Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2459575 C 2012/07/24

(11)(21) 2 459 575

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2002/09/18
(87) Date publication PCT/PCT Publication Date: 2003/03/27
(45) Date de délivrance/Issue Date: 2012/07/24
(85) Entrée phase nationale/National Entry: 2004/03/03
(86) N° demande PCT/PCT Application No.: GB 2002/004234
(87) N° publication PCT/PCT Publication No.: 2003/026244
(30) Priorité/Priority: 2001/09/18 (GB0122507.7)

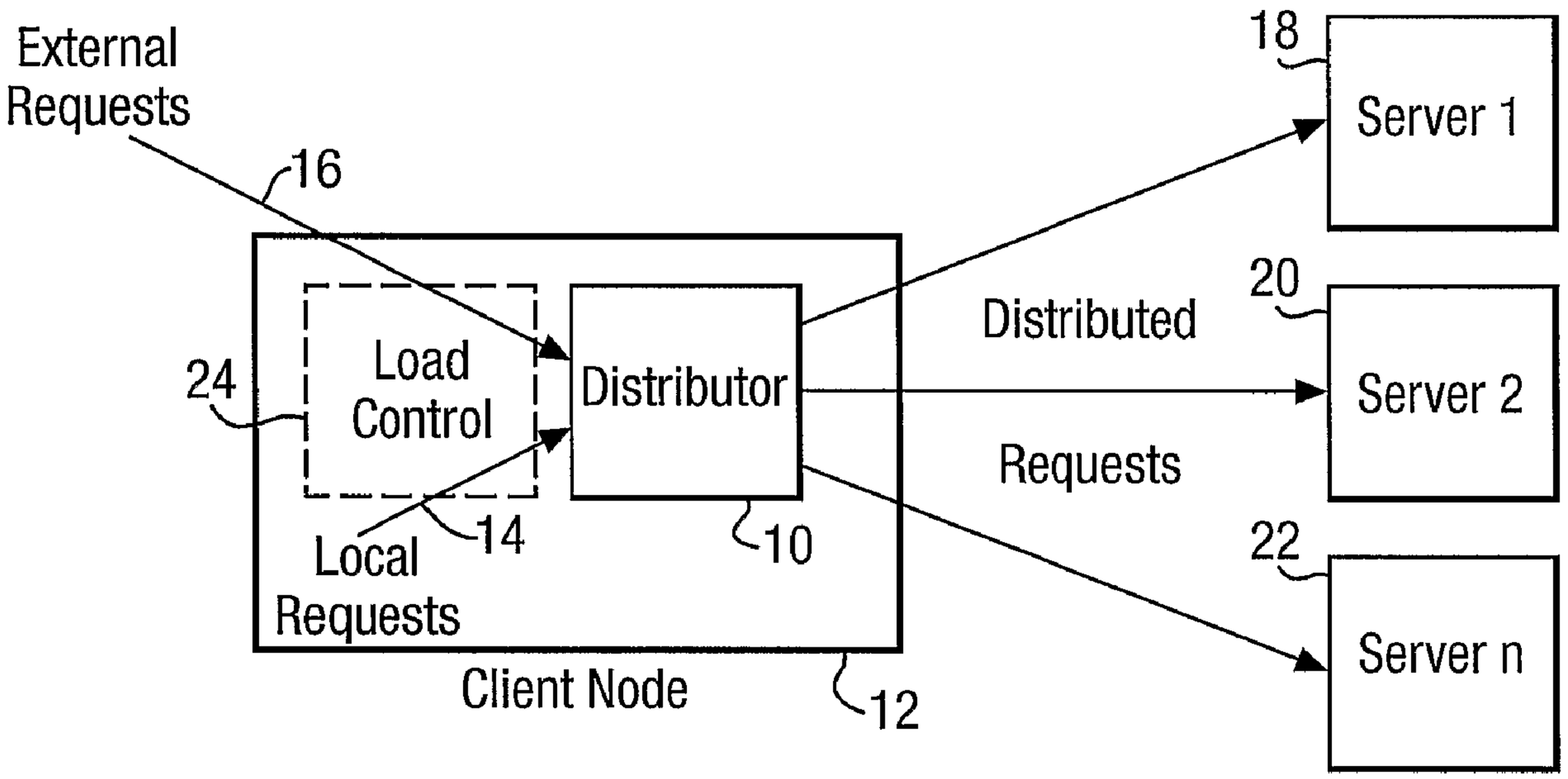
(51) Cl.Int./Int.Cl. *H04L 29/06* (2006.01),
G06F 9/50 (2006.01)

(72) Inventeur/Inventor:
MARTIN, STEPHEN IAN, GB

(73) Propriétaire/Owner:
ERICSSON AB, SE

(74) Agent: ERICSSON CANADA PATENT GROUP

(54) Titre : RESEAUX DE SERVEURS CLIENTS
(54) Title: CLIENT SERVER NETWORKS



(57) Abrégé/Abstract:
A client node in a network communicates with a server system comprising a plurality of servers. The client comprises a distributor which periodically measures server activity, assesses relative loading of the servers and adjusts the request distribution, and therefore the server loadings in accordance with the loading assessment.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 March 2003 (27.03.2003)

PCT

(10) International Publication Number
WO 03/026244 A3

(51) International Patent Classification[7]: H04L 29/06

(21) International Application Number: PCT/GB02/04234

(22) International Filing Date:
18 September 2002 (18.09.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0122507.7 18 September 2001 (18.09.2001) GB

(71) Applicant *(for all designated States except US)*: **MARCONI COMMUNICATIONS LIMITED** [GB/GB]; New Century Park, Coventry CV3 1HJ (GB).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **MARTIN, Stephen, Ian** [GB/GB]; 10 Nuthatch Close, Creekmore, Poole, Dorset BH17 7XR (GB).

(74) Agent: **CARDUS, Alan**; Marconi Intellectual Property, Marrable House, The Vineyards, Great Baddow, Chelmsford, Essex CM2 7QS (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

