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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 January 2005 (27.01.2005)

PCT

(10) International Publication Number
WO 2005/008633 A2

(51) International Patent Classification⁷: G11B

(21) International Application Number: PCT/US2004/022521

(22) International Filing Date: 12 July 2004 (12.07.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/486,791 11 July 2003 (11.07.2003) US

(71) Applicant (*for all designated States except US*): **COMPUTER ASSOCIATES THINK, INC.** [US/US]; One Computer Associates Plaza, Islandia, NY 11749 (US).

(72) Inventors: **SCHWARZ, William**; 251 Middle Road, Sayville, NY 11782 (US). **SYED, Aliabbas, H.**; 46 W. Willow Street, Brentwood, NY 11717 (US). **YOUNG, Raymond, J.**; 6 Cara Drive, Ronkonkoma, NY 11779 (US).

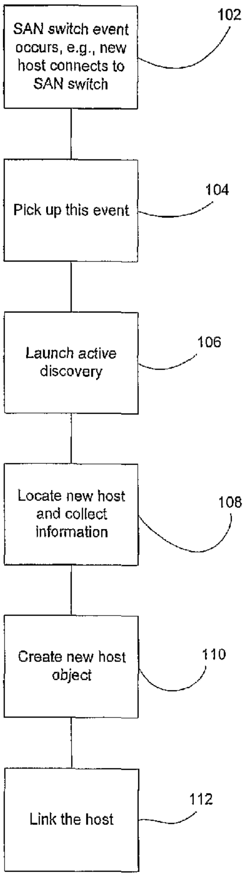
(74) Agents: **PARK, Eunhee** et al.; Baker & McKenzie, 805 Third Avenue - 29th Floor, New York, NY 10022 (US).

(81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, 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, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: ACTIVE STORAGE AREA NETWORK DISCOVERY SYSTEM AND METHOD


SAN switch event occurs, e.g., new host connects to SAN switch
102
Pick up this event
104
Launch active discovery
106
Locate new host and collect information
108
Create new host object
110
Link the host
112

(57) Abstract: An active SAN discovery system and method responds to events occurring in SAN by automatically broadcasting for information related to the occurred events and updating the SAN topology according to the collected information.

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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.

ACTIVE STORAGE AREA NETWORK DISCOVERY SYSTEM AND METHOD

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application No. 60/486,791 entitled ACTIVE SAN DISCOVERY filed on July 11, 2003, the entire disclosure of which is incorporated herein by reference.

TECHNICAL FIELD

This application relates to storage area network management, and particularly to active SAN discovery system and method.

BACKGROUND

The storage area network (SAN) refers to a high-speed special purpose network that interconnects different kinds of data storage devices with associated data servers on behalf of a larger network of users. Providing the storage area network administrator complete, up-to-date information about the SAN without doing a complete sweep of the SAN has been an ongoing problem. Such a discovery effort not only requires manual intervention, but can also take a long time to complete in a large SAN environment. Delay in the time to complete can also lead to an outdated, incorrect view of the SAN and can limit the SAN administrator's flexibility in allocating and maintaining the expensive SAN resources. Accordingly, a discovery method that would overcome the shortcomings of conventional discovery methods is desirable.

SUMMARY

Active storage area network discovery method and system are provided. The method in one aspect includes automatically detecting an event occurring in a storage area network, determining one or more devices associated with the event, requesting information about the one or more devices from a plurality of hosts connecting to the storage area network by automatically broadcasting to the plurality of hosts, receiving the information, and updating one or more properties associates with the storage area network with the information.

The system in one aspect includes an event module operable to capture events occurring on a storage area network switch. A policy module is operable to automatically invoke one or more discovery functions based on one or more events captured by the event module. A discovery module comprising at least the one or more discovery function, is operable to discover current status of the storage area network switch.

Further features as well as the structure and operation of various embodiments are described in detail below with reference to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a flow diagram illustrating a method of the present disclosure in one embodiment.

Fig. 2 is a block diagram illustrating components of the system of the present disclosure in one embodiment.

Fig. 3 is a flow diagram illustrating the DiscoverDevice function in one embodiment.

Fig. 4 is a flow diagram illustrating the Receive_Thread function in one embodiment.

