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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 March 2016 (10.03.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/036760 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 17/21* (2006.01)

(21) International Application Number:
PCT/US2015/047980

(22) International Filing Date:
1 September 2015 (01.09.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/476,237 3 September 2014 (03.09.2014) US

(71) Applicant: **ATIGEO CORPORATION** [US/US]; 800 Bellevue Way NE, Suite 600, Bellevue, Washington 98004-4286 (US).

(72) Inventors: **BISCA, Radu**; 800 Bellevue Way NE, Suite 600, Bellevue, Washington 98004-4286 (US). **SANDOVAL, Michael**; 800 Bellevue Way NE, Suite 600, Bellevue, Washington 98004-4286 (US). **BARBURA, Claudiu**; 800 Bellevue Way NE, Suite 600, Bellevue, Washington 98004-4286 (US). **KOHN, Wolf**; 800 Bellevue Way NE, Suite 600, Bellevue, Washington 98004-4286 (US). **TALBY, David**; 800 Bellevue Way NE, Suite 600, Bellevue, Washington 98004-4286 (US).

(74) Agent: **BERGSTROM, Robert W.**; Olympic Patent Works PLLC, P.O. Box 4277, Seattle, Washington 98194-0277 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, 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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, 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, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR SEARCHING AND ANALYZING LARGE NUMBERS OF ELECTRONIC DOCUMENTS

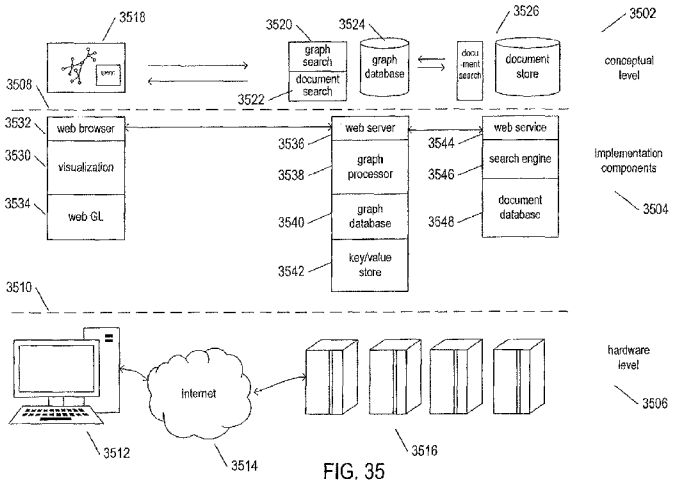


FIG. 35

(57) Abstract: The current document is directed to methods and systems for accessing, searching, analyzing, and visualizing electronically stored information, including electronic documents. These methods and systems construct graph-like representations of information searches that can be visualized and manipulated in three dimensions. The three-dimensional rendering of search results allows for very large numbers of search results to be visualized conveniently using a graphical user interface displayed on an electronic display device. Methods and systems provide for three-dimensional manipulation of graph-like renderings of search results, visualization-assisted searching, and a large number of research tools for discovering and storing various types of links, connections, and relationships between electronically stored information entities.

WO 2016/036760 A1

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

Published:

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

METHOD AND SYSTEM FOR SEARCHING AND ANALYZING LARGE NUMBERS OF ELECTRONIC DOCUMENTS

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation-in-part of Patent Application No. 14/171,681, filed February 3, 2014, which claims the benefit of Provisional Application No. 61/772,467, filed March 4, 2013.

TECHNICAL FIELD

The current document is related to electronic documents, graphical user interfaces, two-dimensional and three-dimensional visual representation of graphs, and, in particular, to a method and system for searching and analyzing electronic documents using graph-like representations of search results and other types of visual representations.

BACKGROUND

The widespread adoption of electronic computer systems and the rapid and ongoing evolution of computer systems over the past 60 years has revolutionized many aspects of modern life and created large new industries and marketplaces. Up through at least the 1970s, governments, corporations, and other large organizations employed armies of typists and file clerks to generate and manage large numbers of paper documents. With the advent of word-processing applications and distributed, database-based, electronic document management systems, much of the former paper-based documents have been replaced with electronic documents. Similarly, it was standard practice, through at least the 1970s, for researchers in science, technology, economics, and other fields to spend hours in libraries to access journal articles and other printed publications in order to research various topics. Various on-line information services became available to supplement library-based research during the 1970s. Beginning in the mid 1990s, a huge amount of information contained in documents and publications was shifted to the Internet, allowing users to access the information through web browsers from document and information sources distributed across the world.

While modern technological advancements have vastly increased the number and accessibility of documents and other stored information, the technological advancements have also resulted in a geometrical growth in the amount of information and stored documents that are available to users of online information services and the Internet. Common, long-used practices of informal browsing and manual note-taking are no longer adequate for searching large numbers of documents and other types of stored information, analyzing electronic documents and other electronically stored information, and attempting to conceptualize many different links, connections, and associations between stored-information entities available from information sources that may provide access to hundreds of thousands, millions, or more electronic documents and other stored information. As a result, those who access and analyze electronically stored information continue to seek methods and systems to facilitate access and analysis of electronically stored information.

SUMMARY

The current document is directed to methods and systems for accessing, searching, analyzing, and visualizing electronically stored information, including electronic documents. These methods and systems construct graph-like representations of information searches that can be visualized and manipulated in three dimensions. The three-dimensional rendering of search results allows for very large numbers of search results to be visualized conveniently using a graphical user interface displayed on an electronic display device. Methods and systems provide for three-dimensional manipulation of graph-like renderings of search results, visualization-assisted searching, and a large number of research tools for discovering and storing various types of links, connections, and relationships between electronically stored information entities.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates information entities and characteristics of information entities to which the currently described methods and systems are applied.

Figure 2 illustrates document sources.

Figures 3A-C illustrate an example document search.

Figures 4A-F illustrate a graph-like rendering of search results provided by the currently described methods and systems.

Figure 5 shows an alternative illustration of the graph-like rendering of the search process and search results illustrated in Figures 4A-F.

Figures 6A-G illustrate details about the three-dimensional display of the graph-like representations of searches and search results as well as basic manipulations of the graph-like renderings available to users through the GUI provided by the currently described methods and systems.

Figures 7A-N illustrate other types of operations and facilities made available to a user through the GUI provided by the currently disclosed methods and systems.

Figures 8A-I illustrate additional operations available to a user of the currently described method and system for manipulation of graph-like representations of search processes and search results.

Figures 9A-B illustrate a search-in-graph operation provided by the currently described methods and systems.

Figures 10A-K illustrate additional features of the GUI provided by the currently discussed methods and systems.

Figures 11A-E illustrate one navigation tool provided by the GUI that is, in turn, provided by the currently discussed methods and systems.

Figures 12A-F illustrate certain elements of a stack-mode display.

Figures 13A-13H illustrate the correspondence between a graph-like representation of a search process and search results and a stack representation of the same search process and search results.

Figure 14 illustrates a third type of representation of search processes and search results referred to as a "heat-map display."

Figure 15 illustrates yet another feature of the GUI provided by the currently discussed methods and systems.

Figure 16 provides a general architectural diagram for various types of computers and other processor-controlled devices.

Figure 17 illustrates generalized hardware and software components of a general-purpose computer system.

Figure 18 illustrates generalized hardware and software components of a general-purpose computer system that includes a virtualization layer.

Figure 19 illustrates an Internet-connected distributed computer system.

Figure 20 illustrates cloud computing.

Figure 21 illustrates electronic communications between a client and server computer.

Figure 22 illustrates the role of resources in RESTful APIs.

Figures 23A-D illustrate four basic verbs, or operations, provided by the HTTP application-layer protocol used in RESTful applications.

Figure 24 provides one example of the data stored within a node object to describe a document node.

Figures 25A-H show data contained in a data script structure to describe an entire graph, in one implementation of the currently described methods and systems.

Figures 26-27 show control-flow diagrams that describe overall operation of one implementation of the currently described methods and systems.

Figure 28 illustrates a graph data object stored by graph databases.

Figure 29 illustrates one logical data-storage model for a graph database that stores graphs, such as the graph illustrated in Figure 28.

Figure 30 shows a simple graph *G* stored in a graph database.

Figure 31 illustrates several path queries.

Figure 32 shows two additional path queries.

Figure 33 shows a number of additional, more complex queries.

Figure 34 illustrates insertion and deletion of graph entities.

Figure 35 illustrates one implementation of the above-described system for searching and analyzing electronic documents using graph-like representations of search results.

Figures 36A-37B illustrate various implementations of an interface provided by the 3D viewer which incorporates graph-database-query functionalities.

Figures 38A-C illustrate the construction of a three-dimensional representation of a search result furnished to the 3D viewer for rendering and display to a user by the remote web server and web service (3536 and 3544, respectively, in Figure 35).

Figure 39 illustrates the implementation of the system for searching and analyzing electronic documents using graph-like representations of search results from a stored-data perspective.

Figures 40A-F illustrate, using control-flow diagrams, interaction of the 3D viewer, web server, and back-end web service to produce a three-dimensional representation of a document-search result to a user of the client computer system.

DETAILED DESCRIPTION

The current document is directed to methods and systems that access, search, analyze, and provide graphical visualization of electronically stored information, including electronic documents. In a first subsection, an example system that accesses, searches, analyzes, and provides graphical visualization of electronically stored information is discussed. In a second subsection, graph databases are discussed. In a third subsection, an implementation of a system that accesses, searches, analyzes, and provides graphical visualization of electronically stored information and that includes a graph database to provide graph-query facilities is discussed.

Example System that Accesses, Searches, Analyzes, and Provides Graphical Visualization of Electronically Stored Information

Figure 1 illustrates information entities and characteristics of information entities to which the currently described methods and systems are applied. In the following discussion, information entities are referred to as "documents." These may be traditional electronic documents, electronically stored publications, or any of many other types of information entities that include natural-language words and text. For purposes of the present discussion, a document 102 is considered to include a number of terms and short phrases, represented in Figure 1 by small rectangles, such as rectangle 104. A document may be traditional electronic document, an electronically stored publication, or any of many other types of information entities that include natural-language words and text. Each document is associated with a document location 106. In many cases, the document location may be a universal resource locator ("URL") or universal resource identifier ("URI") that allows a web browser or other computer application to access the document through the Internet. Documents may also be associated with, or contain, titles. Each document can be processed to generate a list of terms 108 contained within the document. In the following discussion, the word "term" is used to generally mean a word or short phrase of a natural language, but may also include numbers, acronyms, and other groups of symbols that can be parsed from a document. Based on the list of terms 108 extracted from a document and on other considerations, a set of characteristic terms 110 can be selected to describe the document. In addition, the document can be assigned to a particular category of documents 112, with the assignment of the document to the category associated with a relevance score 114 that, in many implementations, is a real number in the range [0,1].

Figure 2 illustrates document sources. In general, electronic documents and other types of electronically stored information are accessed from particular document sources, which may include online information services, electronic databases and archives, particular web sites, or the Internet in general. In Figure 2, four different document sources 202-205 are illustrated as four large collections of documents. Just as a particular document may be characterized by a list of characteristic terms, document sources may also be associated with a generally much larger number of characteristic source terms 208-211.

Figures 3A-C illustrate an example document search. As shown in Figure 3A, the search begins based on a particular term 302. In this example, all of the documents in document source 202 are searched for the occurrence of the initial search term 302. In most document sources, many different types of indexes, including keyword indexes, are maintained for the documents accessible through the document source in order to facilitate this type of searching. As shown by arrows 304-309 and ellipses 310, a large number of documents within document source 202 may be identified as containing one or more instances of the original search term. As shown in Figure 3B, a more complex search may involve the initial search term 302 as well as a second search term 312, with the search directed by a Boolean combination of the two search terms, such as the Boolean operators "OR" 314, "AND NOT" 315, and "AND" 316. Such multi-term searches may be carried out in a stepwise fashion, with the results produced from an initial single-term search, such as that shown in Figure 3A, supplemented or modified by an additional search based on a second term or a Boolean-operator joined combination of the first term and second term. As shown in Figure 3C, a multi-term search may be extended, using a third search term 320, to include documents of a second document source 203. Again, such multi-term searches may be carried out based on a single Boolean expression that includes all of the search terms or in stepwise fashion, with progressive modification of search results produced through a sequence of searches.

The types of document searches illustrated in Figures 3A-C may produce very large result sets. For even modestly sized document sources, tens of thousands or more documents may be retrieved from a single-term search. When a sequence of terms are employed in a sequence of searches that sequentially supplement and modify an initial search, a researcher may wish to somehow keep track of, or annotate, various relationships among the discovered documents. However, when result sets contain thousands, tens of thousands, or more documents, it is clearly impossible to create and maintain these types of annotations and indications of relationships between individual documents. In fact, it is often essentially impossible for the researcher to even cursorily review information about the documents produced by term searches due to the large volumes of documents in

the results sets. In other words, modern technology provides for efficient and cost-effective storage of enormous numbers of electronic documents, easy access to these documents through the Internet, and powerful tools for searching for particular documents, but has, so far, not provided tools for efficient use and understanding of large result sets obtained by searching document sources for documents.

The current document discloses methods and systems for searching for documents that provide a graphical user interface ("GUI") that allows a user to easily visualize search results, even search results containing thousands of documents, and provides tools for sequential searching, searching multiple document sources, annotating and storing search-based investigations and search results, and a variety of additional capabilities. Although searches and search results can be visualized in a variety of different ways, using the currently described methods and systems, a large number of the tools and GUI features are based on a three-dimensional, graph-like rendering of the search process and search results.

Figures 4A-F and 5 illustrate a graph-like rendering of search results provided by the currently described methods and systems. As shown in Figure 4A, a document search and search results obtained by the document search are visualized in three dimensions. In many cases, a Cartesian coordinate system is used, with a horizontal *y* axis 402, a vertical *x* axis 404, and a *z* axis 406. The *x* and *y* axes lie in the plane of an electronic display device and the positive *z* axis projects outward, towards a viewer, orthogonal to the plane of the electronic display device. As shown in Figure 4B, a document search may begin with a single search term. In one implementation, the initial search term 408 is represented by a spherical node and positioned coincident with the origin of the three-dimensional coordinate system. When the search is carried out with respect to documents accessible through a particular document source, a number of documents related to the search term is returned as a result set. As shown in Figure 4C, these returned documents are represented by document nodes 410-413. In Figure 4C, only four documents are shown in the initial result set although, in many cases, the initial result set may contain up to some maximum number of documents, with the maximum number a user-defined or default setting. Documents returned from a term search may be only

those documents that contain more than a threshold number of instances of the search term. This threshold may also be a user-defined or default setting. The mechanics of the searching process and the various settings and parameters that define which documents, and the number of documents, returned by a particular term-based search are generally orthogonal to the currently described methods and systems for visualizing the search process and result sets returned by searched. In other words, the currently described methods and systems are related to display of the search process and search results, rather than implementation of the search process and specific types of search results.

As shown in Figure 4D, each of the documents in the original result set 410-413 may be analyzed to produce a set of characteristic terms for the document. This analysis may involve considering the frequency of occurrences of terms within the document, the characteristic terms for the document source, and other such considerations, and may be controlled by various parameters and settings with respect to various thresholds and maximum numbers, just as for the term search. The characteristic terms for each document are then represented by additional term nodes linked to the respective document. For example, in Figure 4D, the characteristic terms determined for document 410 represented by term nodes 414-416 connected to document 410 by links 418-420. Document 410 is connected to the original search term node 408 by link 422. As shown in Figure 4E, new document searches can be spawned from each of the characteristic terms identified for each of the initial documents. For example, searches are spawned from term nodes 414-416, and these new searches produce additional document nodes 424-426, 428-430, and 432-434, respectively. Similar expansion of the other term nodes has been carried out to produce the graph shown in Figure 4E. As before, the new documents produced by the new searches, such as documents 424-426 produced by a new search based on term node 414, are linked to the term node by links 436-438. The second-level or second-order searches spawned from the term nodes associated with the document nodes produced by the first term search may be based on any of various different Boolean connectors. For example, the search may be carried out for documents containing the initial search term and the characteristic term, for

documents containing the initial search term but not containing the characteristic term, or for various other Boolean operators. The type of search used in the expansion may also be defined in user settings and, as discussed later, may be directly specified by a user for particular expansions. As can be appreciated from the graph shown in Figure 4E, a convention is used in these illustrations in which term nodes are shaded and document nodes are unshaded. Figure 4F illustrates the graph-like rendering of a multi-term search following addition of term nodes to the second level of documents. The number of initial document-node and term-node expansions carried out by the currently described methods and systems for an initial specified search term can be controlled by user-defined settings and parameters.

Figure 5 shows an alternative illustration of the graph-like rendering of the search process and search results illustrated in Figures 4A-F. The graph-like rendering can be viewed as concentric layers. The innermost sphere 502 represents the initial search term. A next layer 504 contains the initial documents found by searching a document source using the initial search term. A next layer 506 includes the characteristic terms generated for each of the initially found documents. A next layer 508 includes a next set of documents found by searches that consider both the initial term 502 and characteristic terms 506 produced from the initially found documents. Each layer is an expansion of the next innermost layer, with document and term expansions alternating.

Figures 6A-G illustrate details about the three-dimensional display of the graph-like representations of searches and search results as well as basic manipulations of the graph-like renderings available to users through the GUI provided by the currently described methods and systems. Figure 6A shows a very small graph-like rendering 600 consisting of three term nodes 602-604 and four document nodes 606-609. Certain implementations of the GUI may provide realistic three-dimensional renderings of the graph. For example, node 603, logically closest to the user as a result of having the greatest z coordinate, is displayed as a sphere of greater radius then the spheres used for display of nodes 602 and 606-609. Node 604, furthest away from the user as a result of having the largest negative z coordinate, would be displayed as a sphere with a smaller radius than the spheres used to display

nodes 602 and 606-609. However, not only the radii of the sphere vary with respect to position along the *z* axis, but the appearance of the nodes may also vary. In certain implementations, nodes are increasingly blurred with increasing negative *z* coordinates and finally not displayed, when the magnitude of the negative *z* coordinates of the nodes exceed a threshold negative *z*-coordinate magnitude. In this fashion, a great deal of information that would otherwise clutter the rendering of the graph is not displayed to the user.

Figure 6B further illustrates the display technique described above with reference to Figure 6A. In Figure 6B, abstract planes orthogonal to the *z* axis, such as abstract planes 610-612, are spaced long the *z* axis at various *z* coordinates. The volumes of Cartesian space between the planes represent regions in which the nodes and links of the graph have constant radii, widths, and degrees of blurring and/or fading. The sphere radii and the degrees of blurring and/or fading may be different for each abstract-plane-bounded region. In the current case, nodes with *z* coordinates greater than the *z* coordinate of abstract plane 610 have large radii, nodes with *z* coordinates less than the *z* coordinate of abstract plane 610 and greater than the *z* coordinate of abstract plane 612 have medium-sized radii, and nodes with *z* coordinates less than the *z* coordinate of abstract plane 612 have small radii. In this example, nodes with *z* coordinates less than the *z* coordinate of plane 612 are faded to invisibility while other nodes are displayed without blurring or fading. However, any number of abstract planes may be defined and the radii of the spherical renderings of the nodes and the degree of fading and/or blurring may differ for each sub-volume bounded by two abstract planes so that nodes appear to be progressively smaller with increasing logical distance from the user and are displayed with progressively increasing blurring and/or fading within increasing distance from the user. Generally, past some threshold negative *z* coordinate, all nodes with greater than the threshold negative *z* coordinate are not displayed.

The GUI of the currently described methods and systems allows the user, using various types of user input, including mouse clicks, touch-screen inputs, or touch-pad inputs, to manipulate the three-dimensionally rendered graph in three dimensions. Figures 6C-F illustrate a subset of the various types of manipulations

that can be carried out by a user through the GUI. As shown in Figure 6C, the user may translate the graph, in any direction, with respect to the original position of the graph in the display screen. In Figure 6C, dashed arrows, such as dashed arrow 620, represent a logical translation of a node, in this case in the positive *y* direction. Similarly, as shown in Figures 6D-E, using the same illustration conventions used in Figure 6C, the three-dimensional rendering of the graph can be arbitrarily rotated in any direction in Cartesian three-dimensional space. As shown in Figure 6F, the three-dimensional rendering of the graph may be scaled differently in order to increase or decrease the apparent size of the nodes and bonds and the spacing between them. Additionally, as shown in Figure 6G, a three-dimensional representation of a search process and search results may be converted to a two-dimensional representation. In Figure 6G, the small example graph shown in Figure 6A is re-displayed as a two-dimensional graph. Note that the conversion of a three-dimensional graph to a two-dimensional graph may not necessarily be a projection of the three-dimensional graph onto an arbitrary logical plane, but may instead involve significant change in illustration conventions and relative positions of nodes and links. However, the connectivity of nodes to other nodes remains unchanged.

Figures 7A-N illustrate other types of operations and facilities made available to a user through the GUI provided by the currently disclosed methods and systems. Figure 7A shows a small portion of a graph-like rendering of a search process and search results. Note that each node is labeled with text. Term nodes, such as term node 702, are labeled with a term or phrase that they represent. Document nodes, such as document node 704, are labeled with some initial portion of the document's title or name. A user may control whether or not labels are displayed for graph nodes. As shown in Figure 7B, a user may position a cursor 706 over a particular node and enter user input to bring up a display 708 for the node. The display may show the full title for a document node 710, the relevance score for the document 712 with respect to a category to which the node is assigned, and various icons 714-718 which provide an interface to various operations and facilities that can be applied to the node. As shown in Figure 7C, one such icon-represented facility,

when invoked by user input 720, results in display of the actual document 722 in a scrollable window.

As shown in Figures 7D-E, a user may also reposition nodes in sub-trees with respect to remaining portions of a graph. In Figure 7D, the user positions a cursor 724 over node 726 and inputs user input to the GUI that results in node 726 and the sub-tree rooted at node 726 being moved upward, shown in Figure 7E. As shown in Figures 7F-I, a user may decide to group together a number of nodes, such as nodes 730-733. In order to do, as shown in Figure 7G, the user invokes a lasso tool to create an abstract volume, shown in Figure 7G in dashed lines 734, that includes the nodes the user wishes to group together. The dimensions and volume of this lassoed volume 734 may be changed by the user through user input. The position of the volume can also be moved by user input. An input panel 736 may be displayed to allow the user to input a name 738 for the group as well as to invoke various other group-related facilities and operations, including choosing a display color for the nodes of the group. As shown in Figure 7H, as a result of the grouping operation, the nodes within the group may be displayed with a different color and are logically associated together by the system in an internal data structure. As shown in Figure 7I, a user can choose to collapse the nodes of the group into a collapsed group rendering 740. As shown in Figures 7J-K, a user may select a group for copying into a workspace, or tray 744. Alternatively, as shown in Figure 7L, the selected group may be removed from the graph and placed into the tray. The tray represents an internal data structure in which user-identified nodes and collections of nodes can be saved for subsequent analysis and manipulation. As shown in Figures 7M-N, a user may also sequentially select a sequence of nodes, indicated in Figure 7M by cursors 750-753, and group the nodes together as a path. As shown in Figure 7N, the user may select, using an input panel, a display color for the nodes of the path or, alternatively, for the links that link the nodes of the path, and may copy a path to the tray 756. Other tools provided by the GUI allow users to alter the colors and sizes of nodes and links and to add various types of user-defined annotations and elements to a graph.

Figures 8A-I illustrate additional operations available to a user of the currently described method and system for manipulation of graph-like representations of search processes and search results. As shown in Figure 8A, a user can position a cursor 802 onto a term node 804 and input user input to the GUI in order for the GUI to display an input panel 806 to allow the user to expand the node, adding additional search results to the search results currently represented by the graph. The display panel 806 may include a selection of a Boolean operation from among multiple Boolean operations 808 and may allow a user to select particular values of various types of attributes 810 in order to direct the search used to expand the node. The display panel 806 may also allow a user to input the name of a different document source for the expansion search 812. Similarly, as shown in Figure 8B, the user can position a cursor 814 to select a particular document node 816 in order for the GUI to display a display panel 818 to allow the user to select any of various parameters 820 for expansion of the document node. Document-node expansion involves analysis of the document with respect to characteristic terms for the document source and/or other information to select new characteristic terms for the document. The display panel 818 may also allow the user to expand the document node with respect to a different document source, the name of which may be entered in a text-input window 822. As shown in Figure 8C, when the user elects not to expand the document node with respect to a different document source, new characteristic terms 824-828 identified for the document are represented by new term nodes 824-828 linked to the document node 816 that is expanded. As shown in Figure 8D, when the user specifies a new document source for a document-node expansion 830, the document node is bifurcated, as shown in Figure 8E, to produce a copy of the document node 832 linked to the original document node 816 by a source-change link 834 and the copy document node 832 is expanded with new term nodes 836-839 selected with respect to the new document source. Similarly, as shown in Figure 8F, expansion of a term node 825 without specifying a new document source results in new document nodes 840-843 selected from the same document source with respect to which the term node was generated, as shown in Figure 8G. However, as shown in Figure 8H, when the user specifies a new document source 850 in a term-node expansion, a copy

of the term node is generated 852, shown in Figure 8I, and linked to the original term node 825 via a source-change link 854 and the copy term node 852 is expanded to generate new document nodes 856-858 corresponding to documents selected from the new document source. These are but a few of the different types of graph-expansion operations available to a user through the GUI provided by the currently described methods and systems. Of course, any particular node expansion may generate 0, 1, or multiple expansion nodes. The illustrations and examples used in this discussion show only small numbers of expansion nodes, for sake of clarity.

Figures 9A-B illustrate a search-in-graph operation provided by the currently described methods and systems. In Figure 9A, a small three-dimensional graph-like rendering of a search process and search results 902 is currently displayed by the GUI provided by the currently discussed methods and systems. The user has provided to the GUI user input which requests the search-in-graph operation. In response, the GUI displays an input panel 904 which allows the user to input a search expression, such as a term or multiple terms connected by Boolean operators. The search-in-graph functionality results in carrying out a next search only on those documents represented by document nodes in the current graph. As shown in Figure 9B, the results of the search-in-graph search, specified in Figure 9A, are stored in an internal list data structure 906. In Figure 9B, those nodes that form the result set for the search-in-graph search are shown associated with asterisks, such as asterisk 908 associated with document node 910. There are a variety of ways that these types of internally stored lists can be used.

Figures 10A-K illustrate additional features of the GUI provided by the currently discussed methods and systems. As shown in Figure 10A, the GUI generally displays a representation of a search process and search results, such as a graph-like representation 1002 and, in addition, may display various selection devices 1004, 1006, and 1008 that allow a user to find and select various operations, utilities, and features. A main-menu selection wheel 1004 may be displayed at the bottom of the display. The upper portion of the main-menu selection wheel is displayed to the user, while the lower portion 1010, illustrated using dashed lines, is not displayed, but is assumed to be logically present. The locator-menu display wheel 1006 is similarly

displayed, with a right-hand portion displayed to the user and a left-hand portion not displayed, but assumed to be logically present. A column of icons 1008 is additionally provided to allow the user to select various types of features, functionalities, and utilities. As shown in Figure 10B, user input to a special sector 1012 of the main-menu selection wheel causes, as shown in Figure 10C, the selection wheel to rotate logically, resulting in display of a different portion of the main-menu selection wheel to the user. The locator-menu selection wheel 1006 operates similarly. The locator-menu selection wheel provides numerous different functions to facilitate location of particular nodes and groups of nodes within a graph-like representation of a search process and search results. For example, as shown in Figure 10D, input to one of the inner-wheel icons, such as inner-wheel icon 1014, may result in the display of various groups of nodes in the outer wheel 1016. User selection of one of these groups, as represented by cursor 1018, results, as shown in Figure 10E, in the system rotating, scaling, and/or translating the graph-like representation in order to position the selected group of nodes in a logical *z* position close to the user and in a way that the group is prominently displayed to the user. In certain implementations, this may involve changing the color, size, or other display properties of the selected node group. In Figure 10E, the selected group of nodes 1020-1023 are shaded. When a user selects multiple groups for location and display, the GUI may rotate, scale, and/or translate the graph to display a first selected group, as shown in Figure 10E, and then after a short period of time, may again rotate, scale, and/or translate the graph to display a second selected group, as shown in Figure 10F. An arbitrary number of selected groups can be sequentially displayed in this fashion. Selection of another of the inner-wheel icons, shown in Figure 10G, may invoke a historical replay function that replays the search process which generated a particular graph-like representation. As shown in Figures 10H-J, this function first displays the original search term 1030, shown in Figure 10H, then carries out an initial expansion of the initial search term 1032, as shown in Figure 10I, and then carries out a next expansion 1034, as shown in Figure 10J. The replay function may, in addition to replaying a sequence of expansions, replay a variety of other operations carried out by a user, including local expansions, alternation of the positions and relative

arrangements of nodes, insertion of user-defined links between nodes, rotations, scaling, and translations, group and path selections, and many other such operations.

Figure 10K illustrates a user-selection-icon layout used in one implementation of the currently described methods and systems. The column of icons 1008 includes: (1) an icon 1050 that, when selected by a user, results in display of a variety of different navigation tools that a user can use to manually navigate within a graph; (2) an icon that, when selected by a user, undoes the last completed operation 1052; (3) an icon 1054 that, when selected by a user, provides an eraser tool to allow for nodes to be deleted from a graph; (4) an icon that, when selected by a user, invokes various different types of expansion functionalities 1056, described above; (5) an icon 1058 that, when selected by a user, invokes the lasso operation for defining groups, described above; (6) an icon 1060 that, when selected by a user, invokes a path-definition utility for defining paths through the graph, as described above; and (7) an icon 1062 that, when selected by a user, invokes the search-in-graph functionality, described above. The main-menu selection wheel 1010 includes the following icons: (1) an icon 1064 that, when selected by a user, invokes the locator-menu selection wheel; (2) an icon 1066 that, when selected by a user, invokes a new search and display of a new graph-like representation of the search process and search results; (3) an icon 1068 that, when selected by a user, toggles between three-dimensional and two-dimensional representations of the search process and search results; (4) an icon 1070 that, when selected by a user, invokes a heat-map representation of a search results, described below; (5) an icon 1072 that, when selected by a user, toggles between displaying and not displaying titles for nodes, as discussed above; (6) an icon 1074 that, when selected by a user, invokes a help utility; and (7) an icon 1076 that, when selected by a user, invokes an aggregation utility that can aggregate the outer level of leaf nodes in order to facilitate viewing lower-level layers of nodes in a displayed graph. The inner wheel of the locator-menu selection wheel includes the following icons: (1) an icon that, when selected by a user, invokes selection of groups and/or paths for automated navigation 1080; (2) an icon 1082 that, when selected by a user, invokes a utility for selection of document categories for automatic navigation; (3) an icon 1084 that, when selected

by a user, selects attribute-value-based node location; and (4) an icon 1086 that, when selected by a user, provides for automatic navigation to nodes representing added terms in sequential searching. Various implementations include different mappings of functionality invocation to icons and additional displayed selection devices for selecting operations, utilities, and functions.

Figures 11A-E illustrate one navigation tool provided by the GUI that is, in turn, provided by the currently discussed methods and systems. In Figure 11A, a small graph-like representation of a search process search results is display on an electronic display device through the GUI. When, as shown in Figure 11B, the user selects a particular node from which to begin navigation, as represented by cursor 1102, the navigation tool illuminates, or changes the color of, nodes connected to the selected node, as shown in Figure 11C. In Figure 11C, nodes 1104-1106 are displayed in a different color to indicate that they are possible continuations of paths that can be constructed from a current node according to user-specified criteria. When, as shown in Figure 11D, the user selects one of these highlighted nodes 1105, the navigation tool highlights a next set of nodes that represent a next level of continuation along paths corresponding to the user-provided path criteria. In certain implementations, two or more subsequent node levels may be illuminated by the navigation tool. In essence, this navigation tool acts like a flashlight to assist a user in traversing potential paths through the graph.

In addition to the graph-like representation of search processes and search results, the currently described methods and systems provide for alternative display modes. The first alternative method is referred to as the "stack method." Figures 12A-F illustrate certain elements of a stack-mode display. The first element, shown in Figure 12A, is a node 1202 which may represent a single document node or a group of document nodes 1204. The next element, shown in Figure 12B, is a dimension plane 1206 which includes an orbit 1208 along which nodes are positioned, such as node 1210. A dimension plane gathers nodes representing documents that have been assigned to a particular category. The position of the nodes along the orbit reflects the computed relevance of the documents corresponding to the nodes with respect to the category. As shown in Figure 12C, a

starting point 1212 of an orbit may correspond to a relevance of 0.0 and a final point along the orbit 1214 may correspond to a relevance of 1.0. The orbit is displayed as a broken helical segment with a very shallow pitch. There is a gap, shown in Figure 12D, 1216 that separates the two ends of the orbit and the two ends are vertically displaced 1218 from one another. As discussed above, a node may correspond to a group or collection of document nodes. When a stack is displayed, a user may zoom in on a node within an orbit. Past a particular threshold zoom level, a node representing a collection of document nodes may be radially expanded outward, from the collection node, to show the individual nodes within the collection node. In Figure 12E, a collection node 1220 is shown positioned on the orbit 1222 of a dimension plane 1224. When the region outlined with dashed lines 1226 is magnified by a zoom-in operation, as shown in Figure 12F, the collection node 1220 is radially expanded to show the individual document nodes 1230-1236 that together comprise the collection node 1220.

Figures 13A-13H illustrate the correspondence between a graph-like representation of a search process and search results and a stack representation of the same search process and search results. In Figure 13A, a graph-like representation of a search process and search results 1302 shown on the left-hand side of the figure and a nascent stack representation 1304 shown on the right-hand side of the figure. The nascent stack representation includes multiple dimension planes 1306-1311 that each represents a different category to which document nodes can be assigned.

Figure 13B shows numeric categories associated with each document node in the graph 1302, such as the category 2 1312 associated with document node 1314. Each dimension plane is labeled with a numeric representation of a category, such as the category 6 1316 labeling dimension plane 1311. In a first logical step of creating the stack representation, each document node in the graph is copied into a particular orbital position of the dimension plane corresponding to the category to which the document node has been assigned. In Figure 13C, curved arrows, such as curved arrow 1320, show the mapping between category-5 document nodes and stack nodes positioned along the orbit 1322 of the category-5 dimension plane 1310. Again, the positions of the stack nodes reflect the relevance of a document or group

of documents with respect to the category represented by the dimension plane. Next, as shown in Figures 13D-H, paths that interconnect nodes in the graph are copied into the stack representation. As shown in Figure 13D, there is a path, represented by arrows, such as arrow 1324, that interconnects nodes 1326-1328 in the graph. Line segments representing connections between document nodes of this path are then created in the stack representation. For example, line segment 1330 represents the interconnection of nodes 1326 and 1327 in the path and line segment 1332 represents interconnection of nodes 1327 and 1328 in the graph. Figure 13E illustrates copying of a second path from the graph to the stack representation. A second path includes node 1334 rather than node 1328, so that line segment 1336 is added to the stack representation to represent the path segment of the new path between node 1327 and node 1334. Figure 13F shows addition of line segments 1340 and 1342 to the stack representation in order to represent paths that begin at node 1324 and extend to nodes 1344 and 1346. Using the same illustration conventions as used in Figures 13D-F, Figures 13G and 13H illustrate construction of more line segments within the stack representation in order to represent more paths within the graph. Thus, a stack representation represents the same information represented by the graph display, but in a different fashion. The stack representation is more convenient for visualizing very large result sets of up to hundreds of thousands of nodes. The GUI provides a full range of three-dimensional display manipulation operations for stack representations that allow stacks to be scaled, rotated, altered, and annotated in similar fashion to graph representations.

Figure 14 illustrates a third type of representation of search processes and search results referred to as a "heat-map display." In the heat-map display, a surface is fitted over the node of a graph. In Figure 14, a graph representation 1402 is shown on the left-hand side of the figure and the equivalent heat-map display 1404 is shown on the right-hand side of the figure. In the heat-map display, the surface has varying shading and color to illustrate the varying relevance of underlying nodes with respect to a particular category, or other such information. In Figure 14, darkened areas, such as darkened area 1406, represents a high degree of relevance of the underlying nodes. The heat-map display can be manipulated in the same fashion as

graphs and stacks. The heat-map display allows a user to quickly visualize relative relevances or other relative characteristics of large numbers of nodes in order to facilitate identifying documents of particular interest or identifying relationships between documents.

Figure 15 illustrates yet another feature of the GUI provided by the currently discussed methods and systems. The GUI may provide a carousel feature 1502 that shows thumbnail-like descriptions of various different graph-like representations of search results and search processes 1504-1510. This allows a user to concurrently work on multiple different parallel searches. Search results may be stored, by the system, to flat files and recovered from flat files to various types of representations in the GUI. In addition, the previously described tray feature 1504 can be used to accumulate groups, paths, and other objects of interest from multiple concurrent searches. In Figure 15, the tray contains a first group 1506 selected from the graph representing one search 1505 and a second group 1508 selected from the currently displayed graph, also shown as the currently selected thumbnail graph 1507 in the carousel.

One implementation of the currently described methods and systems includes a client-side application, that runs on a personal computer, laptop, notebook, smart phone, or other user device in the context of a web browser. This client-side application communicates, through the web browser and device operating system, with one or more server-side applications. The communications is carried out using the REST protocol over HTTP. The server-side application accesses online information services, Internet-connected information services, and/or databases and caches on behalf of client-side applications in order to carry out searches on behalf of the client-side application and return search results. In certain implementations, the client-side application may also undertake partial rendering of search results into the various different types of visual representations provided to users by the GUI, described above. The client-side application executes the above-described GUI interface and carries out lower-level graphics processing in order to render representations of search processes and search results for display to the user. Different implementations may adjust the boundaries between client-side-application

processing and responsibilities and server-side-application processing and responsibilities. In general, the client-side application assumes greater processing overheads when the user device has greater general-processing and special-purpose-graphics processing capabilities.

Figure 16 provides a general architectural diagram for various types of computers and other processor-controlled devices. The high-level architectural diagram may describe a modern computer system, such as a personal computer or server. The computer system contains one or multiple central processing units ("CPUs") 1602-1605, one or more electronic memories 1608 interconnected with the CPUs by a CPU/memory-subsystem bus 1610 or multiple busses, a first bridge 1612 that interconnects the CPU/memory-subsystem bus 1610 with additional busses 1614 and 1616, or other types of high-speed interconnection media, including multiple, high-speed serial interconnects. These busses or serial interconnections, in turn, connect the CPUs and memory with specialized processors, such as a graphics processor 1618, and with one or more additional bridges 1620, which are interconnected with high-speed serial links or with multiple controllers 1622-1627, such as controller 1627, that provide access to various different types of mass-storage devices 1628, electronic displays, input devices, and other such components, subcomponents, and computational resources.

Figure 17 illustrates generalized hardware and software components of a general-purpose computer system. The computer system 1700 is often considered to include three fundamental layers: (1) a hardware layer or level 1702; (2) an operating-system layer or level 1704; and (3) an application-program layer or level 1706. The hardware layer 1702 includes one or more processors 1708, system memory 1710, various different types of input-output ("I/O") devices 1710 and 1712, and mass-storage devices 1714. Of course, the hardware level also includes many other components, including power supplies, internal communications links and busses, specialized integrated circuits, many different types of processor-controlled or microprocessor-controlled peripheral devices and controllers, and many other components. The operating system 1704 interfaces to the hardware level 1702 through a low-level operating system and hardware interface 1716 generally

comprising a set of non-privileged computer instructions 1718, a set of privileged computer instructions 1720, a set of non-privileged registers and memory addresses 1722, and a set of privileged registers and memory addresses 1724. In general, the operating system exposes non-privileged instructions, non-privileged registers, and non-privileged memory addresses 1726 and a system-call interface 1728 as an operating-system interface 1730 to application programs1732-1736 that execute within an execution environment provided to the application programs by the operating system. The operating system, alone, accesses the privileged instructions, privileged registers, and privileged memory addresses. By reserving access to privileged instructions, privileged registers, and privileged memory addresses, the operating system can ensure that application programs and other higher-level computational entities cannot interfere with one another's execution and cannot change the overall state of the computer system in ways that could deleteriously impact system operation. The operating system includes many internal components and modules, including a scheduler 1742, memory management 1744, a file system 1746, device drivers 1748, and many other components and modules.

Figure 18 illustrates generalized hardware and software components of a general-purpose computer system that includes a virtualization layer. Figure 18 uses the same illustration conventions as used in Figure 17. In particular, the computer system 1800 in Figure 18 includes the same hardware layer 1802 as the hardware layer 402 shown in Figure 17. However, rather than providing an operating system layer directly above the hardware layer, as in Figure 17, the virtualized computing environment illustrated in Figure 18 features a virtualization layer 1804 that interfaces through a virtualization-layer/hardware-layer interface 1806, equivalent to interface 1716 in Figure 17, to the hardware. The virtualization layer provides a hardware-like interface 1808 to a number of virtual machines, such as virtual machine 1810, executing above the virtualization layer in a virtual-machine layer 1812. Each virtual machine includes one or more application programs or other higher-level computational entities packaged together with an operating system, such as application 1814 and operating system 1816 packaged together within virtual machine 1810. Each virtual machine is thus equivalent to the operating-system layer

1704 and application-program layer 1706 in the general-purpose computer system shown in Figure 17. Each operating system within a virtual machine interfaces to the virtualization-layer interface 1808 rather than to the actual hardware interface 1806. The virtualization layer partitions hardware resources into abstract virtual-hardware layers to which each operating system within a virtual machine interfaces. The operating systems within the virtual machines, in general, are unaware of the virtualization layer and operate as if they were directly accessing a true hardware interface. The virtualization layer ensures that each of the virtual machines currently executing within the virtual environment receive a fair allocation of underlying hardware resources and that all virtual machines receive sufficient resources to progress in execution. The virtualization-layer interface 1808 may differ for different operating systems. For example, the virtualization layer is generally able to provide virtual hardware interfaces for a variety of different types of computer hardware. This allows, as one example, a virtual machine that includes an operating system designed for a particular computer architecture to run on hardware of a different architecture. The number of virtual machines need not be equal to the number of physical processors or even a multiple of the number of processors. The virtualization layer includes a virtual-machine-monitor module 1818 that virtualizes physical processors in the hardware layer to create virtual processors on which each of the virtual machines executes. For execution efficiency, the virtualization layer attempts to allow virtual machines to directly execute non-privileged instructions and to directly access non-privileged registers and memory. However, when the operating system within a virtual machine accesses virtual privileged instructions, virtual privileged registers, and virtual privileged memory through the virtualization-layer interface 1808, the accesses may result in execution of virtualization-layer code to simulate or emulate the privileged resources. The virtualization layer additionally includes a kernel module 1820 that manages memory, communications, and data-storage machine resources on behalf of executing virtual machines. The kernel, for example, may maintain shadow page tables on each virtual machine so that hardware-level virtual-memory facilities can be used to process memory accesses. The kernel may additionally include routines that implement virtual communications and data-

storage devices as well as device drivers that directly control the operation of underlying hardware communications and data-storage devices. Similarly, the kernel virtualizes various other types of I/O devices, including keyboards, optical-disk drives, and other such devices. The virtualization layer essentially schedules execution of virtual machines much like an operating system schedules execution of application programs, so that the virtual machines each execute within a complete and fully functional virtual hardware layer.

Figure 19 illustrates an Internet-connected distributed computer system. Figure 19 shows a typical distributed system in which a large number of PCs 1902-1905, a high-end distributed mainframe system 1910 with a large data-storage system 1912, and a large computer center 1914 with large numbers of rack-mounted servers or blade servers all interconnected through various communications and networking systems that together comprise the Internet 1916. Such distributed computing systems provide diverse arrays of functionalities. For example, a PC user sitting in a home office may access hundreds of millions of different web sites provided by hundreds of thousands of different web servers throughout the world and may access high-computational-bandwidth computing services from remote computer facilities for running complex computational tasks.

Figure 20 illustrates cloud computing. In the recently developed cloud-computing paradigm, computing cycles and data-storage facilities are provided to organizations and individuals by cloud-computing providers. In Figure 20, a user using a personal computer 2002 accesses a service provided by an organization that has implemented server-side applications to execute in a public cloud 2004. The organization can configure virtual computer systems and even entire virtual data centers and can launch execution of server-side application programs on the virtual computer systems and virtual data centers in order to carry out any of many different types of computational tasks Cloud-computing facilities are intended to provide computational bandwidth and data-storage services much as utility companies provide electrical power and water to consumers. Cloud computing provides enormous advantages to organizations which do not wish to purchase, manage, and maintain in-house data centers. Such organizations can dynamically add and delete

virtual computer systems from their virtual data centers within public clouds in order to track computational-bandwidth and data-storage needs, rather than purchasing sufficient computer systems within a physical data center to handle peak computational-bandwidth and data-storage demands. Moreover, organizations can completely avoid the overhead of maintaining and managing physical computer systems, including hiring and periodically retraining information-technology specialists and continuously paying for operating-system and database-management-system upgrades. Furthermore, cloud-computing interfaces allow for easy and straightforward configuration of virtual computing facilities, flexibility in the types of applications and operating systems that can be configured, and other functionalities that are useful even for owners and administrators of private cloud-computing facilities used by a single organization.

Figure 21 illustrates electronic communications between a client and server computer. In Figure 21, a client computer 2102 is shown to be interconnected with a server computer 2104 via local communication links 2106 and 2108 and a complex distributed intermediary communications system 2110, such as the Internet. This complex communications system may include a large number of individual computer systems and many types of electronic communications media, including wide-area networks, public switched telephone networks, wireless communications, satellite communications, and many other types of electronics-communications systems and intermediate computer systems, routers, bridges, and other device and system components. Both the server and client computers are shown to include three basic internal layers including an applications layer 2112 in the client computer and a corresponding applications and services layer 2114 in the server computer, an operating-system layer 2116 and 2118, and a hardware layer 2120 and 2122. The server computer 2104 is additionally associated with an internal, peripheral, or remote data-storage subsystem 2124. The hardware layers 2120 and 2122 may include the components discussed above with reference to Figure 1 as well as many additional hardware components and subsystems, such as power supplies, cooling fans, switches, auxiliary processors, and many other mechanical, electrical, electromechanical, and electro-optical-mechanical components. The operating

system 2116 and 2118 represents the general control system of both a client computer 2102 and a server computer 2104. The operating system interfaces to the hardware layer through a set of registers that, under processor control, are used for transferring data, including commands and stored information, between the operating system and various hardware components. The operating system also provides a complex execution environment in which various application programs, including database management systems, web browsers, web services, and other application programs execute. In many cases, modern computer systems employ an additional layer between the operating system and the hardware layer, referred to as a "virtualization layer," that interacts directly with the hardware and provides a virtual-hardware-execution environment for one or more operating systems.

Client systems may include any of many types of processor-controlled devices, including tablet computers, laptop computers, mobile smart phones, and other such processor-controlled devices. These various types of clients may include only a subset of the components included in a desktop personal component as well components not generally included in desktop personal computers.

Electronic communications between computer systems generally comprises packets of information, referred to as datagrams, transferred from client computers to server computers and from server computers to client computers. In many cases, the communications between computer systems is commonly viewed from the relatively high level of an application program which uses an application-layer protocol for information transfer. However, the application-layer protocol is implemented on top of additional layers, including a transport layer, Internet layer, and link layer. These layers are commonly implemented at different levels within computer systems. Each layer is associated with a protocol for data transfer between corresponding layers of computer systems. These layers of protocols are commonly referred to as a "protocol stack." In Figure 21, a representation of a common protocol stack 2130 is shown below the interconnected server and client computers 2104 and 2102. The layers are associated with layer numbers, such as layer number "1" 2132 associated with the application layer 2134. These same layer numbers are used in the depiction of the interconnection of the client computer 2102 with the server computer

2104, such as layer number "1" 2132 associated with a horizontal dashed line 2136 that represents interconnection of the application layer 2112 of the client computer with the applications/services layer 2114 of the server computer through an application-layer protocol. A dashed line 2136 represents interconnection via the application-layer protocol in Figure 21, because this interconnection is logical, rather than physical. Dashed-line 2138 represents the logical interconnection of the operating-system layers of the client and server computers via a transport layer. Dashed line 2140 represents the logical interconnection of the operating systems of the two computer systems via an Internet-layer protocol. Finally, links 2106 and 2108 and cloud 2110 together represent the physical communications media and components that physically transfer data from the client computer to the server computer and from the server computer to the client computer. These physical communications components and media transfer data according to a link-layer protocol. In Figure 21, a second table 2142 aligned with the table 2130 that illustrates the protocol stack includes example protocols that may be used for each of the different protocol layers. The hypertext transfer protocol ("HTTP") may be used as the application-layer protocol 2144, the transmission control protocol ("TCP") 2146 may be used as the transport-layer protocol, the Internet protocol 2148 ("IP") may be used as the Internet-layer protocol, and, in the case of a computer system interconnected through a local Ethernet to the Internet, the Ethernet/IEEE 802.3u protocol 2150 may be used for transmitting and receiving information from the computer system to the complex communications components of the Internet. Within cloud 2110, which represents the Internet, many additional types of protocols may be used for transferring the data between the client computer and server computer.

Consider the sending of a message, via the HTTP protocol, from the client computer to the server computer. An application program generally makes a system call to the operating system and includes, in the system call, an indication of the recipient to whom the data is to be sent as well as a reference to a buffer that contains the data. The data and other information are packaged together into one or more HTTP datagrams, such as datagram 2152. The datagram may generally include a header 2154 as well as the data 2156, encoded as a sequence of bytes within a block

of memory. The header 2154 is generally a record composed of multiple byte-encoded fields. The call by the application program to an application-layer system call is represented in Figure 21 by solid vertical arrow 2158. The operating system employs a transport-layer protocol, such as TCP, to transfer one or more application-layer datagrams that together represent an application-layer message. In general, when the application-layer message exceeds some threshold number of bytes, the message is sent as two or more transport-layer messages. Each of the transport-layer messages 2160 includes a transport-layer-message header 2162 and an application-layer datagram 2152. The transport-layer header includes, among other things, sequence numbers that allow a series of application-layer datagrams to be reassembled into a single application-layer message. The transport-layer protocol is responsible for end-to-end message transfer independent of the underlying network and other communications subsystems, and is additionally concerned with error control, segmentation, as discussed above, flow control, congestion control, application addressing, and other aspects of reliable end-to-end message transfer. The transport-layer datagrams are then forwarded to the Internet layer via system calls within the operating system and are embedded within Internet-layer datagrams 2164, each including an Internet-layer header 2166 and a transport-layer datagram. The Internet layer of the protocol stack is concerned with sending datagrams across the potentially many different communications media and subsystems that together comprise the Internet. This involves routing of messages through the complex communications systems to the intended destination. The Internet layer is concerned with assigning unique addresses, known as "IP addresses," to both the sending computer and the destination computer for a message and routing the message through the Internet to the destination computer. Internet-layer datagrams are finally transferred, by the operating system, to communications hardware, such as a network-interface controller ("NIC") which embeds the Internet-layer datagram 2164 into a link-layer datagram 2170 that includes a link-layer header 2172 and generally includes a number of additional bytes 2174 appended to the end of the Internet-layer datagram. The link-layer header includes collision-control and error-control information as well as local-network addresses. The link-layer packet or datagram

2170 is a sequence of bytes that includes information introduced by each of the layers of the protocol stack as well as the actual data that is transferred from the source computer to the destination computer according to the application-layer protocol.

Next, the RESTful approach to web-service APIs is described, beginning with Figure 22. Figure 22 illustrates the role of resources in RESTful APIs. In Figure 22, and in subsequent figures, a remote client 2202 is shown to be interconnected and communicating with a service provided by one or more service computers 2204 via the HTTP protocol 2206. Many RESTful APIs are based on the HTTP protocol. Thus, the focus is on the application layer in the following discussion. However, as discussed above with reference to Figure 21, the remote client 2202 and service provided by one or more server computers 2204 are, in fact, physical systems with application, operating-system, and hardware layers that are interconnected with various types of communications media and communications subsystems, with the HTTP protocol the highest-level layer in a protocol stack implemented in the application, operating-system, and hardware layers of client computers and server computers. The service may be provided by one or more server computers, as discussed above in a preceding section. As one example, a number of servers may be hierarchically organized as various levels of intermediary servers and end-point servers. However, the entire collection of servers that together provide a service are addressed by a domain name included in a uniform resource identifier ("URI"), as further discussed below. A RESTful API is based on a small set of verbs, or operations, provided by the HTTP protocol and on resources, each uniquely identified by a corresponding URI. Resources are logical entities, information about which is stored on one or more servers that together comprise a domain. URIs are the unique names for resources. A resource about which information is stored on a server that is connected to the Internet has a unique URI that allows that information to be accessed by any client computer also connected to the Internet with proper authorization and privileges. URIs are thus globally unique identifiers, and can be used to specify resources on server computers throughout the world. A resource may be any logical entity, including people, digitally encoded documents, organizations, services, routines, and other such entities that can be described and characterized by

digitally encoded information. A resource is thus a logical entity. Digitally encoded information that describes the resource and that can be accessed by a client computer from a server computer is referred to as a "representation" of the corresponding resource. As one example, when a resource is a web page, the representation of the resource may be a hypertext markup language ("HTML") encoding of the resource. As another example, when the resource is an employee of a company, the representation of the resource may be one or more records, each containing one or more fields, that store information characterizing the employee, such as the employee's name, address, phone number, job title, employment history, and other such information.

In the example shown in Figure 22, the web servers 2204 provides a RESTful API based on the HTTP protocol 2206 and a hierarchically organized set of resources 2208 that allow clients of the service to access information about the customers and orders placed by customers of the Acme Company. This service may be provided by the Acme Company itself or by a third-party information provider. All of the customer and order information is collectively represented by a customer information resource 2210 associated with the URI "http://www.acme.com/customerInfo" 2212. As discussed further, below, this single URI and the HTTP protocol together provide sufficient information for a remote client computer to access any of the particular types of customer and order information stored and distributed by the service 2204. A customer information resource 2210 represents a large number of subordinate resources. These subordinate resources include, for each of the customers of the Acme Company, a customer resource, such as customer resource 2214. All of the customer resources 2214-2218 are collectively named or specified by the single URI "http://www.acme.com/customerInfo/customers" 2220. Individual customer resources, such as customer resource 2214, are associated with customer-identifier numbers and are each separately addressable by customer-resource-specific URIs, such as URI "http://www.acme.com/customerInfo/customers/361" 2222 which includes the customer identifier "361" for the customer represented by customer resource 2214. Each customer may be logically associated with one or more orders.

For example, the customer represented by customer resource 2214 is associated with three different orders 2224-2226, each represented by an order resource. All of the orders are collectively specified or named by a single URI "http://www.acme.com/customerInfo/orders" 2236. All of the orders associated with the customer represented by resource 2214, orders represented by order resources 2224-2226, can be collectively specified by the URI "http://www.acme.com/customerInfo/customers/361/orders" 2238. A particular order, such as the order represented by order resource 2224, may be specified by a unique URI associated with that order, such as URI "http://www.acme.com/customerInfo/customers/361/orders/1" 2240, where the final "1" is an order number that specifies a particular order within the set of orders corresponding to the particular customer identified by the customer identifier "361."

In one sense, the URIs bear similarity to path names to files in file directories provided by computer operating systems. However, it should be appreciated that resources, unlike files, are logical entities rather than physical entities, such as the set of stored bytes that together compose a file within a computer system. When a file is accessed through a path name, a copy of a sequence of bytes that are stored in a memory or mass-storage device as a portion of that file are transferred to an accessing entity. By contrast, when a resource is accessed through a URI, a server computer returns a digitally encoded representation of the resource, rather than a copy of the resource. For example, when the resource is a human being, the service accessed via a URI specifying the human being may return alphanumeric encodings of various characteristics of the human being, a digitally encoded photograph or photographs, and other such information. Unlike the case of a file accessed through a path name, the representation of a resource is not a copy of the resource, but is instead some type of digitally encoded information with respect to the resource.

In the example RESTful API illustrated in Figure 22, a client computer can use the verbs, or operations, of the HTTP protocol and the top-level URI 2212 to navigate the entire hierarchy of resources 2208 in order to obtain information about

particular customers and about the orders that have been placed by particular customers.

Figures 23A-D illustrate four basic verbs, or operations, provided by the HTTP application-layer protocol used in RESTful applications. RESTful applications are client/server protocols in which a client issues an HTTP request message to a service or server and the service or server responds by returning a corresponding HTTP response message. Figures 23A-D use the illustration conventions discussed above with reference to Figure 22 with regard to the client, service, and HTTP protocol. For simplicity and clarity of illustration, in each of these figures, a top portion illustrates the request and a lower portion illustrates the response. The remote client 2302 and service 2304 are shown as labeled rectangles, as in Figure 22. A right-pointing solid arrow 2306 represents sending of an HTTP request message from a remote client to the service and a left-pointing solid arrow 2308 represents sending of a response message corresponding to the request message by the service to the remote client. For clarity and simplicity of illustration, the service 2304 is shown associated with a few resources 2310-2312.

Figure 23A illustrates the GET request and a typical response. The GET request requests the representation of a resource identified by a URI from a service. In the example shown in Figure 23A, the resource 2310 is uniquely identified by the URI "http://www.acme.com/item1" 2316. The initial substring "http://www.acme.com" is a domain name that identifies the service. Thus, URI 2316 can be thought of as specifying the resource "item1" that is located within and managed by the domain "www.acme.com." The GET request 2320 includes the command "GET" 2322, a relative resource identifier 2324 that, when appended to the domain name, generates the URI that uniquely identifies the resource, and in an indication of the particular underlying application-layer protocol 2326. A request message may include one or more headers, or key/value pairs, such as the host header 2328 "Host:www.acme.com" that indicates the domain to which the request is directed. There are many different headers that may be included. In addition, a request message may also include a request-message body. The body may be encoded in any of various different self-describing encoding languages, often JSON,

XML, or HTML. In the current example, there is no request-message body. The service receives the request message containing the GET command, processes the message, and returns a corresponding response message 2330. The response message includes an indication of the application-layer protocol 2332, a numeric status 2334, a textural status 2336, various headers 2338 and 2340, and, in the current example, a body 2342 that includes the HTML encoding of a web page. Again, however, the body may contain any of many different types of information, such as a JSON object that encodes a personnel file, customer description, or order description. GET is the most fundamental and generally most often used verb, or function, of the HTTP protocol.

Figure 23B illustrates the POST HTTP verb. In Figure 23B, the client sends a POST request 2346 to the service that is associated with the URI "http://www.acme.com/item1." In many RESTful APIs, a POST request message requests that the service create a new resource subordinate to the URI associated with the POST request and provide a name and corresponding URI for the newly created resource. Thus, as shown in Figure 23B, the service creates a new resource 2348 subordinate to resource 2310 specified by URI "http://www.acme.com/item1," and assigns an identifier "36" to this new resource, creating for the new resource the unique URI "http://www.acme.com/item1/36" 2350. The service then transmits a response message 2352 corresponding to the POST request back to the remote client. In addition to the application-layer protocol, status, and headers 2354, the response message includes a location header 2356 with the URI of the newly created resource. According to the HTTP protocol, the POST verb may also be used to update existing resources by including a body with update information. However, RESTful APIs generally use POST for creation of new resources when the names for the new resources are determined by the service. The POST request 2346 may include a body containing a representation or partial representation of the resource that may be incorporated into stored information for the resource by the service.

Figure 23C illustrates the PUT HTTP verb. In RESTful APIs, the PUT HTTP verb is generally used for updating existing resources or for creating new resources when the name for the new resources is determined by the client, rather

than the service. In the example shown in Figure 23C, the remote client issues a PUT HTTP request 2360 with respect to the URI "http://www.acme.com/item1/36" that names the newly created resource 2348. The PUT request message includes a body with a JSON encoding of a representation or partial representation of the resource 2362. In response to receiving this request, the service updates resource 2348 to include the information 2362 transmitted in the PUT request and then returns a response corresponding to the PUT request 2364 to the remote client.

Figure 23D illustrates the DELETE HTTP verb. In the example shown in Figure 23D, the remote client transmits a DELETE HTTP request 2370 with respect to URI "http://www.acme.com/item1/36" that uniquely specifies newly created resource 2348 to the service. In response, the service deletes the resource associated with the URL and returns a response message 2372.

As further discussed below, and as mentioned above, a service may return, in response messages, various different links, or URIs, in addition to a resource representation. These links may indicate, to the client, additional resources related in various different ways to the resource specified by the URI associated with the corresponding request message. As one example, when the information returned to a client in response to a request is too large for a single HTTP response message, it may be divided into pages, with the first page returned along with additional links, or URIs, that allow the client to retrieve the remaining pages using additional GET requests. As another example, in response to an initial GET request for the customer info resource (2210 in Figure 22), the service may provide URIs 2220 and 2236 in addition to a requested representation to the client, using which the client may begin to traverse the hierarchical resource organization in subsequent GET requests.

Data for the graph-like representation of search processes and search results is stored in numerous data structures. Figure 24 provides one example of the data stored within a node object to describe a document node. Figures 25A-H show data contained in a data script structure to describe an entire graph, in one implementation of the currently described methods and systems. Of course, there are many different possible ways of organizing the data stored to represent a search process and search results.

Figures 26-27 show control-flow diagrams that describe overall operation of one implementation of the currently described methods and systems. Figure 26 illustrates client-side-application operation. In step 2602, the client-side application is launched and the client-side application initializes run-time data structures and establishes connections to local-device services and functionalities as well as to one or more server-side applications. In step 2604, the client-side application renders and displays an initial screen of the GUI. Then, in step 2606, the client-side application waits for the occurrence of a next event. There are many different types of events to which the client-side application responds, including web-browser-generated events, including user-input events and events associated with communication between the web browser and server-side application. When a next event occurs, the client-side application determines, in step 2608, whether server-side handling of the event is needed. When server-side handling is needed, the client-side application generates and transmits an HTTP request to the server, via the web browser and operating system, in step 2610 and receives a response to the request in step 2612. In step 2614, the client-side application determines whether the event needs local handling, such as execution of client-side routines for rendering information for display and, when so, calls appropriate local event handler for the event in step 2616. Local handling may generate additional events for handling by the client-side application. When there are more pending events to handle, as determined in step 2618, a next event is dequeued from an event queue, in step 2620, and control flows back to step 2608. Otherwise, control flows back to step 2606, or the client-side application waits for a next event.

Figure 27 shows a similar control-flow diagram for server-side-application operation. When initially launched, the server-side application initializes run-time data structure, establishes connections with other servers, databases, and data sources, and prepares for responding to requests in step 2702. Then, the server-side application enters an event loop of steps 2704-2708 in which HTTP requests from client-side applications are handled.

