(12) UK Patent Application (19) GB (11) 2 354 349 (13) A

(43) Date of A Publication 21.03.2001

(21) Application No 9921776.2

(22) Date of Filing 16.09.1999

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Peter A Lambros
Stephen James Paul Todd

(74) Agent and/or Address for Service
IBM United Kingdom Limited
Intellectual Property Department, Mail Point 110,
Hursley Park, WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(51) INT CL[7]
G06F 9/46

(52) UK CL (Edition S)
G4A AUDB

(56) Documents Cited
GB 2336920 A **US 5721825 A**

(58) Field of Search
UK CL (Edition R) **G4A APX AUDB AUXF**
INT CL[7] **G06F 9/46**
ONLINE WPI EPODOC PAJ TDB

(54) Abstract Title
Event notification data processing with command and command notification combined into a single event

(57) When command issuing application 402 determines (501 fig 5) that a command is to be sent to command receiving application 403, it 402 publishes a message (encircled numeral 1) to a publish/subscribe broker system 404 on a stream called "command issuance". The broker system 404 then forwards on the published message to both the command receiving application 403 and a system management tool 401 (encircled numeral 2), because both 403 and 401 have previously registered subscriptions on the stream "command issuance" with the broker system 404. Upon receiving the published message, the command receiving application 403 interprets the command as a command. Upon receiving the same published message the management tool 401 interprets the message as notification of command issuance and logs the message in local memory for informational purposes. When the application 403 has finished carrying out the work instructed by the command it publishes a message on the stream "work completed" (encircled numeral 3) to the broker system 404 which in turn forward the message to the management tool which logs the message and is informed that the command has been acted upon. There may be multiple command issuing and receiving applications. The publish/subscribe broker system may be combined with any of the other applications 401, 402, 403. The command issuing application may be a world wide web (WWW) browser and the command receiving application could be a data base containing stock market data.

