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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 June 2013 (13.06.2013)

W I P O | P C T

(10) International Publication Number
WO 2013/083183 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *H04W 40/24* (2009.01)
H04L 12/56 (2006.01)

(21) International Application Number:
PCT/EP20 11/07 1907

(22) International Filing Date:
6 December 201 1 (06.12.201 1)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US):* **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building, Bantian Longgang, Shenzhen, Guangdong 5 18129 (CN).

(72) Inventor; and
(75) Inventor/Applicant *(for US only):* **RECHES, Shlomo** [IL/DE]; c/o Huawei Technologies Duesseldorf GmbH, Dessauerstrasse 3, 80992 Munich (DE).

(74) Agent: **KREUZ, Georg M.;** Huawei Technologies Duesseldorf GmbH, Dessauerstrasse 3, 80992 Munchen (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

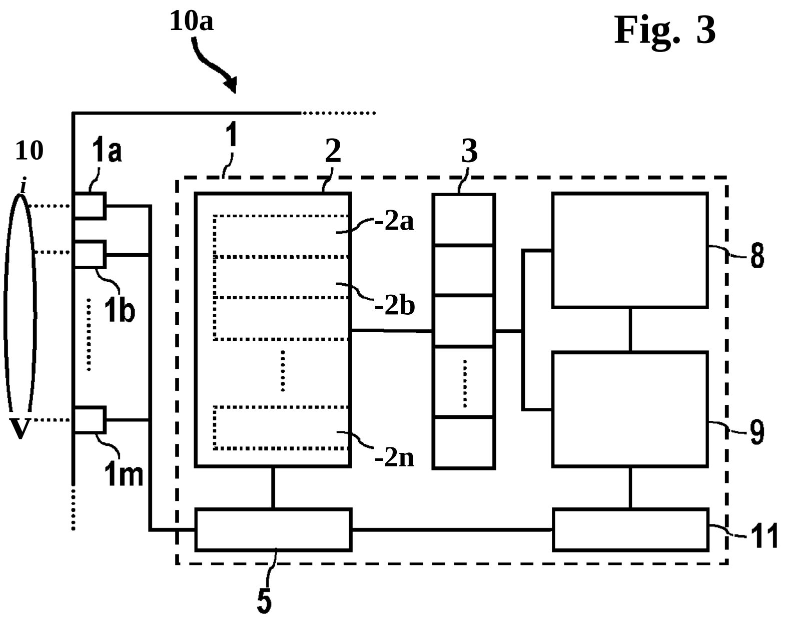
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: DATABASE WITH AGING MECHANISM AND METHOD FOR MANAGING A DATABASE

Fig. 3



(57) **Abstract:** Methods for managing a database and a respective database system are disclosed. A method for managing a database comprises determining a time interval which corresponds to the maximum time span a database entry is to remain in the database before being removed, dividing the time interval into at least a first and a second time slice, and updating at least a first and a second slice database with database events. After a time span corresponding to the duration of the first time slice has elapsed, the method comprises replacing the first slice database with the second slice database and creating a new second slice database as copy of the previous second slice database.

WO 2013/083183 A1

TITLE

Database with aging mechanism and method for managing a database

TECHNICAL FIELD

The present invention pertains to a database with aging mechanism and a method for managing a database. In particular, the invention pertains to dynamic databases in network devices, such as routers, switches, bridges, network traffic managers and the like.

TECHNICAL BACKGROUND

Databases that are dynamically updated and that may only occupy a limited amount of memory space need to be continuously checked for obsolete or old entries. A clean-up, i.e. a deletion or overwriting of these entries helps to maintain the database in a functional state. Such a clean-up can be generally referred to as "aging" of a database, i.e. the removal of older database entries. Aging may for example be performed by means of data collection jobs. Removal of this data reduces the sizes of databases. Aging events usually involve massive amounts of calculation, especially when hierarchical aging effects have to be considered, i.e. effects of a deletion of an entry that cause the need to update other database entries further up or down a database hierarchy.

Document US 6,125,371 A discloses methods for aging a database by assigning time stamps to each of the database entries, creating multiple versions of a database entry that is subject to an update transaction, monitoring the aging time of each of the versions of each of the database entries by comparing the time stamps, and deleting one or more versions of database entries in order to free up memory space, depending on the monitoring of the aging times.

Fig. 1 shows a schematical illustration of an exemplary database 100 that is organized as a hierarchical database with a dataset root 101, several branches 102a, 102b, 102c and corresponding leaves 103a, 103b, 103c, 103d, 103e and 103f. For example, the branch 102a is a first tier hierarchy level entry which the second tier hierarchy level entries 103a, 103b and 103c are dependent upon. The database 100 of Fig. 1 is exemplarily shown with two hierarchy levels, although the number of levels and the respective number of entries per level is not limited in any way. Furthermore, there may be intralevel interconnections 104 between entries of the same tier, which is exemplarily shown for branches 102b and 102c.

When a specific entry at one of the leaves is to be aged, i.e. deleted or over-written, the respective entries of the corresponding branch will have to be checked for any pointers or references to the leaf to be aged, for example timestamps, entry counters or similar values. For example, when the leaf 103d is to be aged, the entry 102b will have to be checked for consistency with the state of the entry after the deletion of the leaf 103d. Moreover, it might be necessary to also check the entry 102c of the database 100, since the entries 102b and 102c are interdependent by way of the interconnection 104. In each of the entries 102b and 102c, several re-calculations will have to be performed in order to age the leaf 103d without leaving a trace in the updated database 100, otherwise severe database corruption might occur during the ongoing operation of the database 100.

For large databases with lots of entries and a large amount of entries to be aged during the operation, these calculations will use up a lot of time and resources. Thus, there is a need for a database managing method and a respective database, with which the amount of updating calculations is reduced when aging entries of the database.