Graph Databases

Graph databases were initially developed in the 1980s as an alternative data model for database management systems. Many other types of data models had been developed prior to that time and were in common usage, including hierarchical, network, and relational data models. In general, a particular data model may be most efficient for certain types of data that is to be stored and accessed through a database management system. Relational databases, as one example, are particularly effective for storing the types of data amenable to representation as tables of columns and rows, including the often-used parts and suppliers tables that represent parts of various products and the suppliers of the products and parts, respectively. Relational-database query languages, such as the structured query language ("SQL"), are based on the relational algebra, which provides a theoretical foundation for formulating and executing *ad hoc* queries and for optimizing query execution. Relational databases employ a data schema, consisting of one or more table declarations that specify the structure of the various relational tables stored within the database, that is generally developed prior to receiving and storing data and responding to queries.

In many modern computing environments, including the so-called "big data" computing environments, it may be difficult or impossible to develop a data schema for data storage prior to receiving and processing the data that is to be stored. Furthermore, many types of data that need to be stored in big-data applications, including representations of social networks, are naturally represented as graphs. This type of data consists of entities are connected together in various ways. The connection information is often as important, or more important than, the stored data representing the entities. Many of the types of queries posed to database management systems that store graphs involve traversing graphs or subgraphs according to specified traversal rules in order to find sets of nodes that are connected in particular ways. These types of operations are efficiently carried out by graph database management systems but, in general, would involve many expensive join operations and/or a proliferation of tables in a relational database management system. Graph databases naturally represent, and provide efficient query-based searching of, social-

network information, information about distributed computing configurations, information related to various types of communications and communication networks, information traffic analysis, and distribution systems.

Figure 28 illustrates a graph data object stored by graph databases. In Figure 28, a graph G is specified using several different notational conventions 2802 and is also pictorially represented 2804. A graph is a set of nodes, or vertices, and a set of edges. In the pictorial representation of the graph, the nodes are represented as disks, such as disk 2806 representing a node designated as node V_3, and the edges are represented by curved arrows, such as curved arrow 2808 that connects node V_3 2806 to node V_2 2810. In the graph illustrated in Figure 28, the edges have directions, indicated by the arrow representations. In various other types of graph data models, edges do not have directions, are generally resented as line segments or curves. The notation 2802 used to describe the graph G in Figure 28 specifies the nodes, or vertices, as V_x, where x is a unique integer identifier of a particular node. The edges are specified by a three-subscript notation $E_{x,y,n}$ where x is the identifier of the node from which the edge emerges, y is the identifier for the node that the edge is directed to, and n is a unique integer identifier for a particular edge connecting nodes x and y. For example, in the pictorial representation 2804 of graph G, there are two edges 2810 and 2812 that emanate from node 2814 and connect node 2814 to node 2816. The edge notations $E_{1,5,1}$ and $E_{1,5,2}$ differ in the final subscript n.

In addition to the identifiers of nodes and edges, a graph database may also store properties associated with nodes and edges. In the example shown in Figure 28, each node and each edge are associated with key/value pairs, such as the key/value pairs in table 2820 associated with node V_1 2822. In general, the keys may be specified by character strings and the values may be specified either by character strings or by another fundamental data type, such as an integer, real, character string, or single character. It is possible for values to specified by user-defined data types, in certain graph database management systems. For illustration purposes, capital letters are used in Figure 28 for key names and lower-case, subscripted letters are used for key values. The graph data model illustrated in Figure 28 also adopts the convention that each node and edge has a key A with a corresponding name value. The value

associated with key A is a name for a node or edge that can be used to access the node or edge or to specify the node or edge in a query For example, node V_1 2822 has a name a_1 which is the value of the key/value pair with key A. In the data model illustrated in Figure 28, nodes and edges have unique identifiers symbolically represented by the V_x and $E_{x,y,n}$ notation as well as names. In many cases, the identifier for a node or edge is an integer that uniquely identifies the node or edge within a graph. A further notational convention can be used to represent a particular value or property for a particular node or edge. The name for node V_1 is represented as $V_1 \cdot A$, which stands for the value associated with key A associated with node V_1.

There are many different possible ways for logically and physically storing graph data within a graph database management system. Figure 29 illustrates one logical data-storage model for a graph database that stores graphs, such as the graph illustrated in Figure 28. In the implementation shown in Figure 29, the graph database management system maintains two large hash tables 2902 and 2904. These two hash tables are essentially integer arrays. The indexes of the entries in the hash tables can be used as the unique numerical identifiers for nodes, in the case of hash table 2902 and edges, in the case of hash table 2904. Thus, the hash table entry for a node with numerical identifier x can be found in the node hash table V at array cell V[x]. In order to find a node by name, a hash function $f(\ )$ can be applied to the name of the node, $f(V_x \cdot \mathrm{A})$, to generate the index of the cell of the array that is the entry for the node with name a_1. Each entry in the node hash table 2902, such as entry 2906, contains a reference, or pointer, to a table 2908 that contains the key/value property pairs for the node represented by the hash-table entry as well as an additional pointer 2910 to a list of associations that include indications of those nodes to which the node represented by the hash-table entry is connected by direct edges, such as the two-entry list of nodes that contain the association data structures 2912 and 2914. The association data structures contain references to one or more edges by which the node represented by the hash-table entry is connected to the nodes represented by association data structures.

Each entry in the edge hash table, such as entry 2916, contains a reference to a table-like data structure 2918 that stores the key/value property pairs

for the edge. In addition, the table contains two additional references 2920 and 2922 that point to the entries in the node hash table corresponding to the nodes connected by the edge. Thus, in the implementation shown in Figure 29, nodes and edges identified by unique identifiers or by names can be easily accessed. Graph traversal from a starting node involves following references from the starting node to connected nodes via the edges that connect the connected notes. In addition, or as an alternative, a two-dimensional binary array can be used to indicate all pairs of nodes that are directly connected by edges. There are many different possible ways for storing data at logical and physical levels corresponding to graphs stored and managed by graph database management systems.

Figure 30 shows a simple graph *G* stored in a graph database. The graph *G* includes nodes that each are associated with a unique numeric identifier as well as a name and edges that each are associated with a unique numeric identifier as well as a type. For example, node 3002 has the unique identifier a_1 3004 and the name "bob" 3006. Node a_1 3002 is connected to node a_2 3008 via edge e_1 3010 with type "m" and is connected to node a_4 3012 via edge e_2 3014 with type "f." The graph *G*, shown in Figure 30, is used to illustrate various types of queries in Figures 31 - 34, which are discussed below.

Figure 31 illustrates several path queries. A hypothetical graph query language is used for the path queries in the example of Figures 30 - 34. Many different graph query languages with different syntaxes have been developed but, in the current document, a simple hypothetical graph query language is used for simplicity of illustration. A first query 3102 seeks all pairs of nodes x_1, x_2, that are connected by a single edge of type *m*. The notation "*e*(1)" refers to the type of an edge and the notation "*e*(1) = *m*" specifies a single edge of type *m*. The SELECT statement indicates that an answer containing pairs of nodes is desired and the WHERE clause specifies that each of the desired pair of nodes is connected by a single edge of type m. As can be seen in the representation of graph *G* in Figure 31, node a_1 3104 is connected to node a_2 3106 by an edge of type *m* 3108 and node a_3 3110 is connected to node a_5 3112 by an edge e_4 3114 of type *m*. Therefore, the answer to the query 3116 includes the two pairs of nodes: (a_1, a_2) and (a_3, a_5). A

second query 3118 seeks triples of nodes where the first two nodes of the triple are connected by an edge of type *f*, the first character in the name of the first node is "z," and the first node is connected to the third node by an edge of type *n*. As can be seen in the graphical representation of graph *G*, node a_8 3120 is connected to node a_9 3122 by an edge e_6 3124 of type *f*, the first character in the name of node a_8 3120 is "z," and node a_8 3120 is connected to node a_4 3126 by an edge e_5 3128 of type *n*. There are no other node triples in the graph that meet the requirements of the WHERE clause of query 3118. Thus, the answer 3130 to the second query 3118 is the triple a_8, a_9, and a_4 corresponding to nodes 3120, 3122, and 3126.

Figure 32 shows two additional path queries. A first query 3202 seeks all pairs of nodes that are connected by three edges. Inspection of the graph *G* shown in Figure 32 reveals that there are five sets of nodes 3204 connected by three edges. For example, node a_1 3206 is connected to node a_9 3208 by the three edges 3210 - 3212. A second query 3214 seeks pairs of nodes connected by three edges and with no duplicate nodes in the path and where at least one of the edges has type *m*. Inspection of graph *G* shown in Figure 32 reveals three pairs of nodes 3216 that are connected by three edges with no duplicate nodes in the path.

Figure 33 shows a number of additional, more complex queries. A first query 3302 seeks those edges that occur in three-edge paths connecting two different nodes, the first of which has the name "bob" or "bill," the sought edges being either the first or second edge in the three-edge path and having type *f*. Inspection of graph *G* included in Figure 33 reveals that edge e_2 3304 occurs in two such three-edge paths, and therefore edge e_2 occurs twice in the answer 3306. A second query 3308 seeks the number of nodes connected to a node with name "xylon" by paths of any length. A third query 3310 looks for the minimum path length connecting two nodes with names "xylon" and "zack," respectively. Thus, as shown in Figure 33, graph database queries may include familiar aggregation operators, such as count( ) and min( ), used in query languages like SQL.

Finally, Figure 34 illustrates insertion and deletion of graph entities. The query 3402 inserts a new node, with name "jerry," 3404 into the graph. A second query 3406 inserts a new edge 3408 connecting newly inserted node 3404 to

node a_2 3410. Finally, query 3412 deletes node a_7 from the graph as well as any edges connecting node a_7 to other nodes. By comparing the representation of graph G in Figure 33 to the representation of graph G in Figure 34, as it appears following execution of the three queries 3402, 3406, and 3412, it can be seen that execution of query 3412 has removed edge 3312 and node 3314 from the graph G shown in Figure 33.

These are but a few examples of the types of queries that can be carried out on graphs stored in graph databases by graph database management systems. These types of queries can be used, for example, to identify paths and nodes connected by paths within the graph, to navigate the graph in a variety of different ways, and to select various subgraphs in the graph having specified properties.

Implementation of a System that Accesses, Searches, Analyzes, and Provides Graphical Visualization of Electronically Stored Information and that Includes a Graph Database to Provide Graph-Query Facilities

Figure 35 illustrates one implementation of the above-described system for searching and analyzing electronic documents that displays graph-like representations of search results. In Figure 35, the implementation is shown at three different levels, including: (1) a conceptual level 3502; (2) an implementation-components level 3504; and (3) a hardware level 3506. These levels are separated by horizontal dashed lines 3508 and 3510. At the hardware level, a user computer 3512 is accessed by a user to search a document database and display graph-like representations of search results, as described above. The user computer is interconnected through the Internet 3514 to one or more cloud-computing facilities 3516 that run various web servers and web services on a multitude of server computers and higher-end computers. At the conceptual level, the user computer provides a search and visualization application 3518, described in detail above. The search and visualization application and support routines are collectively referred to

as the "3D viewer." Data displayed by the 3D viewer is obtained from a remote web server that hosts graph-search 3520 and document-search 3522 server applications and maintains a graph database 3524 that stores one or more graphs corresponding to search results. The web server interacts with a back-end document-store web service 3526 that stores documents obtained from multiple document sources and provides document-access methods through which the web server accesses stored documents on behalf of remote users interacting with 3D viewer applications on remote user computers.

At the implementation-components level 3504, the 3D viewer includes a visualization application 3530 that runs in the context of, or in parallel with, a web-browser application 3532 and that interfaces to a web GL interface 3534 that allows the visualization application 3530 to access graphics-processor functionality within the user computer for efficient display and manipulation of the three-dimensional search-result graphs described above. The web server runs a web-server application 3536 that accesses a graph-processor application 3538, a graph database 3540, and a key/value store 3542. The graph processor generates a search-results graph from data retrieved from the back-end document store 3526 and stores the search-results graph in the graph database 3540. In addition, the graph processor builds a client-side data structure, described above with reference to Figures 24-25H, which the web server transfers to the 3D viewer for rendering and display on the user device. The key/value store 3542 stores the key/value pairs returned by the back-end document store in response to document-related queries. The back-end document-store web service includes a web-service application 3544 that interfaces to a document search engine 3546 and one or more document databases 3548. In certain implementations, the web server and web service may execute within different physical cloud-computing facilities. In other implementations, either or both of the web server and web service may execute on dedicated server computers within a private data center or private cloud-computing facility.

Figures 36A-37B illustrate various implementations of an interface provided by the 3D viewer which incorporates graph-database-query functionalities. As shown in Figure 36A, the 3D viewer may display a graph-like representation of a

document search 3602 as well as provide a graph-query-input window 3604 and a display feature 3606 that, when clicked on by a user, directs the 3D viewer to carry out the graph query entered into the graph-query-input window by interfacing to the remote web server and graph processor. In the example shown in Figure 36A, a user has entered a graph query 3610 into the graph-query-input window 3604 that seeks the nodes connected to a node t_1 by three edges. Figure 36B illustrates a visual response to successful query execution. As shown in Figure 36B, the node t_1 3612 and all of the nodes connected to node t_1 by three edges are shown shaded or displayed in a different color than the remaining nodes, such as shaded node 3614. In addition, a results-display window 3616 displays a list of the node pairs that include node t_1 as the first node, each node pair connected by three edges. Node t_1 is a term node and the nodes connected to node t_1 by three edges are document nodes. Alternatively, as shown in Figure 37A, the 3D viewer may provide a graph-query-construction window 3620 to which a user may issue commands to construct a subgraph 3622 having nodes with certain specified parameters. User input to a query-execution feature 3624 results in the 3D viewer transferring a representation of the subgraph to the remote web server and graph processor which executes a graph query on a search-results graph stored in the graph database. As shown in Figure 37B, the 3D viewer may provide a graph-query-results window 3626 that displays a textural representation of the query results and may also use shading or differential coloring of nodes in the three-dimensional representation of initial search results 3628 to illustrate subgraphs within the search results that match the query subgraph 3622, such as subgraph 3630. Of course, there are many different possible user interfaces that can be developed for allowing users of the 3D viewer to pose graph queries executed by the remote web server and graph processor with respect to a graph stored in the graph database.

The web server receives results from the back-end document store as key/value pairs that the web server stores in the key/value store (3542 in Figure 35). The graph processor then uses the search results stored in the key/value store in order to construct a graph that the graph processor stores in the graph database, using a graph query language and query interface, and additionally uses the search results

stored in the key/value store to construct a client-side data structure that is transferred to the 3D viewer for rendering and display to the user display device. The key/value store, in certain implementations, functions as a buffer in which results sequentially fetched, in groups, from the back-end server are temporarily stored until the results can be processed by the graph processor. The graph database stores a graph constructed from the search results and provides a graph query language and interface, allowing the web server and 3d viewer to provide 3D-viewer users with the ability to execute graph queries on the three-dimensional representation of search results. The graphs stored in the graph database also can be viewed as a type of cache of processed document search results. The 3D viewer may, for example, reissue search requests and commands as users interactively manipulate the three-dimensional representation of the search results. Because the search results are cached in the graph database, they may be obtained directly from the graph database by the web server rather than involving a query/response transaction with the backend web service.

Figures 38A-C illustrate the construction of a three-dimensional representation of a search result furnished to the 3D viewer for rendering and display to a user by the remote web server and web service (3536 and 3544 in Figure 35). Initially, the user enters a term that initiates a document search. The term can be viewed as the initial, central node 3802 within a three-dimensional representation of the search results. A document search request is transferred by the 3D viewer to the remote web server which, in turn, issues a document-search query to the backend document store. The backend document store then returns an initial set of results that includes a set of term/document pairs 3804 which represents the set of documents related to the initial search term. These initial key/value-pair search results are stored in the key/value store 3806 by the web server. The graph processor running within the web server then uses the key/value pairs to create nodes and edges corresponding to the initial return search results 3808 within the graph database 3810. In addition, the graph processor prepares an initial client-side data structure that includes the initial nodes and edges and transmits the initial client-side data structure to the 3D viewer. In Figure 38B, a three-dimensional graph-like representation of the initial

search results 3812 are displayed by the visualization application. In the meantime, the web server has issued additional document-search queries to the backend document database to retrieve a next layer of term nodes related to the initial documents obtained from the search request containing the initial term t_i. These next key/value-pair search results 3814 are stored in the key/value store 3806. The graph processor uses these results to generate additional nodes and edges 3816 that are added to the graph stored in the graph database 3810. In addition, these new nodes and edges are added to the client-side data structure, new entries of which are transmitted to the 3D viewer. In Figure 38C, the next level of term nodes are displayed in the three-dimensional representation of the search results 3812. This process continues in similar fashion until a complete initial search result has been obtained from the backend data store, used to generate the client-side data structure and a graph stored in the graph database, and transferred to the 3D viewer for rendering and display to the user. In certain implementations, the entire client-side data structure may be constructed prior to transfer of data to the 3D viewer. In the above-described implementation, transfer of the client-side data structure occurs incrementally, over time, as search results are obtained from the backend server and processed by the web server. In many implementations, the graph stored in the graph database may contain many more nodes and edges than are entered into the client-side data structure for rendering and display by the 3D viewer. The web server may deliberately request, receive, and store more data than is initially transferred to the remote 3D viewer. Subsequently, when a user chooses to expand a node of the three-dimensional representation of the search results, new nodes and edges corresponding to the expansion may be obtained locally by the web server and the graph database rather than from the backend document store via one or more query/response transactions.

Figure 39 illustrates the implementation of the system for searching and analyzing electronic documents using graph-like representations of search results from a stored-data perspective. The backend web service is viewed, from a stored-data perspective, as a very large document database 3902. The 3D viewer includes a client-side data structure 3904 that contains the information that is needed to render

and display a three-dimensional representation of search results and additionally includes stored queries received from users 3906 and stored query results returned by the web server in response to receiving queries 3908. For each user/3D viewer accessing the document database, the web server instantiates the logical data stores contained within the dashed rectangle 3910 in Figure 39 and referred to as "session data." The session data includes the client-side data structure 3904, a backend query cache 3906, a key/value cache or store 3908, a graph database 3910, a graph-query cache 3912, a graph-query-results cache 3914, and, in many implementations, one or more snapshots of the graph database 3916-3918 that store particular instances of the search results that have been transferred to the 3D viewer. For example, a user may elect to snapshot or store a current three-dimensional representation of the search results in order to be able to return to this particular instance of the three-dimensional representation of the search results after further manipulations and operations on the search results. Similarly, a user may select a subgraph of the three-dimensional representation of the search results and store the subgraph separately for subsequent display and interactive processing. These types of intermediate stored results and snapshots may be supported by archiving snapshots of one or more graphs stored within the graph database. Alternatively, the graph processor may maintain many different graphs concurrently within the graph database to support storing different instances of the search results on behalf of the user. By caching queries and query results, the web server may be able to efficiently re-execute queries on behalf of users and efficiently facilitate potentially circuitous navigation by a user through a large three-dimensional representation of a large search-result set.

Figures 40A-F illustrate, using control-flow diagrams, interaction of the 3D viewer, web server, and backend web service to produce a three-dimensional representation of a document-search result to a user of the client computer system. All of Figures 40A-F use the same illustration conventions, next discussed with reference to Figure 40A. Figure 40A has three columns: (1) a client or user column 4002; (2) a web-server column 4004; and (3) a backend column 4006. Actions shown in each of these three columns are carried out by the 3D viewer, web server, and backend, respectively. In step 4010, a user interacts with a web browser running

on the user's computer to launch the 3D viewer. In various different implementations, the 3D viewer may be a web-browser plug-in, a standalone application launched by the web browser, or may be implemented in other ways. In step 4012, the 3D viewer initializes itself and establishes network connections with the web server. Part of the initialization may involve displaying an initial graphical user interface to the user to allow the user to undertake a document search. In step 4014, the user interacts with the 3D viewer via the graphical user interface to request a new document search. As discussed above, this may involve furnishing a term, phrase, or list of terms and phrases for which related documents are desired. In step 4016, the 3D viewer packages the search request into a communications message and sends the message to the web server requesting the document search. In step 4018, the web server receives the search request and caches the search request in the backend query cache (3906 in Figure 39). In step 4019, the web server establishes a new session, including the session data discussed above with reference to Figure 39, for the client computer and user.

Turning to Figure 40B, in step 4020, the web server generates a new session ID and associates the session ID with the newly created session, the client computer, and the user. The association allows the session to be accessed in order to process subsequently received requests from the user and facilitates transmission of data to the user when data becomes available as a result of graph-processor processing. In step 4021, the web server sends the session ID to the 3D viewer which, in step 4022, receives and stores the session ID. The session ID is included, by the 3D viewer, in subsequent messages sent to the web server. Then, in step 4024, the client sends a fetch-data request to the web server to begin receiving search results. After sending the session ID to the client computer, the web server formulates a document query to initiate return of search results by the backend to the web server, in step 4025. In step 4026, the web server sends the query to the backend web service. In step 4027, the web service receives the document query. In step 4028, the backend processes the query and sends a first set of results, in step 4029, back to the web server. Meanwhile, in step 4030, the web server receives the fetch-

data request sent by the client computer and, in step 4032, waits to respond to the request until search-results data becomes available.

Turning to Figure 40C, in step 4036, the web server receives document data from the backend and stores the received data in the key/value cache (3908 in Figure 39). In step 4038, the web server begins constructing a client-side data structure, or client graph, to forward to the client and concurrently begins constructing a graph from the data in the graph database. When the message containing the received data does not indicate that transfer of data is complete, as determined in step 4039, then, in step 4040, the web server requests more data from the backend, which receives the request in step 4042 and begins processing the request. Ellipses 4044 and 4046 indicate that additional processing occurs in the transfer of data from the backend to the web server. When enough data has been processed by the web server to create a sufficient amount of data stored in the client-side data structure, as determined in step 4050, then at least a portion of the data is sent to the client in step 4051. The data is received by the client in step 4052. When the received data does not include an indication that the search results are complete, as determined in step 4054, then the client sends a next fetch request, in step 4056, to the web server, which receives the fetch request in step 4057 and, in step 4058, waits to respond until sufficient additional data has been processed. Again, ellipses 4059 and 4060 indicate additional processing. The ellipses are used because data transfer from the backend to the web server and from the web server to the client proceeds in parallel until all the data has been transferred. The data transfer involves numerous asynchronously and concurrently operating processes and/or threads.

Turning to Figure 40D, in step 4062, the backend receives a next request for more data from the web server and, in step 4063, returns a final data message with the final search results with an indication that the data is not complete. In step 4064, the web server receives the data and stores the data in the key/value cache. In step 4065, the web server continues, in parallel, constructing the client-side data structure and adding nodes and edges to the graph stored in the graph database. In step 4066, the web server determines that the data is complete and therefore undertakes the *while*-loop that begins with step 4067. Turning to Figure 40E, in the

while-loop of step 4067, the web server determines whether there is additional data to send to the client, in step 4068. When there is additional data, the additional data is sent to the client in step 4069. The client receives the data in step 4070 and then issues a next fetch request, in step 4071, that is sent to the web server. The web server receives the next fetch request, in step 4072, in parallel continuing constructing the client-side data structure and graph stored in the graph database in step 4073. When the client-side data structure and graph stored in the graph database are complete, as determined in step 4074, then the web server sends final data to the client, in step 4075, along with an indication that the graph is complete. In step 4080, the client computer receives the data and stores it in the client-side data structure local to the client computer (3904 in Figure 39). Determining that the search results are complete, in step 4082, the client renders the search results contained in the client-side data structure for display to the user in step 4083.

Figure 40F illustrates a subsequent operation invoked by a user of the 3D viewer on the client computer. In step 4085, the user interacts with the user interface of the 3D viewer to issue a graph query. In step 4086, the 3D viewer transmits the graph query to the web server. In step 4087, the web server receives the graph query and caches the graph query in the query cache (3912 in Figure 39). In step 4088, the web service submits the graph query to the graph database. In step 4089, the web server receives a response from the graph database and caches the response, in step 4090, in the query results cache (3914 in Figure 39). Also in step 4090, the web server returns the query response to the client computer which receives the response, in step 4092, and renders the data received in the response for display to the user. Figure 40F assumes that the query received in step 4087 is a new query not previously executed on behalf of the user. Otherwise, were cached results available from a previous execution of the query, the cached results can be directly returned rather than querying the graph database. However, logic must be included to invalidate cached queries when the search results and graph stored in the graph database are altered by user operations.

Figures 40A-F are intended to show an example of a series of transactions between the 3D viewer, web server, and web service to produce three-

dimensional search results in a display window displayed by the 3D viewer to the user. Many other types of interactions occur in the various implementations of the currently described system. However, they follow a similar pattern of execution. The 3D viewer produces requests and commands that are transmitted to the web server which executes the request and commands on behalf of a user, in some cases via additional transactions with the backend web service and/or with the graph database. The above-described logical data stores and component implementations support a rich and varied document-search facility and three-dimensional search-results display. In addition, a wide variety of graph queries, graph navigation, and other operations are supported.

Although the present invention has been described in terms of particular embodiments, it is not intended that the invention be limited to these embodiments. Modifications within the spirit of the invention will be apparent to those skilled in the art. For example, any of many design and implementation parameters, such as programming language, operating system, data structures, control structures, modular organization, and other such parameters may be varied to produce a large number of possible implementations of the currently described system. Any of many different types of graph databases may be incorporated within the web server to provide graph-query functionality. The data may be stored and transferred in many additional ways from a remote web server to the 3D viewer in order to provide search results to users by the 3D viewer. Although the current discussion has focused on searching, and displaying search results related to, electronic documents, the 3D viewer, web server, and backend web service can be implemented to search many other types of electronically stored information and provide graphical representations of the search results. For example, the system may be used not only for term or phrase searches of documents, but may also be used to produce three-dimensional representations of citation networks within scientific papers and article, issued patents and patent applications directed to related subject matter, commonly owned or licensed, or both, and even complex machines, systems, and processes described by a multitude of component descriptions.

It is appreciated that the previous description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the present disclosure. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without departing from the spirit or scope of the disclosure. Thus, the present disclosure is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

CLAIMS

1. A document-searching system comprising:

a user device having at least one processor, one or more memories, a network interface, an electronic display, and processor instructions, stored in one or more of the one or more memories that, when executed on the processor, control the user device to

receive user input,

transmit requests to a remote server through the network interface, including document-search requests,

receive responses to the transmitted requests from the remote server through the network interface, including document search results, and

display a graphical user interface on the electronic display, the graphical user interface displaying

a graph-like three-dimensional representation of a document search process and corresponding search results, and

selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database.

2. The document-searching system of claim 1 wherein the graph-like three-dimensional representation of the document search process and corresponding search results comprises term nodes, document nodes, and links that link term nodes to document nodes.

3. The document-searching system of claim 2 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include:

a selection feature that, when selected by a user, results in display of a variety of different navigation tools using which a user navigate within the graph-like three-dimensional representation of the document search process and corresponding search results;

a selection feature that, when selected by a user, undoes the last completed operation;

a selection feature that, when selected by a user, provides an eraser tool to allow for nodes to be deleted from the graph-like three-dimensional representation of the document search process and corresponding search results;

a selection feature that, when selected by a user, invokes various different types of expansion functionalities that expand term nodes with additional searches and that expand document nodes with document analyses;

a selection feature that, when selected by a user, invokes a lasso operation for defining groups of nodes, the defined groups having selectable alternative display modes;

a selection feature that, when selected by a user, invokes a path-definition utility for defining paths of nodes through the graph, the defined paths having selectable alternative display modes; and

a selection feature that, when selected by a user, invokes a search-in-graph functionality that searches document nodes within the graph-like three-dimensional representation of the document search process and corresponding search results.

4. The document-searching system of claim 2 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include:

a selection feature that, when selected by a user, directs the user device to forward a graph query, input by the user, to the remote server for execution by a graph database incorporated within, or accessed by, the remote server.

5. The document-searching system of claim 2 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include a main-menu selection wheel that includes:

an icon that, when selected by a user, invokes a locator-menu selection wheel;

an icon that, when selected by a user, invokes a new search and display of a new graph-like representation of the search process and search results;

an icon that, when selected by a user, toggles between three-dimensional and two-dimensional representations of the search process and search results;

an icon that, when selected by a user, invokes a heat-map representation of search results;

an icon that, when selected by a user, toggles between displaying and not displaying titles for nodes;

an icon hat, when selected by a user, invokes a help utility; and

an icon that, when selected by a user, invokes an aggregation utility that can aggregate the outer level of leaf nodes in order to facilitate viewing lower-level layers of nodes in a displayed graph-like three-dimensional representation of the document search process and corresponding search results.

6. The document-searching system of claim 2 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include a locator-menu selection wheel that includes:

an icon that, when selected by a user, invokes selection of groups and/or paths for automated navigation;

an icon that, when selected by a user, invokes a utility for selection of document categories for automatic navigation;

an icon that, when selected by a user, selects attribute-value-based node location; and

an icon that, when selected by a user, provides for automatic navigation to nodes representing added terms in sequential searching.

7. The document-searching system of claim 1

wherein the user device receives and stores a client-side data structure that includes the search results transmitted to the document-searching system by the remote server;

wherein the remote server constructs a data structure that includes search results transmitted to the document-searching system by the remote server; and

wherein the remote server also constructs and stores a graph in a graph database incorporated within, or accessed by, the remote server that represents the search results transmitted to the document-searching system by the remote server.

8. The document-searching system of claim 7 wherein the remote server stores additional search results in the graph database for subsequent transfer to the user device in order to facilitate subsequent operations requested by the user device.

9. The document-searching system of claim 7 wherein the remote server additionally includes, or accesses, a search-results store in which the remote server temporarily stores document-search results received from a remote web service that provides a document-search interface to the remote server through which the remote server executes document-search queries on behalf of the user device.

10. The document-searching system of claim 7 wherein the remote server additionally includes, or accesses, graph-query and graph-query-response caches for storing graph queries received from the user device and graph-query-results returned by the graph database.

11. A document-searching method comprising:

providing a user device having at least one processor, one or more memories, a network interface, and an electronic display, the user device receiving user input, transmitting requests to a remote server through the network interface, including document-search requests, and receiving responses to the transmitted requests from the remote server through the network interface, including document search results;

displaying a graphical user interface on the electronic display that includes selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database; and

in response to invocation of a document search, displaying a graph-like three-dimensional representation of a document search process and corresponding search results.

12. The method of claim 11 wherein the graph-like three-dimensional representation of the document search process and corresponding search results comprises term nodes, document nodes, and links that link term nodes to document nodes.

13. The method of claim 12 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include:

a selection feature that, when selected by a user, results in display of a variety of different navigation tools using which a user navigate within the graph-like three-dimensional representation of the document search process and corresponding search results;

a selection feature that, when selected by a user, undoes the last completed operation;

a selection feature that, when selected by a user, provides an eraser tool to allow for nodes to be deleted from the graph-like three-dimensional representation of the document search process and corresponding search results;

a selection feature that, when selected by a user, invokes various different types of expansion functionalities that expand term nodes with additional searches and that expand document nodes with document analyses;

a selection feature that, when selected by a user, invokes a lasso operation for defining groups of nodes, the defined groups having selectable alternative display modes;

a selection feature that, when selected by a user, invokes a path-definition utility for defining paths of nodes through the graph, the defined paths having selectable alternative display modes; and

a selection feature that, when selected by a user, invokes a search-in-graph functionality that searches document nodes within the graph-like three-dimensional representation of the document search process and corresponding search results.

14. The method of claim 12 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include:

a selection feature that, when selected by a user, directs the user device to forward a graph query, input by the user, to the remote server for execution by a graph database incorporated within, or accessed by, the remote server.

15. The method of claim 12 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include a main-menu selection wheel that includes:

an icon that, when selected by a user, invokes a locator-menu selection wheel;

an icon that, when selected by a user, invokes a new search and display of a new graph-like representation of the search process and search results;

an icon that, when selected by a user, toggles between three-dimensional and two-dimensional representations of the search process and search results;

an icon that, when selected by a user, invokes a heat-map representation of search results;

an icon that, when selected by a user, toggles between displaying and not displaying titles for nodes;

an icon hat, when selected by a user, invokes a help utility; and

an icon that, when selected by a user, invokes an aggregation utility that can aggregate the outer level of leaf nodes in order to facilitate viewing lower-level layers of nodes in a displayed graph-like three-dimensional representation of the document search process and corresponding search results.

16. The method of claim 12 wherein selection features that, when activated by user input, invoke document searches, rotate the graph-like three-dimensional representation, alter the three-dimensional representation, and invoke execution of graph queries input by the user and executed by remote server using a graph database include a locator-menu selection wheel that includes:

an icon that, when selected by a user, invokes selection of groups and/or paths for automated navigation;

an icon that, when selected by a user, invokes a utility for selection of document categories for automatic navigation;

an icon that, when selected by a user, selects attribute-value-based node location; and

an icon that, when selected by a user, provides for automatic navigation to nodes representing added terms in sequential searching.

17. The method of claim 11

wherein the user device receives and stores a client-side data structure that includes the search results transmitted to the document-searching system by the remote server;

wherein the remote server constructs a data structure that includes search results transmitted to the document-searching system by the remote server; and

wherein the remote server also constructs and stores a graph in a graph database incorporated within, or accessed by, the remote server that represents the search results transmitted to the document-searching system by the remote server.

18. The method of claim 17 wherein the remote server stores additional search results in the graph database for subsequent transfer to the user device in order to facilitate subsequent operations requested by the user device.

19. The method of claim 17 wherein the remote server additionally includes, or accesses, a search-results store in which the remote server temporarily stores document-search results received from a remote web service that provides a document-search interface to the remote server through which the remote server executes document-search queries on behalf of the user device.

20. The method of claim 71 wherein the remote server additionally includes, or accesses, graph-query and graph-query-response caches for storing graph queries received from the user device and graph-query-results returned by the graph database.

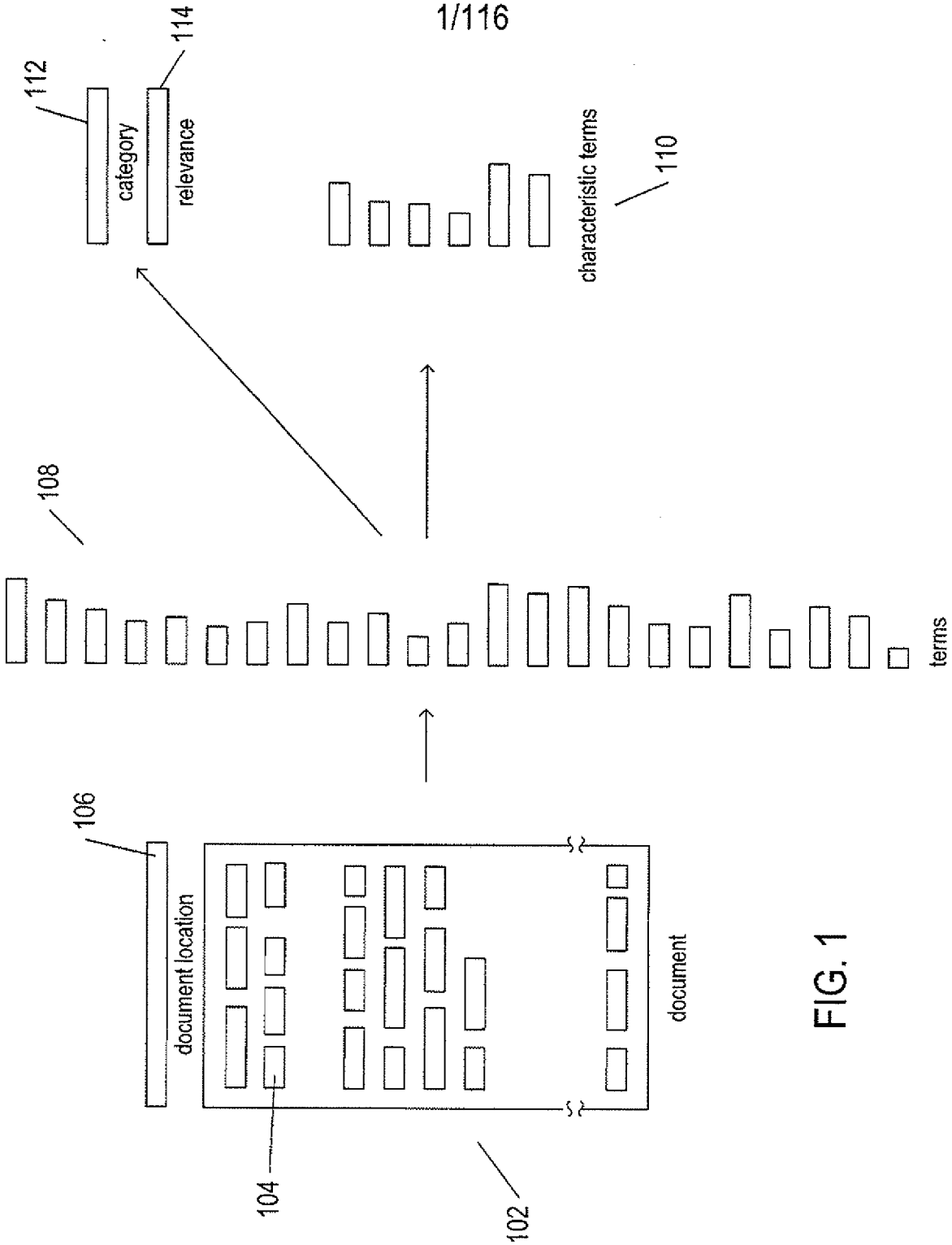

112
114
category
relevance
characteristic terms
110
108
terms
106
document location
document
104
102

FIG. 1

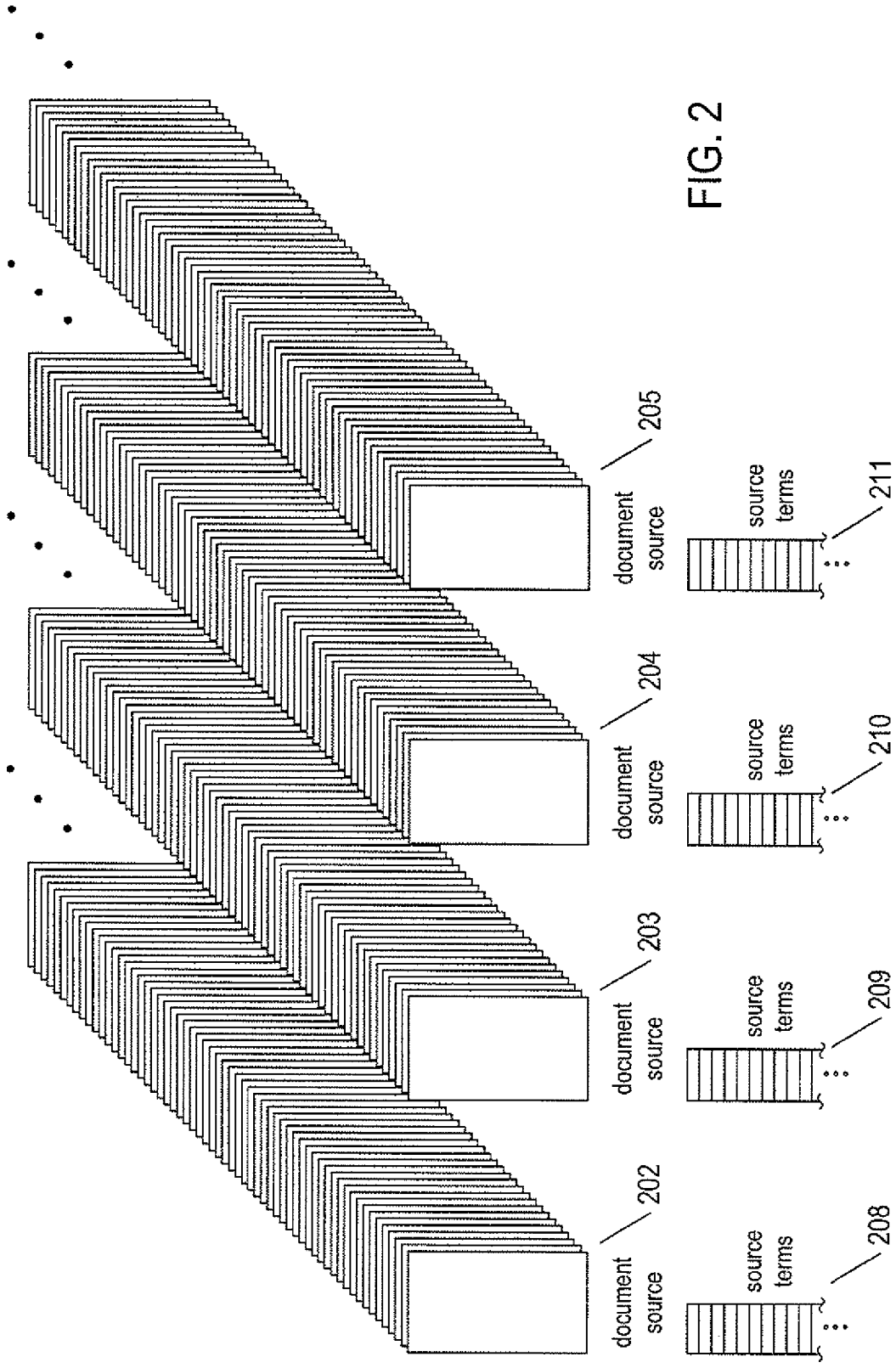

document source
202
source terms
208
document source
203
source terms
209
document source
204
source terms
210
document source
205
source terms
211

FIG. 2

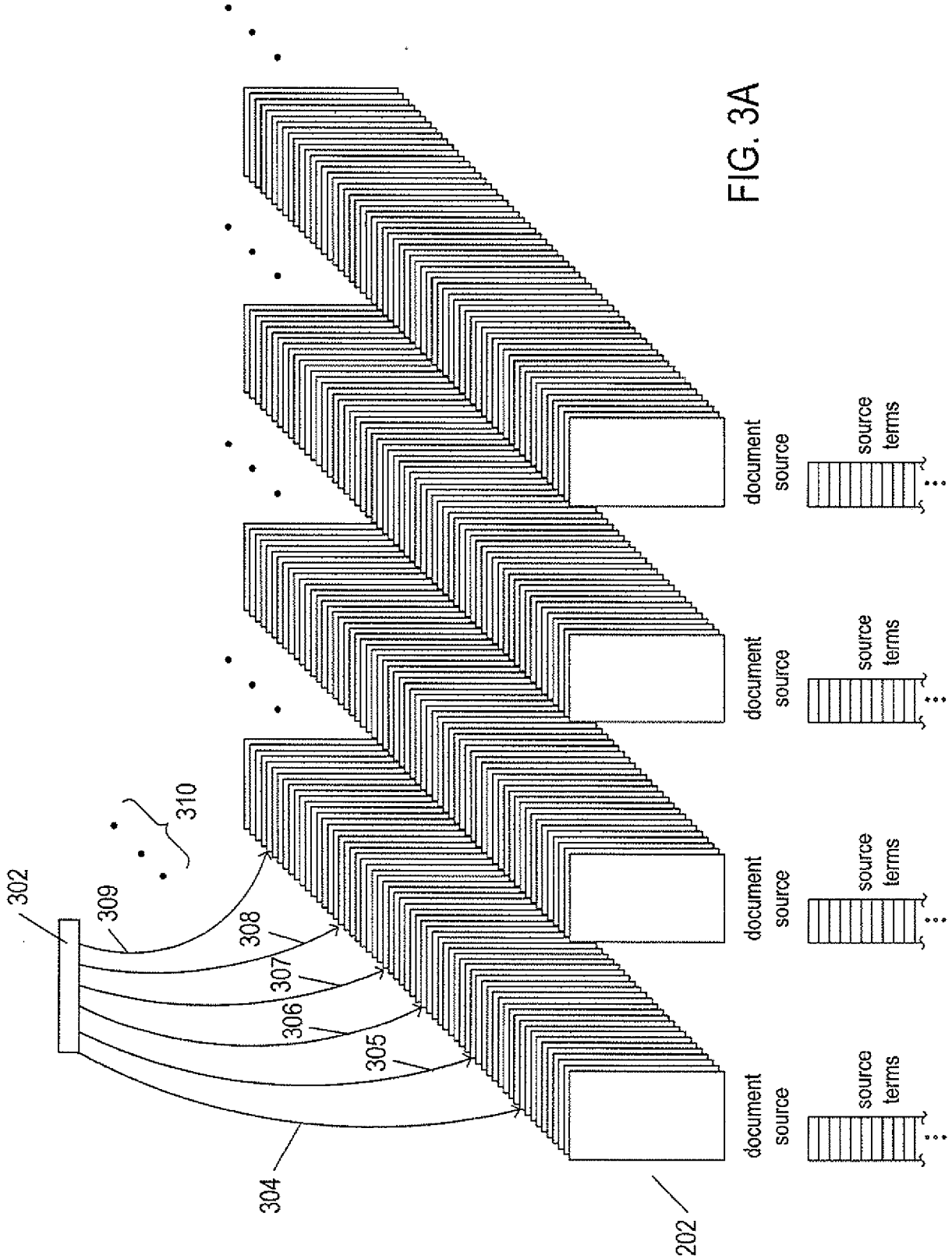

302
309
310
308
307
306
305
304
202
document source
source terms
document source
source terms
document source
source terms
document source
source terms

FIG. 3A

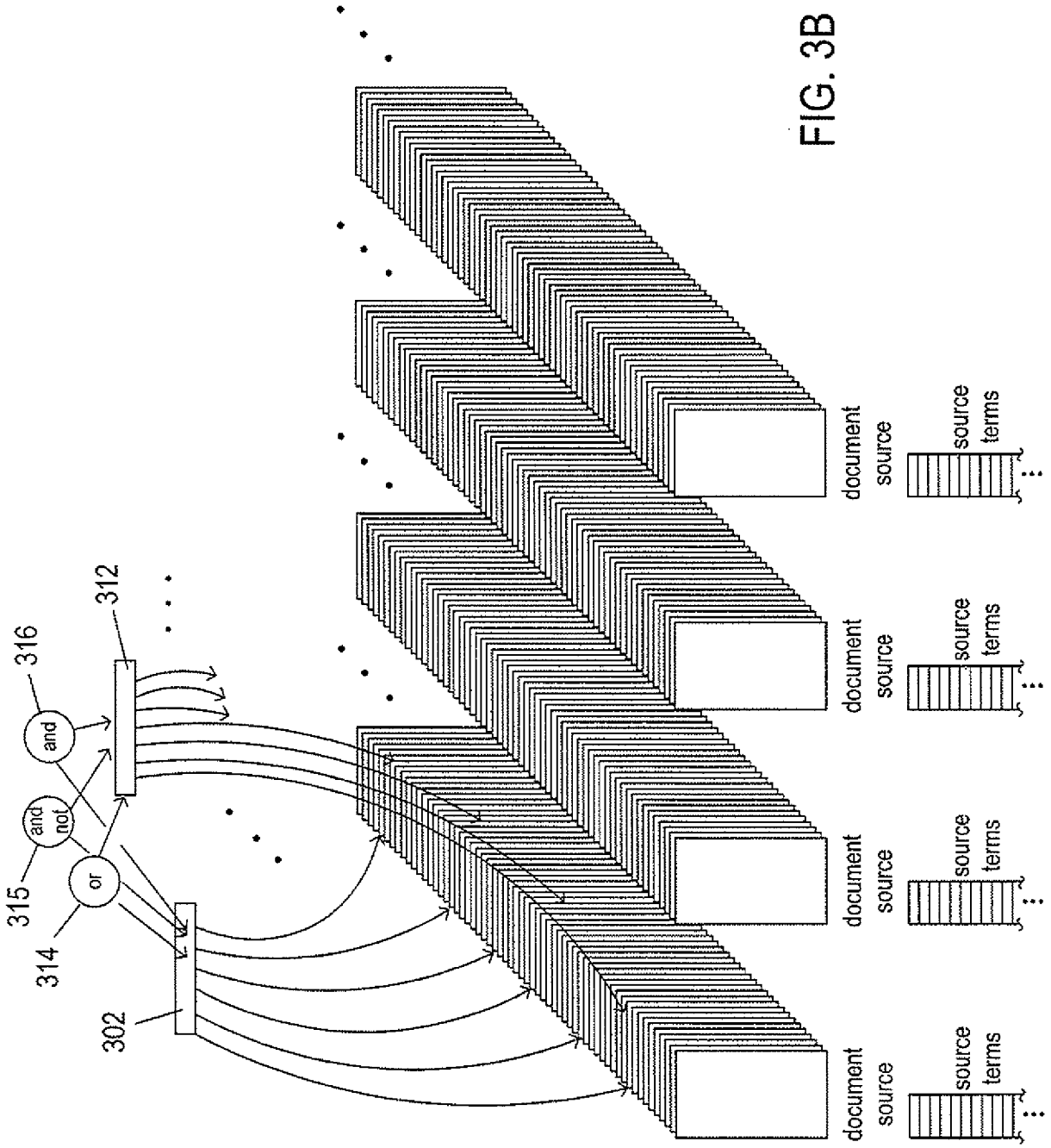

316
312
and
and not
or
315
314
302
document source
source terms
document source
source terms
document source
source terms
document source
source terms

FIG. 3B

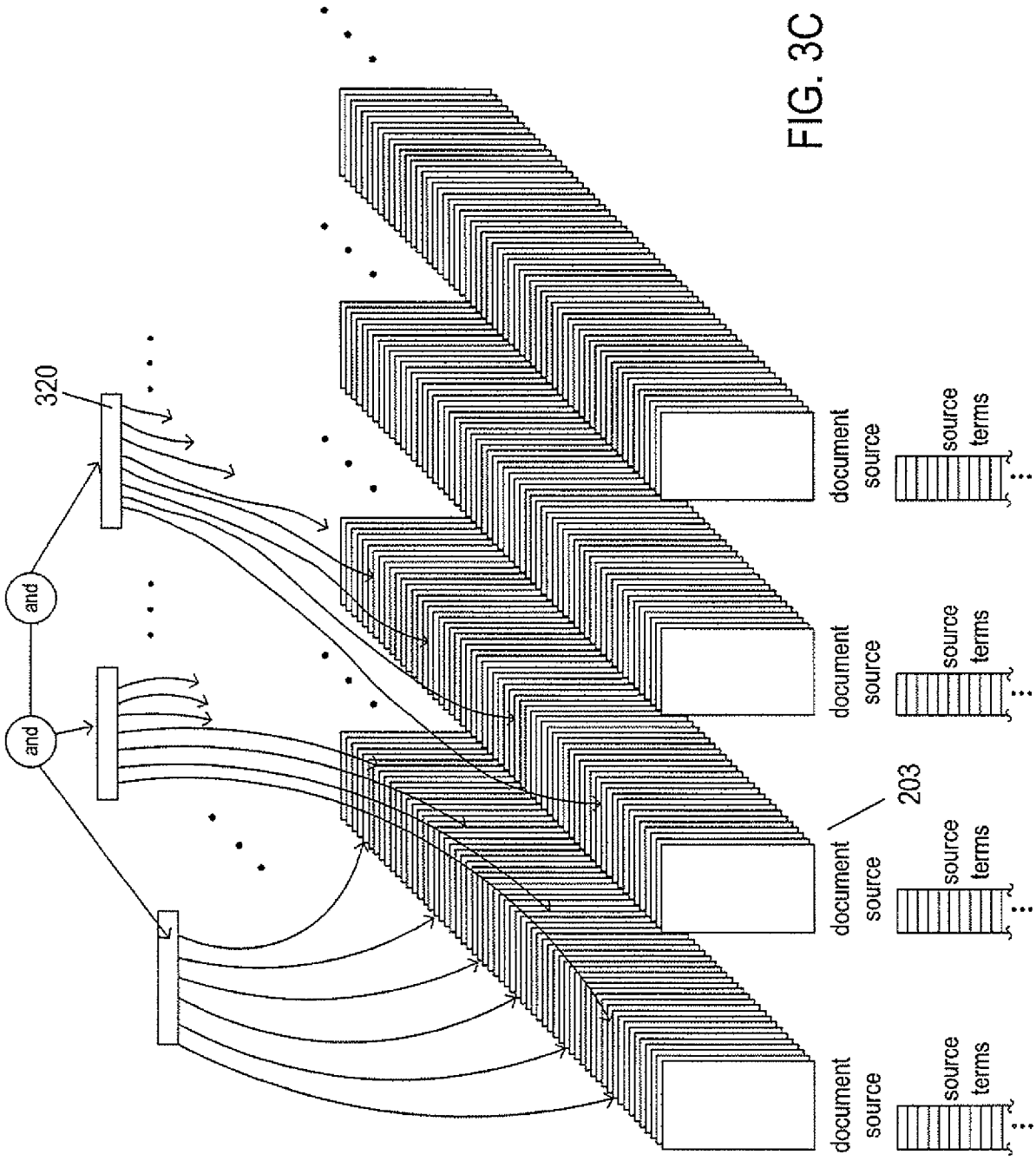


FIG. 3C

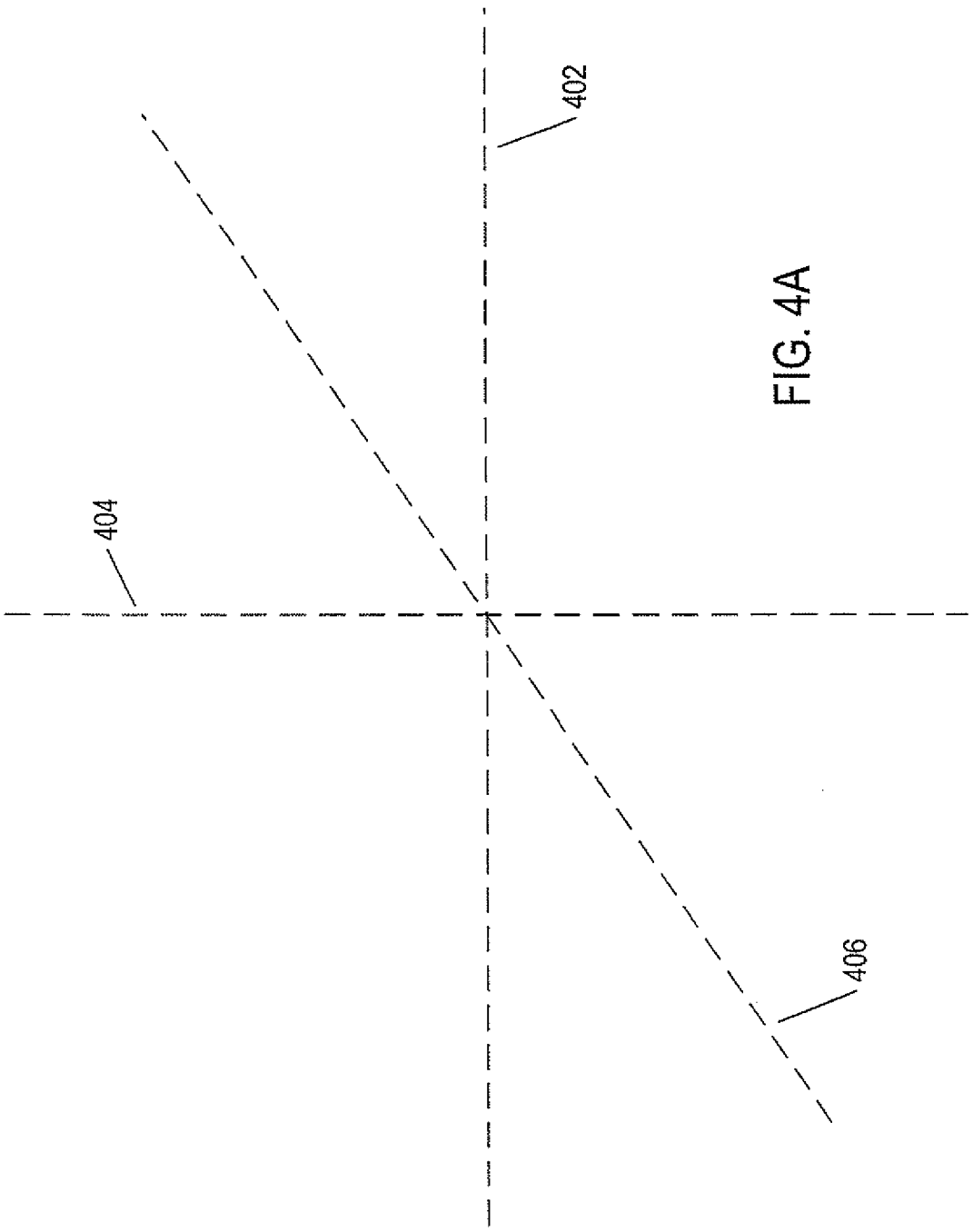


FIG. 4A

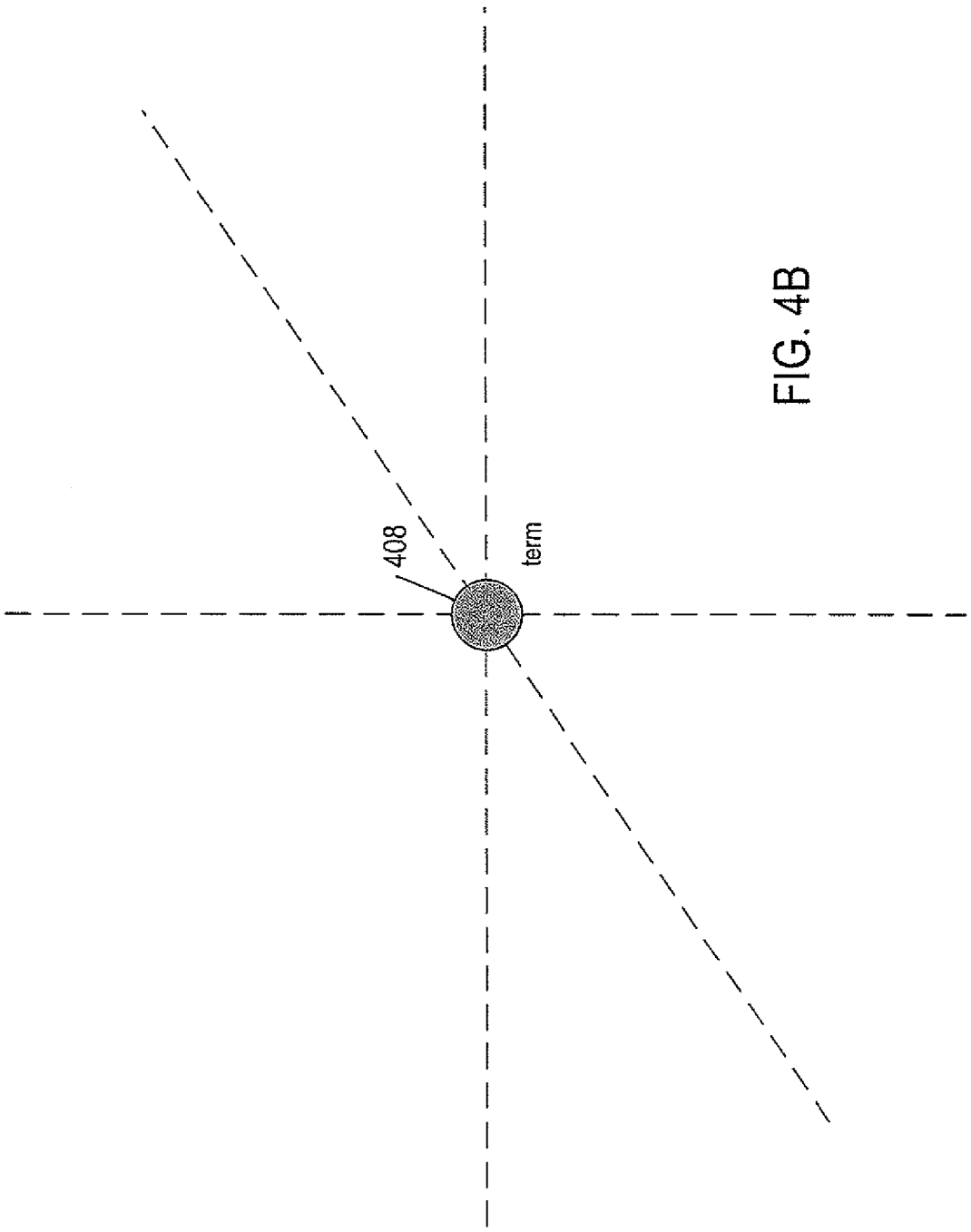


FIG. 4B

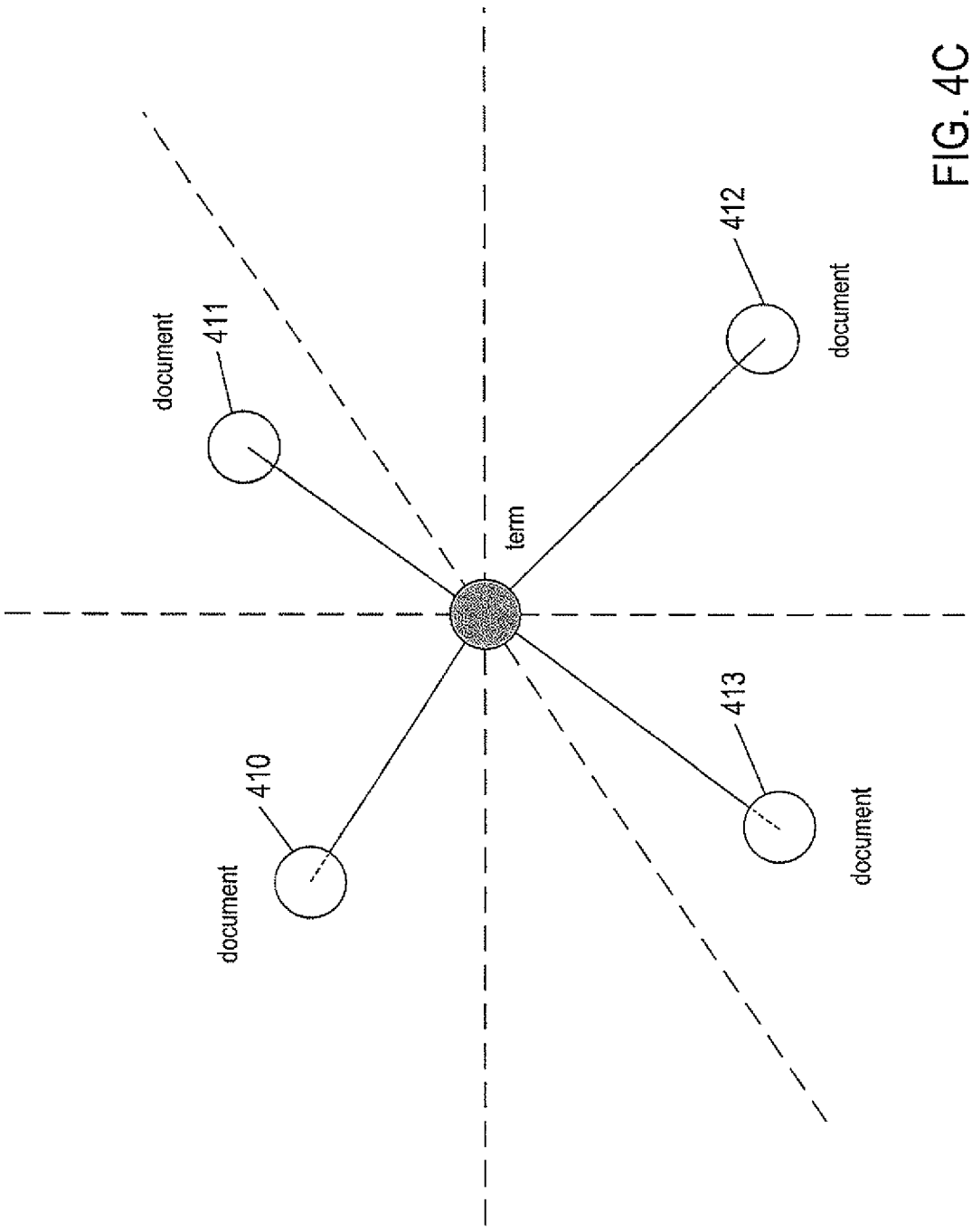
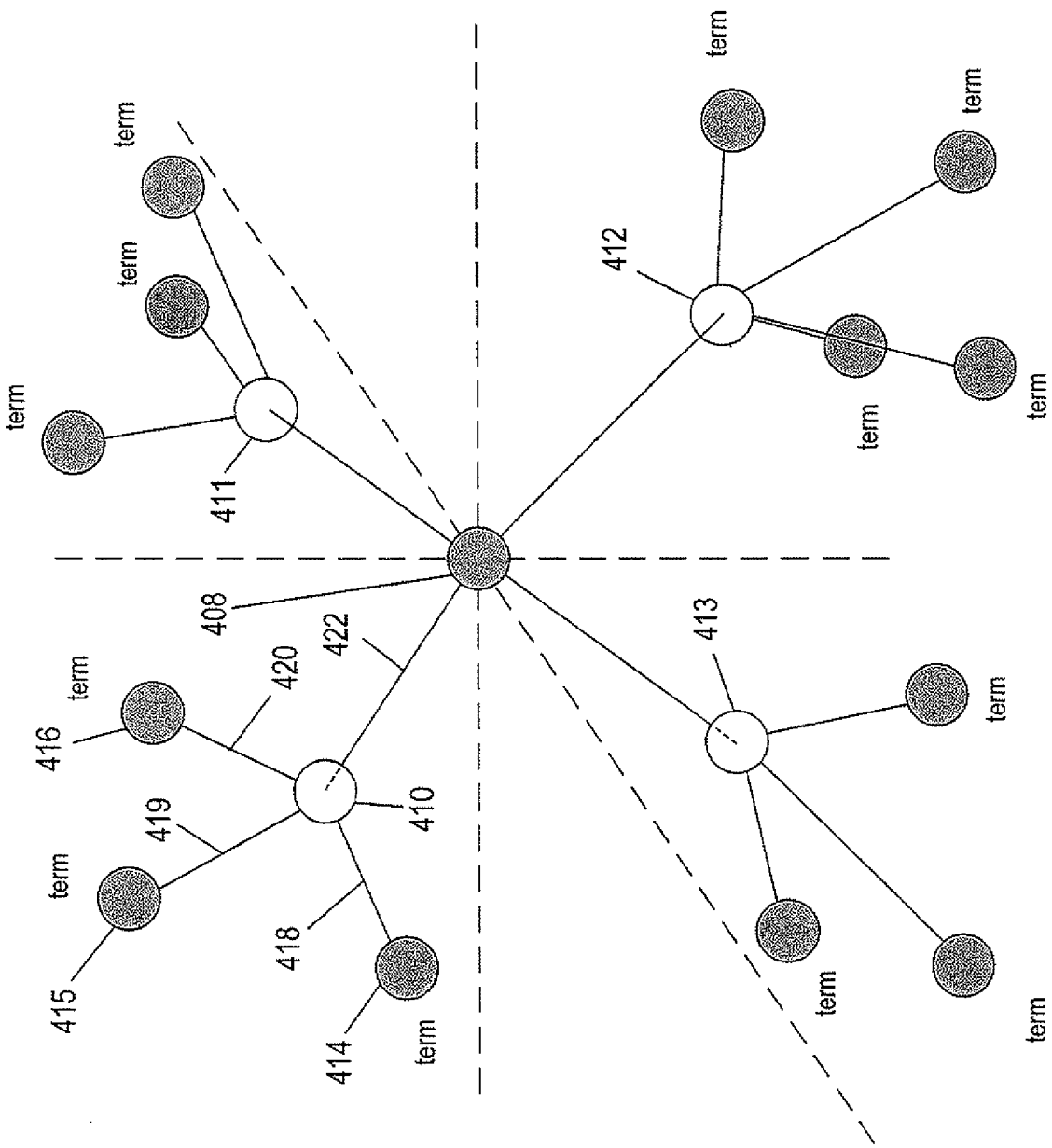


