A hierarchical shortest path first (HSPF) protocol, routers of a network are grouped in areas, and routing and client subscription information are distributed through all levels of the network hierarchy in the same way. Each level of the hierarchy identifies its connections with its peers that have the same level of hierarchy, and represents areas outside its own as individual nodes. The number of levels of hierarchy is not limited to any particular number, and each level performs the same operations to share routing information and generate routes for data. Distribution of link-state and client subscription information begins at the router level, and continues up the levels of the hierarchy until distributed through the network.
FIG. 1
FIG. 2
FIG. 4A
FIG. 4C
FIG. 5A
FIG. 5C
FIG. 6
ADMINISTRATOR CONFIGURES NETWORK WITH NODES HAVING LINKS OR CONNECTIONS TO OTHER NODES 1002

ADMINISTRATOR CONFIGURES NETWORK TOPOLOGY WITH AREAS EACH HAVING ONE OR MORE NODES 1004

ADMINISTRATOR ASSIGNS NODE IDENTIFIERS TO EACH NODE 1006

NODES DISCOVER NEIGHBORS IN RUNTIME 1008

NODES GENERATE LOCAL LINK-STATE INFORMATION BASED ON DISCOVERY 1010

NODES FLOOD LINK-STATE INFORMATION TO OTHER NODES ON SAME HIERARCHY LEVEL 1012

AREAS FLOOD LINK-STATE INFORMATION TO OTHER AREAS ON SAME HIERARCHY LEVEL 1014

NODES DISCOVER CONNECTED CLIENTS (DATA SOURCES AND DATA TARGETS) 1016

NODES DISTRIBUTE CLIENT CONNECTION INFORMATION THROUGHOUT NETWORK 1018

NODES STORE LINK-STATE AND CLIENT INFORMATION FOR LOCAL NODE, AND FOR OTHER NODES AND AREAS BASED ON FLOODED INFORMATION 1020

FIG. 10
ADMINISTRATOR CONFIGURES NETWORK CONNECTIONS AMONG NODES AND CONFIGURES NETWORK TOPOLOGY WITH NODES GROUPED INTO AREAS 1102

NODES DISCOVER LINKS AND GENERATE LOCAL LINK-STATE INFORMATION 1104

NODES DISTRIBUTE LINK-STATE INFORMATION THROUGH LAYERS OF NETWORK 1106

NODES UPDATE LINK-STATE INFORMATION BASED ON DISTRIBUTION 1108

NODES DISCOVER AND DISTRIBUTE DATA SOURCE AND DATA TARGET INFORMATION 1110

NODES CALCULATE ROUTING PATHS FOR DATA BASED ON LINK-STATE INFORMATION AND GENERATE ROUTING TABLES 1112

NODES ROUTE QUERIES TO DATA SOURCES, AND ROUTE QUERY REPLIES TO DATA TARGETS, BASED ON ROUTING TABLES 1114

NODES DYNAMICALLY UPDATE NETWORK VIEW AND LINK-STATE AND CLIENT INFORMATION AS THEY CHANGE, AND PROPAGATE THE CHANGES THROUGHOUT THE NETWORK 1116

FIG. 11
Hierarchical Shortest Path First Network Routing Protocol

Related Applications

This application is based on U.S. Provisional Application 61/116,622 filed Nov. 20, 2008, and claims the benefit of priority of that provisional application. Furthermore, the provisional application is hereby incorporated by reference.

Field

The invention is generally related to data exchange within a network, and more particularly to a routing protocol to route data in a network with a hierarchy that scales to a theoretically limitless number of nodes.

Copyright Notice/Permission

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever. The copyright notice applies to all data as described below, and in the accompanying drawings hereto, as well as to any software described below: Copyright © 2009, SAP AG, All Rights Reserved.

Background

In a network there are many types of protocols that are used to transfer data from one location within the network to another. Among the protocol types that can be used to transfer data between nodes are link-state routing protocols, which dynamically distribute network knowledge and adjust as network links change. An example of a link-state protocol is OSPF (Open Shortest-Path First), which is commonly used on the Internet to route data. OSPF allows the destinations to be organized into a hierarchy to abstract away parts of the network from other areas of the network. However, OSPF has various implementation restrictions that limit its effectiveness for making large amounts of information available. For example, OSPF is limited to allow two levels in a hierarchy, and mandates a different routing protocol in each level. More particularly, the lower level of the hierarchy employs link-state, while the higher level employs the known Bellman-Ford implementation. The routing tables of OSPF provide information that indicates where certain nodes are located. However, OSPF has no knowledge of enterprise events, and cannot indicate where in a network such events can be accessed.

Originally, OSPF was created to deal with internal local area networks (LANs). OSPF was also originally peer-to-peer, and now may be implemented with multicast support. However, the multicast support is an extension of a protocol that was originally designed to be implemented with modest numbers of network nodes. The extension of OSPF from LANs to a large network illustrates a weakness with OSPF in regard to scaling. There are limits on how many nodes may be included in an OSPF implementation, which may be significantly less than what is needed, for example, to allow event data exchanges in an enterprise.

Brief Description of the Drawings

The following description includes discussion of figures having illustrations given by way of example of implementations of embodiments of the invention. The drawings should be understood by way of example, and not by way of limitation. As used herein, references to one or more “embodiments” are to be understood as describing a particular feature, structure, or characteristic included in at least one implementation of the invention. Thus, phrases such as “in one embodiment” or “in an alternate embodiment” appearing herein describe various embodiments and implementations of the invention, and do not necessarily all refer to the same embodiment. However, they are also not necessarily mutually exclusive.

Fig. 1 is a block diagram of an embodiment of network having nodes that route data in accordance with a hierarchical shortest path first (HSPF) protocol.

Fig. 2 is a block diagram of an embodiment of an example of a network configuration.

Fig. 3 is a block diagram of an embodiment of a network that implements neighbor detection with HSPF.

Figs. 4A-4F are block diagrams of an embodiment of a network that shares link-state information via information flooding for HSPF.

Figs. 5A-5D are block diagrams of an embodiment of a network that handles client subscription with HSPF.

Fig. 6 is a block diagram of an embodiment of HSPF in a network configuration having one data source and one data target.

Fig. 7 is a block diagram of an embodiment of HSPF in a network configuration having one data source and multiple data targets.

Figs. 8A-8B are block diagrams of an embodiment of HSPF in a network configuration having multiple data sources and multiple data targets.

Fig. 9 is a block diagram of an embodiment of a network adjusting to network problems with HSPF.

Fig. 10 is a flow diagram of an embodiment of a process for updating network configuration with HSPF.

Fig. 11 is a flow diagram of an embodiment of a process for applying HSPF in data routing in a network.