SUMMARY OF THE INVENTION

An idea of the present invention is to divide an aging period, i.e. the maximum time span an entry of a database may reside in the database without being aged, into a number n of time slices. For each of the n time slices, a separate slice database is maintained in parallel. Each of the n slice databases reflects the respective state changes of the database in slice periods that span different fractions of an aging period, ranging from the current time back in time to different fractions of the aging period. Events that change the database are evaluated with respect to the fraction of the aging period in which database contents are affected by the event. Accordingly, only those k out of n slice databases are updated with the event that do not reach further back than the fraction of the aging period that was evaluated.

Advantageously, with this procedure the oldest slice database, i.e. the one that spans over the whole aging period at that moment, may be aged after 1/n-th of an aging period has passed, since all changes that were caused by events in the following (n-1)/n-th of the aging period have been reflected in the remaining n-1 slice databases already. On the other hand, all events that are older than one aging period only affected the oldest slice database to be aged, hence preventing the need to re-calculate any of the remaining n-1 slice databases which were not affected by these oldest events at all. In other words, by maintaining a number of n slice databases in parallel with the aforementioned method, the need for checking and re-calculating entries of a database upon aging of entries is completely circumvented.

Consequently, a first aspect of the present invention relates to a method for managing a database, comprising determining a time interval which corresponds to the maximum time span a database entry is to remain in the database before being removed, dividing the time interval into at least a first and a second time slice, and updating at least a first and a second slice database with database events. After a time span corresponding to the duration of the first time slice has elapsed, the method comprises replacing the first slice database with the second slice database and creating a new second slice database as copy of the previous second slice database.

According to a first implementation of the first aspect the method further comprises dividing a database storage into at least a first and a second storage area, and updating the first and second storage area with database events, and, after the time span corresponding to the duration of the first time slice has elapsed, declaring the second storage area as the first storage area and overwriting the previous first storage area with database entries of a newly declared second storage area.

According to a second implementation of the first aspect the method further comprises dividing a database processing queue into at least a first and a second partial queue, the first partial queue including queue elements that are to expire after the elapse of the first time slice, and the second partial queue including queue elements that are not to expire after the elapse of the first time slice, creating a first queue pointer that points to the first element of the first partial queue, creating a second queue pointer that points to the first element of the second partial queue, and after the time span corresponding to the duration of the first time slice has elapsed, replacing the first queue pointer with the second queue pointer and replacing the second queue pointer with a new second queue pointer database.

According to a third implementation of the second implementation of the first aspect the first and second queue pointer point to the head of the first and second partial queue, respectively.

According to a fourth implementation of the second implementation of the first aspect the first and second queue pointer point to the tail of the first and second partial queue, respectively.

According to a fifth implementation of the first aspect a working access to the second slice database is denied, before the time span corresponding to the duration of the first time slice has elapsed.

According to a sixth implementation of the first aspect the database is a database for a network device. According to a seventh implementation of the first aspect the time interval is the round trip delay time in the network of the network device.

According to an eighth implementation of the first aspect, on the occurrence of a database event, either updating the first and the second slice database with the event in parallel, if the database event does not pertain to database entries that are to expire after the elapse of the first time slice, or updating only the second slice database with the event, if the database event pertains to database entries that are to expire after the elapse of the first time slice is performed.

According to a second aspect of the present invention, a database system comprises a database storage configured to store database entries of a main database, at least a first and a second slice database as copies of the main database, a database update controller configured to determine a time interval which corresponds to the maximum time span a database entry is to remain in the database before being removed, to divide the time interval into at least a first and a second time slice, and to update at least a first and a second slice database with database events, and a database aging controller configured to replace the first slice database with the second slice database and to replace the second slice database with a new second slice database after a time span corresponding to the duration of the first time slice has elapsed.

According to a first implementation of the second aspect, the database storage is divided into at least a first and a second storage area, the database update controller is configured to update the first and second storage area with database events, and the database aging controller is configured to declare the second storage area as the first storage area and to cause the previous first storage area to be overwritten with database entries of a newly declared second storage area after the time span corresponding to the duration of the first time slice has elapsed.

According to a second implementation of the second aspect, the database system further comprises a database processing queue, which is divided into at least a first and a second partial queue, the first partial queue including queue elements that are to expire after the elapse of the first time slice, and the second partial queue including queue elements that are not to expire after the elapse of the first time slice, and a database queue pointer controller which is configured to create a first queue pointer that points to the first element of the first partial queue, a second queue pointer that points to the first element of the second partial queue, and which is configured to replace the first queue pointer with the second queue pointer and to replace the previous second queue pointer with a new second queue pointer after the time span corresponding to the duration of the first time slice has elapsed.

According to a third implementation of the second implementation of the second aspect, the first and second queue pointer point to the head of the first and second partial queue, respectively.

According to a fourth implementation of the second implementation of the second aspect, the first and second queue pointer point to the tail of the first and second partial queue, respectively.

According to a fifth implementation of the second aspect, the database system is configured to deny a working access to the second slice database, before the time span corresponding to the duration of the first time slice has elapsed.

According to a sixth implementation of the second aspect, on the occurrence of a database event, the database updating controller is configured to either update the first and the second slice database with the event in parallel, if the database event does not pertain to database entries that are to expire after the elapse of the first time slice, or to update only the second slice database with the event, if the database event pertains to database entries that are to expire after the elapse of the first time slice, According to a third aspect of the invention, a network device comprises a database system according to the second aspect of the invention, wherein the database is a database for the network device, and wherein the time interval is the round trip delay time in the network of the network device.

According to a first implementation of the third aspect the network device is a network router. According to a second implementation of the third aspect the network device is a network switch. According to a third implementation of the third aspect the network device is a network gateway. According to a fourth implementation of the third aspect the network device is a network bridge. According to a fifth implementation of the third aspect the network device is a network traffic manager.