DETAILED DESCRIPTION

Active SAN discovery of the present disclosure in one embodiment allows the user to define policies regarding intended connectivity. Whenever a device is connected to or disconnected from the SAN, an event is spawned and policy checks are triggered. If the connectivity is not what was originally intended, for instance, from checking a preset policy, the user has the option to deny the device access.

In one embodiment, the system and method of the present disclosure monitors the configuration changes and preserves information related to those configuration changes.

In one aspect, the system and method of the present disclosure creates new device discovery events, which a user can automate for further setup of a new device, for example, by tying it into a SAN manager's event correlation system and launch, for example, a disk array setup wizard. If the device is a host, access rights are checked by policies, for instance, to provide security by preventing unauthorized access to data.

Another aspect of the system and method of the present disclosure creates a discovery change log to allow the user to report on all configuration changes and check for errors. In one embodiment, the events have the time stamps of when the changes actually happened rather than the timestamp when a scheduled discovery finds the change.

In one embodiment, the system and method described in this application enables the updating of SAN resource information without the need for manually initiating or scheduling a discovery. In one embodiment, the system and method is an ongoing event driven process that responds automatically to changes in the real life SAN environment. It is defined through a set of events generated by agents or agent policies and specific discovery actions.

Any event that has been generated will trigger one or more corresponding discovery functions that will discover or rediscover the parts that were affected by the event and will populate the CA Common Services CORE (Worldview repository) with the discovered or rediscovered objects. Thus, up-to-date view of the SAN topology is made available. In another aspect, an audit log that includes recorded changes may be kept.

Examples of changes in SAN that may occur include a new device being connected to a SAN switch, or a switch becoming the new principal switch in a fabric. In these cases, the policy that generated the event will intelligently select the discovery function. The SAN discovery process communicates its requests for new information by broadcasting to host agents that are located on SAN attached hosts to see what information changed in-band.

For instance, Fig. 1 shows a flow diagram illustrating a method of the present disclosure in one embodiment. At 102, when a new host is joined to a SAN by connecting the host's HBA (host bus adapter) port to a switch port on a SAN switch, a policy in the system and method of the present disclosure at 104, for example, the health policy, picks this event up as a name server change in the SAN switch and automatically launches active discovery procedure of the system and method of the present disclosure at 106. Active discovery then uses the broadcast mechanism to locate the new host and collect information about it at 108. The information sent by the agents, together with the discovery information residing in the switch, are used to create a new host object in the Worldview repository at 110, and to link the host with the corresponding switch in the SAN topology view at 112. The change is recorded in the active discovery change log.

For instance, events may be triggered as a result of the following occurrences in the SAN: HBA added or removed from SAN attached host; device bus rescan on SAN attached host; fabric split or fabric merge; new principal switch in fabric; new host joined (connected to) or disconnected from fabric; new disk array joined (connected to) or disconnected from fabric; new tape library joined (connected to), disconnected from fabric; WWN (world wide name) change on switch port or devices were switched; offline device went online or online device went offline; etc.

Fig. 2 is a block diagram illustrating components of the system of the present disclosure in one embodiment. The events module 204 captures events generated from the SAN switch 202 and automatically, for example as software-driven and controlled, invokes appropriate actions to take place. For example, if a user disconnects a device from a port on the SAN switch 202, the system of the present disclosure automatically removes the device from the Worldview view. Similarly, if the user reconnects the port to a device, the device and link is automatically added to the Worldview view.

When the SAN switch 202 sends a trap SNMP Administrator (aws_sadmin) receives this SNMP request and the SNMP gateway is responsible for the managing SNMP requests. SNMP refers to simple network management protocol that governs network management and the monitoring of network devices and their functions. The message is then put on the Distributed State Bus where DSM (distributed storage matrix) can now manage it. For instance, DSM may change trap data reply due to polling, and user input into object state changes, for example, by using the Finite State Machine (FSM) Logic.

In one embodiment, three event policy functions may be launched after discovering name server changes on the switch. These functions may create the events listed above after analyzing the new configuration.