Descriptions of certain details and implementations follow, including a description of the figures, which may depict some or all of the embodiments described below, as well as discussing other potential embodiments or implementations of the inventive concepts presented herein. An overview of embodiments of the invention is provided below, followed by a more detailed description with reference to the drawings.

Detailed Description

A hierarchical shortest path first (HSPF) protocol provides a framework for establishing a network that, at least theoretically, has no limitations on the number of levels of hierarchy or the number of network destinations. Thus, an HSPF implementation can scale for larger networks. The number of nodes necessary to support an event data network in an enterprise can be supported with an HSPF implementation. Briefly, routers of a network are grouped in areas, and routing and client subscription information are distributed...
through all levels of the network hierarchy in the same way. Each level of the hierarchy identifies its connections with its peers that have the same level of hierarchy, and represents areas outside its own as individual nodes. Distribution of link-state and client subscription information begins at the router level, and continues up the levels of the hierarchy until distributed through the network.

[0020] Each individual router discovers its links to other routers, including links to routers within the same area and any links to other areas. Thus, routers discover links in the same area as themselves (the discovering router) as well as to “external connections” or links to routers in other areas, if any exist. HSPF does not use or require border routers. Each router exists within one or more areas within the network hierarchy. Connections between areas are not made through border routers, but rather through link-state connections that represent the other areas as single nodes to which information can be sent. Such an implementation provides at least two security benefits. The first is the organizational structures are hidden. Thus, different organizations can share information without exposing their structure. There may be limited points of access, but everything behind the available connections is simply seen as a node that has (potentially a very large amount of) information that can be obtained. In this way, large organizations are large nodes with a lot of information, and smaller organizations are smaller nodes. It will be understood that with reference to HSPF, a node refers to an individual router or a group of routers (e.g., an area or groups of areas).

[0021] A second security benefit is changes are localized within an area without directly affecting network topology outside an area. Thus, when an area changes its internal structure or what information is available or what clients are subscribed (either sources or targets of information), the area itself is different in terms of what information it has, and what it broadcasts to other areas, but no configuration change is required for the network. Rather, HSPF allows the area to discover the change, and then flood or distribute the information to the rest of the network, which will then be aware of the changes. Flooding refers to distribution performed by sending information on multiple or all links to distribute network information to multiple or all neighbors or connected peers.

[0022] After discovering link-state information, the link-state information is distributed within a router’s area, and then from the area out to other areas of the same hierarchical level, and then to higher levels of hierarchy. In the router-level distribution, each router becomes aware of the link-state information of its peers. In the higher-level distribution, each router becomes aware of the link-state information for other areas. The process for distributing link-state information is the same at all levels of hierarchy. In one embodiment, the process is the same for the distribution of client connections (both connections for data targets (data requestors) and data sources).

[0023] In implementation, each router includes a link-state storage. The link-state storage may be referred to as a link-state database that is updated with information about the router’s own links as well as the links of other nodes (other routers and external areas). Each router also includes an HSPF protocol stack or network stack, which is an HSPF engine that implements the HSPF protocol described herein. The HSPF protocol defines the discovery and distribution of information in the network, and determines how routing paths are calculated. The routing paths are calculated based on the stored link-state information.

[0024] FIG. 1 is a block diagram of an embodiment of network having nodes that route data in accordance with a hierarchical shortest path first (HSPF) protocol. System 100 represents a network in which the HSPF protocol is implemented. The configuration of the network is discussed generally, rather than in specific. In general, system 100 has multiple nodes, each of which may be connected to one or more other nodes in the network. It will be understood that reference with respect to a node within system 100 may be to an individual router. However, the description also holds true if each node is a collection or group of routers.

[0025] Each node belongs to an area. Each area may have any number of nodes (routers or other areas). There may be any number of areas. Each area may further belong to a larger area that is a collection of areas, each treated as a node for purposes of the larger area(s). While the terminology “hierarchy level” is typically used herein, it will be understood that different language could be used to mean the same thing, such as referring to hierarchy “layers.” The particular label does not affect the general concept described. As shown, system 100 includes areas 120, 130, 140, and 150. The areas can be any logical network organization. Areas can be a collection of nodes that represent separate companies, geographies, business organizations, departments, industry or special interest groups, business partners, etc., or any combination of these.

[0026] Area 120 is depicted with node 110, which includes HSPF engine 112, link-state database (LSDB) 114, and routing tables 116. Other nodes may be present in area 120, as indicated by the ellipses, but are not explicitly shown. It will be understood that the HSPF engine, the LSDB, and the routing tables exist at a router level within area 120. However, from the perspective of an area, there is a router that represents the area with the information, and ultimately makes routing decisions for the area based on the information. Thus, HSPF engine 112, link-state database (LSDB) 114, and routing tables 116 can exist for an area based on one or more routers in that area.

[0027] HSPF engine 112 represents components within node 110 that implement a protocol stack to implement HSPF or a network stack with HSPF integrated. HSPF engine 112 includes discovery mechanisms to discover links, data sources, and data targets. The discovery mechanisms for data sources and data targets may include modules that allow registering of the source or target client with the node. HSPF engine 112 also includes mechanisms to gather link-state information from other nodes in the network and store the information for later use. HSPF engine 112 includes mechanisms to calculate routing paths based on the stored link-state information. Ultimately, based on operations of HSPF engine 112, the node will forward information to other network locations.

[0028] LSDB 114 represents any type of storage of link-state information that may be employed by a network node of system 100. While a specific link-state database protocol or standard may be followed, there may be variations or alternatives that may be alternatively used. In general, LSDB 114 allows the gathering and storage of link-state information both for the local node itself, as well as for other nodes in the network. Higher-level areas in the hierarchy are represented within LSDB 114 as individual nodes. Thus, local connections are represented as nodes, as are higher-level connec-
tions. Such higher-level connections may be referred to as “external” connections, referring to the fact that they connect to an entity in the network that is external to the area to which the node belongs.

Routing tables 116 refer to stored information that indicates how data can be routed through system 100. In one embodiment, routing tables 116 refer to cached data that changes every time a change occurs to a local area configuration, whether it is the addition or deletion of a node, the addition or deletion of a client, or the change to external links for the area. Routing tables are calculated based on link-state information.

It will be understood that node 110 can exist as a standalone hardware component with hardware and software components that enable the implementation of the HSIPF engine, LSDB, and routing tables. In one embodiment, node 110 is a node that is executed on shared hardware within system 100. At a base level, each node operates on hardware resources (e.g., memory, network connections), whether standalone or shared. It will be understood that node 110 communicates over network ports, which may be hardware resources of the device, or virtual ports, which at some level reference physical shared hardware.

Particular components are described with reference to node 110, but it will be understood that each node of system 100 includes similar components, which are not explicitly shown for purposes of simplicity in the drawings and descriptions.