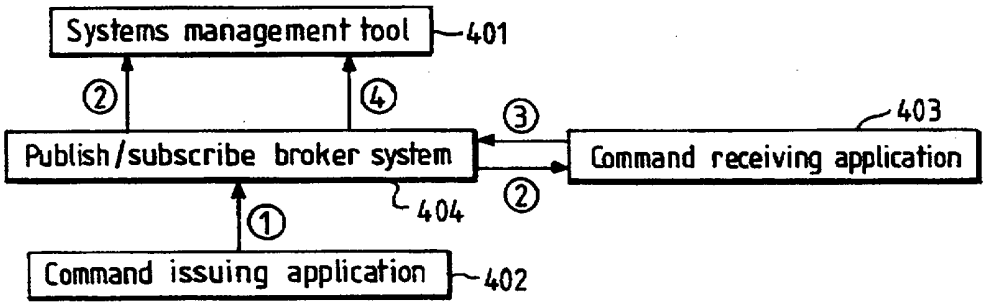

FIG. 4

GB 2 354 349 A

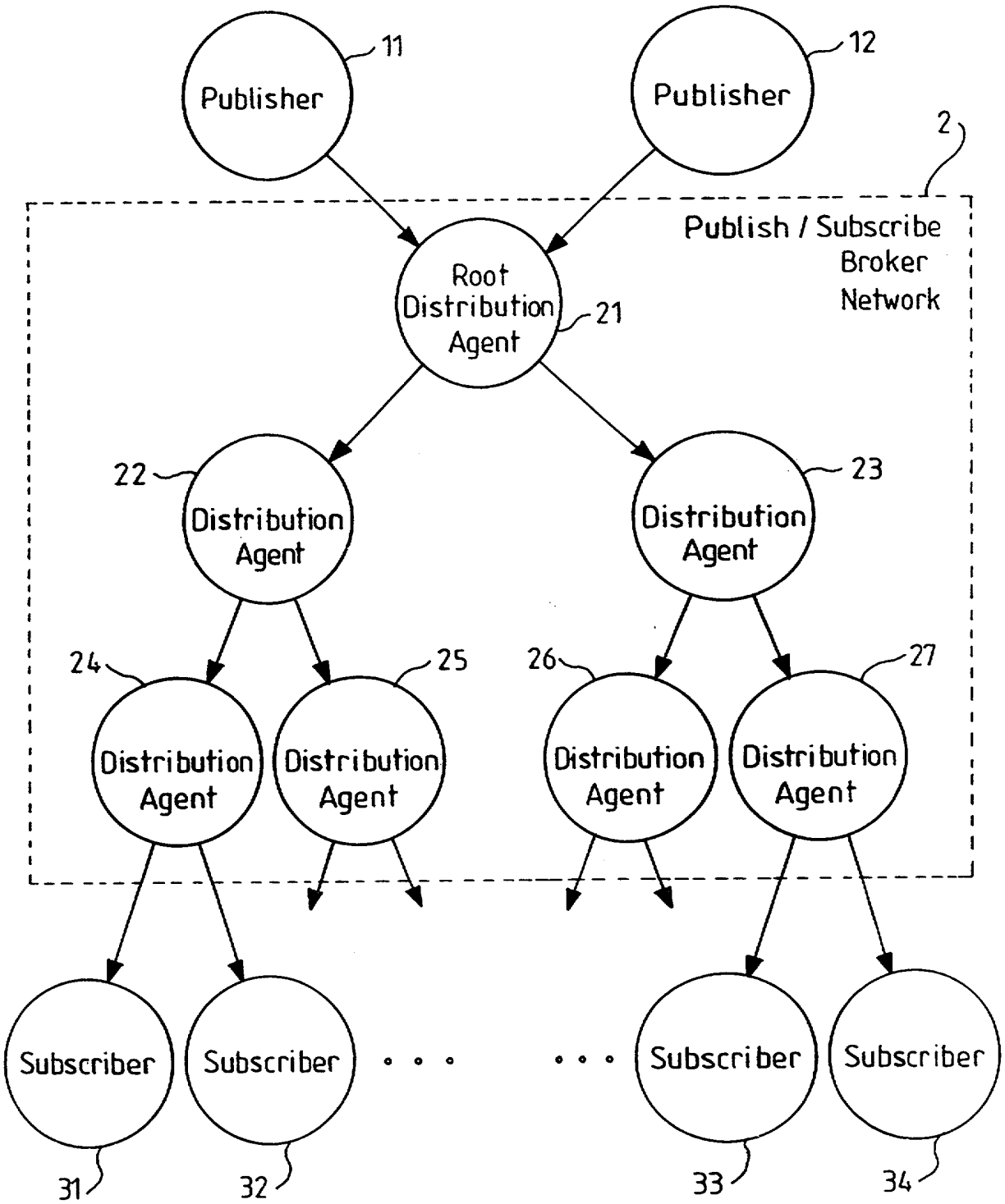
Publisher
11
Publisher
12
2
Publish / Subscribe
Broker
Network
Root
Distribution
Agent
21
22
Distribution
Agent
23
Distribution
Agent
24
Distribution
Agent
25
Distribution
Agent
26
Distribution
Agent
27
Distribution
Agent
Subscriber
Subscriber
Subscriber
Subscriber
31
32
33
34

