

**METHOD FOR UPGRADING NETWORK SERVER PROGRAMMING CONDITIONS,  
ASSOCIATED SYSTEM AND SOFTWARE PRODUCT**

**Technical Field**

The present invention relates to networks involving the  
5 use of the servers; more particularly, the invention relates  
to the function for updating the programming conditions for  
these servers, or in other words, the function currently  
indicated as "software upgrade". In particular, the method in  
accordance with the invention was developed with specific  
10 attention to potential application in servers which manage  
transactions and retain a state whose loss can cause  
permanent or temporary malfunctions.

**Background Art**

In cases where it is necessary to guarantee that a  
15 service be performed uninterruptedly or with a minimum number  
of interruptions, server software upgrades are especially  
delicate, as service unavailability may result.

In this connection, and in order to ensure a certain  
degree of fault tolerance, the commonly used solution is to  
20 replicate the servers. When an upgrading operation is  
performed, the server or servers involved in this operation  
are made temporarily unreachable, typically by eliminating  
databases whereby these servers can be located, for example  
by means of a Domain Name Server (DNS) mechanism, and  
25 transferring the load to the servers which remain active.

The main disadvantages of this solution are as follows:

- It is not always easy to determine when a server can be  
deactivated: it is necessary, in fact, to ensure that there  
are no transactions in progress, and that any local copies of  
30 the server lists in the units acting as clients do not  
contain references to the server in question;

- Another problem which is directly connected to this  
disadvantage is that administrative operations are always  
fairly slow, and in particularly urgent cases (e.g., when it

is necessary to deal with potential security problems), the only possibility is to interrupt the service temporarily.

- In addition, if the duration of the state maintained by the process exceeds that of a transaction and (as is frequently the case), the end of its valid lifetime cannot be readily identified, the state may be lost, causing temporarily malfunction, and

- Finally, if (as occurs in most cases) it is difficult to predict traffic trends and a traffic peak takes place during the upgrading operation, the servers which remain active may encounter difficulties in handling their workload: the only way to prevent this problem is to deploy at least one machine in addition to those which would be sufficient under normal circumstances, which entails an increase in system redundancy which would not be strictly necessary otherwise.

#### **Disclosure of the Invention**

The object of the present invention is thus to provide a solution capable of overcoming the problems outlined above.

In accordance with the present invention, this object is achieved by means of a method having the characteristics detailed in the following claims. The invention also relates to the associated system, or in other words to the system configured to operate on the basis of the aforesaid method, as well as to the corresponding software product, or in other words to the code which, loaded in a computer and run thereon, makes it possible to configure the computer in question so that it can operate on the basis of the aforesaid method.

Essentially, the solution in accordance with the invention involves "hot" server process replacement, proceeding in such a way that the process which is replaced coexists for a certain period of time with its replacement. The duration of this operation is decided by the replaced

process, thus guaranteeing that the operation will always take place after the end of the last transaction. Any state information whose lifetime is longer can be passed from one process to the other, for example by means of a socket or another bidirectional inter-process communication mechanism.

#### **Brief Description of Drawings**

The following description of the invention, which is intended purely by way of example and is not to be construed as limiting, will make reference to the accompanying drawings, where:

- Figure 1 is a functional block diagram illustrating the normal operation of a server operating in accordance with the invention,

- Figure 2 is a further functional block diagram using the same formalism as Figure 1 which illustrates how the server upgrading operation depicted in Figure 1 is performed, and

- Figure 3 is a flow chart illustrating how the operation depicted in Figure 2 is performed.

#### **Best mode for Carrying Out the Invention**

Figure 1 provides a schematic illustration of the general configuration of a server S. The latter may, for example, be a server used in a telecommunications network to perform the SIP (Session Initiation Protocol) proxy function.

For example, it may be an Enterprise server produced by Sun Microsystems of Palo Alto (CA), using the Solaris™ operating system.

The server S is capable of carrying out processing operations such as the processing operation designated in its entirety as 1.

Operation 1 comprises:

- A lower layer (or level - the two terms will be regarded hereunder as equivalent) 2 which interfaces directly

with the network in order to ensure packet reception/transmission, and

- An upper layer 3 which processes the packets, receiving them and passing them to the lower layer.

