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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 February 2002 (28.02.2002)

PCT

(10) International Publication Number
WO 02/17596 A2

(51) International Patent Classification[7]: H04L 29/06

(21) International Application Number: PCT/US01/41937

(22) International Filing Date: 27 August 2001 (27.08.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/648,897 25 August 2000 (25.08.2000) US

(71) Applicant: **TRUEDISK.COM, INC.** [US/US]; 7431 NW Evergreen Parkway, Suite 110, Hillsboro, OR 97124 (US).

(72) Inventors: **STEERE, David, C.**; 15700 SW Bridle Hills Drive, Beaverton, OR 97007 (US). **OTTO, Steven, W.**; 930 NW 12th, Apt. 528, Portland, OR 97209 (US). **McNAMEE, Dylan, J.**; 143 SE 32nd Avenue, Portland, OR 97214 (US). **STUPAK, Michael**; 2328 NW Glisan #4, Portland, OR 97210 (US).

(74) Agents: **DIEHL, Robert, A.** et al.; Columbia IP Law Group, PC, 4900 SW Meadows Road, Suite 109, Lake Oswego, OR 97035 (US).

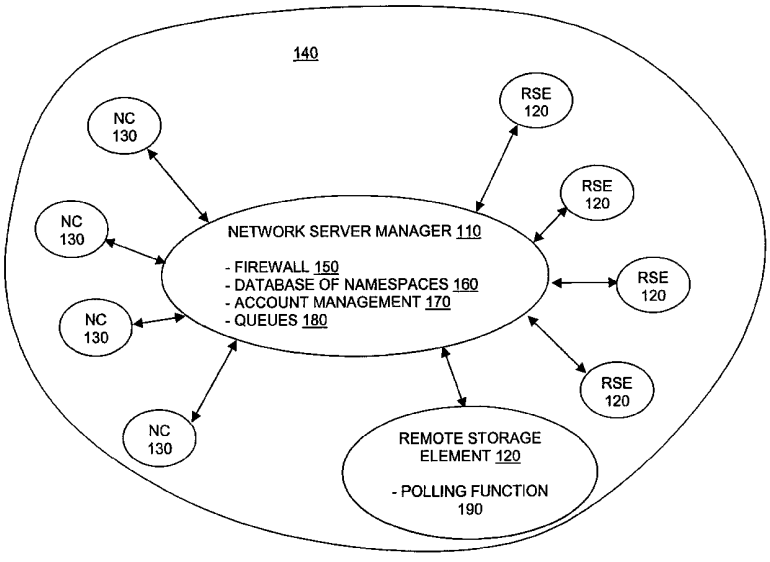
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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

[Continued on next page]

(54) Title: A SECURE DISTRIBUTED SERVER IN AN INSECURE NETWORK



(57) Abstract: A distributed network server includes a network server manager and a number of distributed remote storage elements. The network client to be processed by one of the remote storage elements. The network server manager identifies which of the remote storage elements is to process the first request. The remote storage elements, however, are inaccessible except in response to requests from respective remote storage elements. In which case, network server manager waits to receive a second request from the identified remote storage element. The second request polls the network server manager for any network client requests to be processed by the identified remote storage element. The network server manager sends a response to the second request to the identified remote storage element including an indication of the first request. When the identified remote storage element receives the response, the storage element processes the included request.

WO 02/17596 A2

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A SECURE DISTRIBUTED SERVER IN AN INSECURE NETWORK

Field of the Invention

The present invention pertains to the field of networking. More particularly, this invention relates to providing a secure distributed server in an insecure network.

Background

Networks continue to revolutionize business, entertainment, communications, and countless other aspects of every day life. The potential uses for networks appear to be virtually unlimited, especially for public networks such as the Internet. As the numbers of public network users and content providers continue to increase however, managing access to the vast quantities of available data presents certain challenges. Large network servers may need to provide millions of web pages worth of data to millions of simultaneous network clients. Security, convenience, and scalability to accommodate increasingly larger volumes of data and data traffic are but a few of the challenges facing network server designers.

A wide variety of approaches have been used to implement network servers. For instance, some content providers have turned to large-scale computers that have huge memory banks and vast computing power. Large scale computers, however, tend to be very expensive to buy, maintain, and upgrade. Consequently, large-scale computers often lack the scalability desired to accommodate the virtually limitless increases in data traffic that may be experienced on a public network.

Another approach to implementing a network server involves using a private network to create a distributed network server. Resources within the private network are shared to service requests received from another network. For instance, a company may maintain a private local area network (LAN). The private LAN may include one or more file servers to store the company's data. The LAN may also include an internet interface coupled to the public

Internet to receive requests from Internet clients. Client requests can be processed by retrieving data from file servers in the private LAN.

Compared to a single large-scale computer, a distributed network server may be less expensive and more scalable up to a certain point. For instance, if the existing number of file servers is insufficient to store the volume of data needed, it may be possible to add another file server to the private LAN to store the excess volume. If, however, the data rate of the private LAN is insufficient to keep up with the volume of data traffic, or if the private LAN cannot support an additional file server, upgrading the entire private LAN can involve large investments in new hardware and software.

Another approach to implementing a distributed network server relies on virtual private network technology. In a virtual private network, network entities in a public network use various protocols, cryptography, and the like to share information among themselves in the public network with a heightened degree of security. For instance, an employee may access his or her company's computer from home by establishing a "tunneling" session through the Internet. The Internet is a public, insecure network so anyone can access the network and intercept data. The tunneling session, however, creates a virtual private connection between the employee at home and the company's computer. Data sent through the tunnel can be cryptographically encoded to protect it if it is intercepted along the way.

Using virtual private network technology, multiple servers can be networked together to share server resources to create a distributed network server. Compared to a single large-scale computer or a private LAN, a virtual private network is usually more scalable. For instance, a virtual private network utilizes available public network resources. If a distributed network server needs to support a higher data rate among its networked servers, a higher data rate can often be achieved by simply buying more bandwidth from a public network service provider. Furthermore, if the existing number of networked servers is insufficient to handle the volume of data or data traffic, using public

network resources allows a virtually limitless number of additional servers to be added to the virtual private network to take on the excess volume.

Although virtual private network technology provides heightened security among the networked servers and often provides improved scalability over a large-scale computer or a private LAN, a virtual private network has a number of drawbacks. For instance, a virtual private network may be particularly susceptible to hacker attacks. Each of the multiple servers that make up the distributed network server in a virtual private network monitors a port coupled to the public network for requests either from network clients or other networked servers. Once a hacker knows the address of one of the ports on the public network, the hacker can repeatedly send data to that network port until the hacker finds some combination of data to which the corresponding server will respond. Since each networked server monitors a port, the distributed network server provides multiple points for a hacker to attack, and the entire distributed server may only be as secure as its weakest link. That is, if one of the networked servers falls prey to a hacker, the hacker may be able to access the other networked servers through the virtual private network.