FIG. 4C

FIG. 4D


term
term
term
411
412
term
term
term
term
408
422
420
416
term
419
410
term
415
418
414
term
413
term
term
term
term

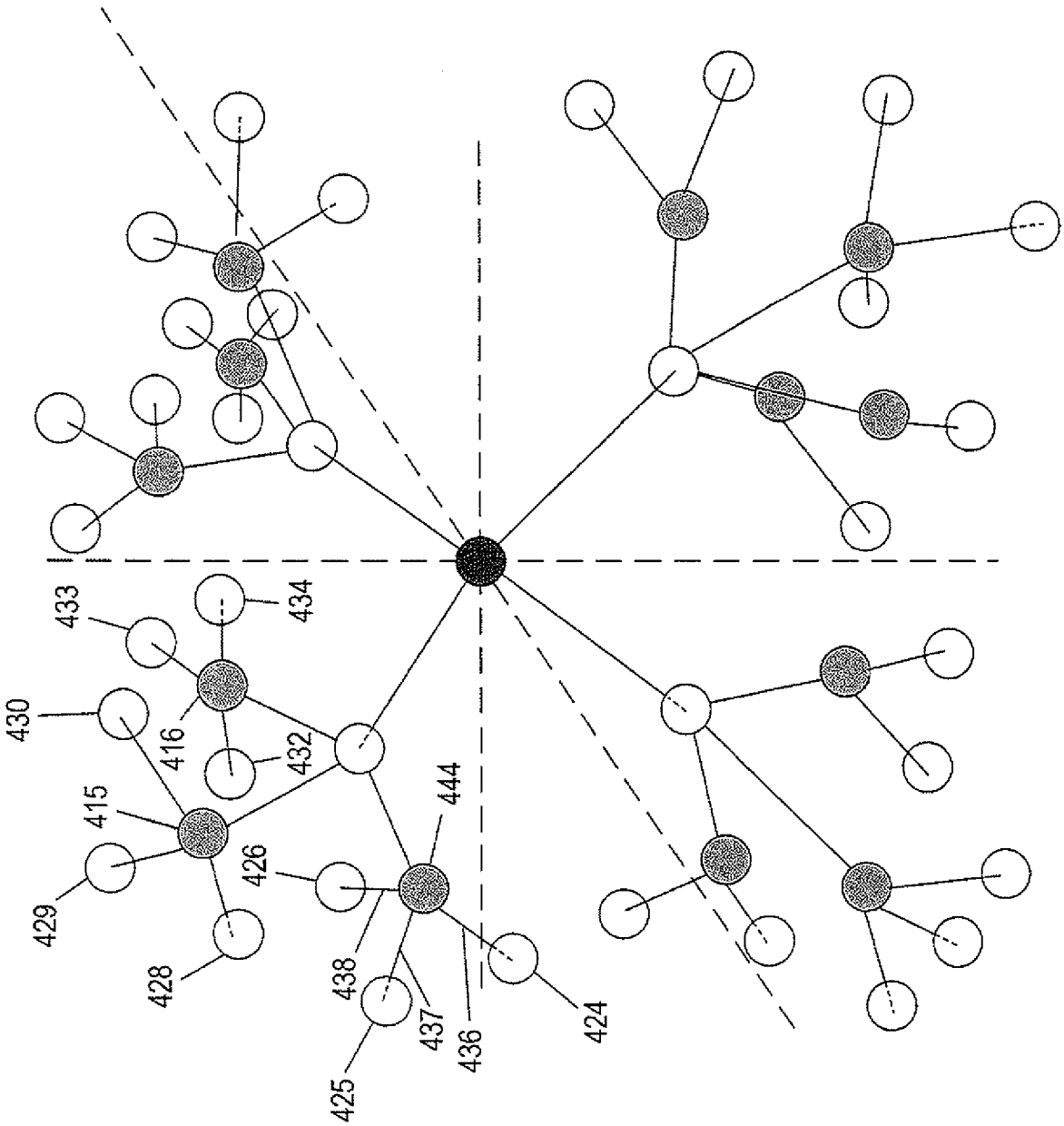


FIG. 4E

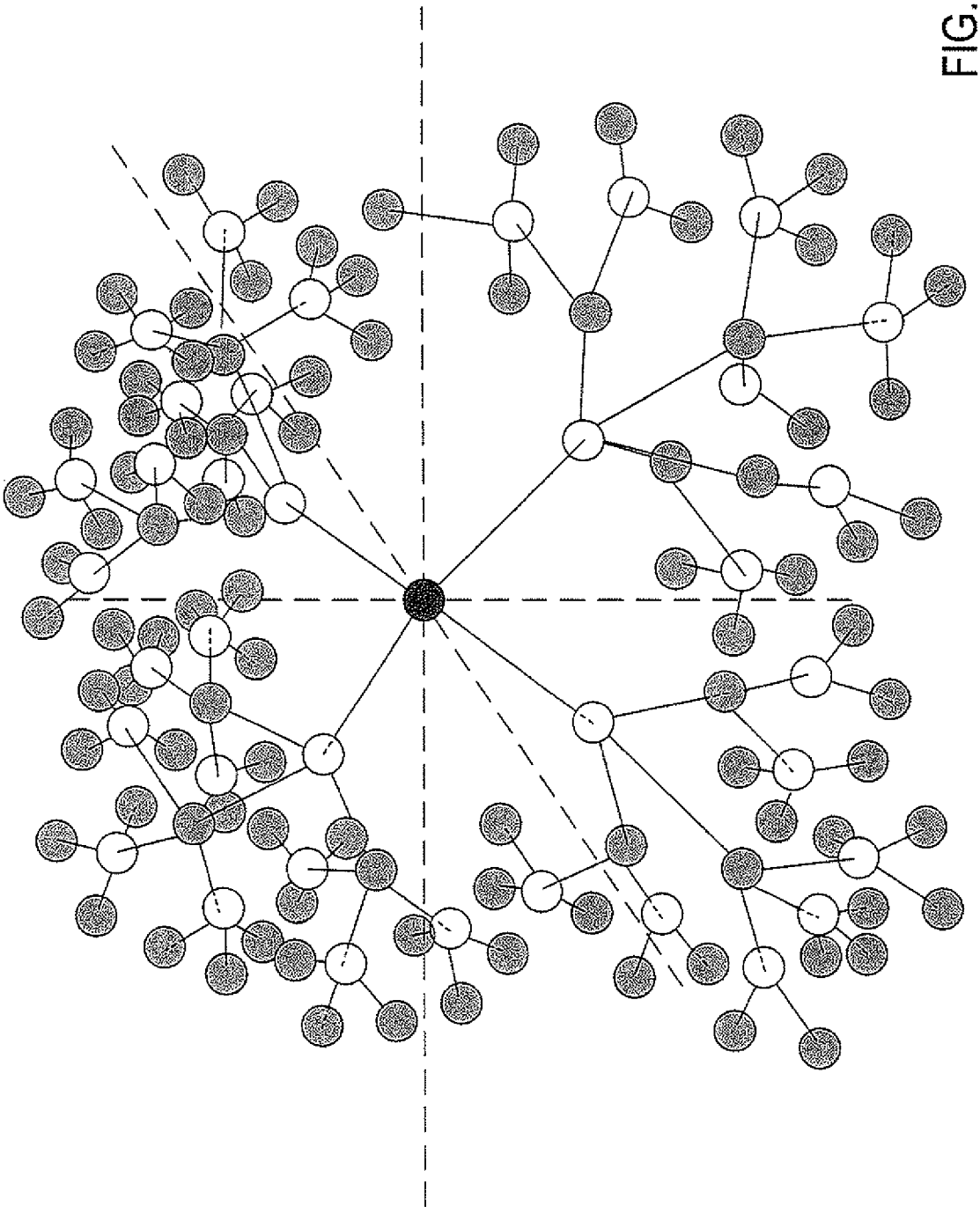


FIG. 4F

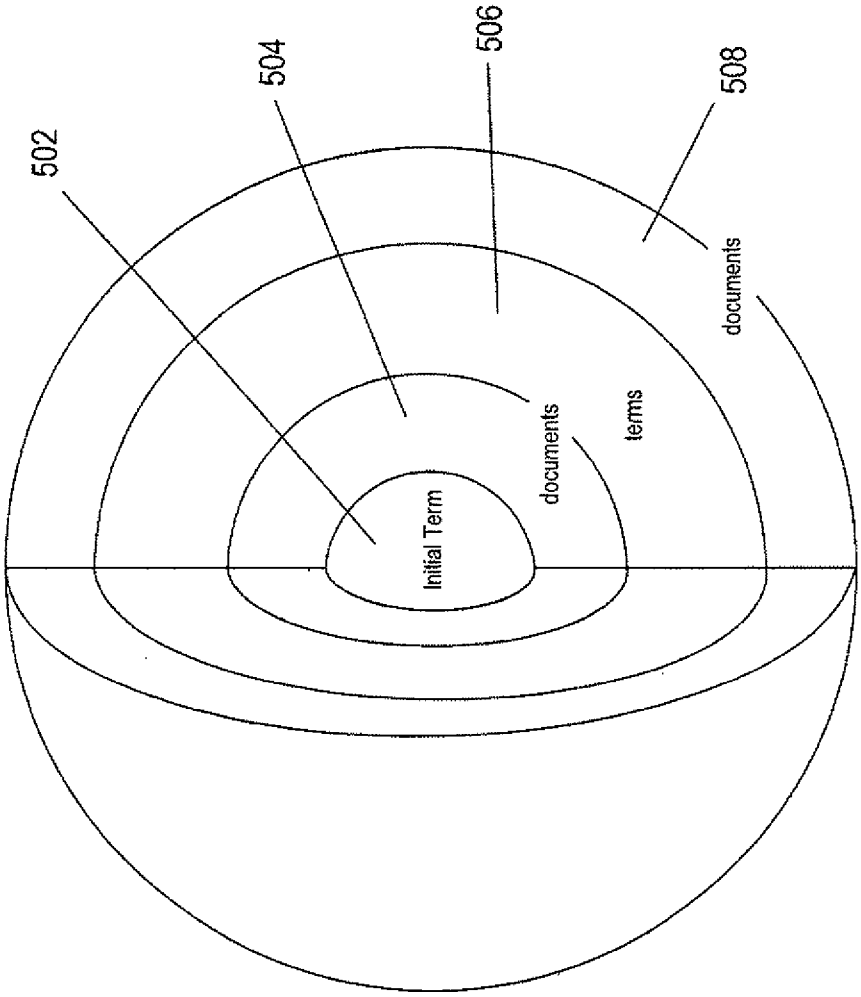


FIG. 5

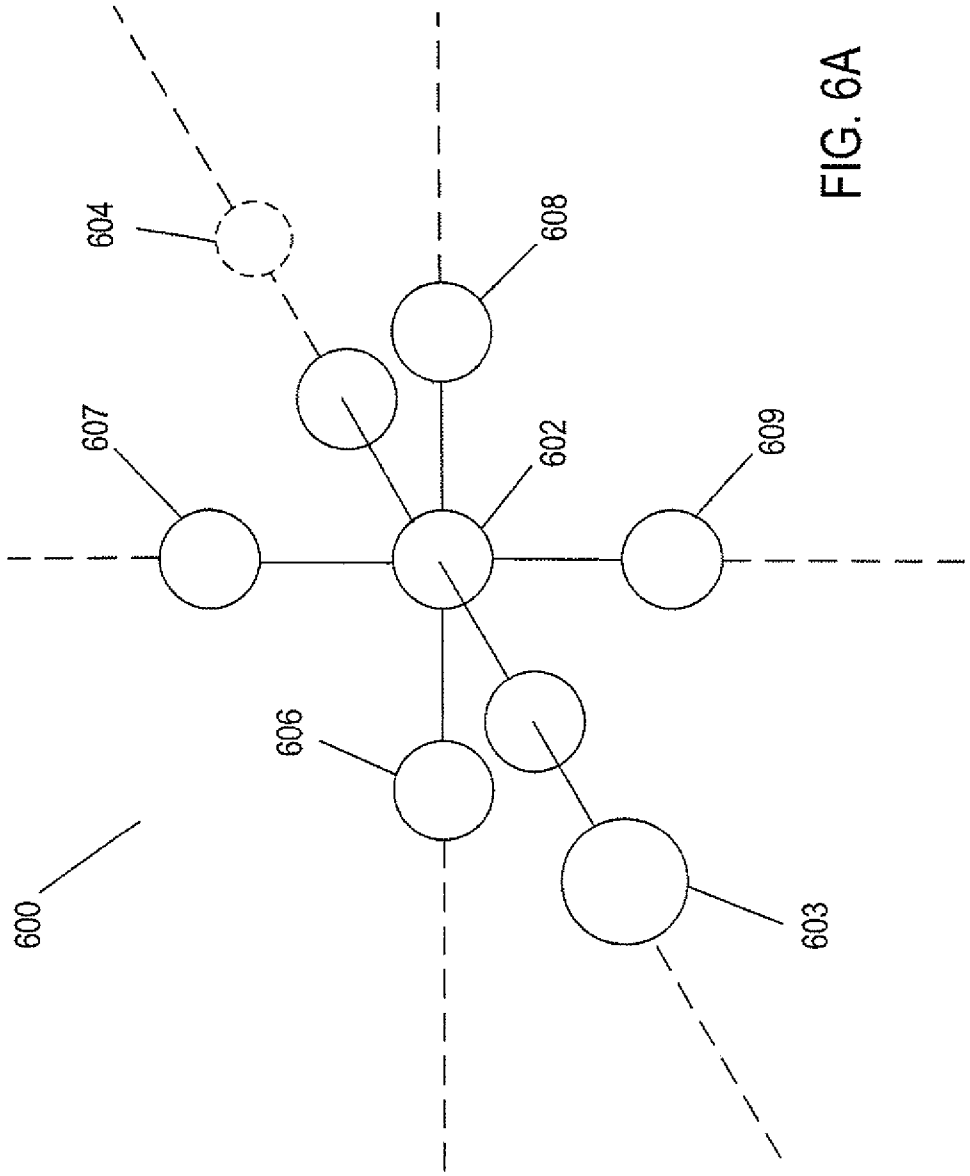


FIG. 6A

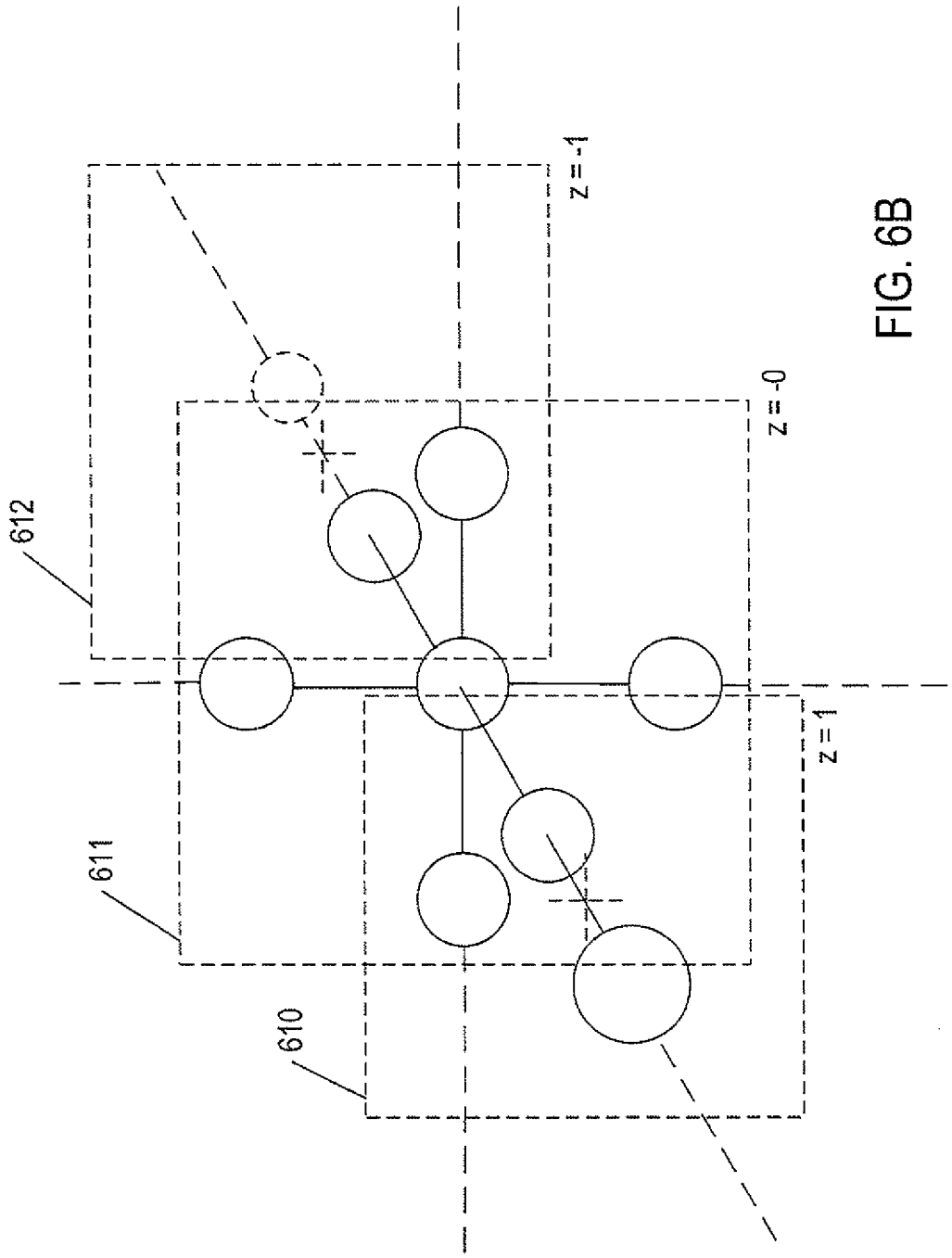
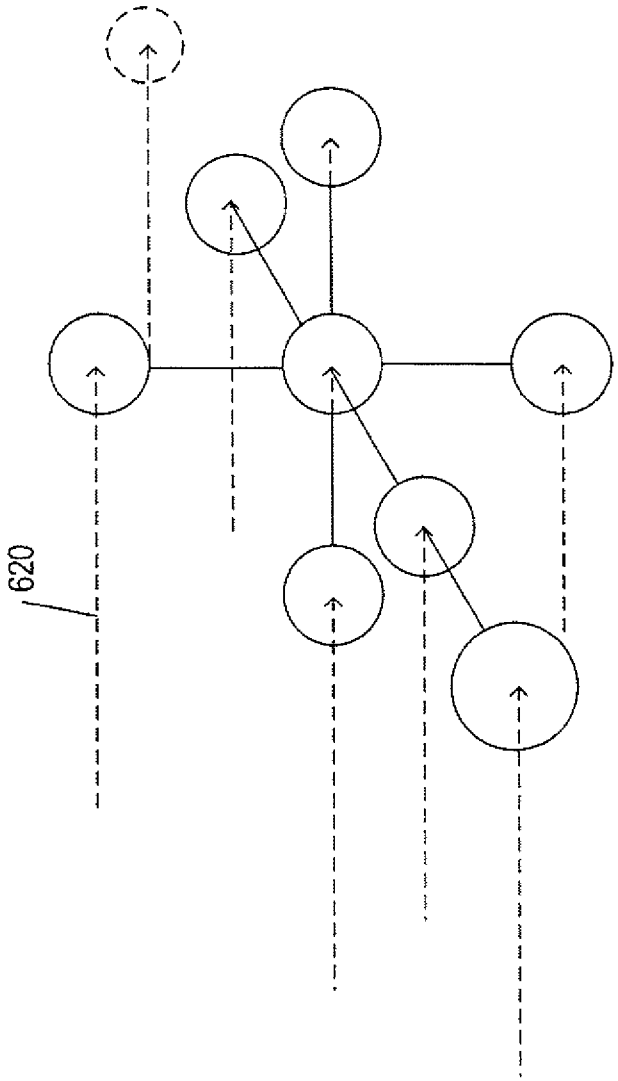


FIG. 6B

FIG. 6C



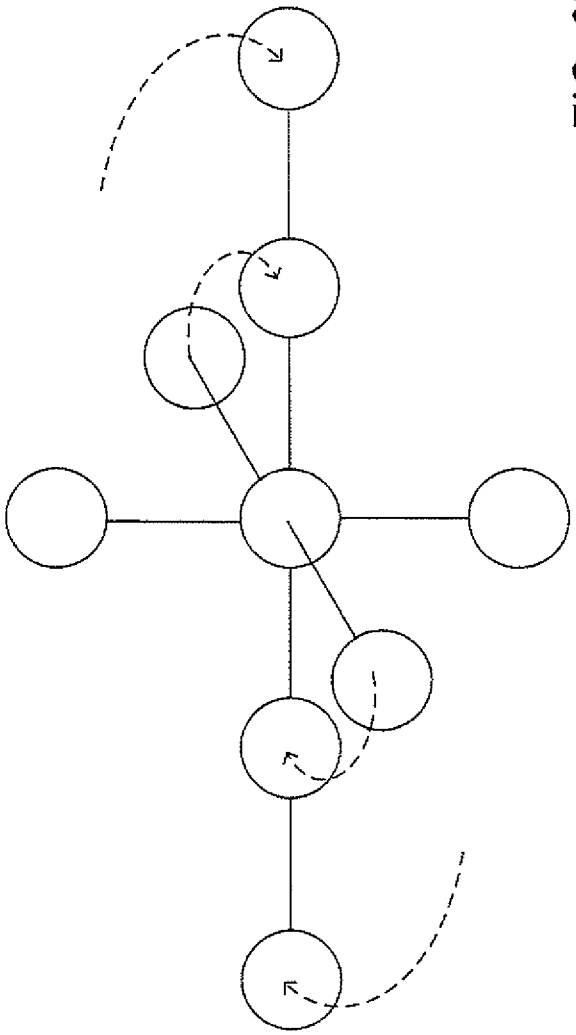


FIG. 6D

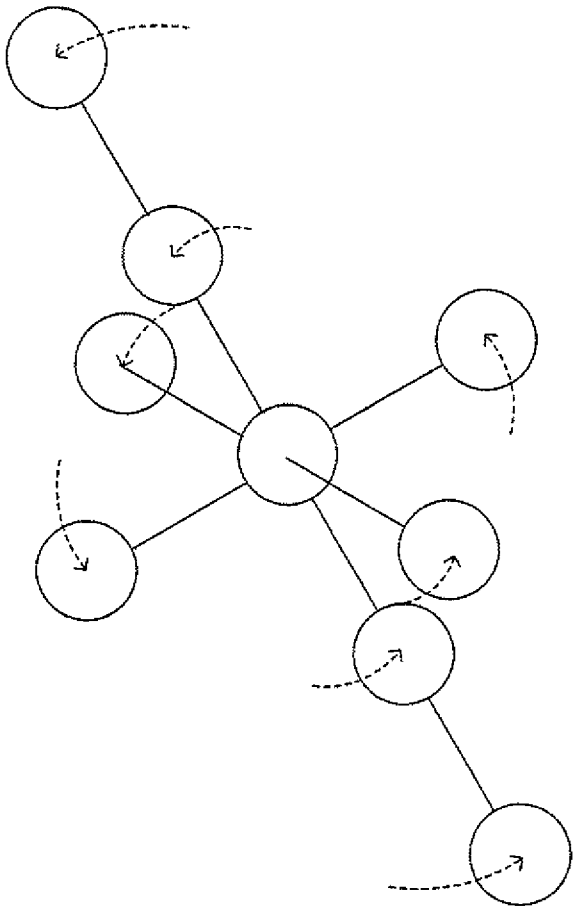


FIG. 6E

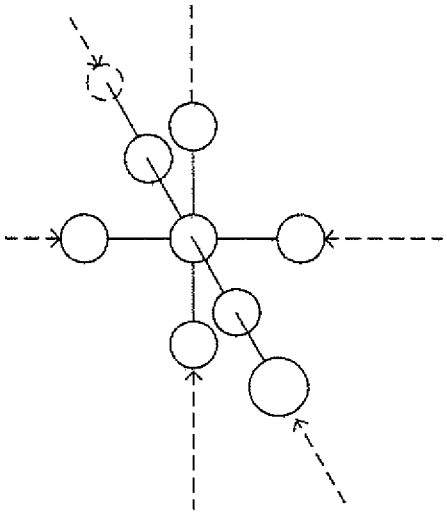


FIG. 6F

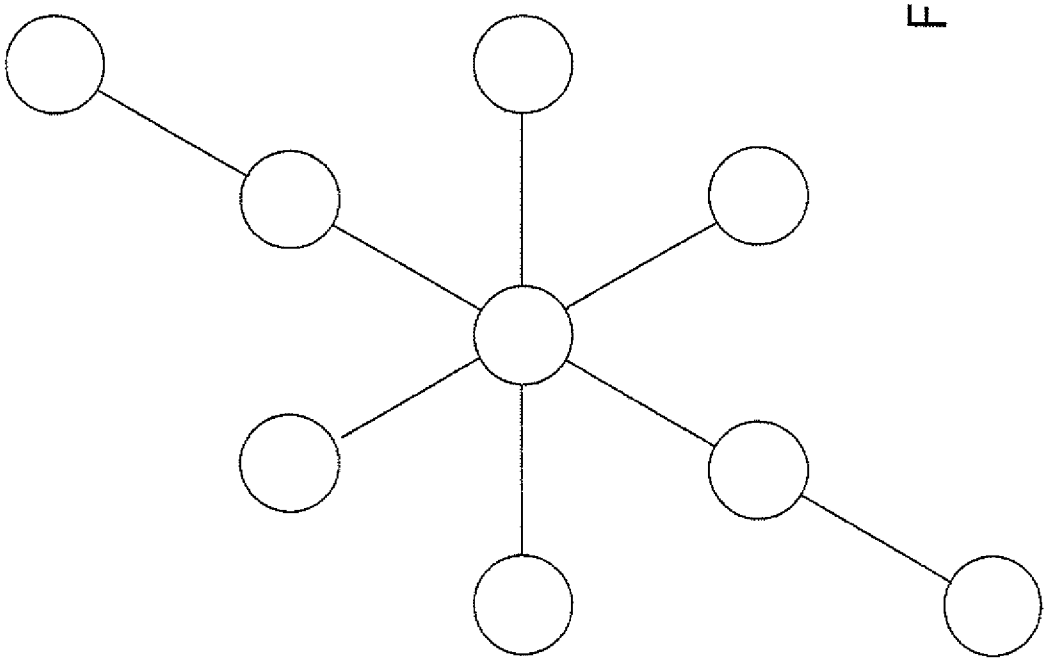


FIG. 6G

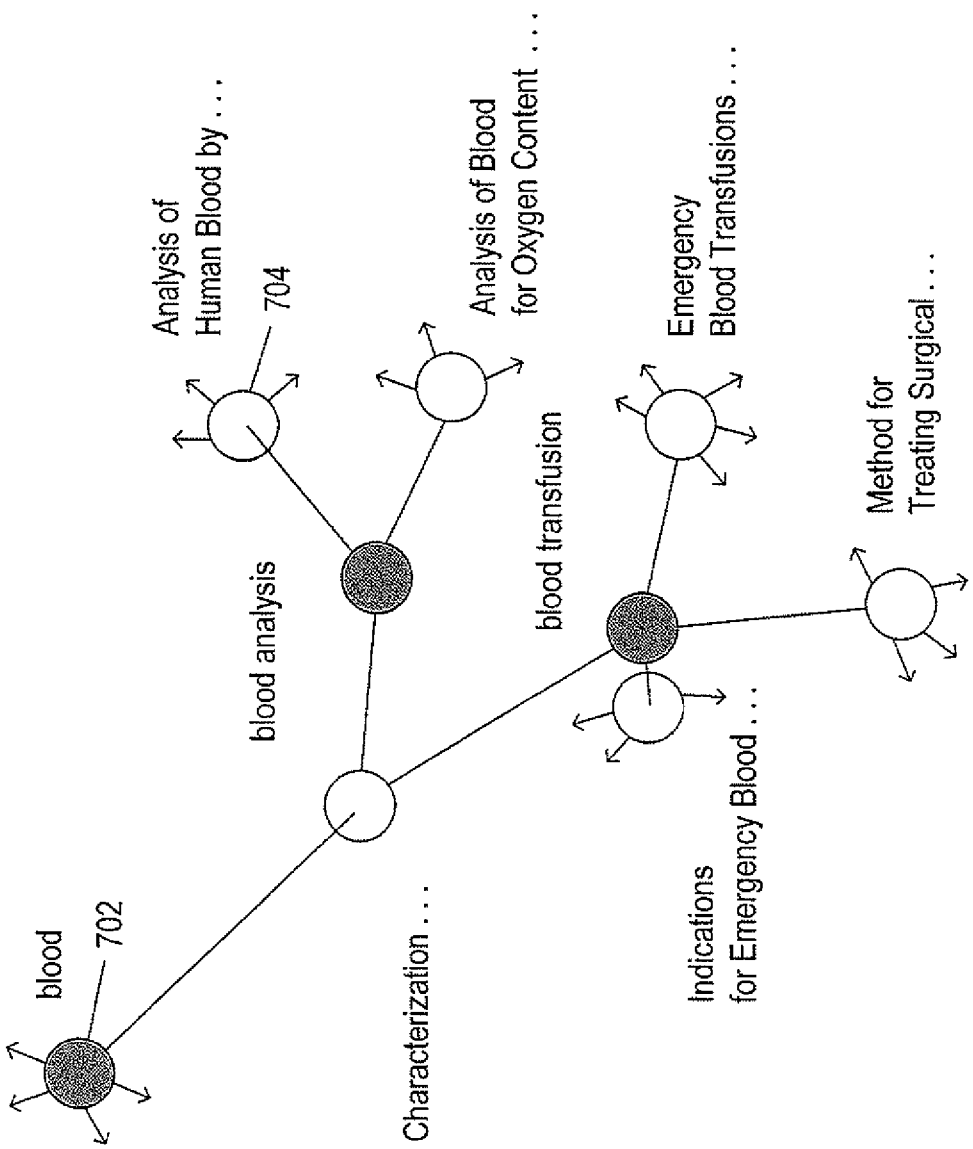

blood
702
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
704
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Indications
for Emergency Blood . . .
Emergency
Blood Transfusions . . .
Method for
Treating Surgical . . .

FIG. 7A

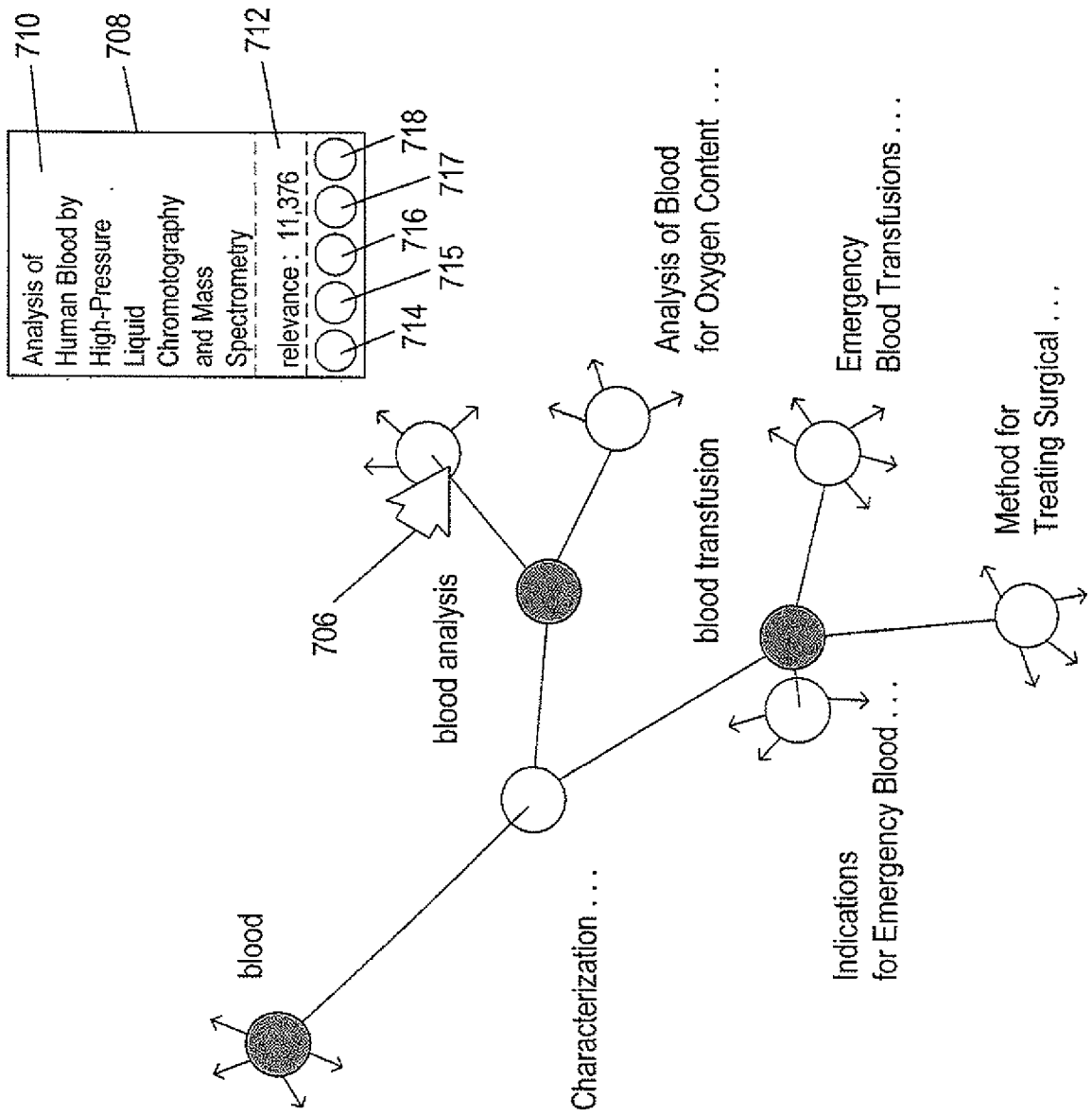

710
708
712
Analysis of Human Blood by High-Pressure Liquid Chromotography and Mass Spectrometry
relevance : 11,376
714
715
716
717
718
706
blood
blood analysis
Characterization . . .
Analysis of Blood for Oxygen Content . . .
blood transfusion
Emergency Blood Transfusions . . .
Indications for Emergency Blood . . .
Method for Treating Surgical . . .

FIG. 7B

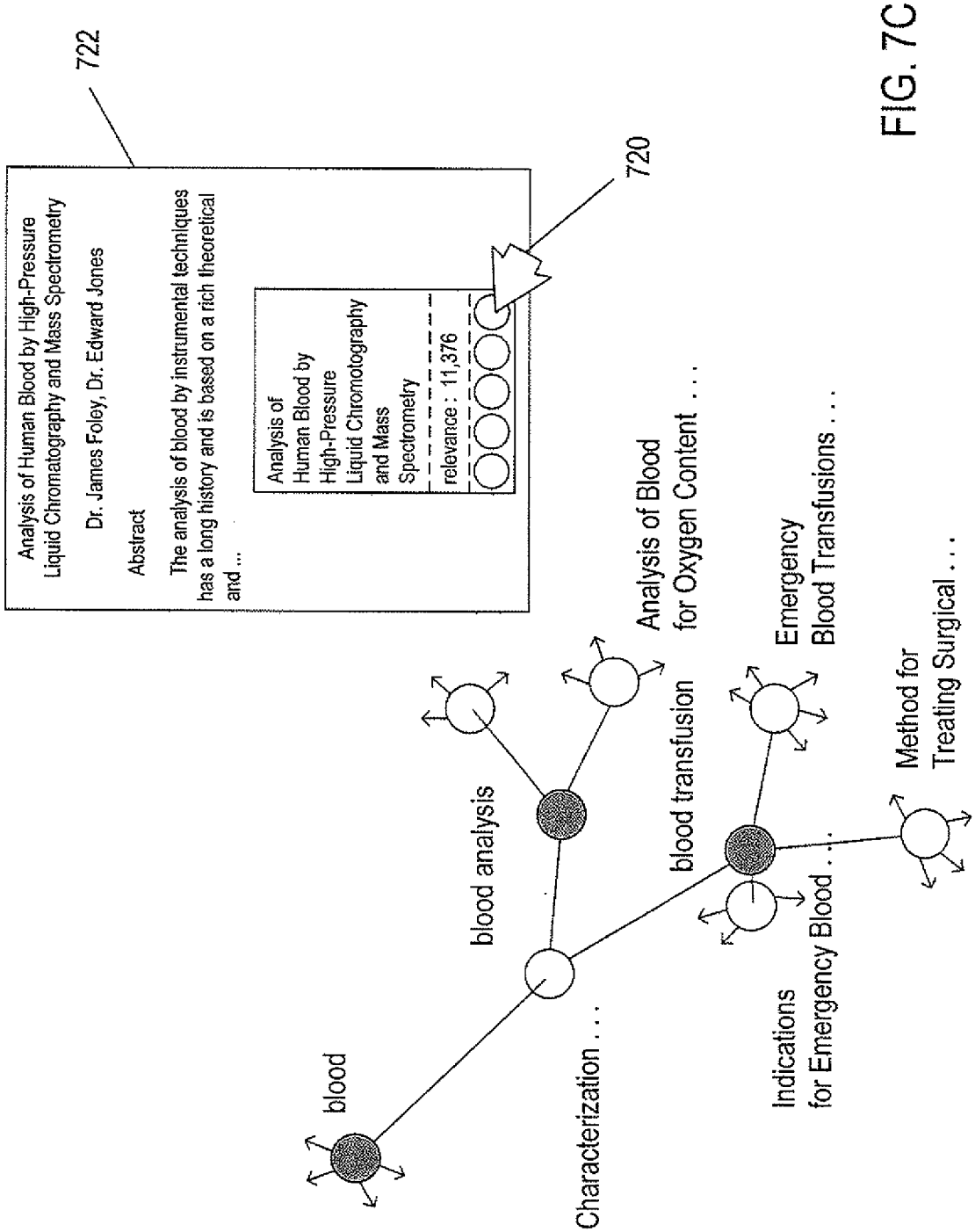

722
Analysis of Human Blood by High-Pressure Liquid Chromatography and Mass Spectrometry
Dr. James Foley, Dr. Edward Jones
Abstract
The analysis of blood by instrumental techniques has a long history and is based on a rich theoretical and ...
Analysis of Human Blood by High-Pressure Liquid Chromotography and Mass Spectrometry
relevance : 11,376
720
blood
Characterization . . .
blood analysis
Analysis of Blood for Oxygen Content . . .
blood transfusion
Indications for Emergency Blood . . .
Emergency Blood Transfusions . . .
Method for Treating Surgical . . .

FIG. 7C

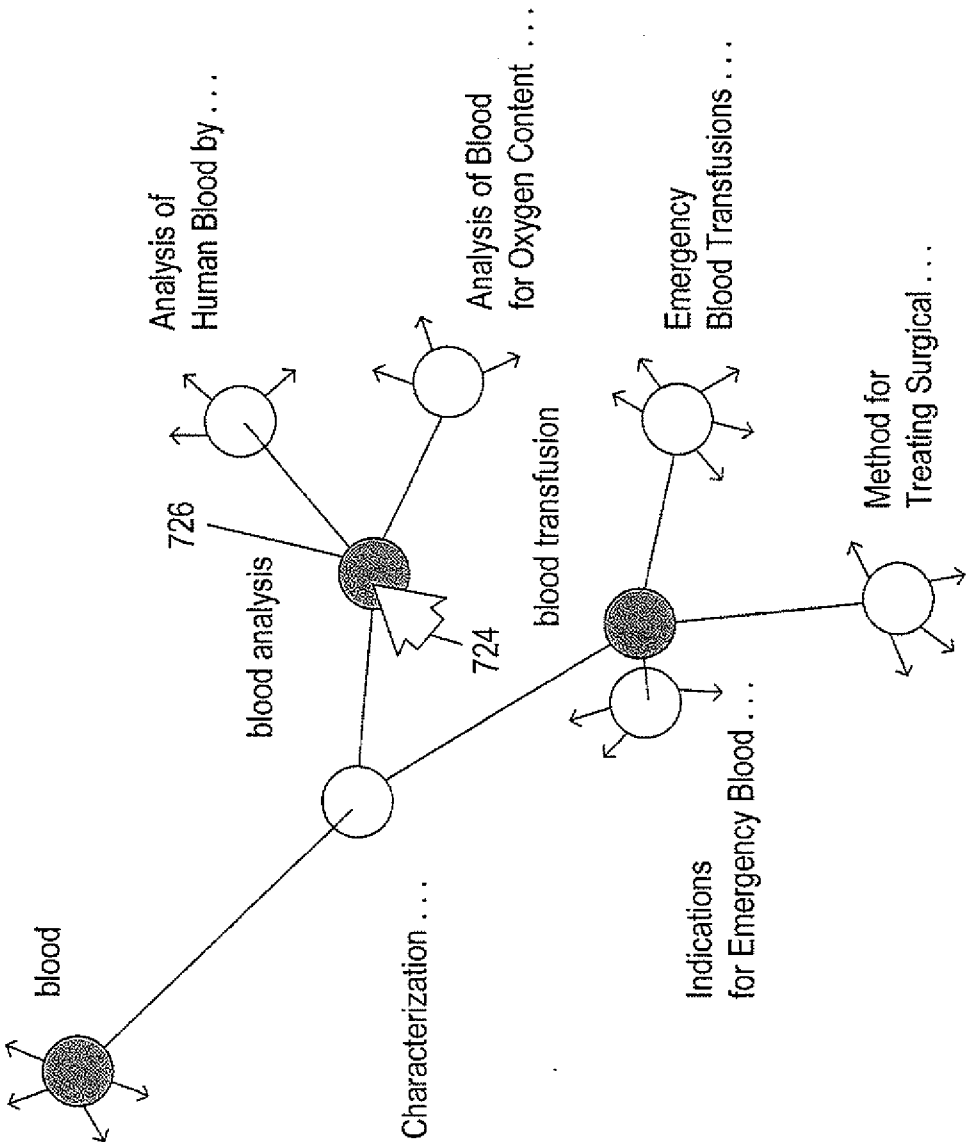

blood
Characterization . . .
blood analysis
726
Analysis of
Human Blood by . . .
724
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Indications
for Emergency Blood . . .
Emergency
Blood Transfusions . . .
Method for
Treating Surgical . . .

FIG. 7D

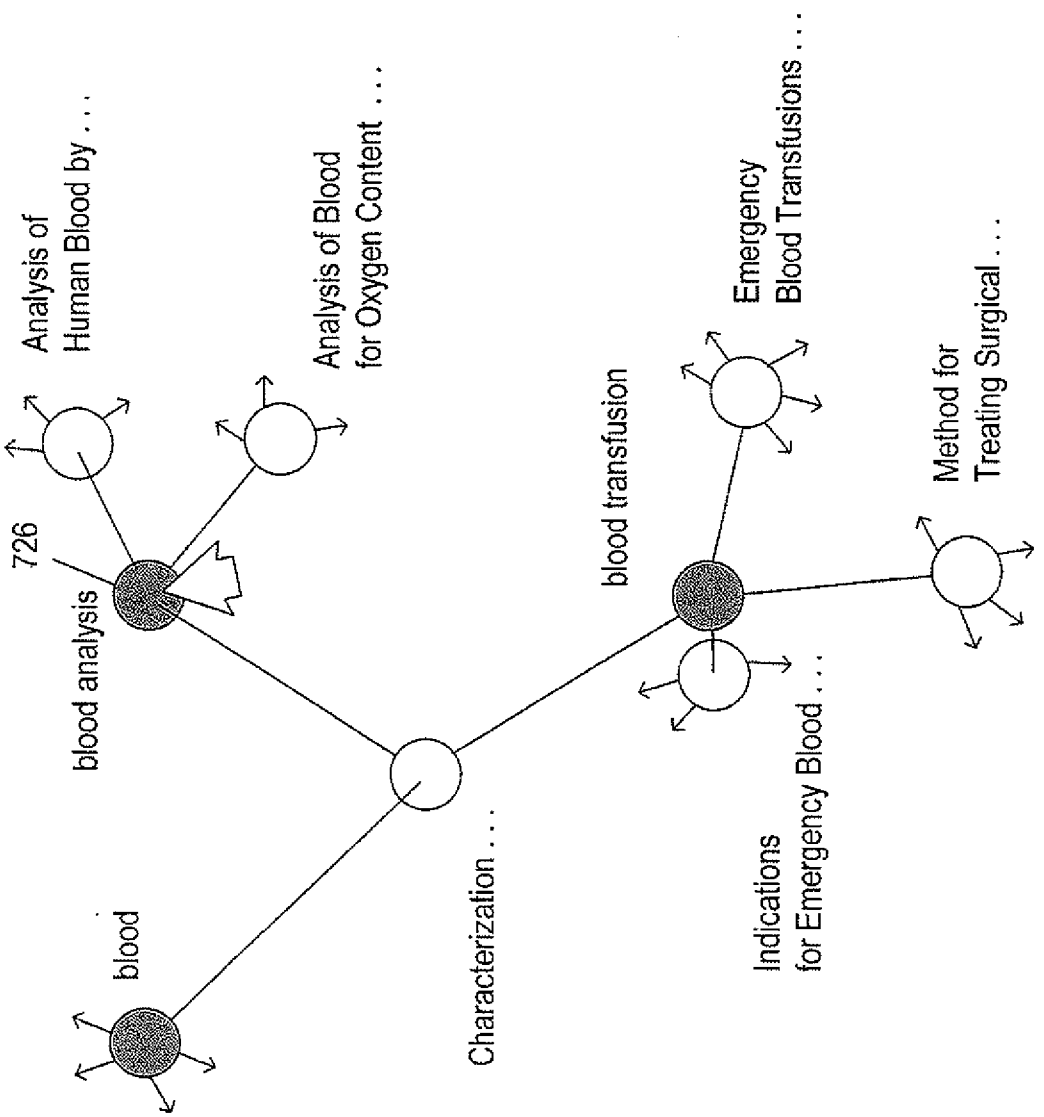

blood
blood analysis
726
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
Characterization . . .
blood transfusion
Emergency
Blood Transfusions . . .
Indications
for Emergency Blood . . .
Method for
Treating Surgical . . .

FIG. 7E

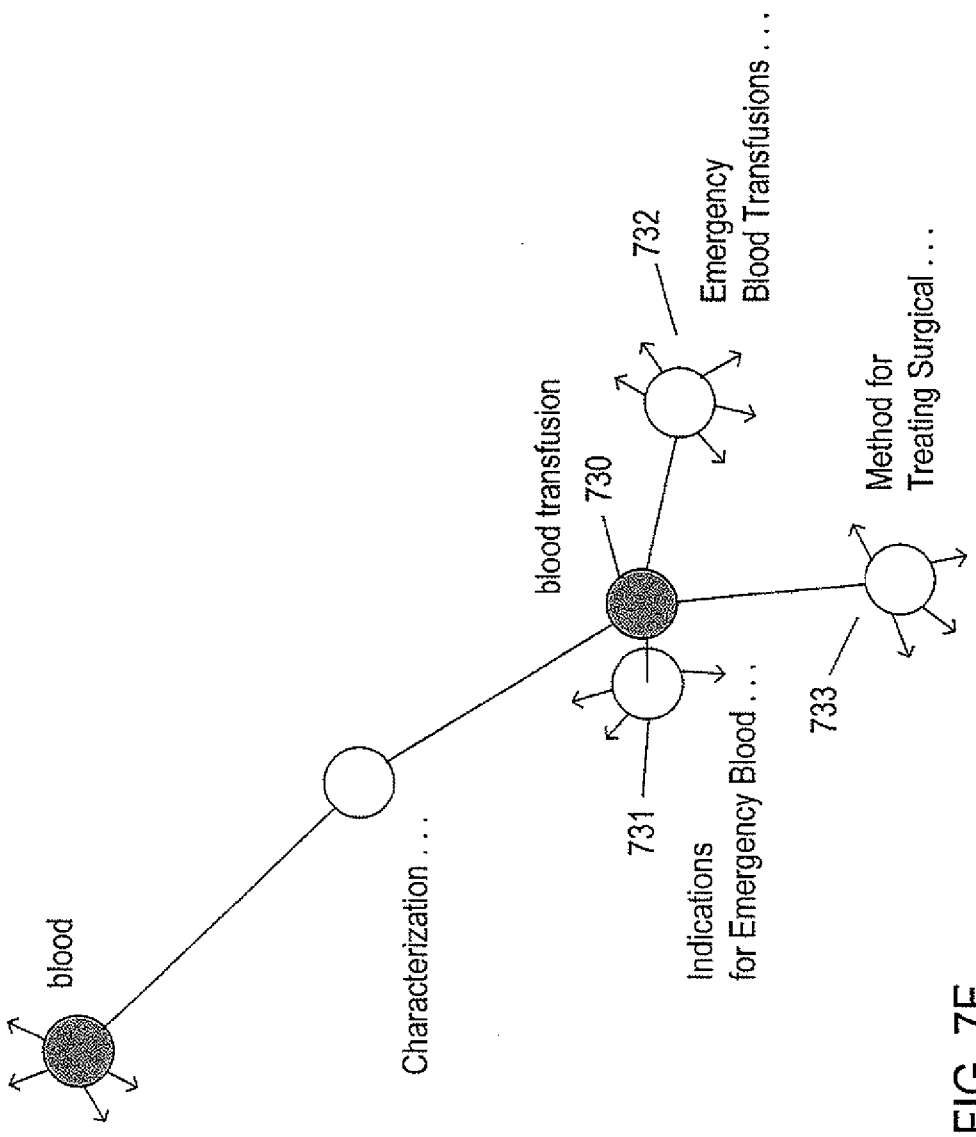

blood
Characterization . . .
blood transfusion
730
731
Indications
for Emergency Blood . . .
732
Emergency
Blood Transfusions . . .
733
Method for
Treating Surgical . . .

FIG. 7F

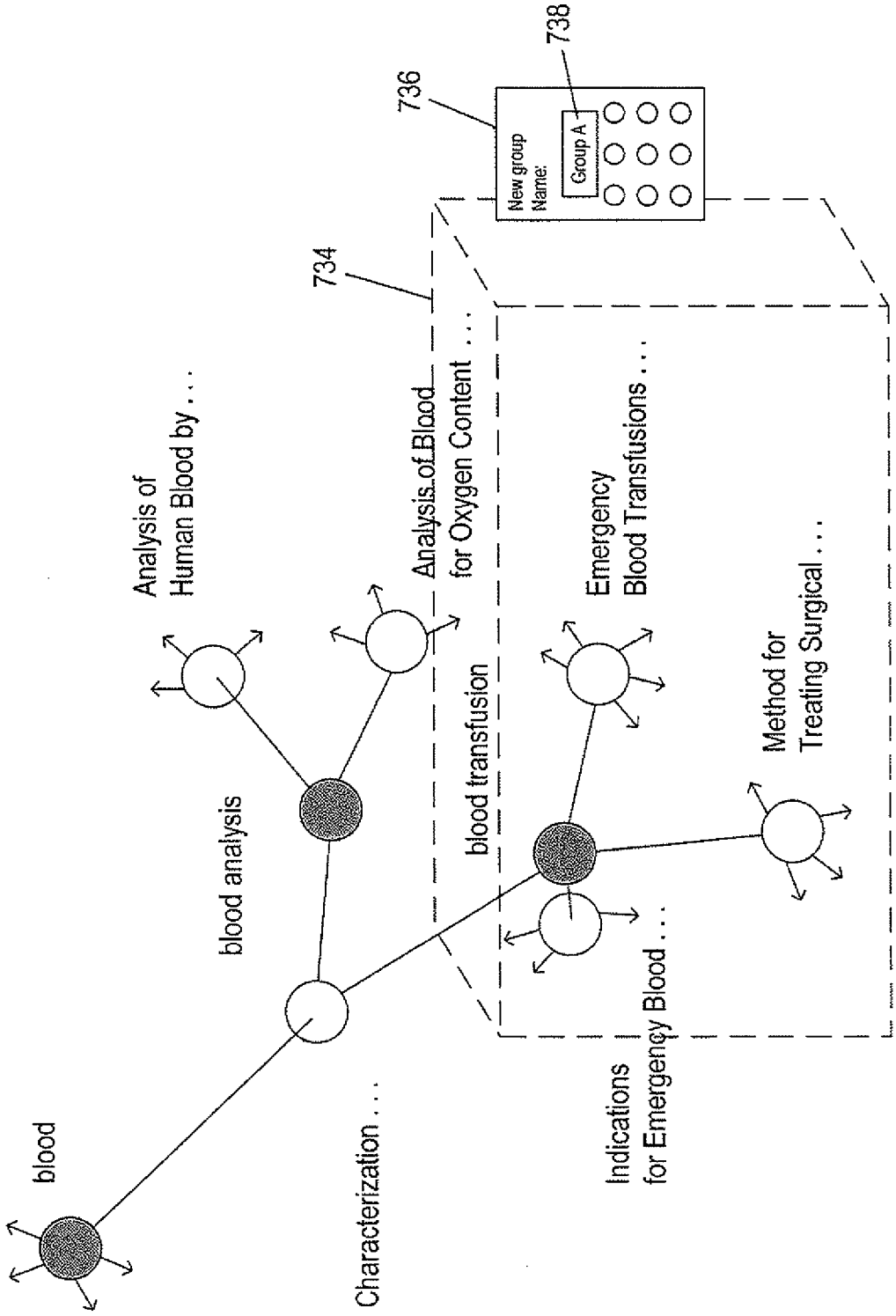

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Emergency
Blood Transfusions . . .
Method for
Treating Surgical . . .
Indications
for Emergency Blood . . .
734
736
738
New group
Name:
Group A

FIG. 7G

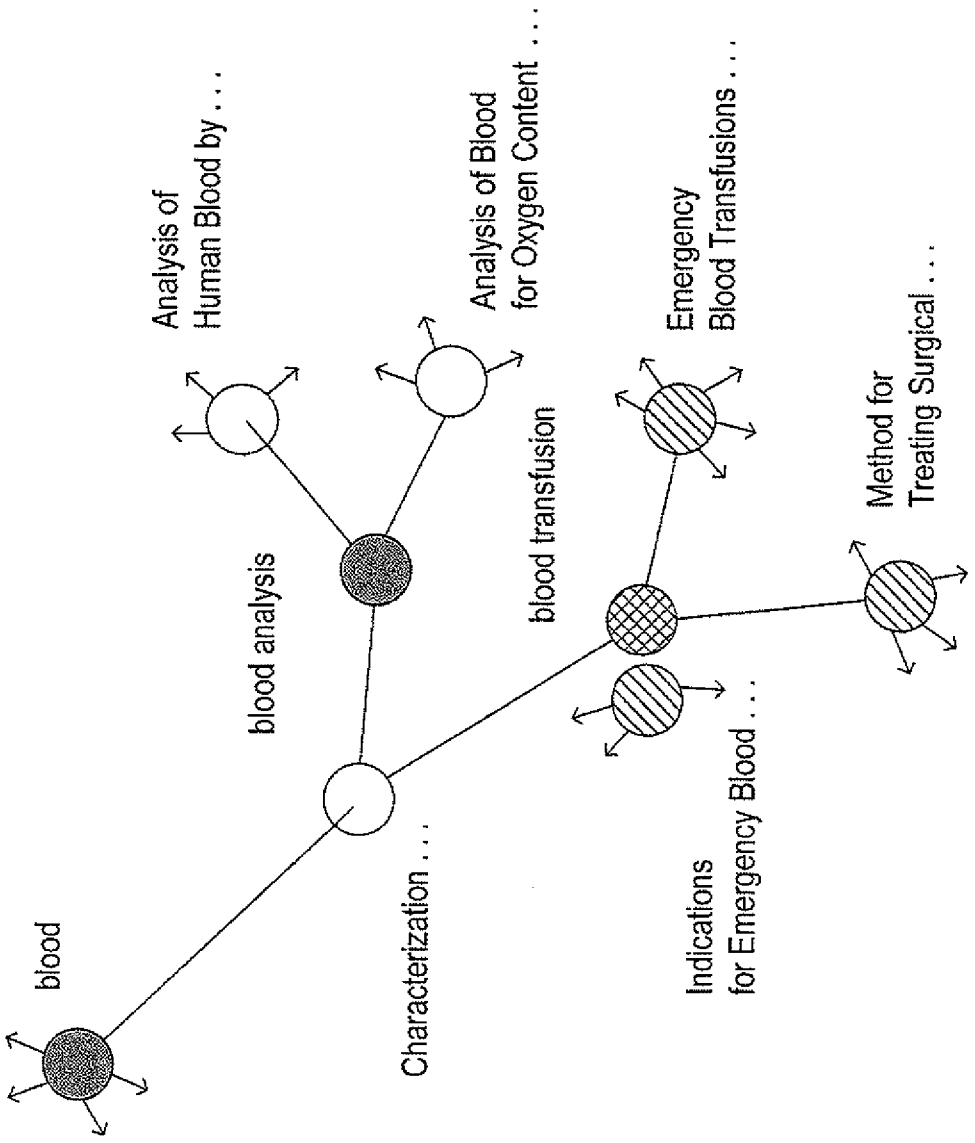

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Emergency
Blood Transfusions . . .
Indications
for Emergency Blood . . .
Method for
Treating Surgical . . .