5       The two layers 2 and 3 are interfaced in a general bidirectional communication scheme as represented by the arrows.

10       As it is understood that all of the foregoing points correspond to configuration and application criteria which will be entirely familiar to persons skilled in the art, it is unnecessary to provide a detailed description here.

15       In updating the server's programming conditions (or in other words, in the so-called software upgrade operation), the current configuration, designated as 1, must be replaced with an updated configuration designated in its entirety as 1' and likewise organized into a lower layer 2' and an upper layer 3', which will substitute the corresponding layers 2 and 3 at the end of the updating or upgrade operation.

20       The method in accordance with the invention is intended to ensure that the operation of substituting or replacing the current process 1 with its updated version 1' is entirely transparent to the network, or in other words takes place in such a way that the clients with whom the server S is interfaced are unable in practice to detect that the  
25       upgrading operation has been started and that said operation is in progress.

30       All of this is accomplished while guaranteeing that the transactions in progress will be managed normally, even in cases where the duration of the state maintained by the process exceeds that of the transaction (which frequently makes it impossible to identify the end of the valid lifetime with any degree of ease), and avoiding any difficulties that could arise following an unforeseen traffic peak.

The updating operation first envisages that the process corresponding to the update or upgrade 1' connect to the current process 1 which must be replaced by means of a socket or other IPC mechanism designated in its entirety as 4, authenticating itself.

Naturally, the current process 1 which is to be replaced also checks that the upgrading operation is carried out by the same machine S.

The current process 1 which is to be replaced releases its ports (lower layer 2). These ports are immediately acquired by the corresponding layer 2' of the upgraded process 1', which is configured so that the packet flow it receives can be selectively diverted preparatory to forwarding it to the upper layer 3 of the current process 1 through the channel 4.

The packets arriving from the networks are then received exclusively by the lower layer 2' of the upgraded process 1'.

These packets are, in any case, also passed to the upper layer 3 of the process 1 which is then being replaced. The latter is thus put into a position to act, as it were, as a filter: if the messages concern new transactions (or in other words, transactions which started after the beginning of the upgrading operation), they are transferred directly - without being processed - to the corresponding upper layer 3' of the upgraded process 1'. Conversely, if the messages concern transactions which are already under way at the upper layer 3 of the process 1 which is being replaced, they continue to be processed by said upper layer 3.

At the same time, the two processes 1 and 1' can exchange state information which is not connected with transactions.

When the upper layer 3 of the process 1 which is being replaced determines that all of the transactions it manages have ended, a signal is generated to indicate that the upgrading operation has been completed. This signal,

transmitted on channel 4, ensures that the lower layer 2' of the upgraded process 1' sends no further packets or messages of any kind to the upper layer 3 of the process 1 which has been upgraded. The communication channel 4 is closed and,  
5 consequently, the upgraded process 1' has at this point replaced process 1 in all respects, and process 1 is deactivated.

The steps employed in carrying out the upgrading operation described above are illustrated in greater detail  
10 in the flow chart shown in Figure 3.

In this flow chart, the reference 100 designates the first step in the upgrading operation, which corresponds to opening the connection 4.

The reference 102 designates the step in which the  
15 current process 1 checks that upgrading is performed on the same machine.

When the outcome of this check is negative (outcome F), the upgrading operation is not followed through and is interrupted at step 104.

20 When the outcome of the check carried out at step 102 is positive (outcome V), the system ports (or in practice the interface with the network) are transferred - in step 106 - from the lower layer 2 of the current process 1 to the lower layer 2' of the upgraded process 1', with the subsequent  
25 transfer - carried out at step 108 - of the state information.

From this moment onwards, the operating condition is established in which the messages concerning transactions which are already under way continue to be handled by the  
30 current process 1, while the messages concerning new transactions are handled by the upgraded process 1'.

In particular, the reference 110 designates the step in which a new message is read, while reference 112 designates the step in which a check is carried out to determine whether

the packet that has just been read concerns a known transaction or a new transaction.

