such as TCP/IP or connectionless protocol such as UDP, though TCP/IP is preferable due to its guaranteed delivery.

Each of the network managers **205**, **212** is preferably a SunNet Manager (version 2.2.2 or later) running on Solaris 2.x or Solaris 1.1 operating system, both the SunNet Manager and Solaris are available from Sun Microsystems, Inc. The SunNet Manager is described in detail in SunNet Manager Reference Guide, by Sun Microsystems, Inc. 1994 and SunNet Manager Programmer's Guide, by Sun Microsystems, Inc. 1994, both of which is hereby incorporated by reference. Although the network managers **205**, **212** are preferably conventional, according to the invention, the network managers **205**, **212** operate in new ways, particularly with respect to interaction with the sender process **206** and the receiver process **214**

FIG. **3** is a detailed block diagram of a network management system **300** in accordance with an embodiment of the invention. The network management system **300** includes a sending station **302** and a receiving station **304**. The sending station **302** includes a local network management (NM) console **306**, a management database **308**, a NM log file **310**, and an event dispatcher **312**. The event dispatcher **312** receives event and traps from agent **314** over link **316**, from agent **318** over link **320**, and from agent **322** over link **324**. The agents **314**, **318** and **322** are used to monitor network elements of a network (not shown). The local network management (NM) console **306**, the management database **308**, the NM log file **310**, and the event dispatcher **312** form a local NM system such as provided by a SunNet Manager console. To implement the sharing of network management information in accordance with the invention, the sending station **302** further includes a sender process **326**, an authorization list **328** and filter files **330**.

The receiving station **304** includes a local NM console **332**, a management database **334**, and an event dispatcher **336**. The local network management (NM) console **332**, the management database **334**, and the event dispatcher **312** form a local NM system such as provided by a SunNet Manager console. To implement the sharing of network management information in accordance with the invention, the receiving station **304** further includes a receiver process **338** and a registration list **340**.

When the receiving station **304** desires to receive information from the sending station **302**, the registration list **340** is modified to contain information identifying the particular sending station from which network management information desired. The registration list **340** preferable identifies a host machine (i.e., sending station), filter file, and database. Hence, the registration information contained in the registration list **340** specifies the particular database involved when sending stations have multiple databases to choose from, the appropriate event forwarding criteria (i.e., filter file) to be used by the sender process, and the sending stations to which a receiving station is to register. It is assumed here that the receiving station **304** desires network management information from the sending station **302**.

Once the receiver process **338** is initiated at the receiving station **304**, the receiver process **338** requests connection to the sender process **326** of the sending station **302** using a connection request **342**. Upon receiving the connection request, the sender process **326** then uses the authorization list **328** to determine whether the receiving station **304** (or the receiver process **338**) is authorized to receive network management information from the sender process **326**. The receiver process **338** is able to successfully register with the sender process **326** and access a given local database, only if that receiving station **304** is authorized to register with the sender process **326** and access the specified database. The authorization list **328** is used to determine whether the receiver process **338** is so authorized. Hence, by using the authorization list **328**, unauthorized eavesdropping can be prevented. For performance reasons, it is preferable that a separate child sender process be spawned for each receiving station that registers with the sending station **302** and that a separate child receiver process be spawned for each sending station that the receiving station **304** desires to receive network management data from.

If the receiving station **304** is authorized, the sender process **326** registers with the event dispatcher **312** so as to receive all the network management information that the event dispatcher **312** receives due to local network management at the sending station. Here, the network management information includes events and traps from the agents **314**, **318** and **322** as well as database traps from the management database **308**.

The sender process **326** thereafter filters the network management information (e.g., event and traps) received from the event dispatcher **312** using a particular one of the filter files **330** as identified by the receiver process **338** during connection. The filtering operation discourages unnecessary forwarding of network management information. The network management information remaining after the filtering is referred to as filtered data. The filtered data can be grouped into two classes of data, namely, database traps and non-database traps and events. The database traps are forwarded to the receiver process **338** using a message **344**.

The receiver process **338** then supplies the database traps to the management database **334** via a link **346**. The management database **334** can then be modified in accordance with the database trap forwarded by the sender process **326**. Thereafter, the local NM console **332** can be updated in accordance with the updates to the management database **334** using link **348**. Typically, the local NM console **332** will update its view based on the changes to the management database **334**. The non-database traps and events are forwarded by the sender process **326** to the event dispatcher **336** directly using a message **350**. The event dispatcher **336** can then update the local NM console **332** over link **352**.

Preferably, as database (topology) traps are received from the sender process **326**, the receiver process **338** uses the database Application Programming Interface (API) of the local NM console **332** to update the local database **334** to reflect the topology information received from the sending station. The updating of the database **334** is preferably achieved as follows. When the receiver process **338** receives a database (topology) trap from the sender process **326**, the receiver process **338** reads the local database **334** to determine if the element associated with the database trap already exists. If the element does not exist in the local database **334**, then it is added to the local database **334**. If an element already exists in the local database **334**, the receiver process **338** determines whether the forwarded features of the element match those already attributed to the element. If the element's characteristics in the local database **334** already match the features reported in the database trap, the database trap is ignored. If the element's characteristics in the local database **334** differ from the forwarded characteristics, the event information in the local database **334** is changed to match the information forwarded from the sending process **326**. Of course, the local database **334** can protect its elements (records) from being overwritten by configuring the sender process **326** such that it does not forward database traps for those elements not to be overwritten.

If view membership information is passed (such as with a Background type database trap), elements are added to the specified views if the views already exist in the local database **334**. If the view is not yet present in the local database **334** at the receiving station **304**, then the element can be added to a temporary holding area view. The system can also be configured so that the elements (components, views, etc.) that are added to the local database **334** can be left in the holding area view and even grouped in a different holding area view for each sending station.

On each receiving station, the local event dispatcher **336** preferably receives event-related traps from the sender process **326** of the sending station **302**. The event dispatcher **336** then forwards the information to the local NM console **332**. The event dispatcher **336** on the receiving station does not receive the NM database traps forwarded by the sender process because such topology information is sent to the receiver process **338** on the receiving station **304**.

Preferably, the event dispatcher **312** receives events produced in response to event requests launched by the NM console **306**. The event dispatcher **312** forwards the events to the sender process **326**. The sender process reformats all traps and NM events into NM traps before forwarding them to the event dispatcher **336** on the receiving station **304**. The event dispatcher **304** passes the traps to the local NM console **332**. A NM event is converted into a trap before being sent to a receiving station because the local NM (SunNet) console at the receiving station will ignore NM events that it cannot match to one of its own event requests. The reformatting, for example, adds an indicator to the event information so that the receiving station can easily identify that the event or trap was forwarded from a sending station.

Each of the local NM consoles **306**, **332** of the network management system **300** preferably has a graphical user interface (GUI) that displays on a display screen. The GUI enables an end-user to configure operations of the network management system **300**. For example, an end-user preferably modifies the registration list **340** using a GUI.

FIGS. **4**A and **4**B are flow diagrams illustrating receiver processing **400** in accordance with an embodiment of the invention. The receiver processing **400** is carried out by a receiver process such as the receiver process **214** of FIG. **2** or the receiver process **338** of FIG. **3**.

The first operation of the receiver processing **400** is to initialize **402** the receiver process. Then, a registration file is read **404**. Next, the receiver processing **400** requests **406** connection of the receiving station **204**, **304** to the sending station **202**, **302**. More particularly, the connection requested **406** is of the receiver process to the sender process. Next, a decision **408** is made based on whether a connection has been established. If no connection has been established, a decision **410** determines whether a time-out has occurred. If a time-out has not yet occurred, the processing returns to the decision block **408**. Otherwise, if a time-out has occurred, the user is notified **412** of the failure to connect and the receiver processing **400** ends.

On the other hand, when the decision block **408** determines that the connection has been established, a decision **414** is made based on whether data (i.e., network management information) has been received from the sender process. If data has been received, the data received is processed **416** in the local network manager **212**, **332**. If, on the other hand, data is not received from the sender process, a decision **418** is made based on whether a time-out has occurred. If a time-out has not yet occurred, the processing returns to repeat the decision block **414** and subsequent blocks. If the time-out has occurred, the receiver processing **400** skips block **416**. Hence, following block **416** or following the decision block **418** in the case of a time-out, a decision **420** is made based on whether user input has been received. If no user input has been received, the receiver processing **400** returns to repeat the decision block **414** and subsequent blocks. On the other hand, if user input has been received, a decision block **422** determines whether the user has requested disconnection. If the user has requested disconnection, the receiver processing **400** requests **424** disconnection from the sender process **206**, **326**. Subsequent to the block **424**, the receiver processing **400** ends. On the other hand, when the decision block **422** determines that disconnection has not been requested by the user, the receiver processing **400** performs **426** other actions as requested by the user. Examples of other actions requested by the user include manual synchronization and configuration of new connection. Following block **426**, the receiver processing **400** repeats the decision block **414** and subsequent blocks.

FIG. **5** is a flow diagram of sender processing **500** in accordance with an embodiment of the invention. The sender processing **500** is carried out by a sender process such as the sender process **206** of FIG. **2** or the sender process **326** of FIG. **3**.

The sender processing **500** is initiated in response to the first connection request from a receiving station. The sender processing **500** first initializes **502** the sender process. During initialization, various internal data structures and data states are initialized. Next, the sender processing **500** reads **504** the authorization list **208**, **328**. The sender processing **500** then registers **506** the sender process with the network manager **205**, **312** (event dispatcher). Next, a decision block **508** determines whether a connection request has been made by a receiver process. If a connection request has been received at the sender process, a decision block **510** determines whether the receiver process (or receiving station) is authorized to receive data from the sending station. The decision block **510** determines whether the receiver process is authorized using the authorization list **208**, **328**.

If the receiver process is authorized, connection of the receiver process to the sender process is established **512**. When the connection is requested, the receiver process preferably also passes the sender process a database name and the filter file name. The database name selects the particular database from which to forward database traps when there are multiple databases at the sending station. The filter file name specifies the filter file that contains the filter table with the desired filter criteria. Otherwise, if the receiver process is not authorized, the user is notified **514** that the receiver process is unauthorized. The connection request will fail if the receiver process passes the name of a non-existent filter file or a database that the receiver process is not authorized to access the selected database. Following blocks **512** or **514** in the case of a receiver connection request, the sender processing **500** returns to repeat the decision block **508** and subsequent blocks.

On the other hand, when a connection request has not been received, the decision block **508** causes the sender processing **500** to proceed to decision block **516**. The decision block **516** determines whether data has been received from the local network manager **205**, **312** (event dispatcher). If no data has been received from the network manager **205**, **312**, the sender processing **500** returns to repeat the decision block **508** and subsequent blocks. Otherwise, when data has been received from the network manager **205**, **312**, the sender processing **500** applies **518**

filtering. The filtering operation is discussed in more detail below with reference to FIG. **6**. Generally speaking, the filtering operation filters the data (i.e., network management information) so that only the data requested by the receiver process (receiving station) remains after the filtering operation. Thereafter, a decision block **520** determines whether there is data remaining after the filtering has been applied **518**. If there is no data remaining, then there is no data to forward to the receiver process; hence, the sender processing **500** returns to repeat the decision block **508** and subsequent blocks. When there is data remaining after the filtering is applied **518**, the filtered data is forwarded **522** to the appropriate receiver process. The appropriate receiver process is the receiver process that has requested the data be transmitted thereto. Then, following block **522**, the sender processing **500** returns to repeat decision block **508** and subsequent blocks. When all the receiver processes that have been registered with the sender process disconnect, then the sender process unregisters from the network manager **205**, **312** (event dispatcher) and exits thereby ending the sender processing **500**.

A receiving station can choose to register with multiple sending stations. Also, multiple receiver processes on various receiving stations can register with a sender process on a given sending station and the filter criteria can be configured separately for each receiving station. Preferably, for each pair sending and receiving stations, a dedicated pair of child processes (i.e., a child sender process and a child receiver process) are used to manage the processing.

FIG. **6** is a flow diagram of filter processing **600** in accordance with an embodiment of the invention. The filter processing **600** is the processing preferably carried out by block **518** in FIG. **5**.

The filter processing **600** begins with a decision block **602** which determines whether the host name of the data received at the sender process (from the network manager **205** or the event dispatcher **312** thereof) matches the host name listed in a filter table residing in or identified by the filter files **210**, **330**. If not, a decision block **604** determines whether or not the component of the data received at the sender process matches the component listed in the filter table. Preferably, the filter table and/or filter file for a particular receiver process (receiving station) are selected by the receiver process when the connection request is accepted. If the decision block **604** determines that the component of the data received does not match the component in the filter table, the data received is dropped **608** and filter processing **600** ends.

On the other hand, following the decision block **602** when the host name matches or following the decision block **604** when the host name does not match but the component does match, a decision block **606** determines whether the priority level of the data received (data priority) is greater than or equal to a priority level identified in the filter table (filter priority level). Typically, the priority levels would be low, medium and high. If the data priority of the data received is less than the filter priority level, then the data is dropped **608** and the filter processing **600** ends. On the other hand, if the data priority exceeds or is equal to the filter priority level, the proper attributes are passed **610**. The attributes to be passed are identified in the filter table. Thereafter, the priority level of the data received is optionally adjusted **612** in accordance with information identified in the filter table. It is preferable to adjust the priority of the data received to low priority because normally the shared network management information has a low priority regardless of the fact that the priority might have been high at its local network manager. Following block **612**, the filter processing **600** ends.

To implement the filter processing **600**, each filter table residing in or identified by the filter files **210**, **330** preferably includes the following six fields.

1. Type—This determines the primary selection criterion for processing a trap or event. The type can be: hostname, component, or default. The type default filter is used if an event or trap does not match the type criterion of any of the other filters. If the type is hostname, this selection criterion is satisfied if the value of the name field matches the host name associated with the event or trap. If the type is component, this selection criterion is satisfied if the component type in the name field (for example, component.router) matches the component type associated with the event or trap.

2. Name—If the type is hostname, the name field value is either an IP address or the hostname of a glyph in the database. If the type is component, the name field value must be the name of a component type (for example, component.router), view type (for example, view.subnet, or bus type (for example, bus.rs232). 3. Priority—This field specifies the lowest priority event or trap that the filter will process. For example, if "low" is specified, then events of any priority will satisfy this criterion. The filter will be selected if the event or trap satisfies both the name field criterion and the priority field criterion.

4. New Priority—If a value is specified here, this is the priority of the trap when forwarded.

5. Action—This field specifies whether an event or trap selected by the filter should be either forwarded or ignored.

6. DB Template—This field specifies the file name of the database template that should be used in selecting information from the database when forwarding NM database traps.

The filter table (via the DB Template field) specifies the database template file to determine the topology information that should be forwarded for NM database traps that match the Name field of the filter table. Each filter in the filter table can specify a database template file used for elements that match the selection criteria of the filter. The database template files are only accessed in response to NM database traps because non-database traps or events do not cause the sender process to forward topology information to the remote receiver process. The database template files specify the information (e.g., agents, attributes, color, connections, and membership) to be passed or forwarded for each type of trap (i.e., add, create, change, delete, load and background.)

Preferably, the network manager **202** or the local NM console **306** generate six different types of NM database traps. These representative types of NM database traps are as follows:

1. Add—Generated when a new element is added to the database via, for example, the database Application Programmatic Interface (API).

2. Background—Generated when a background image is added to a view.

3. Create—Generated when a new element is created via the NM console.

4. Change—Generated when attributes of the element (such as the agents list or the screen coordinates) are changed.

5. Delete—Generated when an element is deleted from the database.

6. Load—Generated when a new management database is loaded.

When the sender process **326** receives NM database traps from the event dispatcher **312**, the sender process **326** uses

the filter file **330** specified for the receiving station **304** to determine which database template file to use in processing the database trap. The DB Template field in the filter table contains the database template file name. The database template file allows you to specify additional topology information that should be forwarded. Because the filter table allows you to specify different DB Template files in each filter, an end-user can use the filter selection criteria to specify different types of topology information to forward for different devices (by hostname or element type.

A representative DB Template file has the format shown in Table 1 below.

TABLE 1

Database Template File Format

Trap Type	Keywords (one or more can be specified)
Add	membership, color, agents, attributes, connections, drop
Create	membership, color, agents, attributes, connections, drop
Change	membership, color, agents, attributes, connections, drop
Delete	drop
Load	membership, color, agents, attributes, connections, drop
Background	drop

The keywords in the above table determine the forwarded content for each of these four trap types. The keywords have the following interpretation:

1. Membership—If specified, the view name and coordinates will be passes for each view the element belongs to.
2. Color—If specified, the element's RGB values are passed.
3. Agents—If specified, the name of each agent selected on the element's properties sheet is passed, and the proxy system name is also passed for each specified proxy agent.
4. Attributes—If specified, the entries for each attribute specified in a schema file for that element (e.g., IP address or contact name) are passed.
5. Connections—If specified, information about the simple connections to the element will be forwarded.
6. Drop—If specified, traps of this type will not be forwarded.

Preferably, a GUI is used to configure a sender and receiver processes of the network management system according to the invention. By using a GUI, the end-user is able to configure the information sharing provided by the invention with minimal effort. The GUI would allow the user to easily define the following: (i) the list of remote receiving stations authorized to register with the local sender process and the databases that the receiving stations are authorized to access, (ii) filter files and database templates that determine the event and topology information forwarded by the sender process, and (iii) the list of remote management stations the receiver process will attempt to register within the database instance and the filter file that it will request at the sending station.

Preferably, all traps directed to the receiver process are sent to the receiver's transient RPC number. The RPC number is provided to the sender process as part of the registration process. The basic format of the trap message is preferably a series of "attribute=value" pairs. For a trap of type "changed" generated against device "andrew" this would look like this:

```
coop_forwarded_by = lobsta
changed = andrew
type = component.ss1
IP_Address =
User =
Location =
Description =
SNMP_RdCommunity =
SNMP_WrCommunity =
SNMP_Vendor_Proxy =
SNMP_Timeout = 0
red = 100
green = 220
blue = 0
view = Home
Xpos = 160
Ypos = 240
X1pos = 0
Y1pos = 0
connected = lobsta
agent = hostif
agent = hostperf:andrew
```

These fields of the trap message could be analyzed as follow

- The first field should be "coop_forwarded_by". Each sender process adds one of these lines at the head of each report. The value of this attribute should be the name of the local host forwarding the report. This is used by the receiver process to prevent message loops, as well as by users, to determine who originated each message.
- The second field is the database trap type. The attribute name in this field is usually one of the following: changed, deleted, added, loaded, or created. The value is the name of the element that this trap is being generated for.
- The next attribute is "type." The value of this attribute is the local NM (e.g., SunNet Manager) component type for the element as represented in the local database.
- The next series of fields—in this example, the fields from IP_Address to SNMP_Timeout—are determined by the schema definition for the component type. These fields can be any set of attributes as determined by the schema.
- The next three attributes specify the glyph color, in terms of red, green, and blue values. The range of values should be an integer from 0 to 255.
- Next there is a list of the views to which the element belongs. These views are listed with one "view=viewname" pair for each view the element should be displayed in. Each "view=viewname" pair is to be followed by the position information (X and Y coordinates) that defines the position of that element within that view. The values Xlpos and Ylpos are only used for elements of type bus.ethernet.
- The "connected" attribute indicates what elements this element has been connected to. These connections are "simple" connections, not manageable components.
- Finally, a list of the element's agents is added to the end of the report. Each entry in this list is of the form: "agent=agentname:proxy". In such an entry agentname:proxy represents the name of the agent that has been enabled, followed by information defining which host should be used as a proxy system. If an agent is not a proxy agent, only the agent name is provided, with no colon(":") or proxy name.