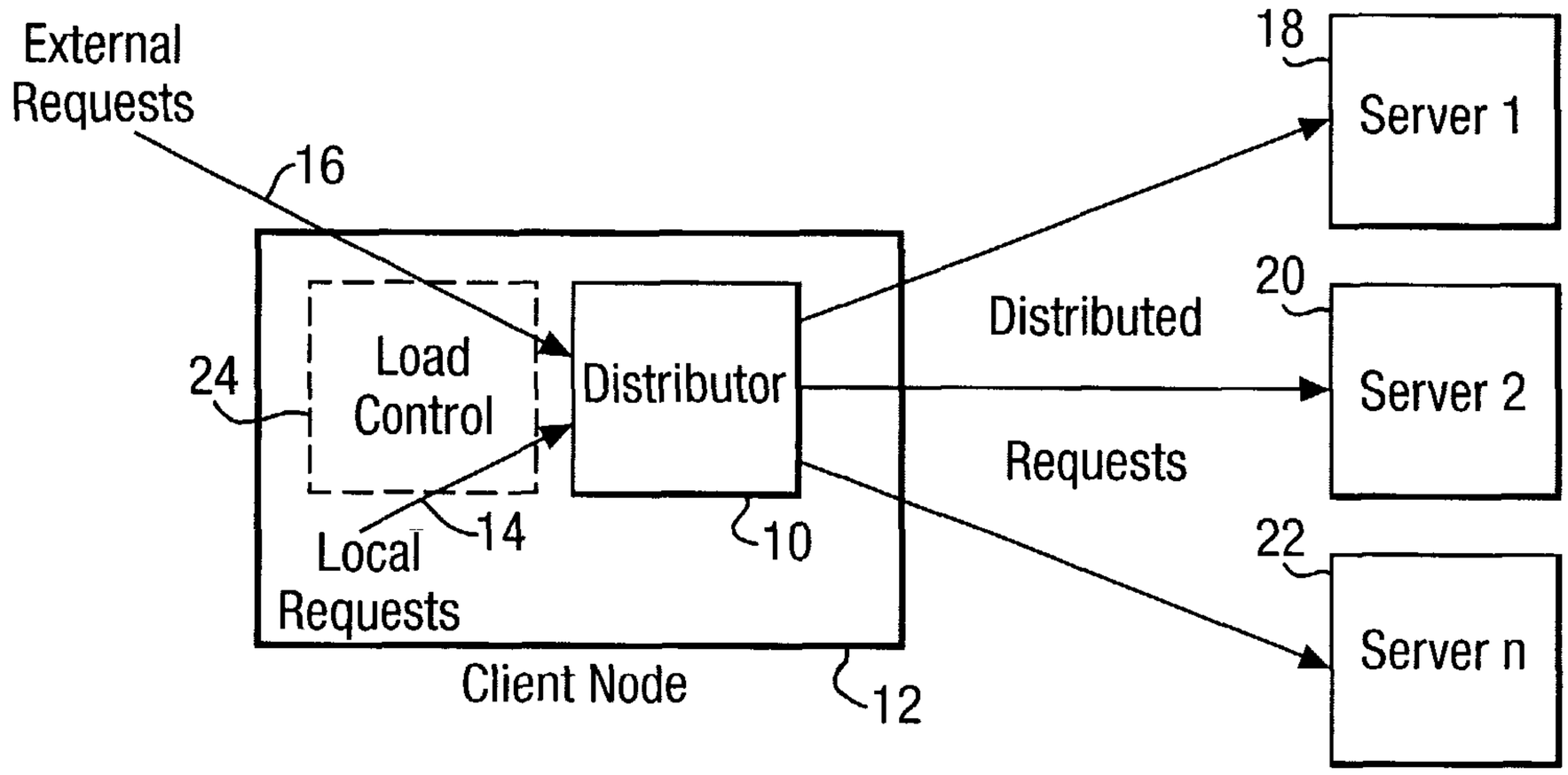
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(88) Date of publication of the international search report:
19 June 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: CLIENT SERVER NETWORKS



(57) Abstract: A client node in a network communicates with a server system comprising a plurality of servers. The client comprises a distributor which periodically measures server activity, assesses relative loading of the servers and adjusts the request distribution, and therefore the server loadings in accordance with the loading assessment.

WO 03/026244 A3

CLIENT SERVER NETWORKS

This invention relates to client server networks and, in particular, to networks in which clients have multiple alternative servers that can handle their requests.

In such client server networks, a problem arises in determining to which of the server nodes a client should send a given request. Known methods of dealing with this problem include load sharing, in which the load is spread across the multiple servers on a round robin basis, and worker standby schemes in which a server is designated as the worker to which requests are usually sent but an alternative server, the standby, handles requests when the worker is unable to. When the worker is again able to handle requests, the requests are again sent to it. However, neither of these approaches is satisfactory. Round robin load sharing is unsuitable if the capacities of the various servers differ significantly as the approach does not take relative server size into account. Worker-standby approaches are unsuitable if the request volume exceeds the capacity of a single server.

The effective capacity of any given server can vary considerably over time, for example because the server is handling requests from other clients, or for other reasons such as maintenance activity, such as archiving.

The present invention aims to address the problems outlined above and to provide an improved handling of requests to multiple servers.

Certain exemplary embodiments can provide a method of distributing requests from a client node to servers of a server system having a plurality of servers, the method comprising periodically repeating: measuring server activity of each server to obtain measured server activities; assessing relative loadings of the plurality of servers from the measured server activities; adjusting a distribution of requests to individual servers of the plurality of servers in accordance with the assessing of the relative loadings by adjusting a proportion of the requests assigned to each server as a function of the measured server activities, a mean activity across the plurality of servers, and an existing proportion of the requests assigned to the respective server; wherein the measuring step being performed by counting a number of the server requests sent to each of the plurality of servers, and counting a number of the server responses received from each server over a sample period; and configuring the sample period to be an aggregate number of requests sent to the plurality of servers.

Certain exemplary embodiments can provide a system for distributing requests from a client node to servers of a server system having a plurality of servers, comprising a distributor at the client node, the distributor including means for measuring the activity of each server, means for assessing the loading of each one of the plurality of servers relative to each other of the plurality from the measured server activities, and means for adjusting the distribution of requests to individual servers of the

plurality of servers in accordance with the assessment of the relative loadings; wherein the distribution adjustment means comprises means for adjusting the proportion of requests assigned to each server as a function of measured server activity, the mean activity across the plurality of servers and the existing proportion of requests assigned to the server; and wherein the activity measurement means comprises a counter for counting the number of requests for each server which have not received a response; wherein the relative loading assessment means comprises a comparator for comparing the number of requests without responses over an aggregate number of requests to all the plurality of servers.

Embodiments of the invention have the advantage that load distribution can be even across servers, taking into account their relative capacities.

Preferably, the client node comprises a load controller for applying a load control algorithm to the requests distributed to all servers of the plurality of servers. This has the advantage that loads can be

controlled to prevent all the servers overloading and that the distributor makes the plurality of servers look like a single server.

Preferably, server activity is measured by counting the number of server requests and responses over a period. Preferably, this period is an aggregate number of requests.

Preferably, the assessing of relative loading comprises comparing the server requests and responses for each of the servers over the sample period.

Preferably, the request distribution of the server is adjusted according to a function of measured server activity, mean activity across all the servers and the existing proportion of requests assigned to the server.

Preferably, an override facility is provided whereby a request related to a previous request is sent to the same server as the previous request.

Embodiments of the invention will now be described, by way of example, and with reference to the accompanying drawings in which:

Figure 1 shows a schematic diagram of a request distribution system embodying the invention;

Figure 2 is a UML use case diagram for the distributor of Figure 1;

Figure 3 is a UML class diagram for measurement of server loadings to assess relative loadings;

Figure 4 is an example sequence for a Distribute use case;

Figure 5 is an example sequence for a Response use case;

Figure 6 is an example sequence for a Measure use case;

Figure 7 is an example sequence for a Re-evaluate use case;

Figure 8 is a UML activity diagram for Distribute use case;

Figure 9 is a graph of responses and requests for a relatively large server showing the effect of overloading;

Figure 10 is a graph of requests and responses for a relatively small server, showing the effect of overloading;

Figure 11 is a flow chart showing a first embodiment of activity measurement;

Figure 12 is a flow chart showing a second embodiment of activity measurement; and

Figure 13 is a flow chart showing a third embodiment of activity measurement.

Referring to Figure 1, the embodiment to be described is based on a request distributor unit 10, within each client node 12. The request distributor receives both local requests 14 generated internally within the client node and external requests 16 received over an external interface. The distributor distributes server requests among n servers, here shown as server 1, 18, server 2, 20 and server n, 22. The client node also comprises a load control unit 24 which runs a load control algorithm. The distributor ensures that the remote servers are evenly loaded. The distributor is effectively front ended by the load control algorithm.

The architecture illustrated in Figure 1 enables the same load control to be used for external requests and requests originating locally at the client node. Also, the distributor makes the server network look like a single node to the load control algorithm. It can be seen from Figure 1 than the distributor is positioned, logically, between the servers and the load control.

Distributing the load to the servers on a request by request basis does not work when a request is related to an earlier request. The distributor includes an override

facility which enables some subset of requests to be sent to the same server. This is typically necessary where the server stores content derived from the first request that is needed to process the subsequent requests.

The UML use cases for the distributor are shown in Figure 2. The local application or remote client 26, time 28 and remote server 30 are shown as being remote from the system. The diagram shows that the distributor distributes 31 server requests in accordance with responses 32 from the servers and some measure 34 based on distribution and response. The distribution is re-evaluated over time 36.

