PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 7/00, H04K 1/00, G06F 13/00, H04M 3/58

A1

(11) International Publication Number: **WO 99/08181**

(43) International Publication Date: 18 February 1999 (18.02.99)

(21) International Application Number: PCT/US98/15860

(22) International Filing Date: 30 July 1998 (30.07.98)

(30) Priority Data:
08/907,752 8 August 1997 (08.08.97) US

(71) Applicant: BELL COMMUNICATIONS RESEARCH, INC. [US/US]; 445 South Street, Morristown, NJ 07960-6438 (US).

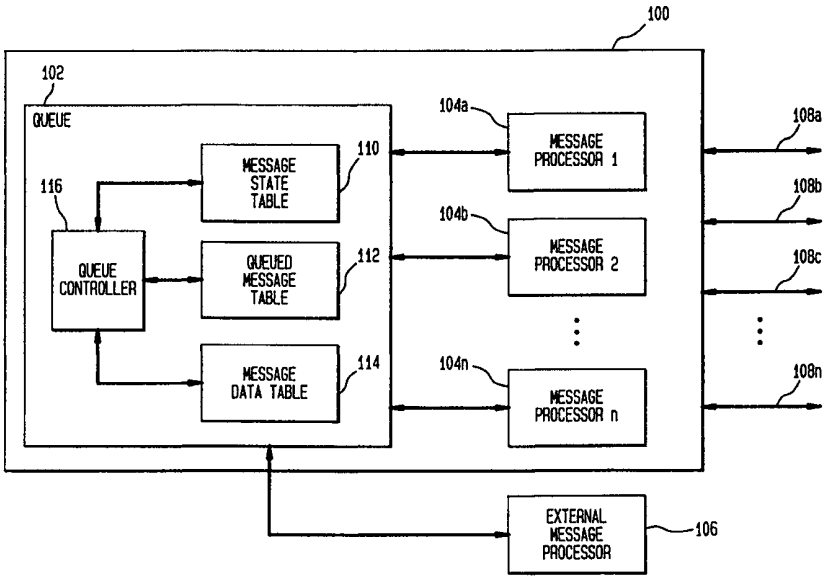
(72) Inventors: COCHINWALA, Munir; 14 Bullion Road, Basking Ridge, NJ 07920 (US). LEE, Kuo-Chu; 30 Revere Court, Princeton Junction, NJ 08850 (US). YU, Min, Tae; 3 Governor Drive, Basking Ridge, NJ 07920 (US). CHENG, Lee-Tin; 8 Wright Court, East Brunswick, NJ 08816 (US). LIU, Won-Pin, Chi; 5 Overlook Drive, Warren, NJ 07921 (US). HWANG, Ai-Hwa; 40 Cambridge Road, Bedminster, NJ 07921 (US). TONG, Chao-Chi; 254 Hyde Park Road, Somerset, NJ 08873 (US).

(74) Agents: GIORDANO, Joseph et al.; International Coordinator, Rm. 1G112R, 445 South Street, Morristown, NJ 07960-6438 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: FAULT TOLERANT DISTRIBUTING CLIENT/SERVER DATA COMMUNICATION SYSTEM AND METHOD



(57) Abstract

A processing environment includes a queuing system (102) and method to provide message communication between computers, processing devices, or network elements. A table configuration (110, 112, 114) and processes within the queue enable replication of table data based on time-stamp information (908). A process control monitoring system (902) and method permits further replication of table data to a secondary database (920) and monitors the processing environment (700) based on the further replicated data.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

FAULT TOLERANT DISTRIBUTING CLIENT/SERVER DATA COMMUNICATION SYSTEM AND METHOD

Field of the Invention

The present invention relates to the fields of process and system management and computer communications. In particular, the present invention relates to the fields of fault tolerance and reliable message passing in distributed systems.

Background of the Invention

Computers, processors, and network elements communicate with each other by sending and receiving control and data messages. In large, complex networks, such as today's advanced intelligent telephone network ("AIN"), demands for message processing are extremely high. Such large networks must process millions of messages every day, with very little margin for error. Thus, efficient and effective message processing in such networks is a premium. Network elements that comprise networks are often connected to a plurality of other network elements and communicate with them in parallel, which further increases the message processing demands.

To effect message processing, computers, processors, and network elements typically include a plurality of message processors that process, send, and receive queued messages.

Conventional message processors typically process messages from a first-in-first-out (FIFO) message queue. Since only one processor can access the queue at one time, problems of scale and performance occur. Adding multiple processors compounds problems because they block each other out trying to access the queue. In these conventional systems, failed messages, i.e. messages destined for a computer, processor, or network element that for some reason is not communicating, are simply placed back into the FIFO queue each time they fail. The continuous processing of these failed messages wastes significant and valuable processing time.

In various processing environments, data must also be replicated to alternative storage databases at various times to ensure that the data is not lost. Processors in many stringent environments, such as telephone networks, have very specific data integrity requirements, which significantly increase the replication burden.

For asynchronous data replication, data is replicated periodically from one database to another. If one database fails, data is available from the other database. A replication interval is defined to control how frequently replication is performed.

Conventional replication approaches use a log to track data. For example, for each database operation (such as insert, update, or delete) on a table to be replicated, a log is created, updated, or deleted to keep track of the operation. This approach results in the overhead of database operations due to the additional manipulation of the log. In addition, the size of the log is proportional to the number of operations. For instance, if there are several updates on a record, there will be the same number of logs even though only the last update may need to be replicated. This approach results in significant replication overhead. Thus, conventional asynchronous data replication techniques are limited by both operations and replication overhead.

To meet strict fault tolerance requirements, conventional systems also employ mechanisms to monitor and control application processes, databases, and machines, ensuring no interruption of services. Distributed systems running on heterogeneous platforms must include mechanisms that provide failure-detection, fail-over, switch-over, and switch-back processes.

Conventional approaches provide proprietary solutions only for hardware failures with operating system and hardware support. These systems are useful in that they monitor certain hardware operations, but they do not monitor application level failures. Nor are they portable to heterogeneous environments.

Accordingly, it is an object of the present invention to overcome the disadvantages of conventional hardware solutions.

It is another object of the present invention to overcome the disadvantages and drawbacks of conventional fault tolerant and data replication systems and techniques.

Additional objectives, features, and advantages of the invention will be set forth in the description which follows, and in part will be apparent from the description, or may be learned by practice of the invention. The objectives and other advantages of the invention will be realized and attained by means of the instrumentalities and combinations particularly pointed out in the written description and appended claims hereof as well as the appended drawings.

Description of the Invention

To achieve these and other advantages and in accordance with the purposes of the invention, as embodied and broadly described, the invention includes a computer system, including one or more message processors in a primary processing environment of the

computer system for communicating with one or more other processing environments in the computer system. A message queue in the primary processing environment includes a second table to store information about each message, including a message state, a message partition, and a time-stamp; the message partition being an identifier to associate a message with a particular one of the one or more message processors; the time-stamp indicating a time each message is entered or changed. A process control monitor monitors processes in the primary processing environment and includes a device for periodically storing table information into the database, a device for monitoring time-stamp information of the stored table information, and a device for initiating action to correct any problems in the primary processing environment when the time-stamp information of the stored data does not correspond to a predetermined time interval.

The invention further includes a computer processing method, including the steps of assigning a partition and time-stamp to a message, creating an entry for the message in a table, the entry including the partition, the time-stamp, and a state, selecting messages for transmission based on the partition, periodically identifying table data having a time-stamp later than a predetermined time, writing the identified table data to a first replication database, and periodically writing said identified table data from said database to a second database.

It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory and are intended to provide further explanation of the invention as claimed.

Brief Description of the Drawings

Fig. 1 is a block diagram of a computer and queue configuration in accordance with a preferred embodiment of the present invention;

Fig. 2 is a block diagram of a message state table in accordance with a preferred embodiment of the present invention;

Fig. 3 is a block diagram of a message data table in accordance with a preferred embodiment of the present invention;

Fig. 4 is a block diagram of a queued message table in accordance with a preferred embodiment of the present invention;

Fig. 5 is a process flow diagram of the operation of a message controller in accordance with one embodiment of the present invention;

Fig. 6 is a process flow diagram of the operation of a message processor in accordance with one embodiment of the present invention;

Fig. 7 is a block diagram of a computer and a table to be replicated in accordance with one embodiment of the present invention;

Fig. 8 is a table replication process in accordance with one embodiment of the present invention;

Fig. 9 is a block diagram of a process control monitoring system in accordance with one embodiment of the present invention;

Fig. 10 is a process flow diagram of the operation of a primary processing environment to provide process control monitoring in accordance with one embodiment of the present invention; and

Fig. 11 is a process flow diagram of the operation of a secondary processing environment to provide process control monitoring in accordance with one embodiment of the present invention.

