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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 June 2006 (29.06.2006)

PCT

(10) International Publication Number
WO 2006/069279 A2

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2005/046694

(22) International Filing Date:
20 December 2005 (20.12.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/021,568 22 December 2004 (22.12.2004) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **ELDAR, Avigdor** [IL/IL]; 15 Steinberg St., Jerusalem (IL).

(74) Agents: **MCCRACKIN, Ann, M.** et al.; Schwegman, Lundberg, Woessner & Kluth, P.A., P.O. Box 2938, Minneapolis, MN 55402 (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, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, 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, 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, IS, IT, LT, LU, LV, 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.

(54) Title: ROUTING OF MESSAGES

(57) Abstract: A method to provide transfer of data without the use of a network from an application program to an embedded device. A routing service establishes a communication channel without the use of a local area network. The routing service then manages the control of this communication channel for the transfer of data and closes the channel in an appropriate period.

WO 2006/069279 A2

ROUTING OF MESSAGES

Technical Field

In general, the present subject pertains to communication systems and more particularly to application software control of an embedded unit.

Background

Typically, third-party application software communicates through a local area network to an embedded processor for managing an embedded device controlled by the embedded processor. This application software will communicate through the local area network, utilizing standard protocol such as: HTTP (hyper text transport protocol), SOAP (standard protocol enabling remote procedure calls) messages, and TLS/SSL (transport layer security/secure socket layer).

Further, some of the services provided by application software may execute on the same computational platform for which manageability of the embedded device is sought. For example, a management console may gather inventory information from all computational platforms in an organization. This inventory information may be gathered from the embedded manageability devices residing on an organization's computational platforms. As an additional example, the information may be located in some type of non-volatile memory. The application software may need to gather such information located within the control of the embedded device to provide, for example, information sharing between local and remote agents.

Simply returning information to the application software through a local device using normal network stacks will result in the information being forwarded incorrectly. Either the TCP/IP (transmission control protocol/internet protocol) stack may not forward the information to a network adapter or the network will not return data packets back on the same physical link from which they came.

Description of the Drawings

FIG. 1 is a block diagram of a communication arrangement in accordance with an embodiment in the present invention.

FIG. 2 is a block diagram depicting various protocol layers associated with an embodiment of the present invention.

FIG. 3 is a flow chart of a communication arrangement in accordance with an embodiment of the present invention.

FIG. 4 is a flow chart of a connect request in accordance with an embodiment of the present invention.

FIG. 5 is a flow chart of a read request in accordance with an embodiment of the present invention.

FIG. 6 is a flow chart of a close request in accordance with an embodiment of the present invention.

Detailed Description

FIG. 1 is a block diagram of a communication arrangement directly between operating system 20 of processor 15 and embedded device 30 without using a local area network in accordance with an embodiment of the present invention. This communication arrangement can enable an application program 22 to provide out-of-band (OOB) management of the communication arrangement. OOB indicates that a system may be an operating system agnostic. Typically, application programs communicate through a network utilizing standard protocols. As an example of such an arrangement, consider a management console of computational platform 10 gathering inventory information from various computational platforms in an organization. This information may need to be gathered from embedded manageability devices (commonly referred to as embedded devices). However, simply sending the information to the embedded device through a normal network stack will result in information being incorrectly forwarded.

Operating system 20 may be associated with the application program 22. Application program 22 is coupled to the operating system (OS) network stack 24. Although application program 22 is shown within the OS 20, program 22 may be separate from the OS 20. Typical flow of messages through operating system 20 is from OS network stack 24 to network LAN controller 28 and via routing 29 to a local area network (not shown).

Incoming messages for embedded device 30 are received from the local area network (not shown) through routing 42 to a network LAN controller 44. Upon the initialization of operating system 20, routing service 25 is bound to OS network stack 24 via a particular destination port.

When application program 22 uses the particular destination port number, which is an internet assigned number authority (IANA) transmission control protocol/internet protocol (TCP/IP) port number, routing service 25 binds to this particular port number. Once this binding is completed, any messages sent to the OS network stack 24 with destination IP address assigned to OS 20 and with the particular destination port number would be routed by OS network stack 24 to routing service 25.

When application program 22 communicates with an embedded device 30, it sends a message using standard network protocols to OS network stack 24. OS network stack 24 may check the destination IP address and TCP destination port associated with the message. OS network stack 24 then identifies the message as either one which should be routed to an entity or one which should be routed to the routing service 25.

An embedded device in some embodiments may be a device that is placed within a computational platform. The embedded device may include a set of resources independent of the main resources of the computational platform, such as, a processor and a memory resources, for example.

Normally messages would be transmitted from OS network stack 24 through network LAN controller 28 and routed via network routing 29 to the network.

In the present example, OS network stack 24 has determined from the destination IP address and the TCP destination port that routing of the message should be to routing service 25, for example.

Routing service 25 may then send a connect message through bus driver 26 through bus 40 to the embedded device 30. Bus arrangement 40 may include a serial bus, a parallel bus, a PCI bus, a buffer, a switch, a memory of various types, or a number of registers.

Bus 40 then passes the message through bus driver 32 of embedded device 30 through network multiplex 34 to internal application program 35 of embedded device 30. If the message is a connect message, for example, the connect request may be successful.