Irrespective of the origin of the server request, that is external or internal to a client node, three requirements for distributing server requests to a set of available servers may be identified as follows:

1. To load the available servers evenly, as far as possible;

2. To avoid high frequency oscillations in server load by not overreacting to past load imbalances, causing new load imbalances; and

3. To allow override requests to insist on a specific server despite the distribution algorithm.

The override request of requirements is invoked when a subsequent request requires context data that would have been created in the server that handled a previous request. The response to the previous request indicated which server handled it.

The distribution used should preferably be independent of the particular details of the client-server

protocol to ensure wide applicability. In the embodiments to be described, the measurements for different servers are compared to assess their relative loadings. The distribution of subsequent requests is then adjusted to converge the server measurement to their mean. Thus, the loadings of the servers are rebalanced. This approach has no idea of whether the individual servers are overloaded as it is a relative measure of loading and so does not provide a complete control mechanism. If there is a concern at a client node to avoid overloading servers, the client node may incorporate a separate load control algorithm in front of the distributor for example as shown at 24 in Figure 1. The load control algorithm works from measurements of the aggregate traffic to all servers. A load control scheme suitable for a server node terminating request traffic should be equally suitable for a client node having a request distributor. It should be understood that load control is not essential.

The approach outlined above is based on measurements of relative loadings. It would be possible to measure traffic related quantities such as request and response counts and their rate derivatives, response time distributions (such as mean, 95th percentile, and standard deviation), and numbers of requests with responses pending. Although these are all universal measurements, they suffer to a greater or less degree from the problem that the threshold separating normal from overload conditions are usually protocol and/or application specific. For example, what constitutes a satisfactory response time for one application is likely to be unacceptable for another. Thus, a design based upon testing measurements against absolute values is unsatisfactory.

In addition to the approach based on measurements of relative loadings, some protocols include an explicit

facility for a server to reject client requests when it is overloaded. Distinguishing and counting these rejected requests at the distributor provides an additional set of measurements which can be used to control distribution.

Figures 3 to 7 show generalised UML class and sequence diagrams for the distributor. Figure 3 is a UML class diagram showing three entities: the server 38, request 40 and response 42. The server has the attributes of measurement, proportion, the request has the attributes of Timestamp, transaction ID and Override, and the response has the attributes of Timestamp and transaction ID. The server has the operations read and update.

Figure 4 shows the sequence for the Distribute use case in which the client or local application and Remote Server 42, 44 are shown outside the distributor system. Requests 46 are sent to the distributor and a read operation 48 performed on every server object instance to obtain current proportions, and which includes receiving responses 50 from the server objects. The requests are then distributed 52 to the appropriate remote server 44 depending on the relative loading.

Figure 5 shows the sequence for the response use case. Here, the remote server 44 sends responses to the Distributor_Control which sends the responses to the client or local application 42. Figure 6 shows the sequence for the Measure use case in which updates to measurements 51 are made between Distribution_Control and Server_Objects and replies 56 returned to the Distribution_Control.

Figure 7 shows the sequence for the Re-evaluate use case. The outside control here is time 58. On expiry of a timer, which is a periodic event, read measurements are made by the Distributor from the Server Objects and an update made. Both receive replies from the server objects and both are invoked on every instance of server object.

The distribution algorithm must not be disrupted by addition and removal of servers. A distribution by parts scheme is preferred although other algorithms are possible. Under the preferred scheme, the server's proportions are expressed as a ratio according to capacity. For example, in a three server case where one of the servers has twice the capacity of other two, the ratio would be expressed as 1:2:1.

The number of parts assigned to each server is P1, P2, P3...Pn. In the 1:2:1 case above, P1=1, P2=2 and P3=3. The distribution algorithm adds these numbers cumulatively from the first server to the last, recording the server identity against each interval. Thus, the range P1-P2 is assigned to server 2, P2-P3 to server 3 and so on. For each server request a random number is generated in the range 0 to $\sum$Pr(r=1-n). The server within whose range the random number lies is the server to which the request is sent.

Where the random number generated is equal to one of the interval boundary values, a tie breaker rule is applied.

Where the distributor starts up, it assumes that all servers have the same capacity and assigns the same proportion to each. Rather than applying the value 1, a larger value such as 10 is applied to allow for downward, as well as upward adjustments.

Figure 8 shows a UML activity diagram for the Distribute use case. At 60, Value B is set to zero. Within box 62, a server range is iterated for each server. This involves the steps of reading a proportion P from each server object 64, assigning a range B to (B+P-1) to the server 66 and then resetting the value of B to B=B+P at 68. A random number R is then generated at 70 and at 72 a value X is set such that X=R mod B. at 74, the server whose range

includes X is selected and at 76 a request is sent to that server.

In the foregoing description, it has been established that the selection of which server to send a request to should be based on a relative measurement of server loadings and that distribution is achieved using an algorithm based on server capacities. The following description concerns the measurement of server loading, and describes three possible ways of measuring server loadings: response time measurements, outstanding response count; and request and response count. The latter is especially preferred but the former two methods are possible. Other methods are also possible and will occur to those skilled in the art.

Response Time Measurements

The distributor can calculate the response time distributions of each server. The relative loadings are then evaluated by direct comparisons of the chosen measure, for example the mean or 95th percentile. As a server becomes overloaded it should exhibit a sharp increase in its response time. This solution is illustrated in the flow chart of Figure 12 which shows response time measurement at step 120 which is performed for all servers, followed by a comparison of the distribution of response times at step 122. This approach, although possible, suffers from a number of problems:

First, individual servers may differ substantially in their natural response times when working normally, making it more difficult to distinguish overloaded servers. Secondly, a certain minimum number of response time samples must be collected before their statistical distribution can be assessed reliably. This may make the sample periods too long and the distributor too sluggish in its reaction to

changes. Third, the distributor has to associate responses with the relevant requests to obtain timings.

Outstanding Responses Counting

As an alternative, the distributor maintains a count of the number of requests sent to each server that has not received at least one response. As this is a continuous measurement, periodic evaluation could be based either on fixed intervals or aggregate numbers of request. As a server becomes overloaded it should show an increase in its outstanding responses. TCP load control is based on this principle. The process is illustrated in Figure 13 which shows the counting of requests which have received no response at step 124 and the comparison of these counts over a period of time or aggregate number of requests at step 126. However, there are two potential problems. First, the approach is indirectly affected by any normal differences between server response times. This follows from the basic queuing theory formula which states that:

mean concurrency = transactions per second (tps)
x mean response time.

Thus, it is again difficult to distinguish distressed servers from naturally slow servers.

Second, the distributor has to distinguish initial responses to requests from subsequent responses in order that the outstanding response count is only decremented when an initial response is sent. This is difficult to achieve.

Request and Response Counting

This option is the most preferred of the three considered. The distributor records, over a short sample

period, the number of requests sent to each server and the number of responses received back from each server. The relative loadings are then evaluated by comparing request and response counts for the sample period. The method is illustrated in Figure 11 in which server requests are counted at 100, server responses counted at step 110 and a comparison made at step 112. This process is repeated for each server.

The absolute numbers of requests or responses may differ between servers due to the differing capacities. However, provided that none is overloaded the request/response ratio for each should be almost identical. This relies on each server receiving the same mixture of request types, which is true over a large number of requests but may not be reliable over short periods containing few requests. It is preferred therefore, to define samples by aggregate number of requests sent, rather than by time periods.

There are many candidates for the values that might be compared to decide relative loading. A straight ratio of requests to responses is only one alternative. The following is a non-exhaustive list of some candidate quantities and their theoretical ranges. In the list Rq is the number of theoretical requests sent to a server and Rs the number of responses returned.