BRIEF DESCRIPTION OF THE FIGURES

The accompanying figures are included to provide a further understanding of the disclosure. They illustrate embodiments and may help to explain the principles of the invention in conjunction with the description. Other embodiments and many of the intended advantages, envisaged principles and functionalities will be appreciated as they become better understood by reference to the detailed description as following hereinbelow. The elements of the figures are not necessarily drawn to scale relative to each other. In general, like reference numerals designate corresponding similar parts.

Fig. 1	schematically illustrates a hierarchically organized database.
Fig. 2	schematically illustrates a hierarchically organized database in a network router according to an embodiment.
Fig. 3	schematically illustrates a database according to an embodiment.
Fig. 4	schematically illustrates a procedure for updating slice databases of a database of Fig. 3 according to an embodiment.
Fig. 5	schematically illustrates a procedure for updating queue pointers of a database of Fig. 3 according to an embodiment.

Fig. 6 schematically illustrates a procedure for updating a database storage of a database of Fig. 3 according to an embodiment.

Fig. 7 schematically illustrates a method for managing a database of a database of Fig. 3 according to an embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS

In the following detailed description, reference is made to the accompanying drawings, and in which, by way of illustration, specific embodiments are shown. It should be obvious that other embodiments may be utilized and structural or logical changes may be made without departing from the scope of the present invention. Unless specifically noted otherwise, functions, principles and details of each embodiment may be combined with other embodiments. Generally, this application is intended to cover any adaptations or variations of the specific embodiments discussed herein. Hence, the following detailed description is not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims.

Embodiments may include methods and processes that may be embodied within machine readable instructions provided by a machine readable medium, the machine readable medium including, but not being limited to devices, apparatuses, mechanisms or systems being able to store information which may be accessible to a machine such as a computer, a calculating device, a processing unit, a networking device, a portable computer, a microprocessor or the like. The machine readable medium may include volatile or non-volatile media as well as propagated signals of any form such as electrical signals, digital signals, logical signals, optical signals, acoustical signals, acousto-optical signals or the like, the media being capable of conveying information to a machine.

In the following, reference is made to methods and method steps, which are schematically and exemplarily illustrated in flow charts and block diagrams. It should be understood that the methods described in conjunction with those illustrative drawings may easily be performed by embodiments of systems, apparatuses and/or devices as well. In particular, it should be obvious that the systems, apparatuses and/or devices capable of performing the detailed block diagrams and/or flow charts are not necessarily limited to the systems, apparatuses and/or devices shown and detailed herein below, but may rather be different systems, apparatuses and/or devices. The terms "first", "second", "third", etc. are used merely as labels, and are not intended to impose numerical requirements on their objects or to establish a certain ranking of importance of their objects.

Fig. 2 schematically illustrates a hierarchically organized database 200 in a network router according to an embodiment. The network router may include a traffic manager that manages data packets arriving at ingress ports of the network router and being forwarded to different destinations via a number of egress ports. The traffic manager may update the respective database twice per packet - once, when the packet is received, i.e. is put in a packet queue, and again, when the packet is finally transmitted, i.e. is dequeued from the packet queue. The state at each of the database elements may for example hold the total number of bytes in the queue, a leaky bucket level of the queue that reflects the credit of this queue to send data and possibly other information about the packets being processed.

The database 200 may for example have a root node 201 that indicates the specific network router, port entries 202a, 202b, 202c that indicate the ingress and/or egress ports of the network router, subscriber groups 203a, 203b, 203c that are associated with a respective port 202a, subscribers 204a, 204b, 204c that are members of a specific subscriber group 203a and packet classes 205a, 205b that a specific subscriber 204a is associated with. The database 200 is organized in a hierarchical structure with branches, subbranches and leaves that each include information about specific entries in the hierarchy at lower tiers. A database update event is triggered, when a packet is enqueued or dequeued, beginning at a leaf of the lowest tier, for example a packet class leaf 205a, 205b. The event propagates through the database tree, updating respective database entries in the parent nodes.

When a packet in the database 200 is to be aged, that is, to be removed from the queue if a maximum time interval that the packet is allowed to remain in the router is exceeded, any updating events in the respective database hierarchy that have been made with regard to this packet have to be reverted to the original state. With (typically) millions of packets to be aged the amount of database operations is considerable, if only one working instance of the database 200 is used.

Instead, a database system 1 as schematically illustrated in Fig. 3 may be used. The database system 1 may be a database for a network device 10a which is part of a network 10. The network device 10a may be a router, a switch, a gateway, a bridge, a network traffic manager or a similar network component. The network component 10a may have a number of ingress/egress ports 1a, 1b, ... , 1m through which data packets may be received and transmitted.

The database system 1 may include a database storage 2, a slice database storage 3, a database update controller 8, a database aging controller 9, a database processing queue 5 and a database queue pointer controller 11. The database storage 2 may be configured to store database entries of a main database, for example data packets and any database hierarchy components associated with the respective data packets, as exemplarily depicted in Fig. 2. To this end, the database storage 2 may be divided in a plurality of storage areas 2a, 2b, ... , 2n the number of which may be adapted to the respective database system 1.

The main database in the database storage 2 may be subject to aging, that is, after a time interval T which corresponds to the maximum time span a database entry is to remain in the database, respective data entries are removed from the database. For example, in the case of a network device 10a, the time interval T may be the round trip delay time or round trip time (RTT) which indicates the maximum time period which elapses between the transmittal of a data packet to any network node of the network 10 and the reception of a response data packet at the network device 10a.

The database update controller 8 may be coupled to the database aging controller 9 which may in turn be coupled to the database queue pointer controller 11. The database queue pointer controller 11 may be coupled to the database processing queue 5. The database processing queue 5 may be coupled to the database storage 2. The database storage 2 may be coupled to the slice database storage 3, which in turn may be coupled to the database update controller 8 and the a database aging controller 9.