Once this communication channel becomes established from routing service 25 to internal application program 35, the message originally sent by application program 22 through OS network stack 24 to routing service 25 is transmitted along the same communication channel as previously mentioned to internal application program 35. This communication channel is from routing service 25, through bus driver 26, through bus 40, through bus driver 32, through network multiplexer 34 to internal application program 35.

When all the messages have been sent by routing service 25, or received by routing service 25, routing service 25 will then close one side of the communication channel that was previously established after receiving a close request from application program 35, for example. That is, routing service 25 will tear down the connection between bus driver 26 and bus 40, and between bus 40 and bus driver 32.

In addition, embedded device 30 may receive input from the network via routing 42 through local area network (LAN) controller 44. Such network routed messages are then sent to interface driver 33, through network multiplexer 34 to the internal application program 35. Further, embedded device 30 may include an embedded processor 31 or other devices. The embedded device may provide for execution of the internal application program 35. Embedded processor 31may be implemented on a chip-set, in an embodiment. In other embodiment, embedded processor 31as well as embedded device 30 may be implemented as part of a chip set, on a mother board as a chip, or on a portion of a network interface card (NIC) inserted into a circuit card slot. However, the implementation is not limited to these configurations. A chip is a semiconductor device.

Bus 40 transmits messages between routing service 25 and internal application program 35 and may be implemented on top of a peripheral controller interface such as PCI/PCI-X/PCI-Express, USB, etc. type bus arrangement. Further, store and forward devices such as memories, etc. may provide the communication channel as well.

Operating system 20 may be implemented on a semiconductor device fabricated by various technologies known to those of ordinary skill in the art such as silicon, gallium arsenate, etc. This semiconductor device or "chip" may include a processor 15 having an operating system 20, an application program 22, OS network stack 24, routing service 25 and bus driver 26.

FIG. 2 is a block diagram which maps various network protocol layers which may be used to communicate with software which handles each layer. The various protocols of 50 correspond to the protocol layers associated with operating system 20. The various protocols of 60 correspond to the protocols associated with embedded device 30. The highest level protocol within writing system protocols is the application layer 51. On the highest level of protocol stack 60 of the embedded device 30 is capability module (CM) 61 which provides the highest level protocol layer. The application layer communicates and corresponds with the capability module level 61 in the embedded device protocol stack 60.

The next lower layer within the protocol stack 50 of operating system 20 is SOAP/HTTP encapsulation layer 52. This layer corresponds and communicates with SOAP/HTTP server 62 in the embedded device protocol stack 60.

The next lower layer in the operating system stack 50 is the transport layer security (TLS) layer 53. The transport layer security 53 communicates with and corresponds to transport layer security (TLS) 63 within the embedded device protocol stack 60.

The next lower layer of the protocol stack 50 of the operating system is a BSD socket layer 54. The BSD socket layer 54 communicates with and corresponds to local socket provider 64 through routing service protocol 55. Routing service protocol 55 communicates with BSD socket layer 54. Routing service protocol 55 communicates with the local socket provider protocol 64 of the embedded device protocol stack 60.

Lastly is shown the transmission control protocol/internet protocol (TCP/IP) or user datagram protocol/internet protocol 57. Device driver protocol 58 communicates between device driver 65 of the protocol stack 60 of embedded device 30 and the transmission control protocol/internet protocol (TCP/IP) or user datagram protocol/internet protocol 57.

As a result of the above-described processing by routing service 25, it can be seen that a data transfer is shown excluding the use of a local area network.

In FIG. 3, a flow chart of an embodiment of the method for routing is shown. This method is performed by routing service 25. The implementation for the processing described herein may take the form of machine-readable software code. This machine-readable software code might be implemented in read-only memory, any non-volatile random access memory, flash memory, electronically programmable read-only memory, etc.

The method for routing of data is started and block 70 is entered. First, the routing service 25 initializes communication with device or bus driver 26, block 70. Next, routing service 25 binds to the port associated with the TCP destination, block 72. The routing service 25 is typically bound to a particular designated TCP port when the computational platform 10 is initially started or is restarted. This IANA port couples OS network stack 24 to routing service 25.

Next, routing service 25 initializes an application socket pair, block 74. That is, routing service 25 establishes the BSD socket layer from 54 to 55 and a pseudo-socket

from 55 to 64. Then routing service 25 awaits the socket requests coming from the socket layer 54 of operating system 20 and the local socket provider 64 of embedded device 30.

When these socket requests are received by routing service 25, the appropriate connection between bus drivers 26 and 32 have been selected and established.

The service 55 then waits for responses from the socket requests at block 76. When a single socket request has been received, block 76 transfers control to block 78. Block 78 determines whether the request may be a connect request to establish the communication channel from routing service 25 through bus driver 26 through bus arrangement 40, through bus driver 32 and network multiplexer 34 to internal application program 35. If the request is a connect request, block 78 transfers control to block 80 via the YES path. Block 80, which will be explained in detail later, handles the connect request from routing service 25 and host processor 15 and then transfers control, upon completion, to block 76 to wait for subsequent socket requests.