In which case, each networked server needs to provide its own security to defend against hacker attacks. Any number of hardware and/or software solutions, or firewalls, can be used to provide security. Security, however, is difficult to get right. A determined and resourceful hacker can take advantage of virtually any loophole in a firewall. Security solutions that minimize loopholes tend to be expensive, and the cost is compounded in a distributed network server that relies on virtual private network technology because each networked server is likely to need the same security solution.

Another potential drawback, at least compared to a single large-scale computer, is that each networked server in a virtual private network requires "server-like" software. A network server often requires more robust, more expensive hardware and software than a network client because a server needs to perform functions like monitoring one or more network ports, recognizing appropriate requests from potentially thousands or even millions of

requests received each day, processing large numbers of requests in any number of ways, and issuing large numbers of responses to requests. Clients, on the other hand, do not respond to requests. Instead, clients merely issue requests and wait for corresponding responses.

Of course, the three approaches to network servers discussed above can be used in a variety of combinations. For instance, one or more large scale computers may be combined with one or more private LANs using virtual private network technology to create an even larger distributed network server. Combining the various approaches, however, tends to heighten their various drawbacks as a result of increased complexity, higher maintenance costs, and the like.

BRIEF DESCRIPTION OF THE DRAWINGS

Examples of the present invention are illustrated in the accompanying drawings. The accompanying drawings, however, do not limit the scope of the present invention. Similar references in the drawings indicate similar elements.

Figure 1 illustrates one embodiment of the present invention.

Figure 2 demonstrates one embodiment of the present invention from the perspective of a network server manager.

Figure 3 demonstrates one embodiment of the present invention from the perspective of a remote storage element.

Figure 4 illustrates one embodiment of a hardware system.

Figure 5 illustrates one embodiment of a machine readable storage medium.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, those skilled in the art will understand that the present invention may be practiced without these specific details, that the present invention is not limited to the depicted embodiments, and that the present invention may be practiced in a variety of alternate embodiments. In other instances, well known

methods, procedures, components, and circuits have not been described in detail.

Parts of the description will be presented using terminology commonly employed by those skilled in the art to convey the substance of their work to others skilled in the art. Also, parts of the description will be presented in terms of operations performed through the execution of programming instructions. As well understood by those skilled in the art, these operations often take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, and otherwise manipulated through, for instance, electrical components.

Various operations will be described as multiple discrete steps performed in turn in a manner that is helpful in understanding the present invention. However, the order of description should not be construed as to imply that these operations are necessarily performed in the order they are presented, or even order dependent. Lastly, repeated usage of the phrase "in one embodiment" does not necessarily refer to the same embodiment, although it may.

As used herein, "server" generally refers to one or more machines running server software to service requests received from other machines. Conversely, "client" generally refers to one or more machines running client software to request service from a server.

The present invention comprises a secure distributed server in an insecure network. As discussed more fully below, various embodiments of the present invention improve network server scalability, convenience, and security. In general, the present invention separates fairly straight forward aspects of network server functionality that are resource intensive, such as data storage and processing, from comparatively complex or mundane functions that are less resource intensive, such as server security and client account management.

Figure 1 illustrates one embodiment of the present invention. Insecure network 140 includes a network server manager 110, a number of remote storage elements 120, and a number of network clients 130. In the illustrated

embodiment, network server manager 110 and remote storage elements 120 together comprise a distributed network server to service requests from network clients 130.

In the illustrated embodiment, network server manager 110 primarily performs administrative functions that are not particularly resource intensive, such as server security and client account management. The remote storage elements 120 perform more resource intensive functions, such as distributed data storage and processing. Manager 110 and storage elements 120 communicate with one another through the insecure network 140. However, they do not rely on virtual private network technology (having all the associated drawbacks) to provide security in the insecure network 140.

Unlike a virtual private network in which each networked server monitors a port coupled the insecure network 140 for requests, the remote storage elements 120 do not monitor their respective network ports for requests. As a result, the remote storage elements 120 are essentially immune from hacker attacks. That is, a hacker can send as much data as he or she wants to a remote storage element. The hacker will not receive a response because the remote storage elements simply are not "listening" to requests.

The network server manager 110 essentially directs traffic between network clients 130 and remote storage elements 120 so that clients 130 and elements 120 never have to communicate directly with one another. Network clients 130 send requests to network server manager 110 to be serviced by the distributed network server. Remote storage elements 120 also send requests to network server manager 110. A request from a remote storage element polls the network server manager 110 for any network client requests that are to be serviced by the respective remote storage element. The network server manager responds to the respective remote storage element including whatever information the remote storage element needs to process any network client requests intended for the respective remote storage element that have been received and remain to be processed. If a network client request requires a response from the distributed network server, the remote storage

element may send another request to the network server manager 110 after processing the network client request including whatever information the network server manager 110 needs to issue a response to the network client.

Although a remote storage element provides whatever information is needed to service a network client request, a remote storage element does not act like a server in the classic server sense. That is, a classic server monitors a port for requests that it can service because the server never knows when a request might be received. On the other hand, a remote storage element only "listens" on insecure network 140 for a response to its own initiated request. Since it is not responsive to requests initiated by other network elements, it is essentially immune to hacker attacks.

Of course, network server manager 110 does monitor one or more ports coupled to insecure network 140 for requests initiated by other elements in the network. In which case, network server manager 110 is not similarly immune to hacker attacks. But, unlike a virtual private network in which each networked server requires its own security, network server manager 110 provides a single security portal for the entire distributed network. In other words, network server manager 110 is the only element in the inventive distributed server that needs a potentially expensive security solution. Virtually all of the security concerns for the inventive distributed network server can be shifted to the network server manager 110.

A further advantage of the present invention compared to virtual private network technology is that network server manager 110 is the only element in the distributed server that needs "server-like" hardware and software to monitor requests. Each remote storage element 120 can be implemented using less expensive, less complicated "client-like" hardware and software. That is, each remote storage element 120 is essentially a client of network server manager 110.

Like a virtual private network, the inventive distributed network server takes advantage of available network resources to provide communications among the various distributed server components. Improved data rates

between elements can often be purchased and a virtually limitless number of remote storage units can be added to handle increases in data volume and data traffic.

By separating the processing power of a distributed server from the administrative overhead, and by providing secure communications using comparatively simple and inexpensive client-server relationships among distributed components, the present invention provides an improved distributed network server. The inventive server can be used for any number of purposes and can take any number of forms. A number of examples are discussed below.

Under various circumstances, a remote storage element 120 may also be a network client 130. For instance, when a remote storage element sends a request to network server manager to read data stored on another remote storage element, the request may be handled just like any other network client request.

The network server manager 110 can be implemented in any number of forms. For instance, network server manager 110 may be a single computer, a private LAN, a virtual private network, or any combination thereof. In whatever form, network server manager 110 only needs a relatively low amount of processing power to direct traffic between network clients and remote storage elements.

Remote storage elements 120 can also take any number of forms. For instance, a remote storage element may be anything from one or more large-scale computers down to a small hand-held device. Collectively, the remote storage elements 120 provide the bulk of data storage and processing power for the distributed storage.

Insecure network 140 may be any number of networks, such as the public Internet. In which case, communications among the components of the distributed network server can be implemented using standard internet protocols including requests and responses in, for instance, hypertext transfer protocol (HTTP).