1. Rq/Rs : 0 ⟹ + ∞
2. Rs/Rq : 0 ⟹ + ∞
3. Rq-Rs : −∞ ⟹ + ∞
4. Rs-Rq : −∞ ⟹ + ∞
5. Rq/(Rc+Rs) : 0 ⟹ 1
6. Rs/(Rs+Rc) : 0 ⟹ 1
7. (Rq-Rs)/(Rq+Rs): -1 ⟹ +1
8. (Rs-Rq)/(Rq+Rs): -1 ⟹ +1

Of these candidate values, nos 1 to 4 have unlimited ranges, caused by the possibility of sample periods with zero responses or requests. Options 5 and 6 give greater emphasis to either the request or response count. Theory suggests that it is the normalised difference between requests and responses counts that is the most informative indication of relative server loadings, suggesting that options 7 and 8 are the most preferable. The polarity of the value is one consideration, that is whether it increases or decreases with loading. It is intuitive that the value should increase as loading increases favouring option 7. However, as most protocols spawn, on average, slightly more than one response per request, the last option, option 8 is favoured as the server measurements would be positive under normal conditions.

Figures 9 and 10 show the request and response counts of two servers as they are subjected to increasing traffic. The figures are both graphs of messages per second (mps) against time. The server of Figure 9 has a relatively large capacity and that of Figure 10 a relatively small capacity. The two servers have equal traffic shares. The figures emphasise how the server's response rate plateaus as the server reaches its capacity and becomes distressed. Figures 9 and 10 are based on the assumption of an average 1.5 responses per request.

In Figure 10, the smaller server starts to plateau at time t1. At this time, the distributor should start to shift request traffic to the larger server.

Distribution Adjustment

It was mentioned previously that it is important not to overcompensate any adjustment of the server distribution, so setting up oscillations. Periodically, or

after a certain number of requests, the distributor will need to adjust the proportions of request traffic assigned to each server. The objective is to reduce the discrepancies between the per server values of the measurements chosen. This may be done as follows:

If M_1, M_2, $M_3 \ldots M_n$ are the measurements calculated for the last sample period for each server, μ is their arithmetic mean, and P_1, P_2, $P_3 \ldots P_n$ their currently assigned proportions per server, the new proportion assigned to the rth server is in general given by a formula of the following kind:

$$P_r = f\ (M_r,\ \mu,\ P_r)$$

Two candidate formulae and their theoretical ranges are as follows:

1. $(1+(M_r-\mu)/(|M_r|+|\mu|))\ x\ P_c\ \ 0 \Longrightarrow 2xP_r$

2. $(1-(M_r-\mu)/(|M_r|+|\mu|))\ x\ P_c\ \ 0 \Longrightarrow 2xP_r$

The difference between the two reflects the polarity choice referred to in the choice between options 7 and 8 above. These formulae produce new values proportional to the old and a proportion must never, therefore, be adjusted to zero due to arithmetic truncation.

Oscillations may be reduced by one of two ways. First, the data from the last few sample periods may be explicitly combined to compute the adjustments to proportions for the next period. This effectively uses a sliding window of retained data. Second, the data from all previous samples may be implicitly combined using a decay formula. Under this approach, data for second period adjustment is the average of true second period data and

data from the first period. Data used for the third period is the average of true third period data and data used for the second period etc.

The embodiment described enables distribution of requests to multiple servers from a client node in a manner which reflects the individual capacities of the servers and based on their relative loadings.

Various modifications to the embodiments described are possible and will occur to those skilled in the art. For example, other methods of measuring server activity may be possible and other distribution adjustment formulae may be adopted. However, such modifications are within the scope of the invention which is defined by the appended claims.

CLAIMS

1. A method of distributing requests from a client node to servers of a server system having a plurality of servers, the method comprising periodically repeating:

measuring server activity of each server to obtain measured server activities;

assessing relative loadings of the plurality of servers from the measured server activities;

adjusting a distribution of requests to individual servers of the plurality of servers in accordance with the assessing of the relative loadings by adjusting a proportion of the requests assigned to each server as a function of the measured server activities, a mean activity across the plurality of servers, and an existing proportion of the requests assigned to the respective server; wherein

the measuring step being performed by counting a number of the server requests sent to each of the plurality of servers, and counting a number of the server responses received from each server over a sample period; and

configuring the sample period to be an aggregate number of requests sent to the plurality of servers.

2. A method according to claim 1, wherein assessing the relative loading of the plurality of servers comprises comparing the server request and response counts for each of the plurality of servers over the sample period, where the sample period is a period of time.

3. A method according to claim 1, wherein measuring the activity of each server comprises calculating the response time of each server.

4. A method according to claim 3, wherein assessing the relative loading of the plurality of servers comprises comparing a distribution of the server response times for each of the plurality of servers.

5. A method according to claim 1, wherein measuring the activity of each server comprises counting the number of requests for each server which have not received a response.

6. A method according to claim 5, wherein assessing the relative loading of the plurality of servers comprises comparing the number of requests without responses over an aggregate number of requests to all of the plurality of servers.

7. A method according to claim 1, wherein adjusting the distribution of requests comprises combining distribution data from previous sample periods to calculate adjustments to distributions for a next sample period.

8. A method according to claim 1 or 7, wherein adjusting the distribution of requests comprises combining data from previous sample periods according to a decay formula.

9. A method according to claim 2, wherein the relative loadings are compared by comparing the ratio $(R_q-R_s)/(R_q+R_s)$ where R_q is the number of requests sent to the server and R_s is the number of responses returned from the server.

10. A method according to claim 2, wherein the relative loadings are compared by comparing the ratio $(R_s-R_q)/(R_q+R_s)$ where R_q is the number of requests sent to the server and R_s is the number of responses returned from the server.

11. A system for distributing requests from a client node to servers of a server system having a plurality of servers, comprising a distributor at the client node, the distributor including means for measuring the activity of each server, means for assessing the loading of each one of the plurality of servers relative to each other of the plurality from the measured server activities, and means for adjusting the distribution of requests to individual servers of the plurality of servers in accordance with the assessment of the relative loadings;

wherein the distribution adjustment means comprises means for adjusting the proportion of requests assigned to each server as a function of measured server activity, the mean activity across the plurality of servers and the existing proportion of requests assigned to the server; and

wherein the activity measurement means comprises a counter for counting the number of requests for each server which have not received a response;

wherein the relative loading assessment means comprises a comparator for comparing the number of requests without responses over an aggregate number of requests to all the plurality of servers.

12. A system according to claim 11, wherein the means for measuring server activity comprises a counter for counting the number of server requests sent to each of the plurality of servers and for counting the number of server responses received from each server over a sample period, where the sample period is a period of time.

13. A system according to claim 12, wherein the relative loading assessing means comprises means for comparing the server request and response counts for each of the plurality of servers over the sample period.

14. A system according to claim 11, wherein the activity measurement means comprises calculating means for calculating the response time of each server.

15. A system according to claim 14, wherein the relative loading assessment means comprises means for comparing a distribution of the server response times for each of the plurality of servers.

16. A system according to claim 14, wherein the request distribution adjustment means comprises a combiner for combining distribution data from previous sample periods to calculate adjustments to distributions for a next sample period.

17. A system according to claim 14, wherein the means for comparing the server requests and responses compares the ratio $(R_q-R_s)/(R_q+R_s)$ where R_q is the number of requests sent to the server and R_s is the number of responses returned from the server.

18. A system according to claim 14, wherein the means for comparing the server requests and responses compares the ratio $(R_s-R_q)/(R_q+R_s)$ where R_q is the number of requests sent to the server and R_s is the number of responses returned from the server.

Fig.1.

