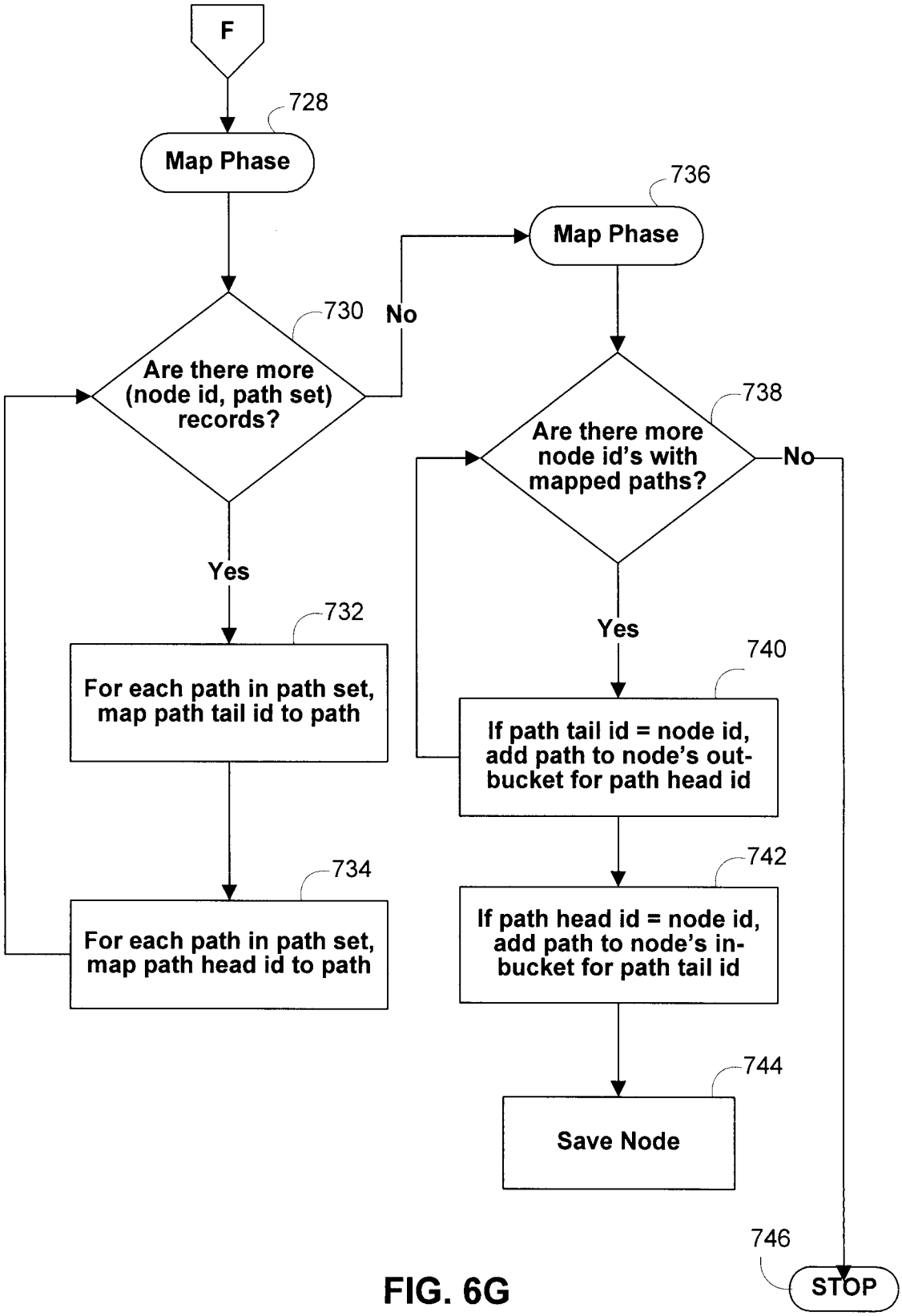
F
728
Map Phase
730
Are there more (node id, path set) records?
No
Yes
732
For each path in path set, map path tail id to path
734
For each path in path set, map path head id to path
736
Map Phase
738
Are there more node id's with mapped paths?
No
Yes
740
If path tail id = node id, add path to node's out-bucket for path head id
742
If path head id = node id, add path to node's in-bucket for path tail id
744
Save Node
746
STOP