Another feature of the invention concerns the synchronization of the databases at the sending station and the

receiving station. When the sending station is forwarding network management information to the receiving station in the form of database traps concerning topology of the network being monitored by the sending station, it is desirable that the portion of the database of the receiving station be synchronized with the database of the sending station. If the databases are not synchronized, then a network administrator at the receiving station would visualize a view based on stale data. Also, traps for devices not in the database of the receiving station will be ignored, thus leading to incomplete monitoring of the remote network. The synchronization of the databases can be achieved whenever the network management system is started-up, or automatically as a user configures, or even manually whenever the site manager (network administrator) so requests. The synchronization provided by the invention is automatic once a request for synchronization is made by the receiving station. The synchronization operation automatically operates to synchronize the topology data contained in the database of the receiving station with the corresponding topology data contained in the database of the sending station.

FIG. **7** illustrates a flow diagram of synchronization processing **700** in accordance with an embodiment of the invention. First, the receiver process **214**, **338** sends a synchronization request to the sender process **206**, **326**. Then, in response to the synchronization request, the sender process **206**, **326** generates **704** database traps at the sending station. Next, the database traps are filtered **706** at the sender process **206**, **326**. The filtering **706** corresponds to the database trap filtering (block **610**) performed in accordance with the filter processing **600** shown in FIG. **6**. Next, the filtered database traps are forwarded **708** to the receiver process **214**, **338**. Thereafter, the receiver process **338** causes the database at the receiving station **204**, **304** to be updated **710** in accordance with the filtered database traps. The updating **710** in effect synchronizes the database at the receiving station **204**, **304** with the database at the sending station **202**, **302** to the extent permitted (via filtering) by the end-user.

FIGS. **8**A and **8**B are flow diagrams of initialization processing **800** according to a more detailed embodiment of the invention. The initialization processing **800** synchronizes databases of the sending station and the receiving station upon initiation of the network management system. Because the initialization processing **800** is done upon initialization, the initialization processing **800** additionally includes operations performed by the initialization operation **402** in FIG. **4**A and the initialization operation **502** in FIG. **5**.

The first operation of the initialization processing **800** is to invoke **802** a parent process at the sending station and the receiving station. Next, a decision **804** is made based on whether the initialization sequence is configured to request a connection at start-up. If not, the initialization processing **800** ends because in this case a connection is not requested upon initialization. On the other hand, when the connection is initially requested at start-up, the initialization processing **800** performs other operations to carry out the connection as well as the synchronization operation. Namely, the initialization processing **800** invokes **806** a child receiver process at the receiving station **204**, **304**. Here, the receiver process **214**, **338** spawns the child receiver process to interact with a particular sending station **202**, **302**. Next, the child receiver process sends **808** a synchronization/registration request to the sender process **206**, **326**. The synchronization/registration request includes a connection request as discussed above together with a request for initial synchronization.

In response to the synchronization/registration request, the sender process **206**, **326** invokes **810** a child sender process at the sending station **202**, **302**. The child sender process is used to forward the filtered data through the child receiver process via the event dispatcher **312**. By spawning child processes the sender process and receiver process effectively off loads processing tasks to the child processes such that they can manage incoming requests or connections and disconnections without being blocked by other processing operations.

Next, the sender process **206**, **326** invokes **812** a child synchronization process to generate database traps and to deliver the database traps to the sender process. In order to generate the database traps needed at the receiving station, the sender process invokes **812** the child synchronization process which is a temporary process that reads through the entire database at the sending station **202**, **302** and generates the database traps as it steps through the database. The child synchronization process then delivers the database traps to the sender process **206**, **326** and then exits. Next, the sender process **206**, **326** forwards **814** the database traps to the child sender process. At the child sender process, the database traps are filtered **816**. The filtering performs the same operations as noted by block **610** in FIG. **6**. After the database traps are filtered so that only the database traps the receiving station **204**, **304** desires remain, then the filtered database traps are forwarded **818** to the child receiver process of the receiving station **204**, **304**. Thereafter, the database at the receiving station is updated **820** in accordance with the filtered database traps. The updating **820** is carried out as discussed above.

To provide accurate synchronization between two network management stations, it is preferable to uniquely identify the data which is being synchronized and what has been synchronized earlier. A property can be added to each of the databases to provide the identification of the data being synchronized. The property has a value which uniquely identifies each object in the database. Each component will have the property and the value of the property will preferably take the form of "Sender_hostname": filter_filename: database_name".

Synchronization is always initiated by the receiver process. If the receiver process is currently synchronizing when another synchronization request is initiated (i.e., a manual synchronization request) toward the same connection, the current synchronization operation is preferably halted and a new synchronization process is started. The processing associated with halting the current synchronization operation and starting a new operation is as follows. First, the receiver process can tell the child receiver process to send a delete request to the sender process. The child receiver process then terminates itself. As a result of the delete request, the sender process terminates the child sender process. The receiver process then invokes a new child receiver process which deletes all the elements from the database at the receiving station having the properties for that connection. The new child receiver process then sends a synchronization request to the sender process. The sender process then invokes a new child sender process and a new child synchronization process. The new child synchronization process generates the database traps and delivers them to the new sender process which in turn forwards them to the child sender process. The child sender process then filters the database traps and forwards the filtered database traps to the new child receiver process. Finally, the new child receiver process will add the objects to the database at the receiving station with the appropriate value of the new property.

If the site is synchronizing daily with two other sites, the value of the property for an object (e.g., hostname X) preferably contains the sender's hostname with which the receiver synchronizes with first. Of course, the user can over-ride this option by manually deleting the object and re-synchronizing with the second site.

The receiver process operates to receive all the database traps from the sending station unless filtered out. After receiving the database traps this receiver process updates the database and the receiving station. With synchronization, the topology data stored in the database at the receiving station can be synchronized to the corresponding topology data stored in the database at the sending station. The filtering allows synchronization to take place for only certain types of database traps.

A graphical user interface (GUI) can be used to provide an end-user with a dialog box in which the user is able to select how and when the synchronization is to occur. For example, permission to delete an object from a database at the receiving station can be restricted to only those sites which are the source of the particular objects being deleted. Another example is that the user can select an option to synchronize at start-up which follows the processing described above with reference to FIGS. **8**A and **8**B. This type of synchronization initiation would occur on a normal start-up or after a crash of the system. The user can also select whether the synchronization of the databases shall occur automatically, and if so, how often (e.g., on a daily or weekly basis).

FIG. **9** is a diagram of a representative network arrangement **900** according to the invention. Usually, but not necessarily, the computer processes in accordance with the present invention and other computer processes are resident on one or more computers linked together by a network. The network may be the same network or a portion of the large overall network being managed by the network management system according to the invention. The network may take any suitable form. By way of example, the network arrangement **900** shown in FIG. **9** includes a first computer **902** which is coupled to a transmission line **904**. The network arrangement **900** further includes a server, router or the like **906** in addition to other computers **908**, **910**, and **912** such that data and instructions can be passed among the networked computers. The design, construction and implementation of networks will be familiar to those of skill in the art. A network management system according to the invention preferably resides and executes on networked computers such as the computers **902**, **908**, **910** and **912** and operates to manage the large overall network in a distributed manner while providing the ability to share critical network management information.

FIG. **10** is a block diagram of a representative computer **1000** suitable for use with the invention. The representative computer **1000** is suitable for use as computers **902**, **908**, **910**, and/or **912** of FIG. **9**. The computer **1000** includes a central processing unit (CPU) **1002** which is coupled bidirectionally with random access memory (RAM) **1004** and unidirectionally with read only memory (ROM) **1006**. Typically, the RAM **1004** is used as a "scratch pad" memory and includes programming instructions and data, including distributed objects and their associated code and state, for processes currently operating on the CPU **1002**. The ROM **1006** typically includes basic operating instructions, data and objects used by the computer to perform its functions. In addition, a mass storage device **1008**, such as a hard disk, CD ROM, magneto-optical (floptical) drive, tape drive or the like, is coupled bidirectionally with the CPU **1002**. Mass storage device **1008** generally includes additional programming instructions, data and objects that typically are not in active use by the CPU, although the address space may be accessed by the CPU **1002**, e.g., for virtual memory or the like. The computer **1002** may optionally include an input/output source **1010** that typically includes input media such as a keyboard, pointer devices (e.g., a mouse or stylus) and/or network connections. Additional mass storage devices (not shown) may also be connected to the CPU **1002** through a network connection. The computer **1000** further includes a display screen **1012** for viewing text and images generated or displayed by the computer system **1000**. The CPU **1002** together with an operating system (not shown) operate to execute computer code. The computer code may reside on the RAM **1004**, the ROM **1006**, or a mass storage device **1008**. The computer code could also reside on a portable program medium **1014** and then be loaded or installed onto the computer **1000** when needed. Portable program mediums **1014** include, for example, CD-ROMs, PC Card devices, RAM devices, floppy disk, magnetic tape.

Additional details pertaining to the invention can be found in "Cooperative ConSoleS™ 1.0 Administrator's Guide," available from Sun Microsystems, Inc. of Mountain View, Calif., which is hereby incorporated by reference.

The many features and advantages of the present invention are apparent from the written description, and thus, it is intended by the appended claims to cover all such features and advantages of the invention. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation as illustrated and described. Hence, all suitable modifications and equivalents may be resorted to as falling within the scope of the invention.

What is claimed is:

1. A network management system for sharing information between a plurality of distributed network managers, said system comprising:

a sending machine including at least

a first network manager for managing a first network, said first network manager receiving event and trap information from agents associated with the first network;

an authorization list containing information indicating whether receiving machines are authorized to receive the event and trap information; and

a sender process, operatively coupled to said first network manager, for receiving the event and trap information received by said first network manager;

a receiving machine including at least

a second network manager for managing a second network;

a receiver process for receiving the event and trap information; and

a registration list for identifying sender machines to which said receiving machine is to connect to receive event and trap information; and

a communication link connecting said sending machine and said receiving machine,

wherein said sender process forwards the event and trap information to said receiving machine if said authorization list authorizes said receiving machine to receive the event and trap information, and

wherein said receiver process forwards the event and trap information received from the sender process to said second network manager for processing thereof.

2. A network management system as recited in claim **1**, wherein said sender process filters the event and trap

information, and then forwards the filtered event and trap information to said receiving machine if said authorization list authorizes said receiving machine to receive the event and trap information.

3. A network management system as recited in claim 2, wherein said sending machine further includes a filter file containing filter tables, the filter tables are utilized by said sender process to filter the event and trap information.

4. A network management system as recited in claim 1, wherein said first network manager maintains a first database and said second network manager maintains a second database,

wherein said sender process receives a synchronization request from said receiver process and in response thereto generates a database trap for each record in the first database and then forwards the database traps to said receiver process, and

wherein said receiver process receives the database traps from said sender process and then synchronizes the second database to the first database in accordance with the database traps.

5. A system as recited in claim 1, wherein the second network is distinct from the first network.

6. A network management system for sharing information between a plurality of distributed network managers, said system comprising:

a first network manager for managing a first network;

a second network manager for managing a second network;

information sharing means for sharing information between said first network manager and said second network manager; and

an authorization list containing information indicating whether said second network manager is are authorized to receive the information from

wherein said information sharing means operates to forwards the information to said second network manager if said authorization list authorizes said second network manager to receive the information.

7. A system as recited in claim 6, wherein the second network is distinct from the first network, and wherein the information being shared includes event or trap information.

8. A system as recited in claim 7,

wherein the first network includes a first set of agents and the second network includes a second set of agents, and wherein each of the agents generates events or traps, and

wherein the information being shared between said first and second network managers includes one or more of the events or traps generated by the agents.

9. A system as recited in claim 6, wherein said information sharing means comprises:

filter means for filtering the information before being transferred to said second network manager; and

means for transferring the filtered information from said first network manager to said second network manager.

10. A system as recited in claim 9, wherein said first network manager maintains a first database of topology data for the first network and the second network manager maintains a second database of topology data for the second network, and

wherein said system further comprises means for automatically synchronizing topology data between said first and second databases.

11. A network management system for sharing information between a plurality of distributed network managers, said system comprising

a first network manager for managing a first network,

a second network manager for managing a second network; and information sharing means for sharing topology data between said first network manager and said second network manager, said first network manager maintains a first database of topology data for the first network and the second network manager maintains a second database of topology data for the second network, and said information sharing means automatically synchronizes topology data between said first and second databases.

12. A computer-implemented method for sharing network management data between first and second network managers, the first network manager locally managing a first network, and the second network manager locally managing a second network, said method comprising the steps of:

(a) connecting a receiver process associated with the second network to a sender process associated with the first network;

(b) receiving, at the sender process, network management data for the first network;

(c) forwarding the network management data from the sender process to the receiver process; and

(d) processing the network management data in the second network manager.

13. A computer-implemented method as recited in claim 12, wherein the network management data includes event and trap data.

14. A computer-implemented method as recited in claim 13, wherein the trap data includes one or both of database traps and non-database traps.

15. A computer-implemented method as recited in claim 12, wherein said method further comprises: (e) filtering, prior to said forwarding (c), the network management data at the sender process to produce filtered data for the receiver process.

16. A computer-implemented method as recited in claim 12,

wherein the first network manager maintains a first database and the second network manager maintains a second database, and

wherein said method further comprises: (e) automatically synchronizing the second database to the first database.

17. A computer-implemented method as recited in claim 16, wherein said synchronizing (e) comprises the steps of:

(e1) sending a synchronization request from the receiver process to the sender process;

(e2) generating database traps for the entire contents of the first database;

(e3) forwarding the database traps to the receiver process; and

(e4) updating the second database in accordance with the database traps.

18. A computer-implemented method as recited in claim 17, wherein said synchronizing (e) further comprises the step of: (e5) filtering the database traps prior to said forwarding (e3) and said updating (e4).

19. A computer-implemented method for sharing network management data between first and second network managers, the first network manager locally managing a first network, and the second network manager locally managing a second network, said method comprising the steps of:

(a) connecting a receiver process associated with the second network to a sender process associated with the first network;

(b) receiving, at the sender process, network management data for the first network;

(c) filtering the network management data at the sender process to produce filtered data for the receiver process;

(d) forwarding the filtered data from the sender process to the receiver process; and

(e) processing the filtered data in the second network manager.

20. A computer-implemented method as recited in claim **19**, wherein said filtering (c) comprises the steps of:

(c1) providing a filter table to the sender process; and

(c2) filtering the network management data at the sender process in accordance with the filter table.

21. A computer-implemented method as recited in claim **20**, wherein during said connection (a), the receiver process identifies the filter table to the sender process.

22. A computer-implemented method as recited in claim **20**, wherein the first and second networks include network elements and network domains, wherein said filtering (c2) filters out at least a portion of the network management data based on one of a type of network elements or a name of network domains.

23. A computer-implemented method as recited in claim **19**,

wherein the first network manager maintains a first database and the second network manager maintains a second database, and

wherein said method further comprises: (f) automatically synchronizing the second database to the first database.

24. A computer-implemented method as recited in claim **19**, wherein said synchronizing (f) comprises the steps of:

(f1) sending a synchronization request from the receiver process to the sender process;

(f2) generating database traps for the entire contents of the first database;

(f3) forwarding the database traps to the receiver process; and

(f4) updating the second database in accordance with the database traps.

25. A computer-implemented method as recited in claim **24**, wherein said synchronizing (f) further comprises the step of: (f5) filtering the database traps prior to said forwarding (f3) and said updating (f4).

26. A computer program product, comprising:

a computer usable medium having computer readable code embodied therein to implement sharing of network management data between first and second network managers, the first network manager locally manages a first network and the second network manager locally manages a second network, and

wherein said computer readable code comprises:

first computer readable program code devices configured to cause a computer to effect connecting a receiver process associated with the second network to a sender process associated with the first network;

second computer readable program code devices configured to cause a computer to effect receiving, at the sender process, network management data for the first network;

third computer readable program code devices configured to cause a computer to effect forwarding the network management data from the sender process to the receiver process; and

fourth computer readable program code devices configured to cause a computer to effect processing the network management data in the second network manager.

27. A computer program product as recited in claim **26**, wherein the network management data includes event and trap data, and wherein the trap data includes one or both of database traps and non-database traps.

28. A computer program product as recited in claim **26**,

wherein said computer readable code further comprises:

fifth computer readable program code devices configured to cause a computer to effect filtering of the network management data at the sender process to produce filtered data, and

wherein said third computer readable program code devices are configured to cause a computer to effect forwarding of the filtered data from the sender process to the receiver process.

29. A computer program product as recited in claim **26**, wherein the first network manager maintains a first database and the second network manager maintains a second database, and

wherein said computer readable code further comprises:

fifth computer readable program code devices configured to cause a computer to effect synchronization of the second database to the first database.

30. A computer program product as recited in claim **29**, wherein said fifth computer readable code devices comprises:

computer readable program code devices configured to cause a computer to effect sending a synchronization request from the receiver process to the sender process;

computer readable program code devices configured to cause a computer to effect generating database traps for the entire contents of the first database;

computer readable program code devices configured to cause a computer to effect forwarding the database traps to the receiver process; and

computer readable program code devices configured to cause a computer to effect updating the second database in accordance with the database traps.

31. A computer program product as recited in claim **30**, wherein said fifth computer readable code devices further comprises:

computer readable program code devices configured to cause a computer to effect filtering the database traps prior to their being forwarded to the receiver process.

32. A computer-implemented method for synchronizing a first database of a first network management station to a second database of a second network management station, comprising the steps of:

(a) sending a synchronization request from the second network management station to the first network management station to request synchronization of the second database to the first database;

(b) generating, at the first network management station, database traps for the entire contents of the first database;

(c) forwarding the database traps from the first network management station to the second network management station; and

(d) updating the second database in accordance with the database traps.

33. A computer-implemented method as recited in claim **32**, wherein said method further comprises the step of: (e) filtering the database traps prior to said forwarding (c) and said updating (d).

34. A computer program product, comprising:

a computer usable medium having computer readable code embodied therein to implement synchronization

of a first database of a first network management station to a second database of a second network management station, and

wherein said computer readable code comprises:

first computer readable program code devices configured to cause a computer to effect sending a synchronization request from the second network management station to the first network management station to request synchronization of the second database to the first database;

second computer readable program code devices configured to cause a computer to effect generating, at the first network management station, database traps for the entire contents of the first database;

third computer readable program code devices configured to cause a computer to effect forwarding the database traps from the first network management station to the second network management station; and

fourth computer readable program code devices configured to cause a computer to effect updating the second database in accordance with the database traps.