Best Mode for Carrying Out the Invention

Reference will now be made in detail to the construction and operation of preferred implementations of the present invention which are illustrated in the accompanying drawings.

The following description of the preferred implementations of the present invention is only exemplary of the invention. The present invention is not limited to these implementations, but may be realized by other implementations.

Fig. 1 is a block diagram of a computer 100 having a queue configuration in accordance with a preferred embodiment of the present invention. Computer 100 may comprise, for example, a service management system ("SMS") ("SCP") within today's AIN. SMSs are typically connected to service control points ("SCP") by X.25 connections.

As shown, computer 100 includes a queue 102 and a plurality of message processors 104a-104n. Queue 102 preferably comprises a database configuration which is connected to and communicates with the plurality of message processors 104a-104n. Message processors 104a-104n preferably include read/write or send/receive processing applications for performing input/output operations between computer 100 and any one of several computers (not shown) connected to computer 100 via communication lines 108a-108n. Message processors from other computers (external message processors 106) may also use queue 102 to communicate with computers to which it is connected (not shown).

In accordance with a preferred embodiment of the invention, queue 102 includes state table 110, queued message table 112, message data table 114, and queue controller 116. In accordance with the present invention, message data is included in message data table 114, while parameters used to optimize communications are stored in message state table 110. Queued message table 112 stores certain parameters also found in the message state table 110 for any failed messages, i.e. messages that could not be communicated to the intended

destination computer. Queue controller 116 preferably includes an application for controlling queue 102 and the three tasks 110, 112, and 114 as further described herein.

Fig. 2 is a block diagram of a message state table 110 in accordance with a preferred embodiment of the present invention. As shown, for each message to be communicated, message state table 110 preferably includes the following information: message I.D. 200, destination I.D. 202, partition 204, state 206, and time-stamp 208.

Message I.D. 200 uniquely identifies the message, destination I.D. 202 identifies a computer for which the message is destined, and time-stamp 208 specifies the time at which the message entered the FIFO queue. The time-stamp 208 is described in more detail below.

Partition 204 links a message to one of the message processors 104a-104n. As described below, in a preferred embodiment, message controller 116 assigns each message a partition as it enters queue 102. Message controller 116 preferably distributes messages evenly across each of the corresponding message processor 104a-104n. For example, if computer 100 includes ten message processors, for 100 messages, 10 are assigned to each message processor. This distribution is, however, configurable. In accordance with the present invention, each message processor 104a-104n processes only messages having the corresponding partition. In this manner, message processors are not locked out and can perform parallel processing without risk of overlap.

Referring again to Fig. 2, state 206 preferably identifies one of five message states or conditions: 1) ready for send; 2) sending; 3) processed; 4) ready for delete; 5) queued. The first four states are conventional communication states. The fifth state, "queued" specifies the state of being included in the queued message table 112, which is described in more detail below.

Fig. 3 shows a block diagram of a message data table 114n in accordance with a preferred embodiment of the invention. Message data table 314 includes message I.D. 300 and data 302. Thus, in accordance with a preferred embodiment of the present invention, the actual message data 302 is listed in a table 314 separate from the message state table 110. Message I.D.s 300 correspond to message I.D.s 200.

Fig. 4 illustrates a block diagram of a queued message table 112 in accordance with a preferred embodiment of the present invention. Queued message table 112 stores messages destined for computers (not shown) that are down or for some other reason are not receiving messages. In accordance with conventional techniques, computer 100 determines from one or more failed communication transmissions that a connected computer (not shown) is no longer "on-line." In accordance with a preferred embodiment of the present invention, messages which cannot be transmitted by one of message processors 104a-104n, are stored in the queued message table 112 until the destination machine provides an indication that it is

again capable of receiving messages. At that time, state 206 of any queued message is changed from "queued" to "ready to send."

As shown in Fig. 4, for each queued message, the queued message table 112 includes message I.D. 400, destination I.D. 402, partition 404, and time-stamp 406. Each of these headings corresponds to the same information of the corresponding heading in the message state table 110 as shown in Fig. 3.

Fig. 5 is a processing flow diagram of the operation of queue controller 116 to help illustrate the methods by which messages are processed in accordance with an embodiment of the present invention.

Initially, queue controller 116 receives a message to be transmitted to any of one or more connected machines (not shown) (step 500). Queue controller 116 then assigns the message a message I.D., a destination I.D., and a time-stamp(step 502). Queue controller 116 then assigns a partition to the message, as discussed above (step 504). Having assigned this information to the message, queue controller 116 assigns the message I.D. 300 and data 302 to the message data table 114 (step 506), and builds an entry for the message state table 110 (step 508).

Queue controller 116 then inquires whether the destination computer for that message is "on-line" (step 510). If not, it builds an entry for the queued message table 112 (step 512) and assigns the message a "queued" state in the message state table (step 514). If, however, the destination machine is "on-line," queue controller 116 assigns the message a "ready for send" state (step 516). Queue controller 116 then inquires whether any "off-line" destination machines have come back "on-line" (step 518). If not, processing is complete. If so, however, queue controller 116 changes the state of those messages intended for the back "on-line" destination computers to "ready for send" (step 520), before returning to process a new message.

Fig. 6 is a process flow diagram of the operation of any one of message processors 104a-104n or 106 for sending a message to a destination machine. Initially, a message processor 104 scans the message state table 110 and selects the first message with a state "ready to send" (step 600). Message processor 104 then sends that message to the destination machine (step 602) and changes the message state to "processed" (step 604). Message processor 104 then waits for a response from the destination machine (step 606). If it receives a response within a predetermined time period, message processor 104 changes the message state to "delete" (step 608). The message can then be removed from the queue 102. The predetermined time period is configurable and is preferably five seconds.

If the message processor 104 does not receive a response from the destination machine within the predetermined time period, it executes process steps like those described in steps 510-514 in Fig. 5.

In a preferred embodiment of the present invention, queue controller 116 and message processors 104-104a comprise software applications programmed to execute the corresponding functions described herein. However, in accordance with the present invention, these elements may comprise any form of conventional hardware processor or controller, independent or otherwise, or any combination of hardware and software.

Fig. 7 is a block diagram of a processing environment requiring a table to be replicated in accordance with one embodiment of the present invention. Processing environment 700 may correspond to any computer, processor, or network component that processes information, builds tables, and replicates those tables to a database, or any other form of data replication environment. Such a processing environment may exist, for example, in a reliable message passing system of the telephone network, including, e.g. communications between a service management system and other network elements for "800 services." Thus, for example, processing environment 700 may correspond to the computer 100 and table 706 may correspond to any of tables 110, 112, and 114 shown in Fig. 1.

As shown in Fig. 7, processing environment 700 includes a processor 702, a replication database 704, and a table 706. Replication database 704 corresponds to any suitable storage configuration for storing a replication table 705 corresponding to table 706. In accordance with conventional techniques, processor 702 builds a table 706 in processing environment 700 by adding, updating, and deleting data as necessary.

In accordance with the present invention, table 706 includes an additional "time-stamp" column 708. Each time processor 702 writes an entry to table 706 or performs an operation affecting a table entry in table 706, processor 702 time-stamps that operation by inserting the time of the operation under the time-stamp column 708.

Fig. 8 provides a processing flow diagram of a replication process in accordance with one embodiment of the present invention.

Initially, processor 702 determines whether a replication cycle has expired (step 800). If not, it waits for a predetermined period (step 802) and returns to the initial step 800. The replication cycle preferably corresponds to a preselected time period for writing to the replication table in replication database 704. In a preferred embodiment, this replication cycle is chosen to be ten seconds; however, the replication cycle is configurable and can be selected to be any predetermined time period depending on the application.

If processor 702 determines that the replication cycle has expired in step 800, it next checks the time of the last replication (step 804). Against that information, processor 702

determines whether the time-stamp of the first table entry is later than the last replication time (step 806). If so, processor 702 marks that table entry for replication (step 808). If the time-stamp of the table entry being checked is not later than the last replication time, processor 702 determines whether any table entries are left (step 810). Likewise, after processor 702 marks a particular table entry for replication, it checks for any table entries left (step 810). If a table entry still exists, processor 102 returns to step 806 and checks through each existing table entry. Once all table entries have been returned, processor 702 replicates each marked table entry by copying those marked table entries to the replication table 705 in replication database 704 (step 812). Processor 702 then updates the stored replication time (step 814) and returns to the beginning of the process.