ABASIC_DiscoverSwitchPort is invoked whenever a user needs to discover a port because an event is received which shows that a port is online and is now connected to a host or a device. Another function, DiscoverSwitchPortByWWN available from SANDISC.DLL, may be called within the ABASIC_DiscoverSwitchPort to make host/device linked to the port. DiscoverSwitchPortByWWN is called with the following parameters: Repository, <User name>, <Password>, SwitchName, SwitchClass, PortWWN, <LogFile>, LOG_LEVEL_DEBUG. NULL is passed for User Name, Password and LogFile. The SANDISC.DLL handles these parameters.

ABASIC_UnDiscoverSwitchPort is called after a disconnect event has been detected and the user acknowledged the change. This means the device is now considered to be offline and more granular discovery actions may have to be performed based on the previous connectivity of the switch. ABASIC_UnDiscoverSwitchPort may be a wrapper function that calls UnDiscoverSwitchPortByWWN available from SANDISC.DLL to further handle the particular undiscovery scenario. In case of a host or a storage device, the device may be moved into an offline device folder in case it goes online again. If the connected device was another switch, this is a fabric split event and may need to be handled accordingly.

ABASIC_DiscoverFabricDomainIDChange function may be launched for Domain ID changes. This means that another switch has taken over the role of the principal switch in the fabric even though there were no connectivity changes. In turn, all fabric related properties may be updated.

The system of the present disclosure may include the following functions for the discovery of devices: DiscoverPort, UnDiscoverPort, DiscoverSwitch, DiscoverFabric, and FreeSandiscReturn. DiscoverPort function retrieves information about the port and what is connected to it. It also updates the Worldview repository with the latest information. The function first signs on to the Worldview repository and switch information is retrieved. Next, the specified port is discovered using SNMP. The follow up discovery action may be classified depending on the connectivity information stored in the repository. This is done by searching the repository for a matching WWN. The DiscoverDevice function is called to search for a remote WWN. If a match is not found, the device is created using proxy-less discovery. Proxy-less discovery uses information from the switch name-server table to create the device.

After the device is created, the policy information for the switch port is checked to make sure that if a device is reserved for the port, it matches the device that was created. If the reserved and actual devices do not match, a policy error is sent to the event console. Next, the device object is created in the Worldview repository and the switch port properties are updated. Finally, the device and switch are linked in the Worldview repository. Additional discovery functions that are launched from this particular function depending on proxy-less discovery methods are: DiscoverNewSwitch->MergeFabric, DiscoverFabric, DiscoverDiskSubsystem, DiscoverTapeSubsystem, DiscoverHost, and DiscoverNewHBA.

UnDiscoverPort function retrieves information about a switch port and removes the link from the port to a connected device. The device is moved to an offline device folder, depending on the type of the device, which was connected. This function spawns the following sub-functions: UndiscoverHost, UnDiscoverSwitch (switch still online)->SplitFabric, UnDiscoverSwitch (switch no longer online)->UnDiscoverSubFabric, UnDiscoverDiskSubsystem, UnDiscoverTapeSubsytem, RemoveHBAFromHost.

DiscoverSwitch function retrieves information about a switch and creates the fabric and topology links between the switch and other SAN devices. This function first signs on to the Worldview repository and switch information is retrieved. The latest switch information is discovered using SNMP. Next, this switch information is used to create the switches and ports in the Worldview repository. Finally, the DiscoverPort function is called for each port. DiscoverFabric function updates the fabric topology with the latest member and link information. This function first determines which devices are members of a fabric by signing on to the WorldView repository and searching for the fabric and devices. It discovers information about the switches in a fabric using SNMP to determine the current fabric membership. The fabric is created if it does not exist in the Worldview repository. Finally, devices are added and removed from the fabric so that it matches up with the discovered information. FreeSandiscReturn function frees the memory allocated for return codes.

SANproxy:DiscoverDevice function is used to dynamically discover changes in the visibility of connections on SAN attached hosts. It uses a broadcast mechanism to find out what devices can be seen from a host. Zone changes may have made new devices visible to a host that previously were not. A message is sent, for instance, using UDP (user datagram protocol) sockets to a list of IP (internet protocol) addresses inquiring if any host has knowledge of the Device IDs (identifiers) in question. This broadcast message is recognized by a proxy agent (sanproxy). The requestor can inquire about a Node Device ID, a Port Device ID or both. If the SAN if FibreChannel, the Device ID may be in the form of a WWN(World Wide Name), that is, a Port WWN or Node WWN.