FIG. 7H

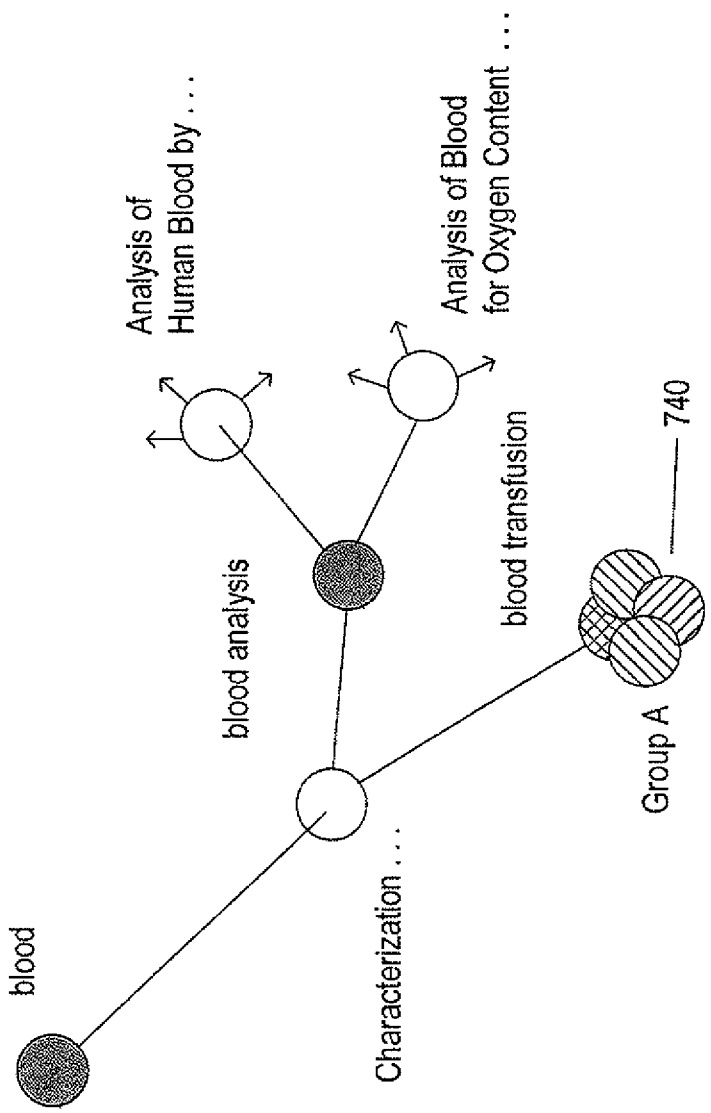

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Group A
740

FIG. 7I

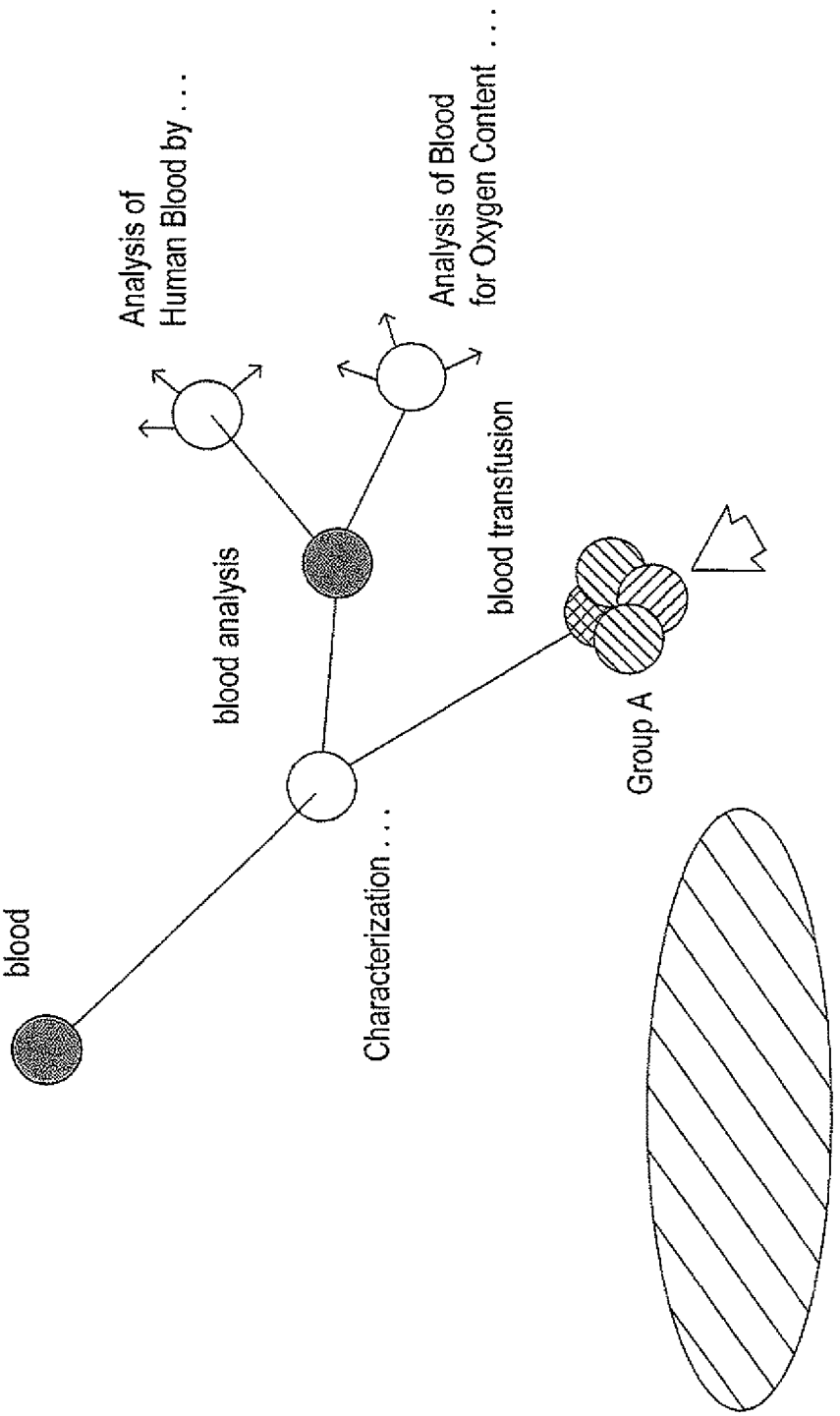

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Group A

FIG. 7J

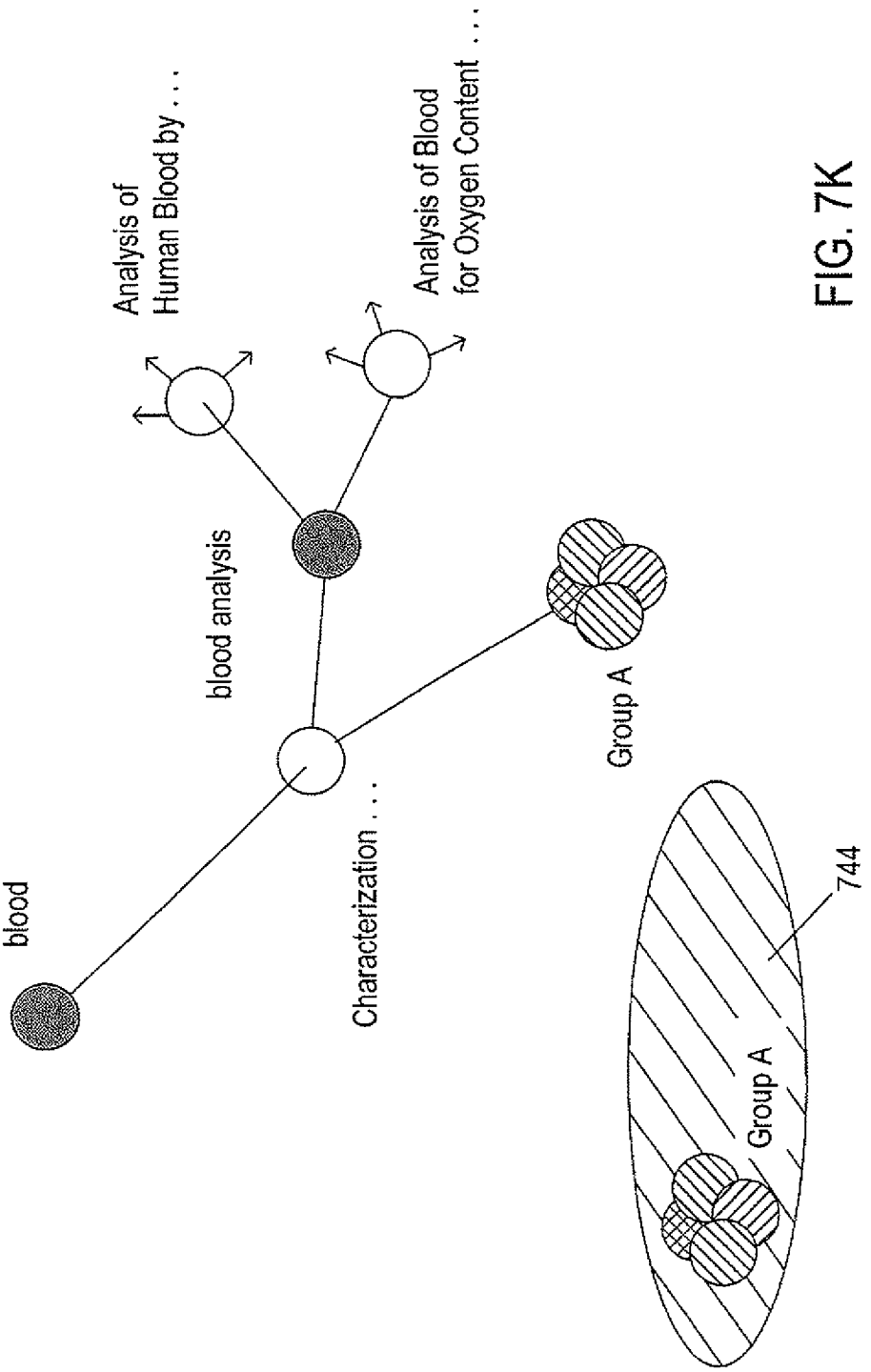

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
Group A
744
Group A

FIG. 7K

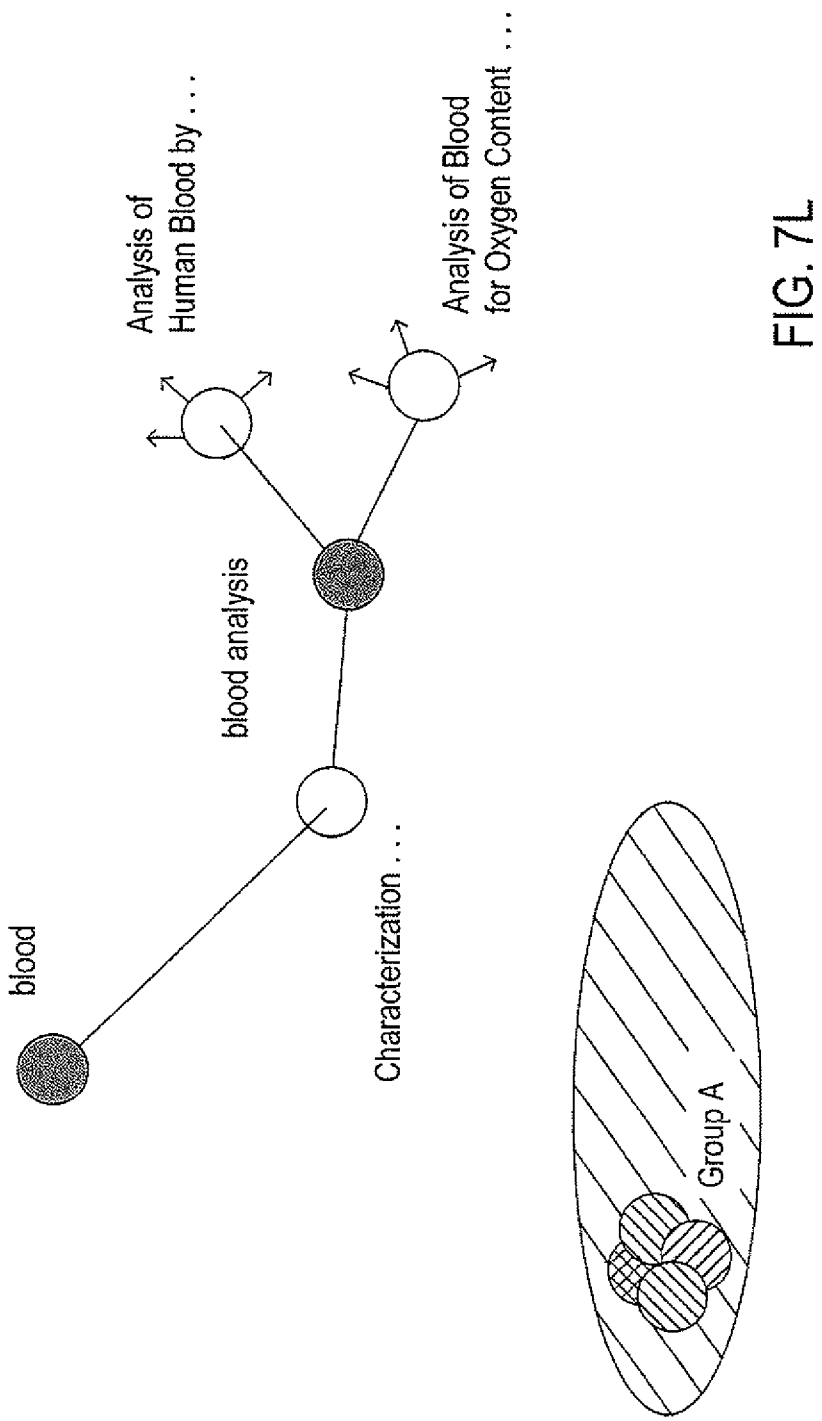

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
Group A

FIG. 7L

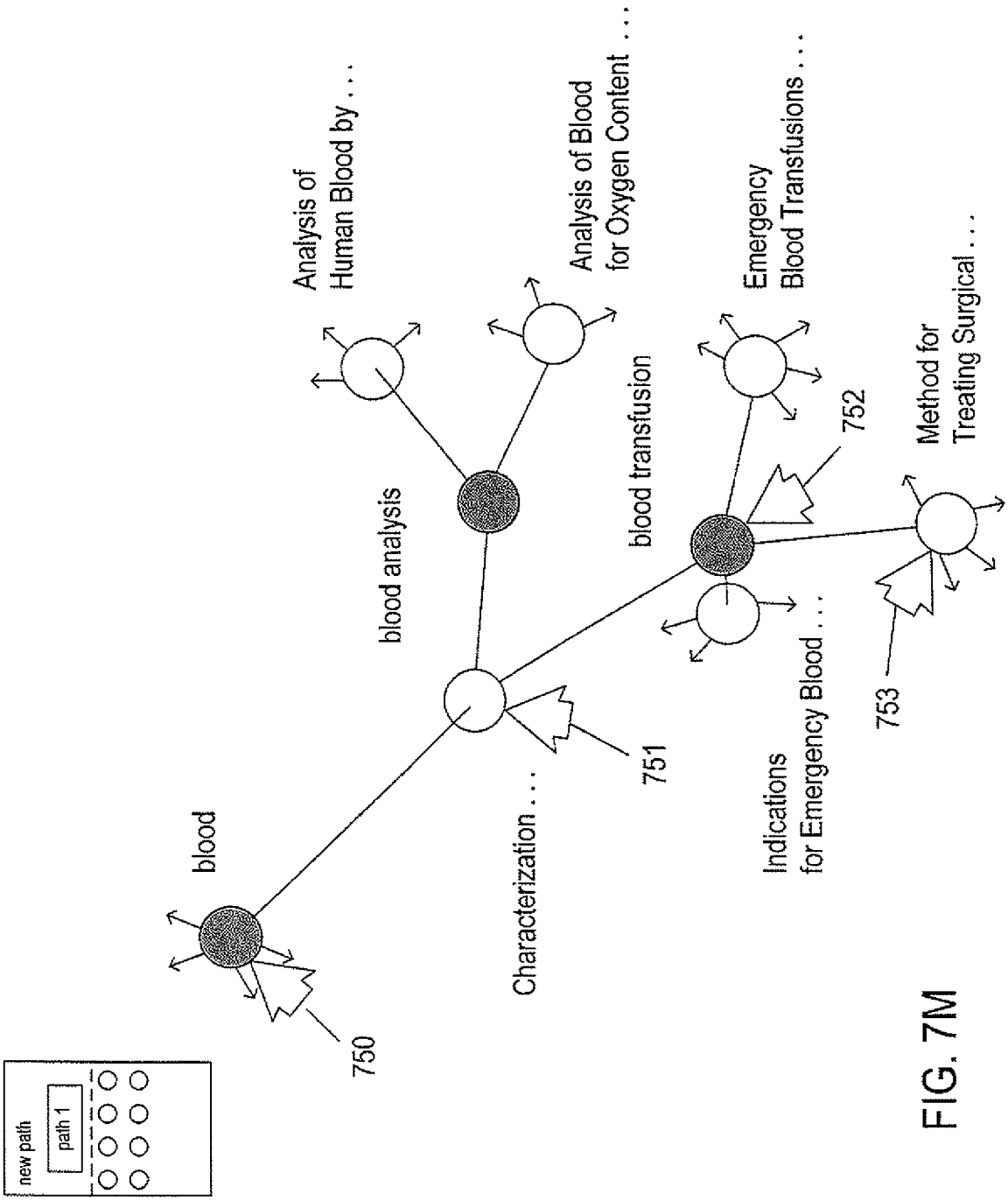

new path
path 1
blood
750
blood analysis
Characterization . . .
751
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Indications
for Emergency Blood . . .
752
Emergency
Blood Transfusions . . .
753
Method for
Treating Surgical . . .

FIG. 7M

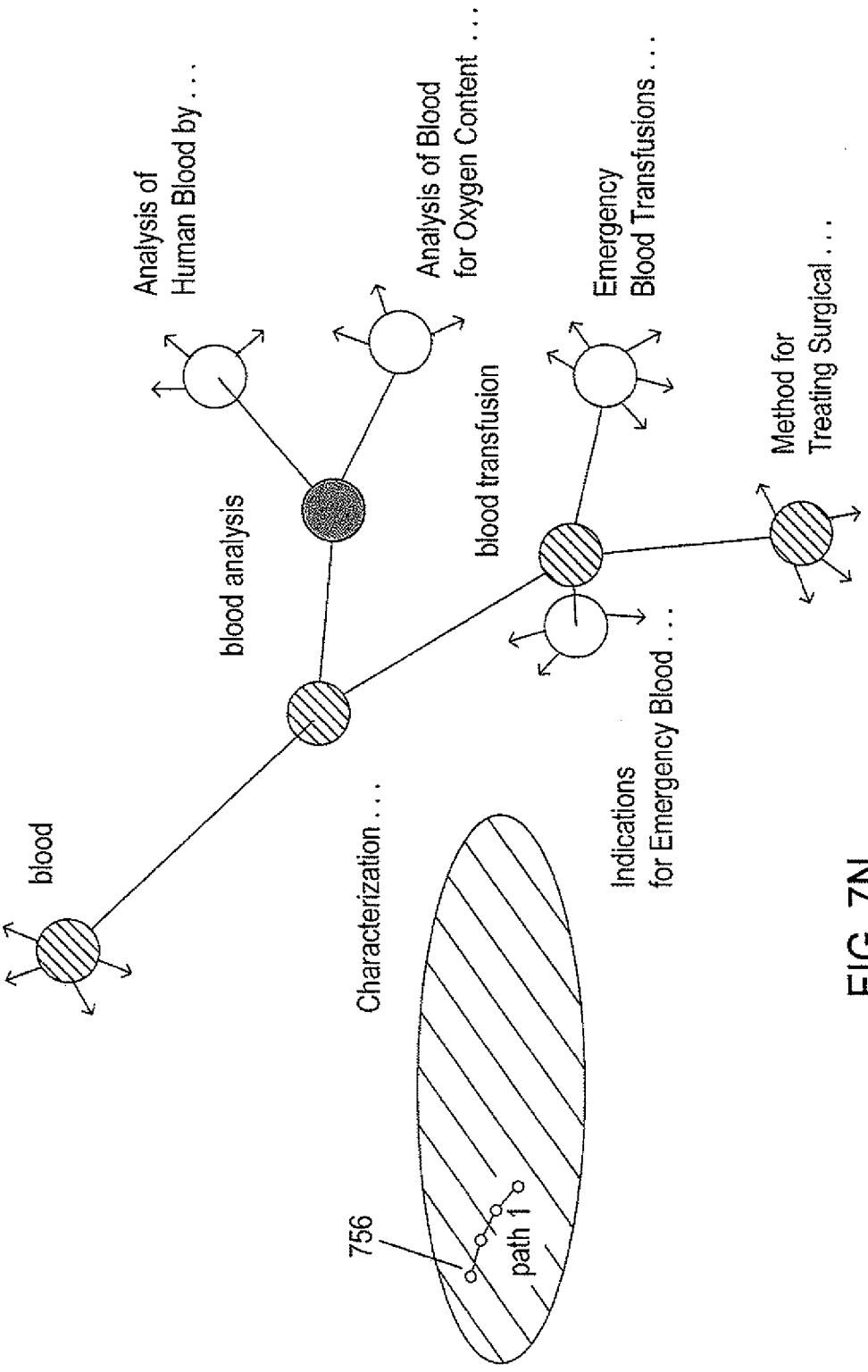

blood
Characterization . . .
blood analysis
Analysis of
Human Blood by . . .
Analysis of Blood
for Oxygen Content . . .
blood transfusion
Indications
for Emergency Blood . . .
Emergency
Blood Transfusions . . .
Method for
Treating Surgical . . .
756
path 1

FIG. 7N

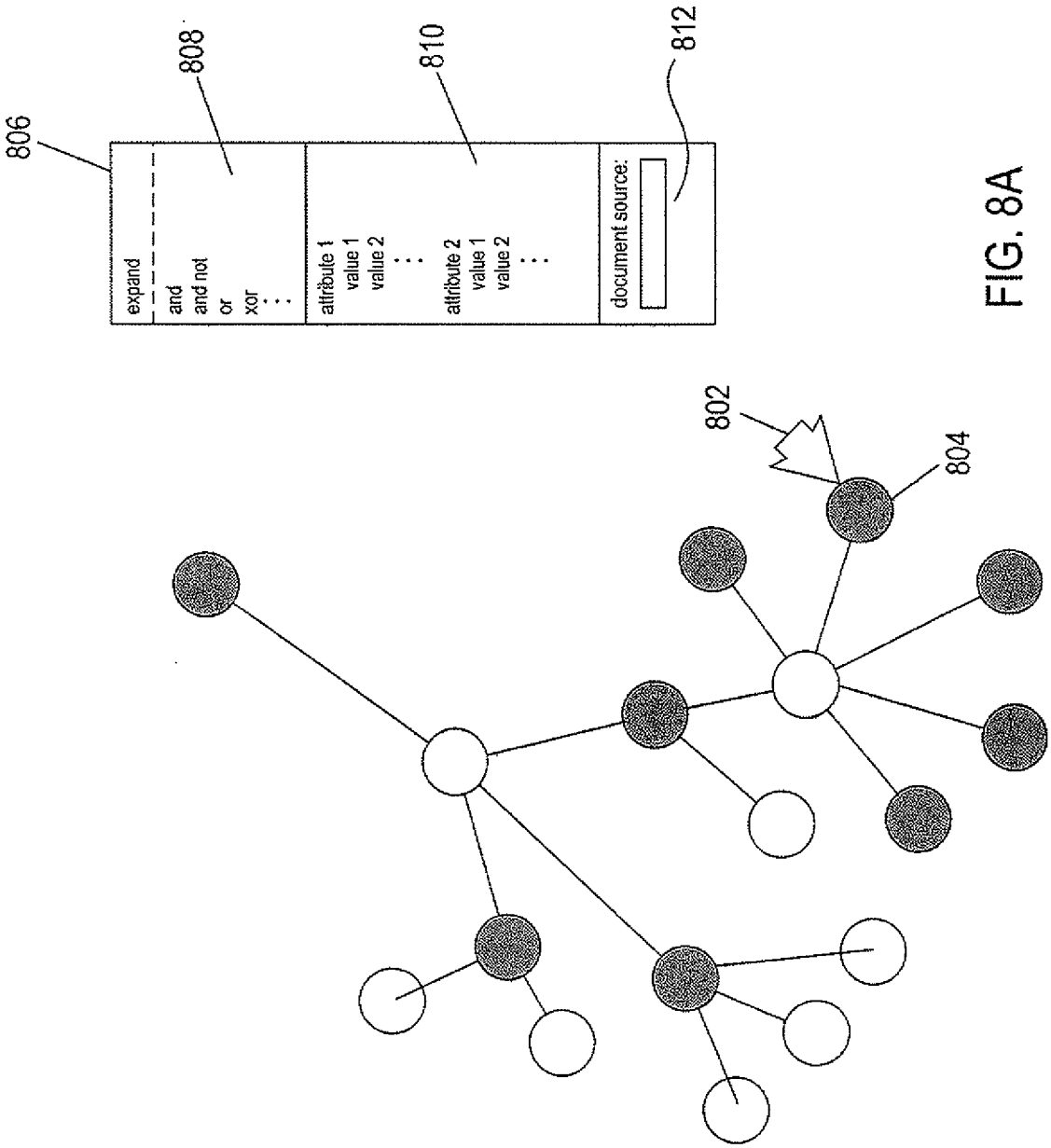

806
808
810
812
expand
and
and not
or
xor
attribute 1
value 1
value 2
attribute 2
value 1
value 2
document source:
802
804

FIG. 8A

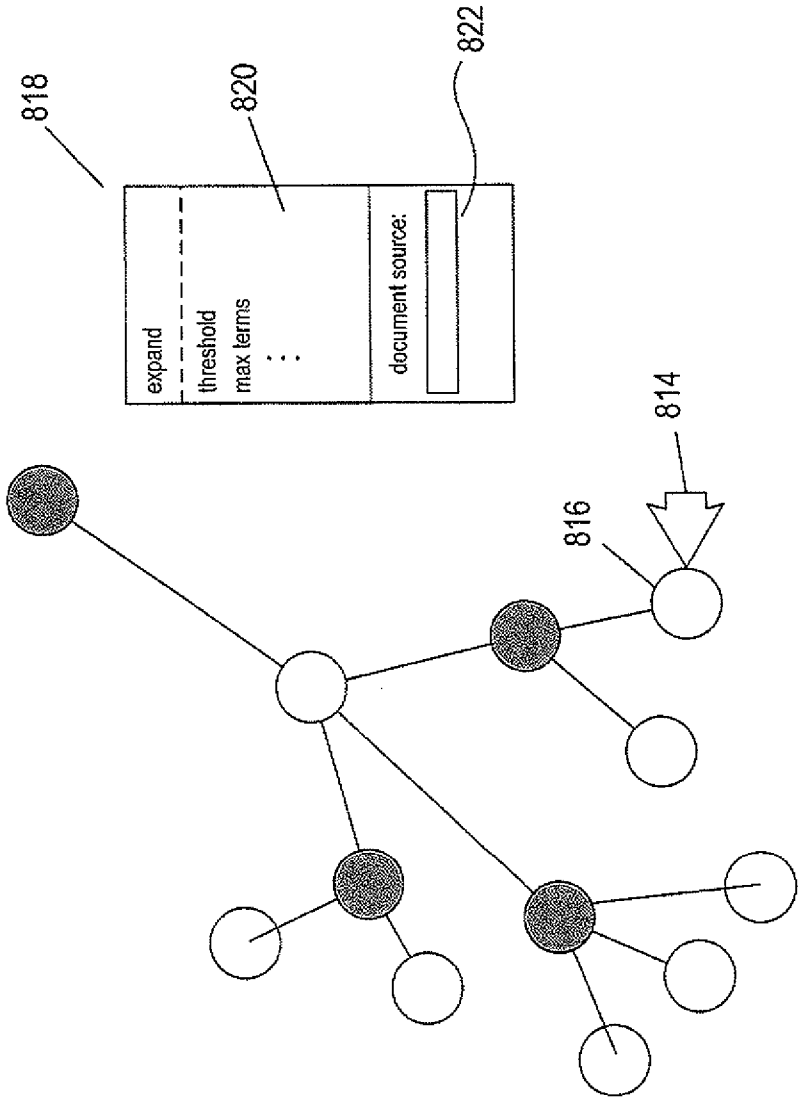

818
820
822
expand
threshold
max terms
document source:
814
816

FIG. 8B

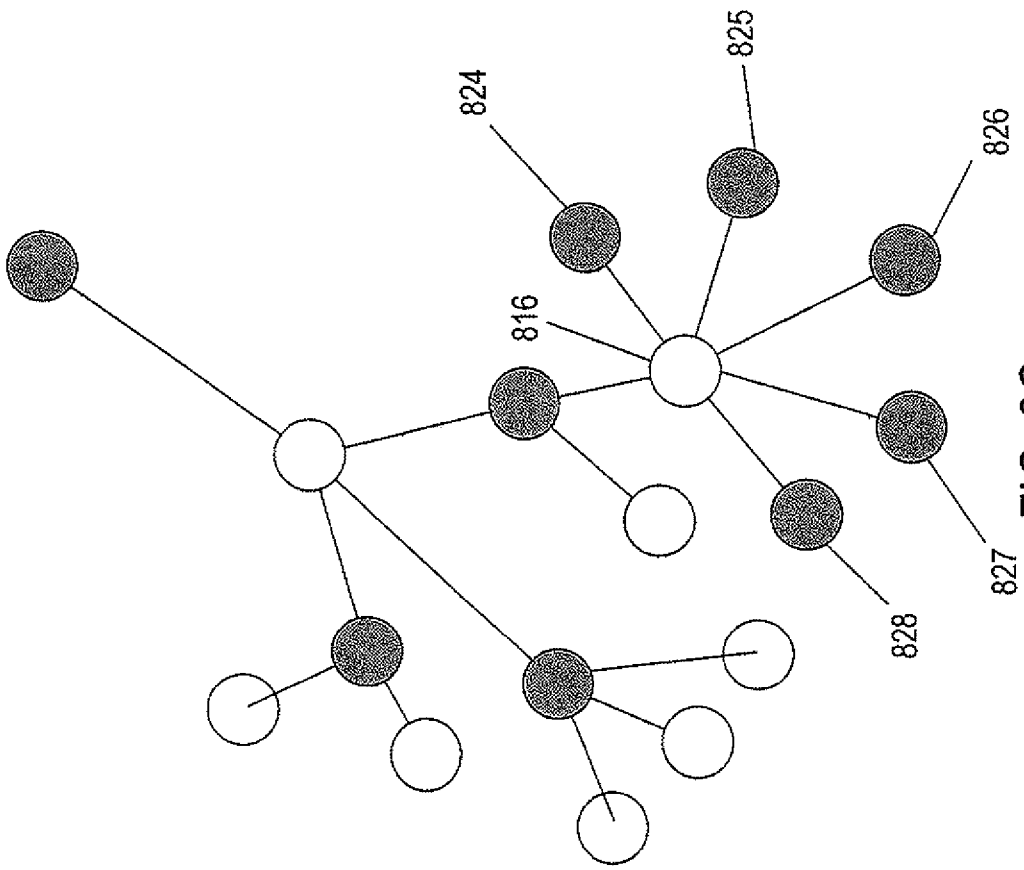


FIG. 8C

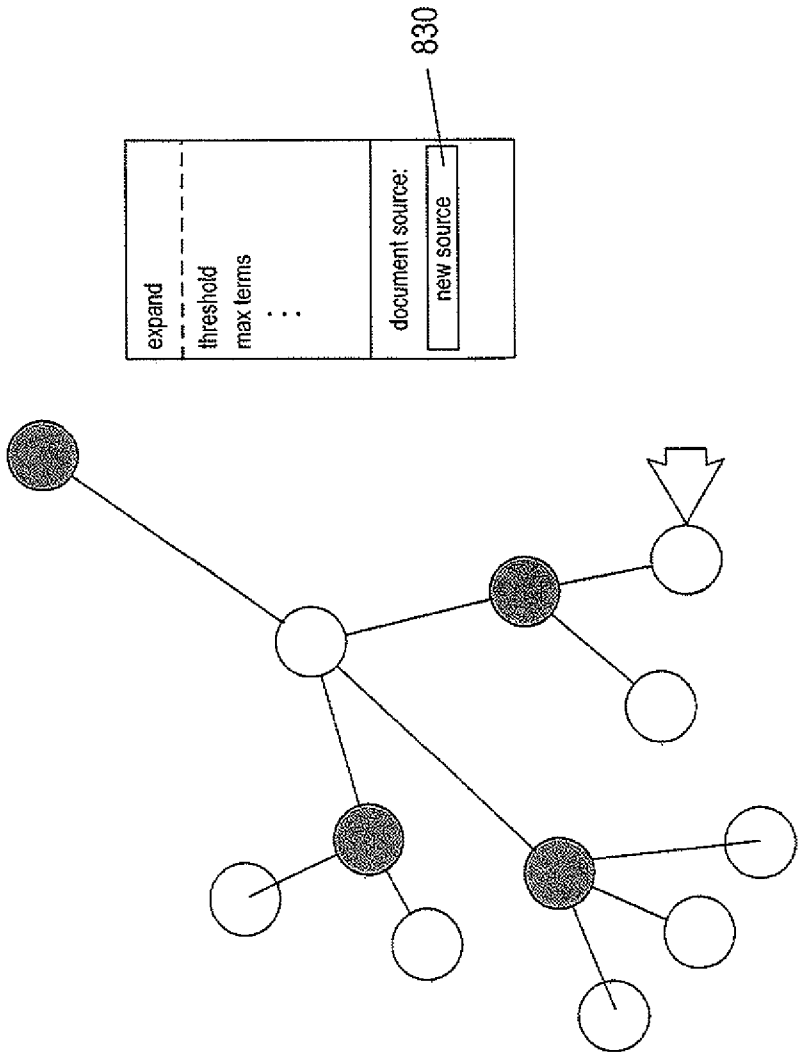


FIG. 8D

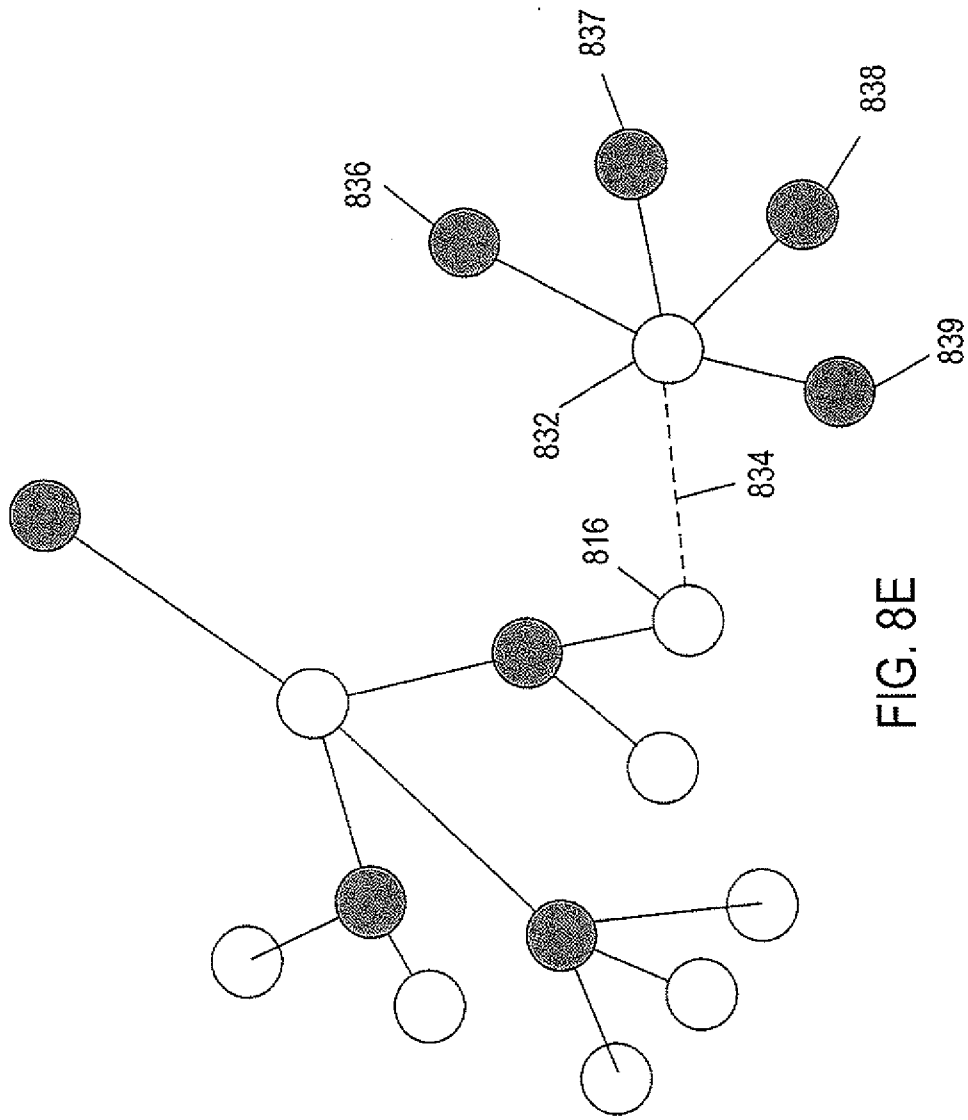


FIG. 8E

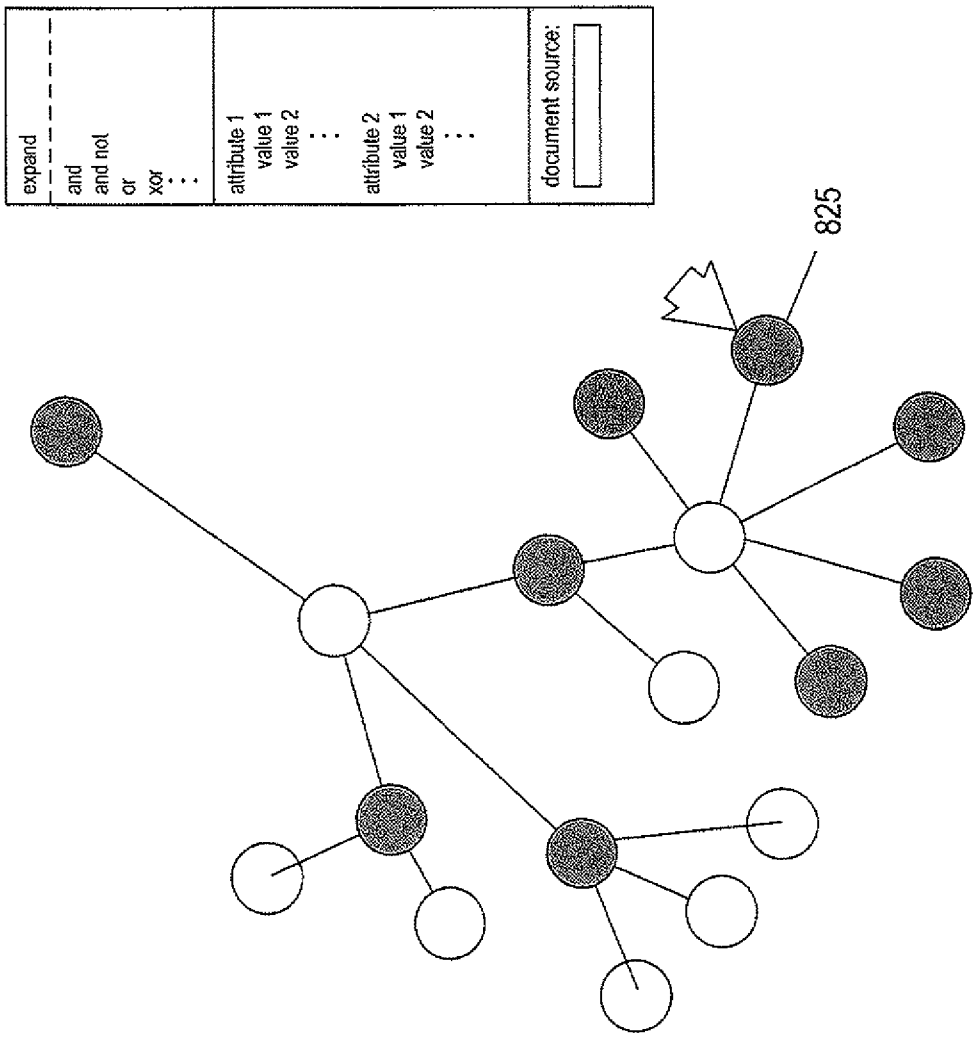

expand
and
and not
or
xor
attribute 1
value 1
value 2
attribute 2
value 1
value 2
document source:
825

FIG. 8F

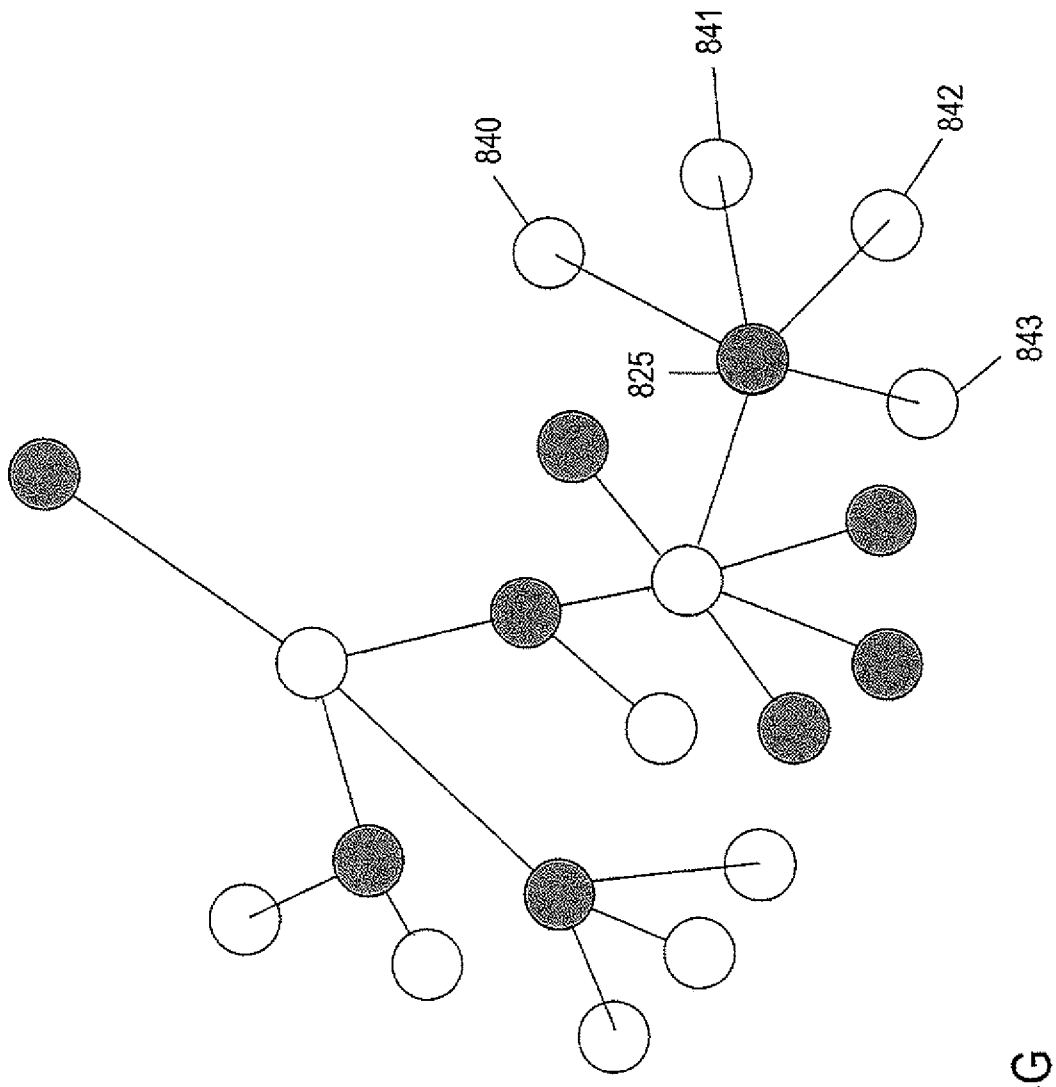


FIG. 8G

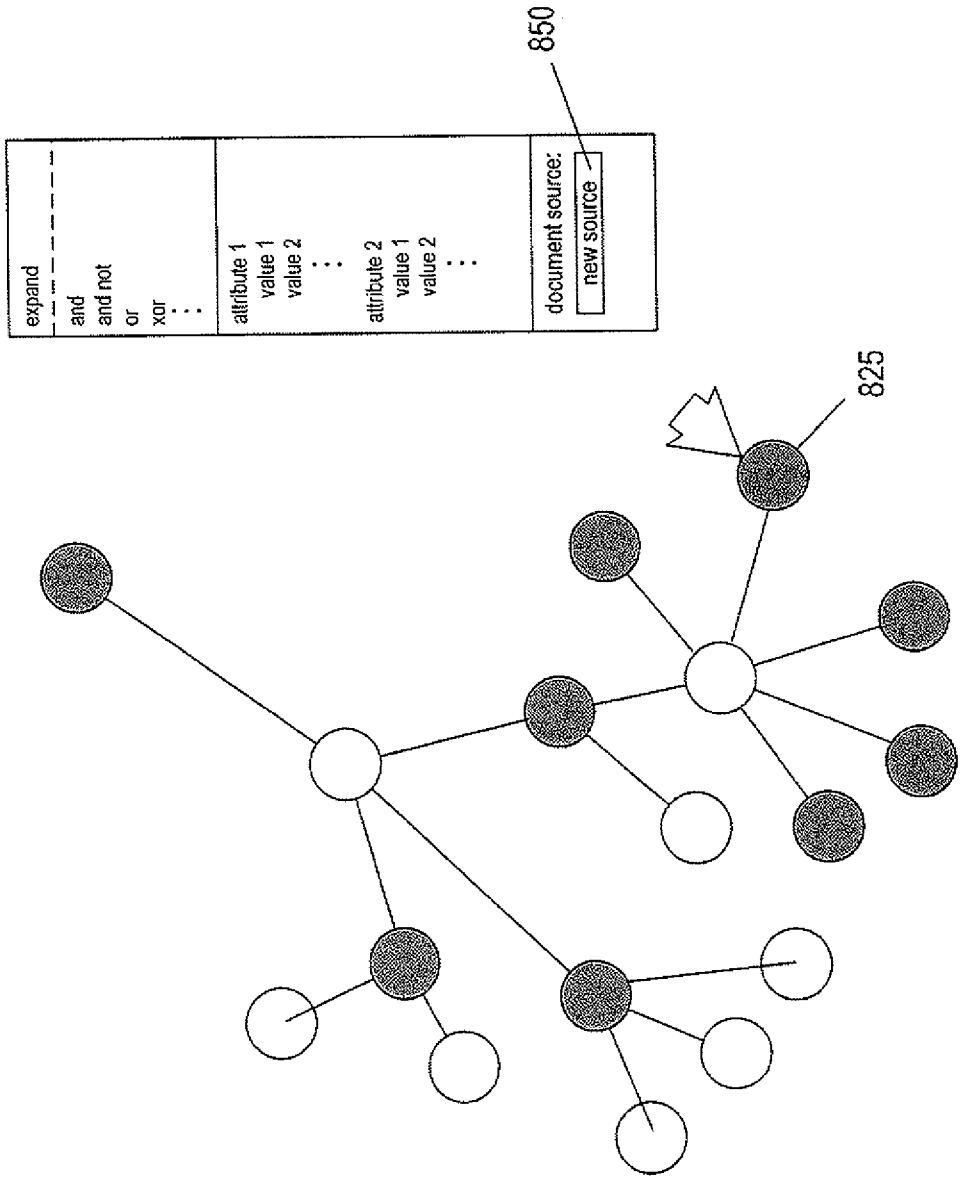

850
expand
and
and not
or
xor
attribute 1
value 1
value 2
attribute 2
value 1
value 2
document source:
new source
825