In this manner, the present invention need not replicate an entire table of data. Instead, the process replicates only table entries that have been modified in some way since a last replication cycle.

Fig. 9 is a block diagram of a process control monitoring system in accordance with one embodiment of the present invention. As shown, a process control monitoring system in accordance with one embodiment of the present invention is distributed across a primary processing environment 902 and a secondary processing environment 904. Processing environments 902 and 904 may correspond to a computer, processor, or network component that processes information and replicates that information. For example, primary processing environment 902 may correspond to the computer or processing environments shown in Figs. 1 or 7 and described above.

Primary processing environment 902 includes a plurality of processes or applications 906a-906n, a primary process control monitor ("PCM") 908 is connected to each of the processes 906, and a shared memory 910. In accordance with the present invention, primary PCM 908 monitors processes 906 and determines whether each should be running, brought-up, and/or shut-down. It further detects which processes are "hung-up." For example, at predetermined time intervals, each process 106 writes status information to primary PCM 908, which then stores the status information in shared memory 910. The information in shared memory 910 can be used to monitor processes 906 and other applications (not shown). To improve fault tolerance, the content of shared memory 910 is periodically replicated to primary database 912. This replication process may be performed as described above with regard to Figs. 7 and 8.

The configuration of secondary processing environment 904 is similar to that of primary processing environment 902. Specifically, secondary processing environment 904 includes secondary PCM 914 for monitoring individual processes 916a-916n. Secondary processing environment 904 also includes a shared memory 918 for storing any process data

from processes 916a-916n. In addition, secondary processing environment 914 is connected to a secondary database 920 to replicate data for efficiency and fault tolerance.

Secondary PCM 914 differs from primary PCM 908, however, in accordance with one embodiment of the invention, in that secondary PCM 914 monitors primary processing environment 902 at the process or application level. As described in more detail below, in accordance with the present invention, process data stored in primary database 912 is periodically replicated to secondary database 920. Secondary PCM 914 uses this replicated data to monitor the process performance of primary processing environment 902 and take over processing where necessary.

Fig. 10 is a process flow diagram of the operation of primary processing environment 902 to provide process control monitoring in accordance with one embodiment of the present invention. As described above, primary PCM 908 monitors the operation of processes 906 at a predetermined interval. Thus, primary PCM 908 initially determines whether monitoring interval T_1 has expired (step 1000). If not, the primary PCM 908 continues to monitor. If time T_1 has expired, primary PCM 908 checks each process (step 1002) and determines whether it is malfunctioning (step 1004). If a malfunction exists, primary PCM 908 preferably restarts or corrects the process (step 1006). If no malfunctions exist, primary PCM 908 determines whether it is time to replicate the shared memory 910 data to primary database 912. Specifically, primary PCM 908 determines whether a preselected replication time interval T_2 has expired (step 1008). If not, primary PCM 908 continues to wait. Once T_2 has expired, primary PCM 908 writes the process data from shared memory 910 to primary database 112 (step 1010).

In one embodiment of the present invention, this replication process writes all data to primary database 912. However, in accordance with another embodiment of the invention, primary PCM 908 only writes predetermined portions of the data. For example, the replication process may be based on time-stamped data as described above.

To enable additional process monitoring, in accordance with the present invention, data from primary database 912 is further replicated to secondary database 920. Accordingly, primary PCM 908 then determines whether it is time to replicate data from primary database 912 to secondary database 920. Specifically, primary PCM 908 determines whether a third preselected time interval T_3 has expired. T_3 is configurable and is preferably selected to ensure accuracy in a fault-tolerant design depending on the system or network configuration and corresponding application. For example, in a telecommunication environment such as the telephone network where certain standards require very strict fault-tolerance, this time period T_3 would be relatively short, for example twenty seconds.

When primary PCM 908 determines that T_3 has not expired, it continues other steps of its normal processing. However, when primary PCM 908 determines that time T_3 has expired, it replicates the primary database information to the secondary database (step 1014). In a preferred embodiment, this data replication process is also performed based on the time-stamped data, much like the data replication between shared memory 910 and primary database 912. In other words, only data whose status has been changed or updated since the beginning of the replication interval T_3 is replicated from primary database 912 to secondary database 920.

In accordance with the present invention, secondary PCM 914 uses the replicated data stored in secondary database 920 to monitor the processes of the primary processing environment 902. For example, secondary PCM 914 may monitor the time-stamp information corresponding to each operation. If the replicated data includes time-stamps corresponding to the most recent replication interval T_3, then processes 906 of primary processing environment 902 are functioning properly and the replication process is functioning properly. However, if the time-stamps are old, then a problem exists in either the processes 906 or the replication process.

Fig. 11 is a process flow diagram of a process executed by a secondary PCM 914 to provide process control monitoring in accordance with one embodiment of the invention. Secondary PCM 914 periodically monitors the time-stamps of data replicated to secondary database 920. The time interval T_4 for this monitoring step is also configurable, and again, in networks or systems requiring strict fault tolerance, this interval would be shortened. For example, in telephone networks, the interval might be every twenty seconds.

Secondary PCM 914 initially determines whether the time interval T_4 has expired (step 1100). If not, it continues to monitor. If time interval T_4 has expired, secondary PCM 914 checks the time-stamps on the replicated data in secondary database (step 1102). Secondary PCM 914 then determines whether any time-stamps correspond to the most recent replication interval T_3 (step 1104). If they are, the processes 906 and the replication process are working properly and secondary PCM 914 continues normal monitoring until the next interval T_4. However, if the secondary PCM 914 determines that none of the time-stamps correspond to the most recent replication interval T_3, then a problem exists in either the processes 906 or the replication process.

In a preferred embodiment, secondary PCM 914 then determines whether the primary processing environment 902 is still "alive" (functioning properly and/or on-line) (step 1106). If it is, secondary PCM 914 preferably requests help (step 1108). Because primary processing environment 902 is still alive but the updated information is not accurate, certain problems can be presumed. For example, certain processes 906 may be down. In one

embodiment, secondary PCM 914 may take steps to automatically reinitialize various processes 906 based on these presumptions or based on various data from additional diagnostic tools or initialize certain automatic diagnostic procedures. Alternatively, secondary PCM 114 may provide an alarm indication and request manual intervention to further diagnose problems at the primary processing environment 902.

If primary processing environment 902 is not alive (step 1106), secondary processing environment 904 takes over the processes of primary processing environment 902 (step 1110). Secondary processing environment 904 is preferably configured to include the same processes and functionality as primary processing environment 902 for fault tolerance and backup purposes. When primary processing environment 902 goes down, secondary processing environment 904 initializes the processes or applications necessary and substitutes itself and the corresponding processes for that of the downed primary processing environment 902. Although not shown in Fig. 11, secondary processing environment 904 continues performing the processes of primary processing environment 102 until primary processing environment 902 comes back on line. In this manner, fault tolerance is highly improved as both processes and machine state are ultimately monitored.

While there has been illustrated and described what are at present considered to be preferred embodiments and methods of the present invention, it will be understood by those skilled in the art that various changes and modifications may be made, and equivalents may be substituted for elements thereof without departing from the true scope of the invention.

In addition, many modifications may be made to adapt a particular element, technique or implementation to the teachings of the present invention without departing from the central scope of the invention. Therefore, it is intended that this invention not be limited to the particular embodiments and methods disclosed herein, but that the invention include all embodiments falling within the scope of the appended claims.

Claims

1. A computer system, comprising:

one or more message processors in a primary processing environment of said computer system for communicating with one or more other processing environments in said computer system;

a message queue in said primary processing environment including a table to store information about each message, said information including a message state, a message partition, and a time-stamp, said message partition being an identifier to associate a message with a particular one of said one or more message processors, said time-stamp indicating a time each message is entered or changed; and

a process control monitor for monitoring processes in said primary processing environment including:

means for periodically storing table information into said database;

means for monitoring time-stamp information of said stored table information; and

means for initiating action to correct any problems in said primary processing environment when said time-stamp information of said stored data does not correspond to a predetermined time interval.