If the request is not a connect request, block 78 transfers control to block 82 via the NO path. Block 82 determines whether the request may be a read request for exchanging data between internal application program 35 and routing service 25. If the request is a read request, block 82 transfers control to block 84 via the YES path. Block 84 handles the read request from either the application program 22 of the operating system (HOST) or the internal application program 35 of the imbedded device (EP). When the read request is completed, block 84 transfers control to block 76 to wait for socket requests.

If the request is determined to be neither a connect or read request, block 82 transfers control to block 86 via the NO path. This method determines that the request should be a close request. That is, the close request is the command to tear down the previously established communication channel from routing service 25 to internal application program 35 or communication channel from application program 22 to routing service 25, depending on the source of this request (HOST or EP). Since this is a close request, block 86 transfers control to block 88. Block 88 handles the close request from either the routing service of the operating system or the internal application program of the imbedded device. Upon completion of the close, block 88 transfers control to block 76 to wait for socket requests. As a result of the above-described processing, it can be seen that a data transfer is shown excluding the use of a local area network.

In FIG. 4, a flow chart of an embodiment of the connect request processing of block 80 is shown. Block 89 diagrammatically depicts the input to the connect request 80

as being the socket identity. Block 90 then determines the socket request is initiated from the application 22. Recall that the application program 22 is typically the one which can request the connection of the communication channel between the operating system and the embedded device, although the embedded 30 may also request. If the socket request is not initiated from the application 22, block 90 transfers control via the NO path to block 93. Block 93 issues a socket request to the application. Then block 93 transfers control to block 94.

If the connect request is from the application 22, block 90 transfers control to block 92 via the YES path. Next, block 92 issues a connect request to the embedded device, resulting in embedded processor socket.

Next, block 94 waits and determines whether the connect request has been successfully completed. If the request does not complete within a predetermined time limit or is returned as being incomplete, block 94 transfers control to block 96. Block 96 indicates a failure of the connect request and ends the process.

When the connect request is successfully complete, block 94 transfers control to block 98 via the YES path. Then a socket pair (SP) is determined: HOST processor 15 and embedded processor 31. Then the socket pair for the application program 22 to internal application 35 connection is stored in a data base by routing service 25. The process is ended.

In FIG. 5, an embodiment of the read request 84 from either the operating system or embedded device is depicted in flow chart form. The input to the read request is the socket identity, block 101.

First, the socket pair structure is found from a data base indicating the host socket, block 102. Next, routing service 25 sets the socket of the embedded device (EP) with the socket pair of the data base, block 104.

Next, block 106 sets a read/write buffer to read from the HOST socket. This assumes that the message is being transferred from the application program 22 to the internal application program 35. If the data transfer happened to be from internal application program 35 to application program 22, the buffer read would be from the embedded device 30.

Next, block 108 determines whether this request to read into the buffer was successful. If the buffer request was successful, block 108 transfers control to block 110 via the YES path. Block 110 then writes the data. This corresponds to sending the data

from the buffer through bus driver 26, through bus arrangement 40, through bus driver 32 and network multiplexer 34 to the internal application program 35 (EP) . If the data transfer is from internal application program 35 to application program 22, the data flow and reading and writing would be performed by the other socket of the socket pair (HOST). If the write is successful, the data is received by internal application program 35, block 112 transfers control and ends the process.

If the attempted write was unsuccessful, block 112 transfers control to block 114 via the NO path. Block 114 first closes the operating system socket, the initiator of the data request. Then block 114 closes the embedded device socket, the receiver of the data transfer request.

Then the socket pair is removed from the data base, block 116. The process is then ended.

Similarly, for an unsuccessful read to the buffer, block 108 transfers control to block 114. Block 114 and block 116 perform as mentioned above. Then the process is ended.

In FIG. 6, an embodiment of the close request 88 is depicted in flow chart form. Block 120 indicates the input being the socket identity. Next, the socket pair structure is found in the data base from the socket identity, block 122. Next, block 124 determines whether the socket identity is a type of embedded device socket (EP). If it is an embedded device, block 124 transfers control to block 128 via the YES path. Block 128 closes the socket pair from the HOST socket. Then the process is ended.

If the socket type is not an embedded device socket, block 124 transfers control to block 126 via the NO path. In block 126, the embedded device closes the socket pair from the embedded device socket (EP). Then the process is ended.

The description and the drawings illustrate specific embodiments of the invention sufficiently to enable those skilled in the art to practice it. Examples merely typify possible variations. Portions and features of some embodiments may be included in or substituted for those of others. The scope of the embodiments of the invention encompasses the full ambit of the claims and all available equivalents. In some embodiments, the present invention relates to communicating by an application program with an embedded device or processor. In other embodiments, the present invention may pertain to communication between an application program 22 and an embedded device 30 without the use of any local area network 28-29, 42-44.

As a result of the above-described processing by routing service 25, it can be seen that a data transfer is shown excluding the use of a local area network. Further, routing service 25 may be implemented on a semiconductor device. This avoids further costs of "add-on" hardware.

Although some embodiments of the invention have been illustrated, and those forms described in detail, it will be readily apparent to those skilled in the art that various modifications may be made therein without departing from the spirit of these embodiments or from the scope of the appended claims.