The hosts that receive the inquiry message and have an Active Discovery agent installed on it will respond, for instance, for instance, using UDP sockets, to the requestor if they have information about the Device IDs. No response is sent if the host does not have information about the Device IDs. The information received from all hosts responding within a given time period is collected and presented to the caller of this function.

SANproxy:NotifyBusRescan function is launched if sanproxy was restarted or a device bus rescan occurred on a SAN attached host. Active discovery will be launched to track all changes that occurred in visibility of attached devices.

Fig. 3 is a flow diagram illustrating the DiscoverDevice function in detail in one embodiment. At 302, request packet is built, for instance, a UDP packet inquiring about devices. At 304, port number to use is determined. At 306, memory buffer is allocated to receive data. At 308, Receive_Thread function is called. This function will be described with reference to Fig. 4. At 310, list of IP addresses is looped through. At 312, if the entry is subnet entry, IP addresses are generated from 1 to 254 at 316. At 318, request packet is sent to IP address. Step 318 is repeated until the last address of subnet is processed at 320. At 312, if the entry is not a subnet entry, the request packet is sent to IP address and the method proceeds to 322.

At 322, if the last entry in the list is processed, at 324, the method waits for a predetermined period of time. At 326, socket connections are shut down. At 328, received data from stored buffer is copied into user buffer.

Fig. 4 is a flow diagram illustrating the Receive_Thread function in detail in one embodiment. At 402, socket connection is set up. At 404, if the connection is not active, the function exits at 406. At 408, the process waits for one or more messages. At 410, connection is checked again. At 412, message is received into local buffer. At 414, a check is made to determine whether enough space is left in stored buffer. If not, at 416, buffer is reallocated to have larger size. At 418, data received is converted from big endian to native endian, if applicable. At 420, connection is ended.

The system and method of the present disclosure may be implemented and run on a general-purpose computer. The embodiments described above are illustrative examples and it should not be construed that the present invention is limited to these particular embodiments. Thus, various changes and modifications may be effected by one skilled in the art without departing from the spirit or scope of the invention as defined in the appended claims.

We claim:

1. An active storage area network discovery method, comprising:

automatically detecting an event occurring in a storage area network;

determining one or more devices associated with the event;

requesting information about the one or more devices from a plurality of hosts connecting to the storage area network by automatically broadcasting to the plurality of hosts;

receiving the information; and

updating one or more properties associates with the storage area network with the information.

2. The method of claim 1, further including creating a discovery change log associated with the event.

3. The method of claim 1, wherein the event includes at least time when a change associated with the event actually occurred.

4. The method of claim 1, wherein the event automatically triggers the determining and requesting step.

5. The method of claim 1, further including creating an audit log that includes history of recorded changes.

6. The method of claim 1, wherein the event includes device changes in the storage area network.

7. The method of claim 1, wherein the event includes occurrence of at least one of host bus adapter added, host bus adapter removed, device bus rescan, fabric split, fabric merge, a new host connected to fabric, a host disconnected from fabric, a new disk array connected to fabric, a disk array disconnected from fabric, a new tape library connected to fabric, a tape library disconnected from fabric, world wide name change on switch port, a device switch, online device went offline, and offline device went online.

8. An active storage area network discovery system, comprising:

an event module operable to capture events occurring on a storage area network switch;

a policy module operable to automatically invoke one or more discovery functions based on one or more events captured by the event module; and

a discovery module comprising at least the one or more discovery function, operable to discover current status of the storage area network switch.

9. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform a method, comprising:

automatically detecting an event occurring in a storage area network;

determining one or more devices associated with the event;

requesting information about the one or more devices from a plurality of hosts connecting to the storage area network by automatically broadcasting to the plurality of hosts;

receiving the information; and

updating one or more properties associates with the storage area network with the information.

10. The storage device of claim 9, further including creating a discovery change log associated with the event.

11. The storage device of claim 9, wherein the event includes at least time when a change associated with the event actually occurred.

12. The storage device of claim 9, wherein the event automatically triggers the determining and requesting step.

13. The storage device of claim 9, further including creating an audit log that includes history of recorded changes.

14. The storage device of claim 9, wherein the event includes device changes in the storage area network.

15. The storage device of claim 9, wherein the event includes occurrence of at least one of host bus adapter added, host bus adapter removed, device bus rescan, fabric split, fabric merge, a new host connected to fabric, a host disconnected from fabric, a new disk array connected to fabric, a disk array disconnected from fabric, a new tape library connected to fabric, a tape library disconnected from fabric, world wide name change on switch port, a device switch, online device went offline, and offline device went online.