FIG. 6G

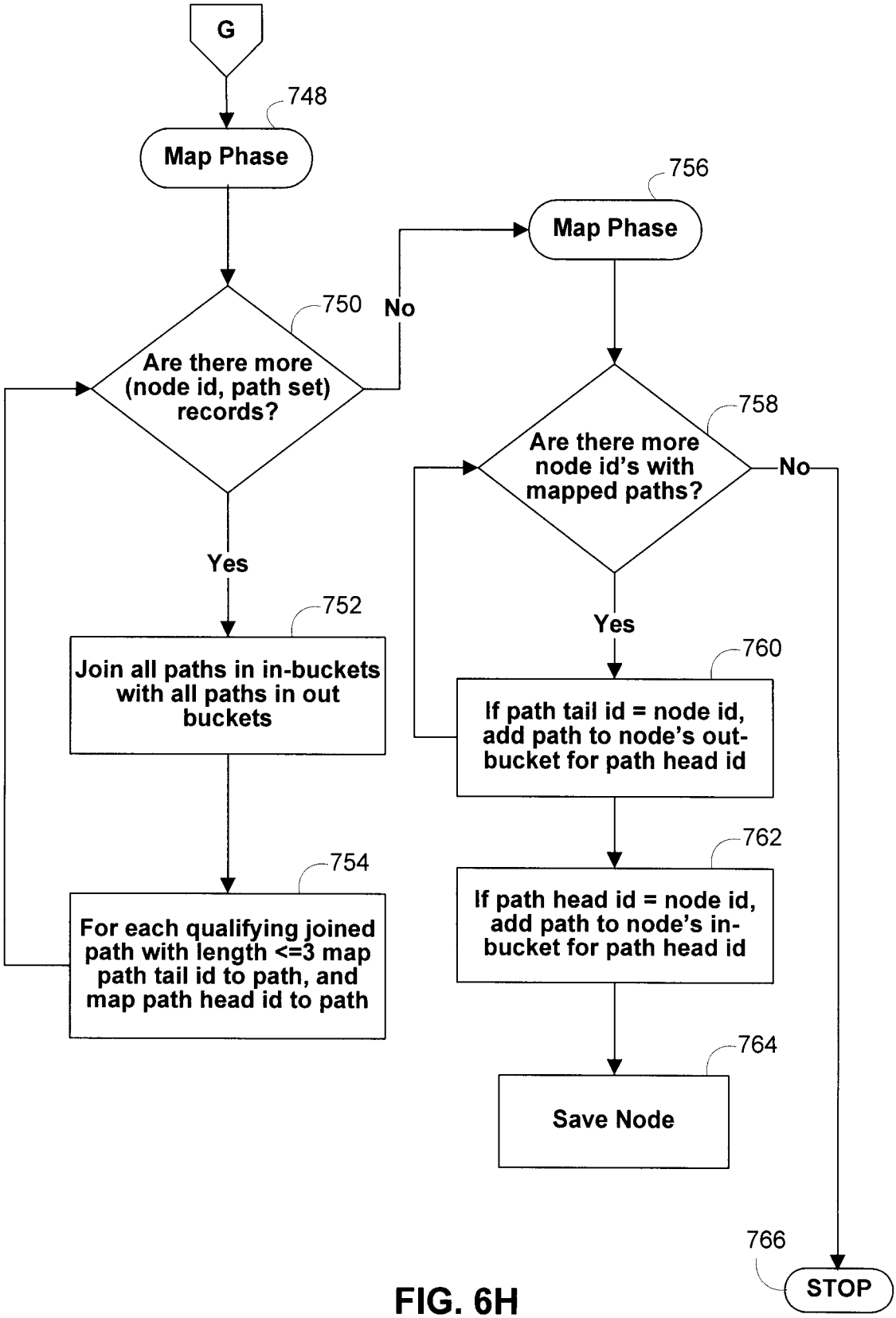
G
748
Map Phase
750
Are there more (node id, path set) records?
No
Yes
752
Join all paths in in-buckets with all paths in out buckets
754
For each qualifying joined path with length <=3 map path tail id to path, and map path head id to path
756
Map Phase
758
Are there more node id's with mapped paths?
No
Yes
760
If path tail id = node id, add path to node's out-bucket for path head id
762
If path head id = node id, add path to node's in-bucket for path head id
764
Save Node
766
STOP