In the illustrated embodiment, network server manager 110 includes a number of administrative and security functions including firewall 150, a database of namespaces 160, client account management 170, and client request queues 180. Other embodiments may include any number of additional functions or may not include all of the illustrated functions.

Firewall 150 can be implemented in any number of ways. For instance, in one embodiment, firewall 150 comprises a computer that physically separates a private local area network (LAN) from the insecure network 140 so that components in the private LAN cannot be directly accessed from the insecure network 140. In another embodiment, firewall 150 is implemented in software on the same machine that performs the other functions of the network server manager 110. In either case, firewall 150 is intended to prevent unauthorized access to network server manager 110. Firewall 150 can use any number of approaches to authenticate and authorize access to the distributed server.

The database 160 is used to route network client requests to respective remote storage elements. In one embodiment, the database defines a virtual memory that includes all of the remote storage elements. In which case, a network client does not need to know which remote storage element stores a particular piece of data. Instead, the network client merely needs to know the virtual address of the piece of data. Based on the database, the network server manager 110 can identify the corresponding remote storage element and the data stored thereon.

In another embodiment, as illustrated in Figure 1, entries in database 160 are namespaces. A namespace is commonly used in internet applications to create and share data using the Extensible Markup Language (XML). A namespace identifies a network address or Uniform Resource Locator (URL) at which information corresponding to the namespace can be found.

Using a database such as database 160, network clients are able to access data on a number of different remote storage elements without knowing exactly where the data is located. This is called location independent naming.

A network client merely needs to know the name of the namespace in order to access the corresponding data. For instance, the data can be moved from one location to another and, as long as the corresponding name space is updated with the new location, a network client can still access the data based on the name of the namespace. In contrast, other distributed storage solutions often require clients to manually copy data directly from where it is stored and to manually manage consistency of data among storage locations. Furthermore, adding storage and/or processing power to the inventive distributed network server can be as easy as adding additional remote storage elements and corresponding database entries. In alternate embodiments, any number of approaches can be used to map network client requests to remote storage elements. For instance, in an alternate database approach, database entries could be implemented in any number of ways including internet protocol (IP) addresses, port numbers, and the like.

Account management 170 can include a variety of management functions. For instance, if network clients have to pay for access to the inventive distributed server, account management 170 could set up accounts, track account activity, store account information either locally or in the remote storage elements, generate billing statements, maintain deposit account balances, and the like. Account management 170 could also include data mining, such as tracking the kinds of data particular network clients access, how often, and the like. Account management 170 could also manage activities such as adding and removing remote storage elements to the database 160, redistributing data among the remote storage elements to balance data loads and data traffic, for instance, as large volumes of data are stored or removed, or as remote storage agents are added or removed. Account management 170 could also include functions like matching network client requests to particular remote storage elements and queuing the requests in queues 180.

Queues 180 store network client requests for corresponding remote storage elements. Depending on the type of network client request, a queue

may only store the request until a remote storage element polls the queue for requests. For instance, if the request is a request to store data, once the data is sent to the respective remote storage element, there may be no reason to continue storing the network client request. On the other hand, if the request is to read data, or if the request requires some kind of response to be sent to the network client or some other network entity, the queue may continue to store the request until the information that is needed to complete the request is received from the corresponding remote storage element. Furthermore, rather than sending the entire request to a remote storage element, it may only be necessary to send an indication of the request, such as an address to be read. In which case, the request may continue to be stored in the queue so that the network server manager can identify where the data is to be sent once the remote storage element returns the data. Any number of queuing techniques can be used to track new network requests, network requests that are waiting for data from a remote storage element, and the like.

In one embodiment, the present invention provides a large scale data sharing system. Data from any remote storage device can be viewed, stored, and/or modified from any authorized network client. From the perspective of remote storage elements, many of the security concerns are shifted out to a third party, namely the network server manager 110. In which case, each remote storage element only needs comparatively simple and inexpensive "client-like" functionality to participate in the data sharing.

The present invention is distinct from traditional third party network security. In traditional third party network security, a client accesses a third party server to get authentication to access a target server. The difference is, in the traditional scheme, the target server is still a server in the classic sense. That is, the target server monitors a port on the network and the client directly accesses the target server. In the present invention, the target device is not a server in the classic sense but instead operates like a client in that it does not monitor a port. Furthermore, in the present invention, the network server

manager provides indirection between the network client and the remote storage element so that the two never have to directly communicate.

In the illustrated embodiment, each remote storage element 120 includes a polling function 190. Polling function 190 can be implemented in any number of ways, including hardware and/or software implementations. The polling function 190 is "client-like" in that it initiates requests to a particular address of the trusted network server manager 110 in network 140 and waits for a response. In various embodiments, polling function 190 may poll network server 110 in any number of ways.

For instance, if a remote storage element is a computer with constant access to the network 140, polling function 190 may automatically poll network server 190 at regular intervals. The intervals may depend on factors such as the anticipated rate of data traffic for the corresponding remote storage element or the nature of the data traffic. For instance, for high volumes of data traffic or time critical data traffic, the polling function 190 may initiate requests several times per second. The polling function 190 may also adapt the interval between requests over time according to configuration settings or automatically adopt as the volume of traffic appears to trend up or down. For instance, the interval may change at different times of day, or different times of the year, or as different kinds of data are stored to the remote storage element.

As another example, if a remote storage element has only intermittent access to the network 140, the polling function may automatically initiate requests at intervals whenever the remote storage element does have access or may only initiate requests when instructed to do so by a user. Again, determining when to initiate requests can depend on any number of factors such as convenience, available resources on the remote storage element, and the like. For instance, polling function 190 may be configured to initiate a request as soon as possible after establishing a connection to network 140, and at regular intervals thereafter, so that a user does not have to worry about it. If a remote storage element has very limited resources though, a user may

not want to expend resources initiating requests and potentially receiving large responses from the network server manager.

Figure 2 demonstrates one embodiment of the present invention as seen from the perspective of a network server manager. In block 210, if a request is received from a network client, the network server manager identifies a remote storage element corresponding to the network client request. For instance, as discussed above, the network server manager may include a database that maps virtual addresses to particular remote storage elements and particular storage locations within the particular storage elements.

A network client request may correspond to multiple remote storage elements where, for instance, the request is to read a large block of data and the block of data spans more than one remote storage element. Similarly, a network client request may correspond to multiple remote storage elements where, for instance, the request is to move data stored in one location in one remote storage element to another location in another remote storage element, or where the request is to process data in some other fashion at one remote storage element and to send a result to another remote storage element.

In the illustrated embodiment, the network server manager also authenticates that the network client that sent the request is authorized to access the requested resources. For instance, as discussed above, the network server manager may include a firewall. The firewall may use any number of criteria to authenticate a network client request. For instance, the firewall may verify that a request came from a known network address and a known port number. A client may also need to provide a password or cryptographic key.

Assuming that a network client request is received, authenticated, and matched to one or more remote storage elements, the network server manager will queue the request and wait for the corresponding remote storage element(s) to poll for requests.