2. A computer system, comprising:

one or more message processors in a primary processing environment of said computer system for communicating with one or more other processing environments in said computer system;

a message queue in said primary processing environment including a table to store information about each message, said information including a message state, a message partition, and a time-stamp, said message partition being an identifier to associate a message with a particular one of said one or more message processors, said time-stamp indicating a time each message is entered or changed; and

a process control monitor for monitoring processes in said primary processing environment, including:

a database;

means for periodically storing table information into said database;

means for monitoring time-stamp information of said stored table information;

means for determining whether said primary processing environment is on-line if said time-stamps of any of said identified table information do not correspond to said predetermined interval; and

means for initializing processes in a secondary processing environment in said computer system and taking over processing functions of said primary processing environment at said secondary environment if said primary processing environment is not on-line.

3. A computer system, comprising:

one or more message processors in a primary processing environment of said computer system for communicating with one or more other processing environments in said computer system;

a message queue in said primary processing environment including a table to store information about each message, said information including a message state, a message partition, and a time-stamp, said message partition being an identifier to associate a message with a particular one of said one or more message processors, said time-stamp indicating a time each message is entered or changed;

said primary processing environment including:

a first database;

means for periodically identifying table information having a time-stamp later than a predetermined time;

means for writing said identified table information to said first database; and

a second processing environment including:

a second database;

means for periodically receiving said identified table information from said first database and storing said identified table information in said second database;

means for determining whether said time-stamps of any of said identified table information correspond to a predetermined time interval; and

means for initiating action to correct any problems in said primary processing environment if said time-stamps of any of said identified table information do not correspond to said predetermined time interval.

4. A computer system, comprising:

one or more message processors in a primary processing environment of said computer system for communicating with one or more other processing environments in said computer system;

a message queue in said primary processing environment including a table to store information about each message, said information including a message state, a message partition, and a time-stamp, said message partition being an identifier to associate a message with a particular one of said one or more message processors, said time-stamp indicating a time each message is entered or changed;

said primary processing environment including:

a first database;

means for periodically identifying table information having a time-stamp later than a predetermined time;

means for writing said identified table information to said first database; and

a second processing environment including:

a second database;

means for periodically receiving said identified table information from said first database and storing said identified data in said second database;

means for determining whether said time-stamps of any of said identified table information correspond to a predetermined time interval;

means for determining whether said primary processing environment is on-line if said time-stamps of any of said identified table information do not correspond to said predetermined interval; and

means for initializing processes in said secondary processing environment and taking over processing functions of said primary processing environment at said secondary environment if said primary processing environment is not on line.

5. A computer system according to claim 1, further comprising a second table to store messages destined for said one or more other processing environments when communication fails with said one or more other processing environments.

6. A computer system according to claim 2, further comprising a second table to store messages destined for said one or more other processing environments when communication fails with said one or more other processing environments.

7. A computer system according to claim 3, further comprising a second table to store messages destined for said one or more other processing environments when communication fails with said one or more other processing environments.

8. A computer system according to claim 4, further comprising a second table to store messages destined for said one or more other processing environments when communication fails with said one or more other processing environments.

9. A computer processing method, comprising the steps of:

assigning a partition and time-stamp to a message;

creating an entry for said message in a table, said entry including said partition, said time-stamp, and a state;

selecting messages for transmission based on said partition;

periodically identifying table data having a time-stamp later than a predetermined time;

writing said identified table data to a first replication database; and

periodically writing said identified table data from said first database to a second database.

10. A computer processing method, comprising the steps of:

assigning a partition and time-stamp to a message;

creating an entry for said message in a table, said entry including said partition, said time-stamp, and a state;

selecting messages for transmission based on said partition;

periodically identifying table data having a time-stamp later than a predetermined time;

writing said identified table data to a first database associated with a first processing environment;

periodically writing said identified table data from said first database to a second database associated with a second processing environment;

determining at said secondary processing environment whether said time-stamps of any of said identified table data correspond to a predetermined time interval;

if said time-stamps of any of said identified data do not correspond to said predetermined time interval, initiating action to correct any problems in said primary processing environment.

11. A computer processing method, comprising the steps of:

assigning a partition and time-stamp to a message;

creating an entry for said message in a table, said entry including said partition, said time-stamp, and a state;

selecting messages for transmission based on said partition;

writing said identified table data to a first database associated with a first processing environment;

periodically writing said identified table data from said first database to a second database associated with a second processing environment;

determining at said secondary processing environment whether said time-stamps of any of said identified table data correspond to a predetermined time interval;

if said time-stamps of any of said table identified data do not correspond to said predetermined interval, determining whether said primary processing environment is on-line;

if said primary processing environment is not on-line, initializing processes in said secondary processing environment and taking over processing functions of said primary processing environment at said secondary environment.

12. A method according to claim 9, further comprising the steps of:

building an entry for a third table when said message is destined for a processing environment that cannot receive messages; and

setting the state for said message to a queued state in said second table.

13. A method according to claim 10, further comprising the steps of:

building an entry for a third table when said message is destined for a processing environment that cannot receive messages; and

setting the state for said message to a queued state in said second table.

14. A method according to claim 11, further comprising the steps of:

building an entry for a third table when said message is destined for a processing environment that cannot receive messages; and

setting the state for said message to a queued state in said second table.

FIG. 1

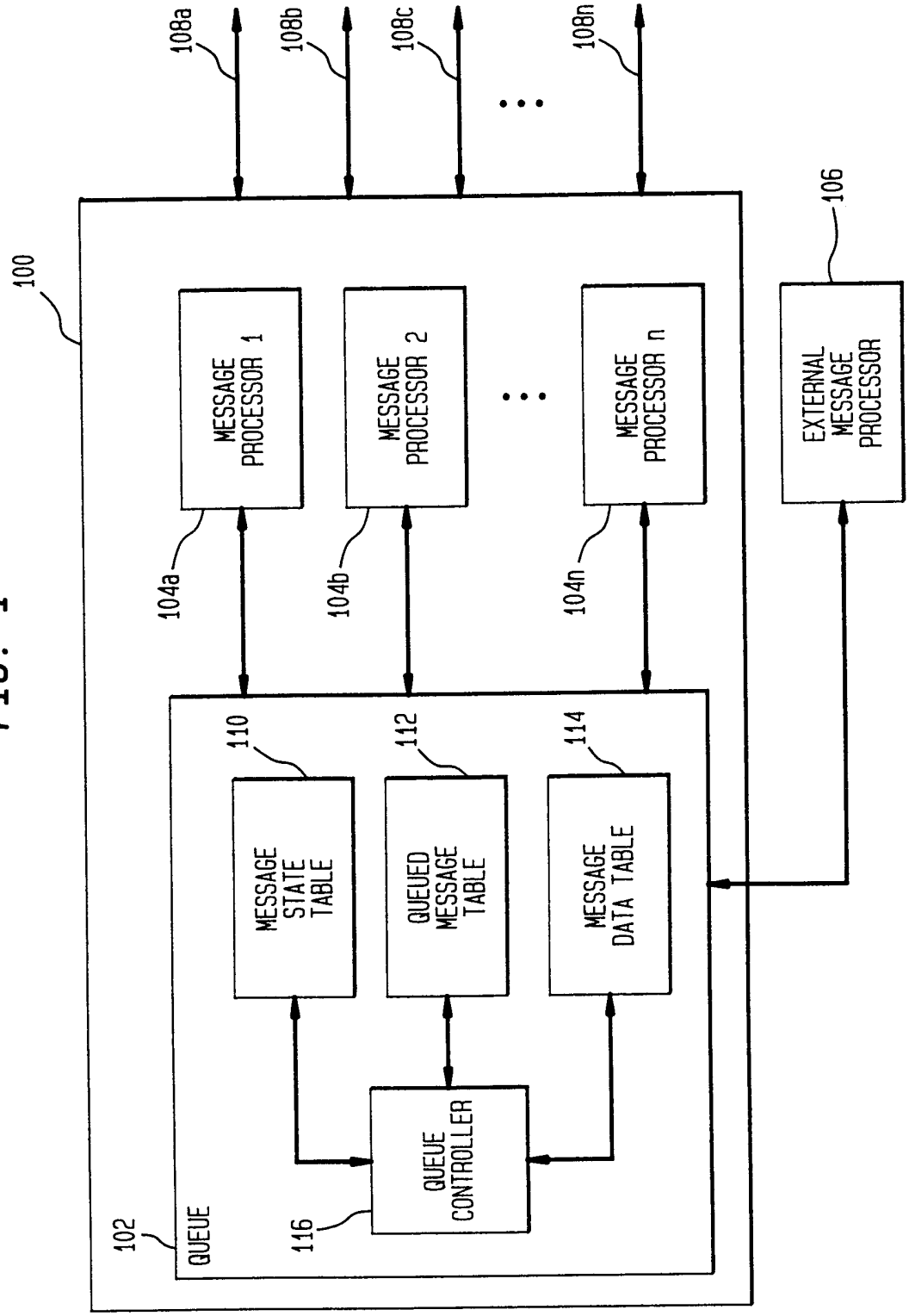

108a
108b
108c
108n
100
106
MESSAGE PROCESSOR 1
MESSAGE PROCESSOR 2
MESSAGE PROCESSOR n
EXTERNAL MESSAGE PROCESSOR
104a
104b
104n
110
112
114
MESSAGE STATE TABLE
QUEUED MESSAGE TABLE
MESSAGE DATA TABLE
QUEUE CONTROLLER
116
102
QUEUE