What is Claimed:

1. A method comprising:

determining to route a message of an application program to an embedded device;

intercepting the message by an operating system; and

sending the message by a routing service to the embedded device after the message is intercepted by the operating system, the routing service excluding use of a local area network.

2. The method of claim 1, wherein sending the message by the routing service to the embedded device includes sending the message via a bus to the embedded device.

3. The method of claim 2, wherein the sending the message via the bus to the embedded device includes sending the message via a bus driver to the bus.

4. The method of claim 2, wherein the sending the message via the bus to the embedded device includes sending the message from the bus via a bus driver to the embedded device.

5. The method of claim 4, wherein the sending the message via the bus driver to the embedded device includes sending the message from the bus driver via a multiplexer to the embedded device.

6. The method of claim 1, wherein sending the message by the routing service to the embedded device includes opening a communication channel between the routing service and the embedded device.

7. The method of claim 6, wherein sending the message by the routing service to the embedded device includes sending data between the routing service and the embedded device when the communication channel is connected.

8. The method of claim 7, wherein sending the message by the routing service to the embedded device includes closing the communication channel when communication is completed.

9. The method of claim 1, wherein intercepting the message to the embedded device includes determining by the operating system that the message indicates a local address.

10. The method of claim 9, wherein determining by the operating system that the message indicates a local address includes examining a destination internet protocol (IP) address.

11. The method of claim 10, wherein examining the destination IP address includes sending the message by the operating system to the routing service.

12. The method of claim 11, wherein sending the message by the operating system to the routing service includes sending the message to a destination port indicated by at least one of: a transmission control protocol/IP (TCP/IP); and a user datagram protocol/IP.

13. The method of claim 9, wherein the determining by the operating system further includes binding the destination port to the routing service.

14. A router comprising:

an embedded device to receive a message from an application program;

a routing service;

an operating system to route the message to the routing service; and

the routing service to send the message to the embedded device without use of a local area network.

15. The router as claimed in claim 14, including a bus coupled to the routing service and to the embedded device, the local bus to send the message from the routing service to the embedded device.

16. The router as claimed in claim 15, further including a bus driver coupled between the bus and the routing service.

17. The router as claimed in claim 15, the embedded device including a bus driver coupled to the bus.

18. The router as claimed in claim 17, the embedded device further including a multiplexer coupled to the bus driver.

19. The router as claimed in claim 17, the embedded device further including an internal application program coupled to the multiplexer.

20. A machine-readable medium that provides instructions, which when executed by one or more processors, cause the processors to perform operations comprising:

determining to route a message of an application program to an embedded device;

intercepting the message by an operating system; and

sending the message by a routing service to the embedded device after the message is intercepted by the operating system, the routing service excluding use of a local area network.

21. The machine-readable medium of claim 20, wherein the sending the message by the routing service to the embedded device includes sending the message via a bus to the embedded device.

22. The machine-readable medium of claim 20, wherein sending the message to the embedded device includes opening a communication channel between the routing service and the embedded device.

23. The machine-readable medium of claim 22, wherein sending the message to the embedded device includes sending data between the routing service and the embedded device when the communication channel is connected.

24. The machine-readable medium of claim 23, wherein sending the message to the embedded device includes closing the communication channel when communication is completed.

25. A system comprising:

an application program to send a message to an embedded processor;

a routing service;

an operating system to route the message to the routing service;

the routing service to send the message to the embedded processor excluding use of a local area network; and

the embedded processor being a portion of a network interface card, the network interface card being inserted into a circuit card slot.

26. The system as claimed in claim 25, wherein the parallel bus is a local bus coupled to the routing service and to the embedded device.

27. The system as claimed in claim 26, including:

a first bus driver coupled between the local bus and the routing service;

the embedded device includes a second bus driver coupled to the local bus;

the embedded device further includes a multiplexer coupled to the second bus driver; and

the embedded device further includes an internal application program coupled to the multiplexer.

28. A semiconductor device comprising:

a processor having:

an operating system to determine to route a message to an embedded device;

a router to receive the message from the operating system, if the determination is made; and

the router to send the message to the embedded device, the router excluding use of a local area network.

29. The semiconductor device as claimed in claim 28, the processor further including an application program to generate the message, the application program being coupled to the operating system.

30. The semiconductor device as claimed in claim 28, the processor further including a bus driver to route the message to the embedded device, the bus driver being coupled to the routing service.