The database update controller 8 may be be configured to control updating a database, for example slice databases held within the slice database storage 3. The database update controller 8 may perform database update events of the database system 1. The database aging controller 9 may be configured to control database aging events of the database system 1. The database queue pointer controller 11 may be configured to create queue pointers.

The function of the database system 1 of Fig. 3 will now be explained with respect to the schematically illustrated procedures of Figs. 4, 5 and 6. In each of the Figs. 4 to 6, a time line is shown. Initially, the database system 1 is assumed to be in a state at a time t+T, wherein t is the point in time furthest in the past of which information about database events still exists in the database and T is the determined time interval corresponding to the aging period of the database system 1. As time elapses, the database system 1 eventually arrives at a point in time t+T+T1, wherein T1 is the time span after which an aging event takes place. The aging event of the database system 1 may be controlled by the database aging controller 9. During the time span T1, the database update controller 8 may perform database update events of the database system 1 according to the procedures described in the following.

In the procedure depicted in Fig. 4, the aging period or time interval T, respectively, may be divided into a number n of time slices t1, t2, ... , tn. Exemplarily, a number n of 4 time slices is shown in Figs. 4 to 6, however, any number or time slices greater than one may be selected for dividing the time interval T. The time slices t1, t2, ... , tn may in one embodiment be of equal duration, however, it may be possible to vary the duration of the time slices t1, t2, ... , tn according to any duration pattern that might be useful for the aging procedure of the database system 1.

Associated with the time slices t1, t2, ..., tn a number n of slice databases 3a, 3b, ..., 3n may be created as copies of the main database. With other words, the slice databases each contains the whole database, with the difference between the slice databases being different update statuses associated with the time slices t1, t2, ..., tn. These slice databases 3a, 3b, ... , 3n may be maintained in parallel in a slice database storage 3. That means that the slice databases 3a, 3b, ... , 3n may get updated by the database update controller 8 at the same time during the operation of the database system 1 according to specific updating rules. One of the slice databases 3a, 3b, ..., 3n may be the working database, that is, any working access to the database system 1 will be directed to the working database and the remaining slice databases will be hidden from database inquiries. For example, in Fig. 4 the slice database 3a may be the working database.

The slice databases 3a, 3b, ... , 3n are ordered in their sequence of expiration, which means that the leftmost slice database 3a will be the first to expire and the rightmost slice database 3n the last to expire. The procedure of Fig. 4 starts at the point in time t+T. If a database event occurs, for example the reception of a new data packet or the transmission of a data packet in the database processing queue 5, the database event is checked what data packet the event pertains to. If the database event spans over all the time slices t1, t2, ... , tn, then it is determined that the event pertains to data packets that were already in the database system during the first time slice t1. These data packets are the oldest data packets and are therefore the first to expire after a time span T1 has elapsed that corresponds to the duration of the first time slice t1. Therefore, it is assumed that information relating to this database event is only relevant for the oldest version of the database, which is the first slice database 3a. Only the first slice database 3a gets updated by the database update controller 8, as indicated by the arrows with the continuous line.

On the other hand, if a database event is determined to only relate to data packets that have been included in the database in a time slice t2, ... , tn later than the first time slice t1, respective other slice databases 3b, ..., 3n get updated as well. For example, a database event determined to relate to data packets having been received in the second time slice t2, the slice databases 3a and 3b get updated. In other words, the later data packets a database event is associated with will expire, the more slice databases 3a, 3b, ..., 3n will get updated simultaneously.

After the time span T1 has elapsed, the database aging controller 9 may be configured to perform an aging event, that is to remove any obsolete data packets and database entries that relate to the respective data packets. In order to do so, the database aging controller 9 may replace the first slice database 3a with the second slice database 3b, the second slice database 3b with the third slice database, and so on, until the n-th slice database 3n replaces the (n-1)-th slice database 3n-1. The database aging controller 9 may then be configured to create a new n-th slice database as the current state of the database, for example as the previous n-th slice database 3n.

Since updates of database events that are to be aged have only been performed with respect to the first slice database 3a, the remaining slice databases 3b, ... , 3n already reflect the state of the database without any update events pertaining to the aged database entries. Therefore, no re-calculations of database entries will have to be performed. The aging of the database is merely done by evaluating database events with regard to their association to data packets and their time of expiration. By updating only select copies of the database, i.e. only certain slice databases 3a, 3b, ..., 3n, the respectively oldest slice database is already in the right state for becoming the new working database. By creating new n-th slice database 3n as copies of the current updating state, the slice databases are shifted with respect to their relative position to the working database. Since after the elapse of every time slice, the database aging controller 9 creates a new n-th slice database 3n as a fresh slice database, only database events that occur after this time slice will affect the fresh slice database. Therefore, the database aging controller 9 constantly creates staggered slice databases with different updating statuses in order to replace the previous working database with a fresh one in an aging event.

In addition to the slice databases 3a, 3b, ... , 3n the data packet queue 5 may be updated in a similar way, as will be explained with regard to the schematically illustrated procedure in Fig. 5. The database processing queue 5 may be divided into a number n of partial queues 5a, 5b, ... , 5n, wherein the first partial queue 5a includes queue elements 6i that are to expire after the elapse of the first time slice t1, and the n-th partial queue 5n includes queue elements 6i that are only to expire after the elapse of the n-th time slice tn. In other words, the first partial queue 5a is the oldest queue, wherein the n-th partial queue is the youngest queue. The database queue pointer controller 11 may be configured to create a first queue pointer 7a that points to the first element of the first partial queue 5a, a second queue pointer 7b that points to the first element of the second partial queue 5b and so on. The queue pointers 7a, 7b, ... , 7n may be prongs of a group pointer 7. The group pointer 7 may for example point to the respective heads of the queues or the tails of the queues. After the time span T1 corresponding to the duration of the first time slice t1 has elapsed the first queue pointer 7a may be replaced with the second queue pointer 7b, the second queue pointer 7b with the third queue pointer, and so on. Last, a new n-th queue pointer is created, that points to the new head or tail of the new n-th partial queue starting with the first data packet to arrive after the point in time t+T+T1. The group pointer 7 may be used to rotate the queue pointers 7, 7b, ... , 7n, that is the replaced first queue pointer 7a will become the new n-th queue pointer 7n upon aging the packet queue 5.