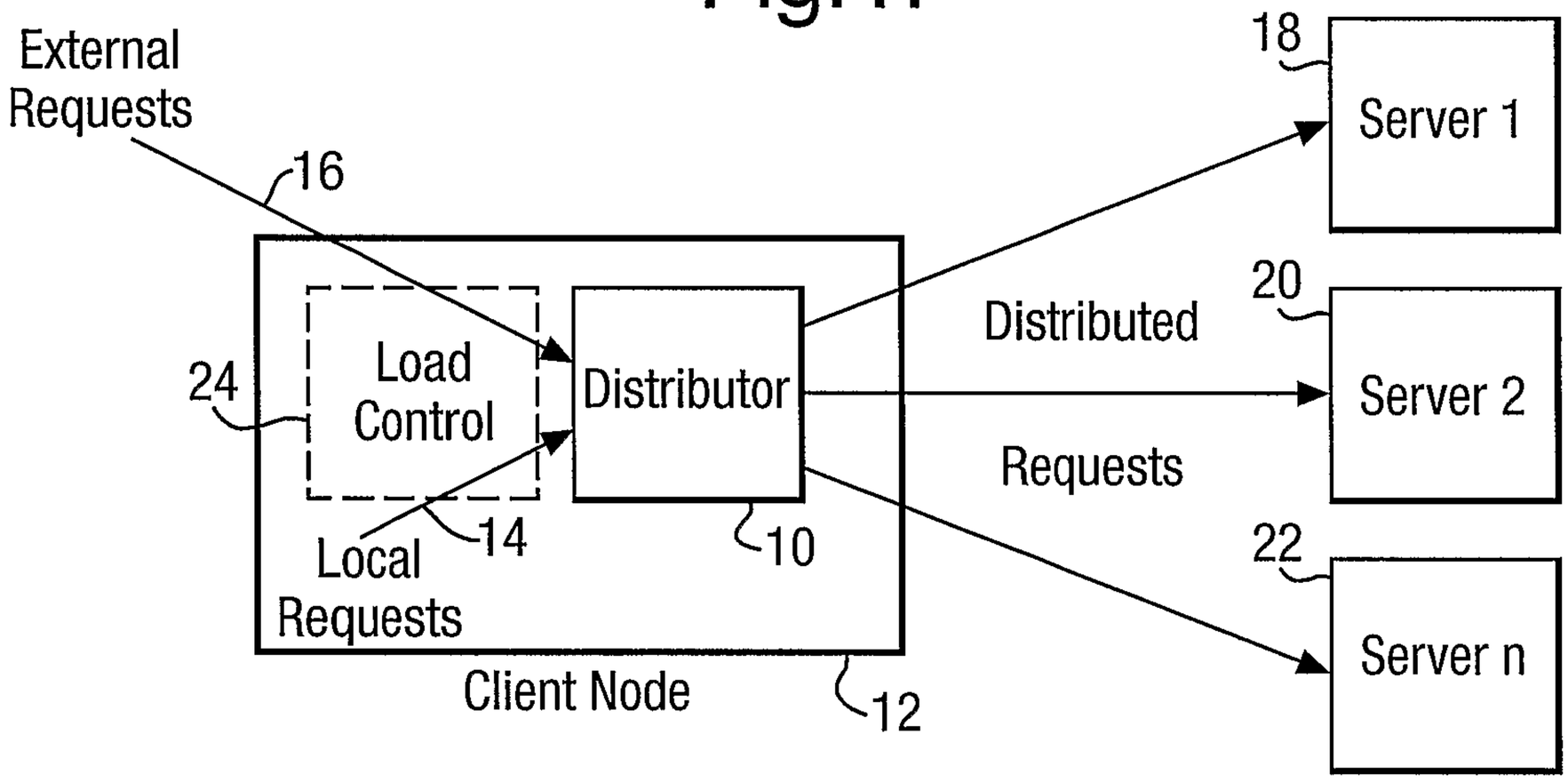

External Requests
16
18
Server 1
Load Control
24
Distributor
Distributed Requests
20
Server 2
22
Server n
Local Requests
14
10
Client Node
12

Fig.2.

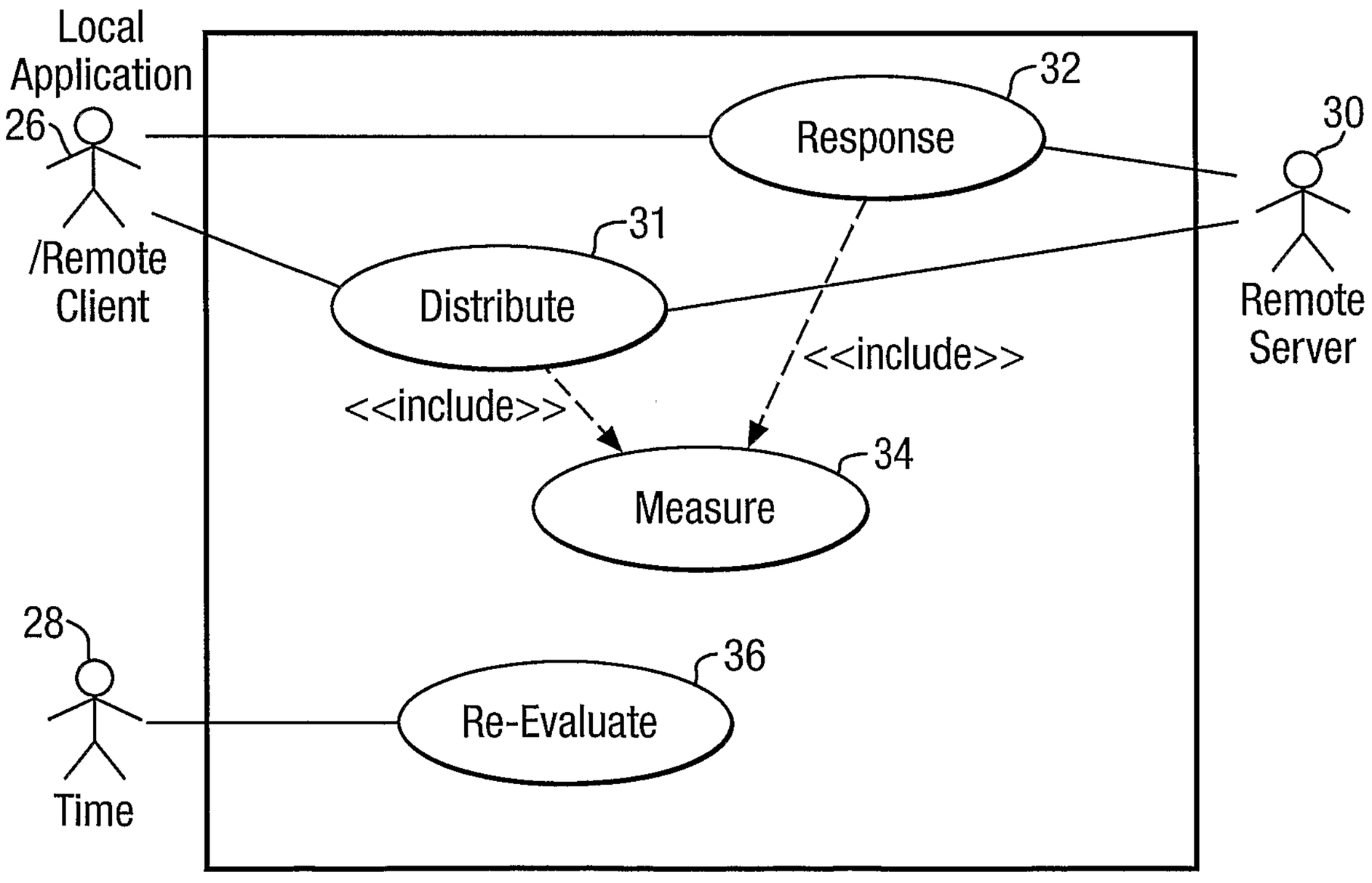

Local Application
26
/Remote Client
32
Response
30
Remote Server
31
Distribute
<<include>>
<<include>>
34
Measure
28
Time
36
Re-Evaluate

Fig.3.