35. A computer program product as recited in claim **34**, wherein said computer readable code devices further comprises:

fifth computer readable program code devices configured to cause a computer to effect filtering the database traps prior to the forwarding by said third computer readable program code devices.

36. A computer-implemented method as recited in claim **19**, wherein said connecting (a) comprises:

(a1) determining whether the receiver process of the second network is authorized to receive the network management data from the sender process of the first network; and

(a2) connecting the receiver process associated with the second network to the sender process associated with the first network if said determining (a1) determines that the receiver process is authorized to receive the network management data from the sender process.

37. A computer program product as recited in claim **26**, wherein said first computer readable program code devices include:

computer readable program code for determining whether the receiver process of the second network is authorized to receive the network management data from the sender process of the first network; and

computer readable program code for connecting the receiver process associated with the second network to the sender process associated with the first network if said determining (a1) determines that the receiver process is authorized to receive the network management data from the sender process.

* * * * *

US005832221A

United States Patent [19]

Jones

[11] Patent Number: 5,832,221

[45] Date of Patent: Nov. 3, 1998

[54] **UNIVERSAL MESSAGE STORAGE SYSTEM**

[75] Inventor: **Mark Alan Jones**, New Providence, N.J.

[73] Assignee: **AT&T Corp**, Middletown, N.J.

[21] Appl. No.: **581,653**

[22] Filed: **Dec. 29, 1995**

[51] **Int. Cl.**[6] **G06F 15/56**

[52] **U.S. Cl.** **375/200.36**; 379/88

[58] **Field of Search** 379/88; 395/200.03, 395/200.08, 616, 200.36, 200.55, 200.43; 707/200; 345/326

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,837,798	6/1989	Cohen et al.	379/88
5,255,305	10/1993	Sattar	379/88
5,333,266	7/1994	Boaz et al.	395/200.36
5,452,351	9/1995	Sattar	379/88
5,613,108	3/1997	Morikawa	395/616
5,632,011	5/1997	Landfield et al.	345/326
5,647,002	7/1997	Brunson	379/88

OTHER PUBLICATIONS

Syd Weinstein, The Elm Reference Guide (The Elm Mail System Version 2.4), pp. 1–29, Oct. 1992.

Syd Weinstein, The Elm Users Guide (The Elm Mail System Version 2.4), pp. 1–12 Oct. 1992.

Syd Weinstein, File name limit.c, The Elm Mail System, Revision 5.4, pp. 1–7 May 1994.

Hilal et al., Designing Large Electronic Mail Systems, IEEE, pp. 402–409 Dec. 1988.

Barbara et al., The Gold Mailer, IEEE, pp. 92–99 Dec. 1993.

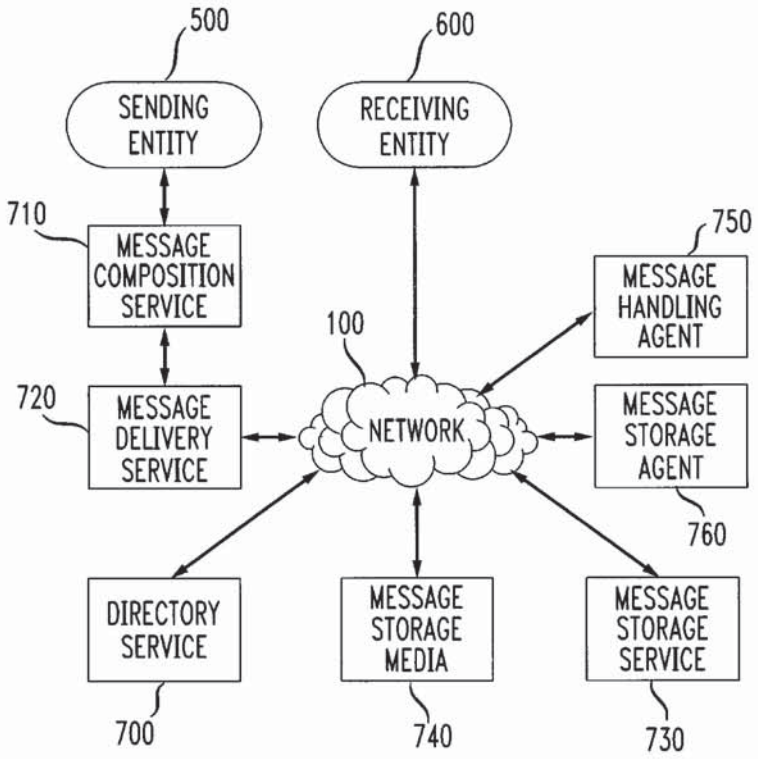
Primary Examiner—Richard L. Ellis

Assistant Examiner—Patrice L. Winder

[57] **ABSTRACT**

A message storage system, for use with a communication network and in which a network presence is provided for an entity, stores a message from a sender to a network presence. The message storage system accepts a query including a specified property, and generates a mailbox including the stored message when one of the message properties is the specified property. A software agent processes the stored message in accordance with a processing preference included in attributes associated with the entity. The message storage system generates a summary of the message and updates the summary of the message in response to modifications to the message.

11 Claims, 2 Drawing Sheets



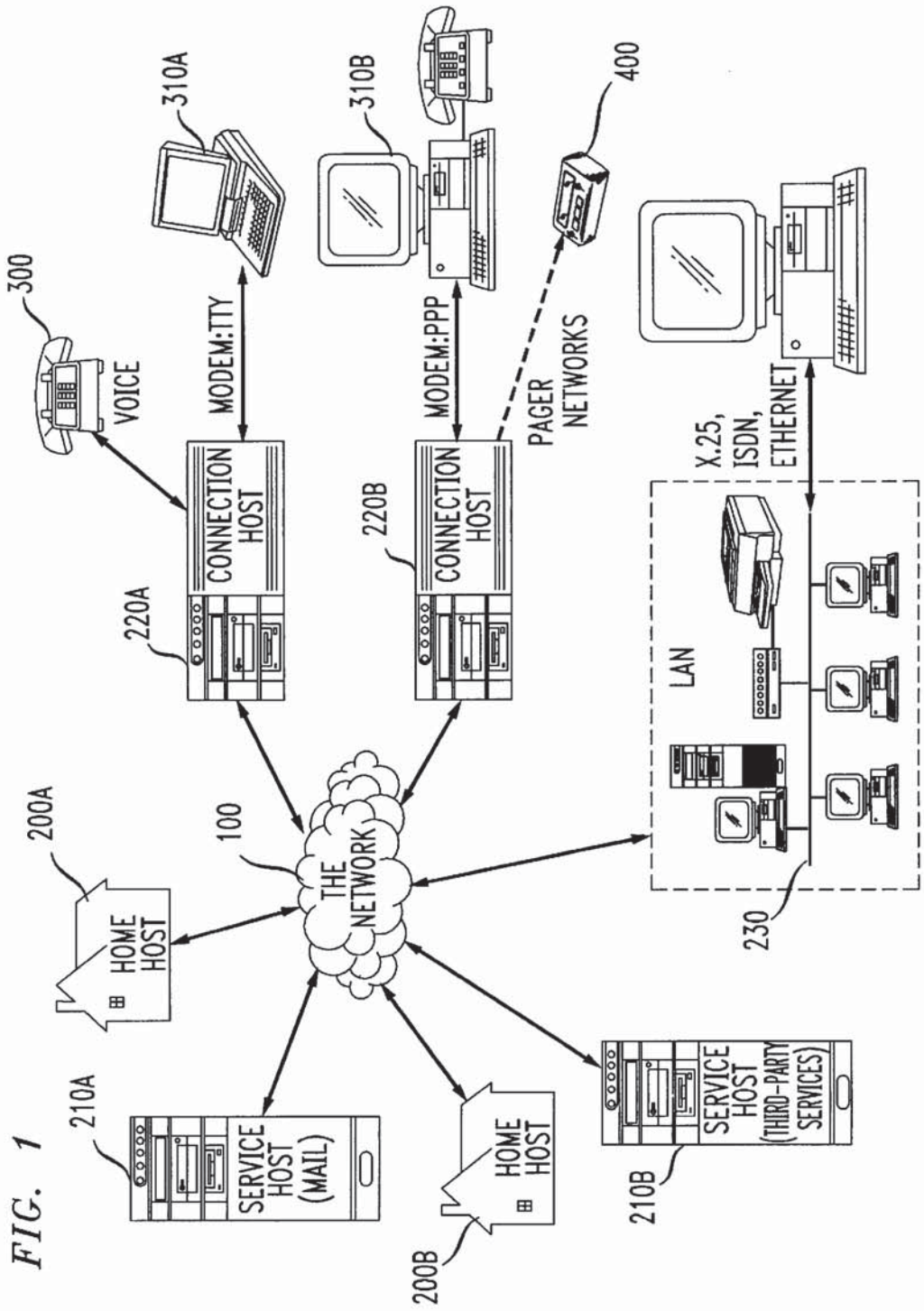
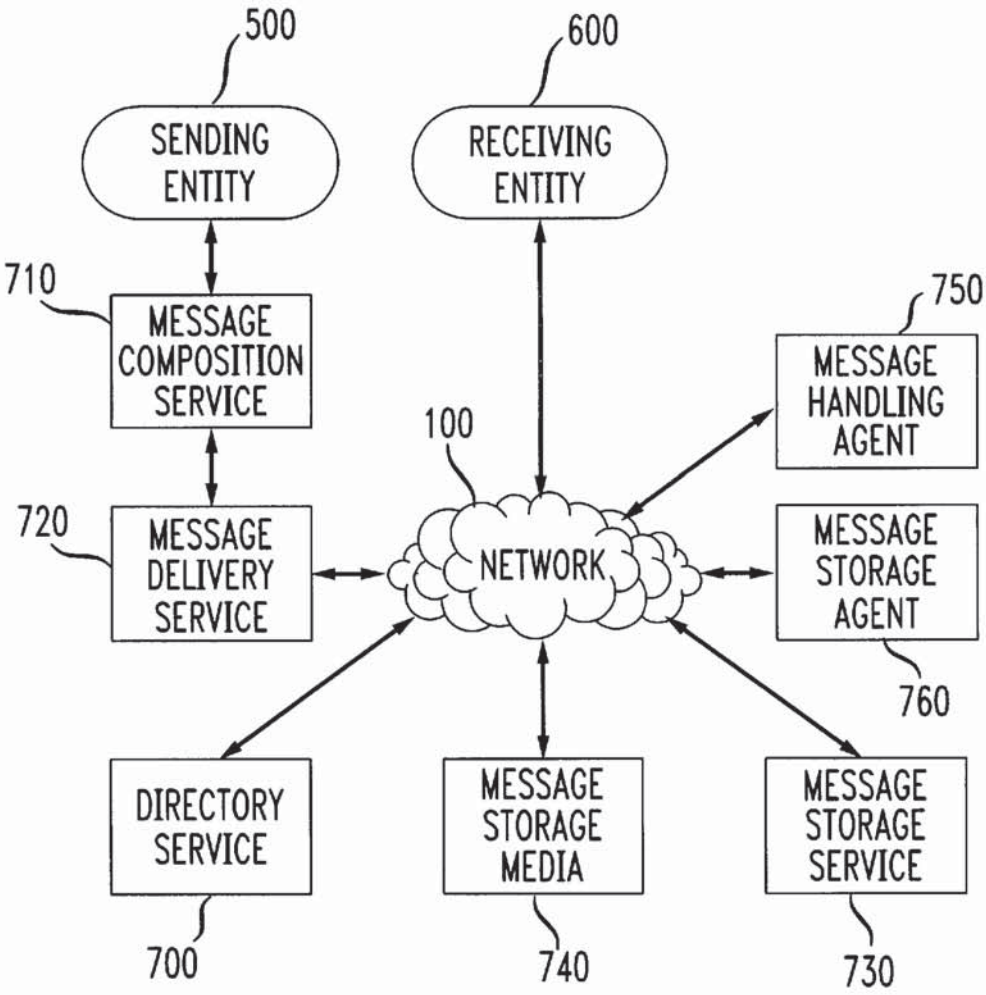

300
VOICE
310A
MODEM:TTY
310B
MODEM:PPP
400
PAGER NETWORKS
X.25, ISDN, ETHERNET
CONNECTION HOST
220A
CONNECTION HOST
220B
LAN
200A
HOME HOST
100
THE NETWORK
230
210A
SERVICE HOST (MAIL)
200B
HOME HOST
210B
SERVICE HOST (THIRD-PARTY SERVICES)

FIG. 1

FIG. 2


500
SENDING ENTITY
600
RECEIVING ENTITY
710
MESSAGE COMPOSITION SERVICE
750
MESSAGE HANDLING AGENT
100
720
MESSAGE DELIVERY SERVICE
NETWORK
MESSAGE STORAGE AGENT
760
DIRECTORY SERVICE
MESSAGE STORAGE MEDIA
MESSAGE STORAGE SERVICE
700
740
730

UNIVERSAL MESSAGE STORAGE SYSTEM

BACKGROUND OF THE INVENTION

The present invention relates to a computer-based communication network service, and, more particularly, is directed to a system in which entities are represented by network presences associated with handle identifiers used as addresses.

Communication by messaging is becoming steadily more popular. Advantages of messaging relative to a personal conversation include more efficient use of communication capacity, that is, text based electronic mail requires far less channel capacity than an equivalent voice message; more time efficient due to less need for time consuming ritual social inquiries; opportunity for more careful composition; and capability of including various types of communication, that is, the message can be in a multimedia format including audio, video and/or text. Furthermore, if the message is broadcast, its composition effort is amortized across the recipients. Also, the message can be buffered when a recipient is unavailable or unwilling to receive the message immediately; the recipient has more time to plan their response; an electronic message is easy to capture and place in long term storage; and software can be used to assist in composing and organizing messages.

One problem with presently available forms of messaging is that it is necessary to determine and remember addressing information which is substantially unrelated to the identity of the recipient. Voice and facsimile messages require a telephone number. Electronic mail messages require an address usually comprising an assigned user name and electronic domain name, and possibly information indicating a communication service provider. Also, the format of an electronic mail address can differ depending on the communication provider.

Telephone numbers are difficult to remember, usually change when a person moves or switches jobs, can be obtained through a directory having only a very limited number of search fields and may lack privacy as it is fairly easy to associate address information with a telephone number.

Personal telephone numbers, such as the proposed AT&T 500/700 personal number services, assign a telephone number to a subscriber, and associate the assigned telephone number with a destination telephone number and, optionally, a backup telephone number having a voice recording and storage device. Callers call the assigned telephone number, and calls are automatically routed to the destination telephone number. If the destination telephone number does not accept the call, then the call is automatically routed to the backup telephone number. The destination telephone number may be changed frequently by the subscriber. These personal number services mask changes in the subscriber's telephone number, that is, allow a subscriber to have a single telephone number even while travelling or moving frequently, and provide increased privacy. However, the personal numbers are still difficult to remember, can be obtained through a directory having only a very limited number of search fields, are accessible through only one medium, and, due to reliance on a telephone number, are tied into a particular addressing infrastructure which has limited call management options.

Electronic mail addresses are often difficult to remember, usually change when a person switches jobs or communication carriers, and are difficult to obtain due to lack of universal directory services.

Another problem with presently available forms of messaging is that if someone is reachable by a variety of message types, e.g., voice mail, facsimile and electronic mail on several networks, a sender is not sure which type of message will be most effective at reaching the intended recipient.

A further problem with presently available forms of messaging is that there may be a conversion problem between an available sending device, such as a twelve-key telephone, and a preferred receiving device, such as a facsimile machine. Also, there may be a conversion problem between the form of the originating message, e.g., voice mail, and the preferred form of received message, e.g., electronic mail. Products for converting the form of the message, such as the AT&T INTUITY product for a PBX/LAN environment, have been introduced, but have not yet achieved widespread usage. A proposed Multipurpose Internet Multimedia Extension (MIME) specification for Internet electronic mail allows senders to provide content in multiple, alternative formats but conversion issues have not been resolved.

SUMMARY OF THE INVENTION

A message storage system, for use with a communication network and in which a network presence is provided for an entity, stores a message from a sender to the network presence.

In an aspect of the invention, the message storage system accepts a query including a specified property, and generates a mailbox including the stored message when one of the message properties is the specified property.

In another aspect of the invention, the message storage system obtains proxy objects from a proxy storage.

In an aspect of the invention, a software agent processes the stored message in accordance with a processing preference included in attributes associated with the entity.

It is not intended that the invention be summarized here in its entirety. Rather, further features, aspects and advantages of the invention are set forth in or are apparent from the following description and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a communications network according to the present invention; and

FIG. **2** is a block diagram showing the logical relationship of various services according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

An entity is represented by at least one handle, described in detail below. Generally, a handle is a more abstract representation of the entity than is found in the prior art, and avoids the problems of prior art entity representations associated with their insufficiently abstract (i.e., too physical) nature. Each handle provides a distinct cyberpresence identifier for an entity.

Directory services, as described in the present disclosure, provide more flexibility than prior art directory services. When used with handles according to the present disclosure, directory services provide further enhanced flexibility. Generally, a network directory service provides information about entities and finds entities based on descriptive queries. Some of the directory information is publicly available, whereas other of the directory information is not publicly available but is usable by the directory service for derefer-

encing addresses. Entities specify the desired privacy level (s) of their directory information. The service provides one or more global and specialized network directories, which may be physically distributed across multiple hosts in the network.

Message composition and delivery services, as described in the present disclosure, provide more flexibility than prior art message composition and delivery services. When used with handles according to the present disclosure, message composition and delivery services provide further enhanced flexibility. Generally, message delivery services provide for specification of policies by entities as to the forwarding of messages to specific endpoints or to a universal message storage facility, notification of message receipt and retrieval of messages. Message notification and retrieval may be according to entity specified criteria, such as priority to particular senders or to particular subjects.

Message storage services, as described in the present disclosure, provide more flexibility than prior art message storage services. When used with handles according to the present disclosure, message storage services provide further enhanced flexibility.

An important feature of the present disclosure is the application to objects such as messages and cyberpresences of information retrieval techniques, such as vector space models, which have heretofore been applied only to documents. Generally, flexibility is accomplished by applying information retrieval techniques to objects, rather than by relying primarily on more structured database query techniques.

Network Envirement

Referring now to the drawings, and in particular to FIG. **1**, there is illustrated a network which is generally assumed as the environment in the present disclosure. The network shown in FIG. **1** comprises a communication network **100**, home host computers **200**, service host computers **210**, connection host computers **220**, gateways to other networks such as a local area network (LAN) **230**, software executed on the various computers, and customer premises equipment such as twelve-key telephone sets **300**, personal computers **310**, terminals, and pager networks **400**. Although not specifically shown in FIG. **1**, Internet connections and wireless transmission may be used in a network contemplated in the present disclosure.

As will be apparent to those of ordinary skill in the art, many different communication protocols may be employed in communicating between the various parts of the network, such as TCP/IP, X.25, ISDN, Ethernet, asynchronous line protocols and analog and/or digital voice transmission. Communication for transactional services are implemented in a secure, flexible remote procedure call (RPC). Also, as appropriate, authentication and encryption protocols are employed, for example, hypertext transfer protocol (HTTP) or secure socket layer (SSL) protocol.