A procedure to manage the database storage 2 is schematically illustrated with respect to Fig. 6. The database storage 2 may be configured to store a database 4 having database entries 4i. In order to manage the database storage 2 directly with the actual data packets or indirectly through pointers to the actual data packets the database storage 2 may be divided into a plurality of storage areas 2a, 2b, ... , 2n, wherein the database update controller 8 is configured to store new database entries 4i that are to expire after the elapse of the first time slice t1 in the first storage area 2a, to store new database entries 4i that are to expire after the elapse of the second time slice t2 in the second storage area 2a, and so on, until data packets that will not expire until after the elapse of the n-th time slice tn are stored in the n-th storage area. By assigning a fixed buffer size to the respective storage areas 2a, 2b, ... , 2n, it may be possible to group up the database entries 4i according to their time of expiration. Therefore, when aging the database as the point in time t+T+T1, the writing pointer will simply skip to the next oldest storage area and the respective memory will be freed automatically for the next data packets or pointers which point to addresses for the data packets to arrive. Since the aged storage area only held data packets that expired after the point in time t+T+T1 , no rearrangement of database entries 4i in the database storage 2 will become necessary. The database aging controller may simply be configured to declare the second storage area 2b as the first storage area 2a, and the n-th storage area 2n as the (n-1)-th storage area 2n-1.

The storage areas 2a to 2n may be cycled continuously through the time slices, so that after the elapse of each subsequent time slice the aged storage area may be replaced with a fresh one that only holds entries not to be aged yet at that specific point in time.

Fig. 7 schematically illustrates a method 20 for managing a database, for example a database 4 in a database system 1, which database system 1 is shown in Fig. 3. The method 20 for managing a database comprises a first step 21 of determining a time interval T which corresponds to the maximum time span a database entry 4i is to remain in the database 4 before being removed. In a second step 22, the time interval T may be divided into at least a first and a second time slice t1 and t2, respectively. Concurrently, in a step 32, a database storage 2 may be divided into at least a first and a second storage area 2a, 2b, ... , 2n and updated with data events, and in a step 42, a database processing queue 5 may be divided into at least a first and a second partial queue 5a, 5b, ... , 5n, the first partial queue 5a including queue elements 6i that are to expire after the elapse of the first time slice t1, and the second partial queue 5b including queue elements 6i that are not to expire after the elapse of the first time slice t1.

In a step 23, at least a first and a second slice database 3a, 3b, ... , 3n may be updated with database events. Simultaneously, in concomitant steps 43a and 43b, a first queue pointer 7a that points to the first element of the first partial queue 5a, and a second queue pointer 7b that points to the first element of the second partial queue 5b may be created.

When setting up the database system with the first and the second slice database, during a first time slice after initializing the database system, if a database event occurs, it may be determined what database entries 4i the database event pertains to. In a step 24a, the first and the second slice database 3a, 3b, ... , 3n may be updated with the event in parallel, if the database event does not pertain to database entries 4i that are to expire after the elapse of the first time slice t1. Similarly, new database entries 4i that are to expire after the elapse of the first time slice t1 may be stored in the first storage area 2a. If on the other hand, the database event pertains to database entries 4i that are to expire after the elapse of the first time slice t1, only the second slice database 3b may be updated with the event. Accordingly, new database entries 4i that are not to expire after the elapse of the first time slice t1 may be stored in the second storage area 2b.

After a time span T1 corresponding to the duration of the first time slice t1 has elapsed, an aging event may take place. In an aging step 25, the first slice database 3a may be replaced with the second slice database 3b and replace the second slice database 3b with a new second slice database. Simultaneously, in an aging step 35, the second storage area 2b may be declared as the first storage area 2a and the previous first storage area 2a may be overwritten with database entries of a newly declared second storage area. Finally, in an aging step 45, the first queue pointer 7a may be replaced with the second queue pointer 7b and the previous second queue pointer 7b may be replaced with a new second queue pointer.

The steps 21, 22, 23 and 25 of the method 20 may then be repeated for further time slices, so that the slice databases are shifted with respect to their relative position to the working database. Since after the elapse of every time slice, a new slice database is created as youngest slice database, only database events that occur after this time slice will affect this youngest slice database. Therefore, staggered slice databases with different updating statuses are created constantly in order to replace the previous working database with a fresh one in an aging event. By initializing a newly created slice database to a certain point in time, the need for re-calculations after each aging event is completely circumvented.

The advantage of the invention is that foldbacks of database updating events are not necessary, since slice databases, queue pointers and slice database storage areas are kept in parallel with staggered updating statuses.

With this database management system, as explained in conjunction with Figs. 3, 4, 5 and 6 and in conjunction with the method of Fig. 7, the need for time and resource consuming foldbacks of database updating events is completely obviated by maintaining a plurality of databases in parallel. Although more bandwidth will most likely have to be used, the advantages by being able to revert a database to an up-to-date state with virtually no calculation effort will in most cases outweigh this disadvantage. The methods and database system may be implemented in network devices, as shown in the exemplary embodiments, however, any other type of technical equipment that makes use of aging databases may be used equally.