FIG. 8H

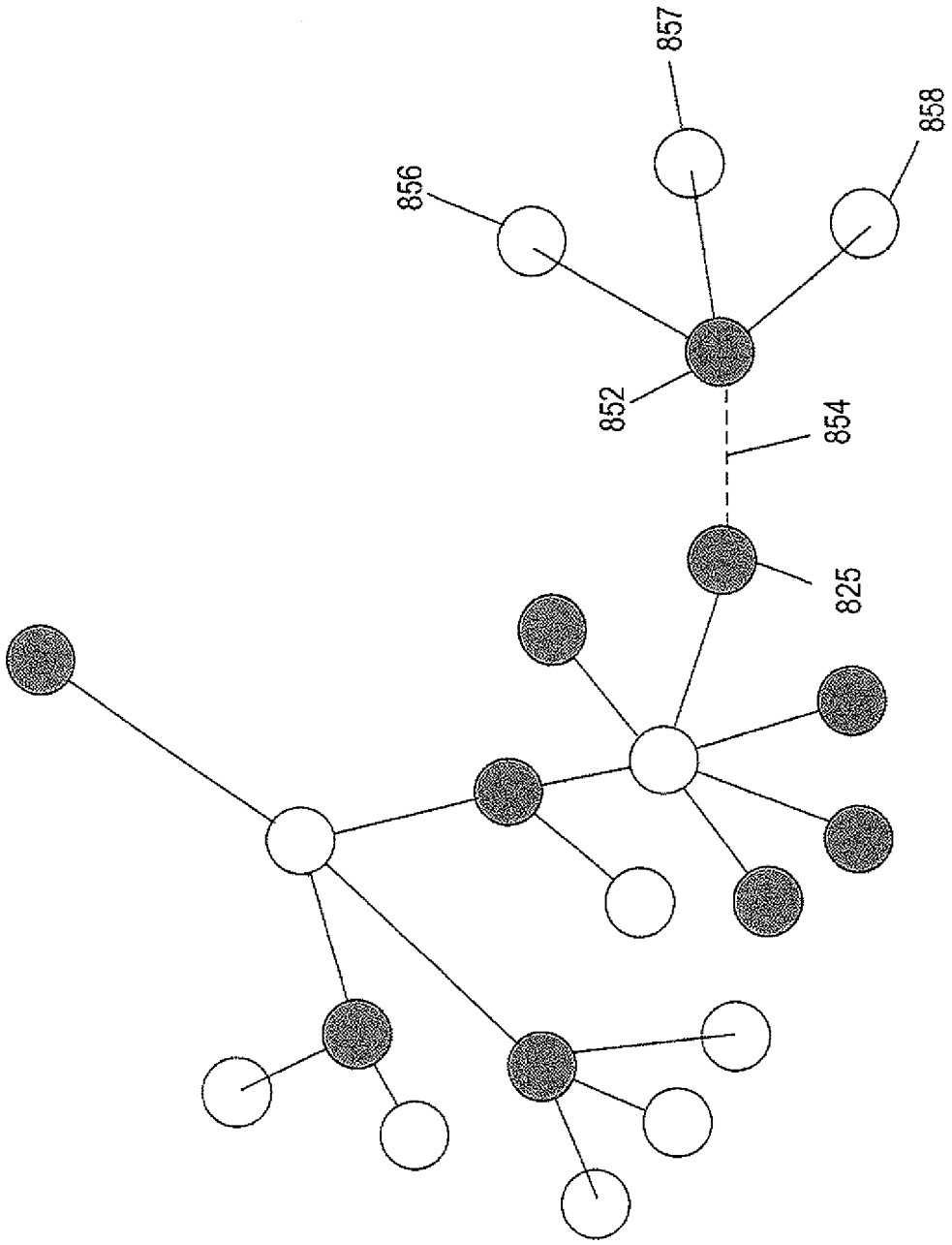


FIG. 8I

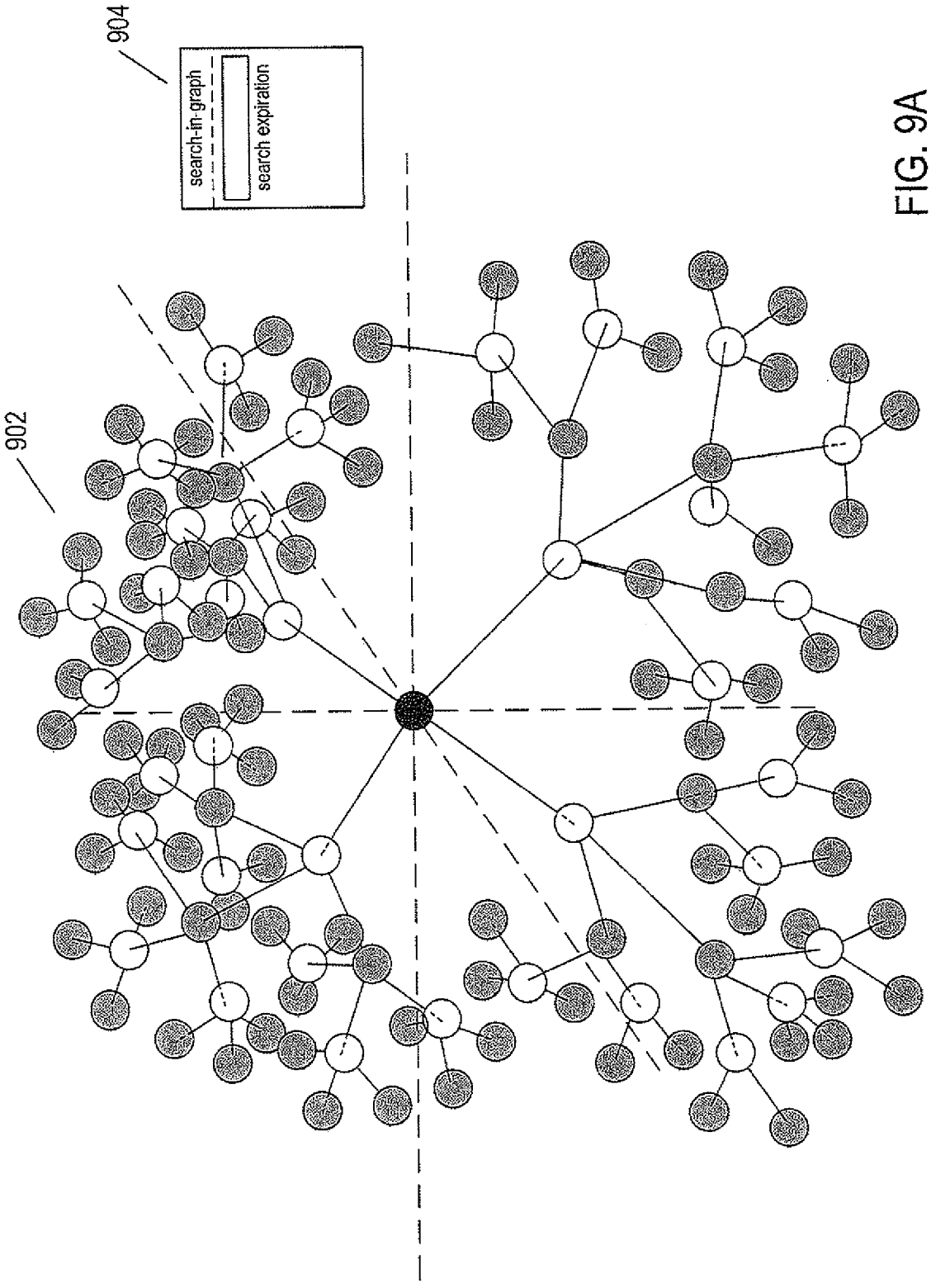


FIG. 9A

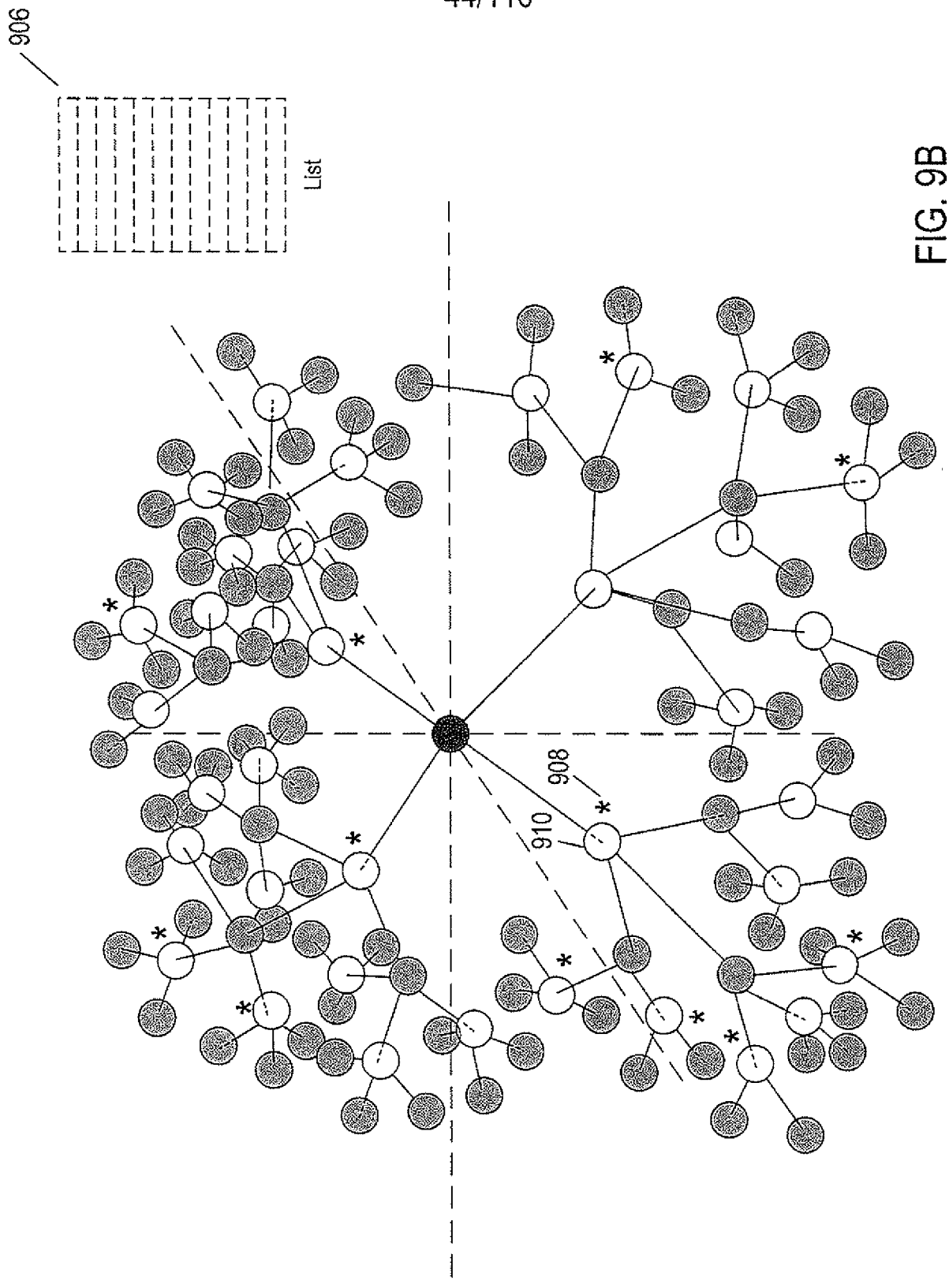


FIG. 9B

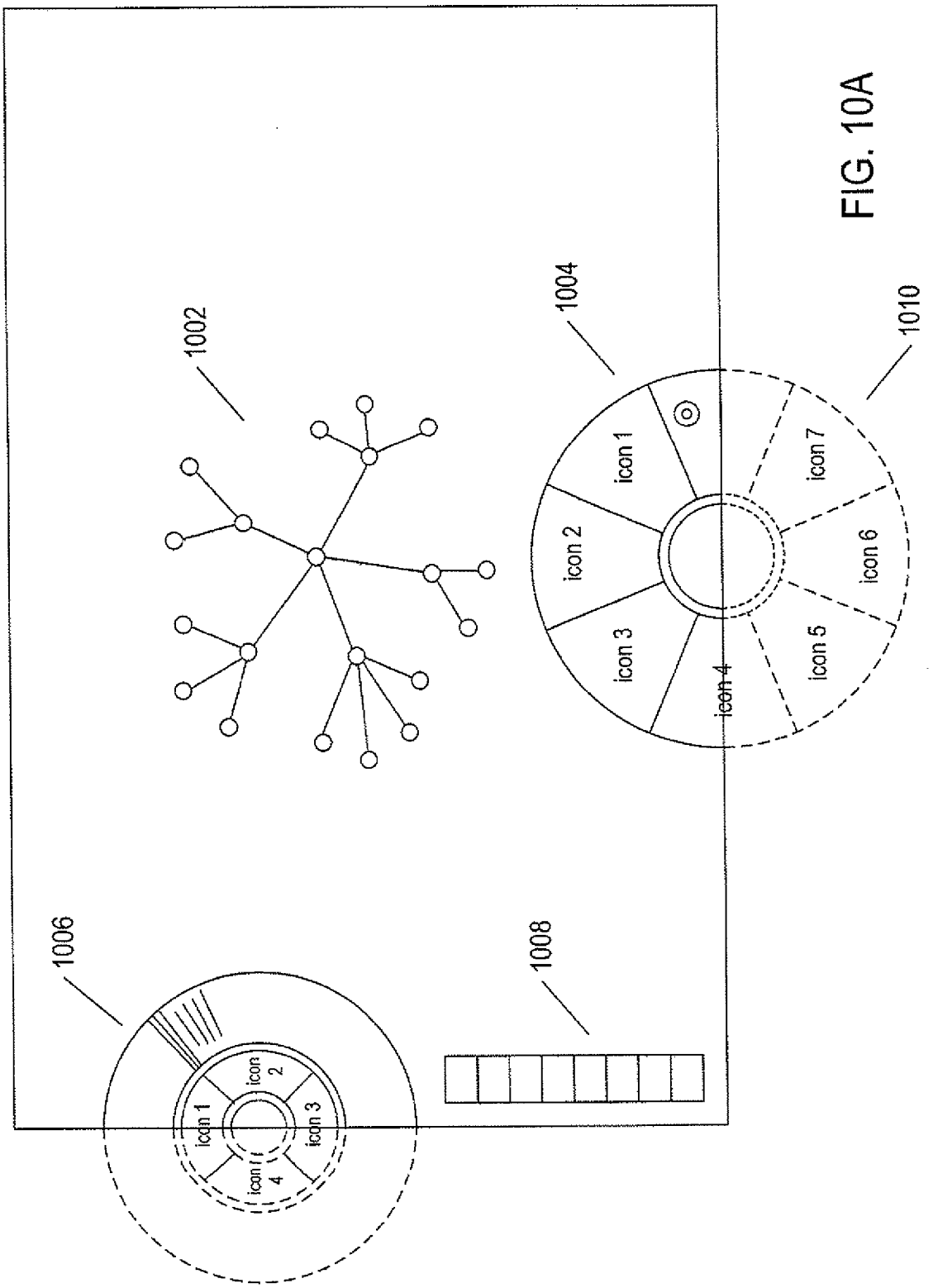

1002
1004
1006
1008
1010
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7

FIG. 10A

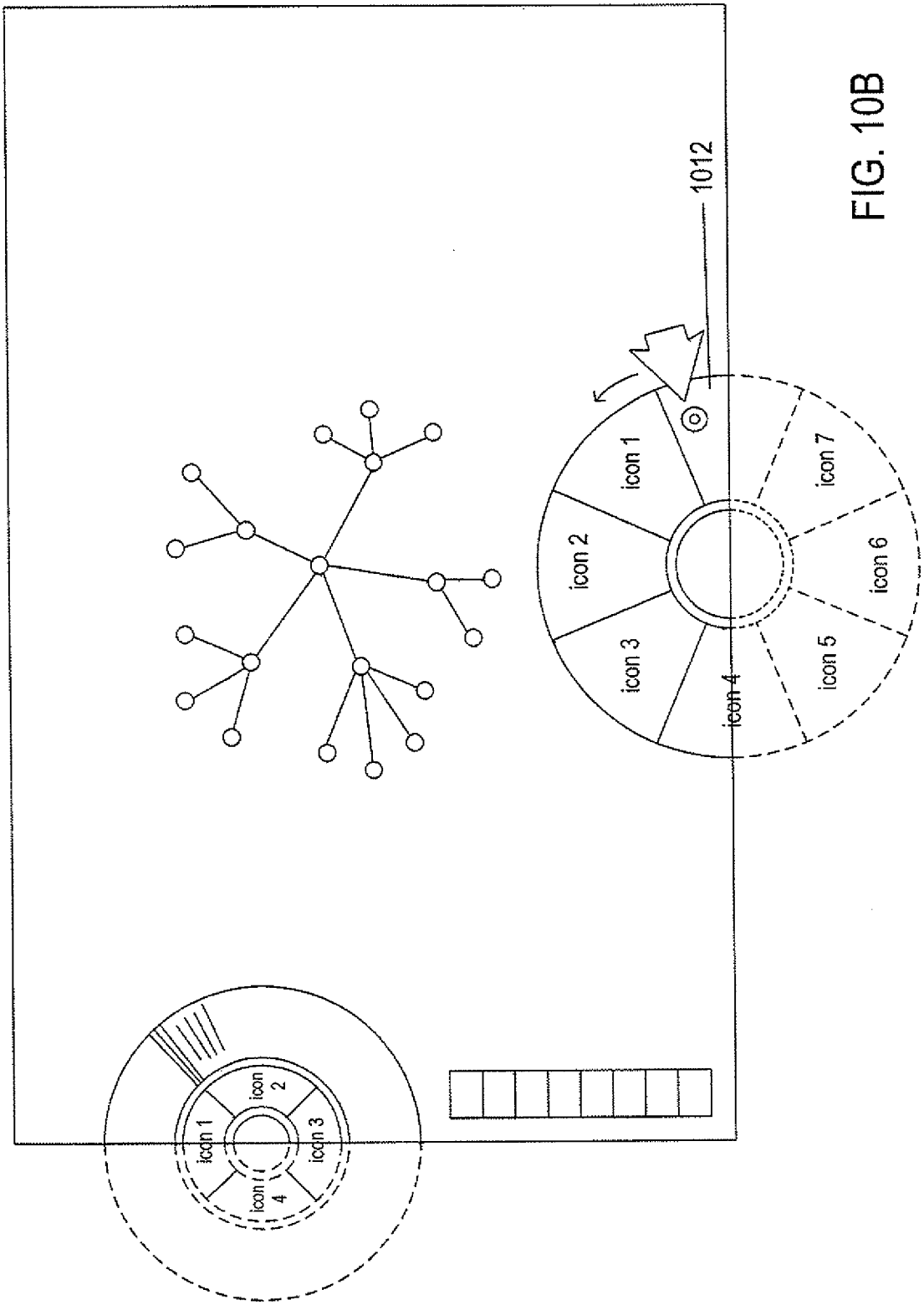

1012
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10B

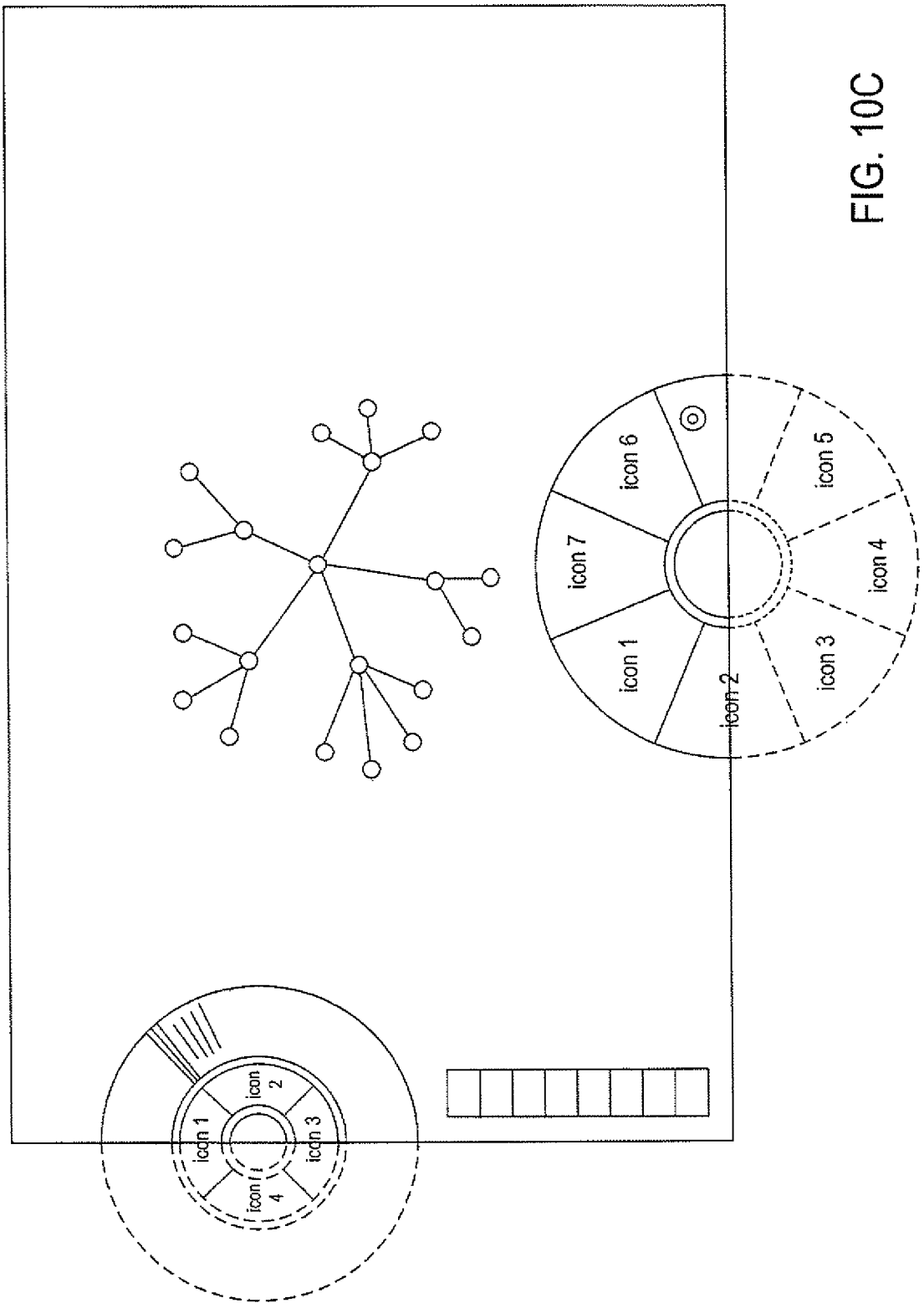

icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10C

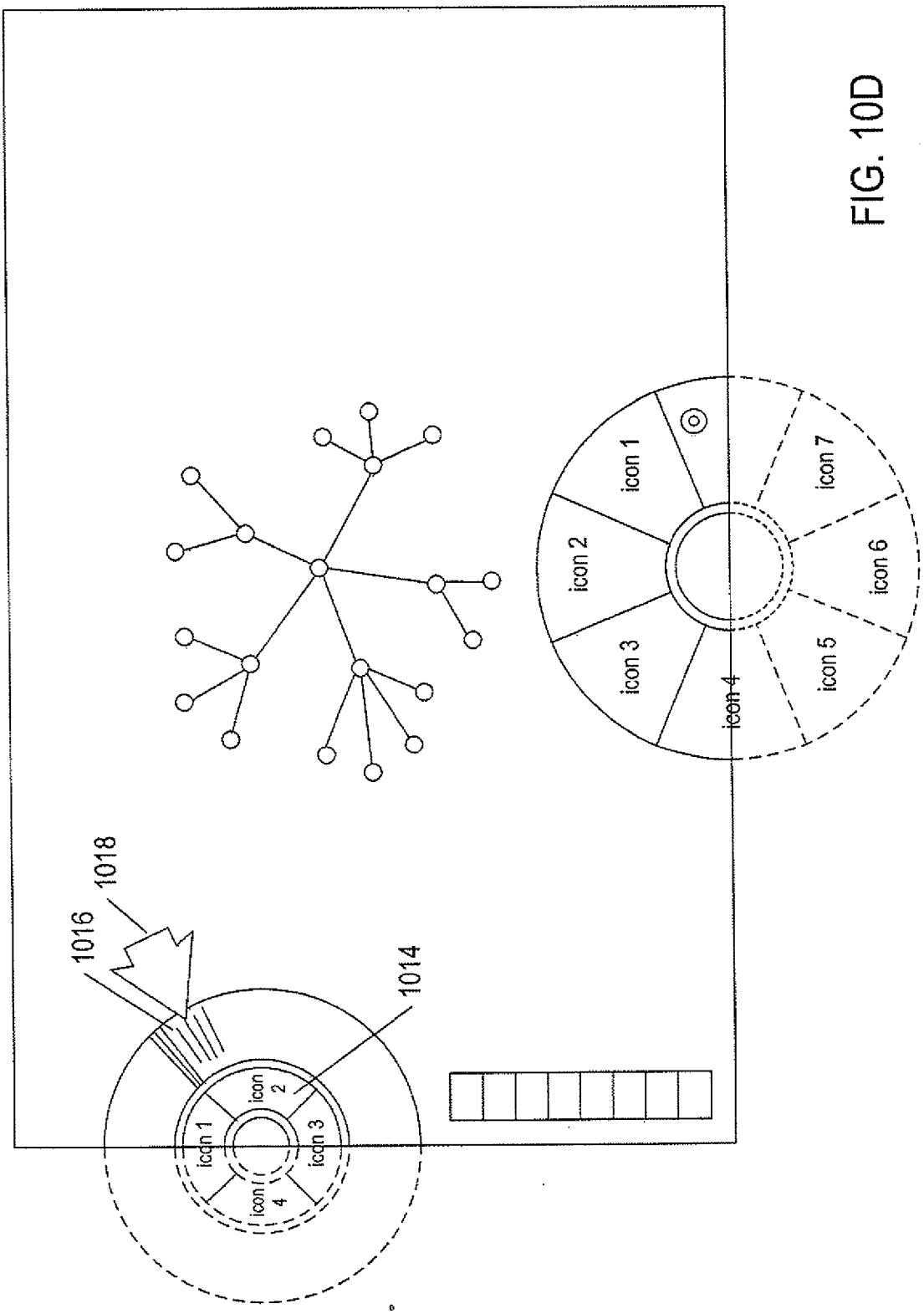

icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
1016
1018
1014
icon 1
icon 2
icon 3
icon 4

FIG. 10D

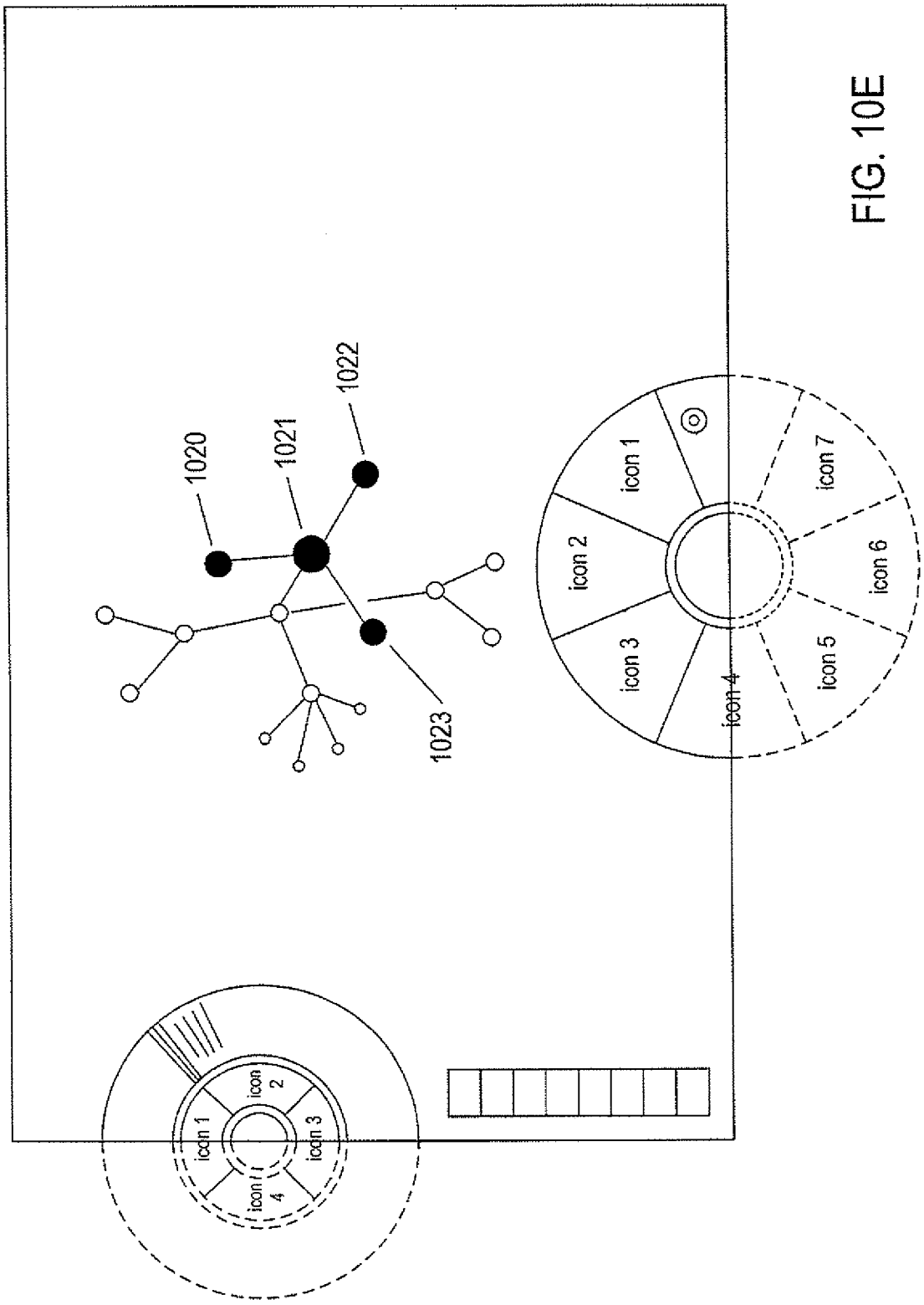

1020
1021
1022
1023
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10E

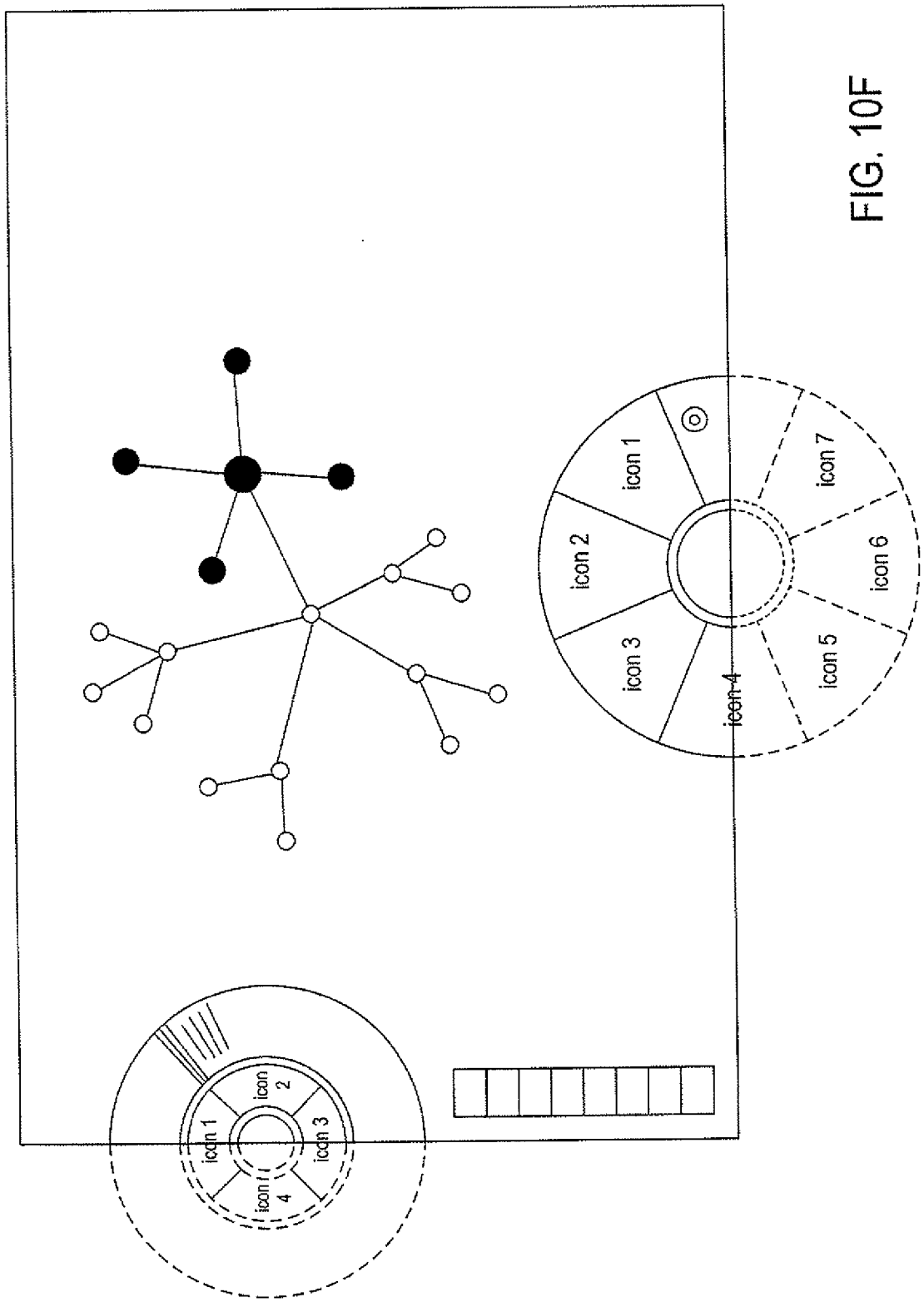

icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10F

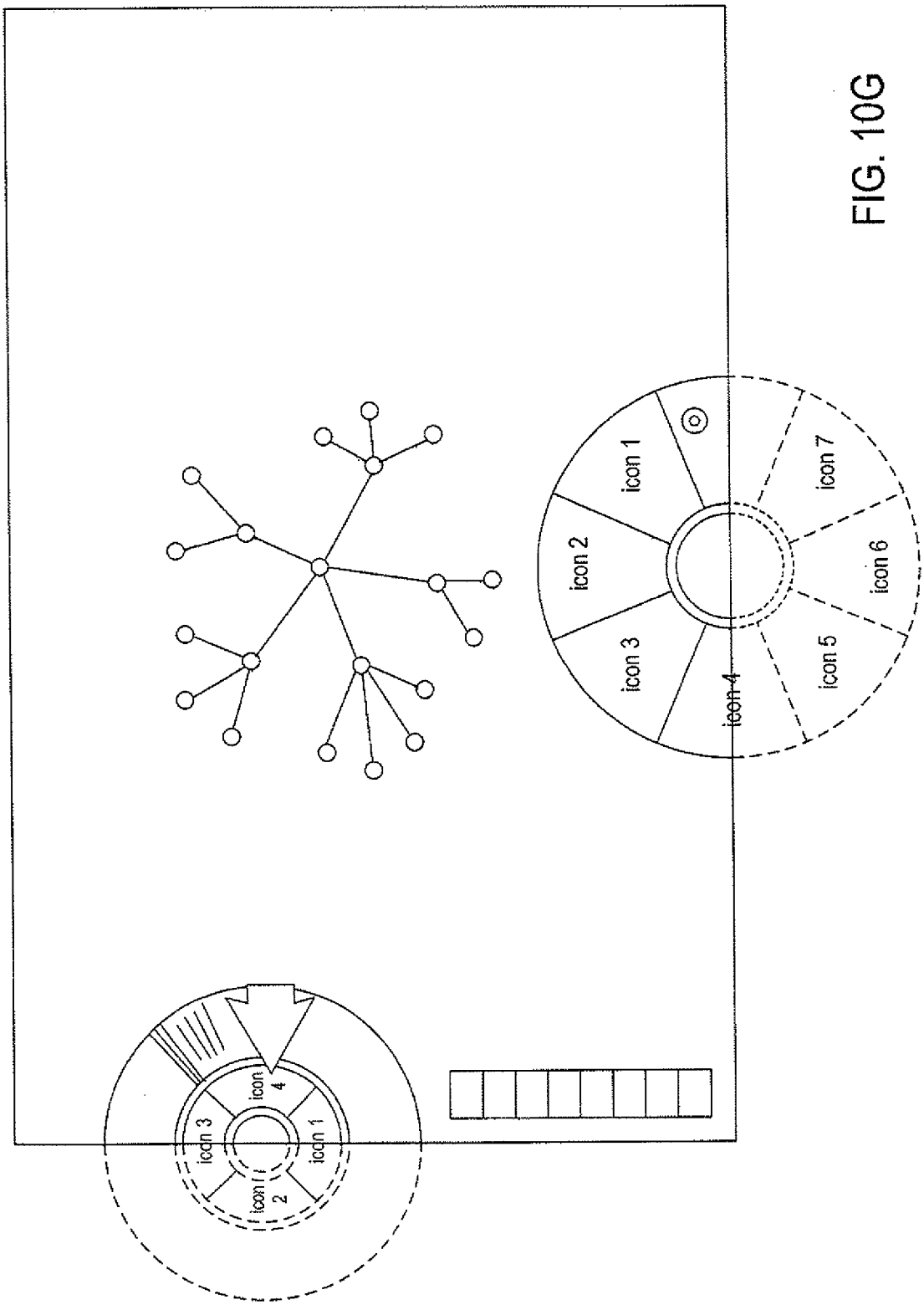

icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10G

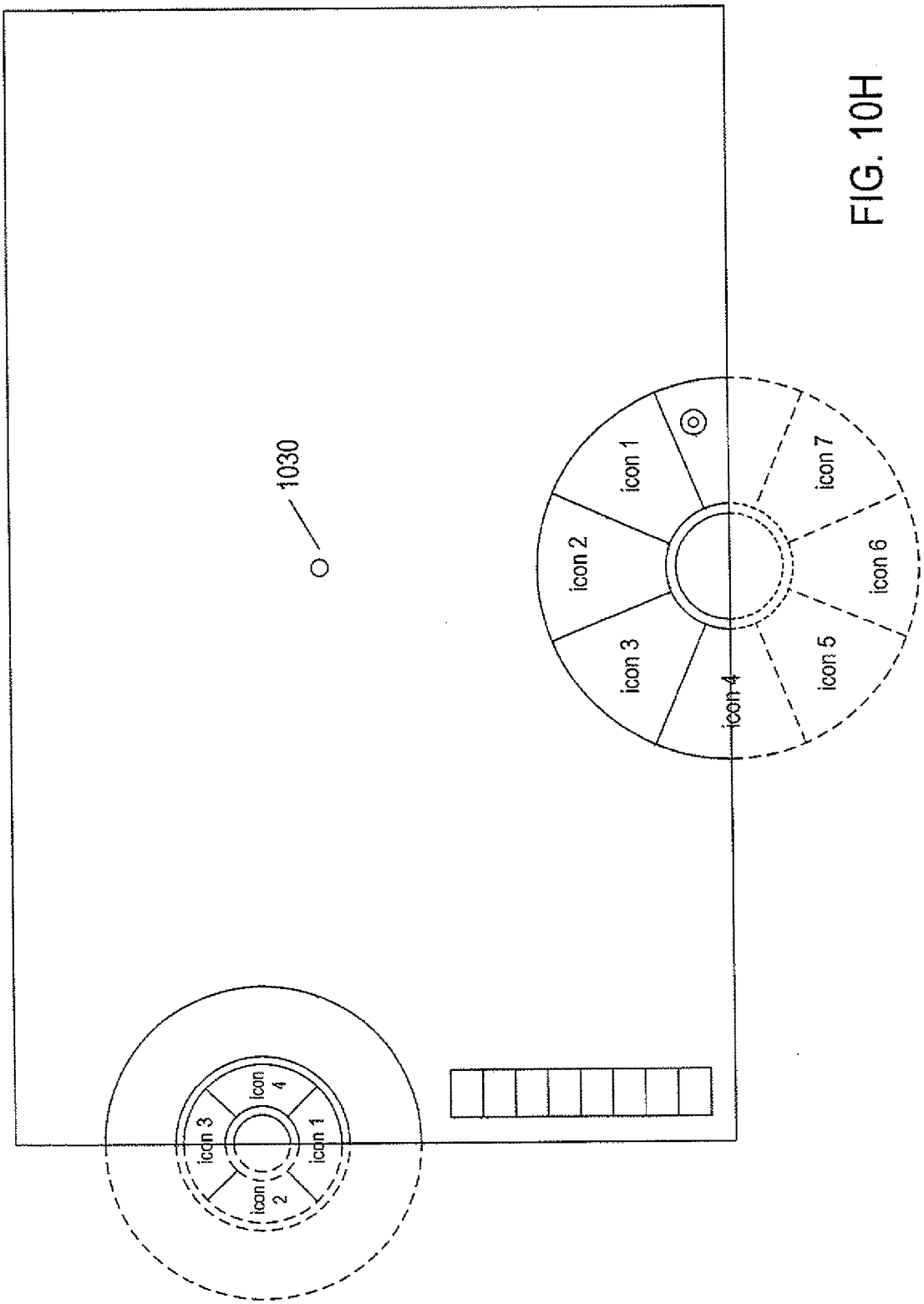

1030
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10H

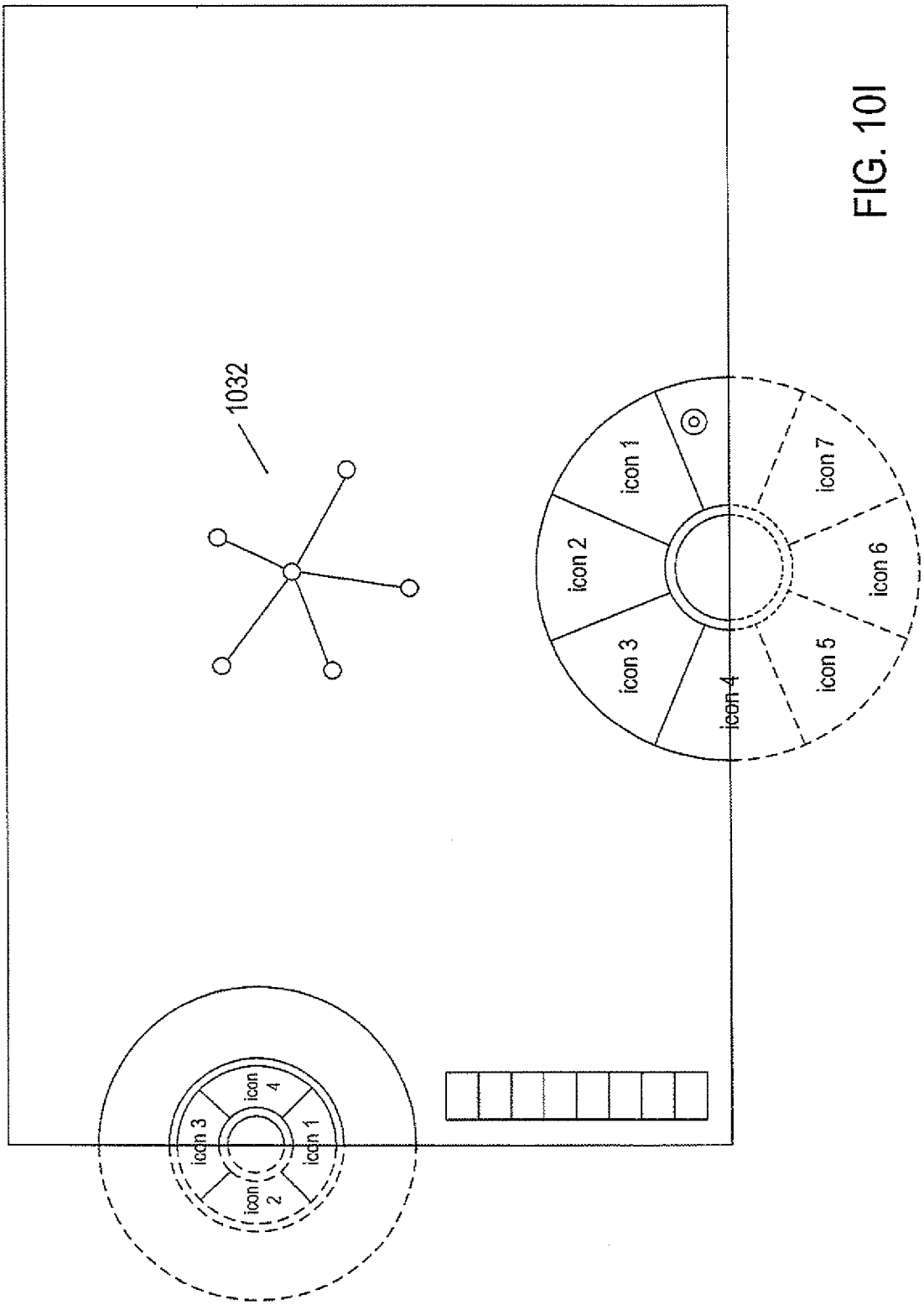

1032
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10I

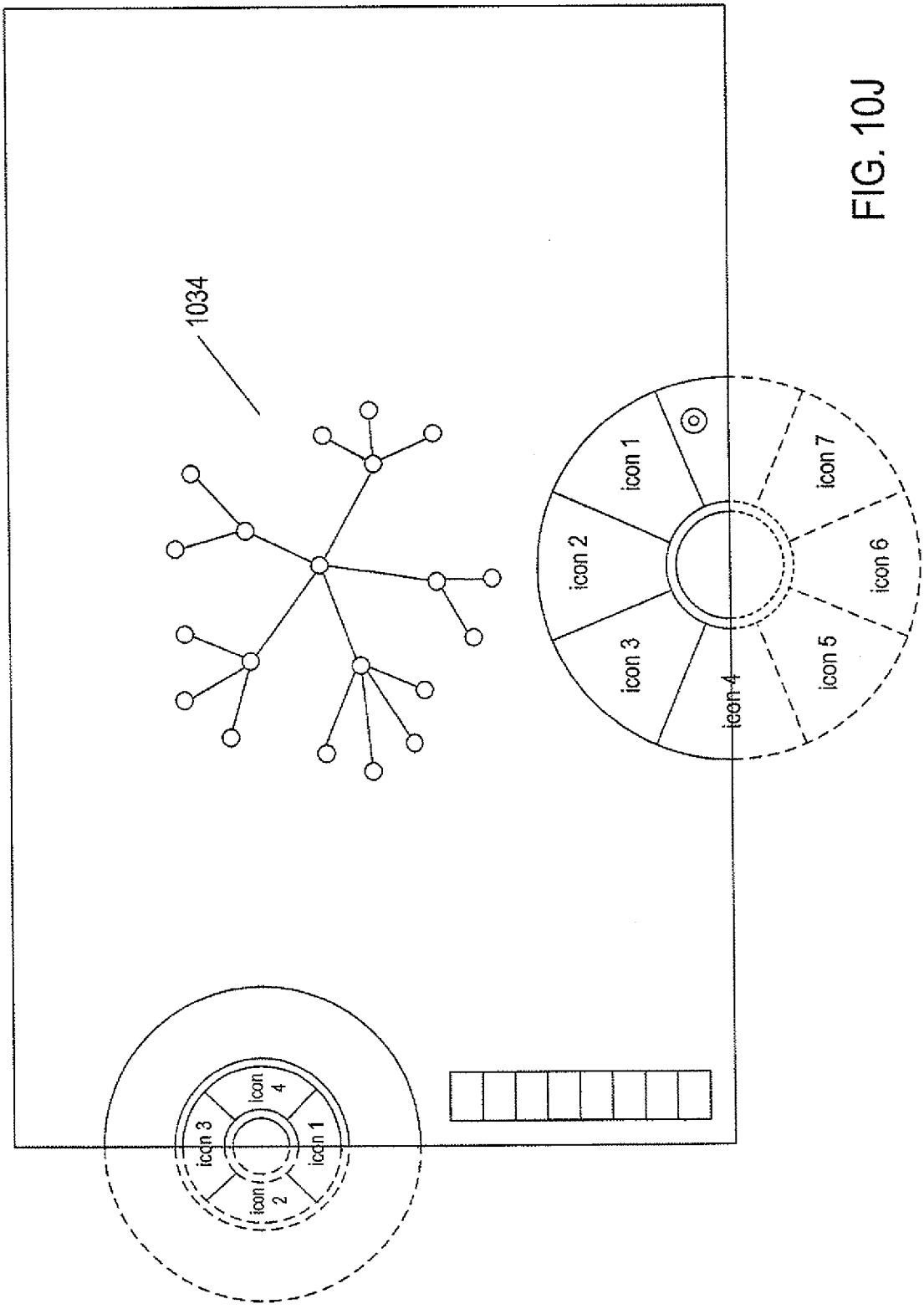

1034
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7
icon 1
icon 2
icon 3
icon 4

FIG. 10J

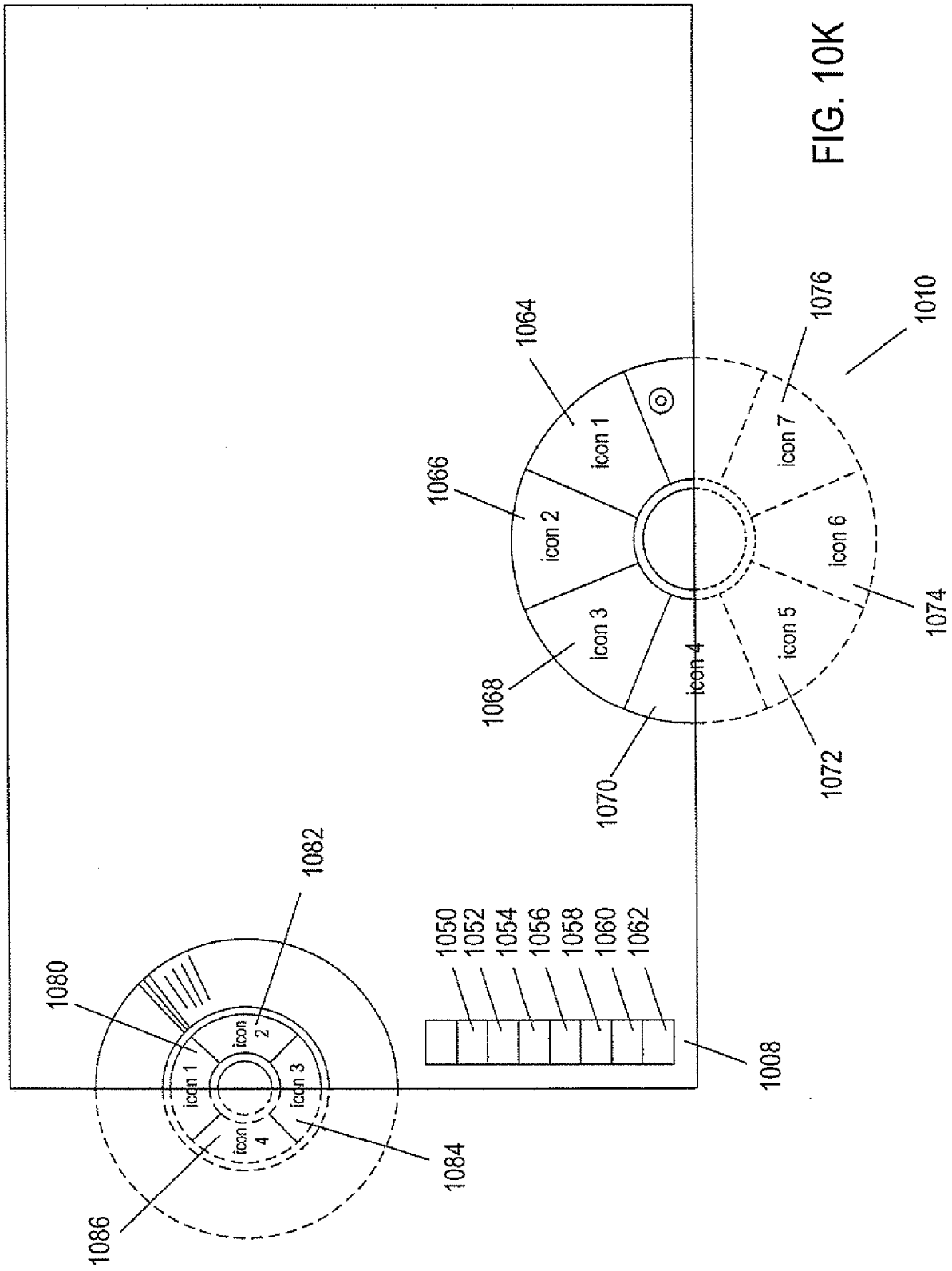

1010
1064
1066
1068
1070
1072
1074
1076
1080
1082
1084
1086
1008
1050
1052
1054
1056
1058
1060
1062
icon 1
icon 2
icon 3
icon 4
icon 5
icon 6
icon 7