Various divisions of communications capability between customer equipment and network equipment are encompassed by the network of FIG. **1**. The network is assumed to provide processing capability for customer equipment which lacks sufficient processing capability to provide the functions described below. The specific type of software programming used to provide these functions is not critical.

In one case, the customer equipment comprises only a twelve-button telephone set. A user dials a connection host which is part of the network, such as the nearest connection host or a toll-free number providing access to a connection host. Using one or more of voice input and touch-tone input, the user establishes network access authority, such as by entering an identification code and password. The connection host verifies access authority with the user's home host, then makes appropriate network resources available to the user by, for example, presenting menus of choices to the user.

In another case, the customer equipment comprises a private host such as a personal computer and a modem. The user instructs the private host to establish a connection to a connection host. In this case, the connection host functions in a more limited manner than in the previously described situation where the customer premises equipment is a telephone set.

In yet another case, the customer equipment comprises a receive only pager network. A connection host somewhere in the network executes software behalf of the pager network.

Handles

An entity may be a person, organization, corporation, department within a corporation, use (interest) group, or a set of entities. Alternatively, the entity may be a functional role, such as president of an organization.

An electronic presence is established for every entity which requires a public identity. The electronic presence is also referred to herein as a network presence or "cyberpresence". The electronic presence is identified by a handle. The network presence for an entity serves as a locus of publicly available information about the entity, as a point of connection to the entity, and as a centralized set of resources available to the entity. Physically, a network presence comprises an account on a home host computer, such as the home host computer shown in FIG. **1**, the actual network resource usage associated with the account, the capability of using additional network resources and identification of the account in network directories. Typically, an account resides on a home host, but some accounts may reside on multiple hosts due to their resource usage.

An entity may have multiple network presences each of which is associated with a distinct handle. For example, an entity which is a person may have one network presence for activities related to their job, another network presence for activities related to their primary hobby, and yet another network presence for activities related to their other personal uses.

As used herein and in the claims, "handle" refers to a unique identifier registered with a universal directory network service for use by the entity. A handle represents an abstract entity, and does not correspond to a physical endpoint although it may be associated with one or more physical endpoints for various purposes, as described below. The handle functions as the network name of the entity, and also functions as the network address of the entity, but is not a physical end point address. An entity may have one or more handles each of which is associated with a network presence. Primarily for billing purposes, each handle is associated with a sponsor that is not necessarily the entity using the handle.

Handles permit decoupling of physical endpoints and delivery systems from the network presence for an entity. That is, a handle is not merely an address, it is a representation of an entity because it is associated with resource usage and availability for the entity.

Since a handle is unique at any point in time, it can be used as a universal address. Another important feature of a

handle is its persistence, that is, its association with one using entity despite changes in the attributes associated with the entity, such as telephone number, address, employment affiliation or sponsor. If the entity is a group, then members or other attributes of the group may change over time, but the group (entity) still retains the handle. Similarly, if the entity is a person performing certain functions, e.g., the president of XYZ Company, then the person associated with the handle may change, but the handle persists; in this case, the handle is a referential expression describing a functional role.

Examples of handles are: "bigbear", "Jane_Farnsworth", "ATT", "usenet.rec.gardening", "empiricists", "president_XYZ" and so on. A handle is not a telephone number. A telephone number is a physical point which is associated with a varying number of users, whereas a handle is not a physical point, and is associated with only the entity represented by the handle. A handle may include alphabetic information which serves a mnemonic purpose.

Advantages of personally chosen handles, relative to handles assigned by a network authority, include memorability, that is, personally chosen handles have mnemonic value for message senders, individuality, ability to be descriptive or representative of a network persona or attributes of an entity, and ability to mask the identity of an entity.

Handles may eventually be reassigned, when the possibility of confusion between entities is deemed to be sufficiently low. For example, when an entity expires, such as a person dies or a corporation is dissolved, and a predetermined time has passed since expiration, the handle of the expired entity may become available for use by another entity.

Examples of entity attributes which may be associated with an individual's handle include password(s), name, address, preferred format for message reception, primary telephone number, forwarding telephone number, fax number, family members, employer, profession, hobbies and so on.

Examples of entity attributes which may be associated with an organization's handle include password(s), name, address, preferred format for message reception, telephone number, fax number, number of members, industry, products or services, annual sales, affiliated companies and so on.

As the name of a network presence for an entity, a handle is a logical place for an entity to obtain and/or offer network services. Generally, the network resources available to an entity include a personalized access point, information storage capacity, information access structures such as an "address book", a personalized set of message spaces, and convenient ways to access frequently used on-line services.

An "address book", as used herein and in the claims, is a personalized directory of frequently accessed message destinations for the entity, that is, a set of handles which identify entities. In other words, the objects in an address book are handles. An address book allows the entity to refer to other entities in a more convenient manner, such as by name, photograph, or nickname; thus, the address book hides the actual handles from the entity.

Since the attributes associated with a handle may change over time, it is preferred to locally store only the handles for an address book with respective temporal information such as date/time stamps. At each use of a handle, the address book automatically queries the directory service as to whether any attributes have changed since the timestamp of the handle. The address book locally stores any local information associated with the handle, such as the entity's nickname or relationship definition for the handle. The initial contents of the address book may be determined with reference to the sponsor of the entity.

A query can define a "special" address book of an entity's base (universal) address book, that is, the query restricts the set of handles in the special address book.

The entity can view a subset of the address book by specifying attributes of the objects in the desired subset. For example, a view of an address book may provide, for each entity, its name, face (or other) picture and telephone number.

An entity obtains services through its handle generally by subscribing to the service; such services are referred to herein and in the claims as "vendor services". Service providers, which may be third party vendors, the provider of the communication network or the provider of the network presence system, then add the service capabilities to the handle in an appropriate manner, such as by authorization to act on instructions from the handle, by adding choices to menu-driven interfaces accessible to the handle, or by adding functional capabilities to software agents associated with the handle. Examples of software agents are a message handling agent and a message storage agent, described below. Examples of vendor services are a message composition service, a calendar scheduling service and a software agent service. FIG. 1 shows a service host for third-party services in which the vendor offers a service directly from its own handle and other handles must explicitly communicate with the vendor's handle for the service.

An entity offers services through its handle by responding to requests directed to the handle. For example, the entity may add functional capabilities to one of the agents associated with its handle to provide a service to other handles. In some embodiments, for provision of certain services, the entity may make special billing arrangements with the network.

When the entity is accessible to at least one messaging service, the attributes of an entity include a physical endpoint to which messages are to be delivered. For example, when the entity is a pager network, the physical endpoint is the pager equipment. When the entity is an individual, the physical endpoint can be non-network equipment, such as a fax machine, or network storage.

When the entity is accessible to at least one messaging service, the attributes of an entity include a preferred media format for receiving messages. For example, when the entity is accessible by more than one message media format, such as fax, voice mail, textual electronic mail and multimedia electronic mail, the entity indicates the media format in which it prefers to receive messages in its "preferred reception media" attribute.

There are several differences between personal telephone numbers, such as the proposed AT&T 500/700 personal number services, and the handles of the present invention. The personal number services provide a customer with only a telephone number, that is, a completely numeric identifier which lacks mnemonic value, whereas the present handles may comprise alphanumeric information having mnemonic value. The personal number services must be associated with at least one destination telephone number for a customer, whereas the present handles need not be associated with a specific telephone number, instead, an entity may opt to have the network store its messages, and then the entity retrieves its messages from the network, for example, by a dial-in telephone call.

Directory Services

The universal network directory service stores attributes associated with handles and responds to queries relating to the stored information to provide a very flexible searching ability. The directory service may be a vendor service.

When a directory user such as a message sender desires to know a handle for an entity, the sender provides sufficient descriptive information to uniquely identify the entity. In some cases, the sender interacts repeatedly with the directory service to uniquely identify the entity. For example, in response to the sender's provision of a person's name, city and state of residence, employer and profession, the directory service returns the requested handle.

In other situations, a directory user knows a handle and provides a query to the directory service to obtain one or more attributes associated with the handle. For example, a directory user may wish to know a daytime telephone number associated with a handle.

Handle attributes have privacy level information specified by the entity represented by the handle. In its simplest form, privacy level information simply indicates whether the attribute is publicly available or not publicly available, i.e, private. Therefore, entities may maintain essentially "unlisted" handles with no attribute information publicly available.

The directory service generally maintains indices of the attributes in a variety of hierarchical structures, and responds to structure sensitive queries.

Each of a directory query and a response thereto generated by the universal network directory service may contain multimedia depending on the kinds of interfaces and applications used. As used herein and in the claims, information in a multimedia format means information in at least two of an internal computer format such as binary format, text format such as ASCII, voice format and video format.

Entities and/or their respective sponsors have the ability to self-administer certain of the entity's attributes in accordance with preferences, such as password(s), preferred format for message reception, forwarding telephone number and privacy status of their attributes, using an automated administration procedure including a software program executed on at least one of the hosts of FIG. **1**.

Handle attributes have authenticity information associated therewith. In its simplest form, authenticity information simply indicates who provided the attribute information. More complicated authenticity information indicates, for example, when the attribute information was provided. The authenticity information provides a basis for forming a trustworthiness opinion of the associated attribute information.

When the user of the directory service is a handle, additional flexibility is contemplated. Specifically, the information returned from the directory service may be automatically transferred to another service, such as a message composition service offered by a third-party vendor. For example, when a handle queries the directory service for all handles having specified attributes, such as:

(type of entity=individual),

(family members=at least one child), and (address=NY or NJ)

the resulting set of handles may be used as a set of addresses for a message broadcast by a message preparation service used by the handle.

In certain embodiments, an additional privacy designation of "secret" is available for information associated with a handle. This is useful for broadcasts prepared by a message preparation service to entities matching specified criteria, where the matching entities wish to remain unknown, for example, persons testing positive for a particular disease. In these cases, the entities may be interested in receiving information related to their attributes, but want their possession of such attributes to be masked from mass marketers and/or probes attempting to guess the information. If secret information is used to resolve a handle, then information identifying the receiving entity is withheld in any delivery receipts provided by the network to the sender or querying party.

Another example of additional flexibility when the user of the directory service is a handle is an updating service for an address book. The updating service may simply add the results of each directory query to the address book. Alternatively, the results of the directory query may automatically be transferred to the updating service, and then the updating service asks the entity associated with the handle using the directory whether and/or how to retain the results. As yet another alternative, a software agent associated with the handle may treat the results of the directory query as an information object to be processed in accordance with general policies specified by the entity for information objects, i.e., policies for information which is not limited to directory information.

Message Composition and Delivery Services

A message composition service permits a message to be composed and associated with a destination query. That is, a message is sent to a destination query, rather than a specified endpoint. The destination query is of the form described earlier for the directory service.

A message delivery service provides delivery of the message to the objects satisfying the destination query associated with the message, with the objects typically being handles.

Messages are assumed to include content information and envelope information, such as sender, destination query determining the recipient(s), network transit history, arrival time, subject and priority. Senders are identified by their handles. Recipients are identified by the destination query, unless their identity is masked (see discussion below). Content information may comprise multimedia and interactive programs; notes from family, friends and business associates; electronic correspondence from businesses, government, associations and so on; electronic postcards; electronic letters; electronic newsletters and magazines; electronic advertising; electronic solicitations and so on.

When the sender knows the preferred media format for the recipient of the message, the sender can instruct the message delivery service to put the message, composed in one format, into the preferred format when technically feasible. For example, the message may be composed as text, and converted to voice using speech synthesis. As will be appreciated, the preferred media format for a message recipient can usually be determined from a query to the network directory service. Certain message preparation services are capable of automatically querying the directory service and using the query results for format conversion.

The message sender can require that it remain anonymous, for example, by composing a message with the sender explicitly identified as "anonymous" or by omitting sender information.

A message recipient can require that it remain anonymous. For example, if an entity has set all of its attribute

information to non-public, it may receive broadcast messages to entities having its attributes, but the message delivery service will not provide an identifying delivery receipt to the message sender. However, the message sender may be informed that a delivery occurred, and possibly the number of messages that were delivered.

Message non-repudiability is provided when the sender requests that the message delivery service guarantee that the sender of the message is correctly identified. Non-repudiability is particularly useful for messages having financial consequences.

Message Handling Agents

When the recipient of a message is a handle, additional flexibility is contemplated. Specifically, the handle may subscribe to the services of a message handling agent (a type of software agent) which performs functions on behalf of the entity represented by the handle in accordance with attributes associated with the handle.

As used herein and in the claims, "software agent" refers to a software program usually executed by one of the host computers shown in FIG. **1**. The software agent is a type of vendor service to which an entity may subscribe through its handle. The software agent has various capabilities, depending on its specific implementation, and is characterized by independent operation or agency operation. The software agent is event-driven. The software agent responds to events and carries out behavior in accordance with the event and environment, such as time of day. A software agent is capable of creating, transferring and deleting objects, invoking other vendor services, notifying, monitoring and keeping statistics.

Independent operation indicates that the software agent performs its functions generally independently of how and when its subscribing entity interacts with its network presence.

Agency operation indicates that the software agent operates on behalf of its subscribing entity, typically inheriting access authority and so on of the subscribing entity, in accordance with entity specified preferences usually recorded as attributes for the entity.

Examples of services provided by a message handling agent include notification of a new message, automatic forwarding of messages to endpoints (e.g., a copy to other handles or a message store), summarizing messages, sorting messages according to entity criteria (ex: priority, size, sender and/or subject), deleting messages according to entity criteria, storing messages according to entity criteria, converting the media format of a message, and preparing simple replies to certain formatted messages. That is, message handling agents exhibit context dependent behavior based on the sending and receiving equipment, the message's characteristics and the recipient's preferences.

In one case, a handle may have a "preferred message media format=text" associated therewith. The entity may then communicate a request such as "speak the contents of the most recent message to me" to its message handling agent. In this case, the message handling agent converts the media format of the message from text to voice, and forwards the voice message to a destination indicated by the entity, such as a telephone.

The message handling agent facilitates message enabled behavior. For example, the message handling agent may check the content of a message for a certain type of information, such as schedule related information, and automatically transfer such information to another service associated with the handle, such as a calendar program.

Message Storage Services

Prior art message storage services typically have a physical association between a mailbox, that is, a physical data file, and a message. A message storage service according to the present disclosure is not so limited. A mailbox is considered to be a set of messages which satisfy a query. By varying the attributes specified in the query, an entity can achieve various levels of mailbox granularity, from considering all the messages for which the entity has read permission (which may include messages received by other entities) to considering only a subset of the messages received by one entity such as itself. Additionally, a mailbox may have different message dispositions, such as who is notified of the mailbox query results.

A mailbox is defined by a query over a set of messages. An address book is defined by a query over a set of cyberpresences.

Typical prior art systems treat notifying a recipient of the arrival of a message as a procedural, event-driven process. For example, "do (x) when (y)" where "y" is the event of a message arrival.

The present disclosure contemplates a persistent query, that is, a query for which an entity maintains a continuing interest. The persistent query is a declarative representation depending on at least one property of an object, and is not event-driven. The query originator can assert the query at regular intervals (polling). For example, "if a message has status NEW or UNREAD then it is of interest".

The persistent query defining a set of objects is always consistent with the data against which it is asserted. The persistent query communicates data changes to objects interested in such changes. The persistent query can be implemented, for example, by having the target of the query notify the originator of the query when the response of the target changes.

A persistent query is useful when an entity has a need to know something. A software agent is useful for responding to events in a predetermined manner.

Notifying a recipient of the arrival of a message is a declarative process, that is, an entity is considered as submitting a persistent query, and when the present result of the persistent query invalidates or logically mismatches the previous result of the persistent query, the entity which submitted the query is notified. For example, if the entity has submitted a persistent query for "all stored unread messages addressed to my handle", and a new message has been stored since the last query was asserted, then the result of the previous query (no unread messages) is invalidated, so the a notification message (one unread message) is generated.

Message processing abilities are dependent upon the handle of the entity. For example, as a default, the handle for an entity has full read, write and modify ability for messages addressed to the handle. For particular types of messages addressed to an entity's handle, the entity may specify read, write and/or modify ability for other handles.

A mailbox according to the present disclosure can be considered a one time object when it is the result of a one time query, or can be considered a persistent object when it is the result of a persistent query. A mailbox which is the result of a persistent query is effectively continously updated. It will be appreciated that an entity can create multiple persistent mailboxes by communicating multiple persistent queries to the message storage service. Such an abstraction is stored as a convenience to an entity.

For example, when the entity is a paging system, the set of persistent queries might be "new messages for each of the users associated with the paging system entity".

The message storage service generates a message ID for each message and provides indexing services for message retrieval so that queries can be satisfied faster. For example, the message storage service may compute message properties such as usage statistics, creation time, message type, message size, current storage medium and so on. Practically and where possible, the message storage service simply extracts certain information from the message envelope as message properties (e.g., sender). If the message object is modified, such as by annotation, or deleted, the message storage service detects this or is notified by the modifier and updates the storage related properties.

The message storage service determines the storage policy for a message according to a general policy (not message specific) specified by a message recipient, including the current storage medium (one of the message properties), and the message persistence, that is, when the message should be moved to archival storage. Finally, the message storage service actually stores the message.

In some embodiments, the message storage service responds to requests for message IDs for messages whose storage is not directly controlled by the service. Such messages, also referred to as "proxy objects", have message IDs and computed properties, and can be queried and retrieved through an interface with the direct controller of the storage of the proxy object. The software which directly controls storage of a proxy object is responsible for notifying the message storage service of message creation, modification and deletion events.

An example of usage of a proxy object is a message shared by several entities. The properties of the proxy object may differ by entity, such as whether the message has been read, or annotations appended thereto. The proxy object may be automatically assigned different priorities for different entities.

Message Storage Agents

When the user of the message store is a handle, additional flexibility is contemplated. Specifically, the handle may subscribe to the services of a message storage agent (a type of software agent) which performs functions on behalf of the entity represented by the handle in accordance with attributes associated with the handle.

Examples of services provided by a message storage agent include notifying an entity of a new message, deleting messages according to message specific entity criteria, archiving messages according to message specific entity criteria, and converting the media format of a message. The summary may include category, thread (relationship to other messages such as topic), content type, content and so on. Activities particularly suited to a message storage agent include archiving messages, aging messages, compressing message and placing messages in different virtual folders.

A message storage agent can assert a persistent query against a message store on behalf of an entity. This function is particularly useful when the entity is a paging system which otherwise expects to be in "receive only" type operation.

For example, a message storage agent might monitor a directory and provide notification of changes in the employer for a particular entity.

As another example, if a vendor service is providing a physical location, such as from a global positioning service, then the message storage agent could notify an entity of the location of another entity, such as a child of the first entity.

FIG. **2** shows the logical relationship of the above-described services. The network directory service **700**, message composition service **710**, message delivery service **720**, message storage service **730**, message handling agent **750** and message storage agent **760** each comprise software programs for execution by at least one of the host computers **200**, **210**, **220** shown in FIG. **1**. The message storage media **740** shown in FIG. **2** comprises storage media, such as RAM or disk, associated with at least one of the host computers shown in FIG. **1**.