In block 220, the process continues to loop back to block 210 to accumulate more authenticated network client requests. When a request is

received from a remote storage element, the network server manager sends a response back in block 230.

In the illustrated embodiment, the response includes any data that was stored by the network server manager in the corresponding queue and that is to be processed by the particular remote storage element. Requests can include a wide variety of functions including instructions to store data and instructions to initiate some function resident on a remote storage element. Functions resident on a remote storage element could include functions such as reading and returning data, deleting data, moving data, executing software code, activating hardware resources, and the like.

Depending on the nature of what a network client is requesting, a variety of different kinds of data can be sent in the response to the storage element's request. For instance, the data may simply be an address or range of addresses to be read and returned. Conversely, for a write operation, the data could include a data payload along with an address or range of addresses to which the payload is to be stored. For initiating execution of software, the data could include some identifier of the software to be executed as well as a data payload to be processed by the software.

If in block 230 no new data is stored in the queue corresponding to the given remote storage element, one embodiment of the network server manager just sends back an "empty" response. In an alternate embodiment however, since the network server manager cannot contact a remote storage element except in response to a request from the remote storage element, the network server manager holds the request from the remote storage element. In which case, when and if a network client request is received that corresponds to the particular remote storage element, the network server manager can send the appropriate data to the remote storage element in response to the held request.

Any number of approaches can be used to manage the accumulation and correlation of network client requests and storage element requests. For instance, if a remote storage element sends requests several times per

second, it may not be necessary to hold requests from that remote storage element. If a remote storage element is only intermittently connected to the network, it may not help to hold a request from that remote storage element, or it may not help to hold a request from that remote storage element for more than a brief period of time, because the remote storage element may not be available to receive the request when and if it is sent, resulting in the possible loss of data.

Various optimizations can also be applied to queued requests. For instance, if a network client repeatedly sends a request until the network client receives a response, a queued request from the network client may be overwritten by a subsequent equivalent request. Consequently, only one equivalent request from the network client is queued at any one time.

Continuing on with the embodiment illustrated in Figure 2, in block 240, the request from the remote storage element may include data that is needed by the network server manager to complete a previously received network client request. For instance, a previously received network client request may have been a read operation. In which case, the data included in the storage element request may include the data that the network client requested. The network server manager directs any received data according to the corresponding, previously received network client request. In which case, as discussed above, the network server manager may store network client requests even after sending data corresponding to the requests to a remote storage element so that when the remote storage element returns the requested data, the network server manager knows what to do with the data.

On the other hand, in one embodiment, the network server manager sends an entire network client request to the corresponding remote storage element and completely deletes any remnant of the request from the queue. In which case, when the remote storage element returns a response to the request to the network server manager, the remote storage element includes instructions for the network server manager to direct the response to the appropriate recipient.

Various alternate embodiments can divide responsibility for routing responses based on a variety of factors such as the processing power and sophistication of particular remote storage elements compared to the processing power and sophistication of the network server manager. For instance, a small, hand-held remote storage element may not be expected to manage routing of responses, but a large scale computer remote storage element may be expected to manage its own routing.

No matter which element manages or directs routing of responses, the data received from the remote storage element will be routed according. For a read operation, the data is likely to be sent back to the network client in the form of a response to the network client's request. For a move operation, the data may be directed to another remote storage element. In which case, the data may need to be stored in a queue corresponding to the target remote storage element until a request from the target storage element is received. Alternately, as discussed above, if the network server manager is already holding a request from the target remote storage element, the network server manager may be able to immediately direct the data to the target storage element in the form of a response to the held request.

From block 240, the process of Figure 2 loops back to continue accumulating network client requests in blocks 210 and 220, and servicing network client requests based on interaction with remote storage elements in blocks 230 and 240.

Alternate embodiments of the process illustrated in Figure 2 may include a variety of additional elements. For instance, a remote storage element may initiate two different kinds of requests. One request may be used solely to poll the network server manager for network client requests. Another request may be used solely to return data in response to a network client request. In which case, the process of Figure 2 would need to be modified to include separate elements to handle the two different kinds of storage element requests.

In another alternate embodiment, the network server manager may need to hold network client requests for an extended period of time where, for

instance, a remote storage element corresponding to the request is only intermittently connected to the network. In which case, the network server manager may return a response to a network client after some predetermined duration to inform the network client that the requested data is not yet available and to instruct the network client to inquire again later. In this embodiment, when and if the requested data is returned, the network server manager may have to store the data for a period of time until the network client initiates another request for the data.

Figure 3 demonstrates one embodiment of the present invention from the perspective of a remote storage element. In block 310, the remote storage element sends a request to the network server manager to poll for network client requests to be processed by the remote storage element. In block 320, the remote storage element receives a response including an indication of any new network client requests corresponding to the remote storage element that may have accumulated at the network server manager. In the illustrated embodiment, the indication of the requests includes data to be stored and/or locations of data to be read and returned. As discussed above, in alternate embodiments, data sent to the remote storage element in response to polling the network server manager can take many forms, including entire network client requests.

In the illustrated embodiment, in block 330, the remote storage element stores any received data. As discussed above, the remote storage element may or may not receive any data that needs to be stored. In alternate embodiments, even if a network client request is found on the network server manager, certain types of requests, such as read requests, may not require any data to be stored.

In block 340, the remote storage element retrieves any indicated data and, if data is indicated, sends the data back to the network server manager in the form of a new request. In the illustrated embodiment, the new request performs the double functions of returning requested data and polling for new network client requests, and the network server manager directs the returned

data according to the stored network client request. As discussed above, alternate embodiments may use separate requests for returning requested data and polling for new requests, and alternate embodiments may return instructions for the network server manager to direct the requested data so that the network server manager does not need to retain network client requests after they are sent to a corresponding remote storage element.

If, in block 350, a request was sent to the network server manager to return data requested by a network client in block 340, the process returns to block 320 to wait for a response. That is, in the illustrated embodiment, each request from the remote storage element performs the duel rolls of returning any requested data and polling for new requests so the remote storage element will expect a response.

If, in block 350, no response was sent in block 340, the process continues on to block 360 to wait for a period of time. As discussed above, the duration between requests depends on a number of factors including the type of data stored, the type of remote storage element, the kind of access the remote storage element has to the network, and the like. In certain embodiments, the remote storage element automatically determines a duration between requests based on various factors such as those described above. In alternate embodiments, the remote storage element is configured to send requests after certain duration, or is instructed to send a request at a particular time, by, for instance, a user of the remote storage element or by the network server manager. In the latter case, the network server manager can include an instruction in a current request for when the remote storage element should next send a request. In any event, after waiting some period of time, the process loops back to block 310 to send the next request to poll for new network client requests.

Depending on the type of network client requests being serviced by a remote storage element, in the time it takes to service a given request, a remote storage element may make several iterations through the illustrated process. In which case, if the network server manager retains network client

requests until the remote storage element completes servicing the request, the network server manager may need to keep track of which network client requests have already been sent to the remote storage element in order to prevent the same outstanding request from being sent more than once. Alternately, the remote storage element can include functionality to recognize repeat network client requests.