FIG. 1

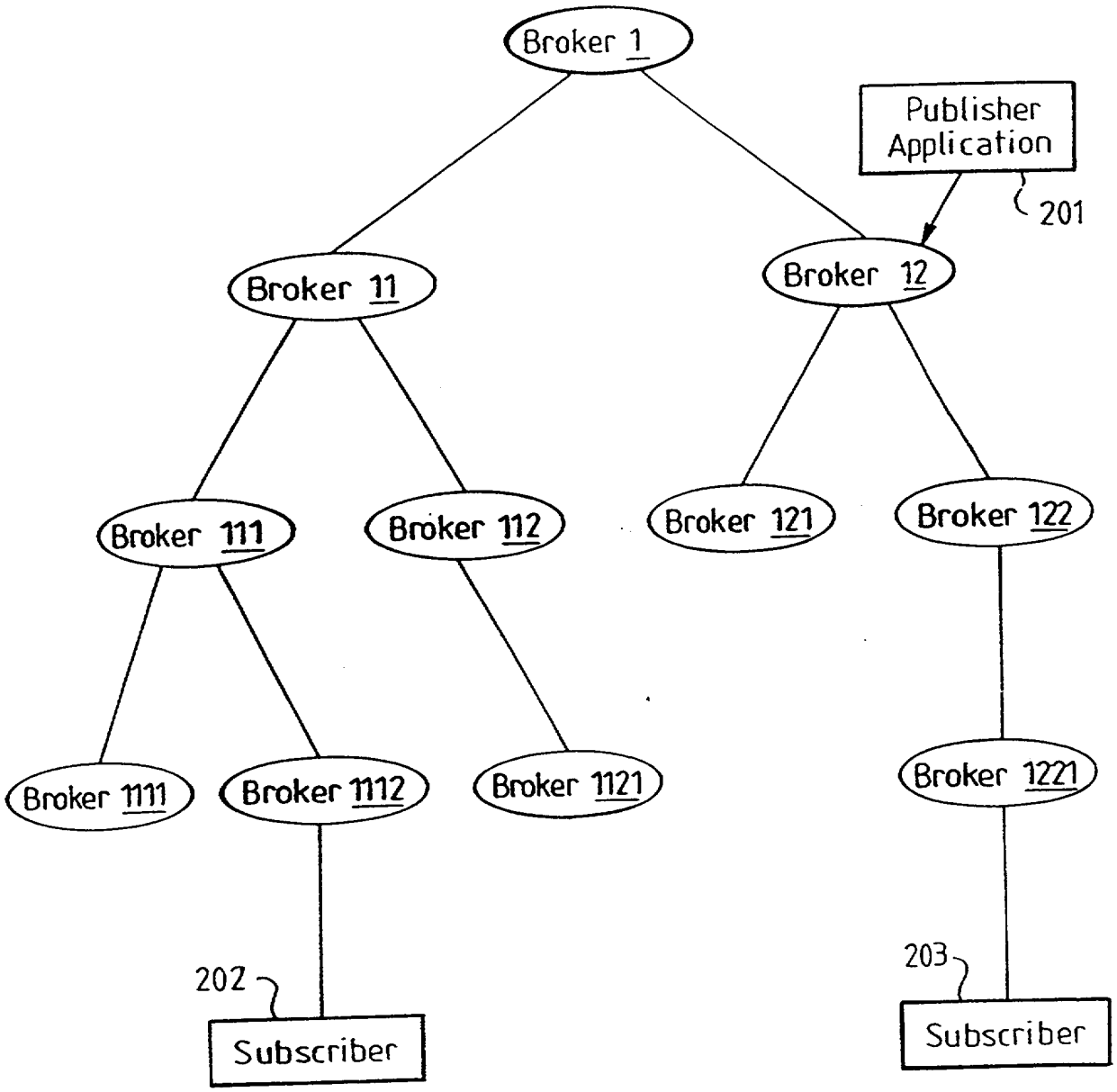
Broker 1
Publisher Application
201
Broker 11
Broker 12
Broker 111
Broker 112
Broker 121
Broker 122
Broker 1111
Broker 1112
Broker 1121
Broker 1221
202
203
Subscriber
Subscriber