Area 130 is illustrated with four nodes: 132, 134, 136, and 138. Other nodes may be present. Node 138 is shown with a dashed line, for purposes of discussion below. Each node in area 130 may be directly connected with any one or more other nodes in system 100, including nodes inside of area 130, and nodes outside area 130. For example, node 134 has no direct connections outside area 130. However, node 132 connects with node 110 of area 120, node 136 connects with a node in area 140 (not explicitly shown), and node 138 connects with a node in area 150 (not explicitly shown).

Each node periodically discovers its links, which include its neighbors (the nodes within area 130) as well as its external links (e.g., node 132 is connected to node 110). The link-state information for each node is stored locally (e.g., in a storage similar to LSDB 114) and shared among its neighbors. At a first level, none of the nodes share the link-state information with linked nodes that are outside of their area. Thus, for example, node 132 indicates nodes 110, 134, and 136 to nodes 134 and 136, but not to node 110. At the area level, a selected node (as discussed in more detail below) represents area 130, and indicates the link-state information in an HSPF implementation, routing is initially on a “macro” level (e.g., a node knows which local area node to
send data to reach a particular area), and becomes more detailed as the data is routed closer to the target location. Thus, routing with HSPF could be referred to as a “bird’s eye routing.” HSPF allows uncoordinated changes within the network, while allowing all nodes to maintain a steady state view of where to route data. Nodes only need to know their neighbors and which links each node and neighbor has. As with other hierarchical network implementations, HSPF reduces the memory requirements of the nodes.

In one embodiment, a specific implementation of HSPF can be used in an enterprise computing system, such as the Live Enterprise (LE) of SAP AG of Walldorf, Germany. LE turns events distributed across multiple servers, locations, and companies into data sources. The “federated data” of these data sources can be exposed in a consistent and contextual way through an open and comprehensive operational framework provided by HSPF. LE manages operational data by correlating event and historical data. With HSPF, LE can route queries to the source of data, rather than relying on replication of data and centralized solutions.

The Hierarchical Shortest Path first (HSPF) technique provides a routing protocol that meets specific needs of LE. In one embodiment, LE is deployed on an overlay network in the application layer, and employs HSPF as a routing protocol to direct data. Link-state routing protocols dynamically distribute network knowledge and adjust as the network changes. HSPF allows the insertion of content information in the form of bit vectors into the routing tables. With the content information in the routing tables, the routers understand the enterprise/event data that is available on those networks and computer nodes. A node chosen by convention becomes the publisher of the bit vector index of the event types and data in an area, or a disjoint grouping of peer LE servers. The publisher collects the index information and publishes it to all LE servers in its area as well as all higher levels of hierarchy for which it is the publisher by convention.

In one embodiment, each LE router has a routing table entry for each destination in its area. It also includes entries indicating the content available in peer areas as well as higher levels of hierarchy. The information on what information is available allows for HSPF to minimize the amount of memory resource needed within the router. As the connections between LE servers change, which changes the topology of the network, these changes are propagated intra-area before propagating the changes at the next higher level of the network hierarchy. The change is then propagated intra-area at this level, and so on. By abstracting a group of nodes into an area and treating them as a single node, the change propagation method can be the same at all levels. At the same time, the routing hierarchy effectively minimizes the needed control traffic overhead because the nodes within an area and any routing changes within that area are hidden from the view of nodes that are outside that area.

With each entry including information on a destination’s event, data content, address, and the shortest routing path to the destination, an LE router can use its table to determine the final destinations of an event query. The use of these particular routing tables by LE routers allows for a decoupling of event consumers and event producers. With a combined implementation of HSPF and an eventing network such as LE, new destinations can be added at run time and without reconfiguration of any component of the network.

FIG. 2 is a block diagram of an embodiment of an example of a network configuration. System 200 represents a network in which HSPF may be implemented. In one embodiment, system 200 is an enterprise system, which may use HSPF to route event data. It will be understood that HSPF can be applied to other network configurations, and for routing any type of data.

System 200 illustrates a sample network configuration that sets the background for the following discussion of hierarchical link-state. The same hierarchical link-state is applied at every level or layer of system 200. In system 200, the smallest blocks (e.g., 1.1.1, 1.1.2, 1.1.3, 1.2.1, 1.3.1, 1.3.2, 1.4.1, 1.4.2, 1.4.3) represent routers, whereas larger blocks represent areas (e.g., 1.1.x, 1.2.x, 1.3.x, 1.4.x). As used herein, “router” refers to any network node that can obtain and/or forward data from one point to another point within the network of system 200. An “area” is a disjoint collection of routers or other smaller areas. As shown, routers can be interconnected or have links to other routers. The routers in an area are interconnected directly or through other routers from the same area. As suggested above, each of the blocks identified here as a “router” could also be generically referred to as a “node” within the network of system 200, and a “node” can also refer to a representation of an area within a router’s local link-state information.

The configuration of system 200 may be poorly configured from the perspective of a production system. However, the exact configuration of system 200 is not as significant as the discussion of HSPF within the configuration of system 200. Among the deficiencies of the configuration of system 200 is the fact that there exist multiple single points of failure, seeing that most of the links are critical to keep the system alive. For example, if the link between 1.2.2 and 1.1.3 were broken, there would be two unconnected subsets of area 1.x.x. The first subset would include routers 1.1.1, 1.1.2, and 1.1.3. The second subset would include routers 1.2.1, 1.2.2, and 1.2.3. One problem with such a configuration is that if the link fails, each subset would believe it represents the entire area 1.x.x. Thus, the failure would cause duplication of sending area information to routers outside area 1.x.x.

Although illustrated adequately, the basic configuration of system 200 is described here, given the configuration is the basis for discussion of HSPF for FIGS. 3 through 9. Area 1.x.x includes a total of six routers grouped in two sub-areas. From the perspective of HSPF, the routers themselves (1.1.1, 1.1.2, 1.1.3, 1.2.1, 1.2.2, 1.2.3) are a first hierarchical layer or level, the sub-areas (1.1.x, 1.2.x) are a second hierarchical layer, and area 1.x.x is a third hierarchical layer, as described in more detail below. Sub-area 1.1.x includes routers 1.1.1, 1.1.2, and 1.1.3, while sub-area 1.2.x includes routers 1.2.1, 1.2.2, and 1.2.3.