Alternate embodiments may include a variety of additional elements. As discussed above, depending on the extent to which administrative functions are performed by either remote storage elements or the network server manager, the illustrated process may be more complex. For instance, the remote storage element may need to have knowledge of the topology of the inventive distributed server and/or the address of a given network client in order to instruct the network server manager how to route a given response.

Figure 4 illustrates one embodiment of a hardware system intended to represent a broad category of computer systems such as personal computers, workstations, and/or embedded systems. In the illustrated embodiment, the hardware system includes processor 410 coupled to high speed bus 405, which is coupled to input/output (I/O) bus 415 through bus bridge 430. Temporary memory 420 is coupled to bus 505. Permanent memory 440 is coupled to bus 415. I/O device(s) 450 is also coupled to bus 415. I/O device(s) 450 may include a display device, a keyboard, one or more external network interfaces, etc.

Certain embodiments may include additional components, may not require all of the above components, or may combine one or more components. For instance, temporary memory 420 may be on-chip with processor 410. Alternately, permanent memory 440 may be eliminated and temporary memory 420 may be replaced with an electrically erasable programmable read only memory (EEPROM), wherein software routines are executed in place from the EEPROM. Some implementations may employ a single bus, to which all of the components are coupled, or one or more additional buses and bus bridges to which various additional components can

be coupled. Those skilled in the art will be familiar with a variety of alternate internal networks including, for instance, an internal network based on a high speed system bus with a memory controller hub and an I/O controller hub. Additional components may include additional processors, a CD ROM drive, additional memories, and other peripheral components known in the art.

One component in particular that may be included in the hardware system of Figure 4 for implementation of the present invention is a random number generator for security purposes. For instance, communications among the network server manager 110 and the various remote storage elements 120 in Figure 1 can be encrypted to provide secure communications among the distributed elements. Security is only as strong as the keys used to encrypt the data. Keys are often generated using a deterministic algorithm based on some starting number. If the starting number is a predictable value, such as the current date or time of day, the cryptographic keys are more easily compromised, leaving the distributed storage vulnerable to attack. If the starting number is truly random and unpredictable, as provided by a random number generator, the communications are much more secure.

Continuing on with Figure 4, in one embodiment, the network server, remote storage elements, and the network clients, as described above, are each implemented using one or more computers such as the hardware system of Figure 4. Where more than one computer is used, the systems can be coupled to communicate over an external network, such as a local area network (LAN), an internet protocol (IP) network, etc. In one embodiment, the present invention is implemented as software routines executed by one or more execution units within the computer(s). For a given computer, the software routines can be stored on a storage device, such as permanent memory 440.

Alternately, as shown in Figure 5, the software routines can be machine executable instructions 510 stored using any machine readable storage medium 520, such as a diskette, CD-ROM, magnetic tape, digital video or versatile disk (DVD), laser disk, ROM, Flash memory, etc. The series of

instructions need not be stored locally, and could be received from a remote storage device, such as a server on a network, a CD ROM device, a floppy disk, etc., through, for instance, I/O device(s) 450 of Figure 4.

From whatever source, the instructions may be copied from the storage device into temporary memory 420 and then accessed and executed by processor 410. In one implementation, these software routines are written in the C programming language. It is to be appreciated, however, that these routines may be implemented in any of a wide variety of programming languages.

In alternate embodiments, the present invention is implemented in discrete hardware or firmware. For example, one or more application specific integrated circuits (ASICs) could be programmed with one or more of the above described functions of the present invention. In another example, one or more functions of the present invention could be implemented in one or more ASICs on additional circuit boards and the circuit boards could be inserted into the computer(s) described above. In another example, field programmable gate arrays (FPGAs) or static programmable gate arrays (SPGA) could be used to implement one or more functions of the present invention. In yet another example, a combination of hardware and software could be used to implement one or more functions of the present invention.

Thus, a method and apparatus for a secure distributed server in an insecure network is described. Whereas many alterations and modifications of the present invention will be comprehended by a person skilled in the art after having read the foregoing description, it is to be understood that the particular embodiments shown and described by way of illustration are in no way intended to be considered limiting. Therefore, references to details of particular embodiments are not intended to limit the scope of the claims.

CLAIMS

What is claimed is:

1. A method comprising:

receiving a first request at a network server from a network client manager, said first request to be processed by a distributed storage, said distributed storage comprising a plurality of remote storage elements;

identifying a particular remote storage element to process the first request from among the plurality of remote storage elements, said plurality of remote storage elements being inaccessible except in response to requests from respective ones of said plurality of remote storage elements;

waiting to receive a second request at the network server manager from the particular remote storage element, said second request to poll the network server manager for any requests to be processed by the particular remote storage element; and

sending a response to the second request from the network server manager to the particular remote storage element, said response comprising an indication of the first request.

2. The method of claim 1 further comprising:

waiting to receive a third request at the network server manager from the particular remote storage element, said third request comprising previously stored data; and

directing the previously stored data from the network server manager according to the first request.

3. The method of claim 2, wherein the response is a first response, and wherein directing the previously stored data comprises:

sending a second response to the first request from the network server manager to the network client, said second response including the previously stored data.

4. The method of claim 2, wherein directing the previously stored data comprises:

identifying a network entity to receive the previously stored data; and

sending the previously stored data to the identified network entity.

5. The method of claim 4, wherein the response comprises a first response, wherein the particular remote storage element comprises a first remote storage element, wherein the network entity comprises a second remote storage element, and wherein sending the previously stored data comprises:

waiting to receive a fourth request at the network server manager from the second remote storage element; and

sending a second response to the fourth request from the network server manager to the second remote storage element, said second response including the previously stored data.

6. The method of claim 1 wherein the indication of the first request comprises data to be stored to the distributed storage.

7. The method of claim 6 further comprising at least one of:

receiving the data to be stored to the distributed storage from the network client; and

receiving the data to be stored to the distributed storage from a separate network entity indicated by the network client.

8. The method of claim 1 wherein identifying the particular remote storage element comprises:

extracting an identifier of the particular remote storage element included in the first request; and

locating an entry in a database using the identifier, said entry to map the first request to the particular remote storage element.

9. The method of claim 8 wherein the identifier comprises at least one of a namespace, an internet protocol (IP) address, and a port number for the particular remote storage element.

10. The method of claim 1, wherein identifying the particular remote storage element comprises:

authenticating that the network client is authorized to access the particular remote storage element.

11. The method of claim 1 wherein the network client, the network server manager, and the plurality of remote storage elements comprise a public Internet.

12. The method of claim 1 further comprising:

receiving a first plurality of additional requests at the network server manager from a plurality of additional network clients, said first plurality of additional requests to be processed by the distributed storage;

identifying remote storage elements to process the plurality of additional requests from among the plurality of remote storage elements;

waiting to receive a second plurality of additional requests at the network server manager from the identified remote storage elements, said second plurality of additional requests to poll the network server for any requests to be processed by respective ones of the identified remote storage elements; and

sending a plurality of additional responses to the second plurality of additional requests from the network server manager to the respective ones of the identified remote storage elements, said plurality of additional responses comprising indications of respective ones of the first plurality of additional requests.