FIG. 2

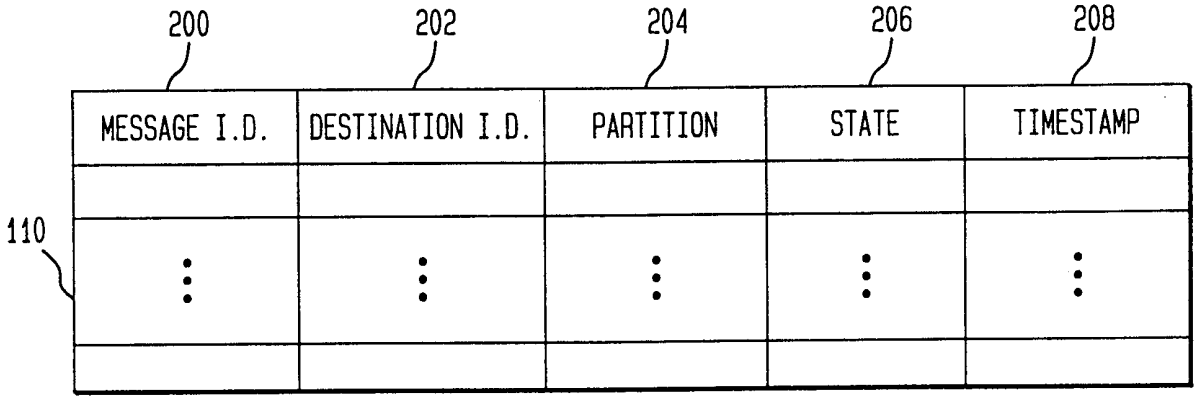


FIG. 3

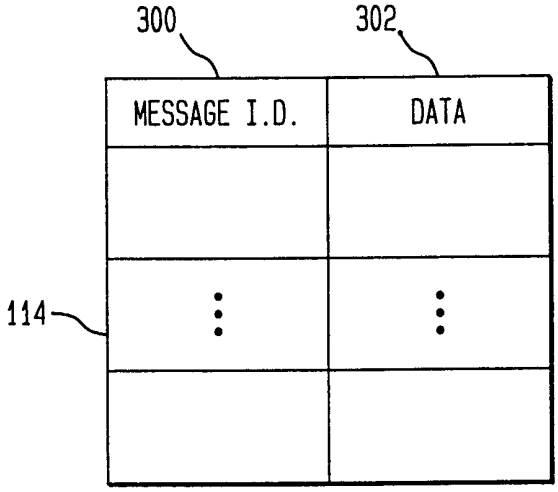


FIG. 4

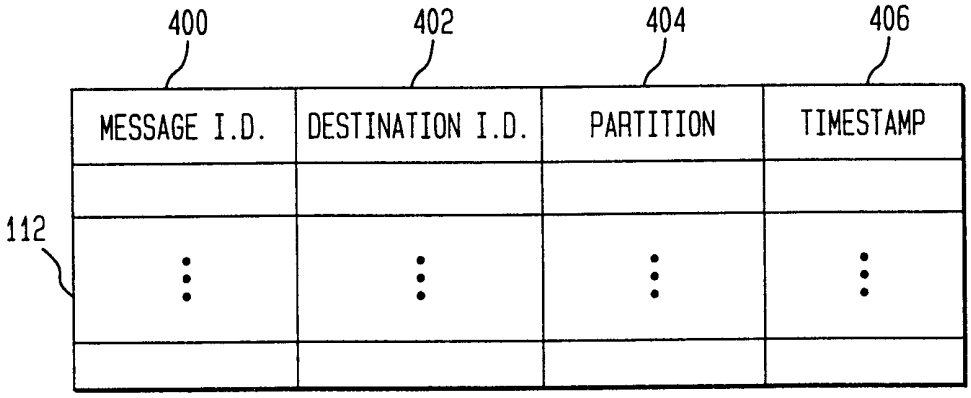


FIG. 5

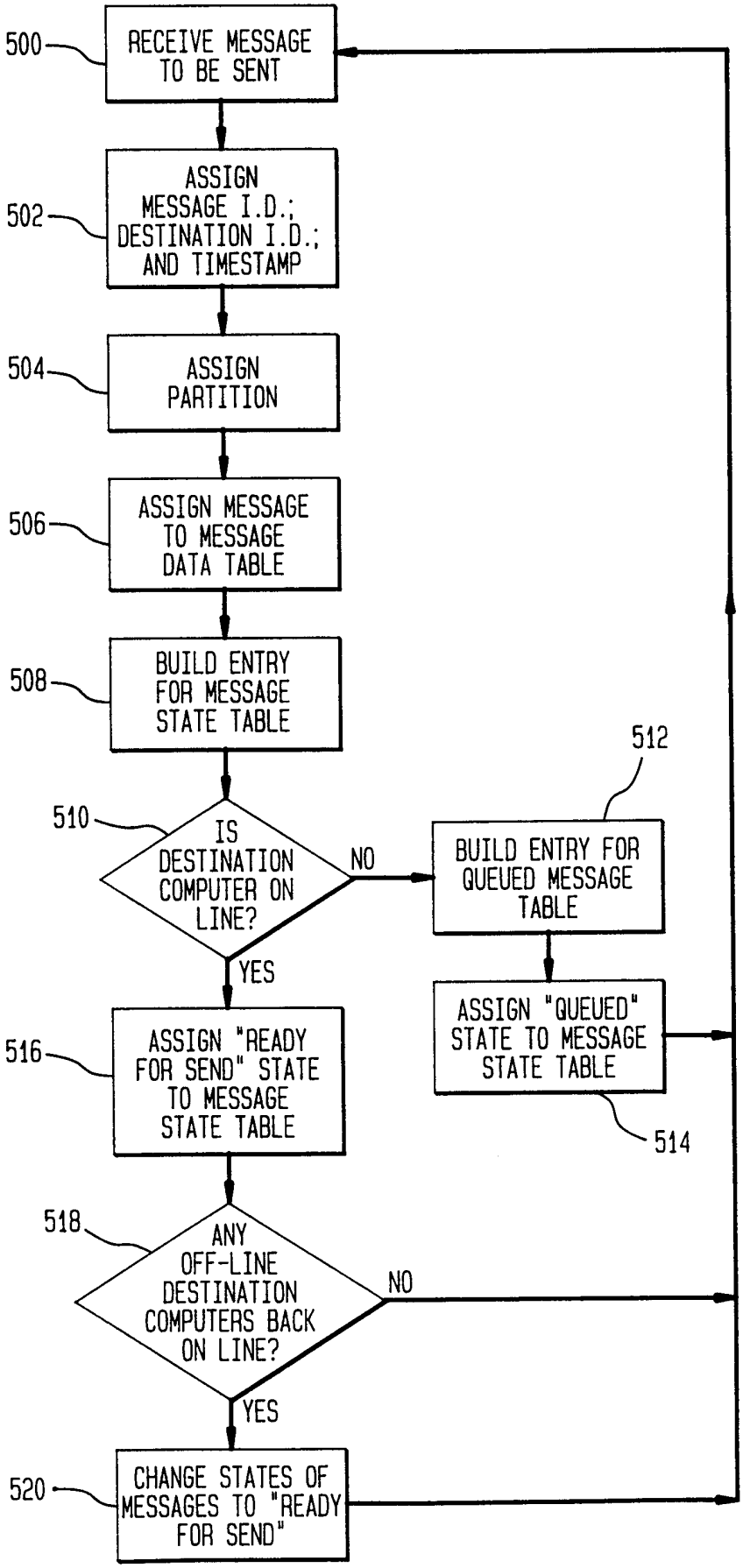

500
RECEIVE MESSAGE TO BE SENT
502
ASSIGN MESSAGE I.D.; DESTINATION I.D.; AND TIMESTAMP
504
ASSIGN PARTITION
506
ASSIGN MESSAGE TO MESSAGE DATA TABLE
508
BUILD ENTRY FOR MESSAGE STATE TABLE
510
IS DESTINATION COMPUTER ON LINE?
NO
YES
512
BUILD ENTRY FOR QUEUED MESSAGE TABLE
ASSIGN "QUEUED" STATE TO MESSAGE STATE TABLE
514
516
ASSIGN "READY FOR SEND" STATE TO MESSAGE STATE TABLE
518
ANY OFF-LINE DESTINATION COMPUTERS BACK ON LINE?
NO
YES
520
CHANGE STATES OF MESSAGES TO "READY FOR SEND"