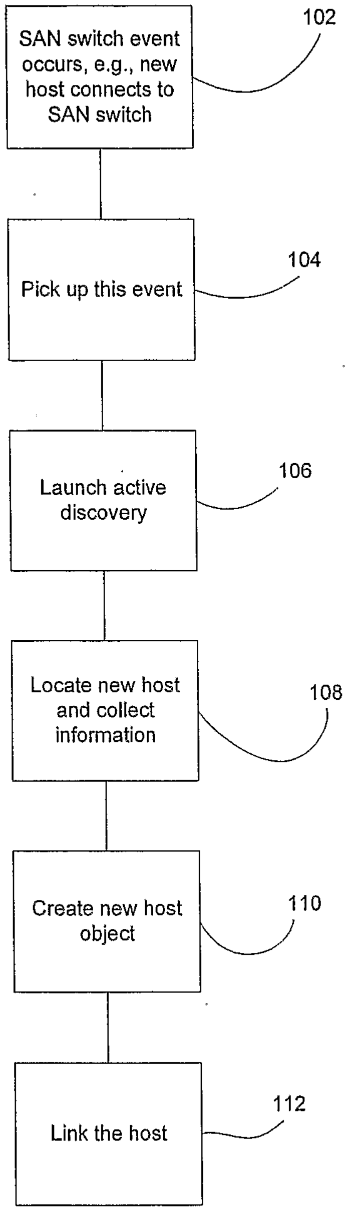

SAN switch event occurs, e.g., new host connects to SAN switch
102
Pick up this event
104
Launch active discovery
106
Locate new host and collect information
108
Create new host object
110
Link the host
112