FIG. 2

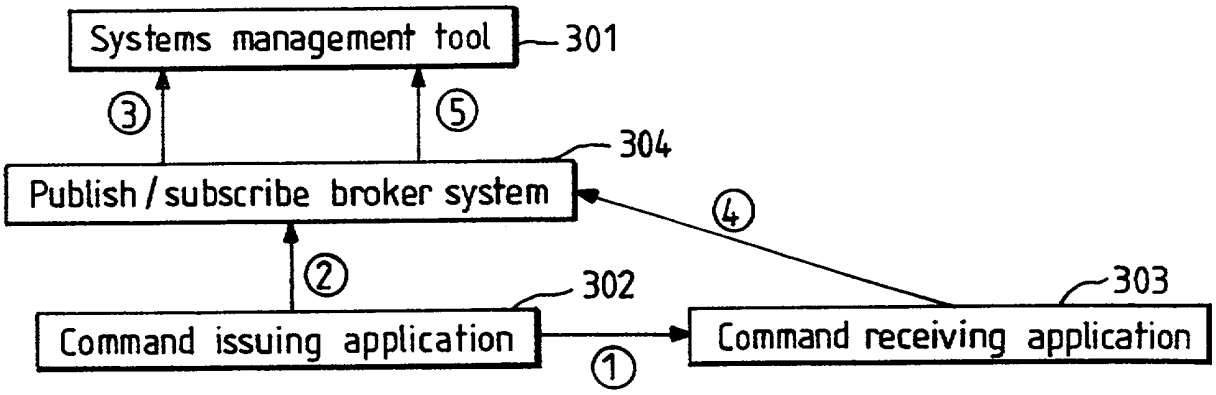

FIG. 3

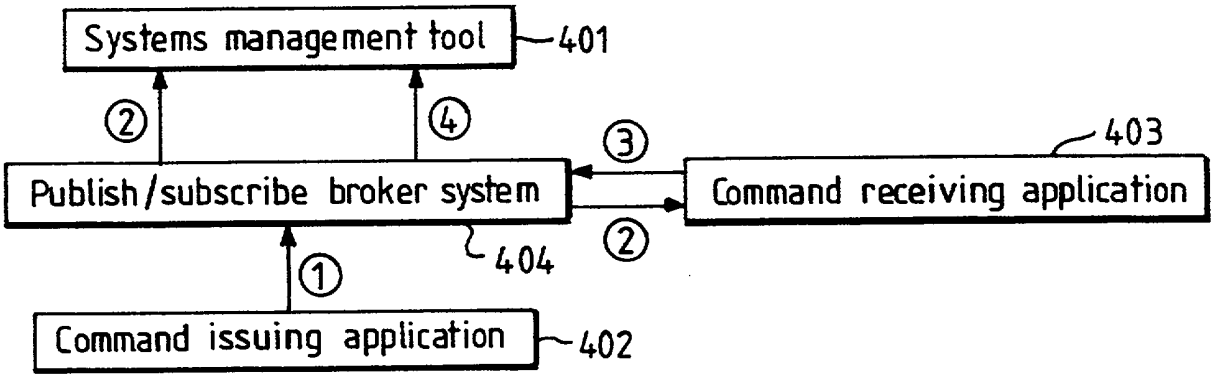

FIG. 4

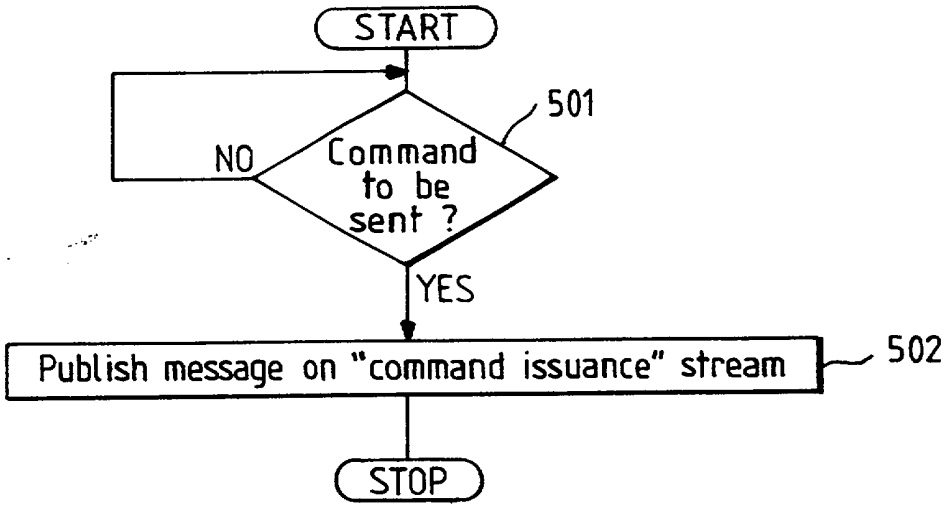

FIG. 5

EVENT NOTIFICATION DATA PROCESSING WITH COMMAND AND COMMAND NOTIFICATION COMBINED INTO A SINGLE EVENT

Field of the Invention

The present invention relates to the field of data processing and more specifically to event notification data processing which distributes event messages from suppliers (called, hereinafter, "publishers") of data messages to consumers (called, hereinafter "subscribers") of such messages. While there are many different types of known event notification systems, the subsequent discussion will describe the publish/subscribe event notification system as it is one of the most common.