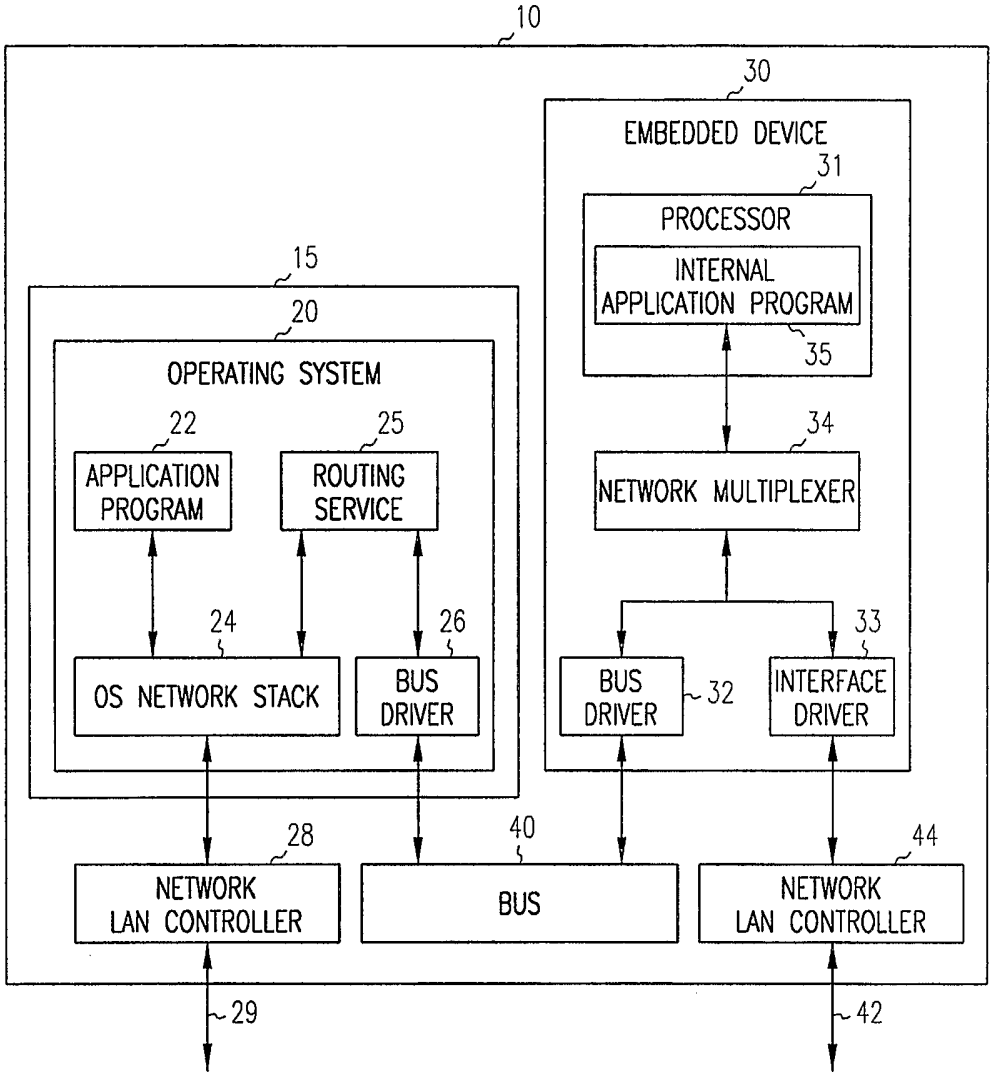

10
30
EMBEDDED DEVICE
31
PROCESSOR
INTERNAL APPLICATION PROGRAM
35
15
20
OPERATING SYSTEM
22
25
APPLICATION PROGRAM
ROUTING SERVICE
34
NETWORK MULTIPLEXER
24
26
33
OS NETWORK STACK
BUS DRIVER
BUS DRIVER
32
INTERFACE DRIVER
28
40
44
NETWORK LAN CONTROLLER
BUS
NETWORK LAN CONTROLLER
29
42

FIG. 1

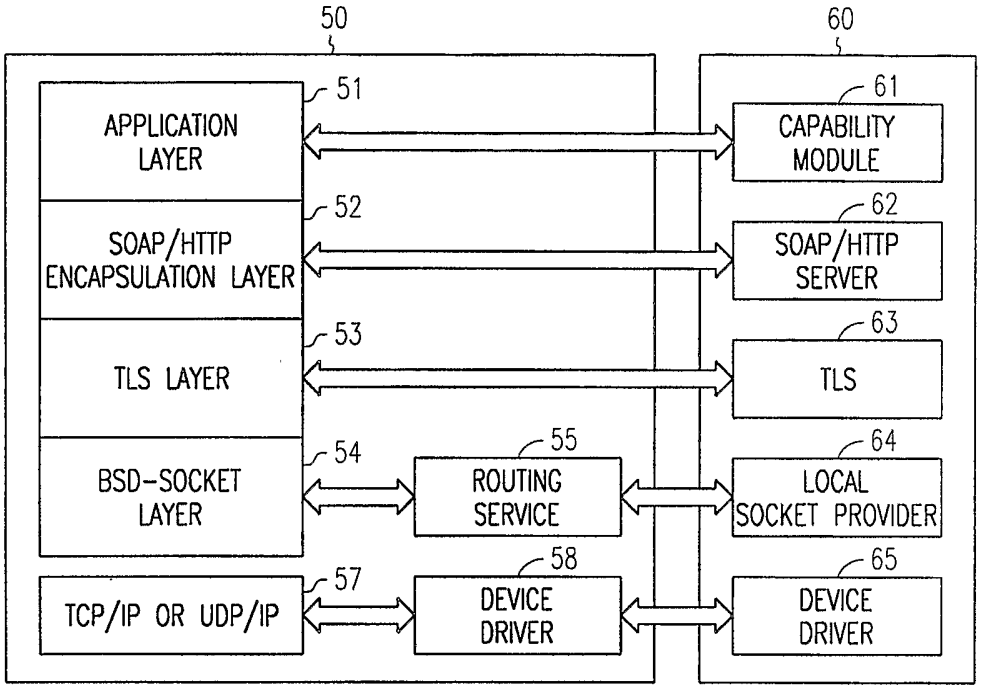

50
60
APPLICATION LAYER
51
CAPABILITY MODULE
61
SOAP/HTTP ENCAPSULATION LAYER
52
SOAP/HTTP SERVER
62
TLS LAYER
53
TLS
63
BSD-SOCKET LAYER
54
ROUTING SERVICE
55
LOCAL SOCKET PROVIDER
64
TCP/IP OR UDP/IP
57
DEVICE DRIVER
58
DEVICE DRIVER
65