Area 2.x.x includes sub-areas 2.1.x and 2.2.x, which include, respectively, routers 2.1.1, 2.1.2, and 2.1.3, 2.2.1, 2.2.2, and 2.2.3. Area 3.x.x includes sub-areas 3.1.x and 3.2.x, which include, respectively, routers 3.1.1 and 3.1.2, and 3.2.1 and 3.2.2. The areas are connected via connection between routers of the areas. The areas can be considered to exist as an abstraction of the routers, which allows the routers to be organized in ways that allow efficient routing. Thus, the connections between areas are simply the connections that exist between the routers. There is not necessarily centralized intelligence in a sub-area or area, other than certain determinations for which routers play what roles within the sub-area or area, which may be defined by the protocol. Thus, router 1.1.2 is linked to router 3.1.1. Router 3.1.1 is also linked to routers 1.2.1, 1.2.3, and 2.1.1. Router 1.2.3 is linked to router 3.1.2, and router 3.2.1 is linked to router 2.2.3.
While a great diversity of numbers of routers and sub-areas is not illustrated in system 200, it will be understood that there is no requirement for any particular number of routers or sub-areas within an area. Additionally, the numbers of routers may vary from sub-area to sub-area, just as the number of sub-areas may vary from area to area, in any combination within the network. As will be understood from the descriptions below, HISPF offers great flexibility for system configuration, number of routers, and number of areas, and the configuration of each. It will also be understood that network configuration is a design choice for network design, and does not directly affect the implementation of HISPF. Rather, the configuration merely affects which routers will communicate with other routers and for what purposes.

FIG. 3 is a block diagram of an embodiment of a network that implements neighbor detection with HISPF. In one embodiment, each routing node in system 200 continuously listens for neighbors using the Hello Protocol, or another (either equivalent or alternative) neighbor detection protocol. For example router 1.2.3 has four routing neighbors: 1.2.1, 1.2.2, 3.1.1, and 3.1.2. The connections to these neighbor routers are illustrated by highlighted (dashed and darkened) lines in the figure. Router 1.2.3 periodically checks that the neighbors are “alive” or still present in the network by checking the incoming traffic from neighbors. If a neighbor router does not send any information for a specified time period, it is assumed dead and the listening router removes it from its own link list.

To ensure that a router remains on a neighbor’s link list, the router needs to send data to the neighbor. In one embodiment, it is based on a predetermined period of time, each router sends a “heartbeat” Hello message (or similar message that indicates it is a neighbor) to each neighbor router to prevent being removed from the neighbor router’s link list when no other information is sent to the neighbor router. The predetermined period of time may be variable for each implementation, and is established by system configuration, which is a system-specific implementation detail. If the router actively uses a link, for example for data transfer or to exchange link-state information, heartbeat messages are not sent.

FIGS. 4A-4F are block diagrams of an embodiment of a network that shares link-state information via information flooding for HISPF. In FIG. 4A, router 1.2.3 shares link-state information with local neighbors. An example of a link-state for router 1.2.3 is illustrated, and is explained as each item of information on the list is gathered. In general, it will be understood that information flows along the lines of hierarchy.

Periodically, a router shares its link-state information with other routers in its area. In one embodiment, a “reliable flooding” algorithm is used to share link-state information. The router floods the network or sends link-state information to each adjacent router from the same local area. When a router receives link-state information from any neighbor, it first checks to see if the same information is already present in its local link-state database. If the received information is already present, the router simply discards the received information, and no other activity is performed. If the received link-state information is not present in the router’s local link-state database or the local database contains outdated information, the link-state information is added to the database. With flooding, the received information is forwarded by the router to each connected router except to the one from where the information was received. In one embodiment, flooding always happens on an area level, and no information is sent to routers from other areas.

In FIG. 4A, router 1.2.3 sends link-state updates to routers 1.2.1 and 1.2.2, as illustrated by the highlighted lines. The arrows on the lines indicate the direction of the flow of information on the links. Routers 3.1.1 and 3.1.2 are in a different area, and thus updates are sent to those neighbors on higher levels. In one embodiment, a link-state update message includes a list of the originating router is able to refer to as routers the originating router is able to “hear”). Note that a link-state information message only defines one direction of information flow, which is from neighbors to an originating router.

FIG. 4A illustrates link-state information that would be provided by router 1.2.3. The information is stored in a database at the router. The first section of the information is highlighted, which is the information that corresponds to the link-state information shared by router 1.2.3 to its neighbors at the level of area 1.2.x. Namely, router 1.2.3 link-state (LS), as provided by router 1.2.3 to area 1.2.x, indicates links to router 1.2.1, 1.2.2, 3.1.1, and 3.1.2.

Where FIG. 4A illustrates link-state information messages on an area-level, the link-state information is also distributed on higher layers to update the information in system 200. FIG. 4B illustrates the exchange of link-state information between areas. Whereas the link-state information is shared from router 1.2.3 to its area neighbors by router 1.2.3 itself, at higher levels information is shared from an appointed or elected lead router. It is assumed that election of a lead router or election of a router to transmit information for a group is understood in the art, and will not be described specifically here. There are many election algorithms and protocols that may be used in various different implementations. Alternatively, network configuration can include designating a router for a particular information sharing duty.

In one embodiment, the router that acts on behalf of areas for the higher levels of HISPF is selected based on router identifier (ID), which could also be referred to as a node ID. For simplicity in describing the interactions of the routers in the various areas, the term “router ID” is used below. However, it will be understood that each area also includes an ID (a node ID), and the areas are represented as nodes by their IDs on link-state information stored at the routers. Assume for purposes of example that the router designations in system 200 correspond to a router ID, where 1.2.1 is a “lower” or “smaller” ID than 1.2.2, but a higher or bigger ID than 1.1.3, for example. Thus, within an area, a router with the lowest or smallest router ID represents the area for distribution of link-state information in HISPF. Alternatively, a router with the highest or largest router ID could be selected to represent the area. It will be understood that each router includes information about the topology of its area, and thus knows what other routers are within its area. Therefore, each router has information indicating the router IDs of other routers within the area.

Just as the routers periodically share link-state information with neighbor routers, each area periodically shares link-state information with neighbor areas. The period of updating for routers and areas is implementation specific, and depends on the size of the network, the amount of bandwidth expected to be available within the network, the speed of the hardware of the routers, and the frequency with which the network configuration may be expected to change. Thus, for
some network configurations, daily or weekly link-state exchanges may suffice, while other network configurations may need to be updated more frequently. Whatever the period for the sharing of link-state information between routers, the period for sharing/updating between areas is longer than the period for updating between routers.

[0059] Referring again to FIG. 4B, the router with the smallest router ID within area 1.x.x is 1.2.x. Thus, router 1.2.1 publishes the link-state information for area 1.2.x to neighbor areas. Similar to what happens inside an area, link-state information about an area is only shared with peer areas that are at the same level and within the same enclosing higher-level area (e.g., 1.x.x encloses areas 1.1.x and 1.2.x, meaning 1.1.x and 1.2.x are peers on the same level). Router 1.2.1 calculates aggregated information about all external links for area 1.2.x, which in the example of system 200 includes 1.1.3, 3.1.1, and 3.1.2. Routers from the local area (1.2.1, 1.2.2, and 1.2.3) are not included in the list. Duplicate items are also not included. For example, router 3.1.1 is heard by two local routers—1.2.2 and 1.2.3, but 3.1.1 is listed only once in the link list for area 1.2.x.