Background of the Invention

Publish/subscribe data processing systems (and event notification systems in general) have become very popular in recent years as a way of distributing data messages (events) from publishing computers to subscribing computers. The increasing popularity of the Internet, which has connected a wide variety of computers all over the world, has helped to make such publish/subscribe systems even more popular. Using the Internet, a World Wide Web browser application (the term "application" or "process" refers to a software program, or portion thereof, running on a computer) can be used in conjunction with the publisher or subscriber in order to graphically display messages. Such systems are especially useful where data supplied by a publisher is constantly changing and a large number of subscribers needs to be quickly updated with the latest data. Perhaps the best example of where this is useful is in the distribution of stock market data.

In such systems, publisher applications of data messages do not need to know the identity or location of the subscriber applications which will receive the messages. The publishers need only connect to a publish/subscribe distribution agent process, which is included in a group of such processes making up a broker network, and send messages to the distribution agent process, specifying the subject of the message to the distribution agent process. The distribution agent process then distributes the published messages to subscriber applications which have previously indicated to the broker network that they would like to receive data messages on particular subjects. Thus, the subscribers also do not need to know the identity or location of the publishers. The subscribers need only connect to a distribution agent process.

One such publish/subscribe system which is currently in use, and which has been developed by the Transarc Corp. (a wholly owned subsidiary of the assignee of the present patent application, IBM Corp.) is shown in Fig. 1. Publishers 11 and 12 connect to the publish/subscribe broker network 2 and send published messages to broker network 2 which distributes the messages to subscribers 31, 32, 33, 34. Publishers 11 and 12, which are data processing applications which output data messages, connect to broker network 2 using the well known inter-application data connection protocol known as remote procedure call (or RPC) (other well known protocols, such as asynchronous message queuing protocols, can also be used). Each publisher application could be running on a separate machine, alternatively, a single machine could be running a plurality of publisher applications. The broker network 2 is made up of a plurality of distribution agents (21 through 27) which are connected in a hierarchical fashion which will be described below as a "tree structure". These distribution agents, each of which could be running on a separate machine, are data processing applications which distribute data messages through the broker network 2 from publishers to subscribers. Subscriber applications 31, 32, 33 and 34 connect to the broker network 2 via RPC in order to receive published messages.

Publishers 11 and 12 first connect via RPC directly to a root distribution agent 21 which in turn connects via RPC to second level distribution agents 22 and 23 which in turn connect via RPC to third level distribution agents 24, 25, 26 and 27 (also known as "leaf distribution agents" since they are the final distribution agents in the tree structure). Each distribution agent could be running on its own machine, or alternatively, groups of distribution agents could be running on the same machine. The leaf distribution agents connect via RPC to subscriber applications 31 through 34, each of which could be running on its own machine.

In order to allow the broker network 2 to determine which published messages should be sent to which subscribers, publishers provide the root distribution agent 21 with the name of a distribution stream for each published message. A distribution stream (called hereinafter a "stream") is an ordered sequence of messages having a name (e.g., "stock" for a stream of stock market quotes) to distinguish the stream from other streams (this is known as "topic based" publish/subscribe, another well known model is called "content based publish/subscribe which involves matching publishers and subscribers by the content of the messages rather than by the topic). Likewise, subscribers provide the leaf distribution agents 31 through 34 with the name of the streams to which they would like to subscribe. In this way, the broker network 2 keeps track of which subscribers are interested in which streams so that when publishers publish messages to such streams, the messages can be distributed to the

corresponding subscribers. Subscribers are also allowed to provide filter expressions to the broker network in order to limit the messages which will be received on a particular stream (e.g., a subscriber 31 interested in only IBM stock quotes could subscribe to the stream "stock" by making an RPC call to leaf distribution agent 24 and include a filter expression stating that only messages on the "stock" stream relating to IBM stock should be sent to subscriber 31).