13. The method of claim 1 wherein waiting to receive the second request comprises:

storing the first request in a queue corresponding to the particular remote storage element.

14. A method comprising:

sending a request to a network server manager from a remote storage element, said request to poll the network server manager for a request to be processed by the remote storage element that has been received by the network server manager from a network client, said remote storage element being one of a plurality of remote storage elements comprising a distributed storage, and said plurality of remote storage elements being inaccessible except in response to requests from respective ones of said plurality of remote storage elements;

receiving a response to the request from the network server manager at the remote storage element, said response including an indication of requests, if any, to be processed by the remote storage element; and

processing any indicated requests.

15. The method of claim 14, wherein the indication comprises data, and wherein processing any indicated requests comprises:

storing the data at the remote storage element.

16. The method of claim 14, wherein the request is a first request, wherein the indication comprises a memory location within the remote storage element, and wherein processing any indicated requests comprises:

retrieving data from the memory location; and

sending a second request to the network server manager from the remote storage element, said second request including the data, and said network server manager to direct the data according to the first request.

17. The method of claim 14 further comprising:

sending a plurality of additional requests at particular intervals to the network server manager to poll the network server manager for new requests to be processed by the remote storage element.

18. A machine readable storage medium having stored thereon machine executable instructions to implement a method comprising:

receiving a first request at a network server from a network client manager, said first request to be processed by a distributed storage, said distributed storage comprising a plurality of remote storage elements;

identifying a particular remote storage element to process the first request from among the plurality of remote storage elements, said plurality of remote storage elements being inaccessible except in response to requests from respective ones of said plurality of remote storage elements;

waiting to receive a second request at the network server manager from the particular remote storage element, said second request to poll the network server manager for any requests to be processed by the particular remote storage element; and

sending a response to the second request from the network server manager to the particular remote storage element, said response comprising an indication of the first request.

19. A machine readable storage medium having stored thereon machine executable instructions to implement a method comprising:

sending a request to a network server manager from a remote storage element, said request to poll the network server manager for a request to be processed by the remote storage element that has been received by the network server manager from a network client, said remote storage element being one of a plurality of remote storage elements comprising a distributed storage, and said plurality of remote storage elements being inaccessible

except in response to requests from respective ones of said plurality of remote storage elements;

receiving a response to the request from the network server manager at the remote storage element, said response including an indication of requests, if any, to be processed by the remote storage element; and

processing any indicated requests.

20. A network server manager comprising:

a network interface to receive a first request from a network client manager, said first request to be processed by a distributed storage, said distributed storage comprising a plurality of remote storage elements;

processing circuitry to identify a particular remote storage element to process the first request from among the plurality of remote storage elements, said plurality of remote storage elements being inaccessible except in response to requests from respective ones of said plurality of remote storage elements;

a queue to store the first request at least until a second request is received at the network interface from the particular remote storage element, said second request to poll the processing circuitry for any requests to be processed by the particular remote storage element; and

said processing circuitry to send a response to the second request to the particular remote storage element, said response comprising an indication of the first request.

21. A remote storage element comprising:

a network interface to send a request to a network server manager, said request to poll the network server manager for a request to be processed by the remote storage element that has been received by the network server manager from a network client, said remote storage element being one of a plurality of remote storage elements comprising a distributed storage, and said

plurality of remote storage elements being inaccessible except in response to requests from respective ones of said plurality of remote storage elements;

said network interface to receive a response to the request from the network server manager, said response including an indication of requests, if any, to be processed by the remote storage element; and

processing circuitry to process any indicated requests.

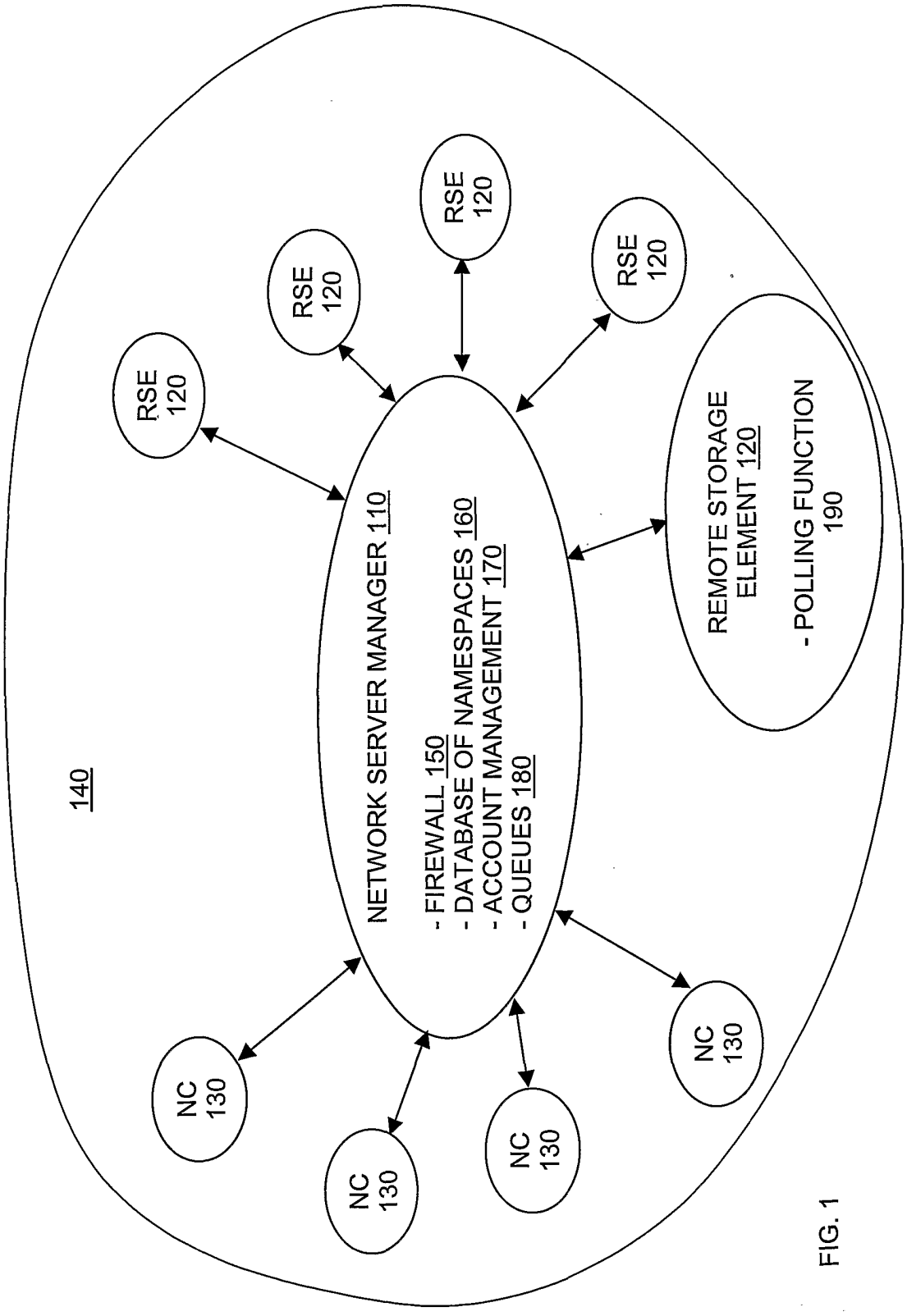

140
NETWORK SERVER MANAGER 110
- FIREWALL 150
- DATABASE OF NAMESPACES 160
- ACCOUNT MANAGEMENT 170
- QUEUES 180
REMOTE STORAGE ELEMENT 120
- POLLING FUNCTION 190
RSE 120
RSE 120
RSE 120
RSE 120
RSE 120
NC 130
NC 130
NC 130
NC 130
NC 130