FIG. 10K

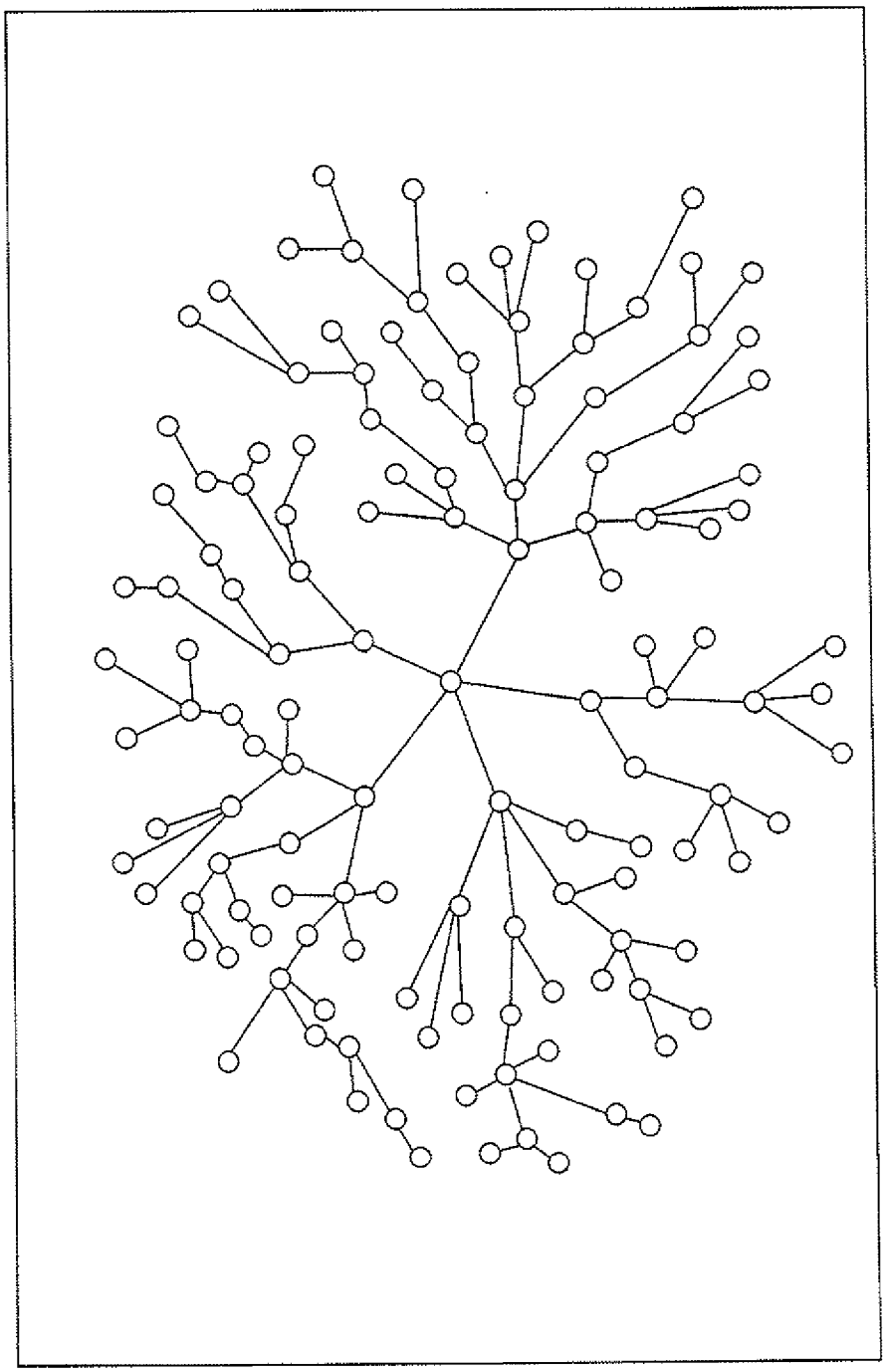


FIG. 11A

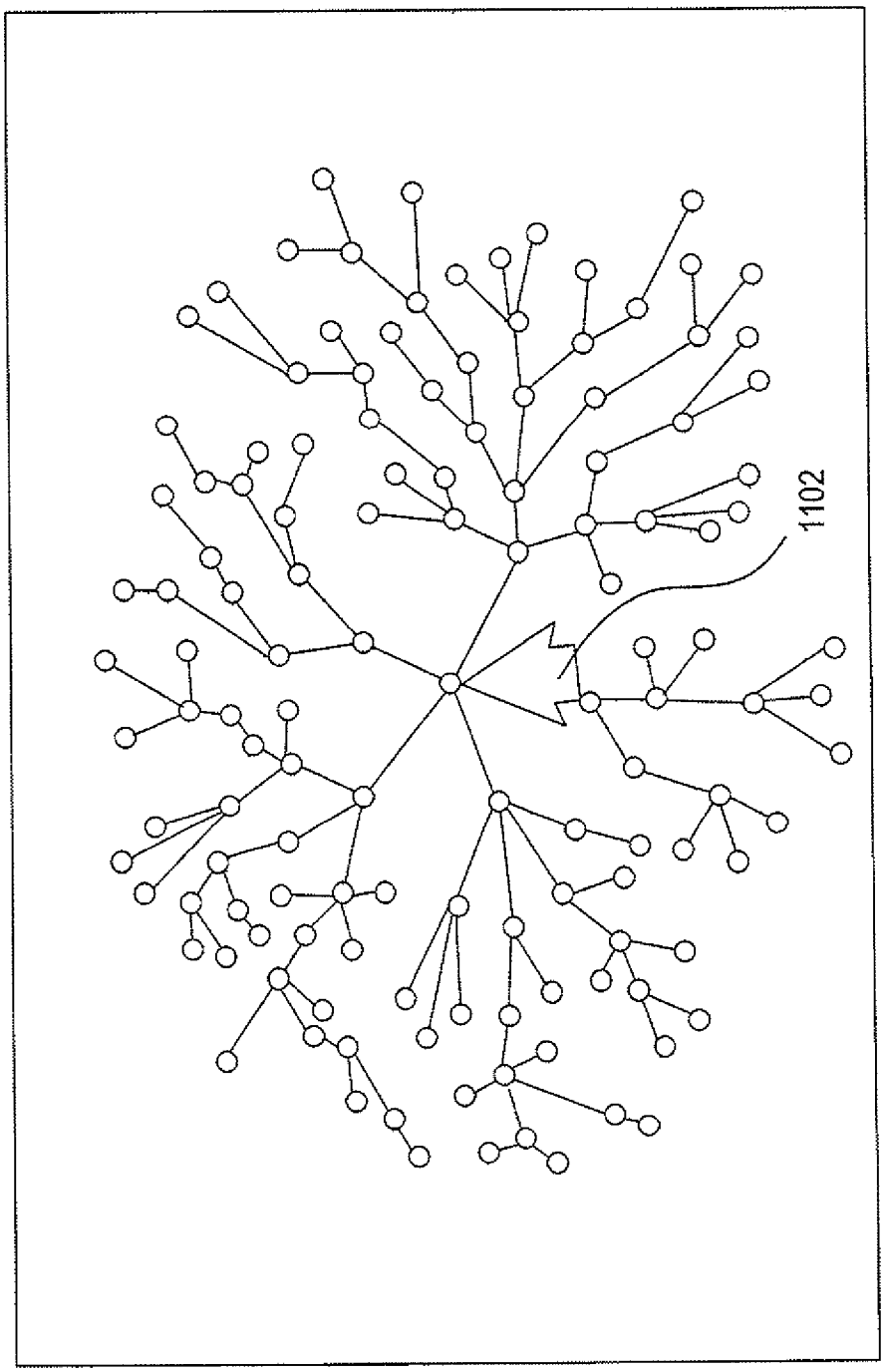


FIG. 11B

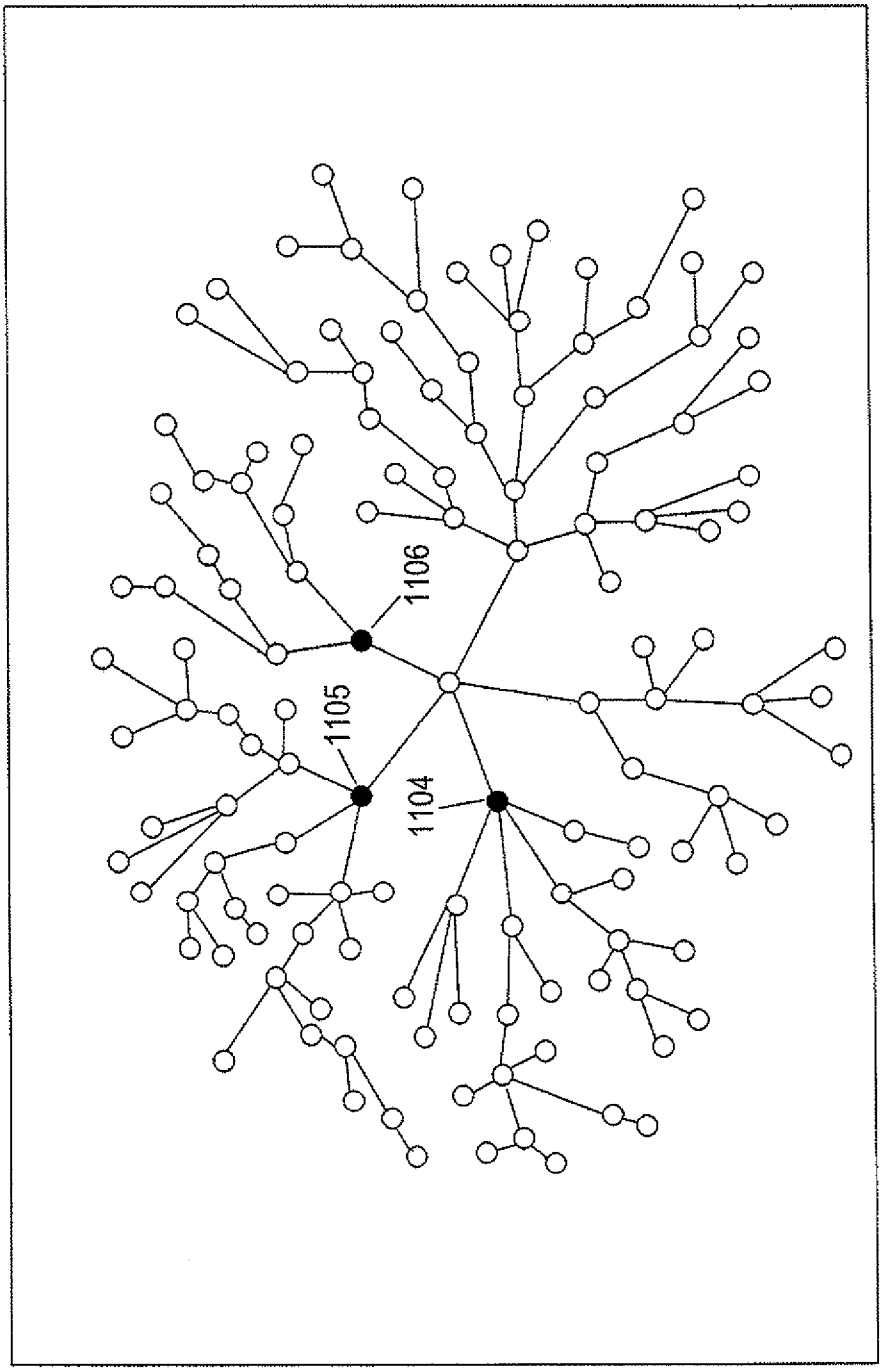


FIG. 11C

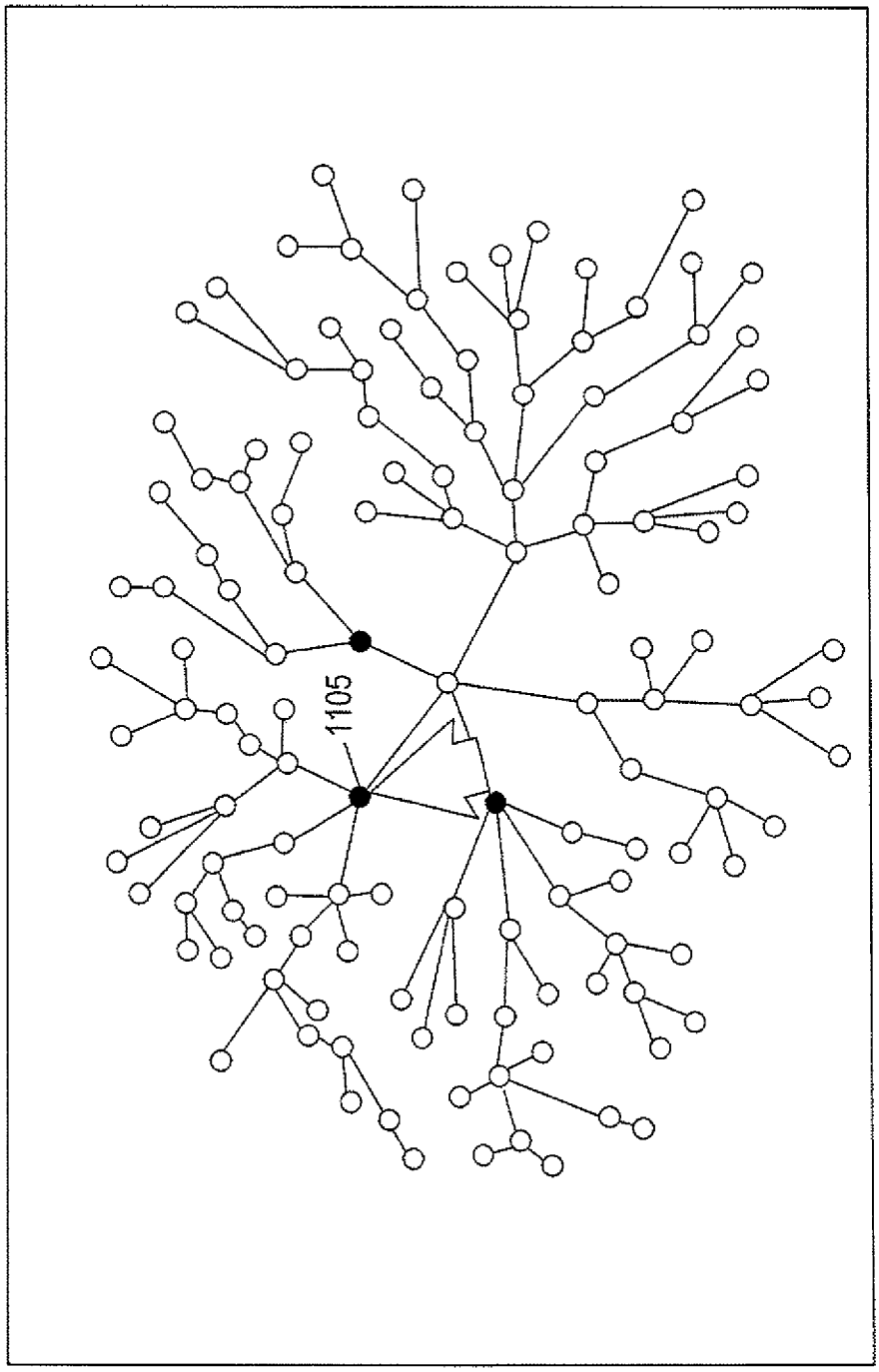


FIG. 11D

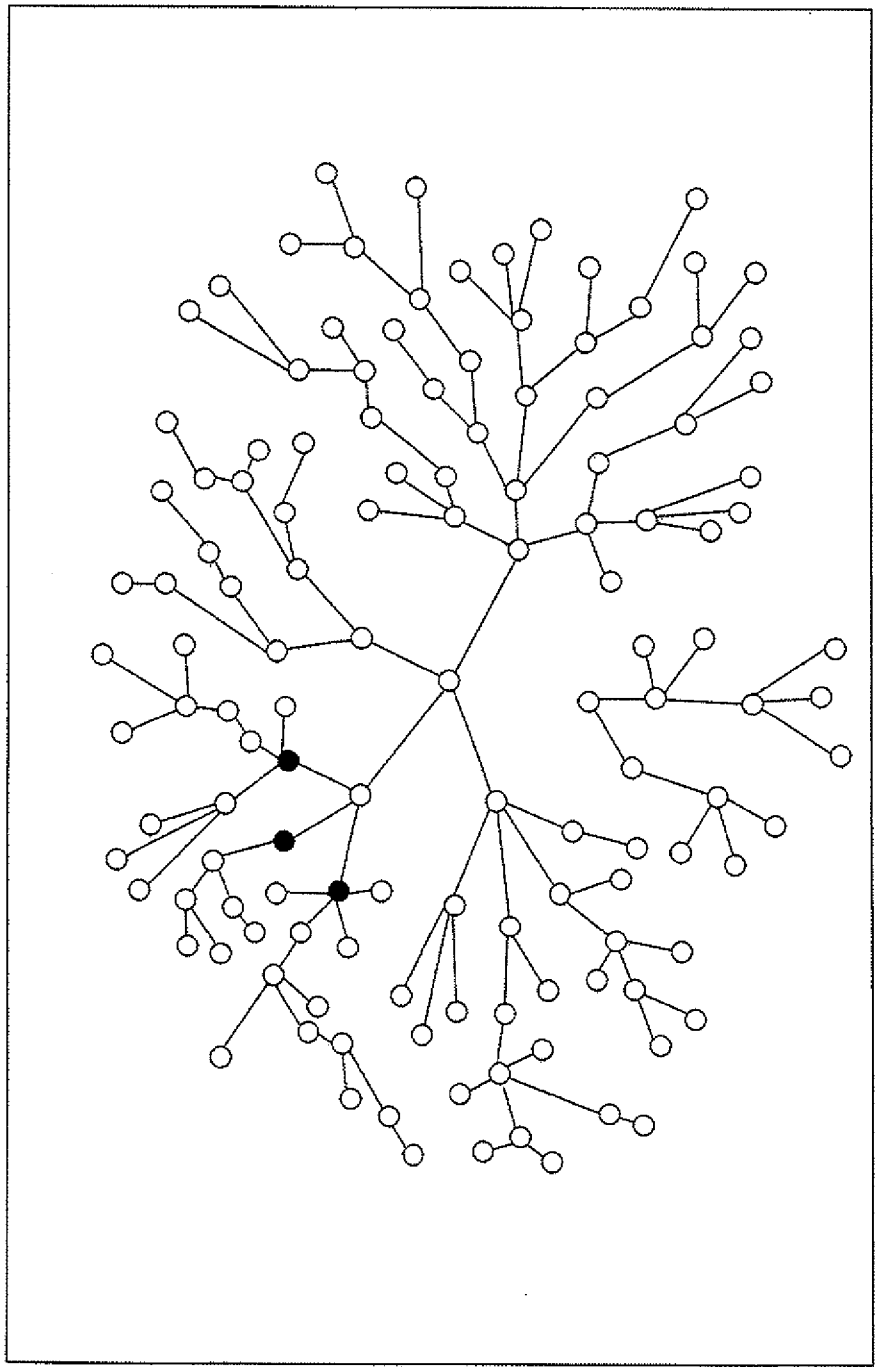


FIG. 11E

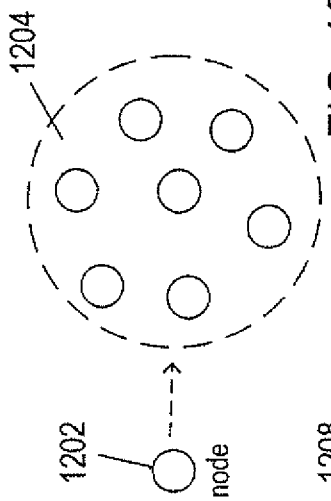


FIG. 12A

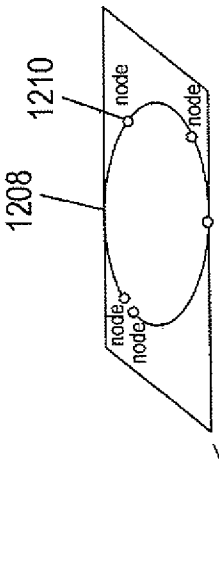


FIG. 12B

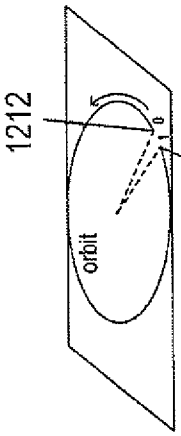


FIG. 12C

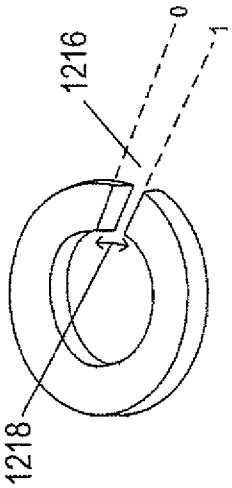


FIG. 12D

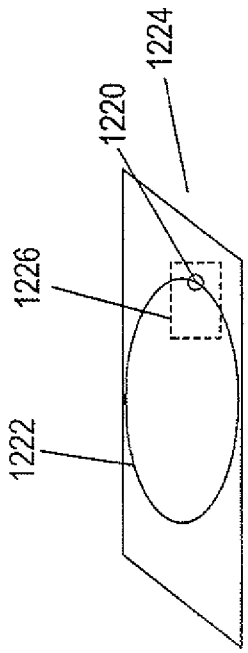


FIG. 12E

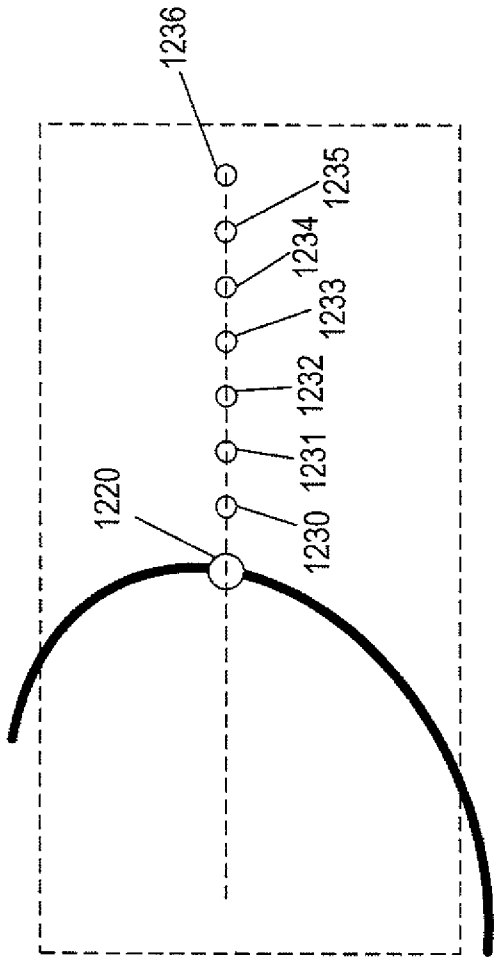


FIG. 12F

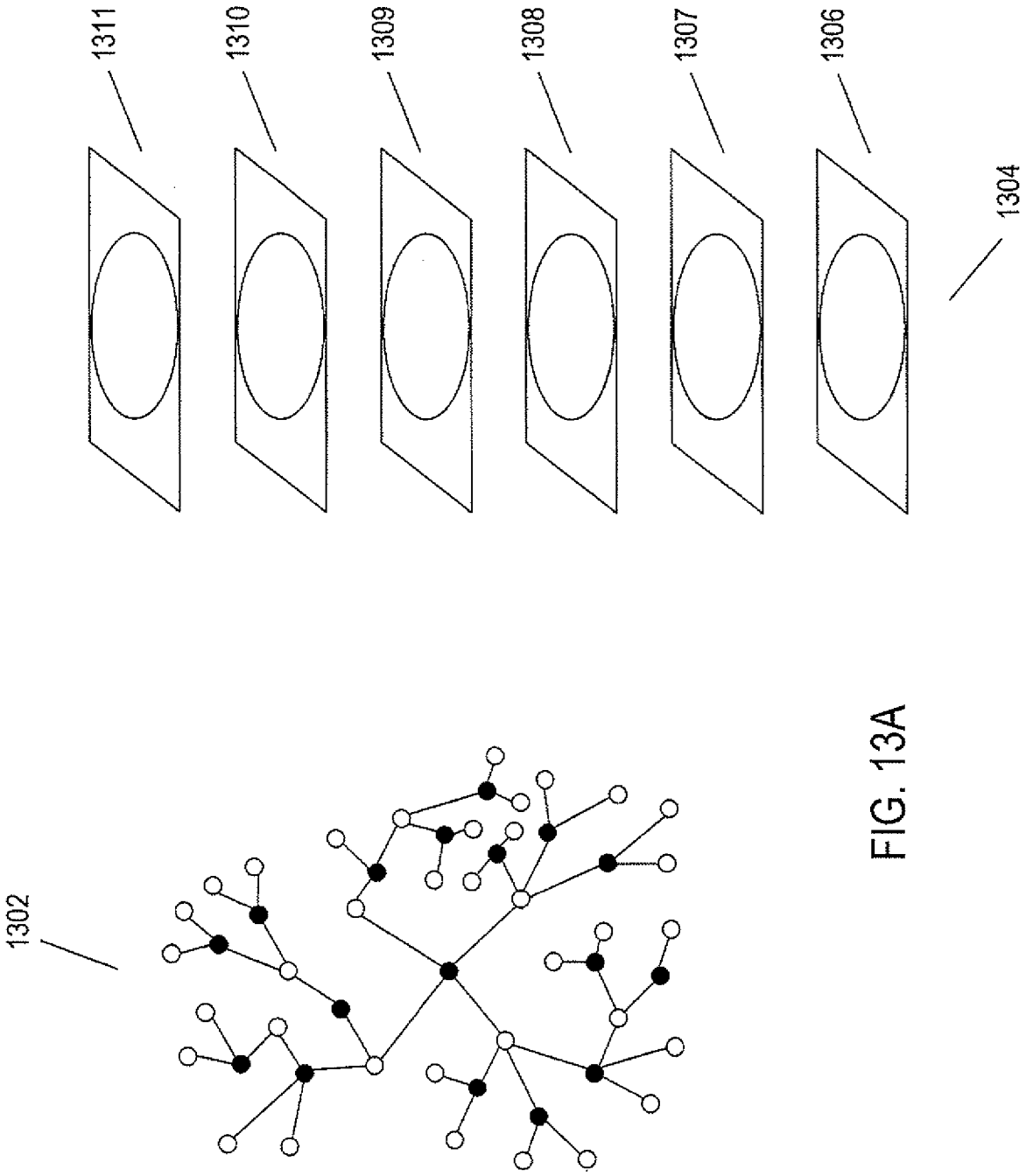


FIG. 13A

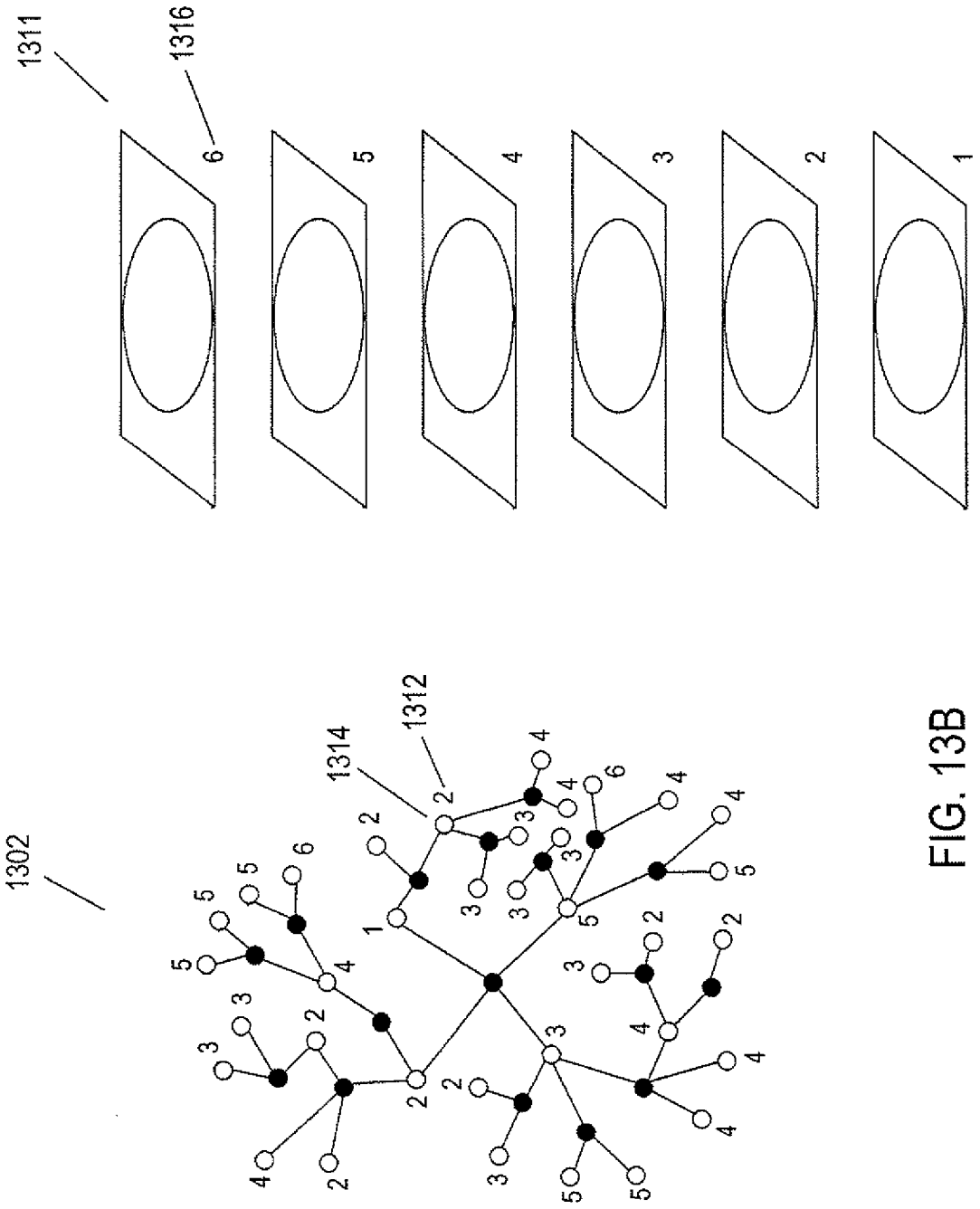


FIG. 13B

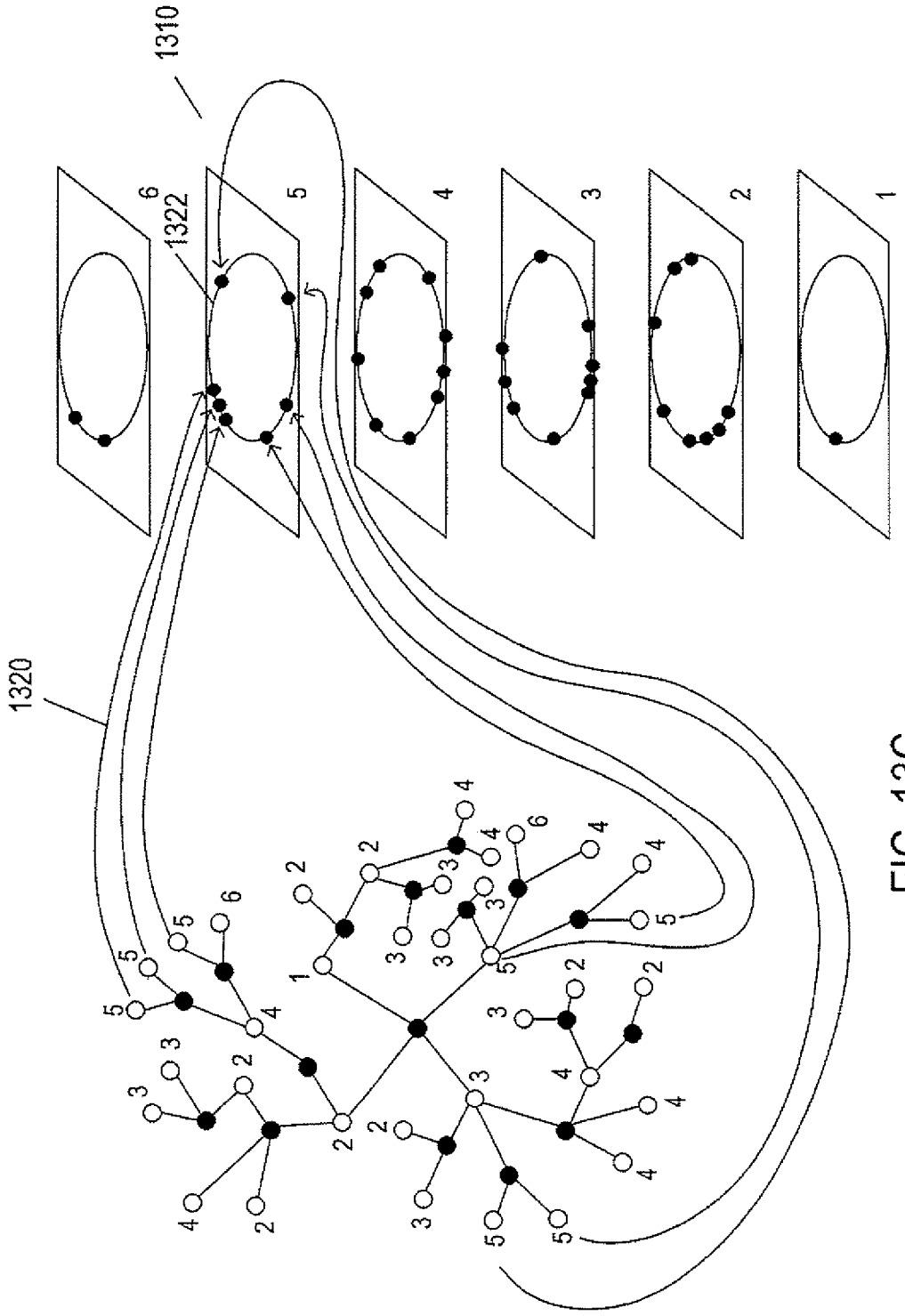


FIG. 13C

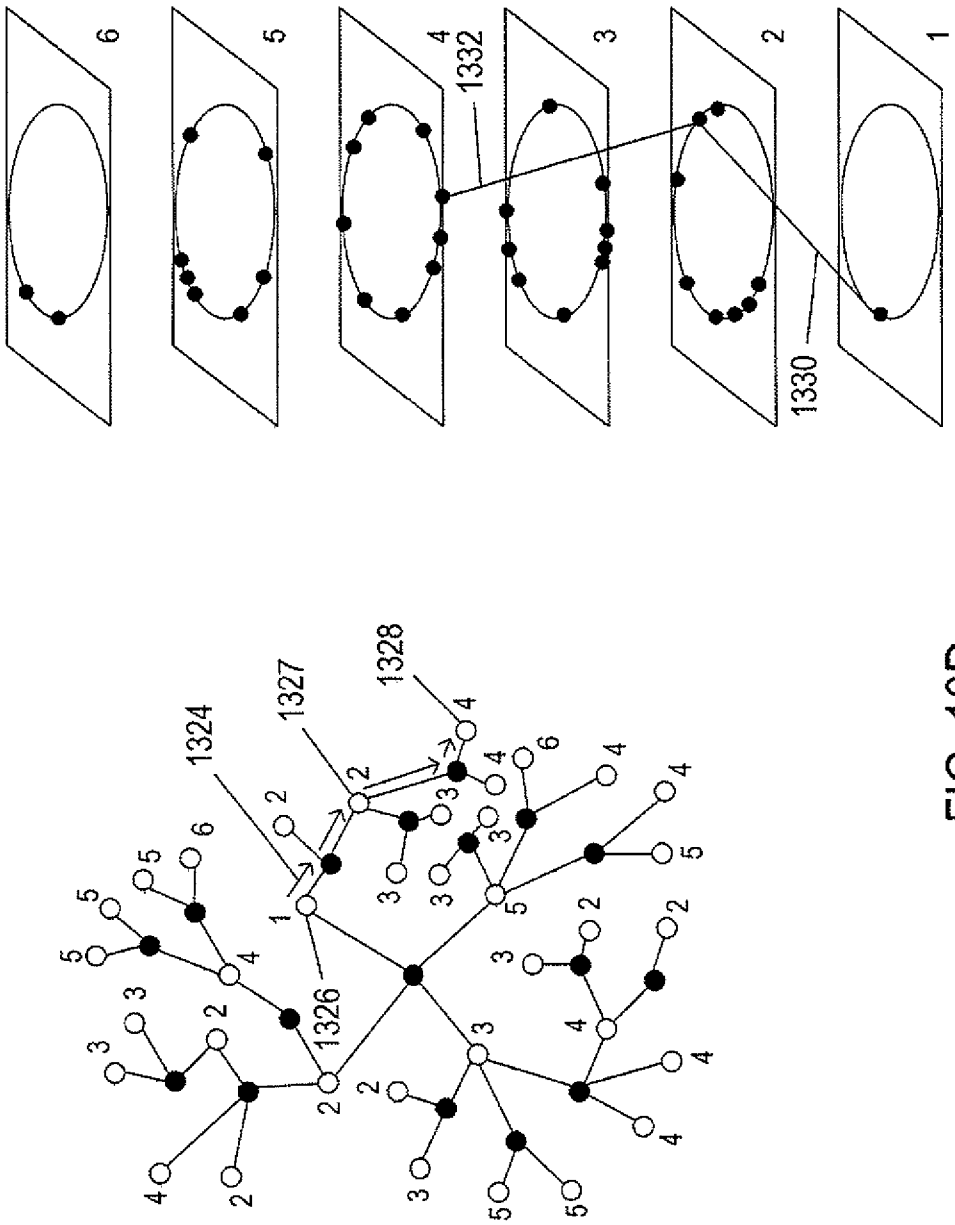


FIG. 13D

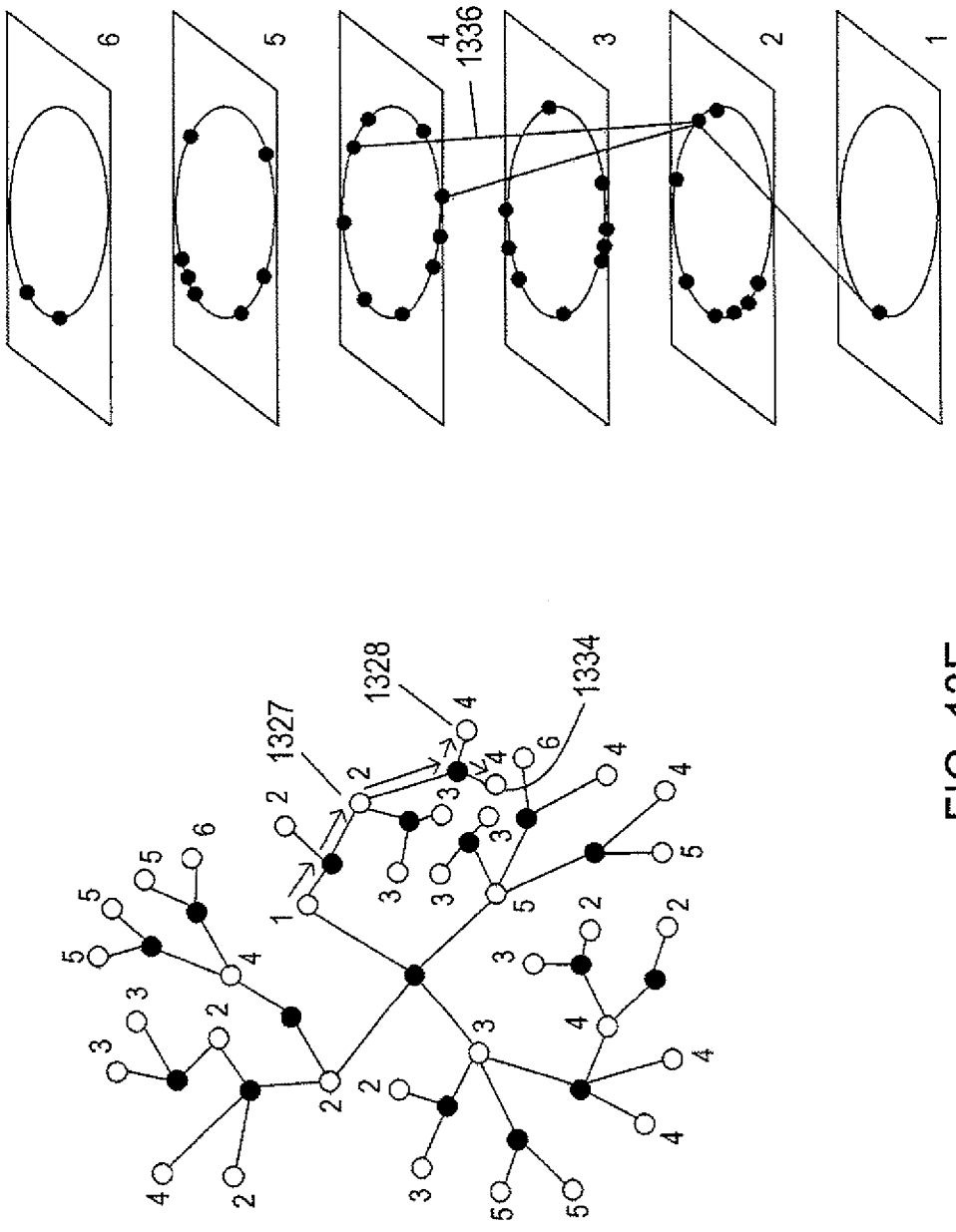


FIG. 13E

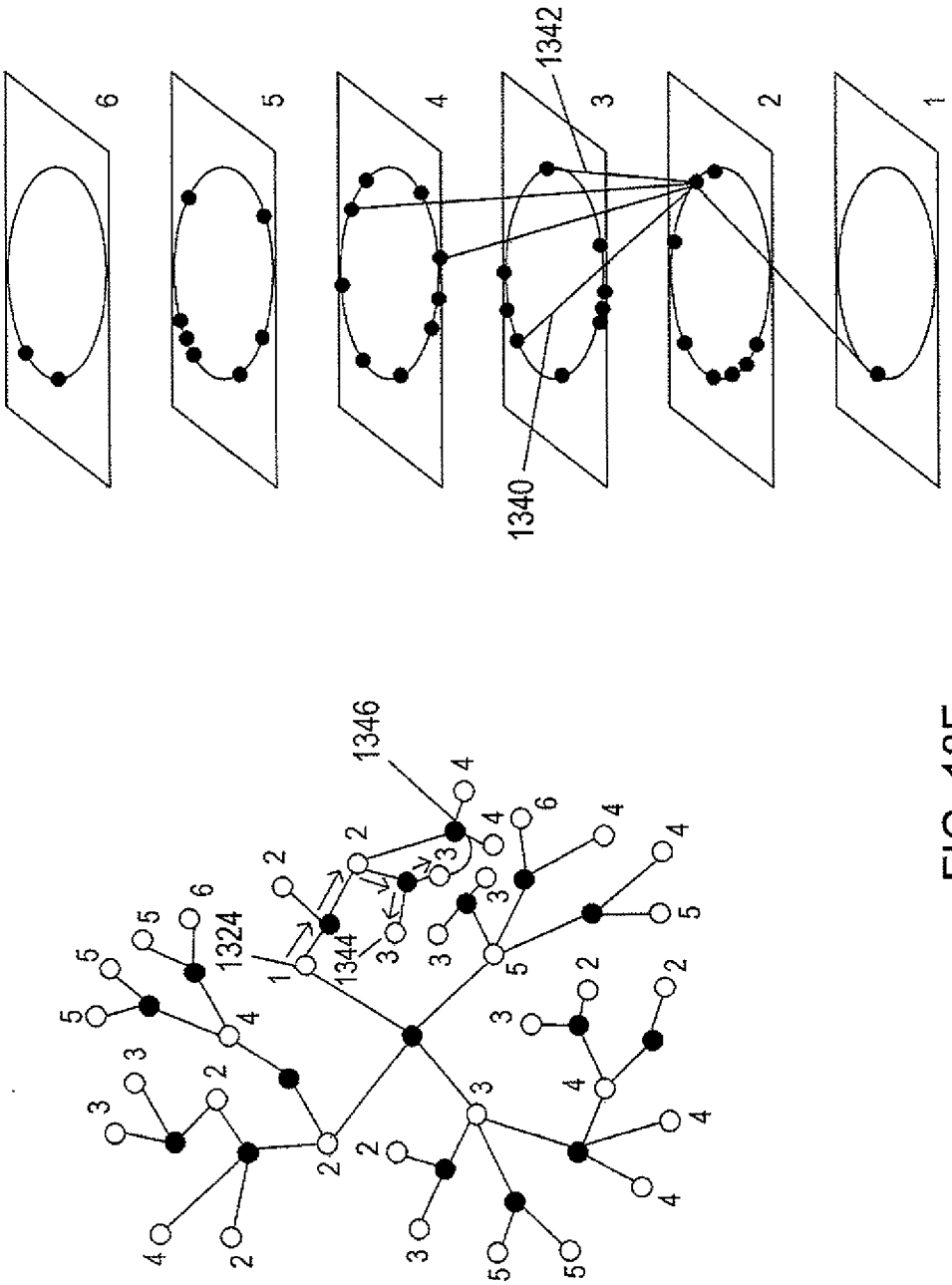


FIG. 13F

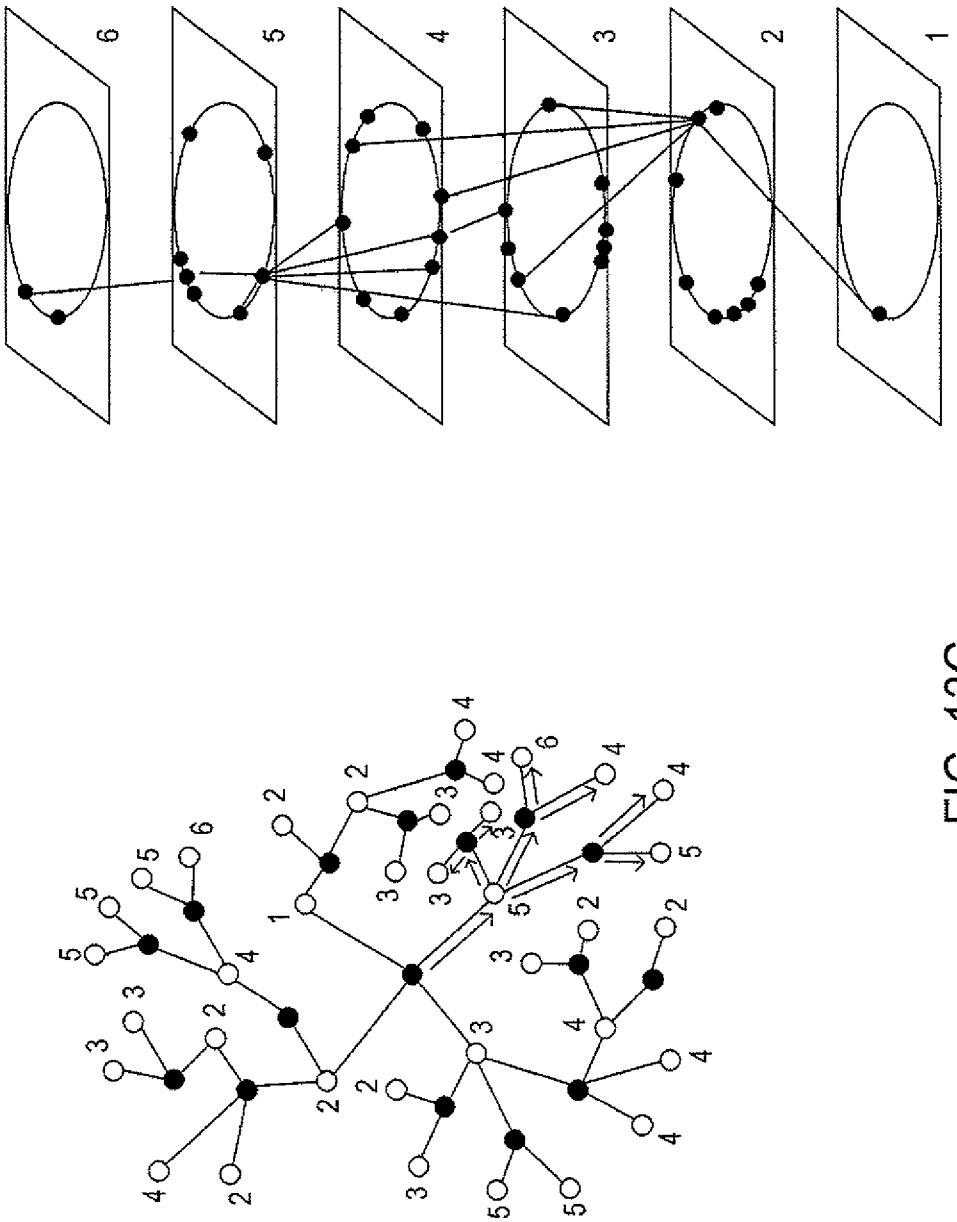
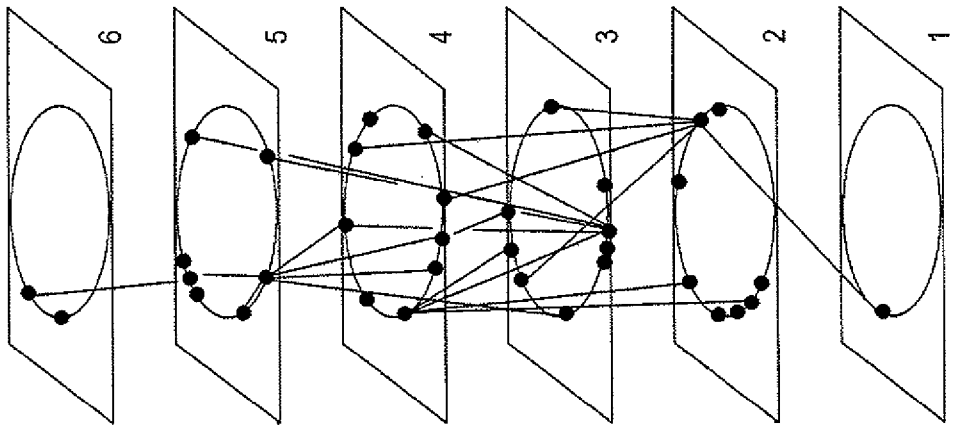


FIG. 13G



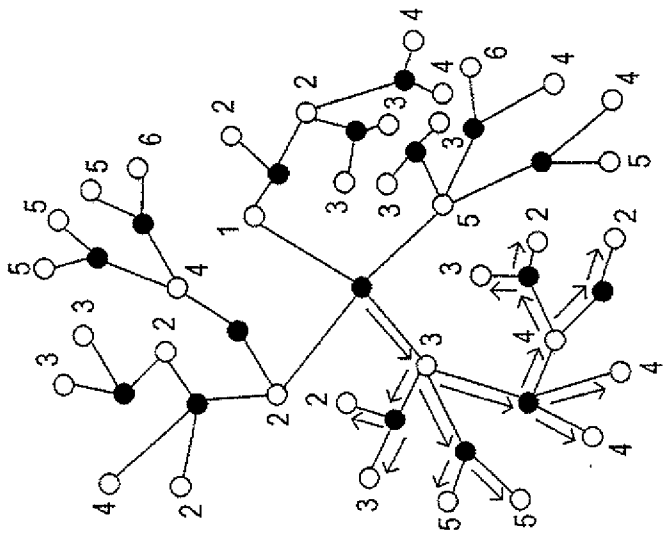


FIG. 13H

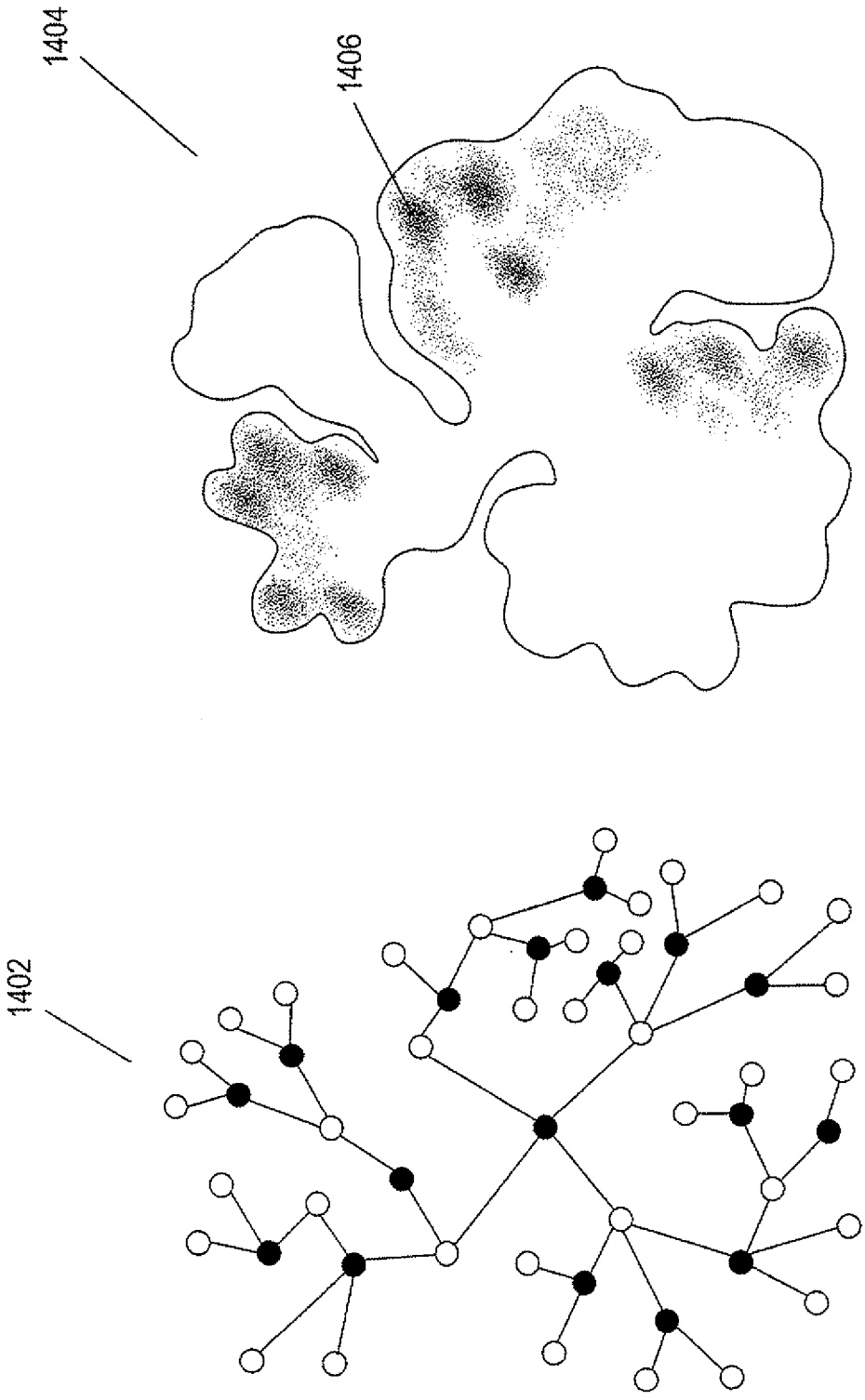


FIG. 14

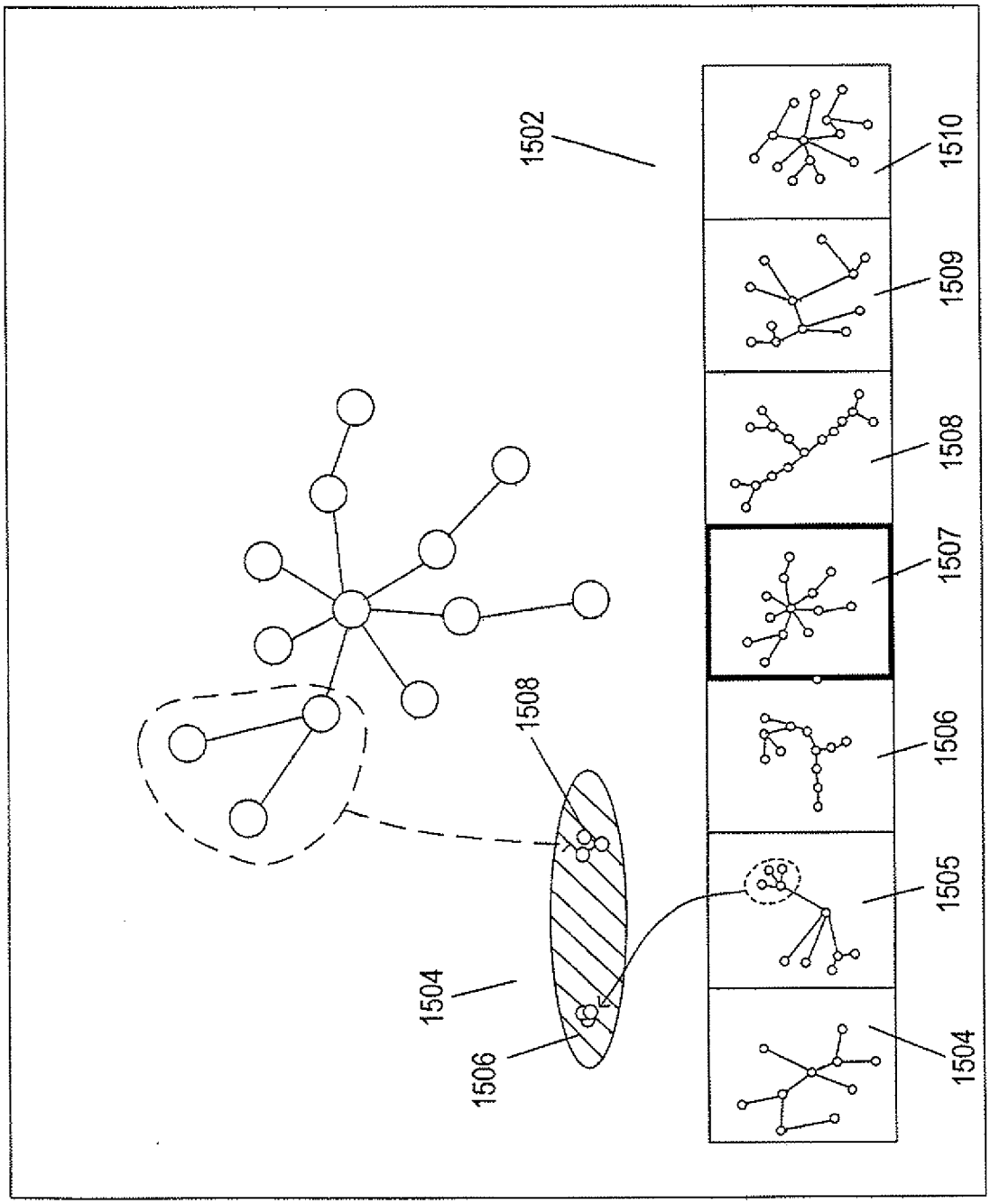


FIG. 15

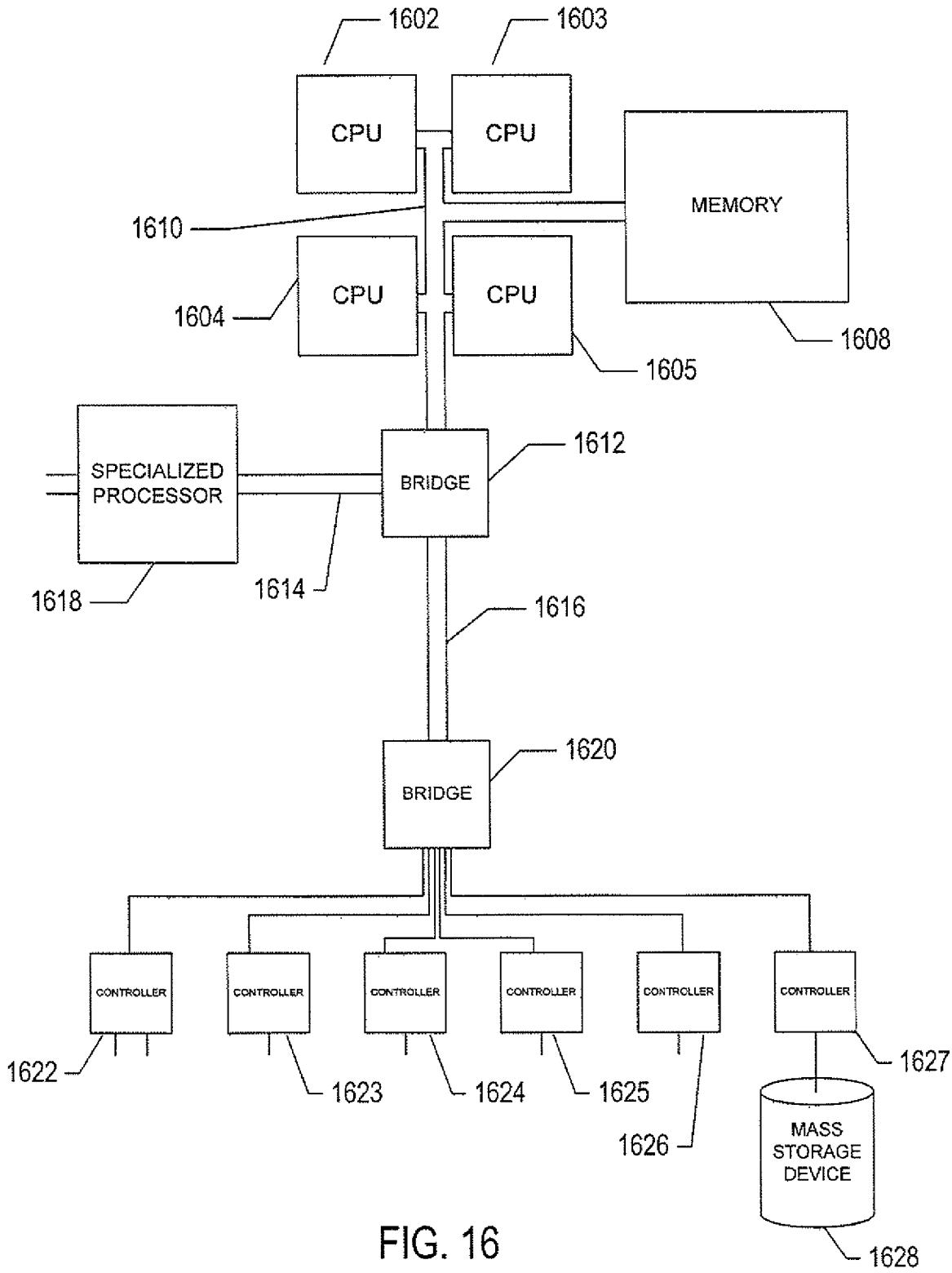

1602
1603
CPU
CPU
MEMORY
1610
CPU
CPU
1604
1608
1605
SPECIALIZED PROCESSOR
BRIDGE
1612
1618
1614
1616
BRIDGE
1620
CONTROLLER
CONTROLLER
CONTROLLER
CONTROLLER
CONTROLLER
CONTROLLER
1627
1622
1623
1624
1625
1626
MASS STORAGE DEVICE
1628

FIG. 16

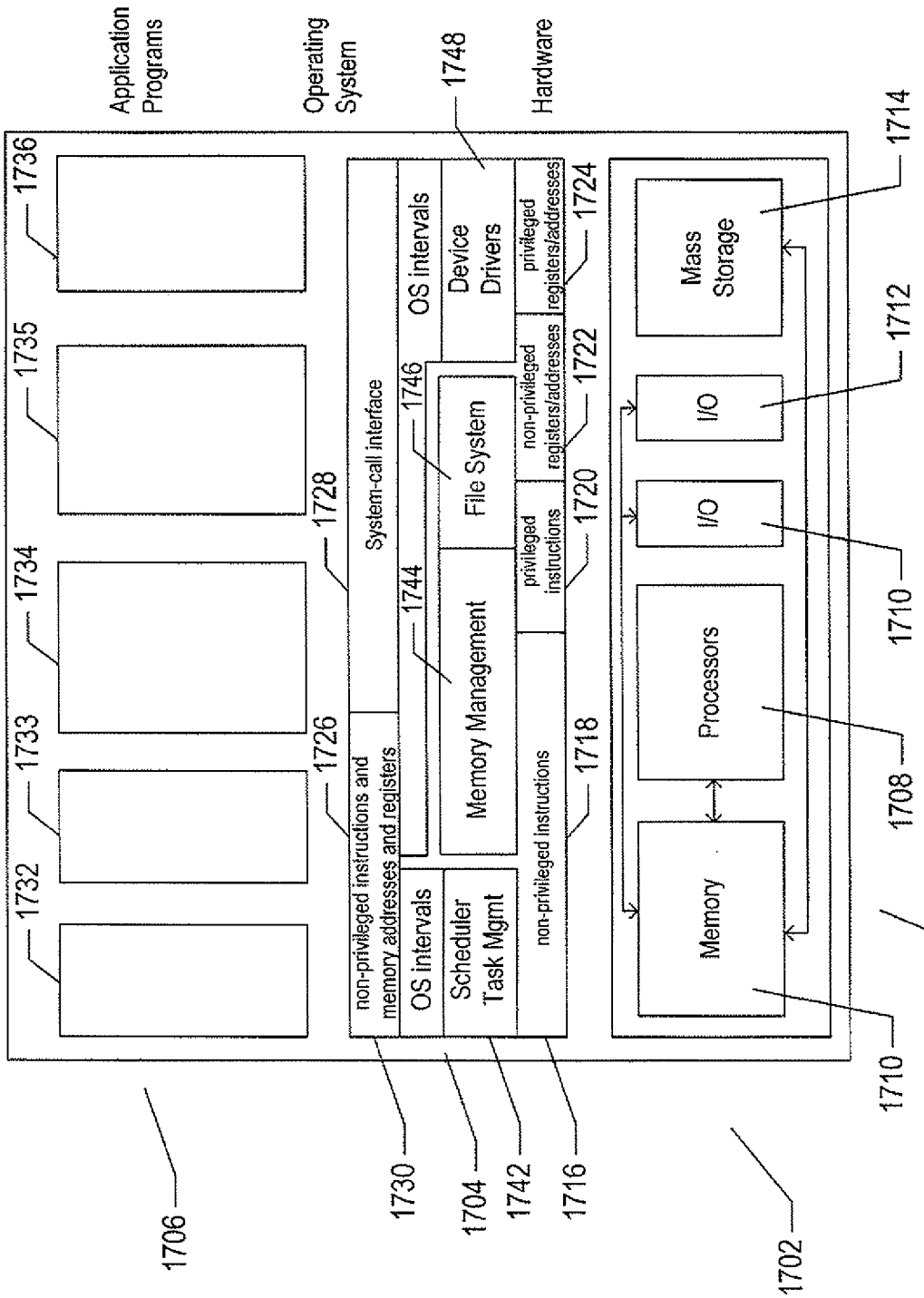

Application Programs
Operating System
Hardware
1732
1733
1734
1735
1736
1726
1728
1748
1724
1722
1720
1718
1746
1744
1714
1712
1710
1708
1700
1706
1730
1704
1742
1716
1702
non-privileged instructions and memory addresses and registers
System-call interface
OS intervals
Scheduler Task Mgmt
non-privileged instructions
Memory Management
File System
Device Drivers
privileged instructions
non-privileged registers/addresses
privileged registers/addresses
Memory
Processors
I/O
I/O
Mass Storage

FIG. 17

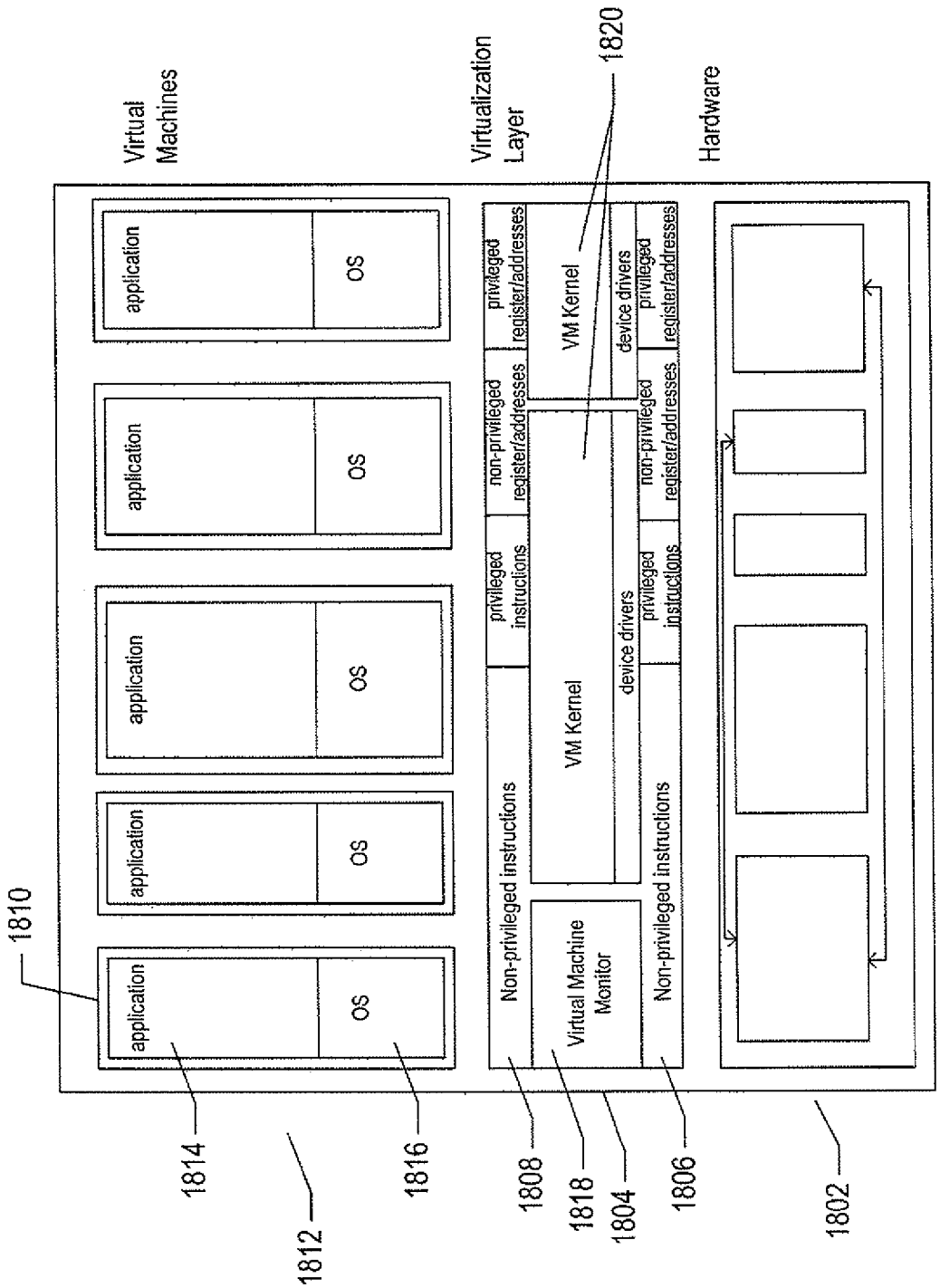

Virtual Machines
Virtualization Layer
Hardware
1820
application
OS
Non-privileged instructions
Virtual Machine Monitor
privileged instructions
non-privileged register/addresses
VM Kernel
device drivers
privileged register/addresses
1810
1814
1812
1816
1808
1818
1804
1806
1802
1800

FIG. 18

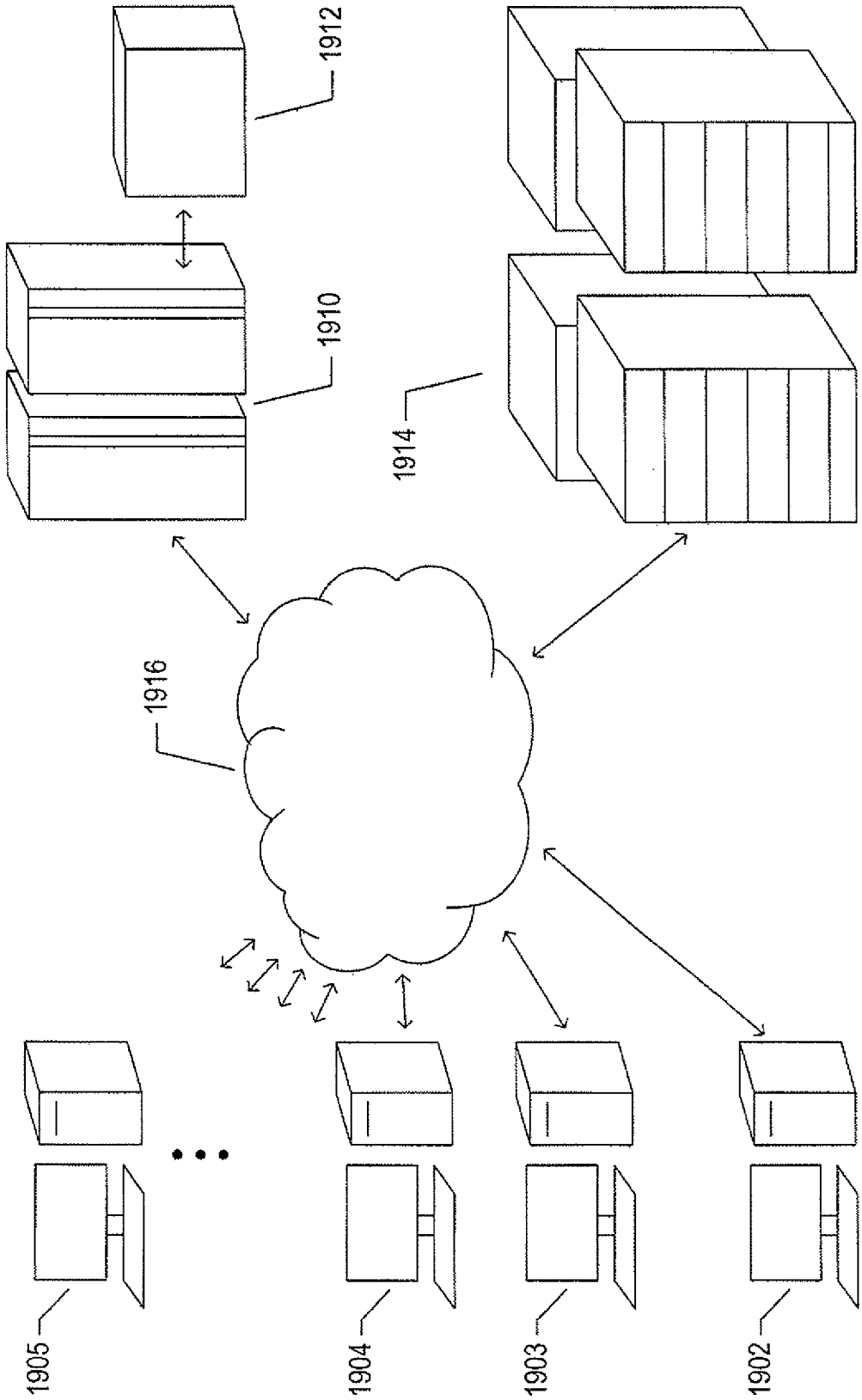


FIG. 19

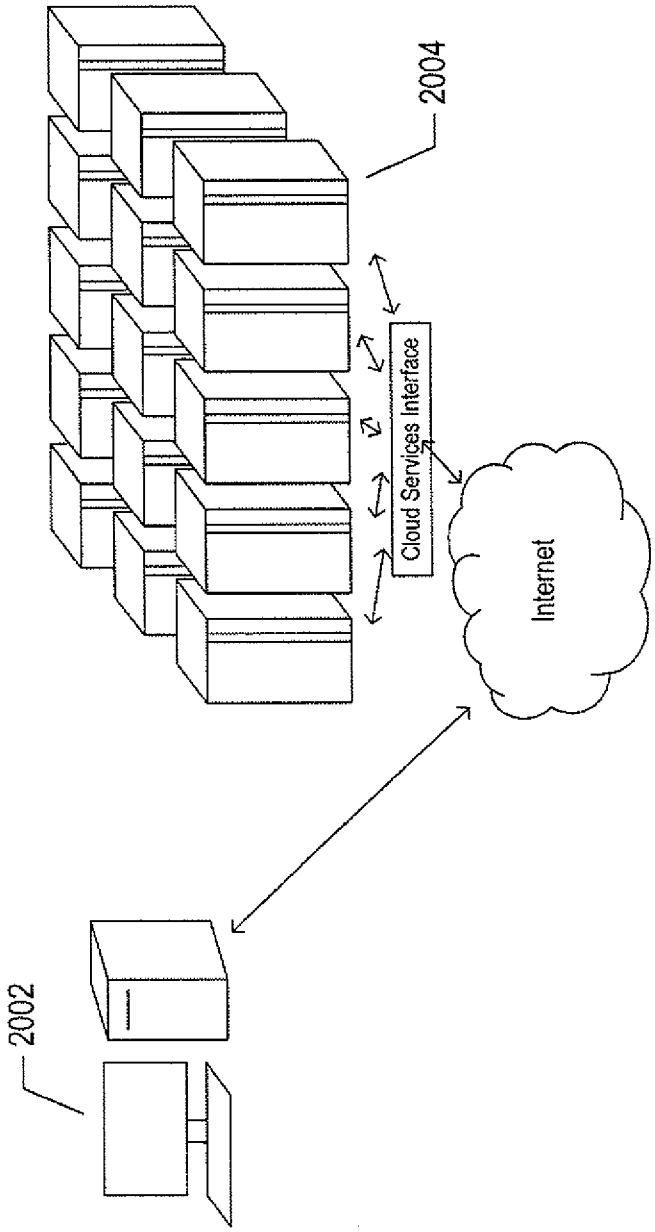


FIG. 20

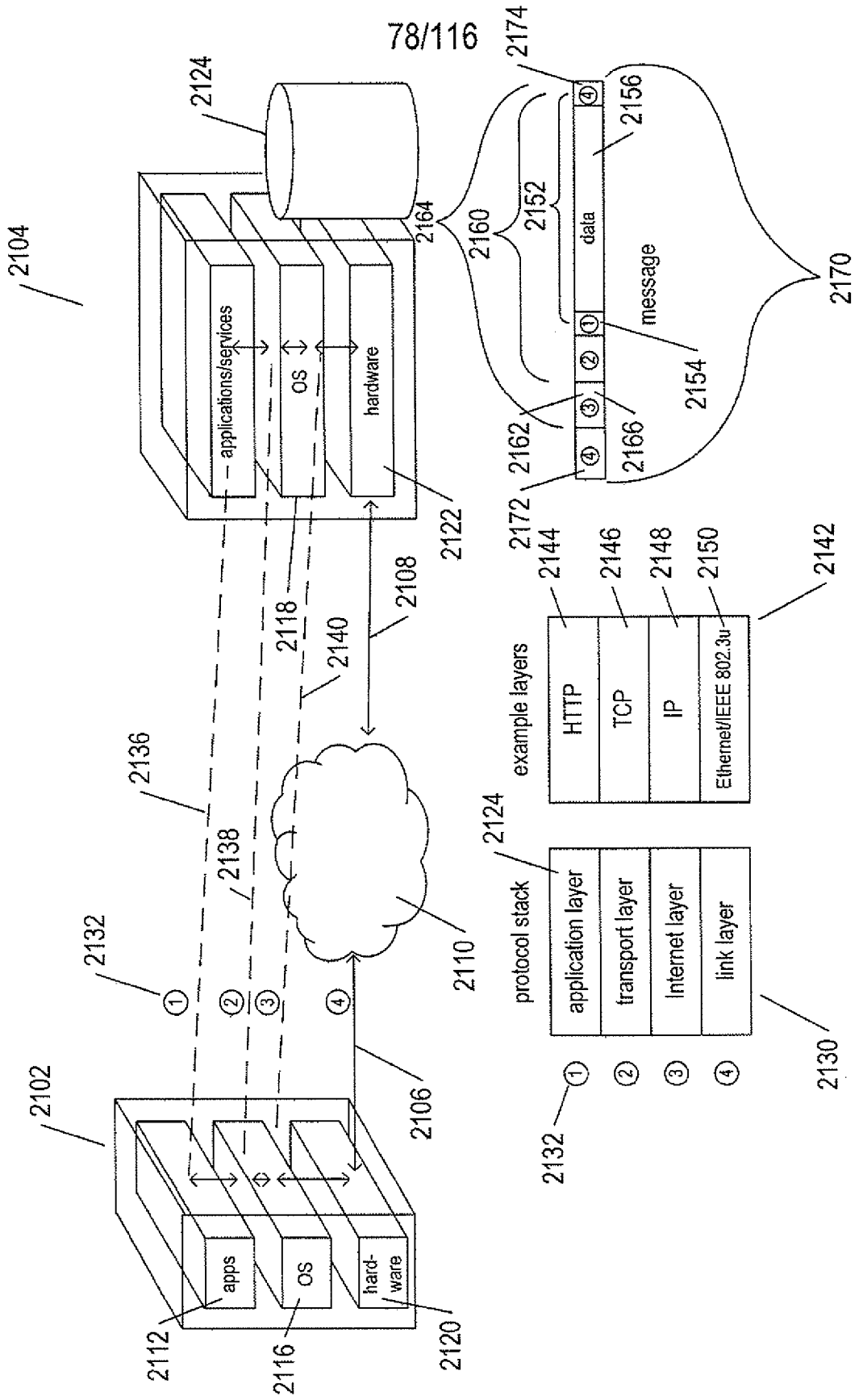

2104
2124
applications/services
OS
hardware
2122
2164
2174
2156
2160
2152
data
message
2154
2162
2166
2170
2172
2108
2118
2140
2136
2138
2132
2110
2106
2102
apps
OS
hard-
ware
2112
2116
2120
example layers
HTTP
TCP
IP
Ethernet/IEEE 802.3u
2144
2146
2148
2150
2142
protocol stack
application layer
transport layer
Internet layer
link layer
2130

FIG. 21

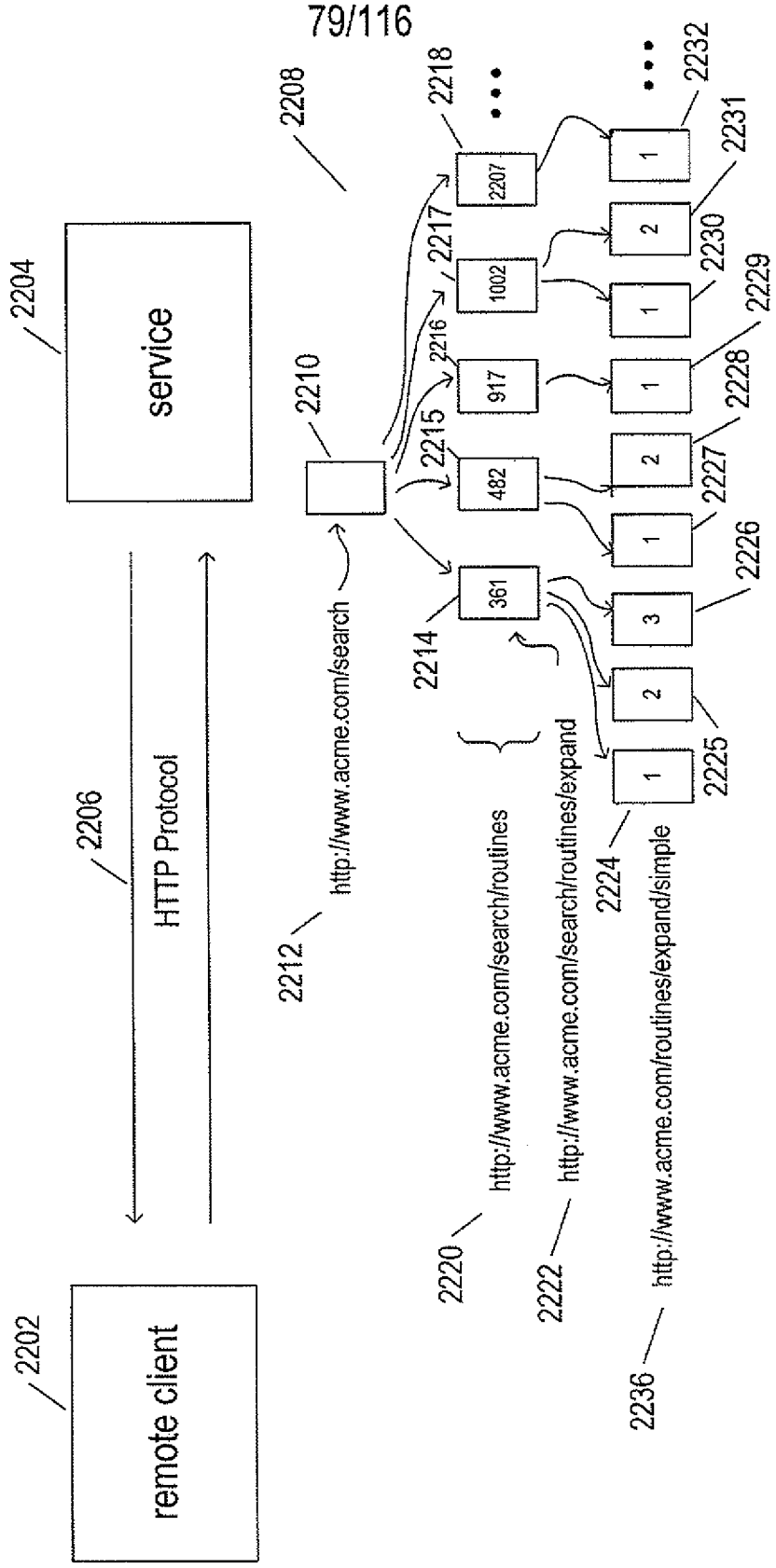

2202
remote client
2206
HTTP Protocol
2204
service
2208
2210
2212
http://www.acme.com/search
2214
361
2215
482
2216
917
2217
1002
2218
2207
2220
http://www.acme.com/search/routines
2222
http://www.acme.com/search/routines/expand
2224
http://www.acme.com/routines/expand/simple
2225
2226
2227
2228
2229
2230
2231
2232
2236