The above-described publish/subscribe architecture provides the advantage of central co-ordination of all published messages, since all publishers must connect to the same distribution agent (the root) in order to publish a message to the broker network. For example, total ordering of published messages throughout the broker network is greatly facilitated, since the root can easily assign sequence numbers to each published message on a stream. However, this architecture also has the disadvantage of publisher inflexibility, since each publisher is constrained to publishing from the single root distribution agent, even when it would be much easier for a publisher to connect to a closer distribution agent.

In the Fig. 1, a publisher application 11, running on one computer, is, for example, a supplier of live stock market data quotes. That is, publisher application 11 provides frequent messages stating the present value of share prices. In this example, publisher application 11 is publishing messages on a stream called "stock" which has already been configured in the broker network 2. As is well known, when publisher 11 wishes to publish a stock quote message to stream "stock", publisher 11 makes an RPC call to the root distribution agent 11 which is at the top level of the broker network tree structure. In this example, subscriber application 32, running on another computer, has sent a subscription request via an RPC call to leaf distribution agent 24, which is at the bottom level of the tree structure, indicating that subscriber 32 would like to subscribe to stream "stock".

Thus, whenever publisher 11 publishes a data message to stream "stock" the distribution tree structure of broker network 2 channels the message down through the root distribution agent 21, through any intermediary distribution agents (e.g., 22 in the example of Fig. 1) and through the leaf distribution agent 24 to the subscriber 32. This involves a series of RPC calls being made between each successive circle in the diagram of Fig. 1 connecting publisher 11 and subscriber 32 (i.e., 11 to 21, 21 to 22, 22 to 24 and 24 to 32).

Figure 2 shows a different publish/subscribe architecture where publisher applications can publish messages to the broker network by directly communicating with any one of a plurality of distribution agents

(brokers). For example, publisher application 201 is shown communicating directly with Broker 12. There is no requirement in this architecture that all publisher applications communicate directly with a top (or root) distribution agent. Publisher application 201 can potentially communicate directly with any of the distribution agents shown in Fig 2, in the described examples below it will be shown communicating directly with Broker 12.

Subscriber applications 202 and 203 would like to receive messages on the stream/topic that publisher application 201 is publishing on. Thus, subscriber applications 202 and 203 communicate directly with Brokers 1112 and 1221, respectively, to provide subscription data thereto informing the broker hierarchy of their desire to receive such published messages. Since the publisher application 201 is allowed to communicate directly with any of a plurality of distribution agents, the subscription data entered by the subscriber applications must be propagated throughout the broker network to each Broker shown in Fig. 2. This way, no matter which distribution agent the publisher application 201 happens to communicate directly with, the published messages will be able to be routed to the subscriber applications 202 and 203.

Such event notification system architectures, as described above, can be used to notify interested parties (e.g., subscribers) of a wide range of different events. For example, a systems management tool may be interested in receiving information on commands which have been sent from any command issuing application (e.g., a client) to any other command receiving application (e.g., a server). This would allow the management tool to determine what work has been requested but has not yet completed, and thus to gain a good overall view of what is happening in a distributed system such as a client/server system where a client process is issuing commands to a server process.

One possible way in which this could be carried out will now be described with reference to Fig. 3. In Fig. 3 systems management tool 301 subscribes to a stream called "command issuance" and thus receives notification of events published to the publish/subscribe broker system 304 by command issuer application 302 (publish/subscribe broker system 304 can be configured either as in Fig.1 or Fig. 2 or in any other known architecture). That is, whenever command issuer application 302 sends a command (along line with encircled numeral 1, these encircled numerals indicate a time sequence) to command receiving application 303 (to ask command receiving application 303 to do some work), command issuer application 302 also publishes an event to the publish/subscribe broker system 304 on stream "command issuance" (arrow with encircled numeral 2) which is then sent (encircled numeral 3) to management tool 301 (which had previously entered a subscription to stream "command issuance").