[0060] Link-state information is distributed across areas with the same reliable flooding algorithm used for distributing link-state information among routers within an area. If for some reason the router with smallest ID does not publish the area’s link-state information, the router with next smallest ID publishes on behalf of the area. The backup publishing can be accomplished by configuring a small delay into the publishing algorithm. The publishing can work as follows: a router checks its router ID against the router IDs for other routers in its area. If the router has the smallest router ID, it publishes the link-state information for the area. If the router does not have the smallest router ID, it does not publish. If the link-state information is not received within a delay period, the router checks its router ID against others to see if it is the next-to-lowest router ID within the area. If so, it publishes on behalf of the area, and if not, it waits again to see if another router publishes. The check and wait can continue until the area’s link-state information is published. Thus, an area’s link-state information will not be updated only if all routers in the area fail. If the area is split into two, then the router with the smallest ID will have incomplete link-state information for the area and, therefore, other areas will have incorrect information about the area’s links.

[0061] In FIG. 4B, the link-state information is highlighted indicating that area 1.2.x hears routers 1.1.3, 3.1.1, and 3.1.2. Router 1.2.1 is indicated as the designated router to publish that link-state information.

[0062] In FIG. 4C, the link-state information is updated for area 1.x.x. In this example, router 1.1.1 is the router with the lowest router ID within area 1.x.x, and thus, router 1.1.1 is designated to publish the link-state information for area 1.x.x. Publishing on the global level uses the same flooding algorithm as the lower levels. As indicated in the highlighted link-state information, router 1.1.1 publishes routers 3.1.1 and 3.1.2 as the link-state list for area 1.x.x. In system 200, these are the only external connections that area 1.x.x has.

[0063] Link-state information may reach a router through several channels. In FIG. 4D, router 3.1.2 may already receive link-state information directly from router 1.2.3 (i.e., in FIG. 4C), and then again receive the link-state information from router 3.1.1. Router 3.1.2 compares the received information with its local database, and discards the duplicate. In this manner, the whole network is protected from information loops, while ensuring the information is reliably distributed. As illustrated, the link-state information as shared by 3.1.1 to 3.1.2 indicates the link list for area 1.x.x as routers 3.1.1 and 3.1.2.

[0064] When flooding link-state information for a non-local area, routers may do additional aggregation and ‘merge’ published link-state information items. In FIG. 4E, information about routers 3.1.1 and 3.1.2 may be merged in a single record which would be indicated as 3.1.x when published to area 3.2.x. The link-state information highlights that the link state information for 1.x.x is 3.1.x when router 3.1.2 publishes to 3.2.1. Observe that from the perspective of router 3.2.1, it does not have any direct connections to 1.x.x, nor do its neighbors in area 3.2.x. Thus, to route information to a router within 1.x.x, the only information 3.2.1 needs to have is that 1.x.x can be reached by sending data to area 3.1.x. Even if area 3.2.x had more than the single link to area 3.1.x, it would still only need to know that 1.x.x can be reached through 3.1.x. Also observe that the link-state information includes a further merged record to indicate 3.x.x as the link list for 1.x.x when 3.x.x publishes link-state information to 2.x.x.

[0065] FIG. 4F illustrates two levels of link-state aggregation. Area 2.x.x has minimal information about area 1.x.x. All 2.x.x knows is that area 3.x.x may be used to deliver data packages to area 1.x.x. Area 2.x.x does not need any other information unless and until one or more routers of area 2.x.x have a direct connection to area 1.x.x. If a router of area 2.x.x obtains a direct connection to a router in area 1.x.x, then area 2.x.x will publish itself as a possible gateway for area 1.x.x and this information will be distributed across the network as described above.

[0066] FIGS. 5A-5D are block diagrams of an embodiment of a network that handles client subscription with HSPE. In one embodiment, system 200 supports the routing of event information through the network. Event information is characterized by having an event and having a consumer or a target of the event information. Joining a target or a client subscriber for event information allows system 200 to route event information to the target. Joining may be considered “registering” or requesting event data, and refers to a target being known within the system as a network location to which certain requested or identified event information should be routed.

[0067] In one embodiment, event information is handled very similarly to link-state information. In one embodiment, bit vectors are used to represent events, where a bit vector indicates a type of event information that is desired by the client. Where bit vectors are used, event aggregation is accomplished by simply combining bit vectors together into a single bit vector that represents all subscribed events. Detailed lists of subscribed events are stored only on the originating router to which the client is connected. All other routers only have aggregated event information.

[0068] In FIG. 5A, E1 target is shown connected to router 1.2.3. Thus, E1 target joins to router 1.2.3, and the event information requested by the target is stored locally at router 1.2.3. As shown in the following figures, the request for the event information is distributed through system 200. As routing information, the target can be indicated in routing tables to what is shown below the target in the figure, and which is developed and discussed in the following figures.

[0069] Client subscription information can be flooded over the network, similar to what is done with link-state information. Thus, each router of system 200 would receive and flood
subscription information similar to what is described for link state information above in FIGS. 4A through 4F. In one embodiment, the client subscription information is distributed as bit vectors, which may be aggregated and distributed across the network. Each router publishes event information about locally connected clients. The published information is flooded to all routers in the local area of the router. In FIG. 5B, router 1.2.3 aggregates all connected client’s events in a single bit vector and publishes it to routers in area 1.2.x (e.g., 1.2.1 and 1.2.2). In one embodiment, event information from the router is distributed with some delay. The delay can help avoid high network load on frequent client subscriptions and unsubscriptions.

[0070] For example, consider if two-hundred clients are connected to router 1.2.3, and are subscribing to new events every 30 seconds. If all changes are propagated immediately to the other routers, it will cause on average 6-7 updates per second. Assuming a bit vector size of 1 Mbit, approximately 700 kB/second of network traffic would be produced from a single router. If changes are delayed for 10 seconds, only one update every 10 seconds would be sent, resulting in approximately 12 kB/second network traffic.

[0071] While FIG. 5B shows “flooding” at a first level (call it “Level 1”), within the area, the subscription information is also flooded to other areas at higher levels (for example, “Level 2”), as shown in FIG. 5C. The router with the smallest ID in area 1.x.x publishes event information on behalf of area 1.x.x. In system 200, router 1.2.1 is the router with smallest ID in area 1.2.x. Thus, router 1.2.1 aggregates all of the area routers’ bit vectors into a single aggregated bit vector, and distributes the aggregated bit vector across whole area 1.x.x.