Fig. 1

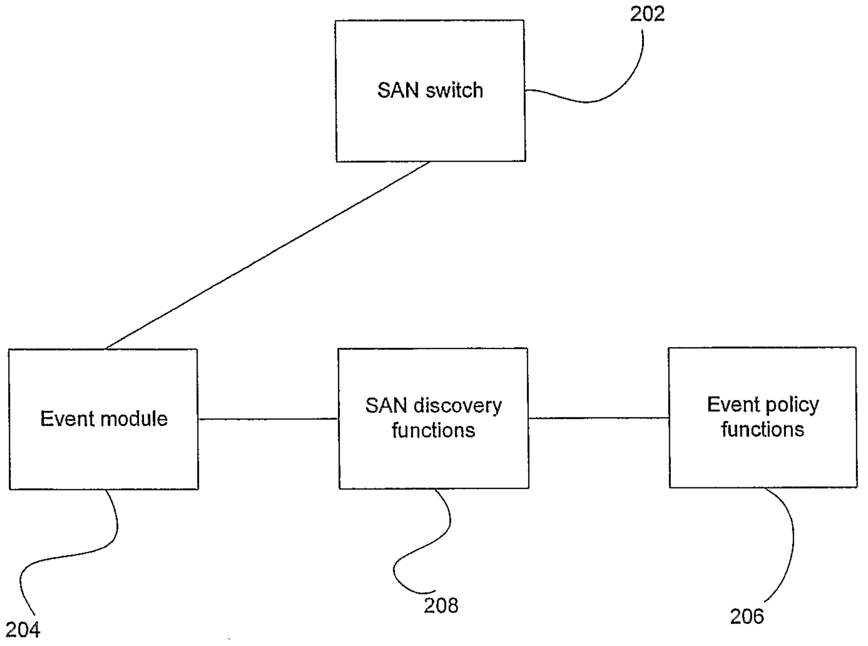

202
SAN switch
Event module
SAN discovery functions
Event policy functions
204
208
206

Fig. 2

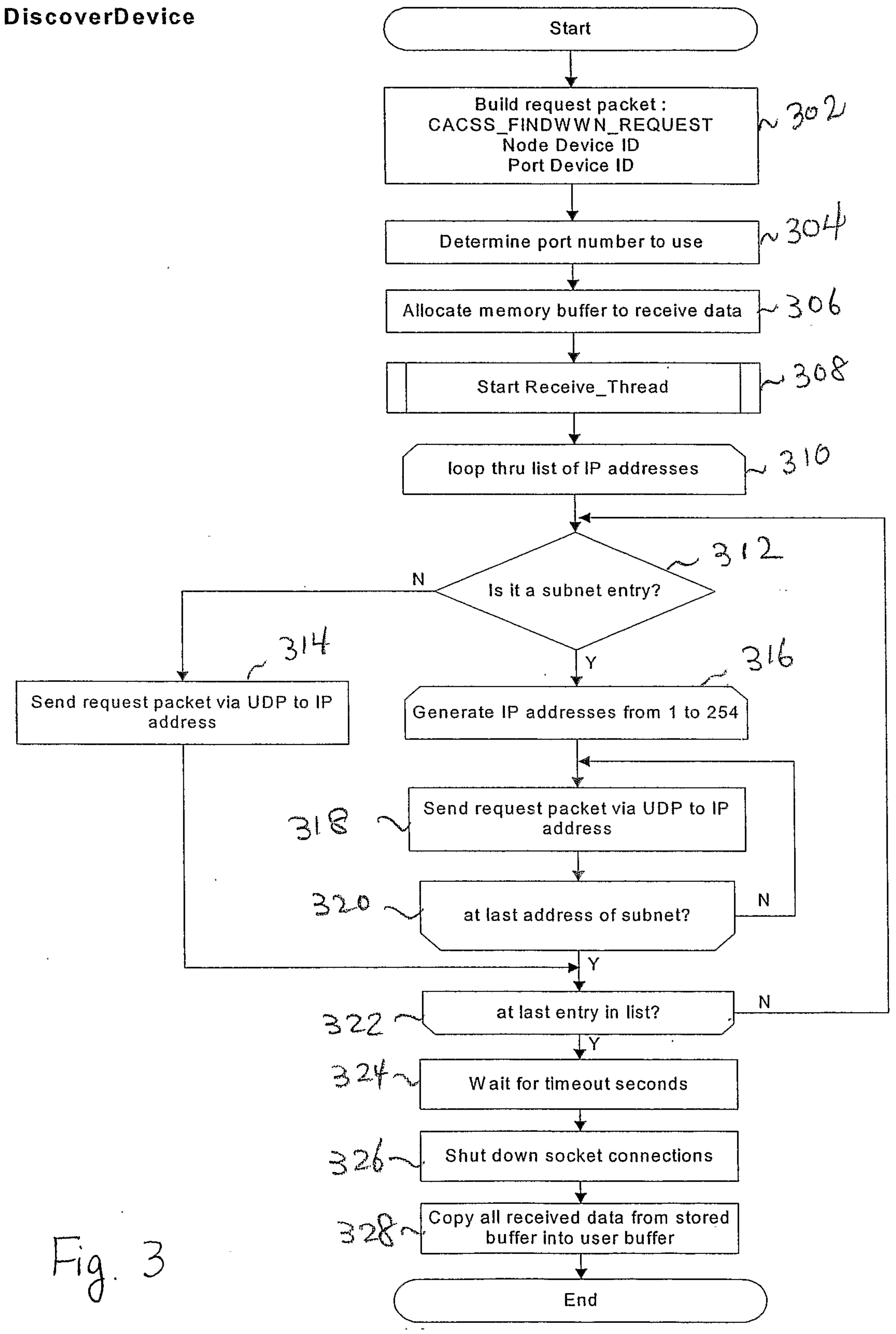

DiscoverDevice
Start
Build request packet : CACSS_FINDWWN_REQUEST Node Device ID Port Device ID
302
Determine port number to use
304
Allocate memory buffer to receive data
306
Start Receive_Thread
308
loop thru list of IP addresses
310
Is it a subnet entry?
312
N
Y
Send request packet via UDP to IP address
314
Generate IP addresses from 1 to 254
316
Send request packet via UDP to IP address
318
at last address of subnet?
320
N
Y
at last entry in list?
322
N
Y
Wait for timeout seconds
324
Shut down socket connections
326
Copy all received data from stored buffer into user buffer
328
End