Claims

1. A method (20) for managing a database (4), comprising:
determining (21) a time interval (T) which corresponds to the maximum time span a database entry (4i) is to remain in the database (4) before being removed;
dividing (22) the time interval (T) into at least a first and a second time slice (t1, t2, ... , tn);
updating (23) at least a first and a second slice database (3a, 3b, ... , 3n) with database events; and
after a time span (T1) corresponding to the duration of the first time slice (t1) has elapsed, replacing (25) the first slice database (3a) with the second slice database (3b) and replacing the second slice database with a new second slice database (3b).

2. The method (20) of claim 1, further comprising:
dividing (32) a database storage (2) into at least a first and a second storage area (2a, 2b, …, 2n) and updating the first and second storage area (2a, 2b, ... , 2n) with database events; and
after the time span (T1) corresponding to the duration of the first time slice (t1) has elapsed, declaring (35) the second storage area (2b) as the first storage area (2a) and overwriting the previous first storage area (2a) with database entries of a newly declared second storage area.

3. The method (20) of one of the claims 1 and 2, further comprising:
dividing (42) a database processing queue (5) into at least a first and a second partial queue (5a, 5b, …, 5n), the first partial queue (5a) including queue elements (6i) that are to expire after the elapse of the first time slice (t1), and the second partial queue (5b) including queue elements (6i) that are not to expire after the elapse of the first time slice (t1); and
after the time span (T1) corresponding to the duration of the first time slice (t1) has elapsed, replacing (45) the first queue pointer (7a) with the second queue pointer (7b) and creating a new second queue pointer as copy of the previous second queue pointer (7b).

4. The method (20) of claim 3, wherein the first and second queue pointer (7a, 7b) point to the head of the first and second partial queue (5a, 5b, ... , 5n), respectively.

5. The method (20) of claim 3, wherein the first and second queue pointer (7a, 7b) point to the tail of the first and second partial queue (5a, 5b, …, 5n), respectively.

6. The method (20) of one of the claims 1 to 5, wherein a working access to the second slice database (3b) is denied, before the time span (T1) corresponding to the duration of the first time slice (t1) has elapsed.

7. The method (20) of one of the claims 1 to 6, wherein the database (4) is a database for a network device (10a), and wherein the time interval (T) is the round trip delay time in the network (10) of the network device (10a).

8. A database system (1), comprising:
a database storage (2) configured to store database entries (4i) of a main database (4);
at least a first and a second slice database (3a, 3b, ... , 3n);
a database update controller (8) configured to determine a time interval (T) which corresponds to the maximum time span a database entry (4i) is to remain in the database (4) before being removed, to divide the time interval (T) into at least a first and a second time slice (t1; t2), and to update at least a first and a second slice database (3a, 3b, …, 3n) with database events; and
a database aging controller (9) configured to replace the first slice database (3a) with the second slice database (3b) and to replace the new second slice database with a new second slice database (3b) after a time span (T1) corresponding to the duration of the first time slice (t1) has elapsed.

9. The database system (1) of claim 8, wherein the database storage (2) is divided into at least a first and a second storage area (2a, 2b, ... , 2n),
wherein the database update controller (8) is configured to update the first and second storage area (2a, 2b, ... , 2n) with database events, and
wherein the database aging controller (9) is configured to declare the second storage area (2b) as the first storage area (2a) and to cause the previous first storage area (2a) to be overwritten with database entries of a newly declared second storage area after the time span (T1) corresponding to the duration of the first time slice (t1) has elapsed.

10. The database system (1) of one of the claims 8 and 9, further comprising:
a database processing queue (5), which is divided into at least a first and a second partial queue (5a, 5b, …, 5n), the first partial queue (5a) including queue elements (6i) that

are to expire after the elapse of the first time slice (t1), and the second partial queue (2b) including queue elements (6i) that are not to expire after the elapse of the first time slice (t1); and

a database queue pointer controller (11) which is configured to create a first queue pointer (7a) that points to the first element of the first partial queue, a second queue pointer (7b) that points to the first element of the second partial queue (5b), and which is configured to replace the first queue pointer (7a) with the second queue pointer (7b) and to create a new second queue pointer as copy of the previous second queue pointer (7b) after the time span (T1) corresponding to the duration of the first time slice (t1) has elapsed.

11. The database system (1) of claim 10, wherein the first and second queue pointer (7a, 7b) point to the head of the first and second partial queue (5a, 5b, ..., 5n), respectively.

12. The database system (1) of claim 10, wherein the first and second queue pointer (7a, 7b) point to the tail of the first and second partial queue (5a, 5b, ... , 5n), respectively.

13. The database system (1) of one of the claims 8 to 12, wherein the database system (1) is configured to deny a working access to the second slice database (3b), before the time span (T1) corresponding to the duration of the first time slice (t1) has elapsed.

14. A network device (10a), comprising a database system (1) of one of the claims 8 to 13, wherein the database (4) is a database for the network device (10a), and wherein the time interval (T) is the round trip delay time in the network (10) of the network device (10a).

15. The network device (10a) of claim 14, wherein the network device (10a) is one of the group of a network router, a network switch, a network gateway, a network bridge or a network traffic manager.