A sending entity **500** communicates with the message composition service **710** to compose a message. Message composition may include interaction with the directory service **700**. The sending entity **500** then instructs the message composition service **710** to transfer the composed message to the message delivery service **720**, which delivers the message to its specified destination and provides various forms of reports on delivered messages to the sending entity **500**.

Messages may be delivered in real time to a receiving entity **600**, or may be delivered to the message storage service **730** logically associated with the message storage media **740**. The message handling agent **750** generally operates on messages received from the message delivery service **720**. The message storage agent **760** generally operates on messages placed on the message storage media **740** by the message storage service **730**. The message handling agent **750** and message storage agent **760** operate on behalf of the receiving entity **600**.

Although an illustrative embodiment of the present invention, and various modifications thereof, have been described in detail herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to this precise embodiment and the described modifications, and that various changes and further modifications may be effected therein by one skilled in the art without departing from the scope or spirit of the invention as defined in the appended claims.

What is claimed is:

1. A message storage system for use with a communication network and means for providing a network presence for an entity having attributes, said message storage system comprising:

storage means for storing a message from a sender to said network presence, said message having properties,

means for accepting a query over a set of messages for which the entity has read permission, the query including a specified property, and

wherein said storage means is also for generating a mailbox including said message when one of said properties of said message satisfies the query.

2. The system of claim **1**, wherein said query is a one-time query.

3. The system of claim **1**, wherein said query is a persistent query.

4. A message storage system for use with a communication network and means for providing a network presence for an entity having attributes, said attributes including a processing preference for messages stored for said network presence, said message storage system comprising:

storage means for storing a message from a sender to said network presence, and

a software agent for processing the stored message in accordance with said processing preference, and for asserting a query against said storage means,

and wherein said storage means is also for generating a mailbox including the message, when the message satisfies the query.

5. The system of claim 4, wherein said processing preference is a preferred media format, and said software agent is operative to convert a media format of said stored message to the preferred media format.

6. The system of claim 4, wherein said processing preference specifies one of a notifying action, a deleting action and an archiving action, and said software agent is operative to perform the action specified by said processing preference on said stored message.

7. A message storage system for use with a communication network and means for providing a network presence for an entity having attributes, said message storage system comprising:

storage means for storing a message from a sender to said network presence, and

means for automatically generating a summary of said message and for automatically updating the summary in response to modification of the message,

and wherein said storage means is also for generating a mailbox including the message, when the message satisfies a query asserted against said storage means.

8. The system of claim 7, further comprising means for automatically appending the generated summary to said message.

9. A method for storing a message received from a communication network, comprising the steps of:

storing, in a message storage system, said message being from a sender to a network presence established for an entity, said entity having attributes, said message having properties, and

accepting a query over a set of messages for which the entity has read permission, the query including a specified property,

wherein the message storage system generates a mailbox including said message when one of said properties of said message satisfies the query.

10. A method for storing a message received from a communication network, comprising the steps of:

storing, in a message storage system, the message from a sender to a network presence established for an entity, said entity having attributes including a processing preference for messages stored for said network presence,

processing the stored message in accordance with said processing preference, and

asserting a query against said message storage system,

wherein said message storage system generates a mailbox including the message, when the message satisfies the query.

11. A method for storing a message received from a communication network, comprising the steps of:

storing, in a message storage system, the message from a sender to a network presence established for an entity, said entity having attributes,

automatically generating a summary of said message, and

automatically updating the summary in response to modification of the message,

wherein said message storage system generates a mailbox including the message, when the message satisfies a query asserted against said message storage system.

* * * * *

US005892916A

United States Patent [19]

Gehlhaar et al.

[11] Patent Number: 5,892,916

[45] Date of Patent: Apr. 6, 1999

[54] **NETWORK MANAGEMENT SYSTEM AND METHOD USING A PARTIAL RESPONSE TABLE**

[76] Inventors: **Jeff B. Gehlhaar**, 11934 Dapple Way, San Diego, Calif. 92128; **James W. Dolter**, 11755 Timberlake Dr., San Diego, Calif. 92131-2329; **Siddharth R. Ram**, 7920 Avienda Navidad, #147, San Diego, Calif. 92122; **Rahul Anand**, 927 Wilbur Ave., #3, San Diego, Calif. 92109

[21] Appl. No.: **997,160**

[22] Filed: **Dec. 23, 1997**

[51] **Int. Cl.**[6] **G06F 9/40**

[52] **U.S. Cl.** **395/200.53**

[58] **Field of Search** ..... 364/DIG. 1 MS File, 364/DIG. 25 MS File; 395/200.3, 200.32, 200.53

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,561,769	10/1996	Kumar et al.	395/200.32
5,651,006	7/1997	Fujino et al.	395/200.53
5,832,226	11/1998	Suzuki et al	395/200.53

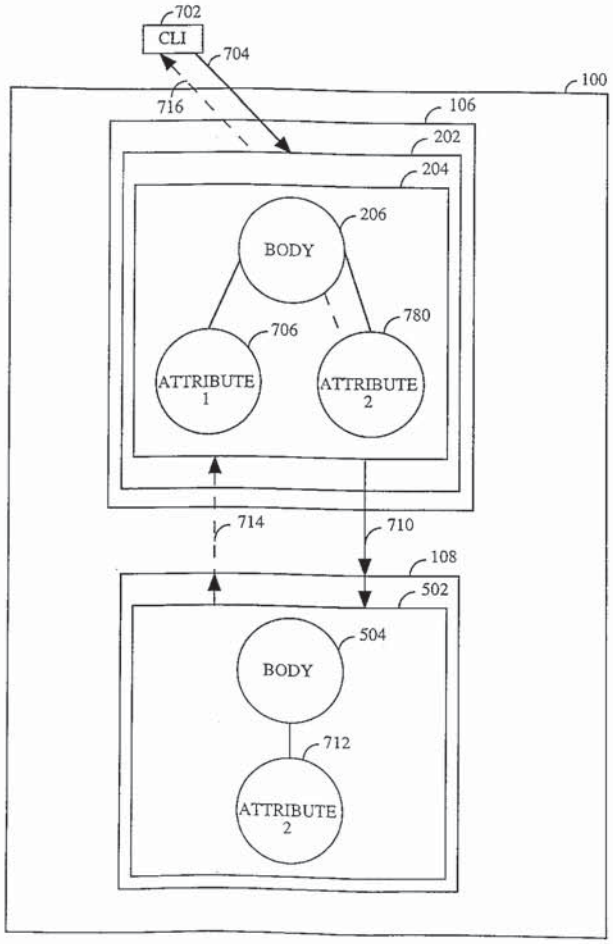
Primary Examiner—Robert B. Harrell

Attorney, Agent, or Firm—Russell B. Miller; Bruce W. Greenhaus; Christopher O. Edwards

[57] **ABSTRACT**

A system and method for network management, including a message handling process, having a network manager and at least one network element is described. The network management messaging system sends and receives multiple concurrent messages between the network element layer and the network management layer. Support of concurrent messaging greatly speeds and simplifies network management. A client request is received at a first managed object, at least part of the client request is fulfilled by a second managed object. A major row is created in response to receiving the client request, and a minor row associated with a managed object request sent to the second managed object. The minor row has an index that associates the minor row with the major row, and correlates a response to the managed object request with the client request.

4 Claims, 8 Drawing Sheets



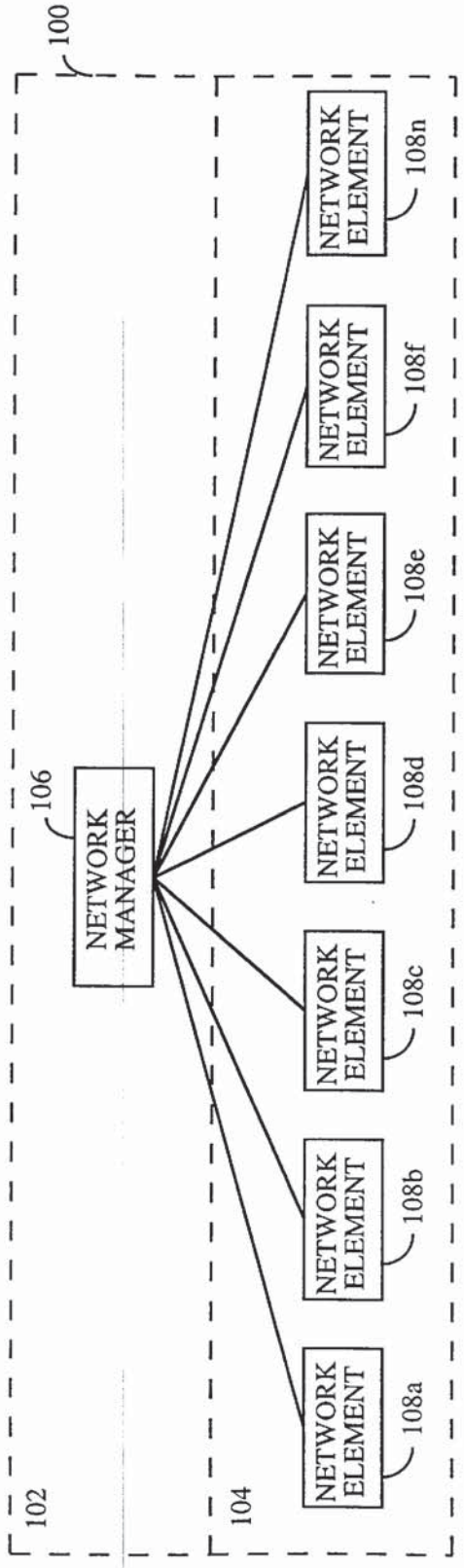


FIG. 1

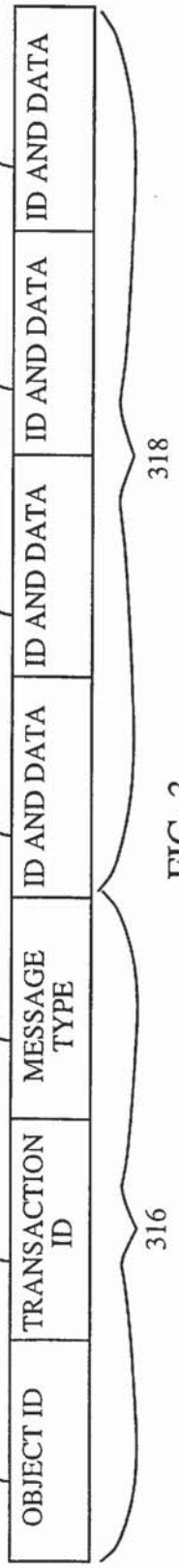


FIG. 3

FIG. 4

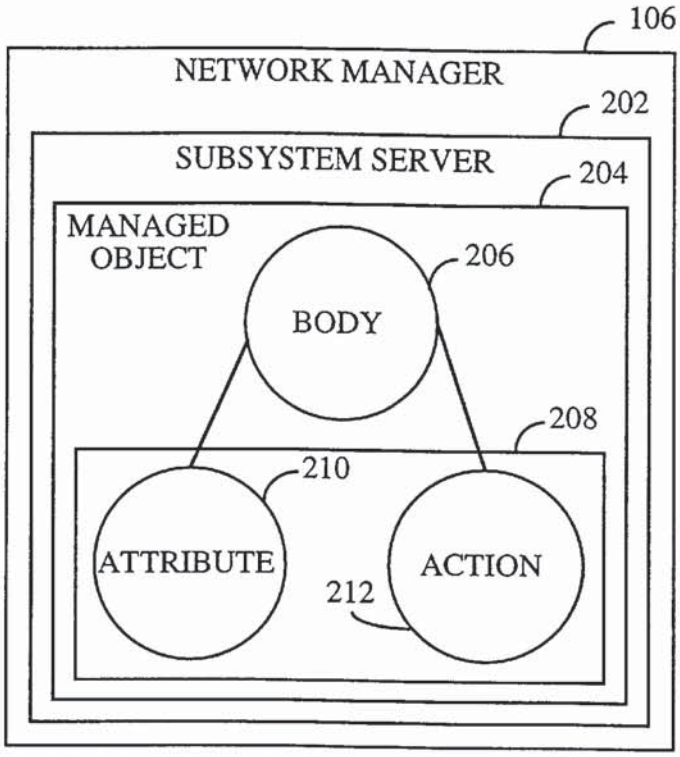

106
NETWORK MANAGER
202
SUBSYSTEM SERVER
204
MANAGED OBJECT
206
BODY
208
210
ATTRIBUTE
ACTION
212

FIG. 2

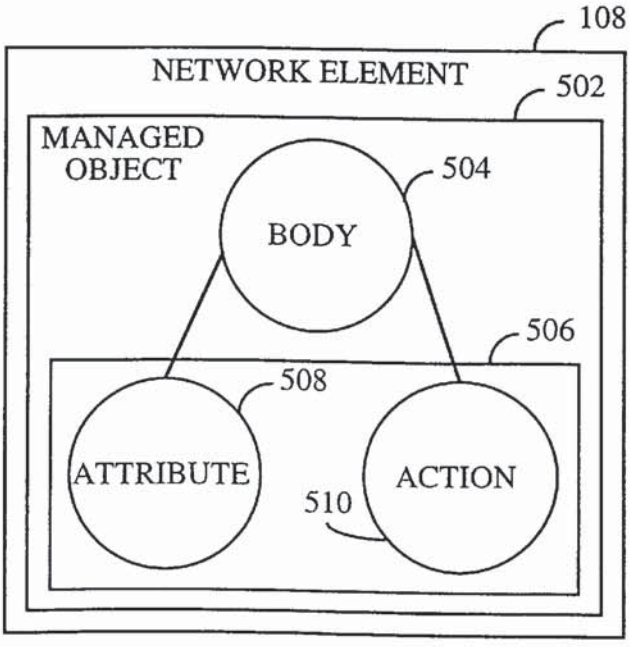

108
NETWORK ELEMENT
502
MANAGED OBJECT
504
BODY
506
508
ATTRIBUTE
ACTION
510

FIG. 5

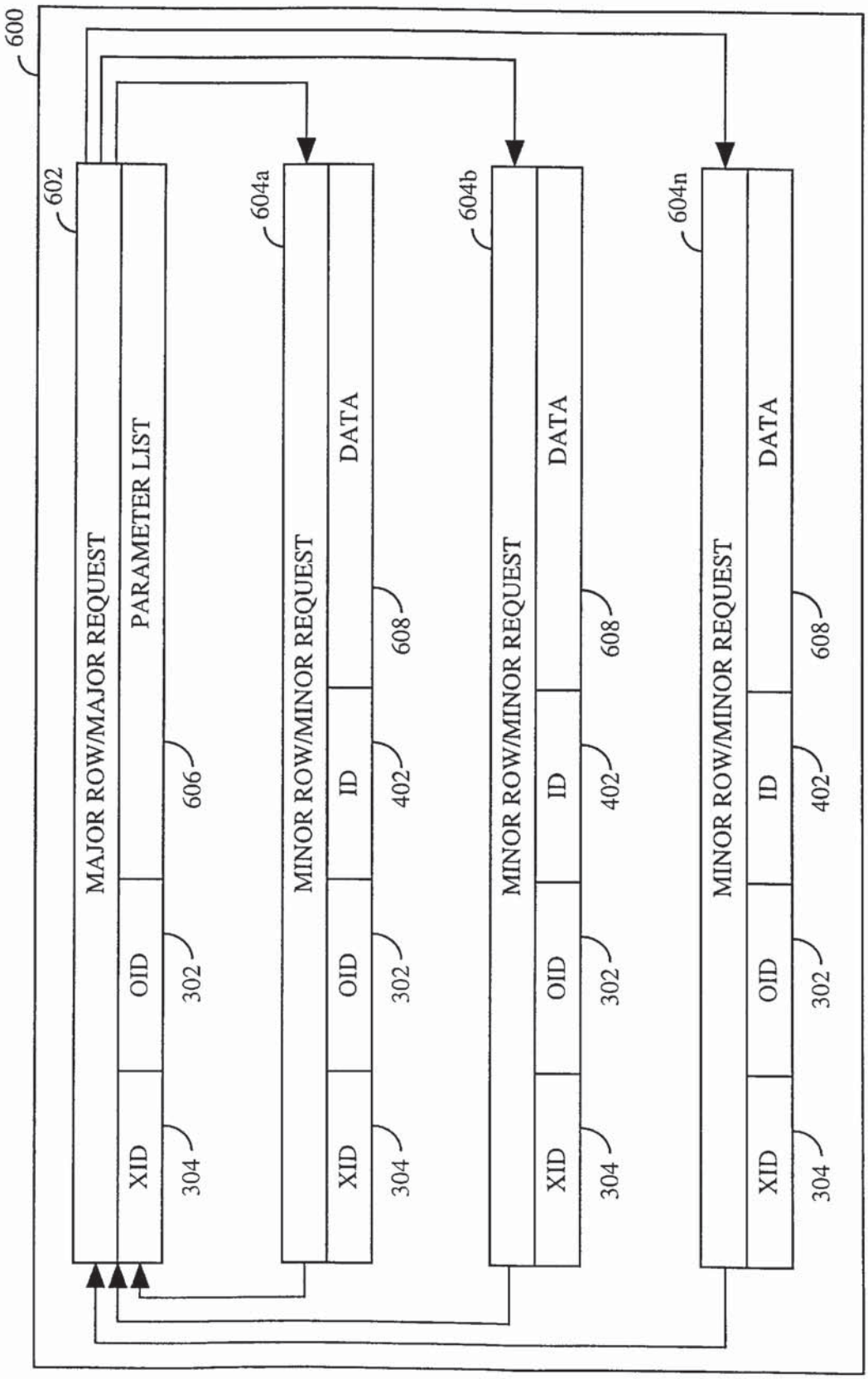

600
602
MAJOR ROW/MAJOR REQUEST
XID
OID
PARAMETER LIST
304
302
606
604a
MINOR ROW/MINOR REQUEST
XID
OID
ID
DATA
304
302
402
608
604b
MINOR ROW/MINOR REQUEST
XID
OID
ID
DATA
304
302
402
608
604n
MINOR ROW/MINOR REQUEST
XID
OID
ID
DATA
304
302
402
608