Fig. 3

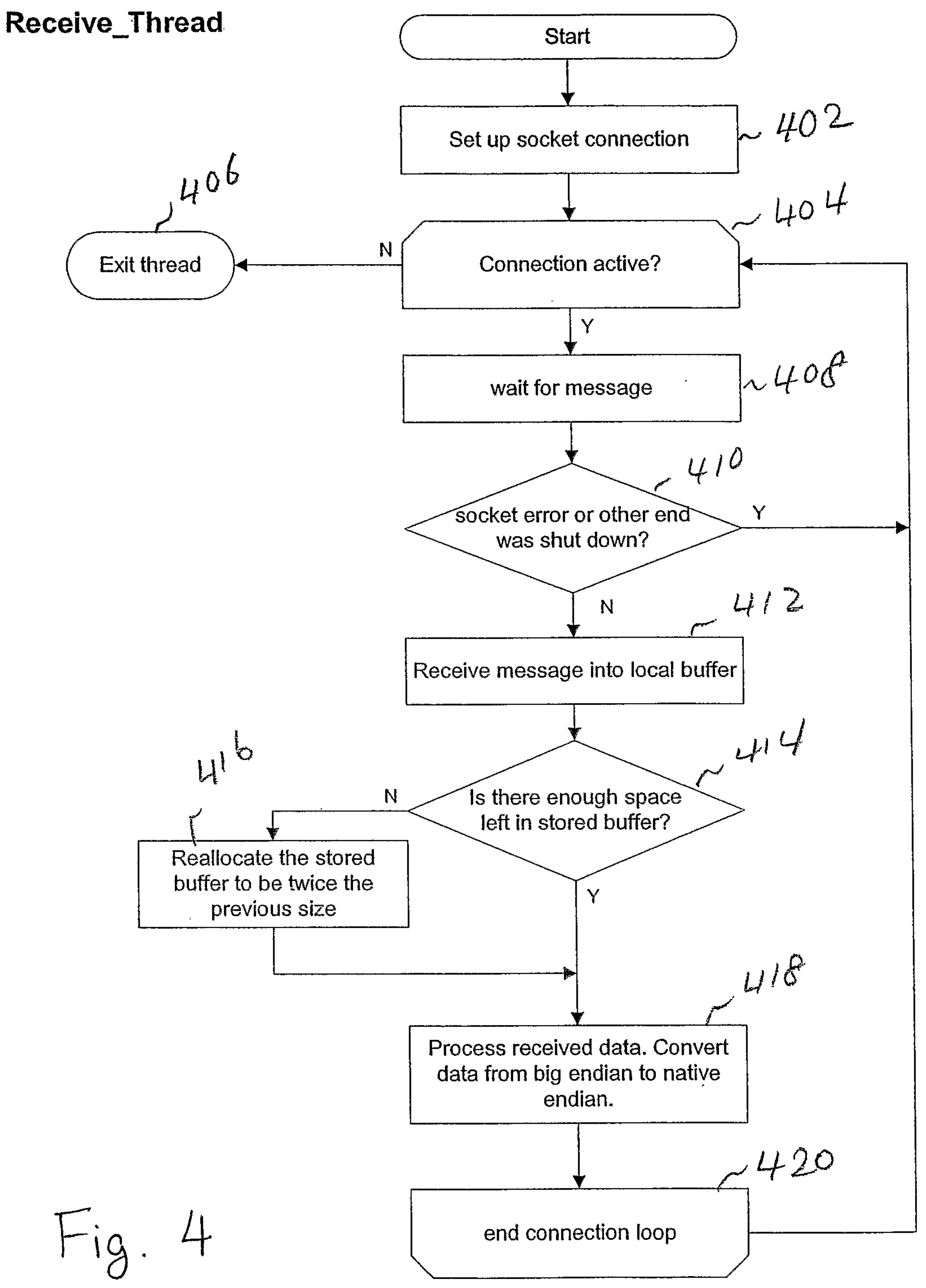

Receive_Thread
Start
Set up socket connection
402
Connection active?
404
N
Exit thread
406
Y
wait for message
408
socket error or other end was shut down?
410
Y
N
Receive message into local buffer
412
Is there enough space left in stored buffer?
414
N
Reallocate the stored buffer to be twice the previous size
416
Y
Process received data. Convert data from big endian to native endian.
418
end connection loop
420

Fig. 4