Command receiving application 303 then performs the commanded work, and when finished, publishes a message on stream "work completed" to the broker system 303 (encircled numeral 4) which is then sent (encircled numeral 5) to management tool 301 (which had previously entered a subscription to stream "work completed"). Command receiving application 303 could, if appropriate, send a reply to command issuing application 302 to inform the latter that the requested work has been completed.

In this way, the management tool 301 is kept updated not only of when requested work has been completed but also of what work has been requested but has not yet been completed. This provides a very complete view of what is happening in the system to the management tool which can then make decisions based on this information to thus improve the overall quality of the system.

The above-described system has a number of inefficiencies, however, that would prevent it from being very useful in a modern data processing environment. Specifically, as described above the command issuer application 302 must send out two different items of information to two different parties: first, a command is sent to command receiving application 303 and then an event is published on stream "command issuance". There is a significant chance that the command issuer application 302 might fail to publish the event or might publish the wrong event. The required two different items of information also adds significantly to the amount of coding involved at the command issuer application 302 and further adds to the escalation of runtime costs for the entire system.

The example given above in Fig. 3 has been given in the management tool environment. However, the problem exists in any event notification system where a command issuer application must, in addition to issuing a command to a command receiving application, also produce an event to be sent via the event notification system to interested parties.

Summary of the Invention

According to one aspect, the present invention provides a data processing apparatus including a unit for determining that a command is to be sent from a command issuing application to a command receiving application; and a unit for forwarding an event to an event notification system which in turn forwards the event to both the command receiving application and a third data processing application; where the command receiving application interprets the event as a command and the third data processing application interprets the event as a notification of command issuance.

According to a second aspect, the present invention provides a data processing method having method steps corresponding to the functionality described above with respect to the first aspect.

According to a third aspect, the present invention provides a computer readable storage medium having a computer program stored on it which, when executed on a computer, carries out the functionality of data processing method of the second aspect of the invention.

Thus, the present invention provides a highly efficient way to keep a third data processing application informed of the status of commands which are sent between a first data processing application which sends a command and a second data processing apparatus which receives the command. The amount of coding at the command issuing data processing application is reduced since the command issuing application need only publish a single message which is then forwarded via the publish/subscribe broker system to both the command receiving application and the third data processing application. Also due to the use of only a single piece of data that needs to be sent, there is a greatly reduced risk that the command sending application will send out data in error. The runtime costs for the overall system are also reduced.

Brief Description of the Drawings

The invention will be better understood by referring to the detailed description of the preferred embodiments which will now be described in conjunction with the following drawing figures:

Figure 1 is a block diagram showing a first architecture of a publish/subscribe data processing system to which the preferred embodiment of the present invention can be advantageously applied;

Figure 2 is a block diagram showing a second architecture of a publish/subscribe data processing system to which the preferred embodiment of the present invention can be advantageously applied;

Fig. 3 is a block diagram showing a systems management tool used in a publish/subscribe data processing system, according to an alternative arrangement which is much less efficient than the preferred embodiment of the present invention;

Fig. 4 is a block diagram showing a systems management tool used in a publish/subscribe data processing system according to a preferred embodiment of the present invention; and

Fig. 5 is a flowchart showing the steps carried out within a command issuing application in the block diagram of Fig. 4.

Detailed Description of the Preferred Embodiments

In Fig. 4, systems management tool 401 is provided for managing the data processing system which is represented in Fig. 4 by command issuing application 402 and command receiving application 403 (but would normally have many other participants which are not directly involved in the below discussion and are thus not illustrated in Fig. 4). 401, 402 and 403 are applications which may be running on separate data processing machines (in which case, a network, such as the Internet, is required to provide communication between the machines) or may all be running on the same machine. 401, 402 and 403 use a publish/subscribe system 404 (such as that of Figs. 1 or 2) in order to communicate with each other, in the preferred embodiment.