FIG. 6H

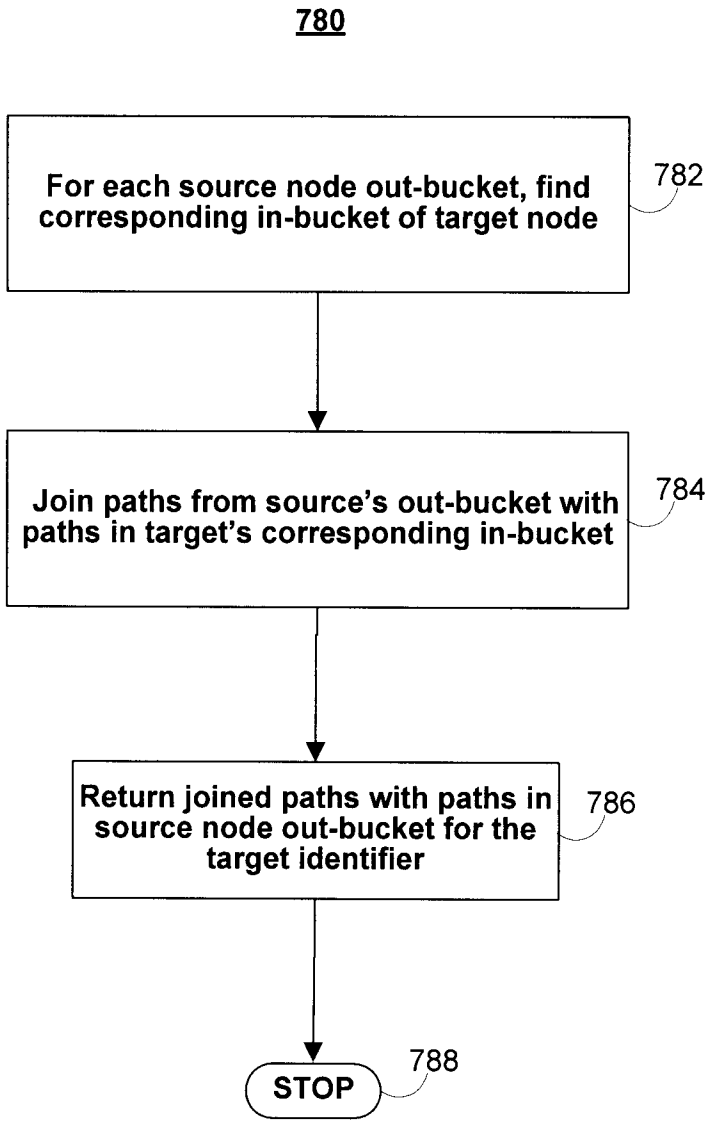
780
For each source node out-bucket, find corresponding in-bucket of target node
782
Join paths from source's out-bucket with paths in target's corresponding in-bucket
784
Return joined paths with paths in source node out-bucket for the target identifier
786
STOP
788

FIG. 7

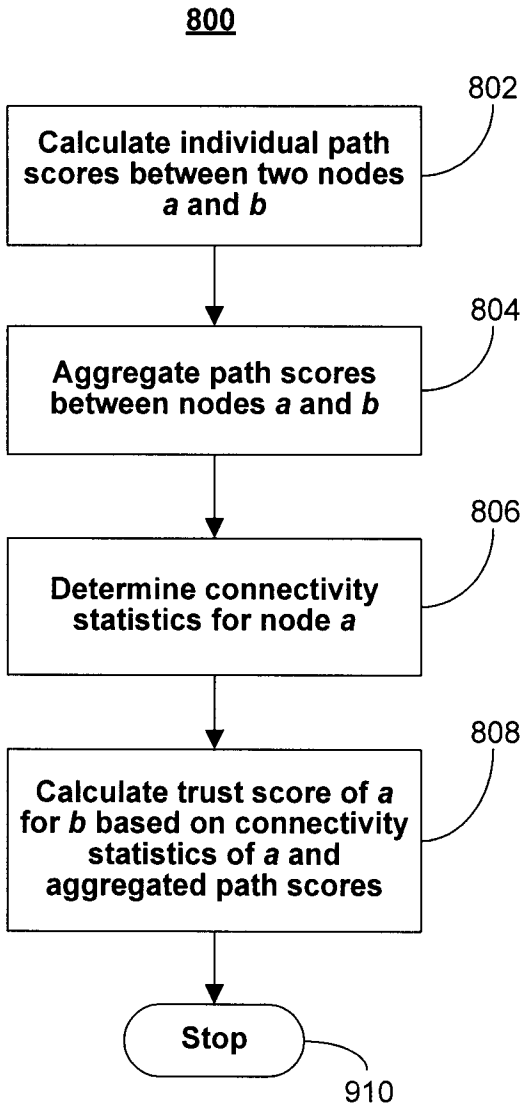
800
802
Calculate individual path scores between two nodes a and b
804
Aggregate path scores between nodes a and b
806
Determine connectivity statistics for node a
808
Calculate trust score of a for b based on connectivity statistics of a and aggregated path scores
Stop
910