FIG. 6

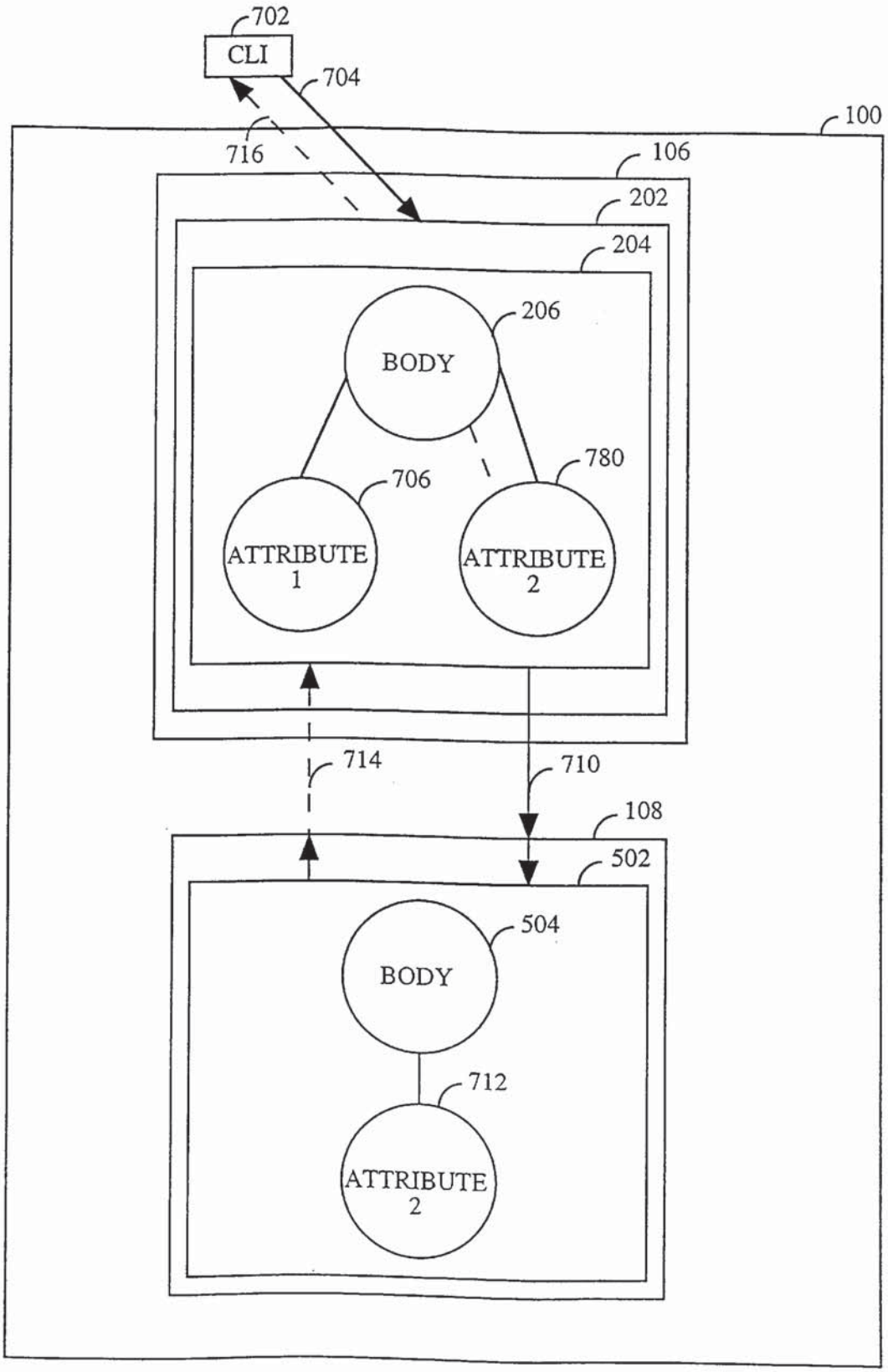

702
CLI
704
716
100
106
202
204
206
BODY
780
706
ATTRIBUTE 1
ATTRIBUTE 2
714
710
108
502
504
BODY
712
ATTRIBUTE 2

FIG. 7

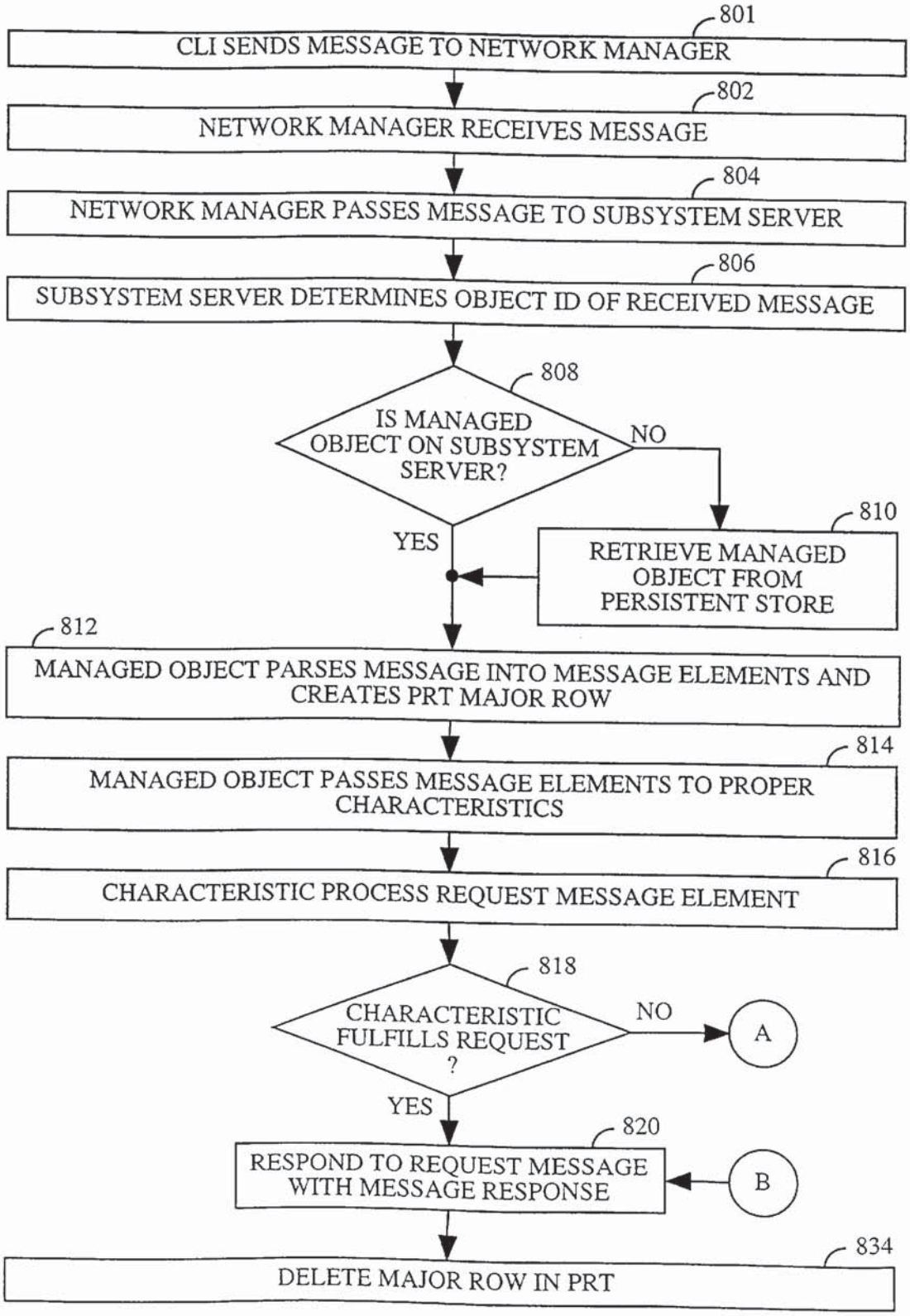

801
CLI SENDS MESSAGE TO NETWORK MANAGER
802
NETWORK MANAGER RECEIVES MESSAGE
804
NETWORK MANAGER PASSES MESSAGE TO SUBSYSTEM SERVER
806
SUBSYSTEM SERVER DETERMINES OBJECT ID OF RECEIVED MESSAGE
808
IS MANAGED OBJECT ON SUBSYSTEM SERVER?
NO
YES
810
RETRIEVE MANAGED OBJECT FROM PERSISTENT STORE
812
MANAGED OBJECT PARSES MESSAGE INTO MESSAGE ELEMENTS AND CREATES PRT MAJOR ROW
814
MANAGED OBJECT PASSES MESSAGE ELEMENTS TO PROPER CHARACTERISTICS
816
CHARACTERISTIC PROCESS REQUEST MESSAGE ELEMENT
818
CHARACTERISTIC FULFILLS REQUEST ?
NO
A
YES
820
RESPOND TO REQUEST MESSAGE WITH MESSAGE RESPONSE
B
834
DELETE MAJOR ROW IN PRT

FIG. 8A

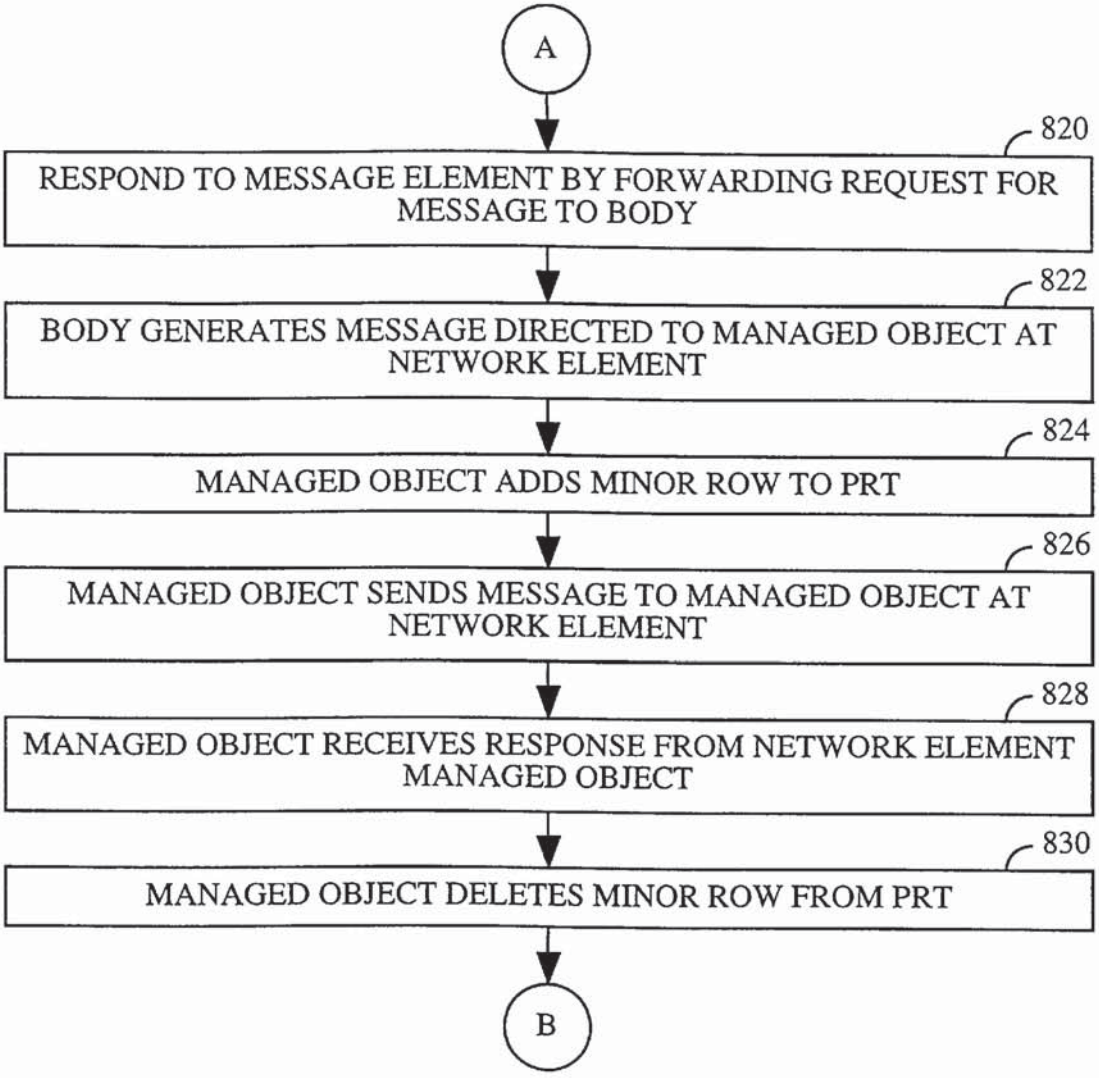

A
820
RESPOND TO MESSAGE ELEMENT BY FORWARDING REQUEST FOR MESSAGE TO BODY
822
BODY GENERATES MESSAGE DIRECTED TO MANAGED OBJECT AT NETWORK ELEMENT
824
MANAGED OBJECT ADDS MINOR ROW TO PRT
826
MANAGED OBJECT SENDS MESSAGE TO MANAGED OBJECT AT NETWORK ELEMENT
828
MANAGED OBJECT RECEIVES RESPONSE FROM NETWORK ELEMENT MANAGED OBJECT
830
MANAGED OBJECT DELETES MINOR ROW FROM PRT
B

FIG. 8B

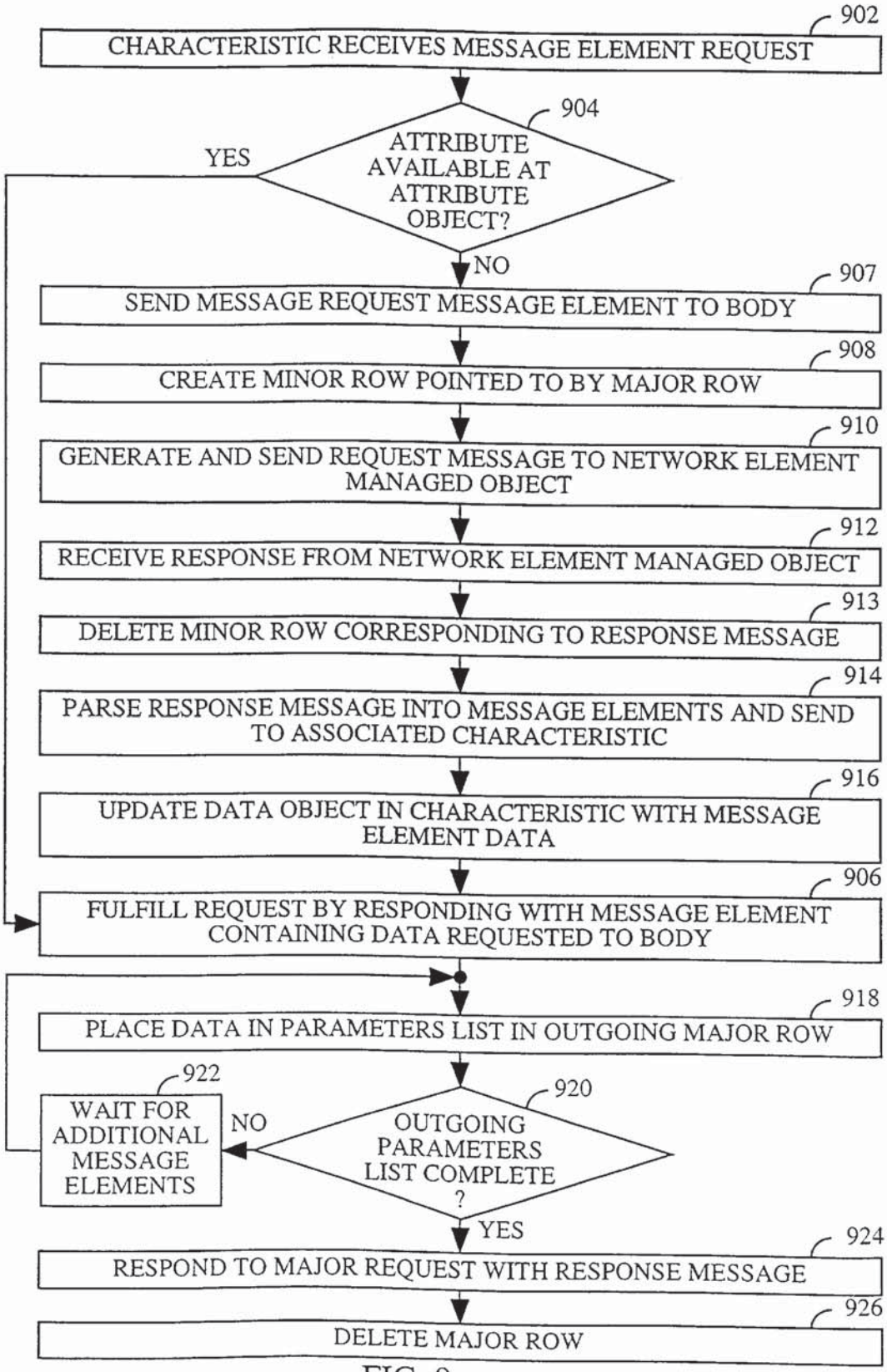

902
CHARACTERISTIC RECEIVES MESSAGE ELEMENT REQUEST
904
ATTRIBUTE AVAILABLE AT ATTRIBUTE OBJECT?
YES
NO
907
SEND MESSAGE REQUEST MESSAGE ELEMENT TO BODY
908
CREATE MINOR ROW POINTED TO BY MAJOR ROW
910
GENERATE AND SEND REQUEST MESSAGE TO NETWORK ELEMENT MANAGED OBJECT
912
RECEIVE RESPONSE FROM NETWORK ELEMENT MANAGED OBJECT
913
DELETE MINOR ROW CORRESPONDING TO RESPONSE MESSAGE
914
PARSE RESPONSE MESSAGE INTO MESSAGE ELEMENTS AND SEND TO ASSOCIATED CHARACTERISTIC
916
UPDATE DATA OBJECT IN CHARACTERISTIC WITH MESSAGE ELEMENT DATA
906
FULFILL REQUEST BY RESPONDING WITH MESSAGE ELEMENT CONTAINING DATA REQUESTED TO BODY
918
PLACE DATA IN PARAMETERS LIST IN OUTGOING MAJOR ROW
922
WAIT FOR ADDITIONAL MESSAGE ELEMENTS
NO
920
OUTGOING PARAMETERS LIST COMPLETE ?
YES
924
RESPOND TO MAJOR REQUEST WITH RESPONSE MESSAGE
926
DELETE MAJOR ROW