[0072] In FIG. 5D, flooding at “Level 3” is illustrated, where all subscription information for area 1.x.x is aggregated and distributed. Router 1.1.1 is the router with the smallest ID, and so it publishes subscription information on behalf of area 1.x.x. As mentioned above, router 1.1.1 would have information to publish based on flooding similar to what is described for link state information. Within the subscription information at each level is the fact that E1 is a target, and so E1 is indicated in the distributed information. Thus, whatever level is under consideration for distribution of information, the flooding and communication is performed in the same way as at lower layers, in accordance with the examples shown.

[0073] FIG. 6 is a block diagram of an embodiment of HSPF in a network configuration having one data source and one data target. Assuming a subscribing client, E1 target, is connected to router 1.2.3, the following describes how information is routed to the target. The subscription information is distributed throughout system 200, which means that every router outside of area 1.x.x receives a record in its local event database indicating that area 1.x.x is interested to receive events matching a particular bit vector (call it “A1”). The request can thus be routed (distributed) to the event sources, and the event data routed back to the requesting target.

[0074] In one embodiment, when a client publishes data, it specifies an event ID with every data packet. The event ID is the target destination, which is used to reconstruct a correct delivery path to each subscribed client. Again, it can be assumed that all routers outside area 1.x.x have a record in their local event database stating that area 1.x.x is interested to receive events matching bit vector A1. For this example, event ID E1 matches bit vector A1. Note that bit vector A1 for area 1.x.x was generated by router 1.1.1 using all sub-area bit vectors. At router 2.2.3, event ID E1 of the E1 source is matched with bit vector A1, and the destination is determined to be area 1.x.x.

[0075] It will be understood from the description that router 2.2.3 does not have any detailed information about the specific router within area 1.x.x where the events must be delivered. Thus, router 2.2.3 may first try to find the destination on a higher hierarchy level. At the highest hierarchy level, only three areas exist: 1.x.x, 2.x.x, and 3.x.x, with area 3.x.x being connected to other two areas.

[0076] In one embodiment, the data packets from the data source to area 1.x.x are routed through router 2.1.1. The data packets may be routed from the area via the router with the smallest router ID. Observe that within system 200 there are not “edge” routers in the various areas. The router that acts as a gateway is the one that is selected to route data to outside areas. As the router with the smallest router ID, router 2.1.1 routes the data packets as a gateway router. While router 2.2.3 is directly connected to a router in area 3.x.x, which looks like it could forward the event to router 3.2.1 directly, the routing occurs without knowledge of what else may exist within the network. Each router simply knows what area to send data to, and data is sent to outside areas via a gateway router. Thus, router 2.2.3 cannot directly send the data to router 3.2.1, but instead sends it to router 2.1.1, which then forwards the packet to its knowledge of connections to area 1.x.x.

[0077] It will be understood that all routers across the network use the same algorithm to calculate the exact same routing tree through the network as router 2.2.3. The source for the routing tree is the originating router, which is router 2.2.3 in this case. In one embodiment, the original source router ID is stored in an event data packet to allow each router in the path to calculate the same path.

[0078] Thus, each router in area 2.x.x repeats the path calculations of router 2.2.3 and chooses the same path selected by router 2.2.3. The event data packet is forwarded first to the area with smallest ID, which is 2.x.x in this case. In order to reach 2.1.x, router 2.2.3 routes the event data packet to router 2.2.2, which routes it to router 2.1.3, which then routes it to router 2.1.1. Once the data packet arrives at router 2.1.1, router 2.1.1 forwards the event data packet to area 3.x.x. Router 3.1.1 knows that data was originally sent from area 2.x.x and performs all routing calculations taking into account the fact that the packet originated from area 2.x.x.

[0079] In system 200, router 3.1.1 is the router in area 3.x.x with the smallest ID. Therefore, it can simply forward the event data packet directly to area 1.x.x. In this example, router 3.1.1 has multiple links into area 1.x.x. In one embodiment, router 3.1.1 may select one link based on tie-breaking criteria. For example, router 3.1.1 may select the link with the shortest communication delay. Assume the link to router 1.2.2 has a shorter delay than the links to routers 1.1.2 or 1.2.3. It will be understood that router 3.1.1 could select the link to router 1.1.2. With a complete picture of the topology of system 200, it appears obvious that routing data intended for E1 target through router 1.1.2 is not optimal. However, it will be understood that router 3.1.1 does not have any internal information about area 1.x.x, and therefore cannot select an optimal path by choosing the link to router 1.2.3. If router 1.2.2 is selected as the path with lowest delay, router 3.1.1 forwards the event data packet to 1.2.2, which in turn forwards the event data packet to router 1.2.3, where the data can be provided to the E1 target.
FIG. 7 is a block diagram of an embodiment of HSPF in a network configuration having one data source and multiple data targets. Consider system 200 with the same target as in FIG. 6, now labeled “E1 target1,” the same source, E1 source, and two new targets for the event data of event E1: E1 target 2 and E1 target3. Observe that adding an extra destination to area 3.x.x does not change the routing path calculation until the event data packet reaches router 3.1.1 because both areas 1.x.x and 3.x.x are accessible from router 2.2.3 through router 2.1.1. When the event packet arrives at router 3.1.1, the router determines that the event data packet has destinations to global areas 1.x.x and 3.x.x. The calculation to route the event data packet to area 1.x.x is the same or similar to what is described for FIG. 6. Additionally, router 3.1.1 knows that it is included within area 3.x.x, and so it will also calculate the destination at a lower hierarchy level. Calculating the destination at the lower hierarchy level leads router 3.1.1 to discover the destination is in area 3.2.x. In this case, duplicate event data packets are sent to global area 1.x.x, and area 3.2.x.

The calculations are similar to forward or route the event data packet to E1 target2. Router 2.1.1 recognizes that the event data packet has destinations both outside its area (in areas 3.x.x and 1.x.x, which are both accessible through router 3.1.1), as well as within area 2.x.x. Processing in the lower hierarchy levels enables router 2.1.1 to discover that the destination for area 2.x.x is within 2.1.x.x, and to E1 target2, which is directly connected to router 2.1.1. Thus, router 2.1.1 forwards the packet to the local target, as well as sending a copy to router 3.1.1.

FIGS. 8A-8B are block diagrams of an embodiment of HSPF in a network configuration having multiple data sources and multiple data targets. As with the description above with respect to FIG. 7, when multiple targets are added, the path calculation is repeated multiple times. The path calculation occurs at various levels, to forward the packet “towards” its destination (routing at the higher hierarchy levels), and eventually to forward the packet “to” its destination (routing at the lowest hierarchy level).

In one embodiment, all calculated routing trees are rooted at the originating router. Thus, similarly to how the path calculation is performed multiple times for the multiple separate targets, when a new event publishing source is added (e.g., the E1 source from FIG. 7 becomes E1 source1, and new source E1 source2 is added), the same calculations are repeated with respect to the multiple sources and the multiple targets. Once again, it will be understood that the same processing or calculations are performed for the various targets and sources, and at all the various hierarchy levels. It will be understood that while three hierarchy levels are shown, more levels of hierarchy could easily be added, and each level would perform the same processing as described herein.