FIG. 8

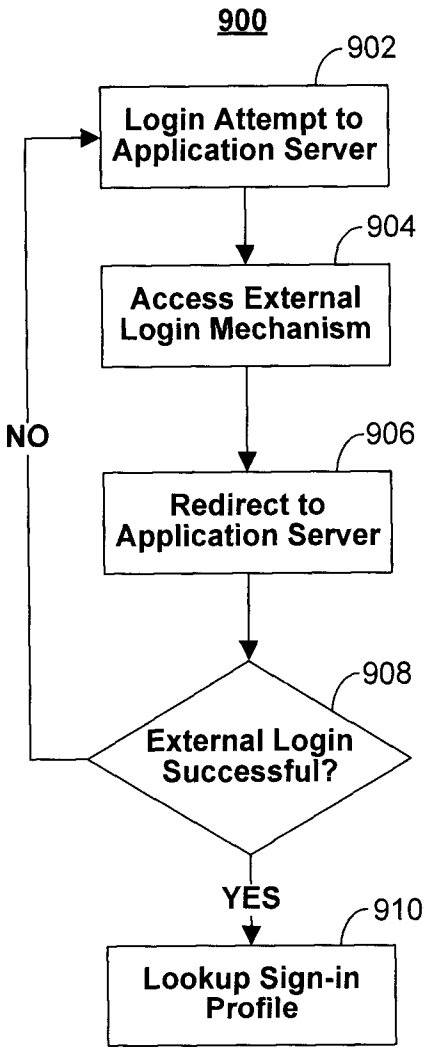
900
902
Login Attempt to Application Server
904
Access External Login Mechanism
906
Redirect to Application Server
908
External Login Successful?
NO
YES
910
Lookup Sign-in Profile

FIG. 9

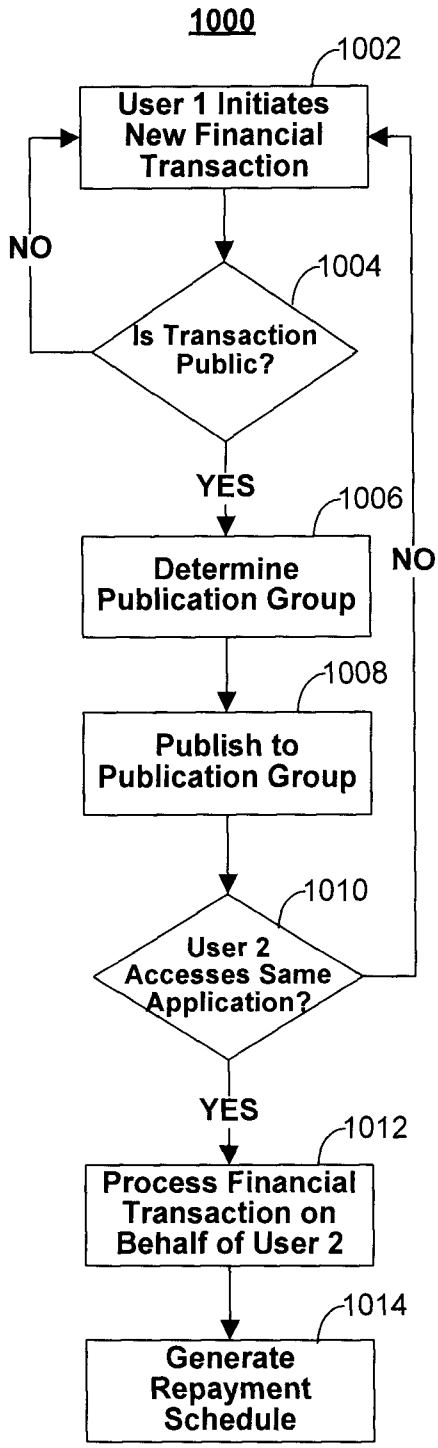
1000
1002
User 1 Initiates New Financial Transaction
NO
1004
Is Transaction Public?
YES
1006
Determine Publication Group
NO
1008
Publish to Publication Group
1010
User 2 Accesses Same Application?
YES
1012
Process Financial Transaction on Behalf of User 2
1014
Generate Repayment Schedule