FIG. 1

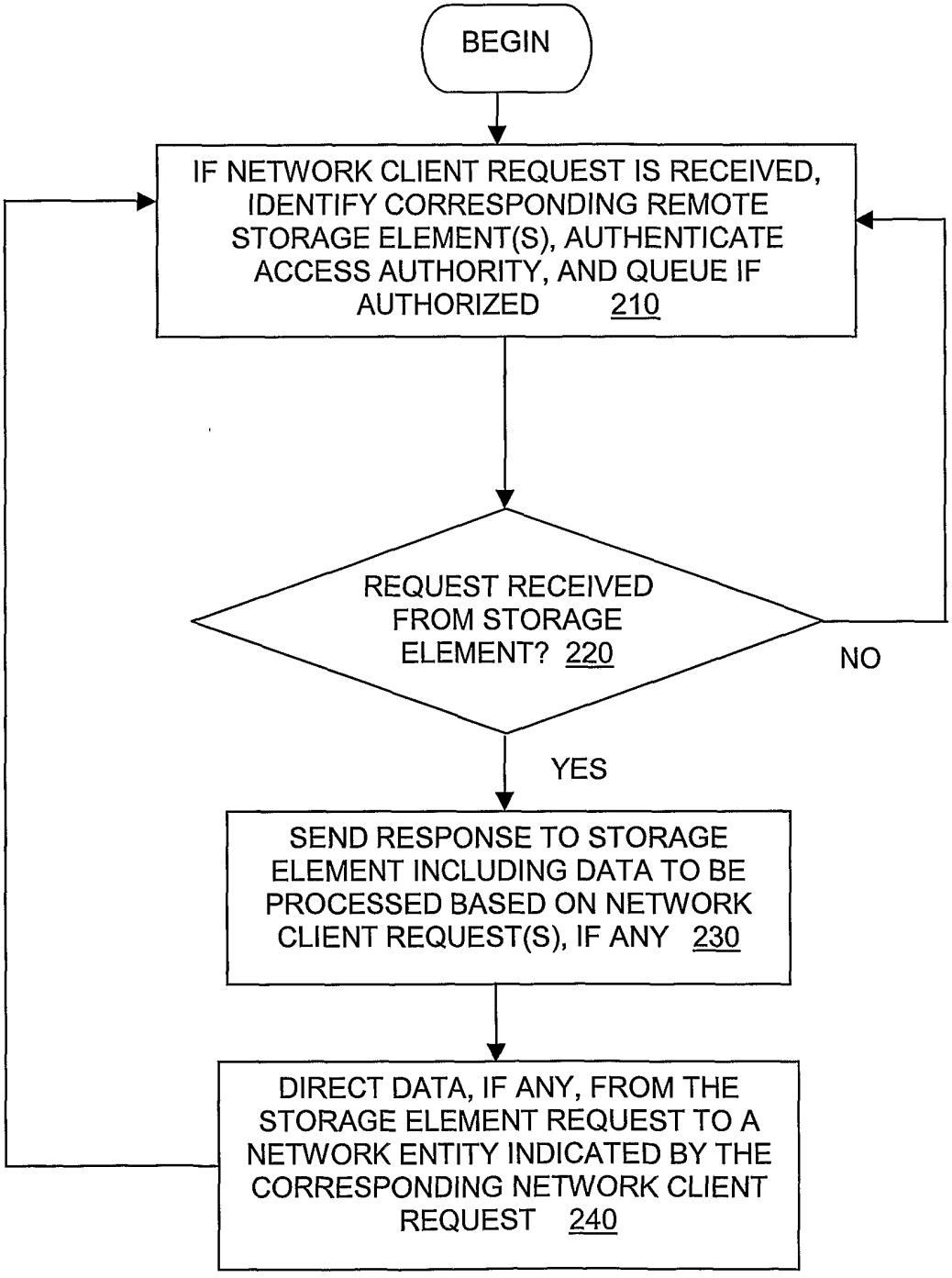

BEGIN
IF NETWORK CLIENT REQUEST IS RECEIVED, IDENTIFY CORRESPONDING REMOTE STORAGE ELEMENT(S), AUTHENTICATE ACCESS AUTHORITY, AND QUEUE IF AUTHORIZED 210
REQUEST RECEIVED FROM STORAGE ELEMENT? 220
NO
YES
SEND RESPONSE TO STORAGE ELEMENT INCLUDING DATA TO BE PROCESSED BASED ON NETWORK CLIENT REQUEST(S), IF ANY 230
DIRECT DATA, IF ANY, FROM THE STORAGE ELEMENT REQUEST TO A NETWORK ENTITY INDICATED BY THE CORRESPONDING NETWORK CLIENT REQUEST 240