When command issuing application 402 determines (step 501 of Fig. 5) that a command is to be sent to command receiving application 403, command issuing application 402 publishes (step 502) a message (encircled numeral 1) to the publish/subscribe broker system 404 on a stream called "command issuance". The broker system 404 then forwards on the published message to both the command receiving application 403 and the systems management tool 401 (see lines with encircled numeral 2), because both the command receiving application 403 and the systems management tool 401 have previously registered subscriptions on stream "command issuance" with the broker system 404.

Upon receiving the published message, the command receiving application 403 interprets the message as a command and carries out the work that has been requested in the command. Upon receiving the same published message, the systems management tool 401 logs the message in local memory for informational purposes, but does not interpret the message as a command. That is, while the command receiving application 403 interprets the published message as a command, the management tool 401 interprets the same published message as a notification that a command has been sent.

Once the command receiving application 403 is finished carrying out the work instructed by the command, the command receiving application 403 publishes a message on stream "work completed" (encircled numeral 3) to the broker system 404 which in turn forwards on the published message to the systems management tool 401 (encircled numeral 4) since tool 401 had previously registered a subscription to stream "work completed" with the broker system 404. When the tool 401 receives this message, the tool logs the information in local storage and thus is informed that the

command received along the line with encircled number 2 has now been completely executed. Command issuing application 402 could also subscribe to stream "work completed" and thus be informed that command receiving application 403 has finished the work.

Because of the flexibility of the publish/subscribe broker system 404, it is, of course, possible that there could be multiple command receiving applications 403 all of which receive the same command from a command issuing application 402. Each command receiving application would then subscribe to the stream "command issuance" in order to receive the command, and each command receiving application would also publish to the stream "work completed" to notify interested parties that the work for that particular command receiving application is now done. Further, there could also be multiple instances of the systems management tool 401, each of which would subscribe to the stream "work completed". Still further, there could be multiple command issuing applications 402 each of which publishes on the stream "command issuance".

Further, although the publish/subscribe broker system 404 is illustrated in Fig. 4 as separate and distinct from the other three applications 401, 402 and 403, the system 404 could be combined with any of these other three applications 401, 402 and 403.

CLAIMS

1. A data processing apparatus comprising:

means for determining that a command is to be sent from a command issuing application to a command receiving application; and

means for forwarding an event to an event notification system which in turn forwards the event to both the command receiving application and a third data processing application;

wherein the command receiving application interprets the event as a command and the third data processing application interprets the event as a notification of command issuance.

2. The apparatus of claim 1 wherein:

the event notification system is a publish/subscribe system;

the command issuing application is a publisher; and

the command receiving application and third data processing application are subscribers.

3. The apparatus of claim 1 wherein the third data processing application is a systems management tool.

4. The apparatus of claim 2 wherein said publish/subscribe system operates over the Internet and wherein at least one of the subscribers and the publisher runs in conjunction with a World Wide Web browser.

5. A data processing method comprising steps of:

determining that a command is to be sent from a command issuing application to a command receiving application; and

forwarding an event to an event notification system which in turn forwards the event to both the command receiving application and a third data processing application;

wherein the command receiving application interprets the event as a command and the third data processing application interprets the event as a notification of command issuance.

6. The method of claim 5 wherein:

the event notification system is a publish/subscribe system;

the command issuing application is a publisher; and

the command receiving application and third data processing application are subscribers.

7. The method of claim 5 wherein the third data processing application is a systems management tool.

8. The method of claim 6 wherein said publish/subscribe system operates over the Internet and wherein at least one of the subscribers and the publisher runs in conjunction with a World Wide Web browser.

9. A computer program product stored on a computer readable storage medium for, when run on a computer, instructing the computer to carry out the method steps recited in claim 5.

The Patent Office

INVESTOR IN PEOPLE

Application No: GB 9921776.2
Claims searched: ALL

Examiner: Russell Maurice
Date of search: 7 April 2000

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.R): G4A (AUDB, AUXF, APX)

Int Cl (Ed.7): G06F (9/46)

Other: Online WPI EPODOC PAJ TDB

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	GB 2336920 A	IBM (see eg the abstract)	-
A	US 5721825 A	Lawson (see eg the abstract)	-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.