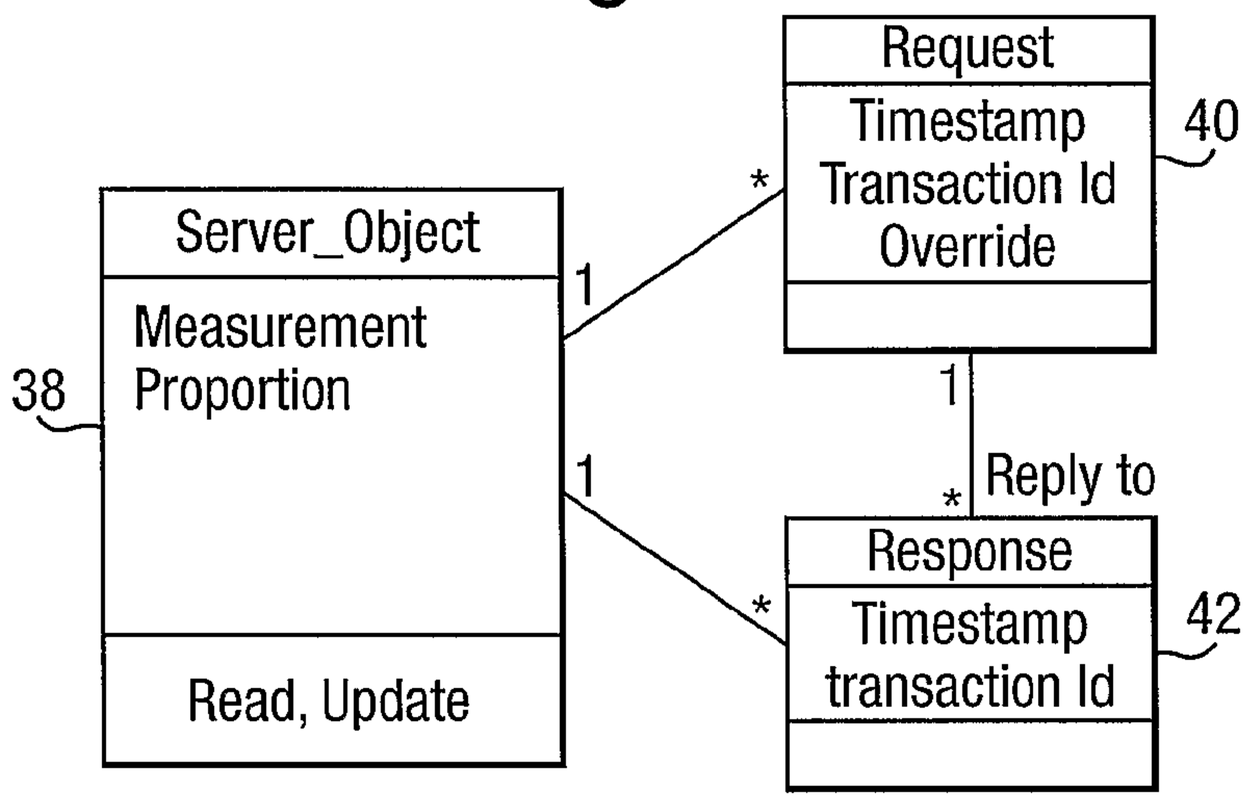
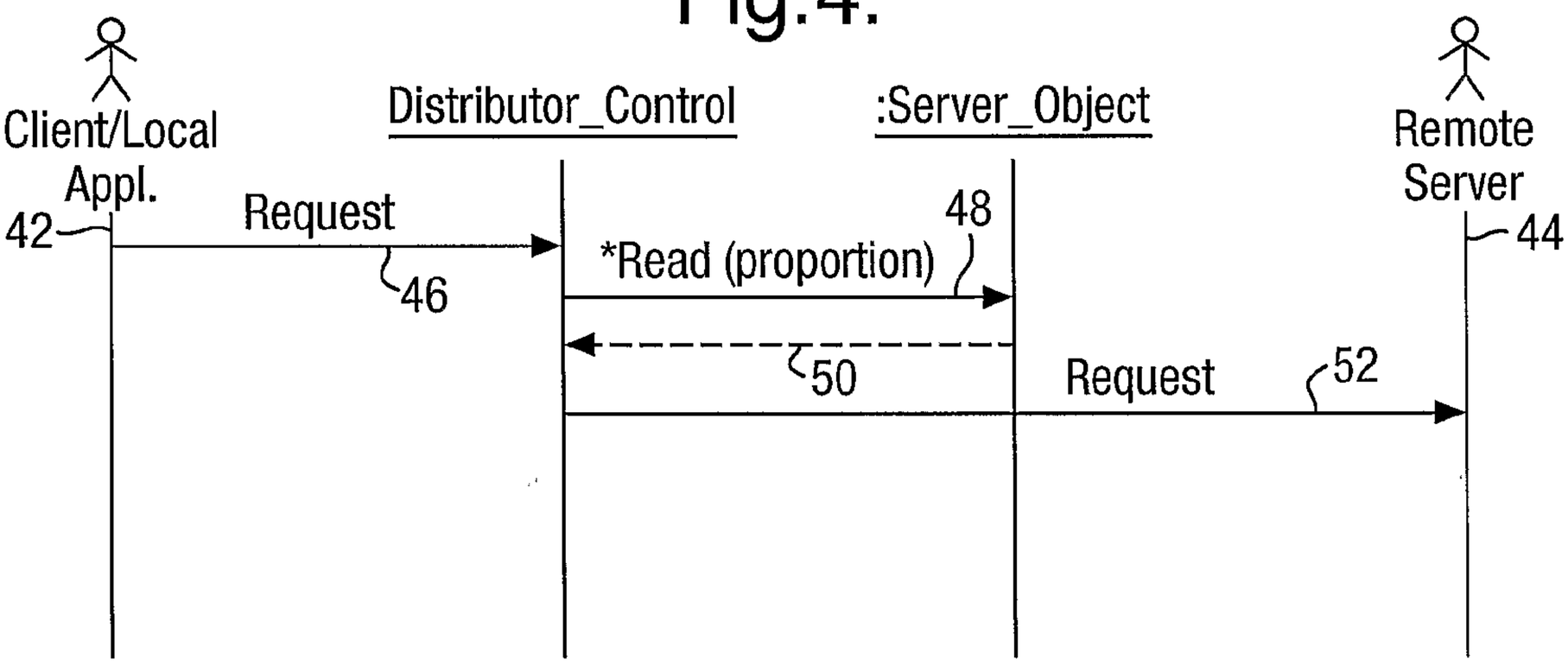


Fig.4.



Note* : This operation is invoked on every server object instance

Fig.5.