FIG. 2

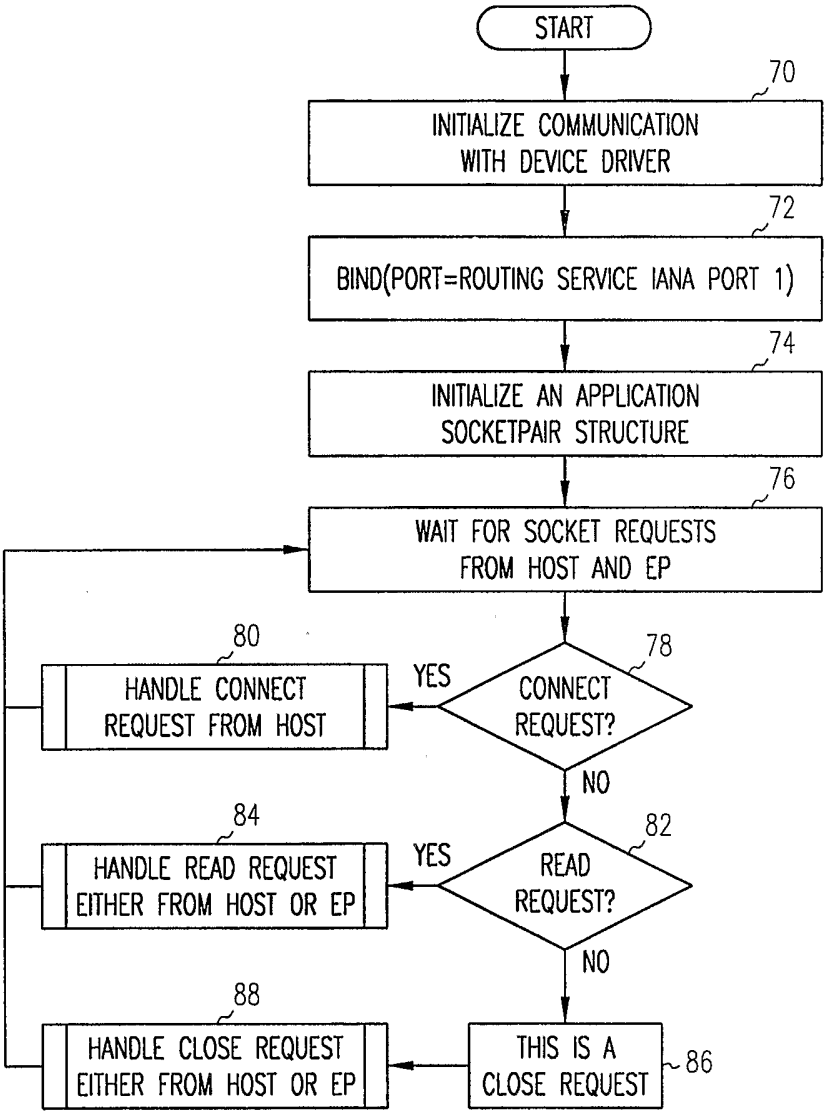

START
70
INITIALIZE COMMUNICATION WITH DEVICE DRIVER
72
BIND(PORT=ROUTING SERVICE IANA PORT 1)
74
INITIALIZE AN APPLICATION SOCKETPAIR STRUCTURE
76
WAIT FOR SOCKET REQUESTS FROM HOST AND EP
78
CONNECT REQUEST?
YES
80
HANDLE CONNECT REQUEST FROM HOST
NO
82
READ REQUEST?
YES
84
HANDLE READ REQUEST EITHER FROM HOST OR EP
NO
86
THIS IS A CLOSE REQUEST
88
HANDLE CLOSE REQUEST EITHER FROM HOST OR EP