Fig. 1

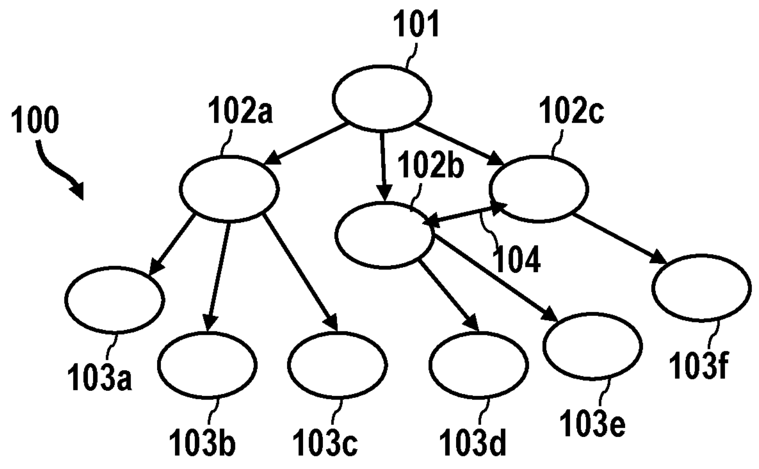


Fig. 2

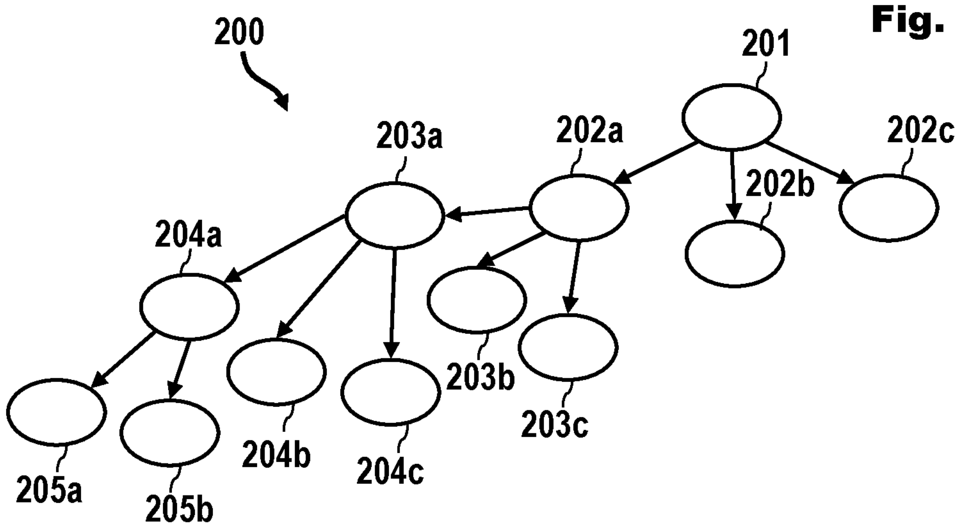


Fig. 3

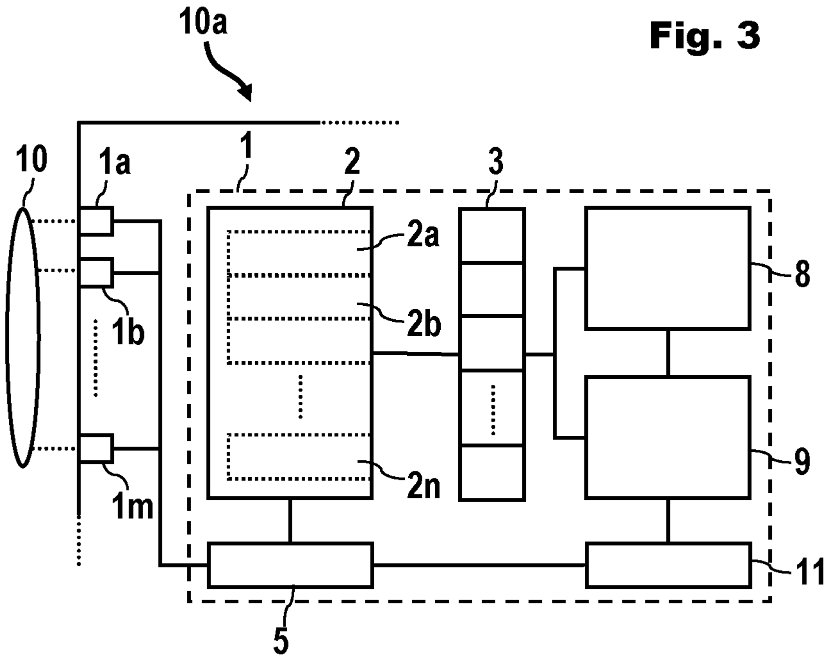


Fig. 4

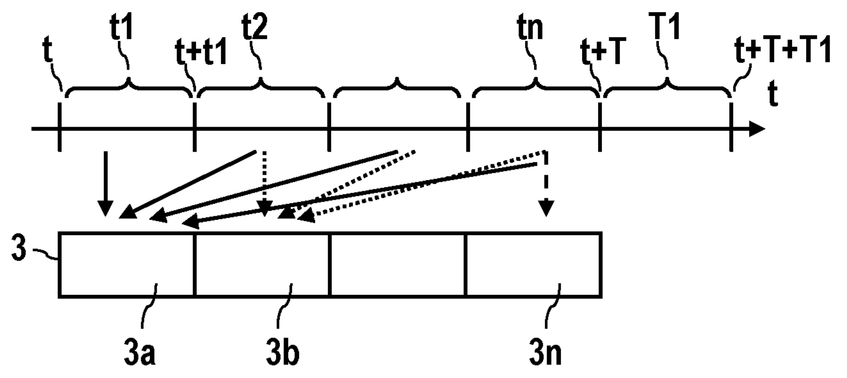


Fig. 5

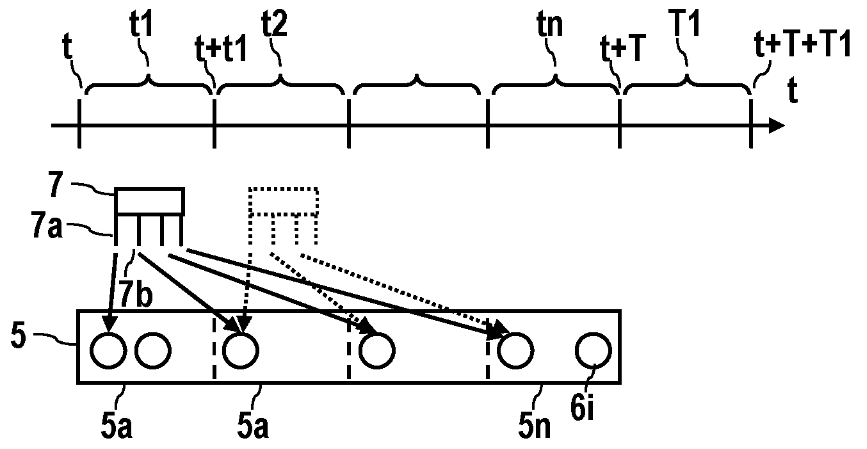


Fig. 6

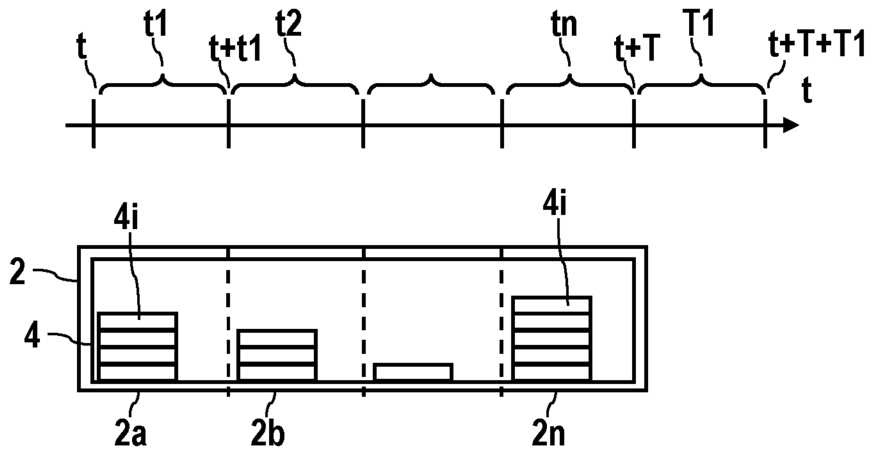
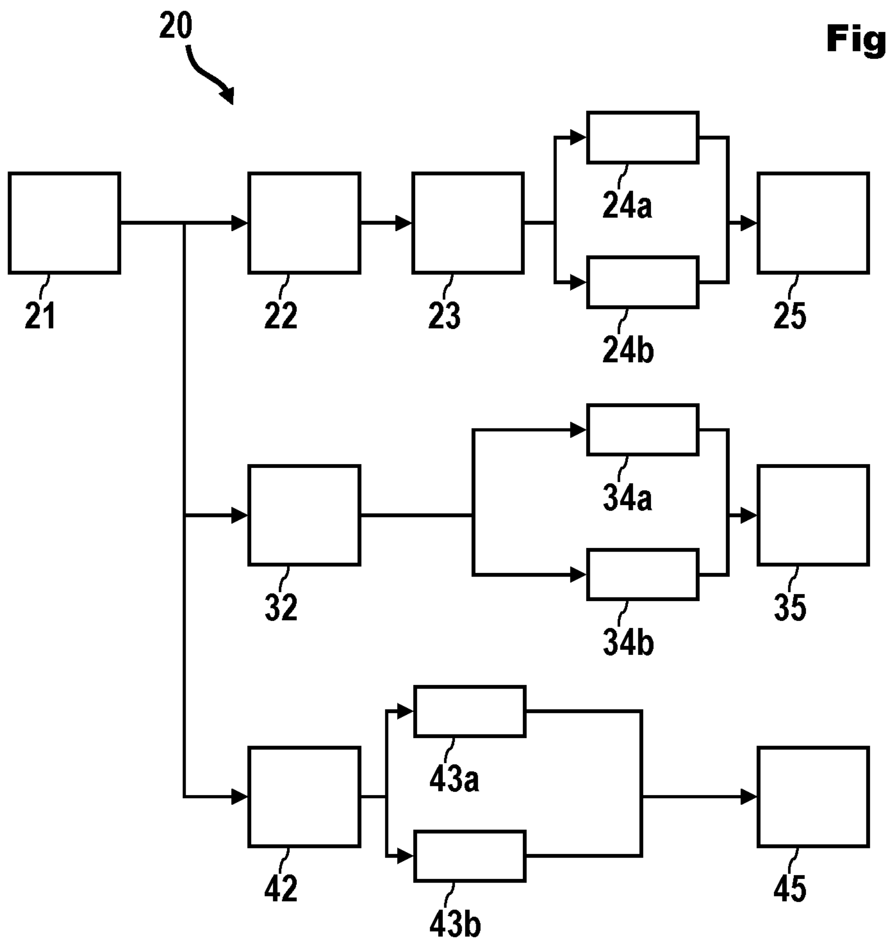


Fig. 7



INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2011/071907

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30 H04L12/56 H04W40/24
ADD.

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)
G06F H04L H04W

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)

EPO-Internal , WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 125 371 A (BOHANNON P L; LEINBAUGH D W; RASTOGI R; SESHADRI S; SILBERSCHATZ A; SU) 26 September 2000 (2000-09-26) cited in the application column 2, line 63 - column 3, line 14 -----	1-15
A	US 2008/222111 A1 (H0ANG CHI KIM [US] ET AL) 11 September 2008 (2008-09-11) paragraph [0051] - paragraph [0053] claims 8-10, 12 -----	1-15
A	US 2008/107052 A1 (V0GETY RAMANAG0PAL [US] ET AL) 8 May 2008 (2008-05-08) paragraph [0005]; claim 1 -----	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"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)
"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

"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
"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
"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
"&" document member of the same patent family

Date of the actual completion of the international search 20 July 2012	Date of mailing of the international search report 30/07/2012
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Haffner, Ronald

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2011/071907

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6125371 A	26-09-2000	NONE	
US 2008222111 A1	11-09-2008	US 2008222111 A1	11-09-2008
		US 2008222159 A1	11-09-2008
US 2008107052 A1	08-05-2008	NONE	