FIG. 22

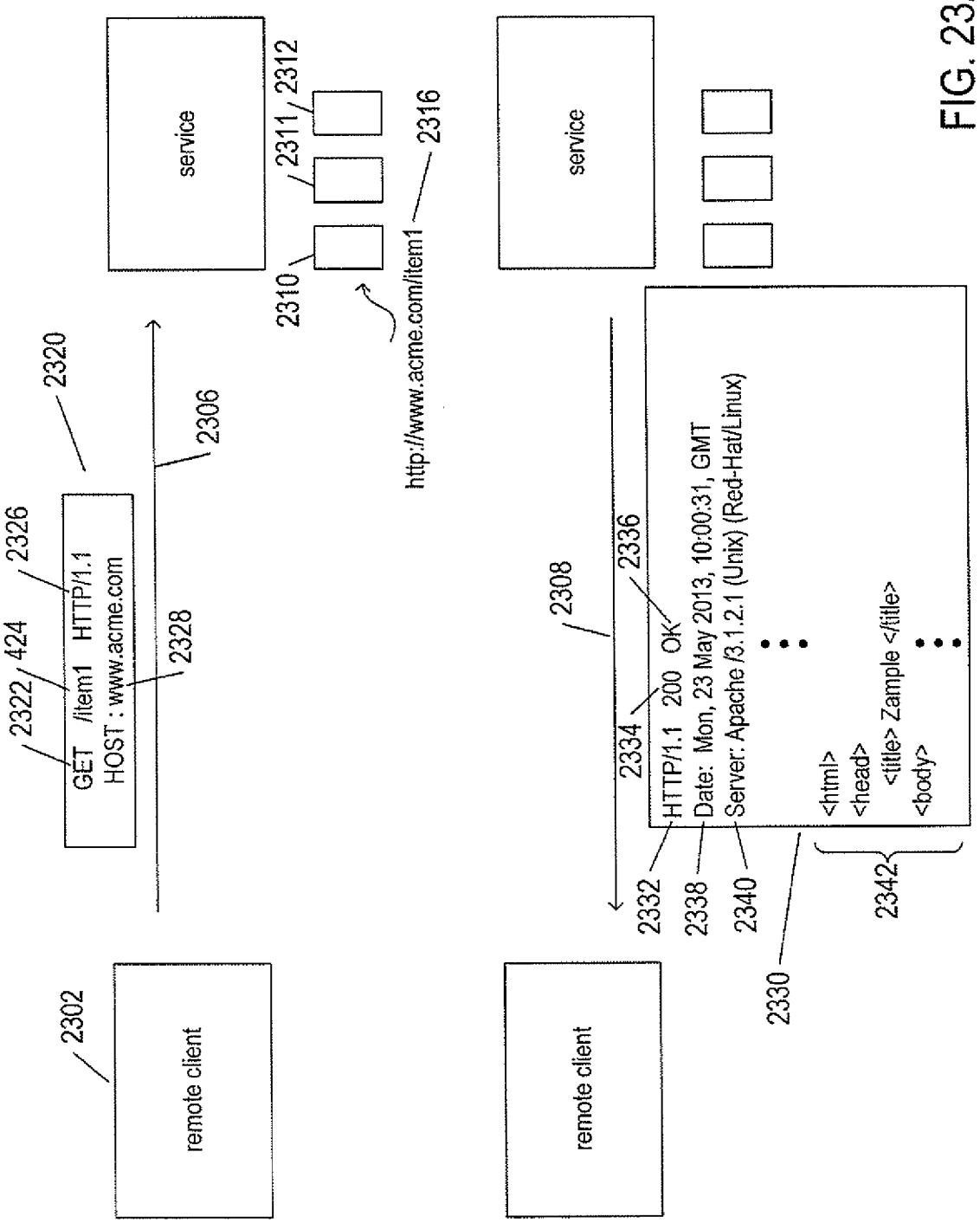

remote client
2302
service
2306
2320
2326
GET /item1 HTTP/1.1
HOST : www.acme.com
2322
424
2328
2310
2311
2312
2316
http://www.acme.com/item1
remote client
service
2308
2336
HTTP/1.1 200 OK
Date: Mon, 23 May 2013, 10:00:31, GMT
Server: Apache /3.1.2.1 (Unix) (Red-Hat/Linux)
<html>
<head>
<title> Zample </title>
<body>
2334
2332
2338
2340
2330
2342

FIG. 23A

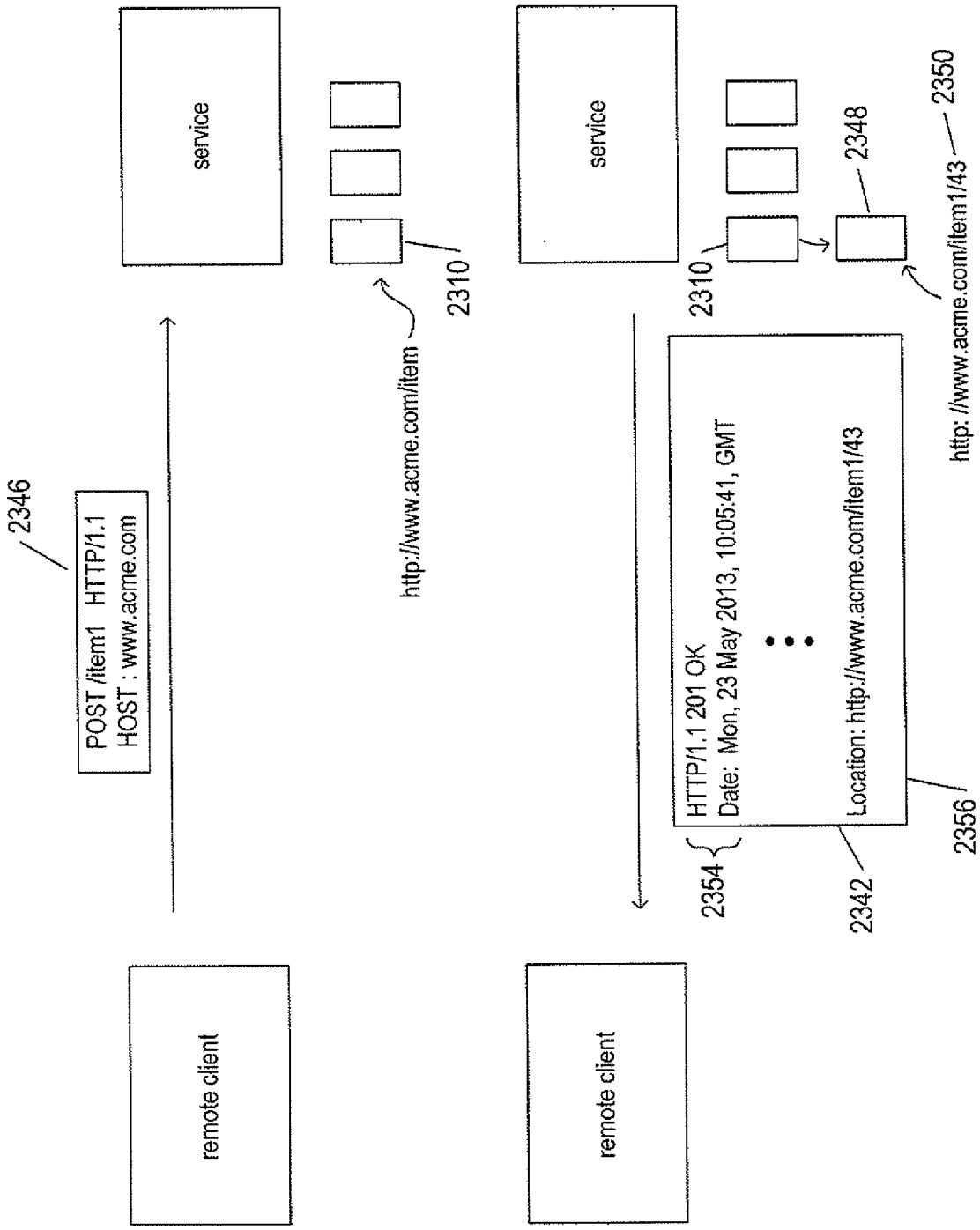

remote client
service
2346
POST /item1 HTTP/1.1
HOST : www.acme.com
http://www.acme.com/item
2310
remote client
service
2310
2348
2350
http: //www.acme.com/item1/43
HTTP/1.1 201 OK
Date: Mon, 23 May 2013, 10:05:41, GMT
Location: http://www.acme.com/item1/43
2354
2342
2356

FIG. 23B

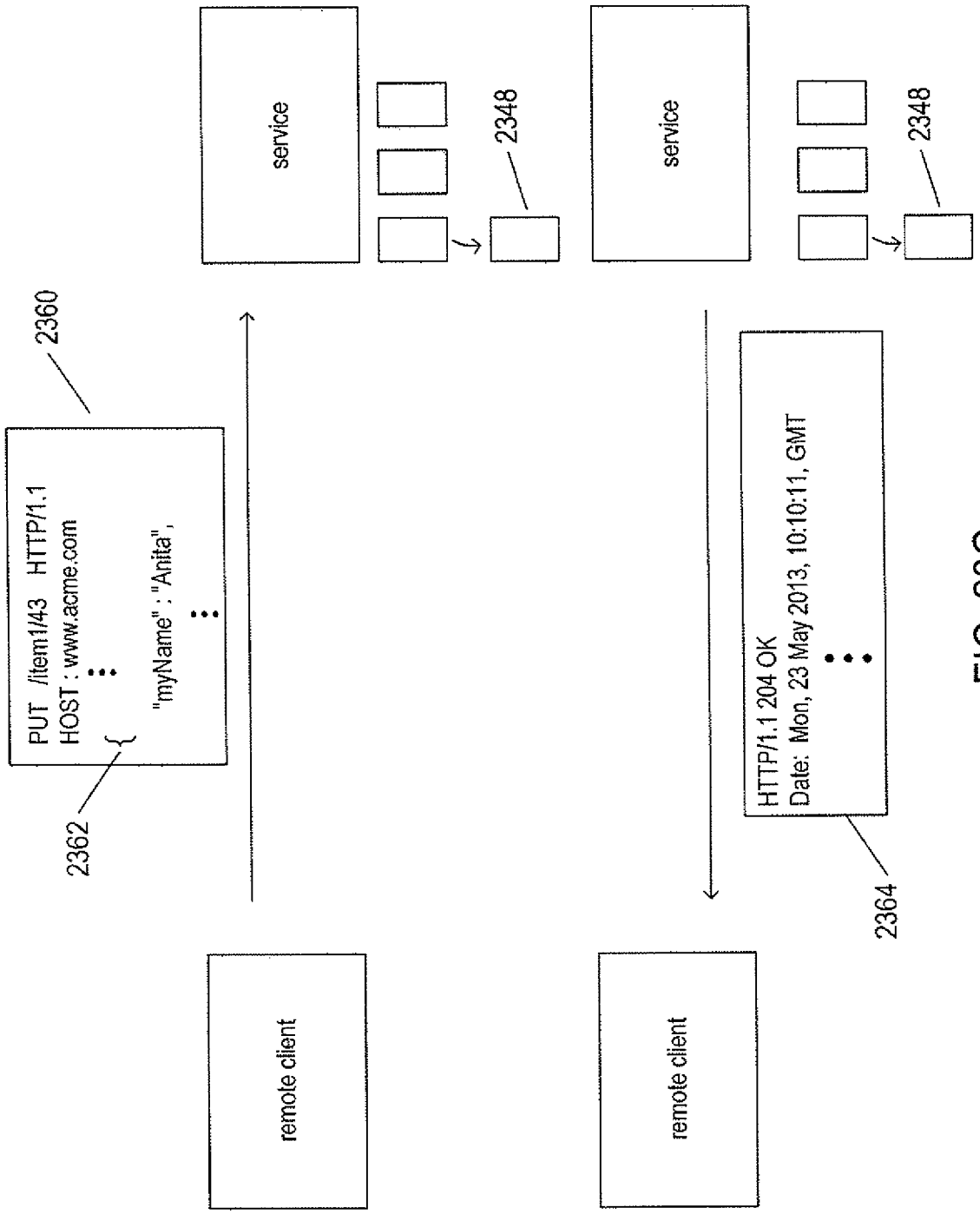

service
2348
service
2348
2360
PUT /item1/43 HTTP/1.1
HOST : www.acme.com
"myName" : "Anita",
2362
HTTP/1.1 204 OK
Date: Mon, 23 May 2013, 10:10:11, GMT
2364
remote client
remote client

FIG. 23C

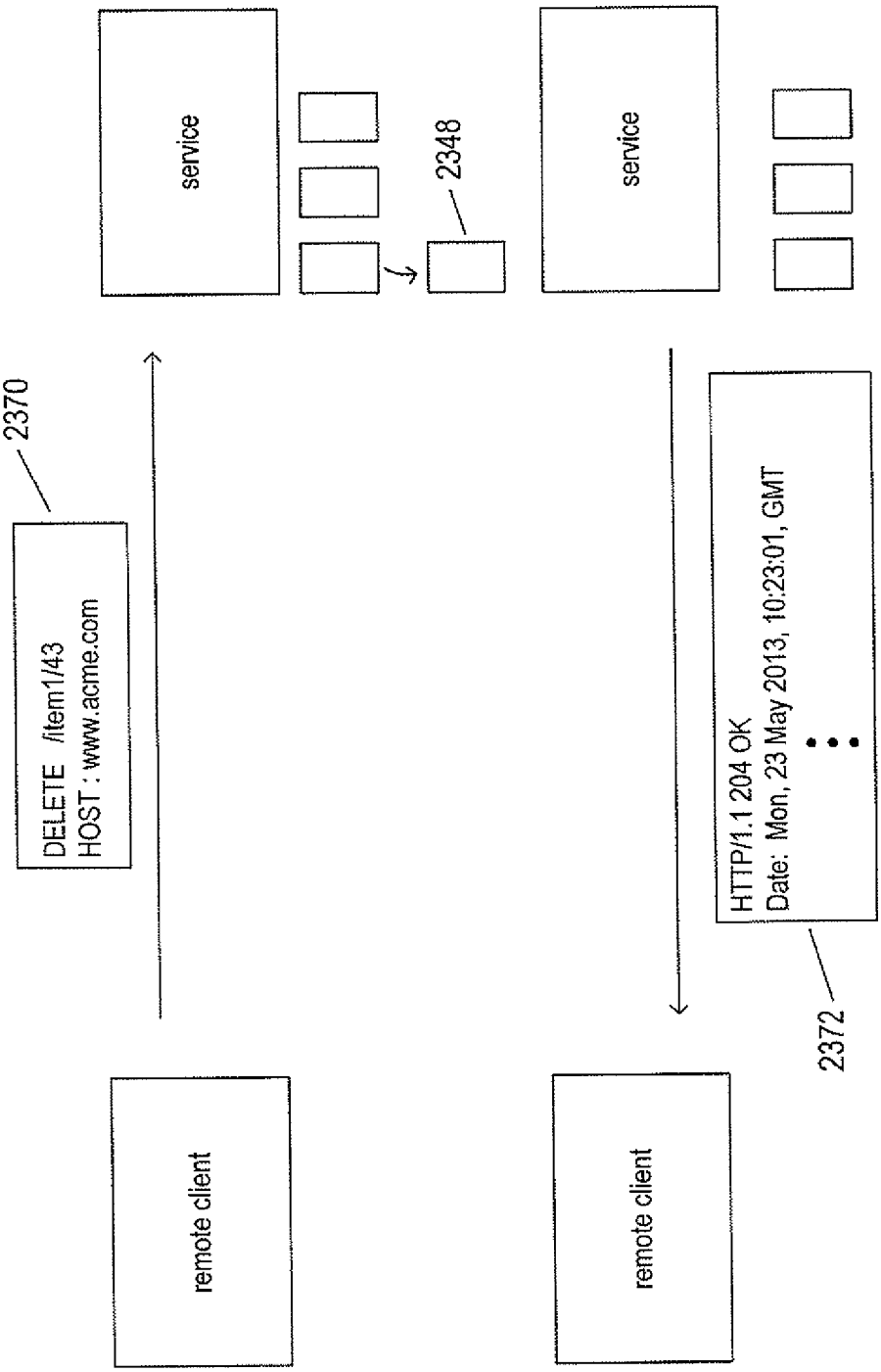

service
2348
service
2370
DELETE /item1/43
HOST : www.acme.com
HTTP/1.1 204 OK
Date: Mon, 23 May 2013, 10:23:01, GMT
2372
remote client
remote client

FIG. 23D

x	Decimal	X coordinate of the node in the canvas	
y	Decimal	Y coordinate of the node in the canvas	
z	Decimal	Z coordinate of the node in the canvas	
color	Array[Integers]	RGB color of the node	
title	String	This is the full text without crapping it.	
type	String	It could have two values, term or article	
level	Decimal	Dept of the node. The root node is at level 0, the child's of the root node has level 1, the grandchildren's 2, etc.	
source	Decimal	URL of the article. Starting with http://, can be empty""	If type = article
dinamic_field	String	Dynamic field to query when getting extended results	If type = article
document_id	String	This is the Atigeo identifier of a document	If type = article
call_id	Integer	Identifier of the call. For example, the call that gets the children's of the root term it has id 0, then for each children we made a call so it will have an identifier for each call. It the root has 4 children's, the nodes that gets this calls will have 1,2,3,4 as ids.	
visible	Boolean		
score	Decimal	This is the score/rating of the document/term	
expanded	Boolean	This value says if the node is with the tooltip expanded or not	

FIG. 24

Item	Ty[e	Description	Validations
graph			
name	String		
nodes			
node			
x [To Migrate]	Decimal	X coordinate of the node in the canvas	
y [To Migrate]	Decimal	Y coordinate of the node in the canvas	
z [To Migrate]	Decimal	Z coordinate of the node in the canvas	
color [Obsolet]	Array[Inte-gers]	RGB color code of the node	
title	String	This is the full text without crapping it.	
type	String	It could have two values, term or article	
level	Decimal	Dept of the node. The root node is at level 0, the child's of the root node has level 1, the grandchildren's 2, etc.	
source	Decimal	URL of the article. Starting with http://, can be empty""	If type = article
dinamic_field	String	Dynamic field to query when getting extended results	If type = article
document_id	String	This is the Atigeo identifier of a document	if type = article
call_id	Integer	Identifier of the call. For example, the call that gets the children's of the root term it has id 0, then for each children we made a call so it will have an identifier for each call. If the root has 4 children's, the nodes that gets this calls will have 1,2,3,4 as ids.	
visible	Boolean		
score	Decimal	This is the score/rating of the document/ term	
expanded	Boolean	This value says if the node is with the tooltip expanded or not	
shape_index [To be added]	Integer	Index of the node in the shapes array. The first node starts at index 0.	
group_that_belongs [To be added]	String	The name of the group that belongs. For example: Hidden, Default or the name of the created group	

FIG. 25A

bonds			
bond			
id_from	Integer	"from" node	
id_to	Integer	"to" node	
is_linked	Integer		
visible	Boolean		
shapes			
shape3d			
x	Decimal	X coordinate of the sphere. It is different from the node X position. For example, the original x position is in the center of the sphere but this x position is the left top coordinate of the image.	
y	Decimal	Y coordinate of the sphere. It is different from the node Y position. For example, the original y position is in the center of the sphere but this y position is the left top coordinates of the image.	
z	Decimal	Z coordinate value	
radius	Decimal	This value indicates the radium of the sphere from a graphic standpoint	
image_radius	Decimal	When the node is a representative image (in the case of Red/Blue app), this value indicates the radius of the image	
text	String		
type	String	type of the node. It could have two values, term or article	
source	String	Here we have the URL of the article. Starting with http://, can be empty ""	
position	Integer	This is the identifier of the node	
hide	Boolean	This value indicates if it is hidden or not	
originalx	Decimal	This value has the original X coordinate (because is different fro the graphical x coordinate). It matches with the x of the node	

FIG. 25B

originaly	Decimal	This value has the original Y coordinate (because is different from the graphical y coordinate). It matches with the y of the node	
originalz	Decimal	This is the original z value.	
threedshape	Boolean	This value indicates if it is a 3D or 2D sphere	
tooltip			
x	Decimal	x coordinate of the tooltip	
y	Decimal	y coordinate of the tooltip	
z	Decimal	z coordinate of the tooltip	
width	Integer	Width of the tooltip	
height	Integer	Height of the tooltip	
long_text	Array[String]	Array of text lines to show in the tooltip. This text is shown in different lines when expanded.	
short_text	String	This is the short text ending with ellipse. This text is shown when the tooltip is collapsed	
expanded	Boolean	This value tells if it is expanded or not	
sizeFactor	Decimal	This is the factor that we apply to the node size. It is used for zoom in or zoom out.	
x_action_zone	Decimal	This is the x position of the action zone	
y_action_zone	Decimal	This is the y position of the action zone	
w_action_zone	Decimal	This is the width of the action zone	
h_action_zone	Decimal	This is the height of the action zone	
w_button_zone	Decimal	This is the width of the tooltip buttons	
h_button_zone	Decimal	This is the height of the tooltip buttons	
distance_between_buttons	Decimal	This is the size between buttons	

FIG. 25C

x1_plus_node	Decimal	This is the left top x position of the plus button	
x2_plus_node	Decimal	This is the right bottom x position of the plus button	
y1_plus_node	Decimal	This is the left top y position of the plus button	
y2_plus_node	Decimal	This is the right bottom y position of the plus button	
x1_minus_node	Decimal	This is the left top x position of the minus button	
x2_minus_node	Decimal	This is the right bottom x position of the minus button	
y1_minus_node	Decimal	This is the left top y position of the minus button	
y2_minus_node	Decimal	This is the right bottom y position of the minus button	
x1_open_node	Decimal	This is the left top x position of the open button	
x2_open_node	Decimal	This is the right bottom x position of the open button	
y1_open_node	Decimal	This is the left top y position of the open button	
y2_open_node	Decimal	This is the right bottom y position of the open button	
Shape2d			
x	Decimal	X coordinate of the sphere. It is different from the node X position. For example, the original x position is in the center of the sphere but this x position is the left top coordinate of the image.	
y	Decimal	Y coordinate of the sphere. It is different from the node Y position. For example, the original y position is in the center of the sphere but this y position is the left top coordinate of the image.	
z	Decimal	Z coordinate value	
radius	Decimal	This value indicates the radius of the sphere from a graphic standpoint	
image_radius	Decimal	When the node is a representative image (in the case of Red/Blue app), this value indicates the radius of the image	

FIG. 25D

text	String		
type	String	Type of the node. It could have two values, term or article	
source	String	Here we have the URL of the article. Starting with http://, can be empty ""	
position	Integer	This is the identifier of the node	
hide	Boolean	This value indicates if it is hidden or not	
twodx	Decimal	This value has the 2D x coordinate	
twody	Decimal	This value has the 2D y coordinate	
threedshape	Boolean	This value indicates if it is a 3D or 2D sphere	
tooltip			
x	Decimal	x coordinate of the tooltip	
y	Decimal	y coordinate of the tooltip	
z	Decimal	z coordinate of the tooltip	
width	Integer	Width of the tooltip	
height	Integer	Height of the tooltip	
long_text	Array[String]	Array of text lines to show in the tooltip. This text is shown in different lines when expanded.	
short_text	String	This is the short text ending with ellipse. This text is shown when the tooltip is collapsed	
expanded	Boolean	This value tells if it is expanded or not	
sizeFactor	Decimal	This is the factor that we apply to the node size. It is used for zoom in or zoom out.	
x_action_zone	Decimal	This is the x position of the action zone	
y_action_zone	Decimal	This is the y position of the action zone	

FIG. 25E

w_action_zone	Decimal	This is the width of the action zone	
h_action_zone	Decimal	This is the height of the action zone	
w_button_zone	Decimal	This is the width of the tooltip buttons	
h_button_zone	Decimal	This is the height of the tooltip buttons	
distance_between_buttons	Decimal	This is the size between buttons	
x1_plus_node	Decimal	This is the left top x position of the plus button	
x2_plus_node	Decimal	This is the right bottom x position of the plus button	
y1_plus_node	Decimal	This is the left top y position of the plus button	
y2_plus_node	Decimal	This is the right bottom y position of the plus botton	
x1_minus_node	Decimal	This is the left top x position of the minus button	
x2_minus_node	Decimal	This is the right bottom x position of the minus button	
y1_minus_node	Decimal	This is the left top y position of the minus button	
y2_minus_node	Decimal	This is the right bottom y position of the minus button	
x1_open_node	Decimal	This is the left top x position of the open button	
x2_open_node	Decimal	This is the right bottom x position of the open button	
y1_open_node	Decimal	This is the left top y position of the open button	
y2_open_node	Decimal	This is the right bottom y position of the open button	
groups			
group		Defaults groups (hidden, default, etc)	
id	Decimal	This value is the id of the group	
name	String	This is the name of the group	
visible	Boolean	This value indicates if it is visible or not	
term_color	Array[Integer]	Color of the terms on this group	
document_color	Array[Integer]	Color of the document on this group	

FIG. 25F

node_group_association			
group_list_association		User groups	
id	Decimal	This value is the id of the group	
name	String	This is the name of the group	
visible	Boolean	This value indicates if it is visible or not	
decoration	Integer	The decoration id	
decoration_list			
id			
type		The type of the decoration. It could be fill or another type	
term_color	Array[Integer]	Color of the terms on this group	
document_color	Array[Integer]	Color of the document on this group	
group_count	Integer		
settings	JSON		
helper [To be removed]			
searchResult [To be removed]	Integer		
recursive_hash_global	Array[Array[Integer]]	This is the internal structure that is used to identify links between nodes. Each row has an array of child id nodes.	
is3d	Boolean	This indicates if the saved graph is 3D or not	
recursiveHash [To be removed]	Array[Array[Integer]]	Same as recursive_hash_global	
environment	String	This is the environment/indexURL value	
selected_env	String	This is the selected environment/ indexURL	
selected_net	String	This is the CRN selected value	
selected_exp	String	This is the DE selected value	
term_radius_3d	Decimal	When zoom in or zoom out this values are modified so we save it. This is the graphical radius scale of a shape	

FIG. 25G

politics_image_height_3d	Decimal		
politics_image_width_3d	Decimal		
image_height_3d	Decimal		
image_width_3d	Decimal		
zback_3d	Integer	Minimum Z value of the camera when 3D activated	
zrange_3d	Integer	Range value between z back and z front 3D activated	
term_radius_2d	Decimal		
politics_image_height_2d	Decimal		
politics_image_width_2d	Decimal		
image_height_2d	Decimal		
image_width_2d	Decimal		
zback_2d	Integer	Minimum Z value of the camera when 2D activated	
zrange_2d	Integer	Range value between z back and z front when 2D activated	

FIG. 25H

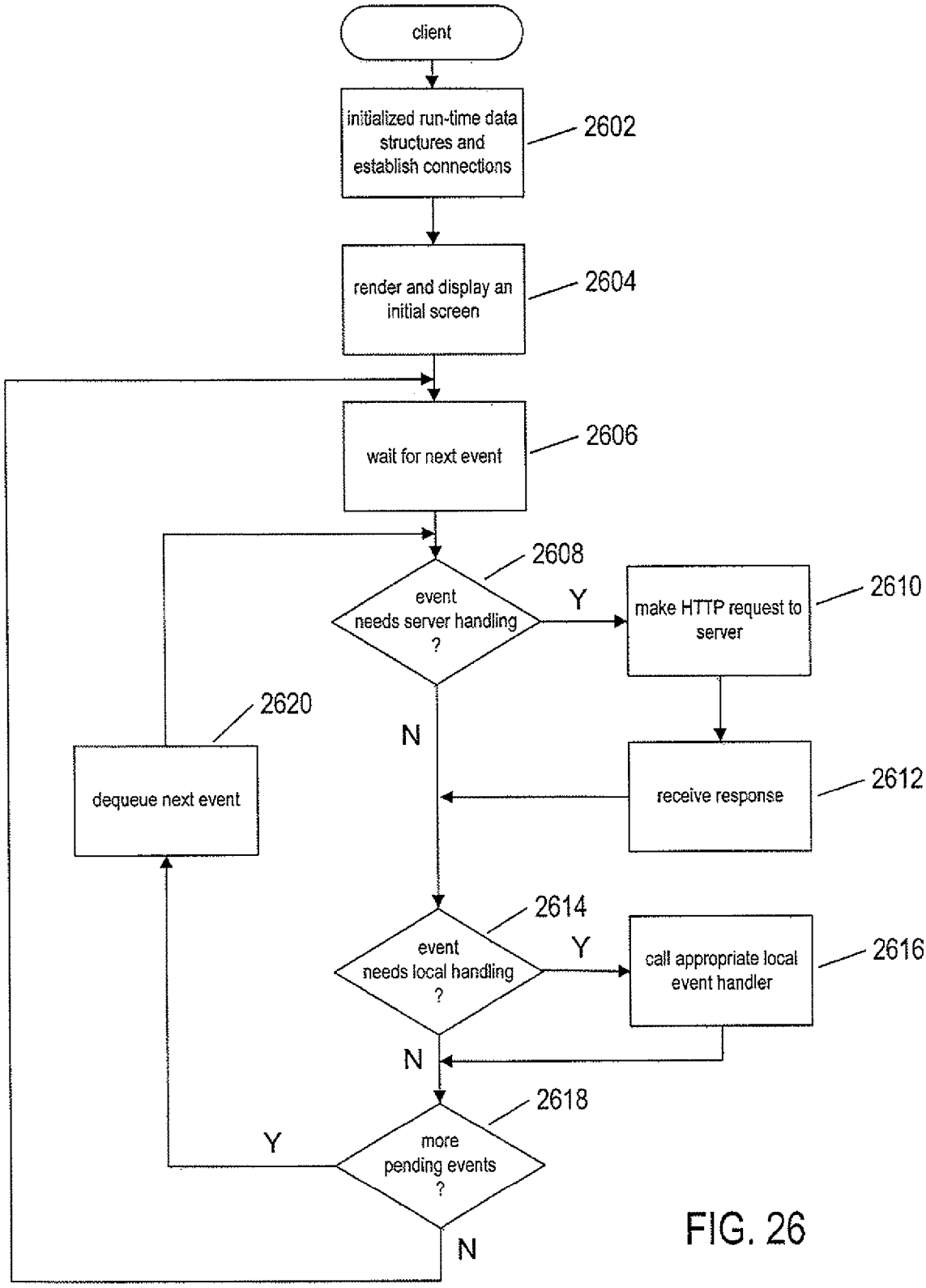

client
initialized run-time data structures and establish connections
2602
render and display an initial screen
2604
wait for next event
2606
2608
event needs server handling ?
Y
make HTTP request to server
2610
N
receive response
2612
2620
dequeue next event
2614
event needs local handling ?
Y
call appropriate local event handler
2616
N
2618
more pending events ?
Y
N

FIG. 26

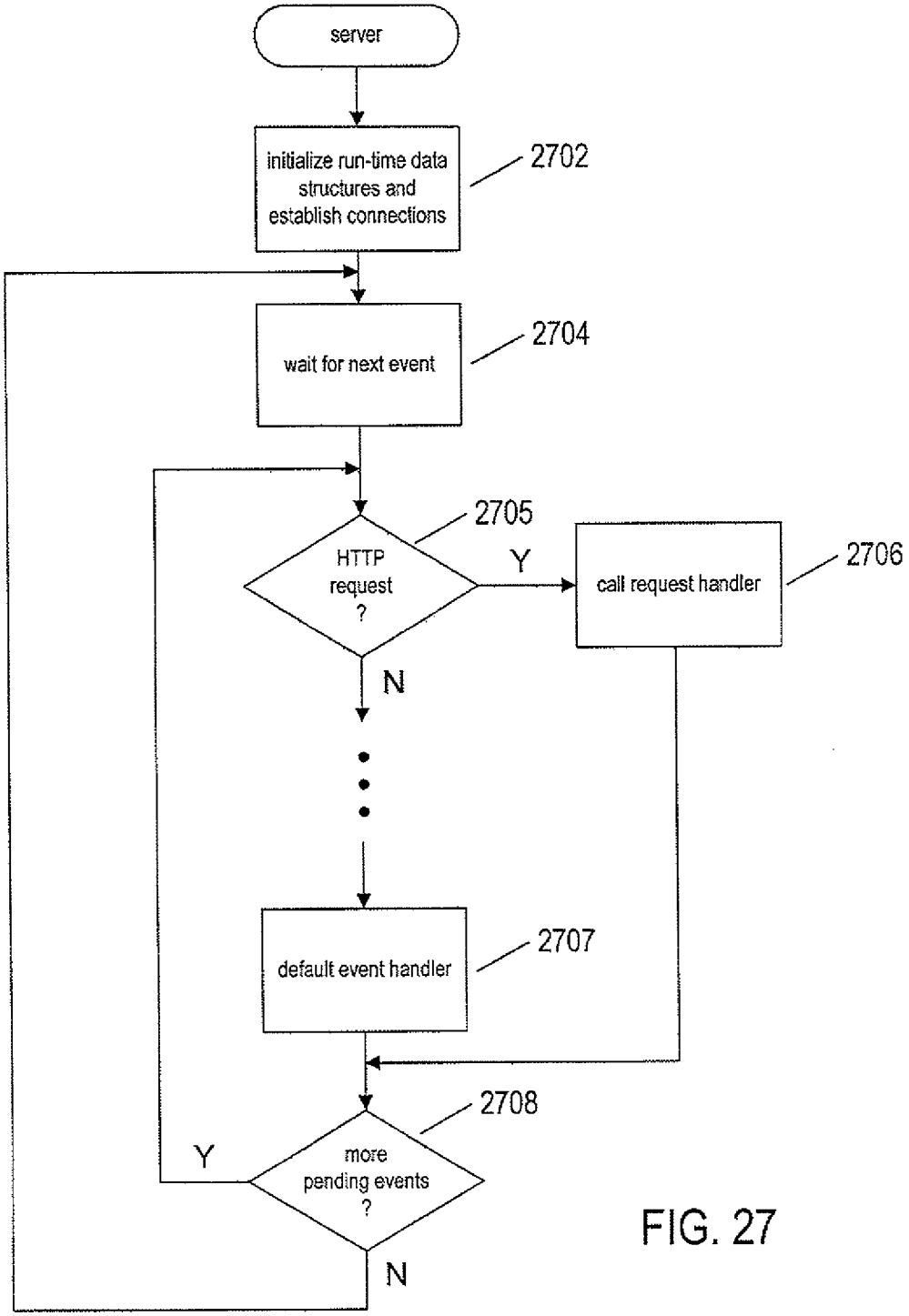
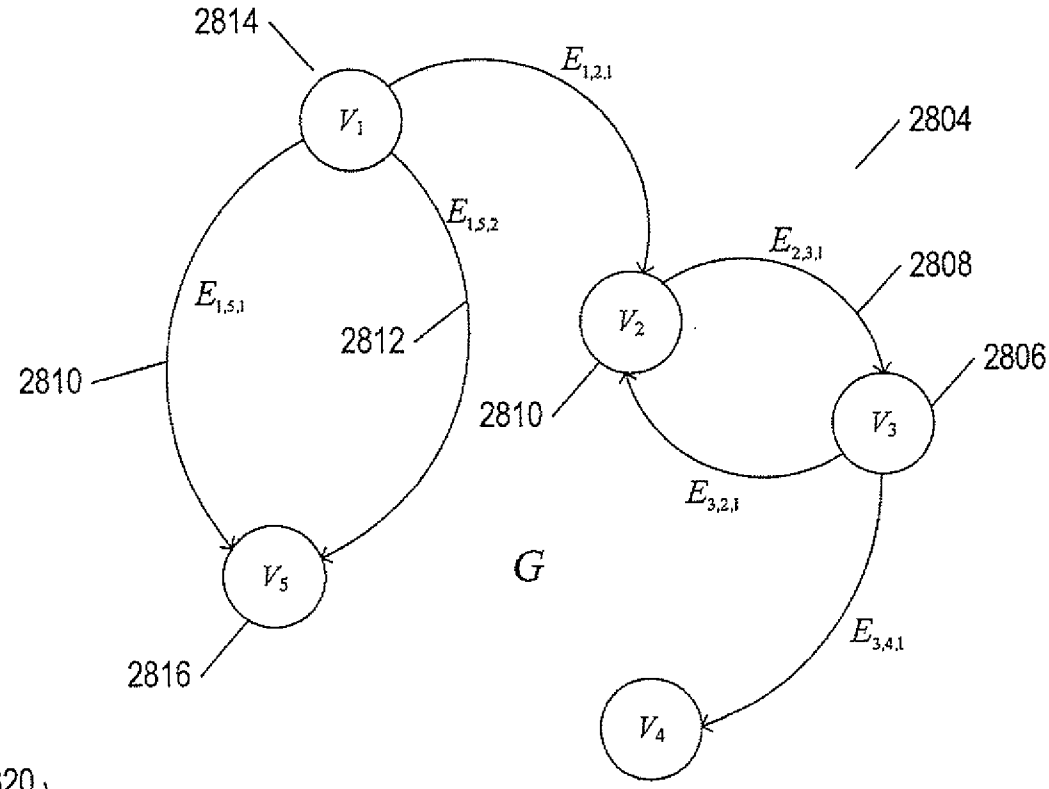

server
initialize run-time data structures and establish connections
2702
wait for next event
2704
2705
HTTP request ?
Y
call request handler
2706
N
default event handler
2707
2708
more pending events ?
Y
N

FIG. 27

$$G = \{\{V_1, V_2, V_3, V_4, V_5\}\{E_{1,5,1}, E_{1,5,2}, E_{1,2,1}, E_{2,3,1}, E_{3,2,1}, E_{3,4,1}\}\}$$
$$= \{\{\text{nodes}\}, \{\text{edges}\}\}$$
$$= \{\mathbf{V}, \mathbf{E}\}$$

2802



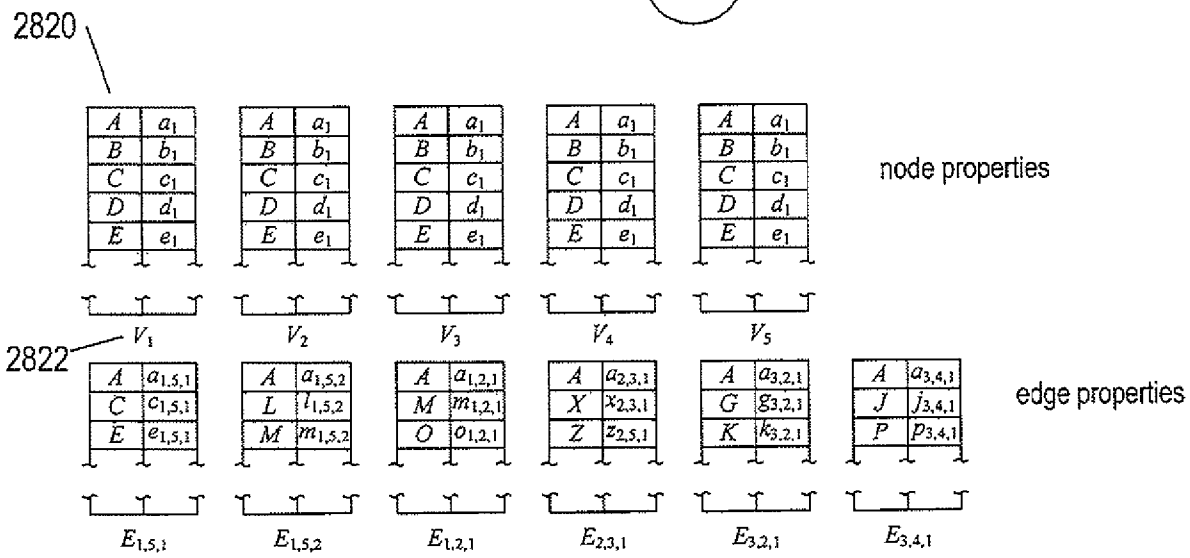


FIG. 28

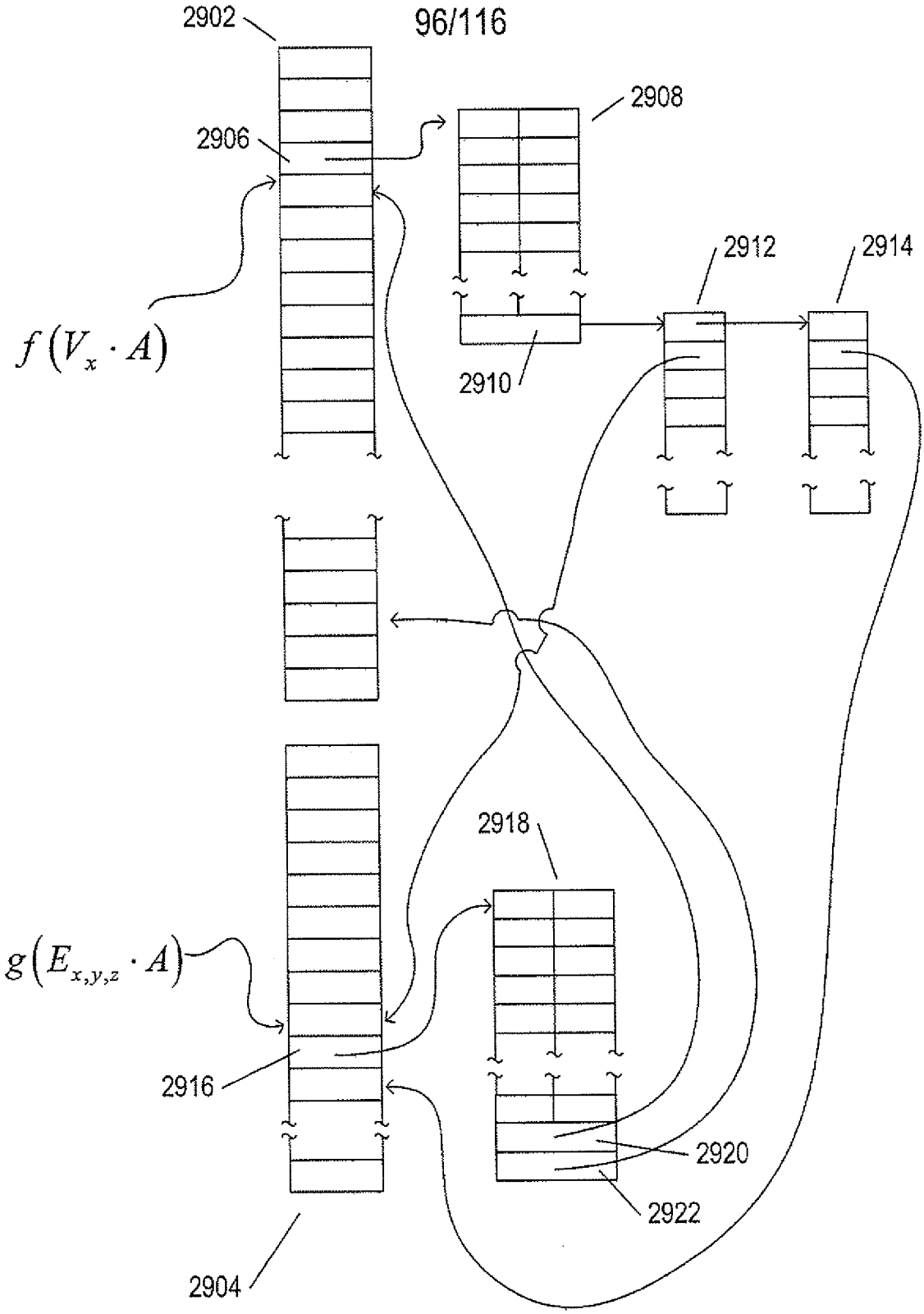

2902
2906
2908
2910
2912
2914
$f(V_x \cdot A)$
2918
$g(E_{x,y,z} \cdot A)$
2916
2920
2922
2904

FIG. 29

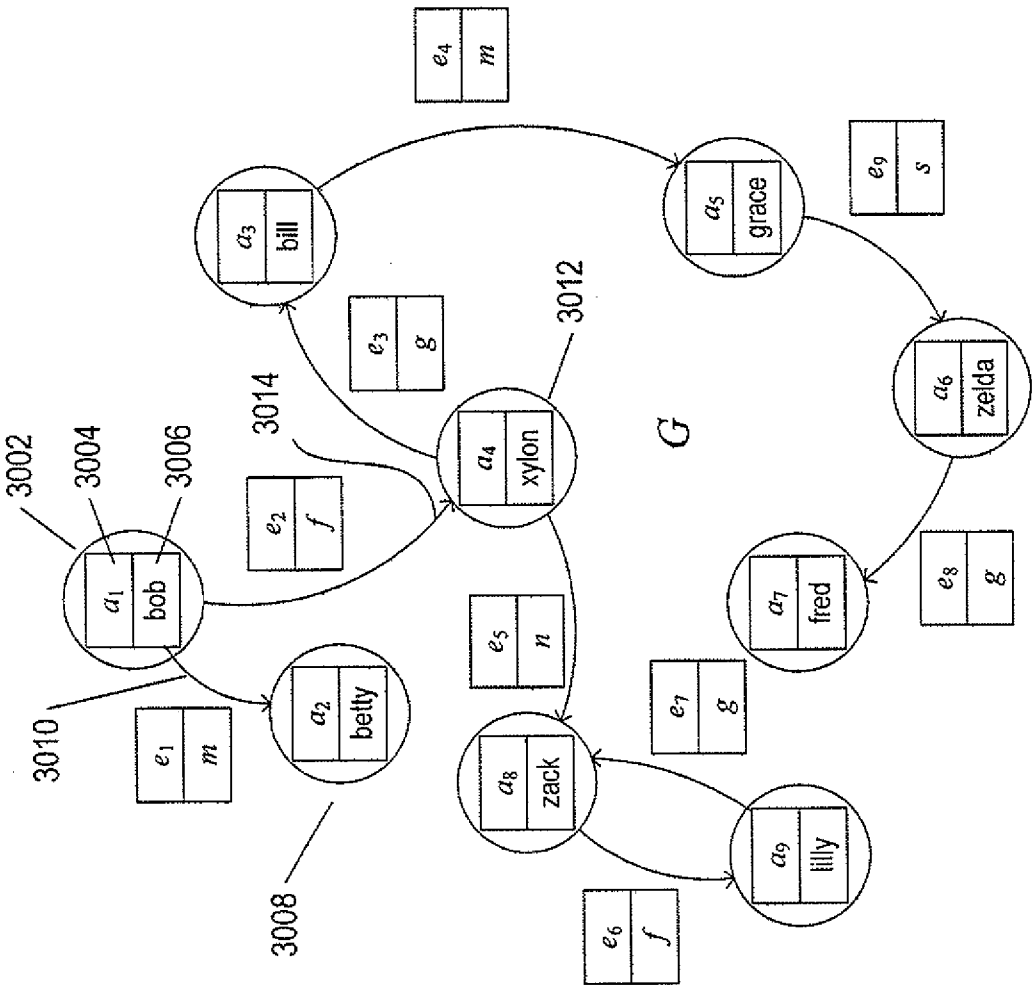

e_4 m
a_3 bill
a_5 grace
e_9 s
e_3 g
3012
3014
a_6 zelda
a_4 xylon
G
3002
3004
3006
e_2 f
a_1 bob
a_7 fred
e_8 g
e_5 n
e_7 g
3010
e_1 m
a_2 betty
a_8 zack
a_9 lilly
3008
e_6 f

FIG. 30

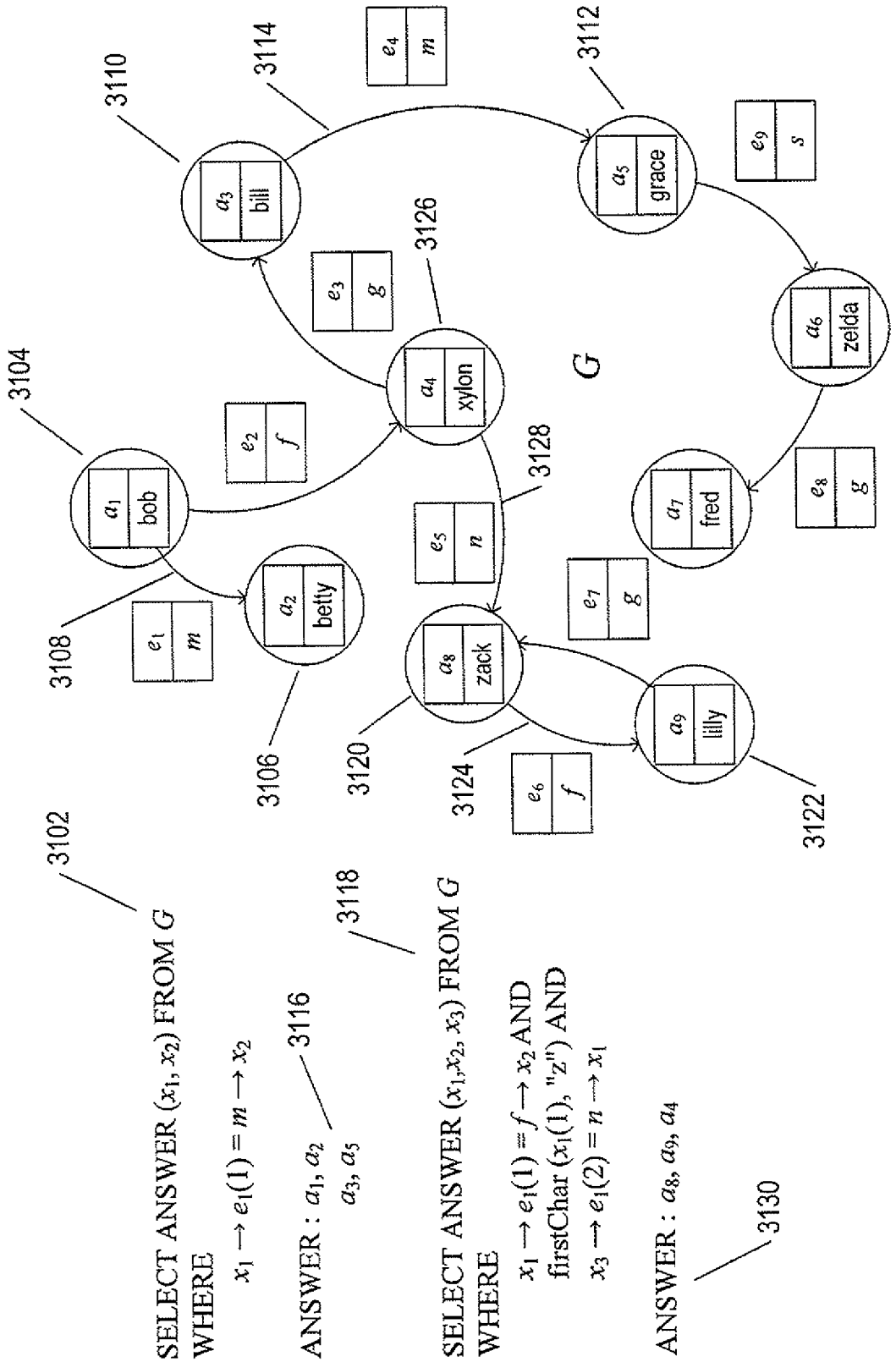

3102
SELECT ANSWER (x_1, x_2) FROM G
WHERE
$x_1 \longrightarrow e_1(1) = m \longrightarrow x_2$
3116
ANSWER : a_1, a_2
a_3, a_5
3118
SELECT ANSWER (x_1, x_2, x_3) FROM G
WHERE
$x_1 \longrightarrow e_1(1) = f \longrightarrow x_2$ AND
firstChar $(x_1(1),$ "z") AND
$x_3 \longrightarrow e_1(2) = n \longrightarrow x_1$
3130
ANSWER : a_8, a_9, a_4
3104
3110
3112
3114
3126
3128
3106
3108
3120
3122
3124
a_1 bob
a_2 betty
a_3 bill
a_4 xylon
a_5 grace
a_6 zelda
a_7 fred
a_8 zack
a_9 lilly
e_1 m
e_2 f
e_3 g
e_4 m
e_5 n
e_6 f
e_7 g
e_8 g
e_9 s
G

FIG. 31

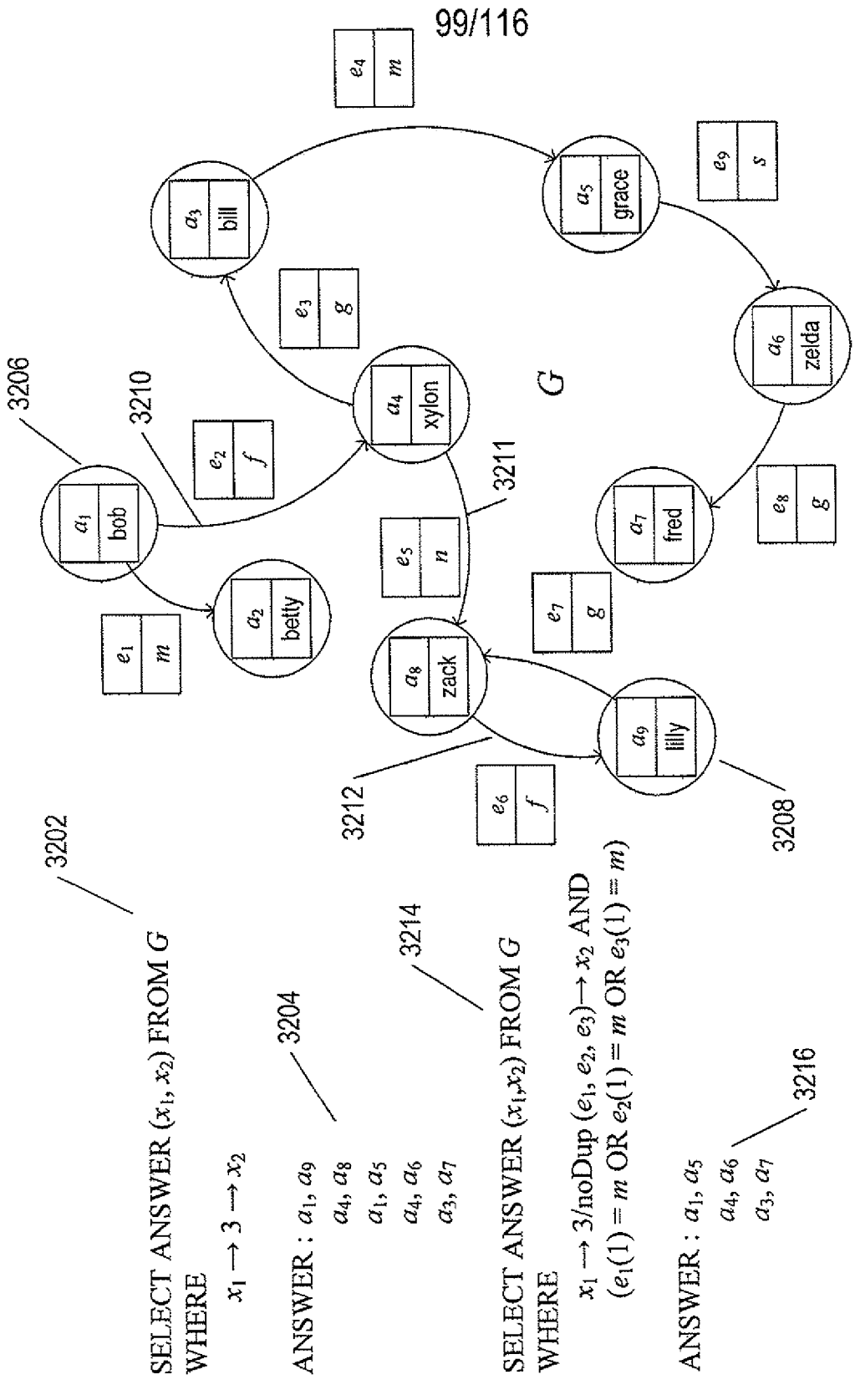

3202
SELECT ANSWER (x_1, x_2) FROM G
WHERE
$x_1 \longrightarrow 3 \longrightarrow x_2$
ANSWER : a_1, a_9
a_4, a_8
a_1, a_5
a_4, a_6
a_3, a_7
3204
3214
SELECT ANSWER (x_1, x_2) FROM G
WHERE
$x_1 \longrightarrow 3/\text{noDup}\ (e_1, e_2, e_3) \longrightarrow x_2$ AND
$(e_1(1) = m$ OR $e_2(1) = m$ OR $e_3(1) = m)$
ANSWER : a_1, a_5
a_4, a_6
a_3, a_7
3216
3206
3210
3211
3212
3208
G
a_1 bob
a_2 betty
a_3 bill
a_4 xylon
a_5 grace
a_6 zelda
a_7 fred
a_8 zack
a_9 lilly
e_1 m
e_2 f
e_3 g
e_4 m
e_5 n
e_6 f
e_7 g
e_8 g
e_9 s

FIG. 32

SELECT ANSWER (x_1) FROM G 3302
WHERE
$v_1 \rightarrow 3/\text{noDup}\ (e_1, e_2, e_3) \rightarrow v_2$ AND
x_1 in (e_1, e_2) AND
$x_1(1) = f$ AND
($v_1(1)$ = "bob") OR ($v_1(1)$ = "bill")

ANSWER : e_2 3306
e_2

3308

SELECT ANSWER (x_1, count (x_2)) FROM G
WHERE
$x_1 \rightarrow */\text{noDup} \rightarrow x_2$ AND
$x_1(1)$ = "xylon"

ANSWER : a_4, 6

SELECT ANSWER (x_1, min(#)) FROM G
WHERE
$x_1 \rightarrow \# \rightarrow x_2$ AND
$x_1(1)$ = "xylon" AND
$x_2(1)$ = "zack"

3310

ANSWER : a_4, 1

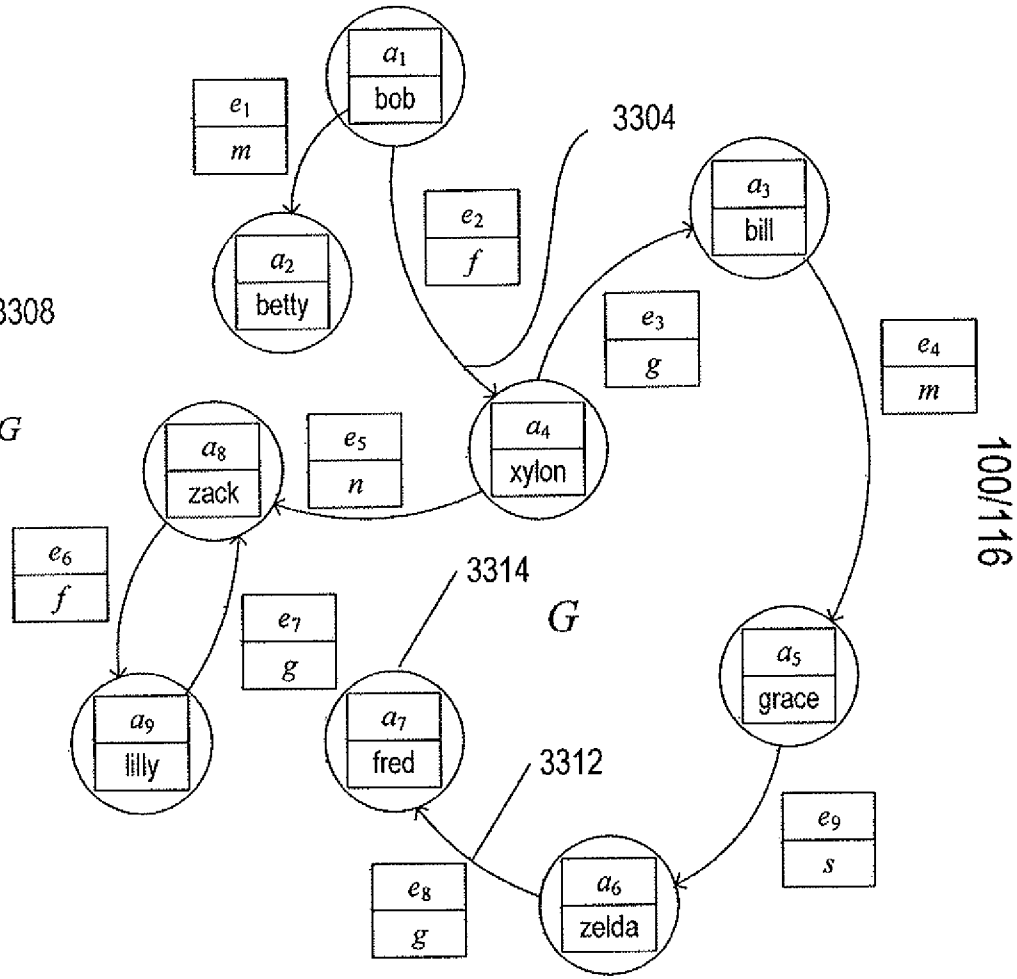


FIG. 33

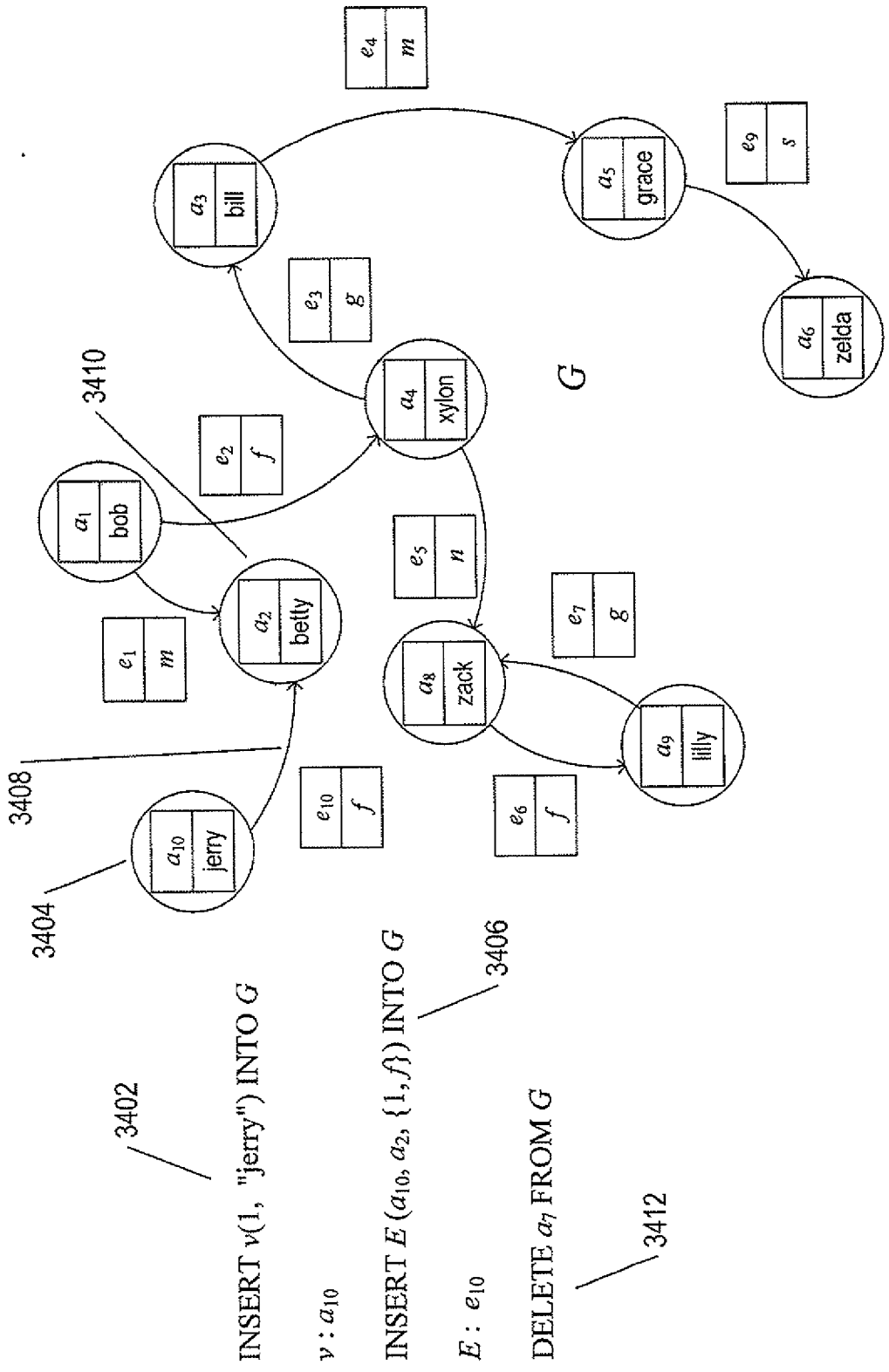

INSERT v(1, "jerry") INTO G
v : a10
INSERT E (a10, a2, {1, f}) INTO G
E : e10
DELETE a7 FROM G
3402
3404
3406
3408
3410
3412
G
a1 bob
a2 betty
a3 bill
a4 xylon
a5 grace
a6 zelda
a8 zack
a9 lilly
a10 jerry
e1 m
e2 f
e3 g
e4 m
e5 n
e6 f
e7 g
e9 s
e10 f