FIG. 3

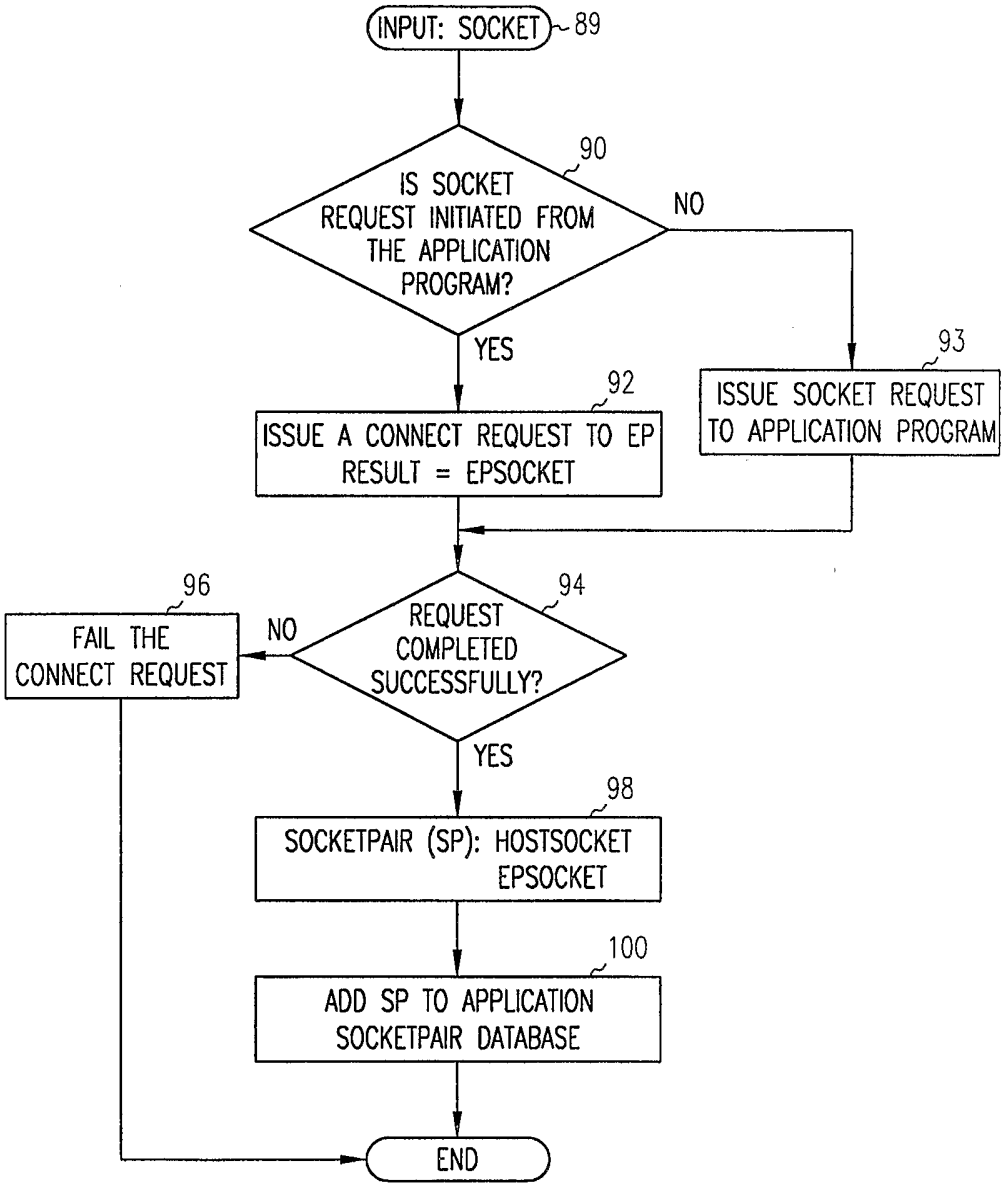

INPUT: SOCKET
89
IS SOCKET REQUEST INITIATED FROM THE APPLICATION PROGRAM?
90
NO
YES
ISSUE A CONNECT REQUEST TO EP RESULT = EPSOCKET
92
ISSUE SOCKET REQUEST TO APPLICATION PROGRAM
93
REQUEST COMPLETED SUCCESSFULLY?
94
NO
FAIL THE CONNECT REQUEST
96
YES
SOCKETPAIR (SP): HOSTSOCKET EPSOCKET
98
ADD SP TO APPLICATION SOCKETPAIR DATABASE
100
END