Where the message concerns a new transaction (outcome F of step 112), the packet is identified as regarding a "new" transaction and is thus handled by the upgraded process 1' in a step 114.

In cases where the message that has been read is identified as corresponding to a transaction which is already known (outcome V of step 112), it is handled by the current process 1 in a step 116.

Starting from both steps 114 and 116, the upgrading operation continues, converging towards a further step designated as 118, in which it determines whether the transactions for the process which is being replaced, or in other words the current process 1, are to be regarded as finished.

Should there still be transactions which have not yet been completed (outcome F of step 118), system operation returns upstream of step 110 again, given that a new packet is to be read.

Conversely, if the transactions that must be handled by the process 1 which is being replaced have been terminated (outcome V of step 118), the system goes on to a final step designated as 120, in which the communication channel 4 is closed, thus completing the upgrading operation.

Naturally, and without detriment to the invention's underlying principles, details and forms of implementation may vary widely with respect to the descriptions and illustrations provided herein, without for that reason failing to fall within the scope of the present invention.

For instance, possible variations, which are intended purely by way of example and are not to be construed as limiting in any respect, on the general approach described

above with reference to Figures 1 and 3 could include those in which:

- The communication mechanism identified by the channel 4 is not a socket: what is essential is that, in general, the channel permit bidirectional communications, and that it likewise make it possible to establish with certainty where the other process is being performed; and

- The initial mechanism whereby the lower layer 2 is substituted or replaced by the lower layer 2' is applied, in addition or as an alternative to the ports, to other dedicated resources used exclusively for the input/output functions.

Essentially, the method in accordance with the invention is applied in all situations in which the duration of the service interruption must be minimal (virtually nil) and it is not possible to identify a period in which the traffic handled is reasonably low.

**CLAIMS**

1. A method for upgrading the programming conditions of a network server (S) which performs at least one process (1), said process (1) comprising a lower layer or level (2) interfacing with network packets and an upper layer or level (3) for processing said packets, the method being characterized in that it comprises the steps of:

- Providing an updated process (1') comprising a respective lower layer (2') interfacing with network packets and a respective upper layer (3') for processing said packets; said respective lower layer (2') being associated with a communication channel (4) for diverting the network packets from said respective lower layer (2') to said upper layer (3),

- Replacing (106) said lower layer (2) with said respective lower layer (2'),

- Checking (112) whether the packets received from said respective lower layer (2') relate to transactions which are already under way on said upper layer (3), or to new transactions,

- Processing (114), in said upper layer (3), the packets relating to transactions which are already under way on the upper layer (3),

- Processing (116) the packets relating to new transactions in said respective upper layer (3'),

- Determining (118) whether the transactions which are already under way on said upper layer (3) have been completed, and

- Closing (120) said communication channel (4), completing the programming upgrade for server (S), when said transactions which are already under way on said upper layer (3) have been completed.

2. A method in accordance with claim 1, characterized in that it comprises the step of checking (102), with a view to

carrying out said programming upgrade, that said programming upgrade is carried out on a single server (S).

3. A method in accordance with claim 1 or claim 2, characterized in that said step of replacing said lower layer (2) with said respective lower layer (2') comprises the step of transferring the network ports of said lower layer (2) to said respective lower layer (2').

4. A method in accordance with any of the foregoing claims, characterized in that said step of checking (112) whether the packets received relate to transactions which are already under way on said upper layer (2) or to new transactions is carried out on said upper layer (3), said upper layer (3) being configured to forward the packets relating to new transactions to said respective upper layer (3') by means of said communication channel (4).

5. A method in accordance with any of the foregoing claims, characterized in that it comprises the step of configuring said communication channel (4) to exchange state information (108) between the process (1) which is to be upgraded and the upgraded process (1').

6. A method in accordance with any of the foregoing claims, characterized in that it comprises the step of configuring said communication channel (4) as a socket.

7. A network server configured to upgrade the programming conditions using the method in accordance with any of claims 1 through 6.

8. A server in accordance with claim 7, characterized in that said server is a Session Initiation Protocol (SIP) proxy.

9. A software product which can be loaded directly into a computer memory comprising portions of programming code to implement the method in accordance with any of claims 1 through 6 when run on a computer.