FIG. 34

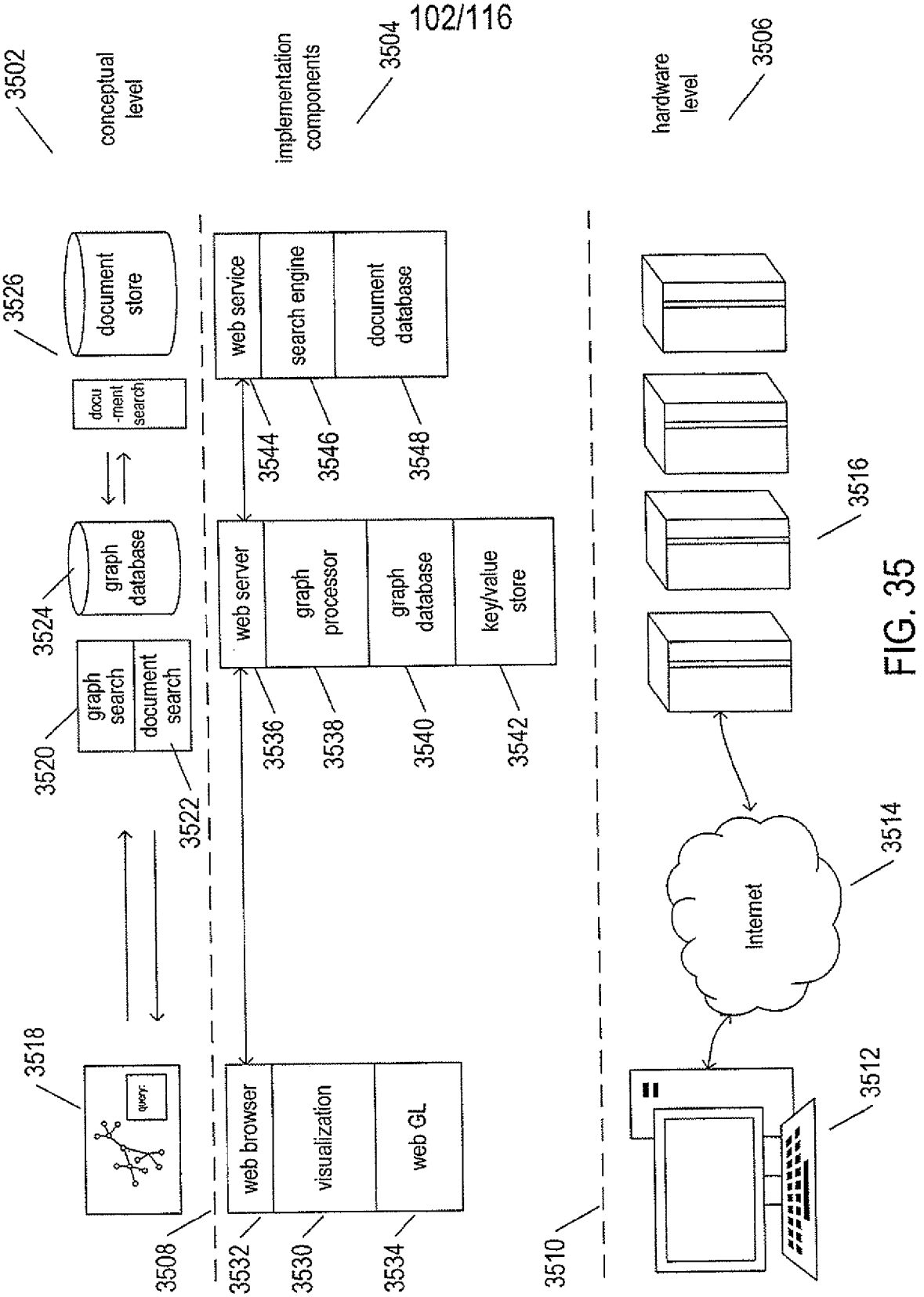

3502
conceptual level
implementation components
3504
hardware level
3506
3526
document store
docu-ment search
web service
search engine
document database
3544
3546
3548
3524
graph database
web server
graph processor
graph database
key/value store
3520
graph search
document search
3522
3536
3538
3540
3542
3516
3518
query:
web browser
visualization
web GL
3508
3532
3530
3534
3510
Internet
3514
3512

FIG. 35

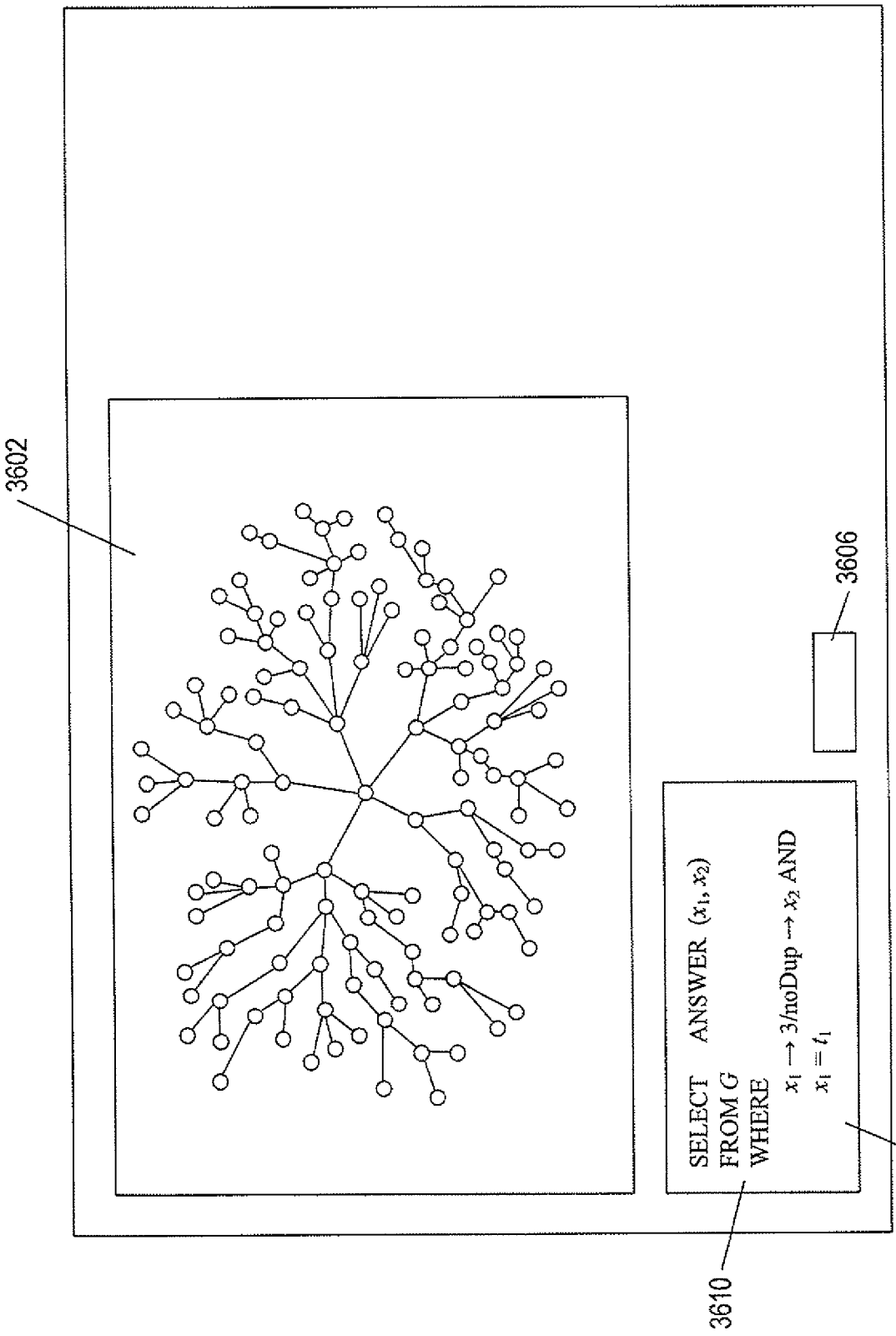

3602
3606
SELECT ANSWER (x_1, x_2)
FROM G
WHERE
$x_1 \rightarrow 3/noDup \rightarrow x_2$ AND
$x_1 = t_1$
3604
3610

FIG. 36A

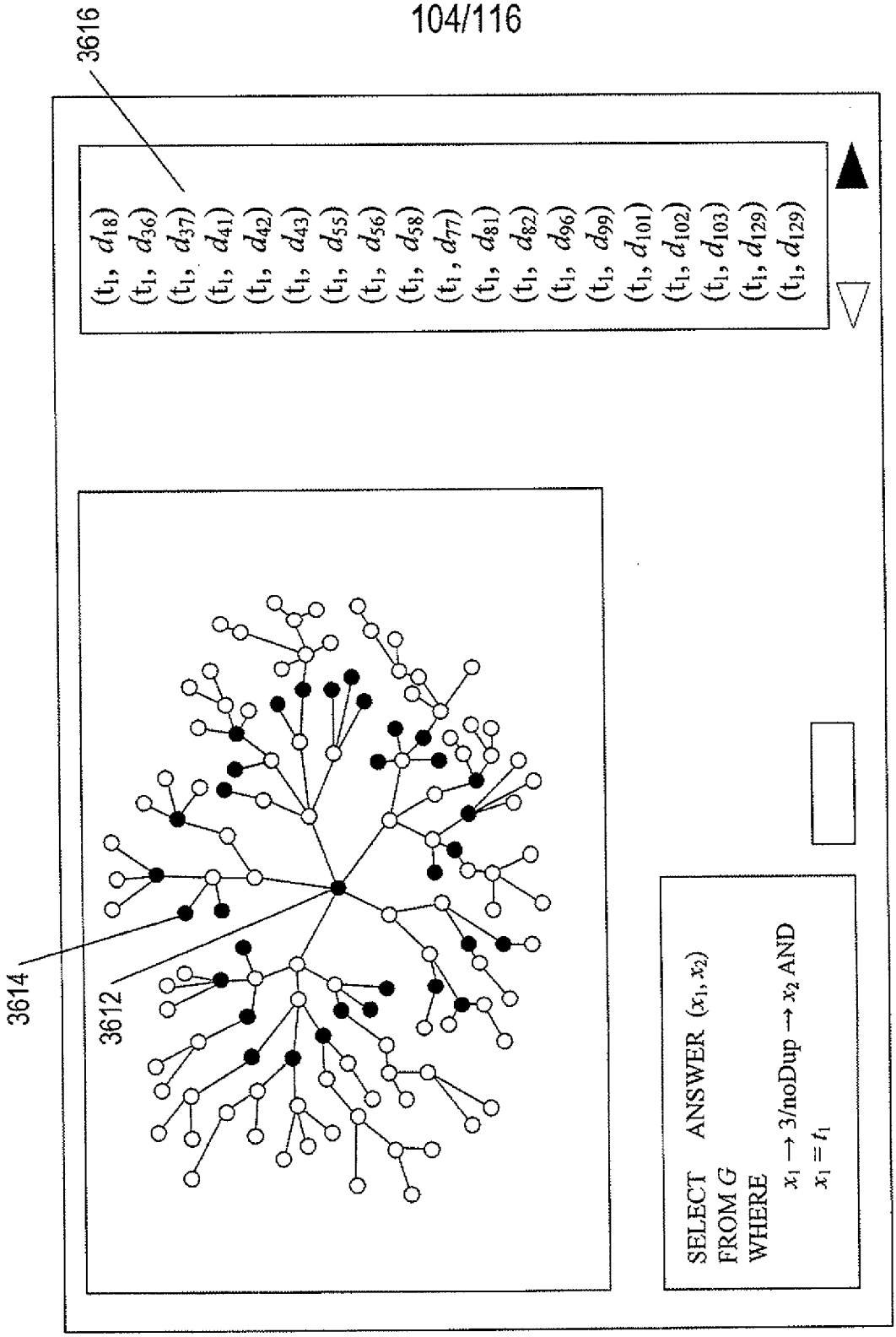

3616
(t1, d18)
(t1, d36)
(t1, d37)
(t1, d41)
(t1, d42)
(t1, d43)
(t1, d55)
(t1, d56)
(t1, d58)
(t1, d77)
(t1, d81)
(t1, d82)
(t1, d96)
(t1, d99)
(t1, d101)
(t1, d102)
(t1, d103)
(t1, d129)
(t1, d129)
3614
3612
SELECT ANSWER (x1, x2)
FROM G
WHERE
x1 → 3/noDup → x2 AND
x1 = t1

FIG. 36B

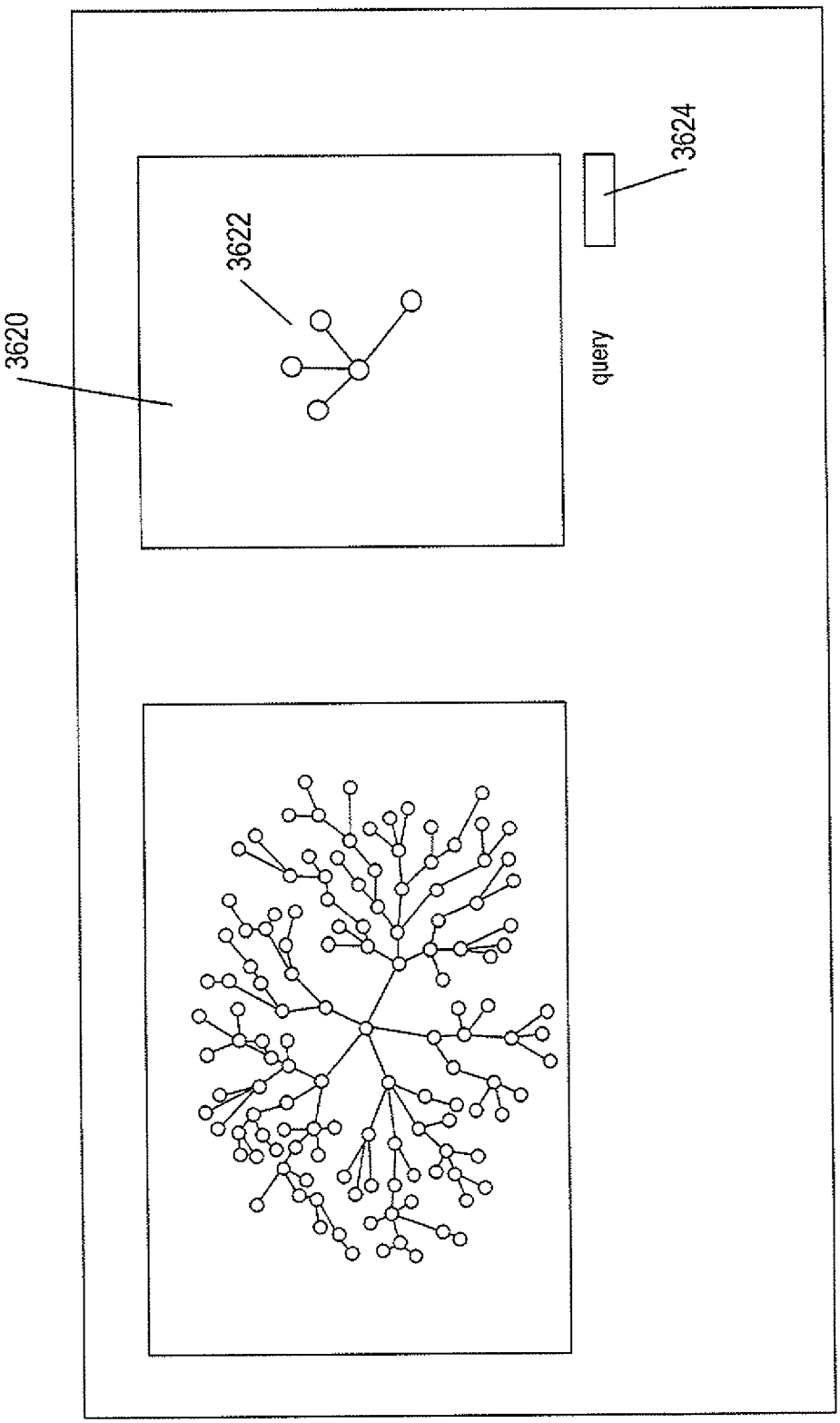


FIG. 37A

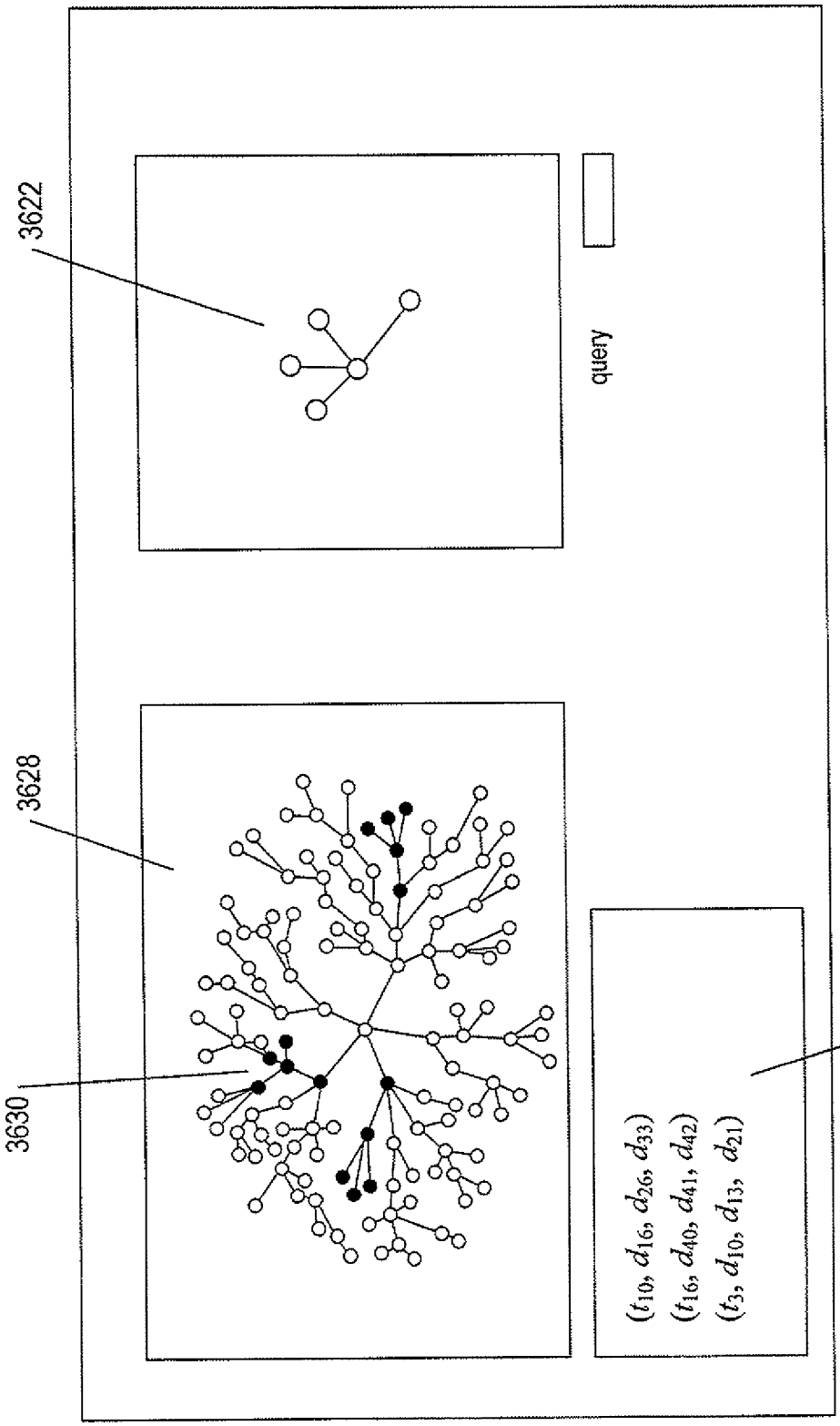
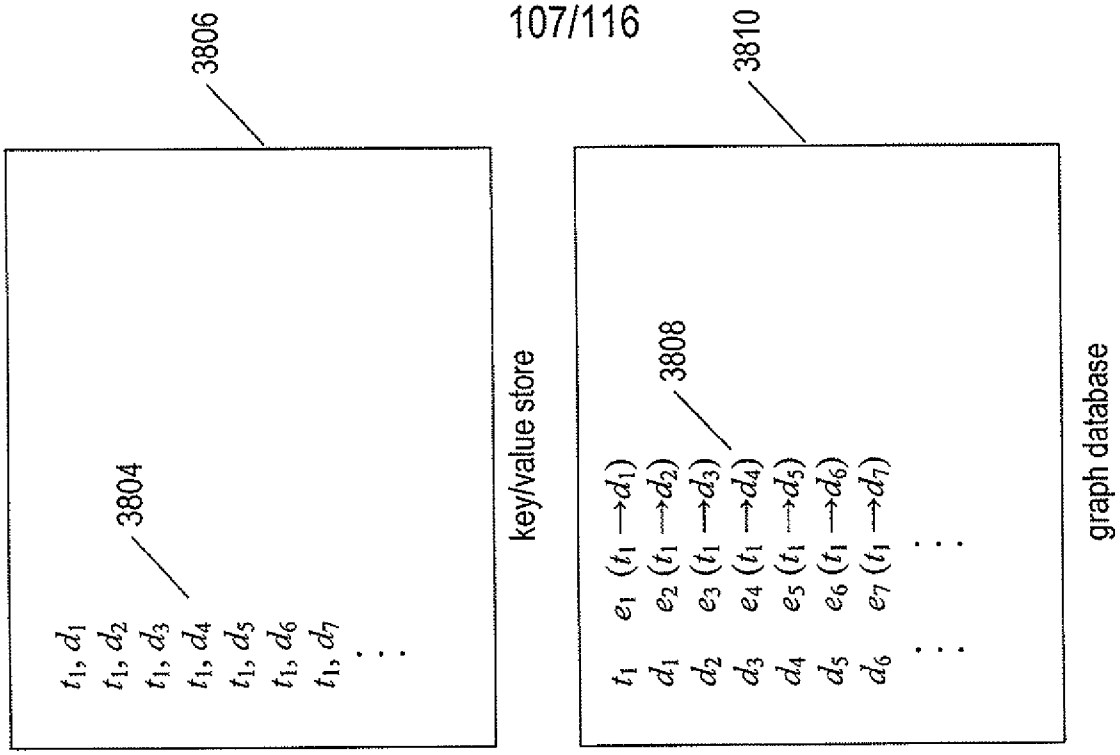

3622
query
3628
3630
3626
$(t_{10}, d_{16}, d_{26}, d_{33})$
$(t_{16}, d_{40}, d_{41}, d_{42})$
$(t_3, d_{10}, d_{13}, d_{21})$

FIG. 37B


3806
3804
t_1, d_1
t_1, d_2
t_1, d_3
t_1, d_4
t_1, d_5
t_1, d_6
t_1, d_7
key/value store
3810
3808
$t_1 \quad e_1 \ (t_1 \rightarrow d_1)$
$d_1 \quad e_2 \ (t_1 \rightarrow d_2)$
$d_2 \quad e_3 \ (t_1 \rightarrow d_3)$
$d_3 \quad e_4 \ (t_1 \rightarrow d_4)$
$d_4 \quad e_5 \ (t_1 \rightarrow d_5)$
$d_5 \quad e_6 \ (t_1 \rightarrow d_6)$
$d_6 \quad e_7 \ (t_1 \rightarrow d_7)$
graph database

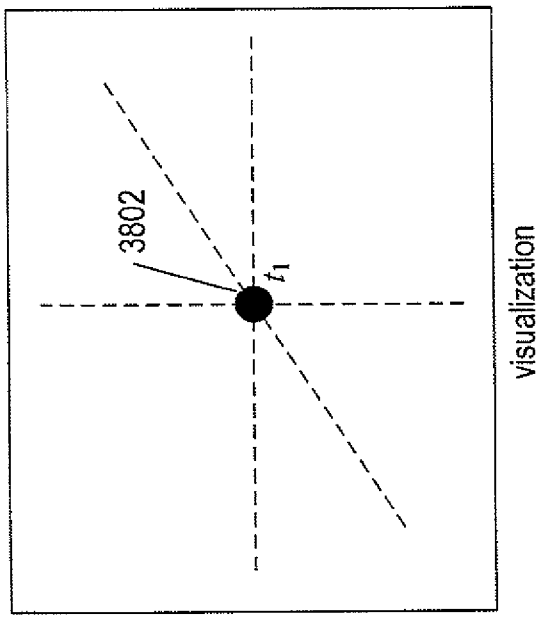

3802
t_1
visualization

FIG. 38A

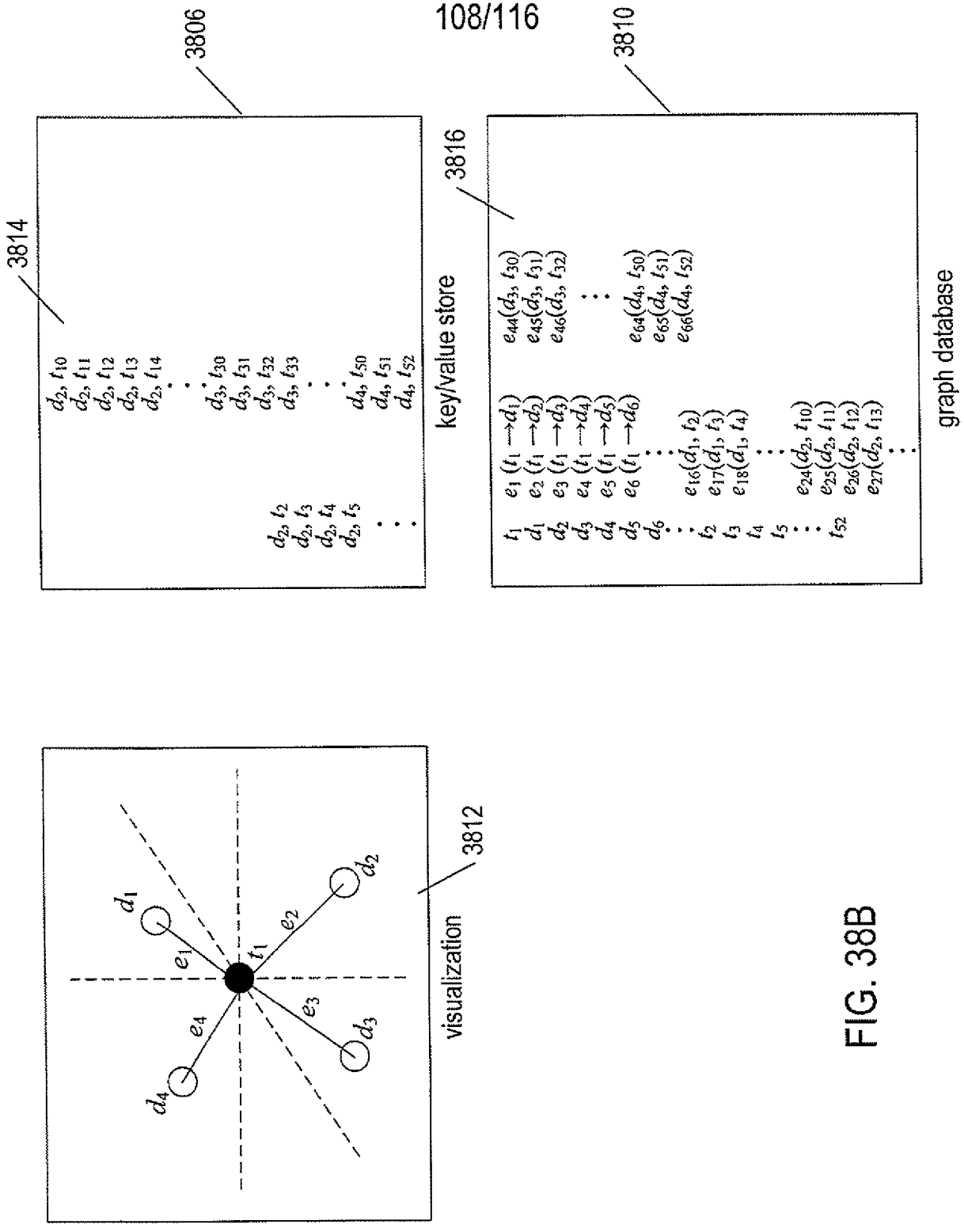

3806
3814
3816
3810
key/value store
graph database
visualization
3812
d_1
d_2
d_3
d_4
e_1
e_2
e_3
e_4
t_1

FIG. 38B

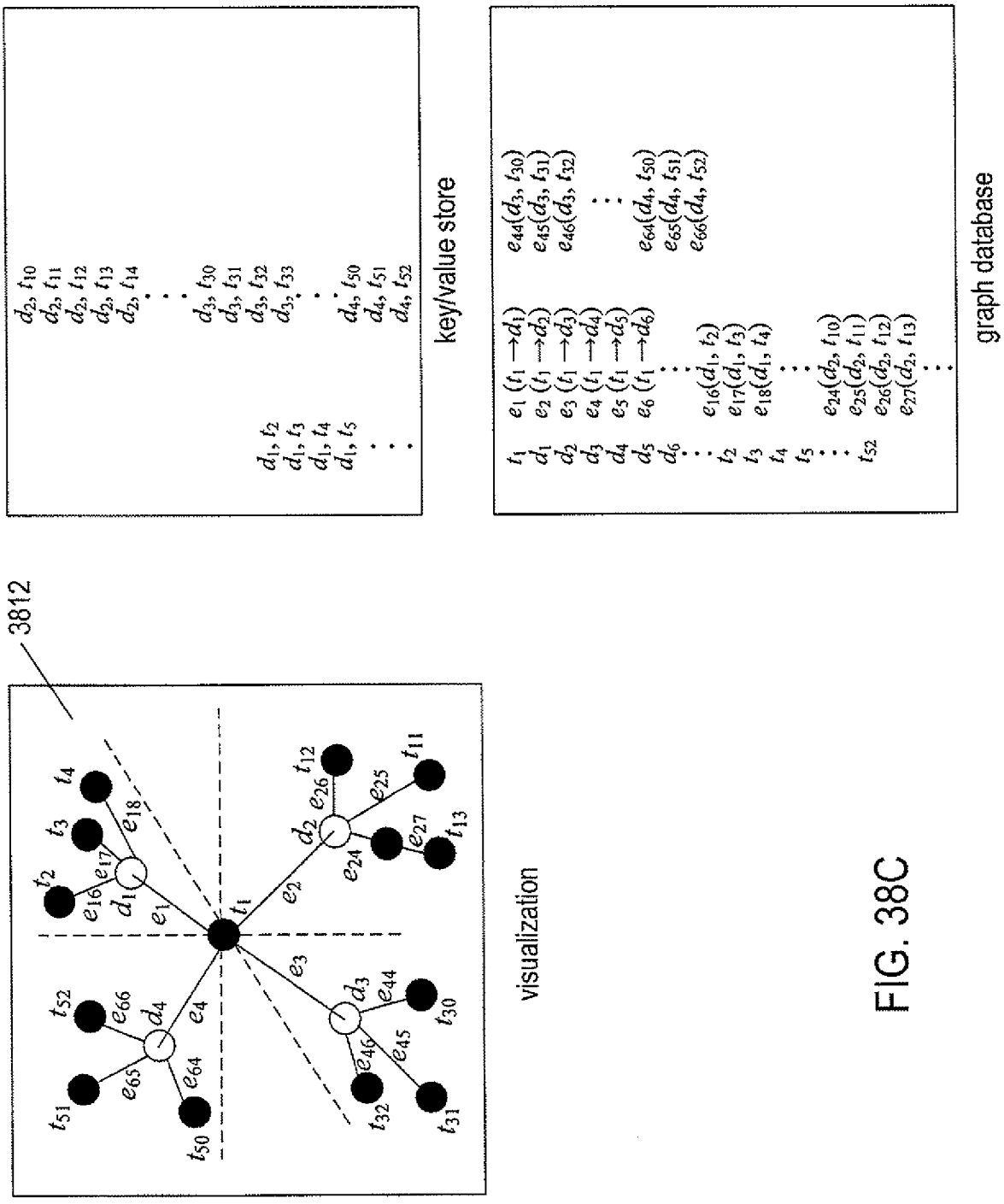


FIG. 38C

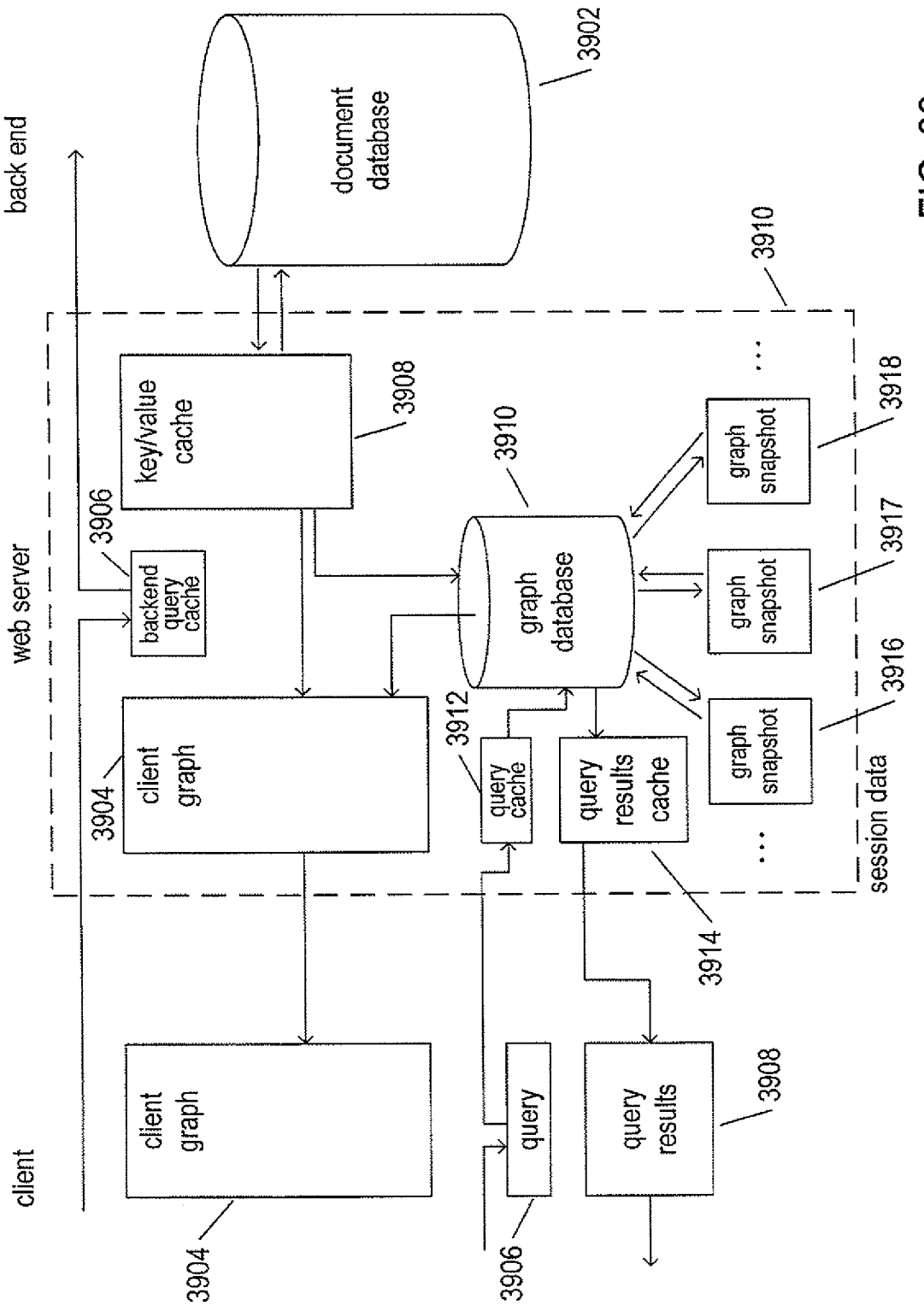

back end
document database
3902
web server
3910
key/value cache
3908
3906
backend query cache
3910
graph database
graph snapshot
3918
graph snapshot
3917
graph snapshot
3916
session data
3904
client graph
3912
query cache
query results cache
3914
client
client graph
3904
query
3906
query results
3908

FIG. 39

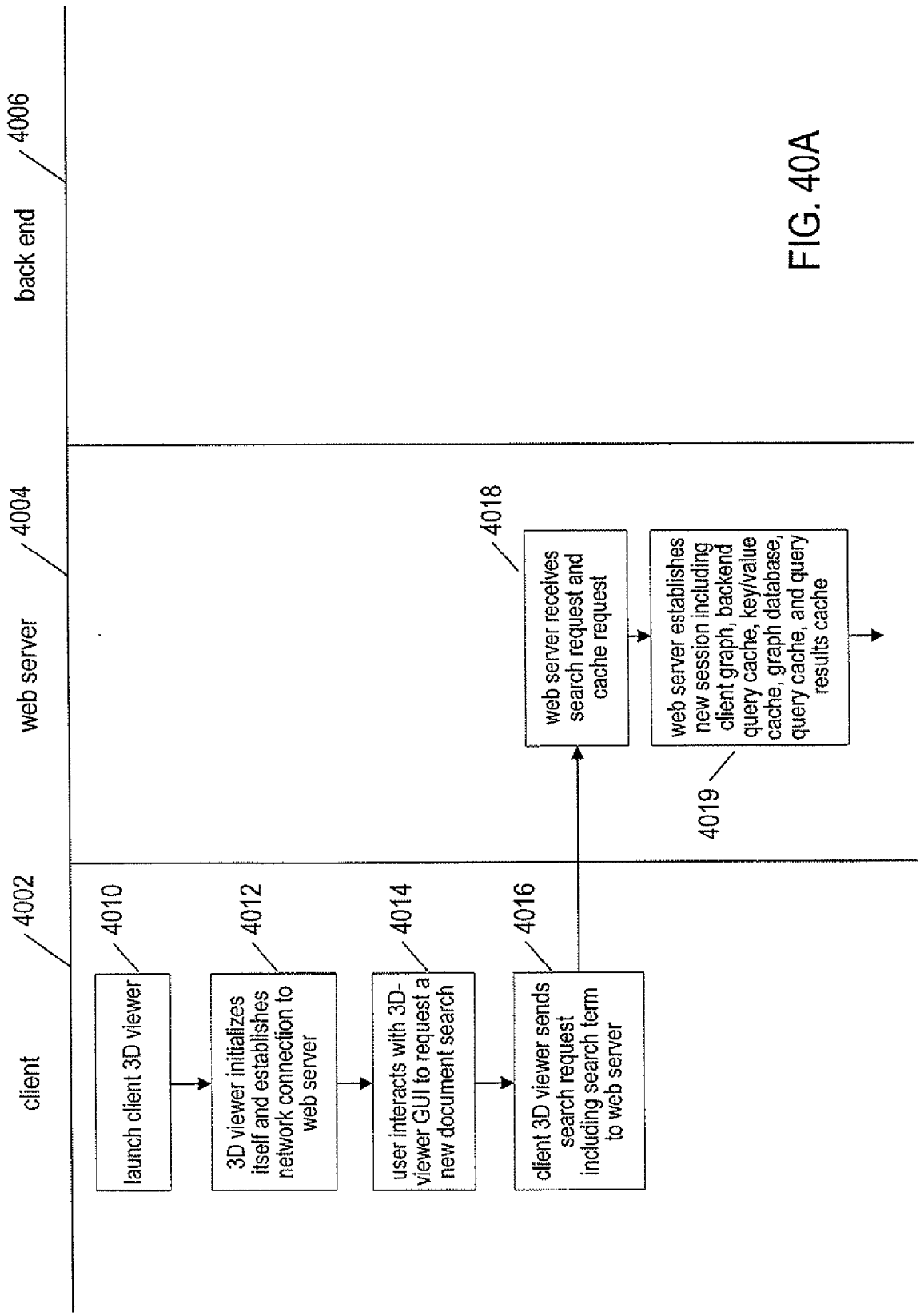

client
4002
web server
4004
back end
4006
launch client 3D viewer
4010
3D viewer initializes itself and establishes network connection to web server
4012
user interacts with 3D-viewer GUI to request a new document search
4014
client 3D viewer sends search request including search term to web server
4016
web server receives search request and cache request
4018
web server establishes new session including client graph, backend query cache, key/value cache, graph database, query cache, and query results cache
4019

FIG. 40A

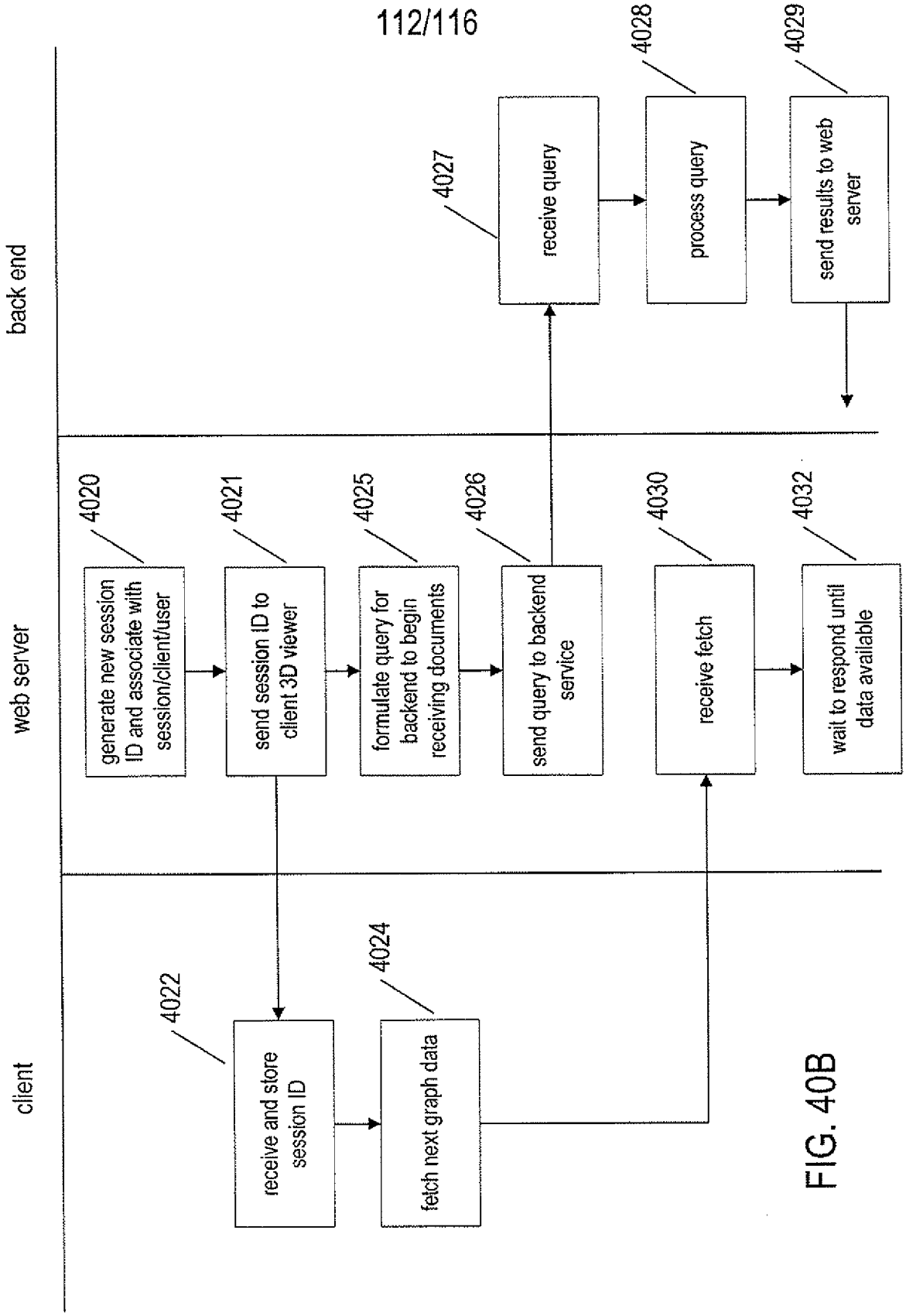

client
web server
back end
4020
generate new session ID and associate with session/client/user
4021
send session ID to client 3D viewer
4022
receive and store session ID
4024
fetch next graph data
4025
formulate query for backend to begin receiving documents
4026
send query to backend service
4027
receive query
4028
process query
4029
send results to web server
4030
receive fetch
4032
wait to respond until data available

FIG. 40B

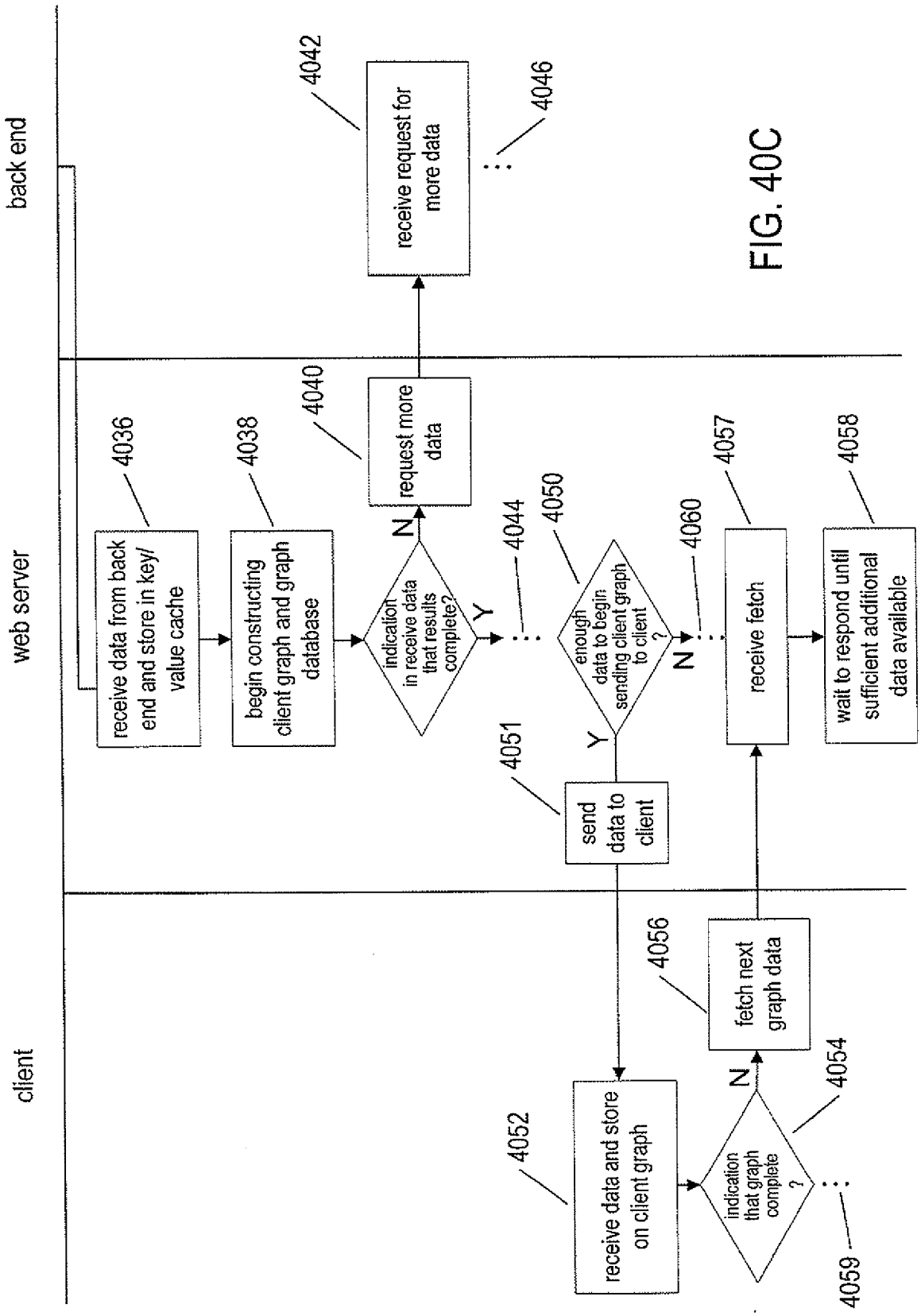

back end
web server
client
4042
receive request for more data
4046
4040
request more data
4036
receive data from back end and store in key/ value cache
4038
begin constructing client graph and graph database
indication in receive data that results complete?
N
Y
4044
4050
enough data to begin sending client graph to client ?
N
Y
4060
4057
receive fetch
4058
wait to respond until sufficient additional data available
4051
send data to client
4056
fetch next graph data
4052
receive data and store on client graph
indication that graph complete ?
N
4054
4059

FIG. 40C

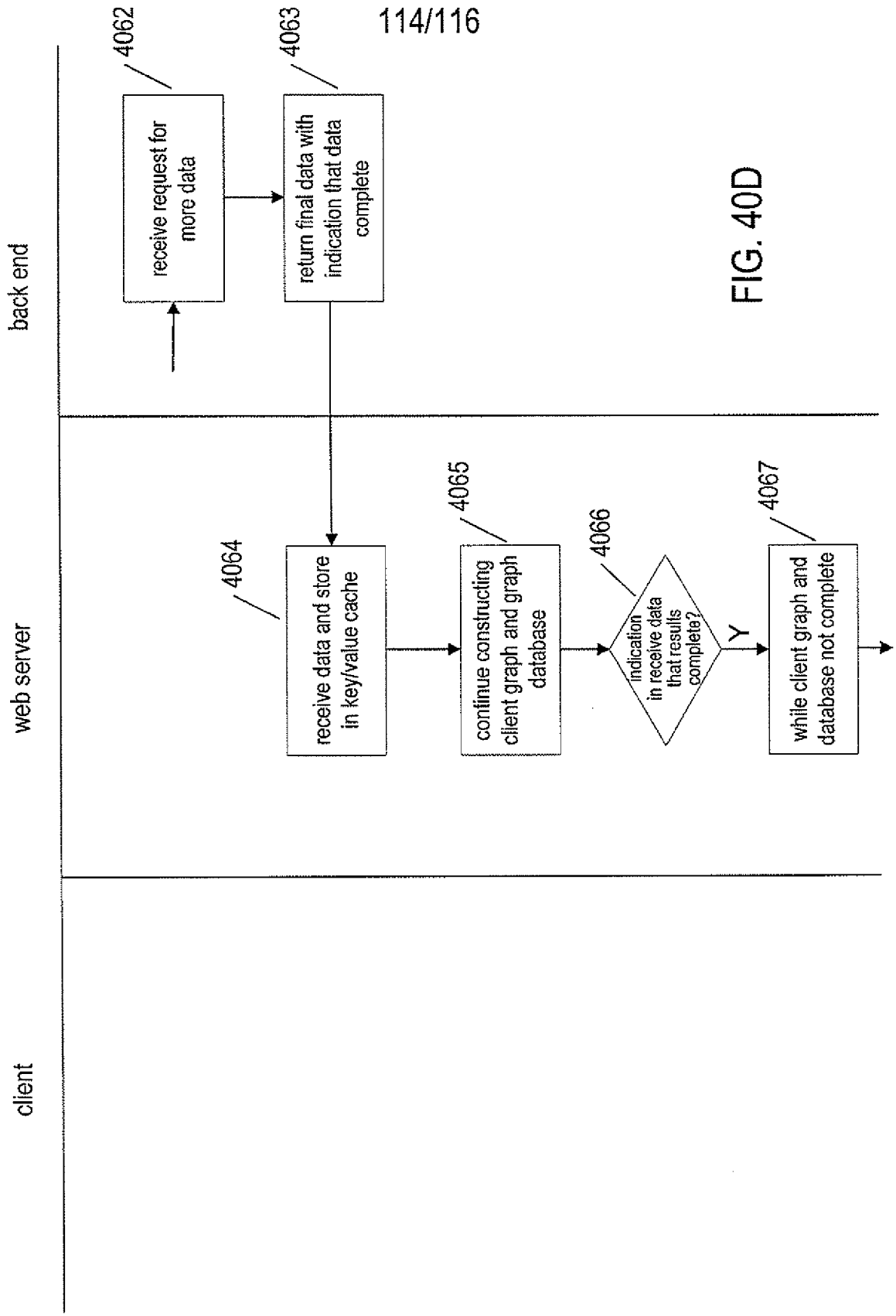

back end
web server
client
4062
receive request for more data
4063
return final data with indication that data complete
4064
receive data and store in key/value cache
4065
continue constructing client graph and graph database
4066
indication in receive data that results complete?
Y
4067
while client graph and database not complete

FIG. 40D

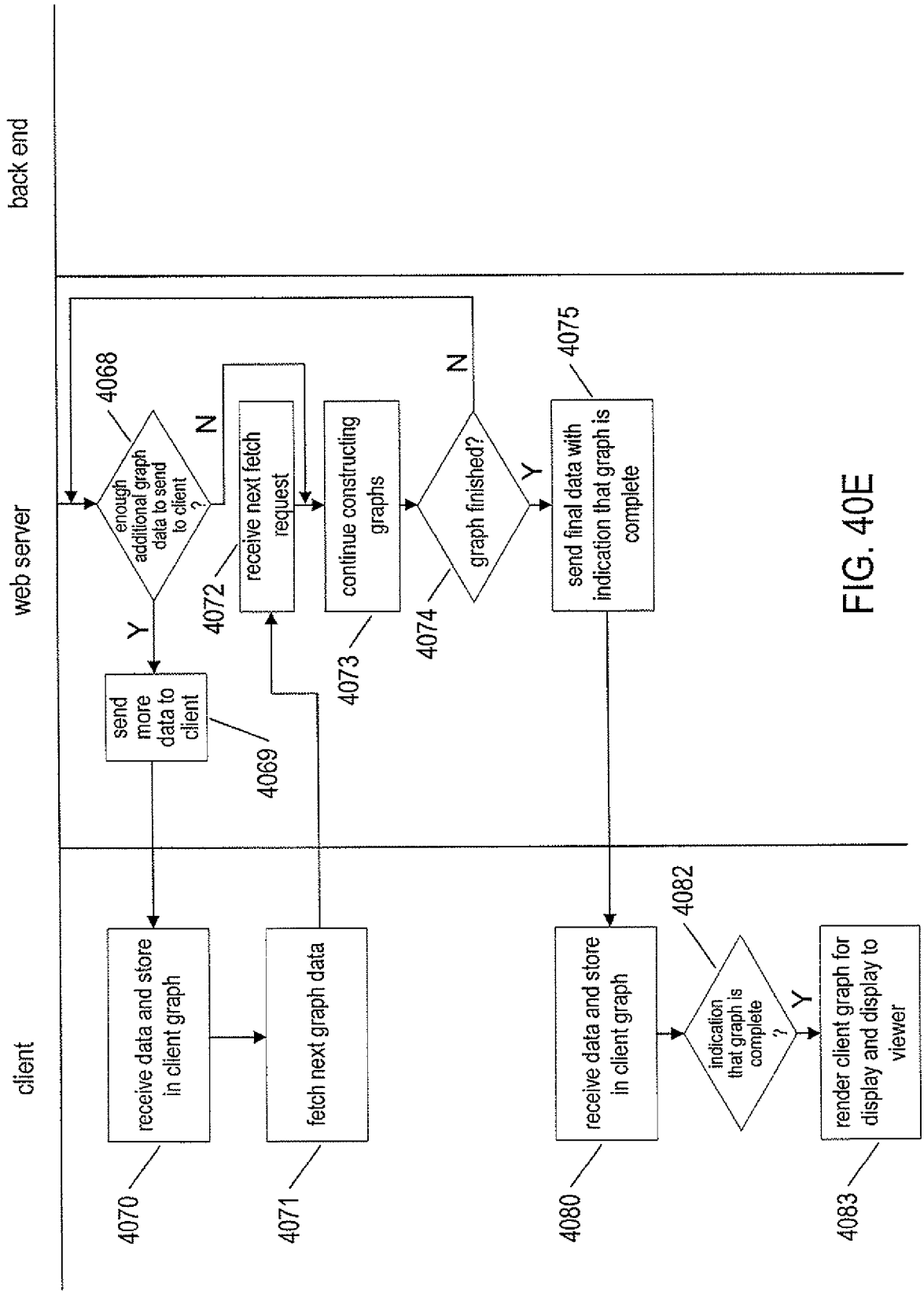

back end
web server
client
4068
enough additional graph data to send to client ?
N
Y
receive next fetch request
4072
continue constructing graphs
4073
graph finished?
4074
N
Y
4075
send final data with indication that graph is complete
send more data to client
4069
receive data and store in client graph
4070
fetch next graph data
4071
receive data and store in client graph
4080
4082
indication that graph is complete ?
Y
render client graph for display and display to viewer
4083

FIG. 40E

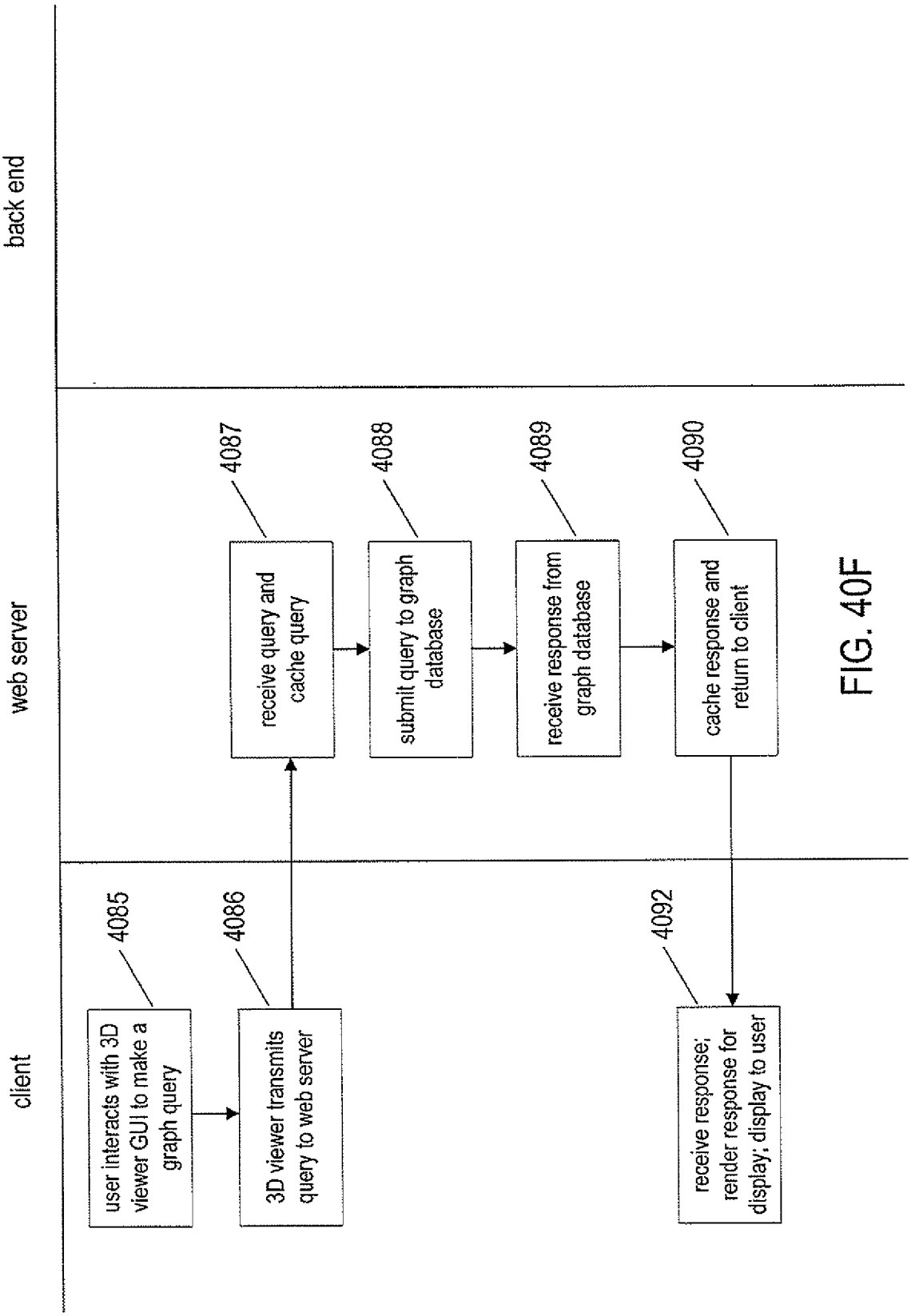

client
web server
back end
user interacts with 3D viewer GUI to make a graph query
4085
3D viewer transmits query to web server
4086
receive query and cache query
4087
submit query to graph database
4088
receive response from graph database
4089
cache response and return to client
4090
receive response; render response for display; display to user
4092

FIG. 40F

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/047980

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30(2006.01)i, G06F 17/21(2006.01)i

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)
G06F 17/30; G06F 3/048; G06F 7/00; G06F 17/21

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: electronic document, search result, graph, three-dimensional, display, query, GUI, visualization, node, icon, user input, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2008-263022 A1 (EVANGELOS P. KOSTORIZOS et al.) 23 October 2008 See paragraphs [0016]-[0018], [0031]-[0049], and [0075]-[0095]; and figures 1 and 6.	1-20
A	US 2004-0103090 A1 (CHRISTIAN DOGL et al.) 27 May 2004 See paragraphs [0037]-[0054] and figures 1-2.	1-20
A	US 2012-0221553 A1 (PHILIP L. WITTMER et al.) 30 August 2012 See paragraphs [0061]-[0063] and figures 15-16.	1-20
A	US 2008-0275871 A1 (VIKTORS BERSTIS et al.) 06 November 2008 See paragraphs [0023]-[0031] and figure 1.	1-20
A	US 8,805,842 B2 (SIMON JOHN DENNIS) 12 August 2014 See column 2, line 64 - column 4, line 55; and figures 1-3.	1-20

☐ Further documents are listed in the continuation of Box C.

☒ 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
08 December 2015 (08.12.2015)	**08 December 2015 (08.12.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/047980

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008-0263022 A1	23/10/2008	WO 2008-130671 A1	30/10/2008
US 2004-0103090 A1	27/05/2004	AU 2002-13933 A1	02/04/2002
		EP 1189148 A1	20/03/2002
		WO 02-25484 A1	28/03/2002
US 2012-0221553 A1	30/08/2012	EP 2678774 A1	01/01/2014
		EP 2678774 A4	08/04/2015
		JP 2014-510968 A	01/05/2014
		KR 10-2014-0041452 A	04/04/2014
		WO 2012-116287 A1	30/08/2012
US 2008-0275871 A1	06/11/2008	US 2006-0173818 A1	03/08/2006
		US 7421426 B2	02/09/2008
		US 8005811 B2	23/08/2011
US 8805842 B2	12/08/2014	US 2013-262442 A1	03/10/2013