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2011/050569

A. CLASSIFICATION OF SUBJECT MATTER

IPC: ***H04L 12/26*** (2006.01) , ***G06Q 20/00*** (2006.01) , ***G06Q 30/00*** (2006.01) , ***G06Q 40/00*** (2006.01) , ***H04L 12/16*** (2006.01)

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)

IPC: H04L 12/26 (2006.01), G06Q 20/00 (2006.01), G06Q 30/00 (2006.01), G06Q 40/00 (2006.01), H04L 12/16 (2006.01)

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

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)

Databases: EPOQUE (Epodoc, English Full-text), Canadian Patents Database, IEEEXplore, Google

Keywords: virtual/electronic/pseudo currency, network connectivity/trust, financial transaction, threshold, exchange rate, unit/point/marker/incentive, node

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	GUPTA, R. et al., "Reputation Management Framework and its use as Currency in Large-Scale Peer-to-Peer Networks", Proceedings of the Fourth IEEE International Conference on Peer-to-Peer Computing P2P2004, Zurich, Switzerland, pages 124-132, 25-27 August 2004 (25-08-2004) *pages 124, 125, 127-129*	1-20
Y	MUI, L. et al., "A Computational Model of Trust and Reputation", Proceedings of the 35th Annual Hawaii International Conference on System Sciences, HICSS '02, vol. 7, pages 2431-2439, 7-10 January 2002 (07-01-2002) *sections 2, 3, 5-7*	1-20
A	WALLENTIN, L. et al., "A Cross-Layer Route Discovery Strategy for Virtual Currency Systems in Mobile Ad Hoc Networks", Proceedings of the Seventh International Conference on Wireless On-demand Network Systems and Services IEEE/IFIP WONS 2010, Kranjska Gora, Slovenia, pages 91-98, 3-5 February 2010 (03-02-2010) *whole document*	1-20

[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 application or patent 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
17 November 2011 (17-11-2011)	5 December 2011 (05-12-2011)

Name and mailing address of the ISA/CA
Canadian Intellectual Property Office
Place du Portage I, C114 - 1st Floor, Box PCT
50 Victoria Street
Gatineau, Quebec K1A 0C9
Facsimile No.: 001-819-953-2476

Authorized officer

Daniela Savin (819) 934-4890

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	FELDMAN, M. et al., "Robust Incentive Techniques for Peer-to-Peer Networks", Proceedings of the fifth ACM Conference on Electronic Commerce EC'04, New York, New York, USA, pages 102-111, 17-20 May 2004 (17-05-2004) *whole document*	1-20
A	ZHANG, K. et al., "A Review of Incentive Mechanisms in Peer-to-Peer Systems", Proceedings of the First International Conference on Advances in P2P Systems AP2PS'09, Sliema, Malta, pages 45-50, 11-16 October 2009 (11-10-2009) *whole document*	1-20
A	GUPTA, M. et al., "A Reputation System for Peer-to-Peer Networks", Proceedings of the 13th International Workshop on Network and operating systems support for digital audio and video NOSSDAV'03, Monterey, California, USA, 1-3 June 2003 (01-06-2003) *whole document*	1-20
A	MORI, T. et al., "Improving Deployability of Peer-Assisted CDN Platform with Incentive", Proceedings of IEEE Global Telecommunications Conference GLOBECOM 2009, Honolulu, Hawaii, USA, pages 1-7, 30 November - 4 December 2009 (30-11-2009) *whole document*	1-20
A	ZHANG, Z. et al., "MARCH: A Distributed Incentive Scheme for Peer-to-Peer Networks", Proceedings of the 26th Annual IEEE Conference on Computer Communications INFOCOM 2007, Anchorage, Alaska, USA, pages 1091-1099, 6-12 May 2007 (06-05-2007) *whole document*	1-20
A	US 2005/0096987 A1 (MIYAUCHI) 5 May 2005 (05-05-2005) *paragraphs [0031]-[0094]; fig. 3*	1-20
A	WO 2009/109009 A1 (KOSHY et al.) 11 September 2009 (11-09-2009) *whole document*	1-20

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CA2011/050569

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2005096987A1	05 May 2005 (05-05-2005)	CN1619567A	25 May 2005 (25-05-2005)
		EP1528498A2	04 May 2005 (04-05-2005)
		EP1528498A3	02 August 2006 (02-08-2006)
		JP2005135071A	26 May 2005 (26-05-2005)
		KR20050040805A	03 May 2005 (03-05-2005)
		US7707068B2	27 April 2010 (27-04-2010)
WO2009109009A1	11 September 2009 (11-09-2009)	AU2009221644A1	11 September 2009 (11-09-2009)