FIG. 2

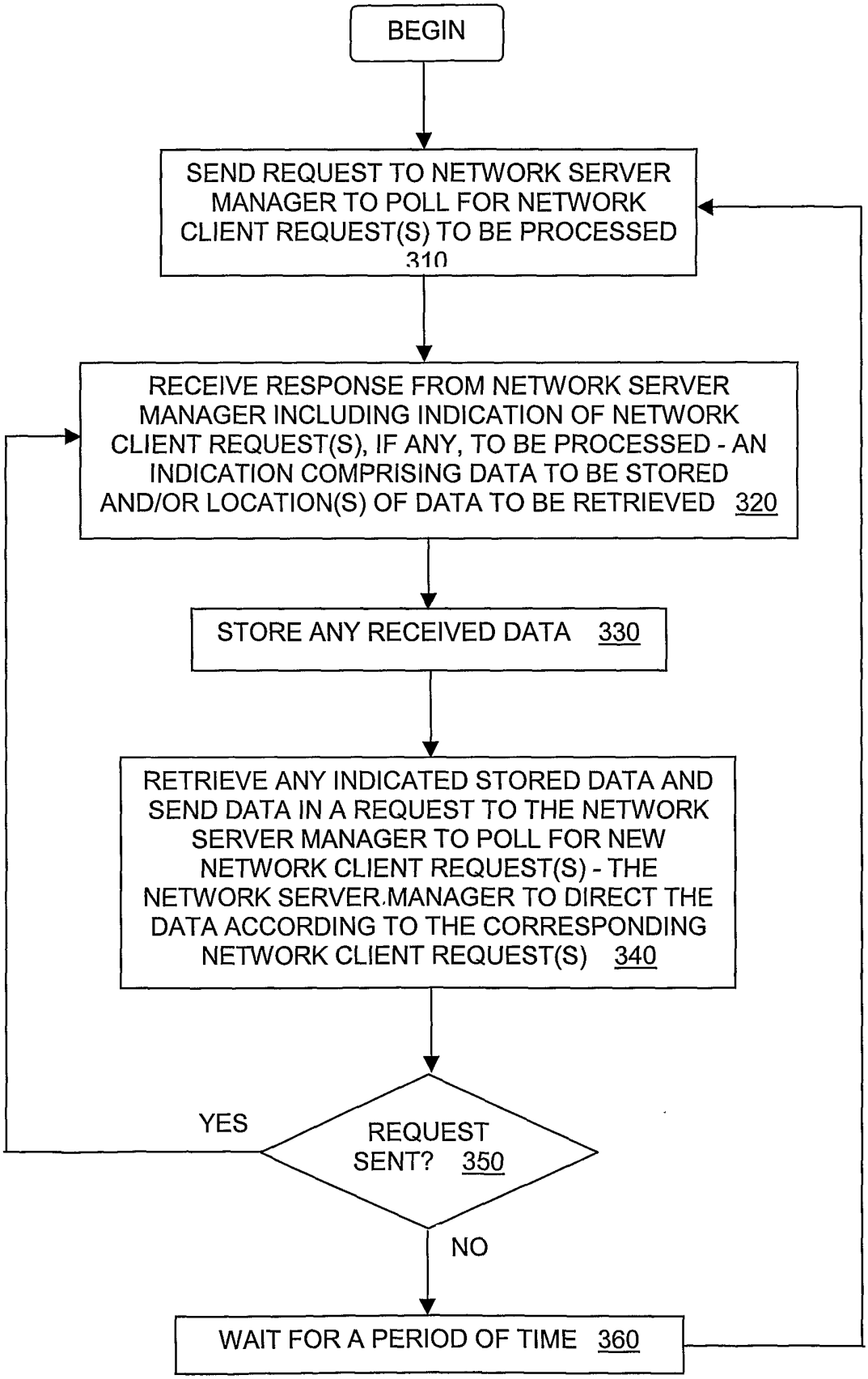

BEGIN
SEND REQUEST TO NETWORK SERVER MANAGER TO POLL FOR NETWORK CLIENT REQUEST(S) TO BE PROCESSED 310
RECEIVE RESPONSE FROM NETWORK SERVER MANAGER INCLUDING INDICATION OF NETWORK CLIENT REQUEST(S), IF ANY, TO BE PROCESSED - AN INDICATION COMPRISING DATA TO BE STORED AND/OR LOCATION(S) OF DATA TO BE RETRIEVED 320
STORE ANY RECEIVED DATA 330
RETRIEVE ANY INDICATED STORED DATA AND SEND DATA IN A REQUEST TO THE NETWORK SERVER MANAGER TO POLL FOR NEW NETWORK CLIENT REQUEST(S) - THE NETWORK SERVER MANAGER TO DIRECT THE DATA ACCORDING TO THE CORRESPONDING NETWORK CLIENT REQUEST(S) 340
REQUEST SENT? 350
YES
NO
WAIT FOR A PERIOD OF TIME 360

FIG. 3

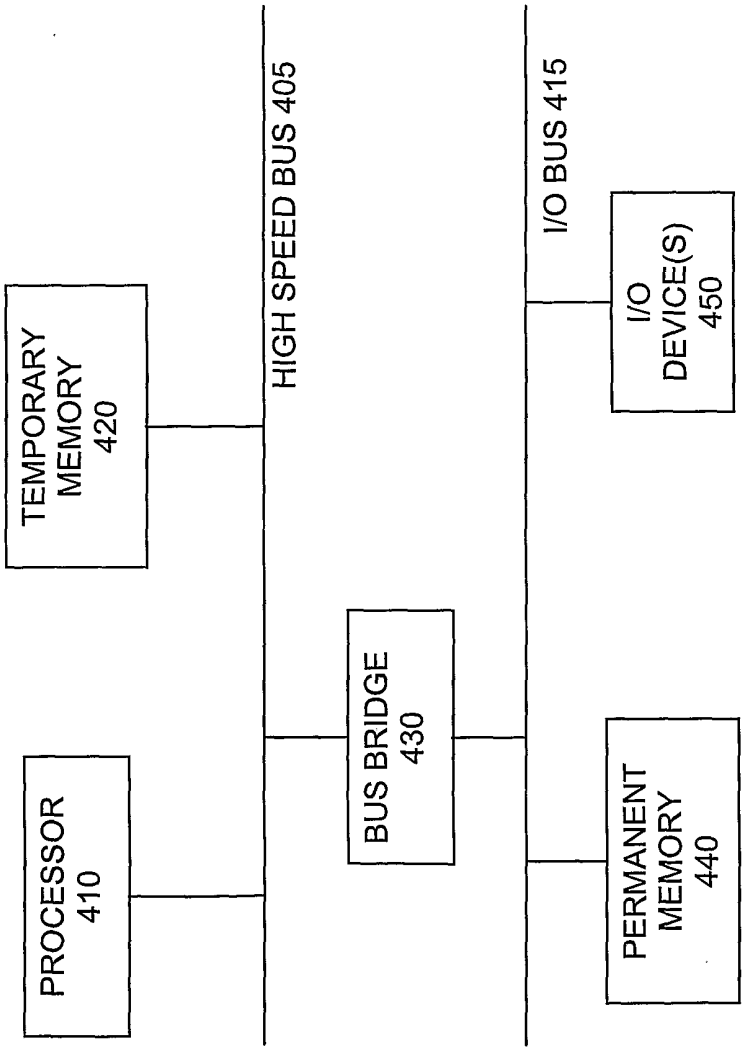

TEMPORARY MEMORY 420
HIGH SPEED BUS 405
I/O BUS 415
I/O DEVICE(S) 450
PROCESSOR 410
BUS BRIDGE 430
PERMANENT MEMORY 440

FIG. 4

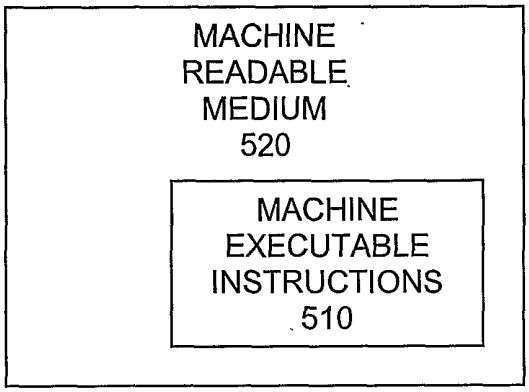

MACHINE
READABLE
MEDIUM
520
MACHINE
EXECUTABLE
INSTRUCTIONS
510

FIG. 5