In one embodiment, each data packet keeps an original source router, so each router may find a correct path for every data packet. In FIG. 8A, when a data packet with event E1 arrives to router 3.1.1, router 3.1.1 chooses different destinations depending on the event’s source router. If an event data packet originates from area 1.x.x, it will be forwarded to routers 2.1.1 and 3.1.2, but if the packet originates from 2.x.x, it will be forwarded to 1.2.2 and 3.1.2. In the drawing, the short-dashed lines show the routing of an event data packet from E1 source2, while the long-dashed lines show the routing for an event data packet from E1 source1. Observe that the short-dashed line (packets originating in 1.x.x) points from 3.1.1 to 3.2.x and 2.x.x, while the long-dashed line (packets originating in 2.x.x) points from 3.1.1 to 3.2.x and 1.x.x.

In FIG. 8B, the same routing is performed with the addition of another source, E1 source3. E1 is connected to router 3.2.2 of area 3.2.x. Once again, following the lines shows how the packets are routed through the network to reach each E1 target. E1 source2 again has a short-dashed line, E1 source1 has a long-dashed line, and E1 source3 has a line with mixed dashes and dots, which will be referred to as a “dotted” line for purposes of distinction.

In one embodiment, calculated paths are cached by each router. Therefore, when another event data packet from the same source is received, the list of destination routers is obtained from the cache, and does not need to be recalculated. The cache can be discarded or refreshed, for example, every time a link-state or event database is updated.

As before, each event data packet may include a source designation, which can enable the routers to determine how to route the packets toward the correct area. Alternatively, each router could have “directional” ports, such that packets received on ports “facing” one direction are forwarded from ports “facing” the other direction towards any targets on the opposite facing ports. As another alternative, each packet could simply be forwarded in all directions, where receiving routers determine if the packet is known, in which case the packet can be discarded. Such an approach is less desirable in a scenario where bandwidth conservation is more significant than reducing caching.

FIG. 9 is a block diagram of an embodiment of a network adjusting to network problems with HSPF. System 200 as depicted in FIG. 9 assumes the multi-target, multi-source scenario of FIG. 8B. However, one of the routers is presumed to become non-operational. If a router becomes non-operational, the link-state information of the router’s neighbors will be updated (e.g., via a heart message) and distributed across the network. All caches on affected routers will be discarded and new paths will be calculated. Since distribution of link-state information incurs some delay, for some time period after a router fails, individual events may be lost.

For example, assume router 3.1.1 becomes non-operational. The router could become non-operational due to a failure of hardware or software, or due to a human action or error. The removal of router 3.1.1 affects all clients. In one implementation, normal operation may be restored in approximately two minutes. The delay consists of several smaller delays: 1) a timeout for dead router detection (which may take roughly 60 to 90 seconds); 2) a link-state information propagation delay (which may take roughly 0 to 15 seconds per level); and, 3) the network trip time (which depends on physical network characteristics).

When router 3.1.1 becomes unavailable, all routes for the event data packets must be rerouted, because router 3.1.1 had a key role in routing the packets. It will be understood that area 3.1.x, as well as area 3.x.x, need to select a new router to represent the area. If the new router is selected based on smallest router ID, consistent with the example above, the new router to be selected is 3.1.2. With the updated configuration, the event data packets from E1 source2 will be forwarded to areas 3.x.x and 2.x.x via router 1.2.3, which will also receive event data packets from E1 source1 and E1 source3 to send to E1 target1. Additionally, the link between
3.2.1 and 2.2.3 will be utilized instead of the (missing) link between 2.1.1 and 3.1.1 to exchange packets between areas 3.x.x and 2.x.x.

[0091] FIG. 10 is a flow diagram of an embodiment of a process for updating network configuration with HSPF. Flow diagrams as illustrated herein provide examples of sequences of various process actions. Although shown in a particular sequence or order, unless otherwise specified, the order of the actions can be modified. Thus, the illustrated implementations should be understood only as an example, and the process can be performed in a different order, and some actions may be performed in parallel. Additionally, one or more actions can be omitted in various embodiments of the invention; thus, not all actions are required in every implementation. Other process flows are possible.

[0092] An administrator configures a network with nodes connected to other nodes within the network, 1002. An administrator configures a network topology with the nodes, grouping the nodes into areas and sub-areas, each having one or more nodes, 1004. An administrator may assign a node ID to each node, 1006, where the node ID is based on the association of the node with its areas and sub-areas. These operations may all be considered configuration activities, which do not necessarily affect the uniqueness of the runtime operations described.

[0093] During runtime of the network, each node discovers its links, both to neighbor nodes within the same area, and links to external nodes, 1008. The discovery of neighbors can occur dynamically throughout runtime of the network (e.g., through status updates), as well as in an initialization of the node and/or the network. Each node generates or updates local link-state information based on the discovery of links, 1010.

[0094] Nodes distribute their link-state information with other nodes having the same level or hierarchy via reliable flooding, 1012, and the areas likewise distribute the link-state information to areas having the same level of hierarchy via the same reliable flooding, 1014. Nodes also discover clients (e.g., data targets and data sources) directly connected to the nodes, 1016. The nodes and areas likewise distribute the client connection information throughout the network, 1018. Each node stores the link-state and client connection information for itself (the local node), as well as for other nodes and areas based on the received flooded information, 1020. The stored information is used to determine how to route information as described in more detail below with respect to FIG. 11.

[0095] FIG. 11 is a flow diagram of an embodiment of a process for applying HSPF in data routing in a network. Similarly to what is described with respect to FIG. 10, an administrator configures network connections among nodes, and configures a network topology by associating nodes with particular areas, 1102. In one embodiment, the association of a node with a particular area is based on the assignment of a node ID, which indicates the area(s) to which the node belongs, as well as its identifier for the local area to which it belongs. The node ID may be separate from an IP address assigned to a node within the network.

[0096] Each node discovers its links and generates link-state information, 1104, which is distributed throughout the layers of the network, 1106. Each node updates its local link-state information with information provided as data is distributed throughout the network, 1108. Each node also discovers and distributes data source and data target connection information, which is flooded through the layers of the network, 1110.

[0097] Each node calculates routing paths based on the link-state information and generates routing tables, 1112. The nodes route queries for data to data sources that have the requested data, and route data to query sources that request the data, based on the routing tables, 1114. The nodes dynamically update the network view, including link-state information and client information, as the changes occur in the network, which changes are propagated through the network, 1116. Thus, each node can maintain a consistent view of the network and how to route data.