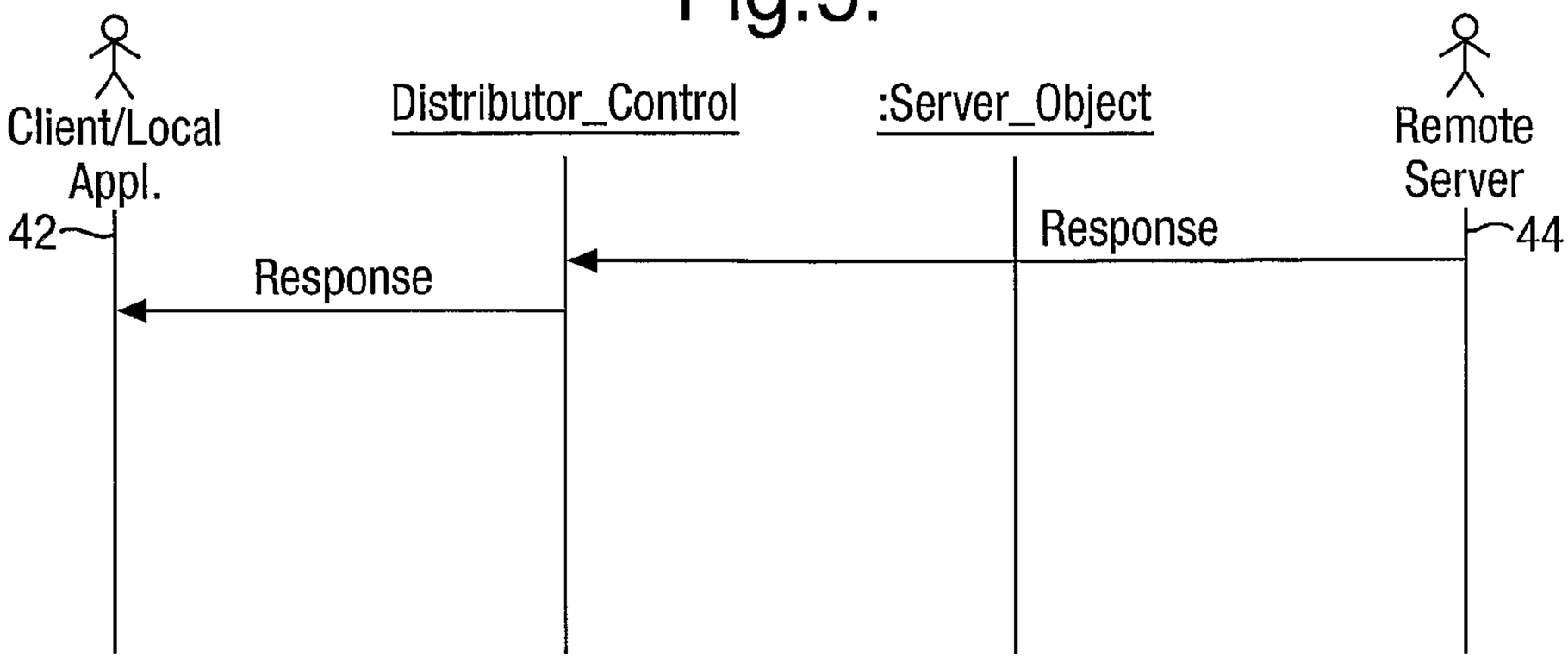


Fig.6.

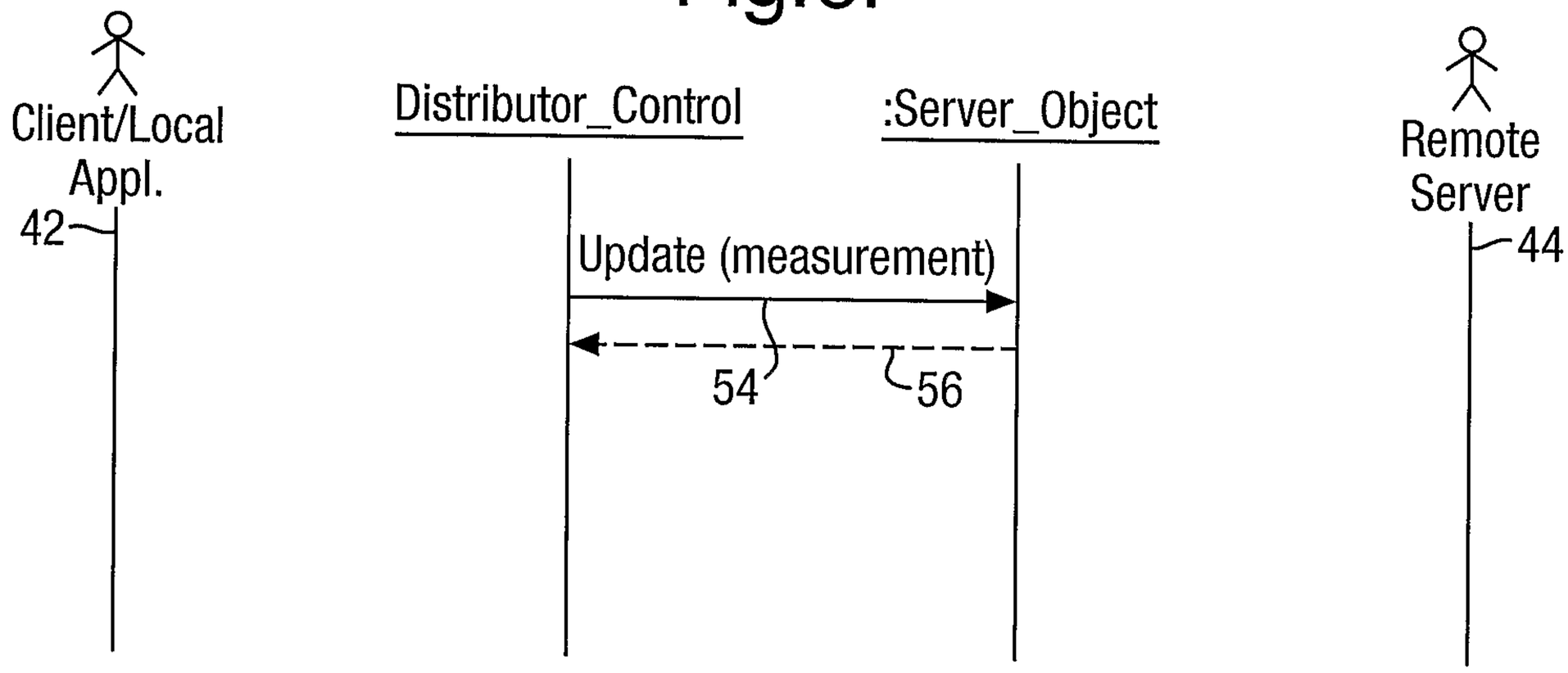

Client/Local Appl.
42
Distributor_Control
:Server_Object
Remote Server
44
Update (measurement)
54
56

Fig.7.