FIG. 9

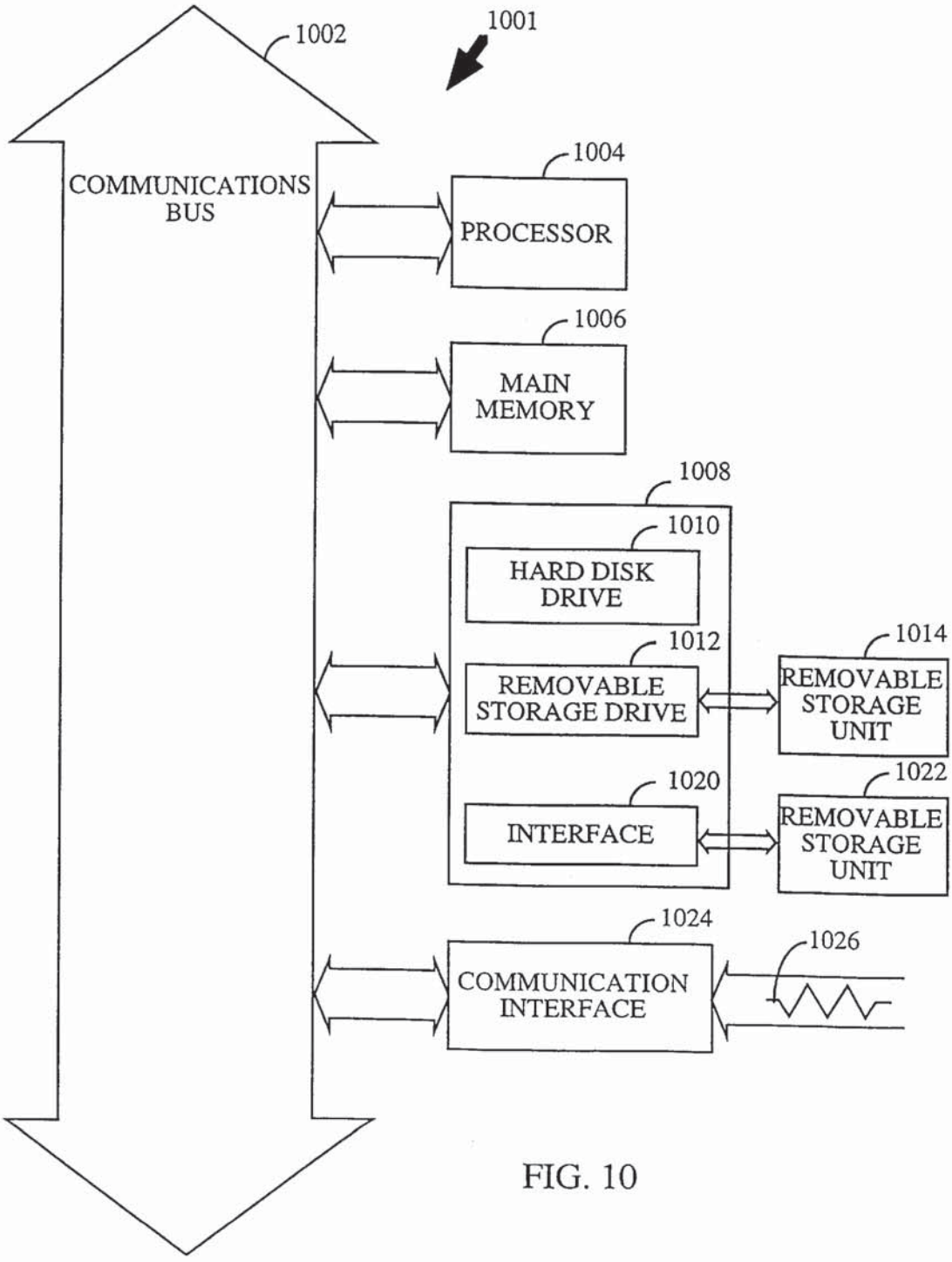

1002
1001
COMMUNICATIONS BUS
1004
PROCESSOR
1006
MAIN MEMORY
1008
1010
HARD DISK DRIVE
1012
REMOVABLE STORAGE DRIVE
1014
REMOVABLE STORAGE UNIT
1020
INTERFACE
1022
REMOVABLE STORAGE UNIT
1024
COMMUNICATION INTERFACE
1026

FIG. 10

NETWORK MANAGEMENT SYSTEM AND METHOD USING A PARTIAL RESPONSE TABLE

BACKGROUND OF THE INVENTION

I. Field of the Invention

The present invention relates generally to network management systems, and more specifically is directed toward management of network resources using distributed intelligence and state management.

II. Related Art

Telecommunication service providers provide a wide range of services to their customers. These services range from the transport of a standard 64 kbit/s voice channel (i.e., DS0channel) or subrate thereof to the transport of higher rate digital data services (e.g., video). Both voice channels and digital data services are transported over the network via a hierarchy of digital signal transport levels. For example, in a conventional digital signal hierarchy 24 DS0 channels are mapped into a DS1 channel. In turn, 28 DS1 channels are mapped into a DS3 channel. The number of customers served and the complexity of services offered by telecommunication service providers is always increasing.

The wide range of services, signals and channels require a complex network of telecommunications equipment. Management of the complex telecommunications network is necessary in order to maintain optimum levels of service to the customer as well as efficiency in the maintenance and usage of the equipment itself. As networks grow increasingly complex, both in the size of the network and the range of services provided by the network, network management becomes increasingly important. Telecommunications service providers provide for management of the network by implementing network management systems designed to manage, provide for growth and ensure optimum performance of the network.

Network management systems include at least two layers. The first layer is the network management layer. The network management layer includes a network manager that monitors and controls the configuration of the network. The network manager is usually a server and software that maintains a logical representation of the state and condition of the network. The network manager provides an interface to the network for users and applications wishing to manage the network.

The second layer of the network is the network element layer. The network element layer includes all of network elements. Examples of network elements are the mobile switching center (MSC), call detail adjunct (CDA), home location registry (HLR), channel service unit (CSU), customized dial plan (CDP), CDMA interconnect subsystem (CIS), etc. The network elements provide the functionality and services of the entire network, independent of the network manager.

The network management system implements a set of procedures, software, equipment and operations designed to keep the network operating near maximum efficiency. The goals of network management include configuration management, fault location and repair management, security management, and performance management.

Configuration management deals with installing, initializing, loading, modifying and tracking configuration parameters, network elements and their associated software. The network manager accomplishes configuration management by downloading configuration parameters and software to the network elements. The network manager also tracks the configuration of the network by retrieving data indicating the configuration of the network elements and their associated software.

Fault location and repair management predicts and diagnoses problems with the network and provides a methodology for replacing or rerouting the network around the affected network elements. The network manager accomplishes fault location and repair management by retrieving fault information from the network elements in the network element layer. For example, the network manager may retrieve the number of severely errored seconds (SES) or the frame error rate (FER) from a network element in order to locate faults and diagnose problems with the network. If, upon retrieval of fault information from the network element layer, the network manager determines that particular network elements are experiencing degraded performance or are inoperative, the network manager may reroute the network around the affected elements. The network manager accomplishes the rerouting function by downloading additional configuration information to the network elements in order to reconfigure the network.

Security management allows the network manager to restrict access to various resources in the network, thereby giving customers different levels of access to different network resources. The network manager accomplishes security management by retrieving the current security information from the network elements and analyzing the retrieved information. If the access to resources in the network is to be changed, the network manager accomplishes the change by downloading additional or changed security information to the network elements, thereby changing the levels of access the users have to the network resources.

Performance management provides statistical information about the network's operation allowing the network manager to manage the resources of the network to ensure optimum performance. The network manager monitors the usage and traffic levels of the network element to ensure that the traffic on the network is properly distributed. Proper distribution of traffic among the network elements helps to ensure that the network does not experience performance degradation because a few network elements are carrying most of the communications load, while other network elements are carrying too little. The network manager accomplishes performance management by retrieving information pertaining to the traffic loading of the particular network elements. If the network manager determines that the performance of the network would be improved by redistributing the network traffic from one set of network elements to another, the network manager downloads additional configuration information to the network elements, thereby reconfiguring them.

Network management, therefore, is accomplished by communication between the network manager and the network elements. A telecommunications network may contain tens of thousands of network elements. A network manager, therefore, may be interacting with thousands of elements at a time. It is not practical for the network manager to spend time establishing synchronous connections with each of the network elements that it wishes to communicate with, if it must establish thousands of such connections to manage the network.

A more practical solution to the problem of communication between the network manager and the network elements is asynchronous communication. In asynchronous

communication, messages containing requests, commands and data are transmitted between the network manager and network elements over the network. In such a system, messages transmitted from a network element before others may arrive at the network manager after the later transmitted message. Context information about the network management system, therefore, may not be assumed or inferred since a received message may not reflect the current state of the system.

Asynchronous communications pose additional management problems. Often, when the network manager sends a message to a network element, it must wait for a response. The management functions of the network manager often require information to be retrieved from multiple subsystems. If the network includes thousands of network elements, this serial process of messaging and response becomes too slow for practical network management. Current asynchronous packet network management systems are unable to handle concurrent outstanding messages sent to the subsystems.

Furthermore, messages between the network elements and network manager perform differing functions. Some messages communicate information, or data, between the network manager and the network elements, or between the network elements themselves. Other messages cause actions to be performed. Current messaging systems for network management implement different systems to accommodate the different types of network management messages.

Additionally, a network may contain multiple hierarchical layers. In order for a network manager to communicate with the multiple layers, current messaging systems require the implementation of multiple messaging protocols. Multiple messaging protocols allow the network manager to communicate with network elements existing at multiple levels within the network management system hierarchy. Multiple network management protocols make the communication between the network manager and the network elements, as well as between the network elements themselves, complex and error prone.

What is needed, therefore, is a network management messaging system which is capable of sending and receiving multiple concurrent messages to the network element layer from the network management layer. Such a system would greatly increase the speed with which the network can be managed. The network management messaging system should provide a universal message handling method that handles both information and action messages. Such a system should also provide for interacting with network elements at differing levels within the network management hierarchy, and between the network elements themselves.

SUMMARY OF THE INVENTION

The present invention comprises a comprehensive network management messaging system that can efficiently accomplish network management though asynchronous messaging between the network entities. The network management messaging system is capable of sending and receiving multiple concurrent messages to the network element layer from the network management layer. Support of concurrent messages greatly speeds and simplifies network management. The network management messaging system of the present invention provides a universal message handling method that handles both information and action messages. The network management messaging system of the present invention also provides for hierarchical interaction with network elements at differing levels within the network management hierarchy, and between the network elements themselves.

In the present invention, a plurality of network entities are defined for a plurality of managed network resources which include physical (e.g., network element hardware) and logical (e.g., circuit termination points) resources. A network entity is any network manager, network element, user or system that originates network management messages in the network management system.

The present invention comprises a system and method for network management, including a message handling process, having a network manager and at least one network element. The system and method includes receiving a client request at a first managed object, at least part of the client request being fulfilled by a second managed object. The system and method creates a major row in response to receiving the client request, and a minor row associated with a managed object request sent to the second managed object. The minor row has an index that associates the minor row with the major row, and correlates a response to the managed object request with the client request.

BRIEF DESCRIPTION OF THE DRAWINGS

The features, objects, and advantages of the present invention will become more apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference numbers indicate identical or functionally similar elements. Additionally, the left-most most digit of a reference number identifies the drawing in which the reference number first appears.

FIG. **1** is a block diagram of one embodiment of a network management system in accordance with the present invention;

FIG. **2** is a block diagram illustrating one embodiment of the network manager in accordance with the present invention;

FIG. **3** is a diagram illustrating the format of a message according to the present invention;

FIG. **4** is a diagram illustrating the format of a message element according to the present invention;

FIG. **5** is a block diagram illustrating one embodiment of a network element in accordance with the present invention;

FIG. **6** is a diagram illustrating an entry in a partial response table;

FIG. **7** is a diagram illustrating one embodiment of a network management system in accordance with the present invention;

FIG. **8** is a flowchart illustrating the process of receiving a CLI request message at the network manager and generating a CLI response message;

FIG. **9** is a flowchart illustrating the process of adding and deleting major and minor rows to a partial response table; and

FIG. **10** is a block diagram illustrating one embodiment of a computer for implementing the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the description that follows, the first portion describes the messaging system. The second portion describes the partial response table. Both of these features comprise elements of this invention.

Messaging System

FIG. **1** illustrates a network management system **100** which is an example of the environment for one embodiment

of the the present invention. Network management system **100** preferably complies with the International Telecommunications Union (ITU) telecommunications management network (TMN) standard. The TMN standard defines a layered framework for a service provider to implement its own network management processes.

Network management system **100** includes two layers **102** and **104**. Layer **102** is the network management layer **102**. Network management layer **102** comprises network manager **106**. Network manager **106** is shown as a single entity. In implementation, network manager **106** can comprise equipment or software present at one or more sites. For example, multiple service centers (not shown) can exist in different parts of the country (e.g., east coast and west coast). Network manager **106** can also be split among services and/or network elements. For example, in one embodiment, a first portion of the network manager is dedicated to satellite-based communications, and a second portion of the network manager is dedicated to cell-based communications. Generally, the network manager **106** is accessed by client applications, such as users and systems, to engage network management functions. Client applications access network manager **106** by transmitting messages to the manager **106** and receiving messages from the manager.

Layer **104** is designated as the network element layer **104**. Network element layer **104** is a physical layer that includes various network elements (e.g., mobile switching center, call detail adjunct, home location registry, channel service unit, customized dial plan, CDMA inner connect subsystem, etc.) used in the transport and routing of network traffic. Each network element **108***a*–**108***n* in network element layer **104** receives and transmits configuration, fault location, security and performance information associated with management of the network. In particular, network elements **108***a*–**108***n* are connected to network manager **106** in network management layer **102**. The present invention is applicable to, and contemplates, handling of any management information passed between client applications and the network manager **106**, or network manager **106** and network elements **108***a*–**108***n*.

Although the present invention is described as having only two layers, alternative embodiments of network management system **100** have a plurality of layers. For example, in accordance with one embodiment of the present invention, the network management system has a plurality of network management layers (more than two) hierarchically arranged. In another alternative embodiment, network management systems are arranged hierarchically. In in one embodiment having multiple network management system layers, network manager **106** interacts with network elements **108***a*–**108***n* through a plurality of communications protocols. In one such embodiment, each layer in the network management system interacts with the network manager **106** using a different protocol. Alternatively, network manager **106** interacts with network elements **108***a*–**108***n* through an intervening network management layer (i.e., network manager **106** would interact with a first network entity, which in turn would interact with a second network entity in a different layer). In general, a network entity is any network manager, network element, user or system that originates network management messages in the network management system **100**. However, the first and second network entity referred to here are preferably software applications that are responsible for controlling the interaction between the layers of the system.

In the preferred embodiment, network management system **100** is a message based system. Client applications, such as users, applications or systems, interact with network manager **106** by transmitting messages to the manager **106**. The messages usually contain requests or commands and are usually packetized. Likewise, network manager **106** communicates with network elements **108***a*–**108***n* by transmitting and receiving similar messages. Such a messaging system is said to be asynchronous and transaction based. Transaction based systems rely upon messages and responses thereto in order to manage the network. This is contrasted with connection based systems in which dedicated connections are established between external entities and network manager **106** or between network manager **106** and network elements **108***a*–**108***n*. In asynchronous message based systems, such as network management system **100**, messages are transmitted without acknowledgment from the network system or network entity to which the message is transmitted.

FIG. **2** further illustrates network manager **106**. The preferred embodiment of network manager **106** is a server and software that maintains a logical representation of the state and condition of network management system **100**. Network manager **106** monitors and controls the configuration of the network by interacting with network elements **108***a*–**108***n* in network element layer **104**. Client applications and other systems interact with and manage network elements **108***a*–**108***n* in network element layer **104** through network manager **106**.

Network manager **106** includes subsystem server **202**. Subsystem server **202** is a software process or application that executes on network manager **106**. Subsystem server **202** provides the interface for network manager **106** to all of the network entities, that are not part of the network manager **106**. Network manager **106** receives messages from client applications attempting to interact with or control the network and passes them to subsystem server **202**. In operation, subsystem server **202** is an application, running on a computer, which acts as a clearinghouse for all of the messaging that goes on within the network management system **100**. Subsystem server **202** examines a message received from network manager **106** and determines to which managed object **204** the message is to be routed. One managed object **204** is shown in FIG. **2** for the sake of simplicity and ease of understanding. However, in the preferred embodiment of the present invention, many such managed objects **204** exist.

Each managed object **204** is a logical representation of a particular network element **108***a*–**108***n* in the network management layer **102**. Subsystem server **202** provides an environment for the execution of managed objects **204**. Accordingly, each network element **108***a*–**108***n* in network element layer **104** preferably has an associated representative managed object **204** on subsystem server **202**. The managed objects **204** on network manager **106**, therefore, are preferably a logical representation of all the network elements **108***a*–**108***n* in network management layer **102**. Managed objects **204** provide a logical interface at network manager **106** to the network element layer **104**. The interface provides a means for the network manager **106** to communicate with, retrieve data from, and cause actions to be performed in the network element layer **104**. In addition to managed objects **204** representing network elements **108***a*–**108***n*, additional managed objects **204** may reside on subsystem server **202** to represent management functions available to users, applications and other systems interacting with network manager **106**.

Managed object **204** comprises body **206** and characteristics **208**. Characteristic **208** includes attributes **210**, actions

212 or both. Body 206 controls the behavior of the managed object 204. Messages sent or received by managed object 204 within the network management system 100 are processed and generated by body 206. Body 206 receives, parses and distributes to characteristics 208 any messages received from network management system 100. Likewise, any message to be sent by managed object 204 are generated by body 206. Characteristics 208 represent the data (attributes 210) available for retrieval and storage by the network manager 106 and the functionality (actions 212) that is available at managed object 204 and corresponding network element 108*a*–108*n*. Network manager 106 sets and retrieves attributes 210 by sending and receiving messages from network element 108*a*. For example, managed object 204 will have data members corresponding to its associated network element 108*a*, including the data transmission error rate, number of errored seconds and number of severely errored seconds of the network element 108*a*. Actions 212 represent functionality available to network manager 106 at managed object 204 or corresponding network elements 108*a*–108*n*. Network manager 106 sends command messages to network element 108*a*–108*n*, causing network element 108*a*–108*n* to perform an action, and network element 108*a*–108*n* responds with a response message confirming the execution of the action. Together, attributes 210 and actions 212 represent the characteristics 208 of managed object 204.

Managed objects 204 are not constantly running on the subsystem server. Managed objects 204 only execute, or exist, on subsystem server 202 when a message for a particular managed object 204 has been received at subsystem server 202, and while the managed object 204 is processing the message. When managed objects 204 are not active, they are stored on persistent store off the subsystem server 202. Examples of persistent store are magnetic or optical media, read only memory (ROM), or other permanent type storage. Storage of the managed objects 204 on persistent store during periods of non-use promotes efficient use of network manager resources. If the message received by subsystem server 202 is destined for a managed object 204 which is not in existence on the subsystem server 202, subsystem server 202 retrieves the desired managed object 204 from persistent store and passes the message to the desired object 204. If the original message received from the external system by network manager 106 is directed to more than one managed object 204, subsystem server 202 parses the original message into pieces that are interpreted and acted upon by individual managed objects 204.

FIG. 3 illustrates one example of the format of a message 300 to a managed object 204, from another network entity, such as a user, application or system external to network management system 100. In accordance with one embodiment, the message 300 is routed through network manager 106. The format of message 300 is illustrated for the purposes of example. One of the advantages of the present invention is the protocol insensitivity of the messaging system and partial response table architecture. Accordingly, alternative embodiments of the present invention use message formats which differ from those described herein.

Message 300 includes header 316 and payload 318. Header 316 includes object identification (OID) 302, transaction identification (XID) 304, and message type 306. OID 302 identifies the particular managed object 204 for which message 300 is intended. For example, a message 300 intended for, or related to, network element 108*a* would have an OID 302 identifying the managed object 204 corresponding to a given network element 108*a*. If managed object 204 were to generate a message for transmission to network element 108*a*, the one-to-one correspondence between managed objects 204 and the network elements 108*a*–108*n* would allow the address of network element 108*a* to be algorithmically computed directly from OID 302.