FIG. 6

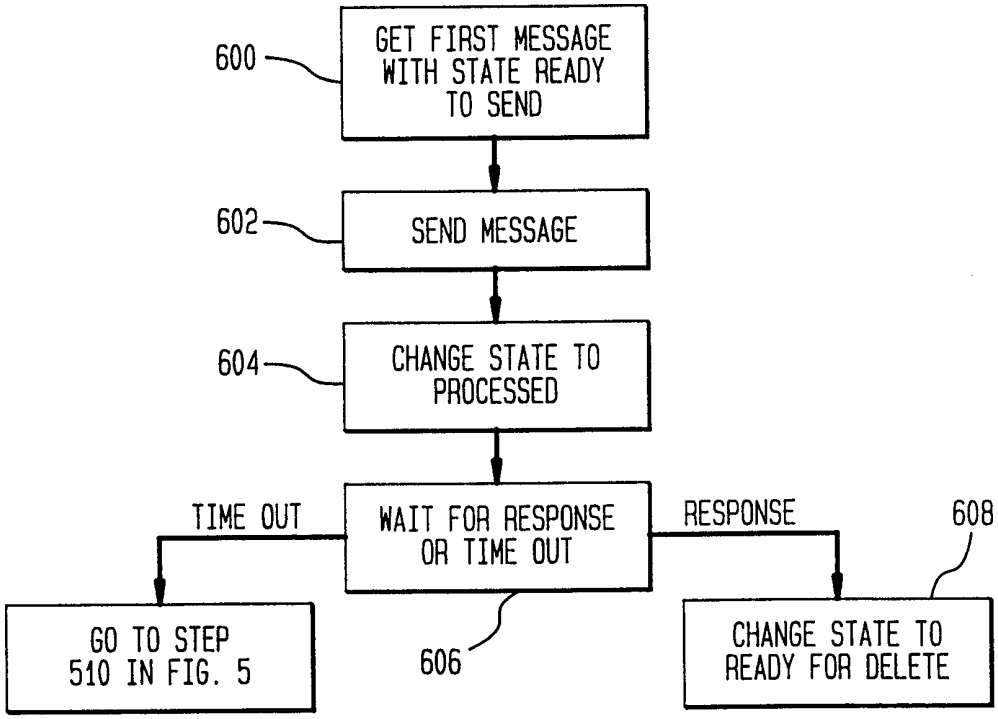

600
GET FIRST MESSAGE WITH STATE READY TO SEND
602
SEND MESSAGE
604
CHANGE STATE TO PROCESSED
TIME OUT
WAIT FOR RESPONSE OR TIME OUT
RESPONSE
608
GO TO STEP 510 IN FIG. 5
606
CHANGE STATE TO READY FOR DELETE