[0098] Various operations or functions are described herein, which may be described or defined as software code, instructions, configuration, and/or data. The content may be directly executable ("object" or "executable" form), source code, or difference code ("delta" or "patch" code). The software content of the embodiments described herein may be provided via an article of manufacture with the content stored thereon, or via a method of operating a communication interface to send data via the communication interface. A machine readable storage medium may cause a machine to perform the functions or operations described, and includes any mechanism that stores information in a form accessible by a machine (e.g., computing device, electronic system, etc.), such as recordable/non-recordable media (e.g., read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, flash memory devices, etc.). A communication interface includes any mechanism that interfaces to any of a hardwired, wireless, optical, etc., medium to communicate to another device, such as a memory bus interface, a processor bus interface, an Internet connection, a disk controller, etc. The communication interface can be configured by providing configuration parameters and/or sending signals to prepare the communication interface to provide a data signal describing the software content. The communication interface can be accessed via one or more conduits or signals sent to the communication interface.

[0099] Various components described herein may be a means for performing the operations or functions described. Each component described herein includes software, hardwore, or a combination of these. The components can be implemented as software modules, hardware modules, special-purpose hardware (e.g., application specific hardware, application specific integrated circuits (ASICs), digital signal processors (DSPs), etc.), embedded controllers, hardwired circuitry, etc.

[0100] Besides what is described herein, various modifications may be made to the disclosed embodiments and implementations of the invention without departing from their scope. Therefore, the illustrations and examples herein should be construed in an illustrative, and not a restrictive sense. The scope of the invention should be measured solely by reference to the claims that follow.

What is claimed is:

1. A method comprising: discovering links at routers of a distributed network, where the routers are logically, hierarchically grouped in areas, where the links include links to peer neighbor routers for which a direct connection exists, and which are in the same area, and links to routers in other areas;
distributing link-state information from each router to peer neighbor routers, where each router updates local link-state information to represent the links of its peer neighbor routers;

distributing link-state information between areas to peer neighbor areas that exist at a same level of hierarchy, where each router in the areas updates local link-state information to represent links of the peer neighbor areas, where each area outside of a router’s area is represented as a node within the router’s local link-state information;

and storing link-state information at each router to be used to determine routing paths for data distribution through the distributed network.

2. The method of claim 1, wherein the areas comprise one of separate companies, departments of an enterprise, or business organizations, and routers are logically grouped based on company, department, or business organization, respectively.

3. The method of claim 1, wherein the areas comprise separate geographic locations, and logically group routers based on geography.

4. The method of claim 1, wherein the routers are nodes in an eventing network.

5. The method of claim 4, wherein the eventing network parses queries into component parts and routes each component part in accordance with the routing paths toward a data source to satisfy a query.

6. The method of claim 1, wherein distributing link-state information between areas further comprises:

selecting a router to represent each area to other areas based on a node identifier of the routers within the areas.

7. The method of claim 6, wherein selecting the router based on the node identifier further comprises:

selecting the router based on which router has a smallest node identifier within the area.

8. The method of claim 6, wherein distributing link-state information between areas further comprises:

a router not selected to represent the area waiting for the selected router to distribute the link-state information for the area; and

the router not selected to represent the area distributing the link-state information to other areas if the selected router fails to distribute the link-state information within a period of time.

9. The method of claim 1, further comprising:

subscribing a client as a data target at a router;

distributing client subscription information from the router to peer neighbor routers, where each router stores client subscription information to represent the client subscriptions of its peer neighbor routers; and

distributing client subscription information between areas to peer neighbor areas that exist at a same level of hierarchy, where each router in the area stores client subscription information to represent the client subscriptions of its peer neighbor areas.

10. The method of claim 1, further comprising:

subscribing a client as a data source at a router;

distributing client subscription information from the router to peer neighbor routers, where each router stores client subscription information to represent the client subscriptions of its peer neighbor routers; and

distributing client subscription information between areas to peer neighbor areas that exist at a same level of hierarchy, where each router in the area stores client subscription information to represent the client subscriptions of its peer neighbor areas.

11. A computer readable storage medium having content stored thereon to provide instructions, which when executed, cause a processor to perform operations, including:

discovering links from a router to all routers directly connected to the router in a network having routers hierarchically grouped in areas, including discovering links to peer neighbor routers for which a direct connection exists, and which are in the same area, and links to routers in different areas;

generating local link-state information at the router to indicate the discovered links;

distributing the local link-state information from the router to the peer neighbor routers;

receiving link-state information for peer neighbor routers and areas;

updating the local link-state information to indicate links of the peer neighbors and areas, including representing the areas as individual nodes; and

generating routing paths based on the link-state information.

12. The computer readable storage medium of claim 11, wherein the areas comprise one or more of separate companies, geography, business organizations, departments, industry groups, business partners, or a combination of these.

13. The computer readable storage medium of claim 11, wherein the content to provide instructions for distributing link-state information between areas further comprises content to provide instructions for:

selecting a router to represent each area to other areas based on a node identifier of the routers within the areas.

14. The computer readable storage medium of claim 11, further comprising content to provide instructions for:

detecting that a directly connected router is unavailable;

updating the local link-state information to reflect the unavailable router;

distributing the updated local link-state information; and

generating updated routing paths based on the updated link-state information.

15. The computer readable storage medium of claim 14, wherein the content to provide instructions for detecting that the router is unavailable further comprises content to provide instructions for:

determining that the directly connected router has not sent data or a heartbeat message within a threshold amount of time.

16. The computer readable storage medium of claim 11, wherein the content to provide instructions for distributing the local link-state information comprises content to provide instructions for:

flooding the network with link-state information to multiple peer neighbor routers.

17. The computer readable storage medium of claim 16, wherein the content to provide instructions for flooding the network with link-state information comprises content to provide instructions for:

sending the link-state information to all neighbor nodes in the network.

18. A router in a distributed network, comprising:

network ports to connect the router to the network, the router to send communications over the ports to the other routers;
a memory device to store a link-state database to store link-state information for the router, the link-state information including information indicating links from the router to peer neighbor routers for which a direct connection exists and which are in the same area, and links to routers in other areas, and link-state information for the peer neighbor routers, including information about links external to the area, where other areas are represented as individual nodes; and a hierarchical shortest path first (HSPF) network stack to access the link-state database and calculate routing information based on the link-state information, to send the communications over the ports to other locations in the network, wherein calculating the routing information is performed the same for each hierarchical level of the network based on knowledge of peer nodes for the hierarchical level as indicated in the link-state information.

19. The router of claim 18, wherein the areas comprise one or more of separate companies, geography, business organizations, departments, industry groups, business partners, or a combination of these.

20. The router of claim 18, wherein the HSPF network stack is to further:
determine if the router is to be selected to represent the area of which the router is a part to other areas based on a node identifier of the router, and node identifiers of other routers within the area.

* * * * *