FIG. 4

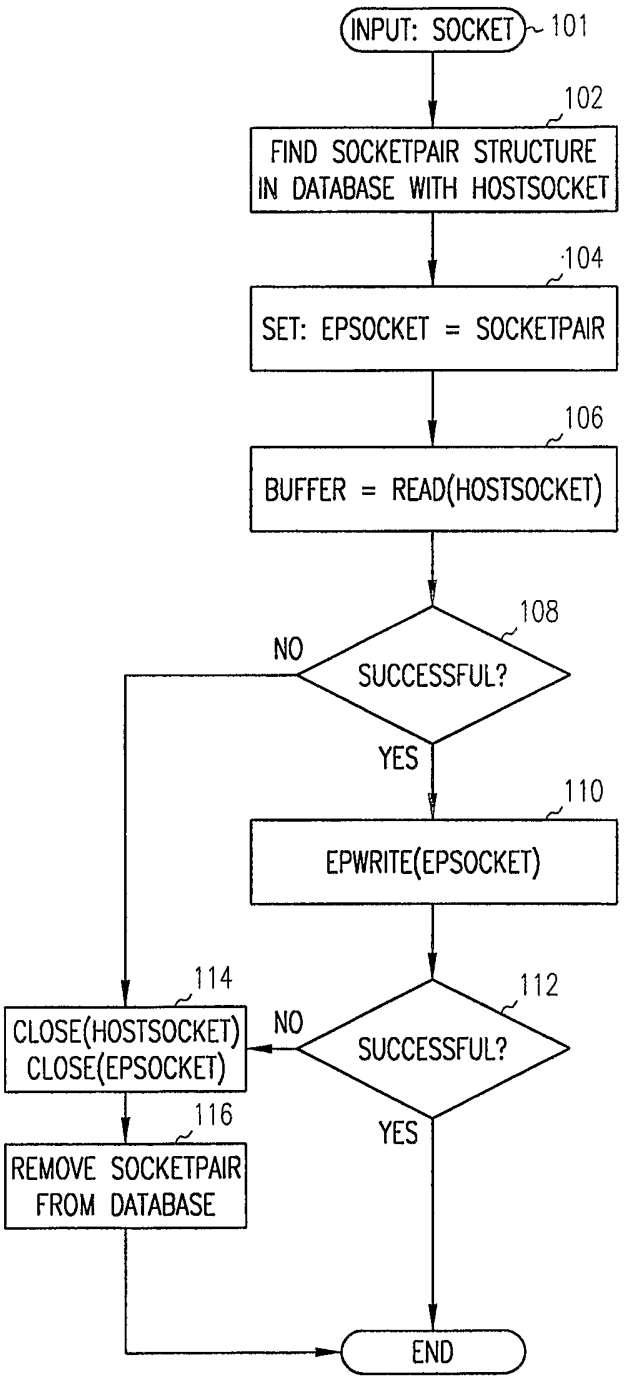

INPUT: SOCKET
101
102
FIND SOCKETPAIR STRUCTURE IN DATABASE WITH HOSTSOCKET
104
SET: EPSOCKET = SOCKETPAIR
106
BUFFER = READ(HOSTSOCKET)
108
NO
SUCCESSFUL?
YES
110
EPWRITE(EPSOCKET)
112
NO
SUCCESSFUL?
YES
114
CLOSE(HOSTSOCKET)
CLOSE(EPSOCKET)
116
REMOVE SOCKETPAIR FROM DATABASE
END

FIG. 5

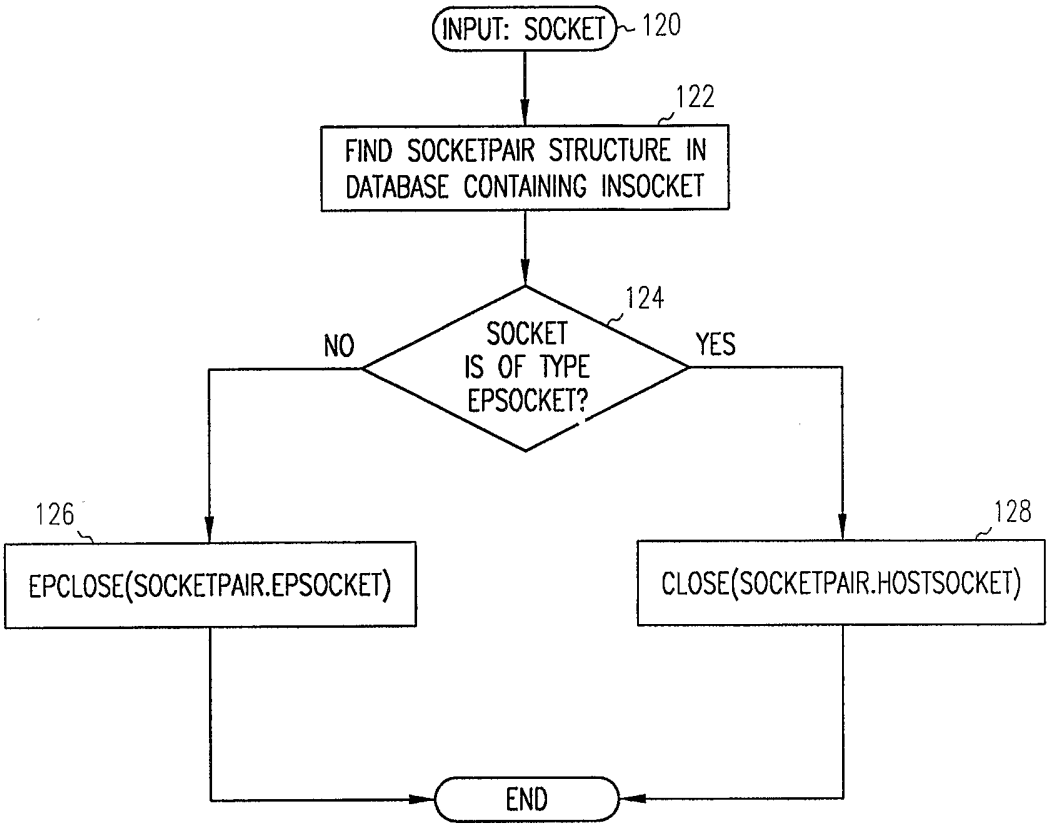

INPUT: SOCKET
120
122
FIND SOCKETPAIR STRUCTURE IN DATABASE CONTAINING INSOCKET
124
SOCKET IS OF TYPE EPSOCKET?
NO
YES
126
EPCLOSE(SOCKETPAIR.EPSOCKET)
128
CLOSE(SOCKETPAIR.HOSTSOCKET)
END

FIG. 6