FIG. 7

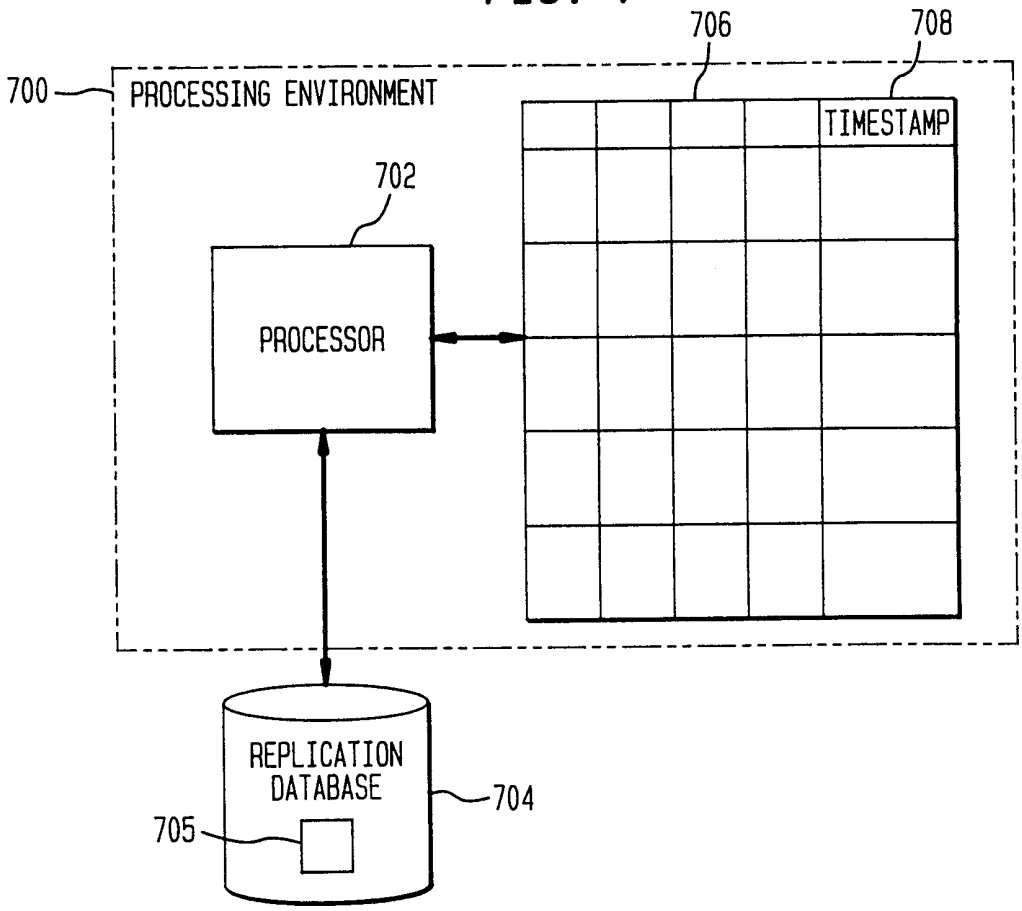

706
708
700
PROCESSING ENVIRONMENT
TIMESTAMP
702
PROCESSOR
REPLICATION DATABASE
704
705

FIG. 8

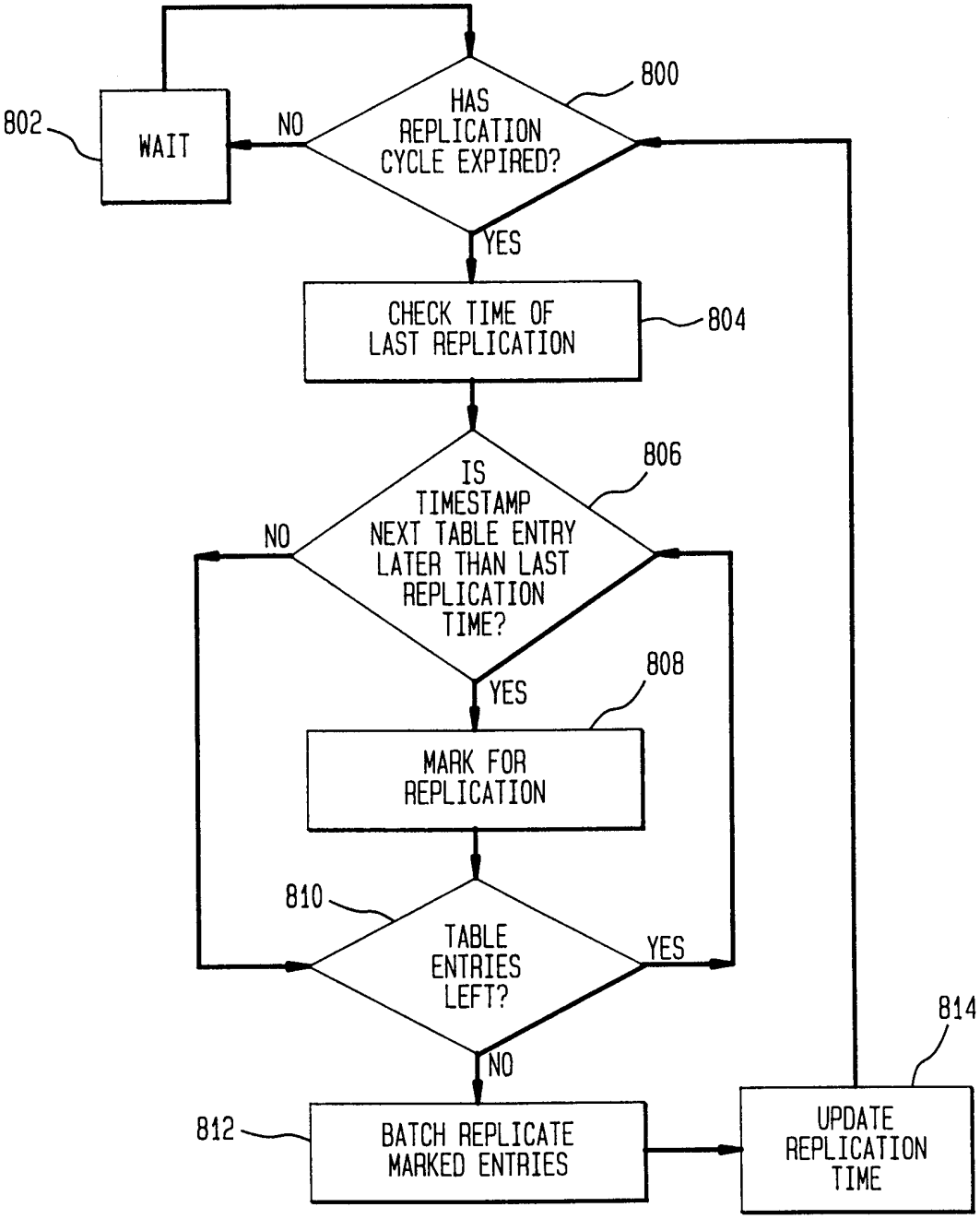

800
HAS REPLICATION CYCLE EXPIRED?
NO
802
WAIT
YES
804
CHECK TIME OF LAST REPLICATION
806
IS TIMESTAMP NEXT TABLE ENTRY LATER THAN LAST REPLICATION TIME?
NO
YES
808
MARK FOR REPLICATION
810
TABLE ENTRIES LEFT?
YES
NO
812
BATCH REPLICATE MARKED ENTRIES
814
UPDATE REPLICATION TIME

FIG. 9

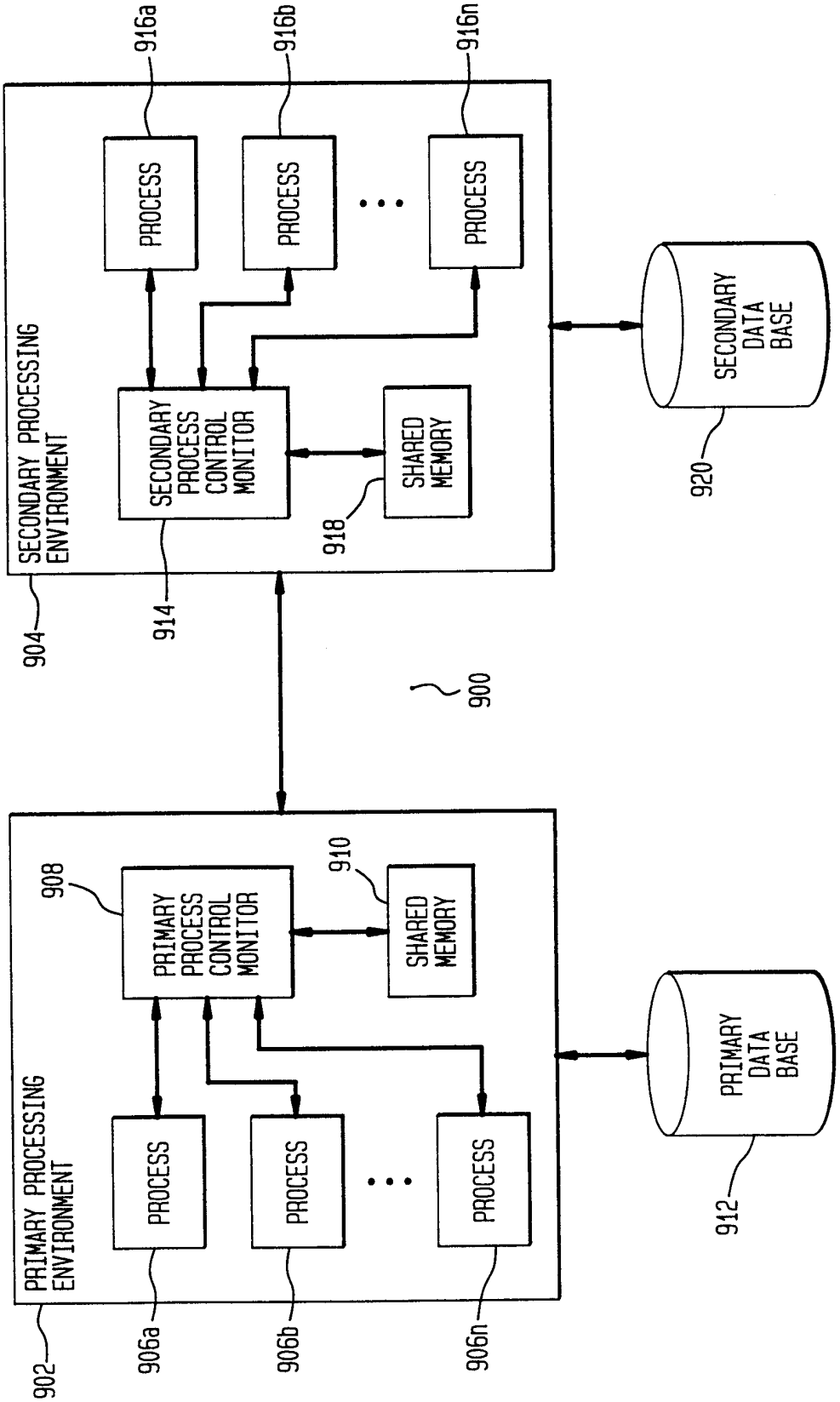

916a
916b
916n
PROCESS
PROCESS
PROCESS
SECONDARY PROCESSING ENVIRONMENT
SECONDARY PROCESS CONTROL MONITOR
SHARED MEMORY
SECONDARY DATA BASE
904
914
918
920
900
908
910
PRIMARY PROCESS CONTROL MONITOR
SHARED MEMORY
PRIMARY PROCESSING ENVIRONMENT
PROCESS
PROCESS
PROCESS
PRIMARY DATA BASE
902
906a
906b
906n
912