XID 304 identifies the particular transaction with which message 300 is associated. For example, a message from a user external to network management system 100 to network manager 106 would have a unique XID 304. The response message from network manager 106 to the user would use the same unique XID 304, or an XID 304 which was algorithmically determinable from the original XID 304. The XID 304 provides for the identification and management of transaction, and transaction response, messages transmitted between entities in network management system 100. Unique XIDs 304 guarantee that concurrent messages do not result in corruption of the information passed between the network entities during network management. Message type 306 indicates the nature of the operation requested or specified by message 300. For example, message type 306 may indicate that message 300 is a "get-attribute" message. A get-attribute message is a request to retrieve data members stored by attributes 210 and respond with the value of attributes 210. Alternatively, message type 306 may specify that message 300 is a "set-attribute" message. A setattribute message is a request to set (or reset) the data values of attribute 210. Other examples of message types of the preferred embodiment include a response to a get-attribute message, a response to a set-attributemessage, an "action" message, a response to an action message, a "create-object" message and a "destroy-object" message for creating and destroying managed objects, etc. An action message requests that an action be performed at managed object 204. Since managed object 204 is a logical representation of network element 108*a*–108*n*, an action request message usually specifies actions to be performed at a network element 108*a*–108*n* corresponding to the managed object 204. In such cases, managed object 204 will send an additional action message to its corresponding network element 108*a*–108*n*. For example, in accordance with one embodiment of the present invention, a "lock-action" message is provided to network management system 100 to bar a particular resource from providing a service. In response, the network element specified in the lock-action message is essentially "locked out". An "unlock-action" message results in a locked network element or resource becoming "unlocked." Responses to action messages are usually action status messages that indicate that the action request was completed or was not completed. Managed objects 204 are created and destroyed on network manager 106 in order to maintain one-to-one correspondence between network elements 108*a*–108*n* and managed objects 204. For example, if network element 108*a* is added to network element layer 104, a corresponding managed object 204 must be created with a create-object message on network manager 106. Managed objects 204 on network manger 106 are also created and destroyed to implement and remove additional functionality at the network management layer 102. The create-object message is preferably sent from a client external to network manager 106. Alternatively, the initialization process of the network element 108*a* includes sending a create-object message to network manager 106. If network element 108*a* is removed from network element layer 104, the corresponding managed object 204 in network management 106 is removed with a destroy-object message.

Message 300 further includes payload 318 comprising identifier (ID) and data 308*a*–308*n*. Payload 318 identifies

the particular characteristics **208** by which message **300** is to be implemented. ID and data **308***a*–**308***n* are used by body **206** to generate message element **400** as described below. For example, when managed object **204** receives a set-attribute message, data value(s) in attributes **210** will change.

ID and data **308***a*–**308***n* identify the particular characteristics **208** to which the message **300** is directed. For example, suppose message **300** is received by network manager **106**. OID **302** identifies the particular managed object **204** to which message **300** is to be routed. Message **300** is routed by subsystem server **202** to managed object **204**. Managed object **204** receives message **300**. Body **206** of managed object **204** examines ID and data information **308***a*–**308***n* of payload **318**. Body **206** parses message **300** into smaller "message elements" to be passed to characteristics **208**. Message type **306** of header **316** and ID and data **308***a*–**308***n* of payload **318** are passed to characteristic **208** corresponding to ID and data **308***a*–**308***n* of payload **318**.

FIG. **4** illustrates the format of one example of a message element transferred between body **206** of managed object **204** and characteristics **208** of that object **204**. As noted above, body **206** receives message **300** from the subsystem server **202** and parses it into at least one message element **400**. Message element **400** includes an internal identifier (ID) **402**, message type **306** and data **404**. Internal ID **402** is determined from ID information in ID and data fields **308***a*–**308***n* of payload **318**. Internal ID **402** identifies the particular data member in characteristics **208** which is to be affected by the message. For example, internal ID **402** may identify an associated data value within attribute **210**. The data value within attribute **210** may represent a feature, or a function in the network element **108***a*–**108***n* associated with managed object **204**.

Message type **306** within message element **400** is copied from message type **306** within message **300**. Message type **306** identifies the type of message. For example, message type **306** may identify message **300**as a get-attribute message. Accordingly, when message type **306** is copied to message element **400**, message element **400** is identified as a get-attribute message element. In such an example, body **206** would route message element **400** to attribute **210**. The data field of the message element **400** would contain no data, since the data is to be retrieved from the target attribute **210**. Internal ID **402** identifies the particular data member within attribute **210** from which data is being requested. Attribute **210** responds to the get-atrribute message element **400** by generating a response message element **400**, including the values of the data members identified by internal ID **402** in attribute **210**. The response message element **400** is sent to body **206** of managed object **204**. The response message element **400** includes a message type **306** indicating that the message element is a get-attribute response message element. Internal ID **402** of the response message element **400** is unchanged, thereby identifying the response message element **400** as a response to the original get-attribute message element **400** when the message element **400** is received at the body **206**. In an alternative embodiment, internal ID **402** of the response message element is algorithmically determined within attribute **210** based on the get-attribute message element Internal ID **402** received by the attribute **210** from the body **206**.

FIGS. **3**. and **4** illustrate embodiments of message **300** and message element **400** in accordance with the present invention. It should be noted, however, that one of the features of the present invention is the flexibility with which messaging systems, or protocols, can be accommodated. If, for example, the network management system **100** includes a plurality of hierarchical levels, the protocol of the messages and message elements can be changed to accommodate the nature of the network entities that will be receiving the messages.

Characteristics **208** on managed object **204** determine the nature of the messages to be sent from managed object **204** to network elements **108***a*–**108***n*. For example, if managed object **204** receives a get-attribute request message, the message is parsed into message elements **400** and passed to the appropriate attribute **210** in managed object **204**. If attribute **210** is associated with particular data values stored by network element **108***a*, managed object **204** preferably sends a get-attribute request message to network element **108***a* in order to retrieve the data associated with the get-attribute request message received by managed object **204**.

Network element **108***a* receives the get-attribute request message from network manager **106** and responds by sending a get-attribute response message containing the data requested to network manager **106**. The get-attribute response message is passed to managed object **204**, where the get-attribute response message is parsed into message elements **400** as described above. Attribute **210** is updated with the data **404** in message element **400** from network element **108***a*. After attribute **210** is updated with data **404** from message element **400**, attribute **210** sends a response message element **400** to body **206**. Body **206** translates the response message element **400** into a response message **300** containing the data to be sent to the requesting client application.

FIG. **5** further illustrates one embodiment of the network element **108***a*. Network element **108***a* comprises managed object **502** which in turn comprises body **504** and characteristics **506**. Characteristic **506** includes attributes **508**, actions **510** or both. Body **504** controls the behavior of the managed object **502**. Messages sent by managed object **502** to network manager **106** are processed and generated by body **206** within the network manager **106**, as described above. In a manner similar to that described above with regard to managed object **204**, any message sent by managed object **502** is generated by body **504**. Characteristics **506** represent the data and functionality that is available at network element **108***a*–**108***n*.

Managed object **502** interacts with the hardware of network element **108***a*. For example, message **300** is received from network manager **106** at network element **108***a*. OID **302** identifies the particular network element **108***a* and managed object **502** to which message **300** is directed. Managed object **502** parses received message **300** into message elements **400**, at body **504**. In one case, characteristics **506** respond to received message element **400** passed to them by body **504** by executing an action at the network element (e.g., switching a digital cross-connect, changing the configuration of the network element, etc.) or setting data attributes **508** with the data **404** in message **400**. Alternatively, message **300** received at network element **108***a* may request information owned by network element **108***a*. In such instances, managed object **502** will formulate a response message **300** and transmit it over the network to network manager **106**.

Partial Response Table

In the preferred embodiment, network management system **100** is a message based system. Clients, such as users, applications or systems, interact with network manager **106** by sending asynchronous messages. Network manager **106** in network management layer **102** communicates with net-

work elements **108***a*–**108***n* in network element layer **104** by transmitting and receiving similar messages. Such a messaging system is said to be transaction based. Transaction based systems rely upon the messages and responses thereto in order to accomplish the management of the network. In asynchronous message based systems, such as network management system **100**, messages are transmitted without acknowledgment from the network management system or the entity to which the message is transmitted.

Modern telecommunications networks have tens of thousands of network elements. Asynchronous messaging systems ensure efficient use of network management resources. However, tracking the thousands of pending and outstanding messages between network manager **106** and network elements **108***a*–**108***n* or between network manager **106** and the client poses a difficult management problem.

A partial response table in accordance with the present invention provides a mechanism for managing messages transmitted by network entities of network management system **100**. For the purposes of explanation, the partial response table is described as a table with a series of related rows, or tuples. In actuality, the partial response table may be implemented on any addressable storage device, through the method described herein. In the preferred embodiment, a partial response table is implemented by each managed object **204**, **502**. Alternative embodiments of the partial response table may be implemented in the hardware of the network manager **106**, network elements **108***a*–**108***n*, subsystem server **202**, or on an independent general computer system.

FIG. **6** illustrates the preferred embodiment of partial response table **600**. Each managed object **204**, **502** preferably has a partial response table **600**. In the preferred embodiment, when managed object **204**, **502** receives a message **300**, managed object **204**, **502** creates a major row **602** in the partial response table **600** associated with the message **300**. The major row **602** remains in the partial response table **600** until managed object **204**, **502** responds to the message **300**. If managed object **204**, **502** must send secondary messages to retrieve the information or implement the command of message **300**, minor rows **604***a*–*n* are added to the partial response table **600** associated with each secondary message sent to an additional network entity, such as a network element **108**. When responses are received from the secondary messages, the associated minor rows **604** are deleted from the partial response table **600**. The partial response table **600** and the process of addition and deletion of major and minor rows **602**, **604** in response to transmission and reception of messages is described in more detail hereinbelow.

Partial response table **600** comprises at least one major row **602**. Each major row **602** corresponds to a message received by managed object **204** or **502**. Each major row in partial response table **600** is uniquely identified by XID **304** and OID **302**. Each major row **602** of partial response table **600** includes a parameter list **606**.

One major row **602** is created for each message from the client received by managed object **204**, **502**.. Each minor row **604***a*–**604***n* is associated with a particular major row **602**. Each minor row **604***a*–**604***n* is uniquely identified by XID **304**, OID **302** and ID **402**. Each minor row **604***a*–**604***n* also includes associated data **608**. XID **304** and OID **302** of minor row **604***a*–**604***n* allows managed object **204** to determine to which major row **602** each minor row **604***a*–**604***n* belongs. Parameter list **606** is a set of data fields, each of which corresponds to a data member that will be transmitted in response to the message that corresponds to the minor row **604***a*–**604***n*.

FIG. **7** illustrates one example of a network management hierarchy. A brief overview of the operation messaging system will be described in conjunction with FIG. **7**. The network management system of FIG. **7** will be used in conjunction with FIGS. **8**–**9** to describe the operation of the partial response table and method of the present invention in more detail below.

A command line interface (CLI) **702** is one example of a client application for accessing network management system **100**. In accordance with one embodiment, CLI **702** is a terminal that controls the network elements in the network management hierarchy. Alternatively, CLI **702** is a software application designed to manage the network. However, it should be understood that CLI **702** may be any software, hardware, or combination which is capable of originating a request message **704**. CLI **702** sends request messages **704** to network manager **106**. Network manager **106** passes CLI request message **704** to subsystem server **202**. Subsystem server passes CLI request message **704** to managed object **204**. Managed object **204** processes CLI request message **704** according to the process of the present invention. Managed object **204** in turn generates a managed object request message **710** in response to CLI request message **704**. Alternatively, the network manager **106** generates a message to send to the subsystem server **202** based upon the information contained in the CLI request message **702**. Likewise, in one embodiment, subsystem server **202** generates a message to send to managed object **204** based upon the information contained in the message received by managed object **204** from subsystem server **202**.

Network element **108***a* receives a managed object request message **710** and passes it to managed object **502**. Managed object **502** processes request message **710** according to the present invention and responds by sending a managed object response message **714** to managed object **204**. Response message **714** contains the data or action confirmation message corresponding to request message **710**. Request message **710** and response message **714** represent a single "transaction."

Managed object **204** processes response message **714** according to the present invention and generates a CLI response message **716**, which provides the data or action confirmation corresponding to CLI request message **704**. Although the environment of the present invention is illustrated as a single network manager **106**, managed object **204** and corresponding network element **108***a* and managed object **502**, it should be understood that the environment of the present invention preferably includes thousands of such network entities and managed objects.

For the purposes of explanation, managed object **204** has data members represented by a characteristic **706** (attribute **1**) and a characteristic **708** (attribute **2**). Managed object **502** has a data member represented by the actual value of attribute **2** stored at characteristic **712**. Attribute **2** of characteristic **708** is a logical representation at network manager **106** of actual data member in characteristic **712**. In order for managed object **204** to respond to CLI **702** with the value of characteristic **708** (i.e., generate a CLI response message **716**), the value of attribute **2** of characteristic **712** is preferably retrieved.

FIG. **8** illustrates the process of receiving a CLI request message **704** at network manager **106** and generating a CLI response message **716**. Step **801** sends CLI request message **704** from CLI **702** to network manager **106**. Step **802** receives CLI request message **704** at network manager **106**. Step **804** passes received CLI request message **704** to subsystem server **202** from network manager **106**. Step **806**

receives CLI request message **704** and determines the OID **302** of the received CLI request message **704** at subsystem server **202**. Step **808** determines whether managed object **204** identified by OID **302** is present on subsystem server **202**. Step **810** retrieves managed object **204** identified by OID **302** from persistent store if it is not present on subsystem server **202**, as determined by step **808**. If, on the other hand, managed object **204** identified by OID **302** exists on the subsystem server when the message is received, the process continues at step **812**.

If the managed object **204** identified by OID **302** exists on subsystem server **202**, then, in step **812**, managed object **204** parses CLI request message **704** into message elements **400** and creates major row **602** in the partial response table **600** associated with CLI request message **704**. Major row **602** comprises XID **304**, OID **302** and parameter list **606**. Parameter list **606** is a set of data fields each of which corresponds to a data member that will be transmitted (i.e., outgoing parameters) in CLI response message **716** to CLI **702**. For example, network manager **106** receives a request message from CLI **702** to provide the lock status of network element **108***a*. The lock status of network element **108***a* is expressed as a single data member. The parameter list **606** in the outgoing message from network element **108***a*, therefore, would contain a single data member representing the lock status of network element **108***a*.

In Step **814**, body **206** passes each of the message elements **400** generated by step **812** to its associated characteristics **706**, **708** in the managed object **204**. In Step **816**, the associated characteristics **706**, **708** process each message element **400**. In Step **818**, the characteristics **706**, **708** determine if an adequate response to message element **400** can be provided. In the preferred embodiment, step **818** is performed by characteristics **706**, **708**. In the example of FIG. **7**, attribute **1** of characteristic **706** is local to managed object **204**. Characteristic **706**, therefore, can respond to the request of the message element **400** passed to it by body **206**. Attribute **2** of characteristic **708**, on the other hand, is a logical representation of attribute **2**, which originates from managed object **502** in characteristic **712**. In the preferred embodiment, in order for managed object **204** to respond adequately to the CLI request message **704** for attributes **1** and **2**, therefore, managed object **204** must retrieve the data members of attribute **2** from managed object **502** on network element **108***a*.

Examples of attributes local to managed object **204** include: faults that the network manager **106** issues, the name of the managed object, and other such values that the network management layer **102** needs in order to provide information to the user, but which network elements **108***a*–**108***n* do not need in order to provide data associated with the network elements **108***a*–**108***n*, in response to requests. Often, however, data requested by get-attribute messages, or action messages, is not available at managed object **204** on network manager **106**. Examples of such data are the frame error rate, the severely errored seconds, or any other data collected at network elements **108***a*–**108***n*. In such cases, it is necessary for managed object **204** on network manager **106** to send a request to network element layer **104** in order to retrieve the information necessary to fulfill the request.

If in step **818**, it is determined that characteristics **706**, **708** can fulfill CLI request message **704** received by network manager **106**, then in step **832**, managed object **204** responds to CLI request message **704** with CLI response message **716** which includes the data requested. Since the transaction with CLI **702** has been completed, in step **834**, managed object **204** deletes the major row **602** in partial response table **600**.

If, on the other hand, characteristics **706**, **708** determine that the CLI request message **704** cannot be fulfilled at managed object **204**, the process continues at step **820**. In the particular example of FIG. **7**, attribute **1** of characteristic **706** is a network management layer characteristic. The data members of characteristic **706**, therefore, are available on managed object **204**. In Step **822**, characteristic **706** responds to message element **400** by forwarding a request for a message to body **206**. In the present example, characteristic **708** has determined that the data members of attribute **2** are not available and must be retrieved from network element **108***a*. Characteristic **708** passes to body **206** the values necessary to formulate managed object request message **710** to retrieve the data members of attribute **2** from managed object **502** on network element **108***a*.

In Step **822**, body **206** generates managed object request message **710**. In Step **824**, managed object **204** adds minor row **604***a* to partial response table **600**. Minor row **604***a* includes XID **304**, OID **302** and ID **402** associated with characteristic **708** of managed object **204**, and the data requested from network element **108***a*. Minor row **604***a* serves as a place holder at the managed object **204** for the outstanding request to another network entity.

Although the example of FIGS. **7** and **8** show a single attribute request forwarded to network element **108***a*, often a single CLI request message **704** results in multiple minor rows **604***a*–**604***n* being added to partial response table **600**, associated with major row **602**. Each minor row **604***a*–**604***n* is associated with a particular managed object request message **710**. In such cases, multiple managed object request messages **710** are sent to multiple network entities in network management system **100**. Furthermore, a single minor row **604***a* may represent the consolidation of multiple requests to a single network element **108***a*. For example, multiple requests for attributes on a single network element **108***a* may be consolidated into a single managed object request message **710** at minor row **604***a*, resulting in a more efficient method of handling management messaging between network entities.

In Step **826**, managed object **204** sends managed object request message **710** to managed object **502** at network element **108***a*. Managed object **502** processes managed object request message **710** in the manner described above. Managed object request message **710** may result in the generation of additional request messages which, in turn, are sent to additional network entities. The additional request messages are generated in response to requests for characteristics not owned by network element **108***a*. In such cases, CLI request message **704** can result in a cascade of request and response messages throughout network management system **100**. One of the advantages of the present invention is that a single implementation of the partial response table system and method manages and tracks the outstanding messages in network management system **100** at all levels in the network management hierarchy.

In Step **830**, managed object **504** sends, and managed object **204** receives, managed object response message **714**. In Step **830**, minor row **604***a* associated with managed object request message **710** is deleted upon the receipt of managed object response message **714**. In Step **832**, network manager **106** generates the CLI response message **716** and sends it to CLI **702** after managed object **204** processes managed object response message **714** in order to update the data members of attribute **2** of characteristic **708**. CLI response message **716** includes the data requested by CLI request message **704**. CLI response message **716** concludes