FIG. 10

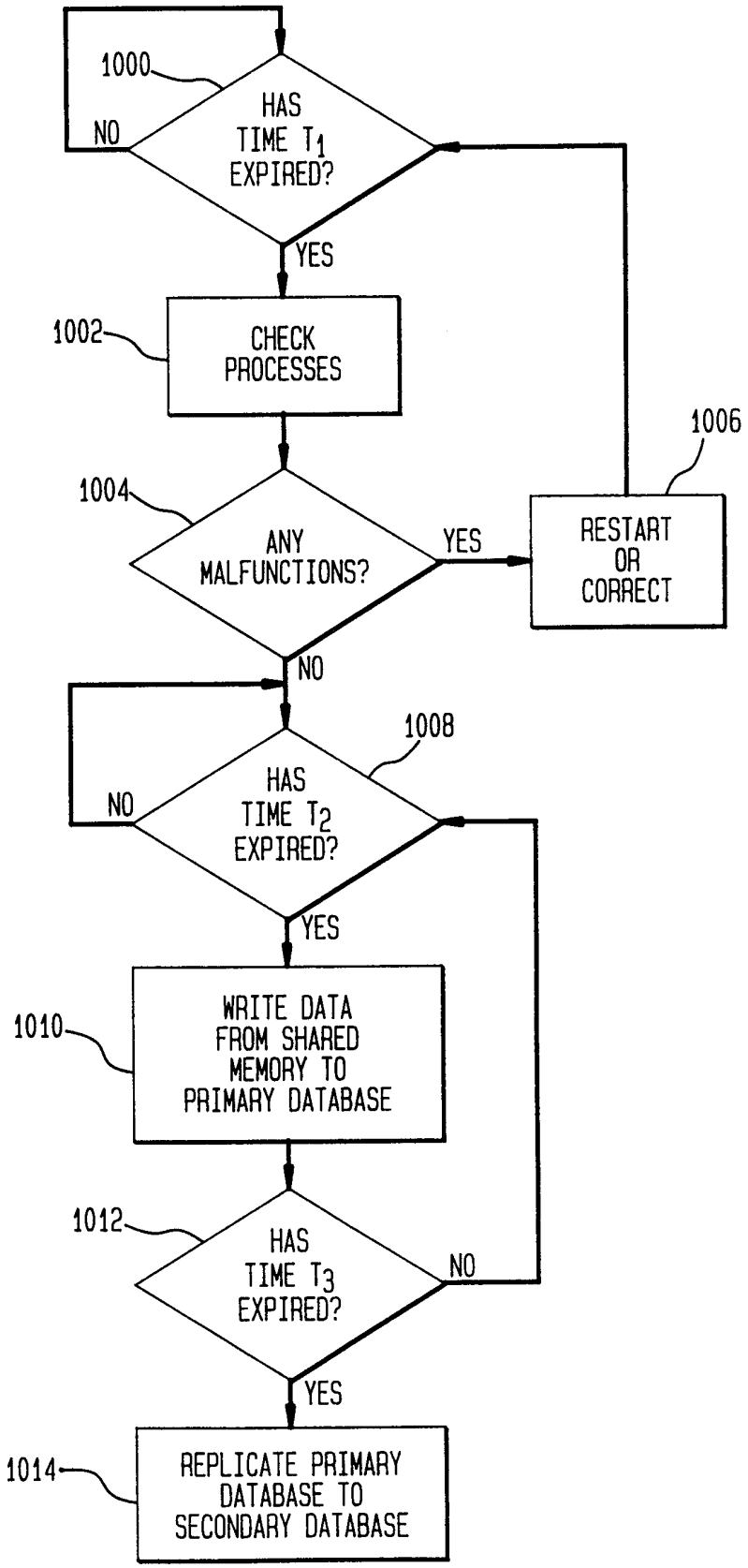
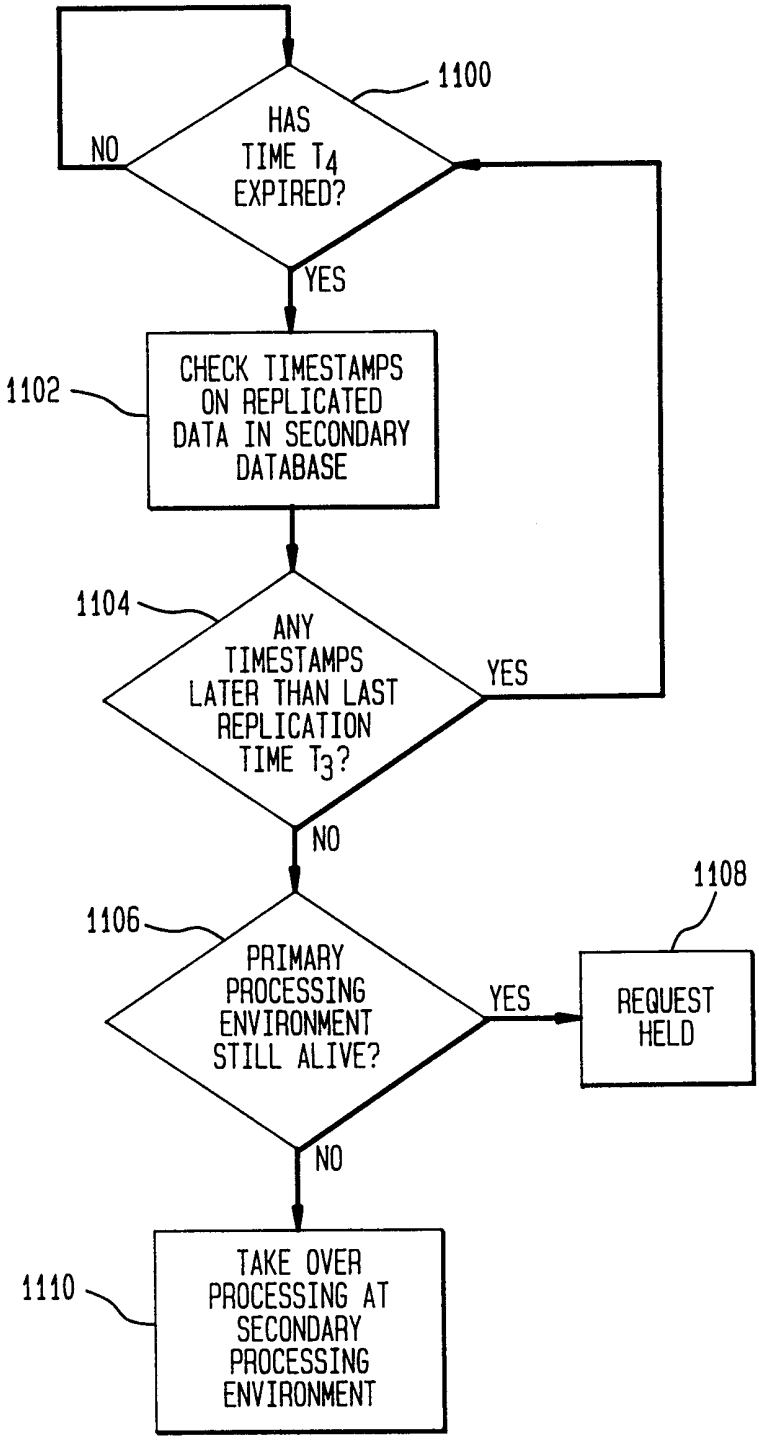

1000
HAS TIME T_1 EXPIRED?
NO
YES
1002
CHECK PROCESSES
1004
ANY MALFUNCTIONS?
YES
1006
RESTART OR CORRECT
NO
1008
HAS TIME T_2 EXPIRED?
NO
YES
1010
WRITE DATA FROM SHARED MEMORY TO PRIMARY DATABASE
1012
HAS TIME T_3 EXPIRED?
NO
YES
1014
REPLICATE PRIMARY DATABASE TO SECONDARY DATABASE

FIG. 11


1100
HAS TIME T_4 EXPIRED?
NO
YES
1102
CHECK TIMESTAMPS ON REPLICATED DATA IN SECONDARY DATABASE
1104
ANY TIMESTAMPS LATER THAN LAST REPLICATION TIME T_3?
YES
NO
1106
PRIMARY PROCESSING ENVIRONMENT STILL ALIVE?
YES
1108
REQUEST HELD
NO
1110
TAKE OVER PROCESSING AT SECONDARY PROCESSING ENVIRONMENT

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/15860

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 7/00; H04K 1/00; G06F 13/00; H04M 3/58
US CL :Please See Extra Sheet.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/249; 395/680; 380/25; 395/680; 395/200.68; 395/675

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,608,720 A [BIEGEL et al.] 10 June 1994 (10.06.94), col. 16, lines 14-16, col. 17, lines 29-48, col. 20, lines 12-16, col. 20, lines 28-50, col. 29, lines 65-67, col. 52, lines 45-52, col. 17, lines 29-48, col. 56, lines 20-41.	1-14

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 06 OCTOBER 1998	Date of mailing of the international search report 17 NOV 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer Chris Tanner Diane Smith for Telephone No. (703) 308-0000

Form PCT/ISA/210 (second sheet)(July 1992)★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/15860

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

370/249; 395/680; 380/25; 395/680; 395/200.68; 395/675; 370/412, 394

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

APS, EIC CD TOWER "Computer Select"
search terms:

(REPLICATION DATABASE) AND TIMESTAMP#
(SECONDARY DATABASE) AND TIMESTAMP#
REPLICATION DATABASE
TIMESTAMP
PROCESS MONITOR
MESSAGE DATABASE
MESSAGE QUEUE

(MESSAGE QUEUE) AND (MESSAGE STATE) AND (MESSAGE PARTITION)
(TIMESTAMP OR (DATE (2A) TIME))