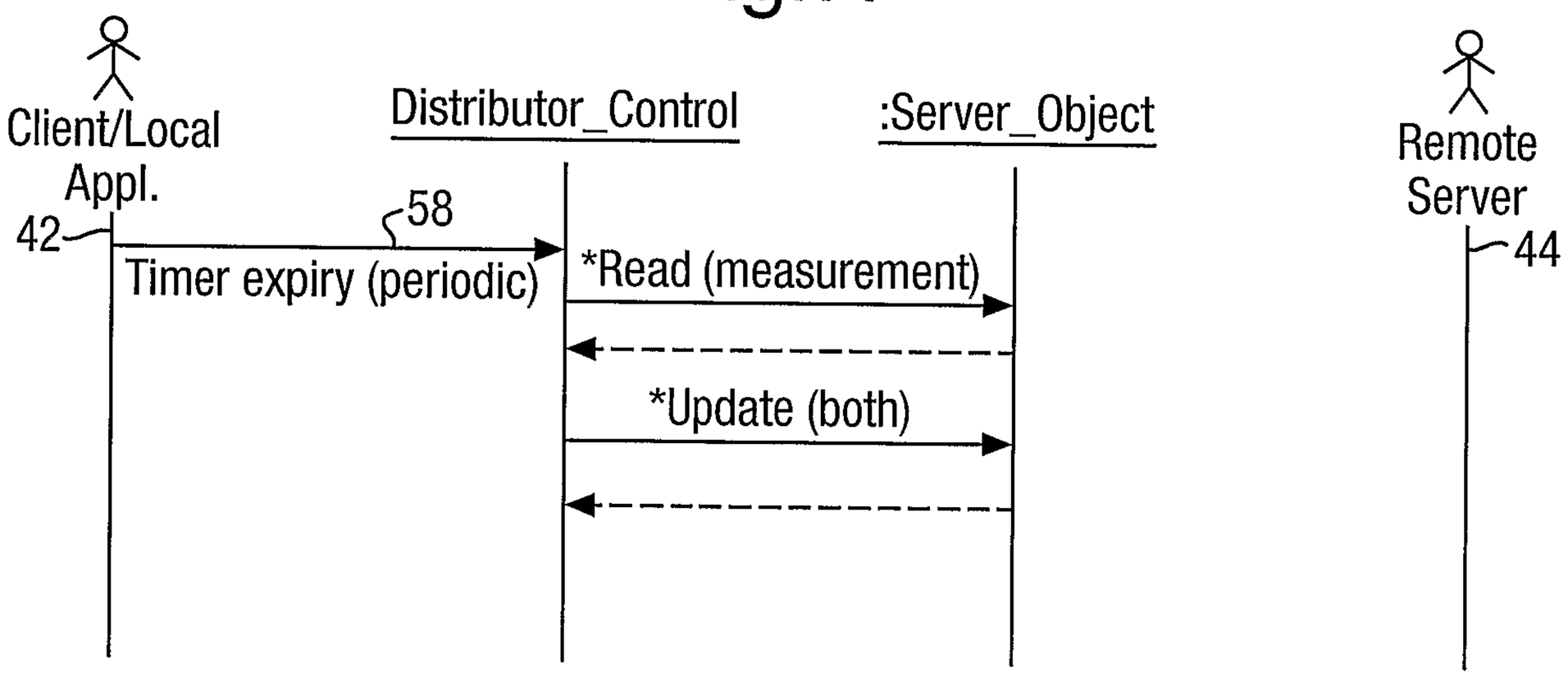
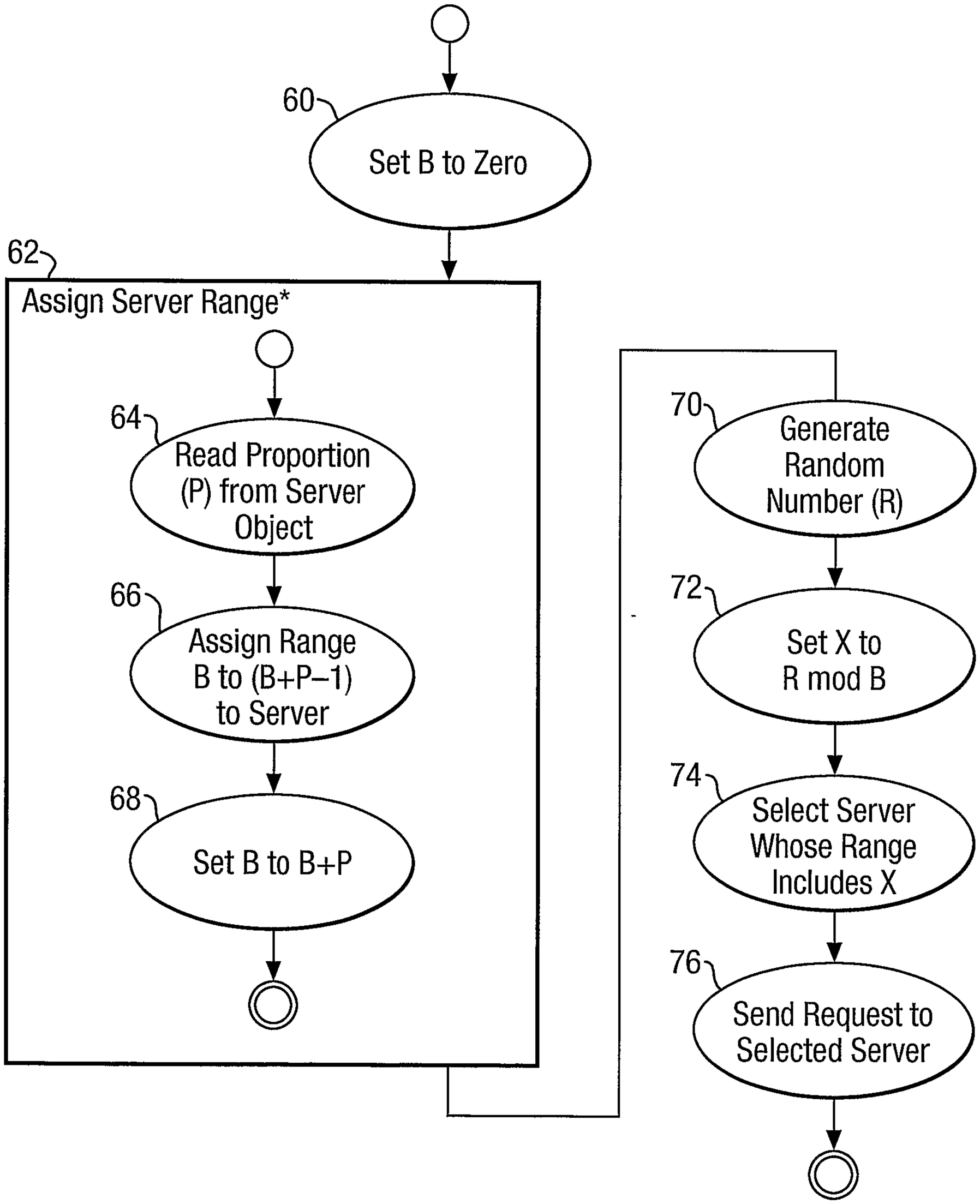

Client/Local Appl.
42
Distributor_Control
:Server_Object
Remote Server
44
58
Timer expiry (periodic)
*Read (measurement)
*Update (both)
Note* : This operation is invoked on every server object instance

Fig.8.



Note* : This composite activity is iterated for each server

Fig.9.

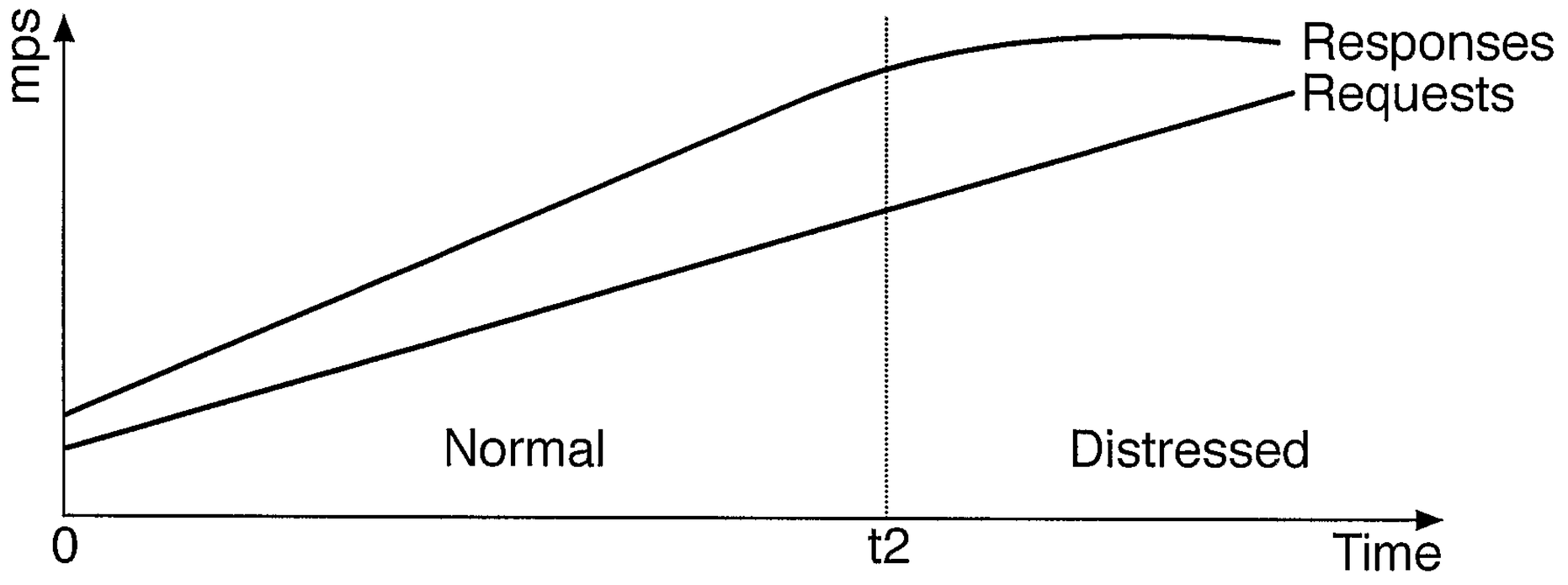

mps
Responses
Requests
Normal
Distressed
0
t2
Time

Fig.10.

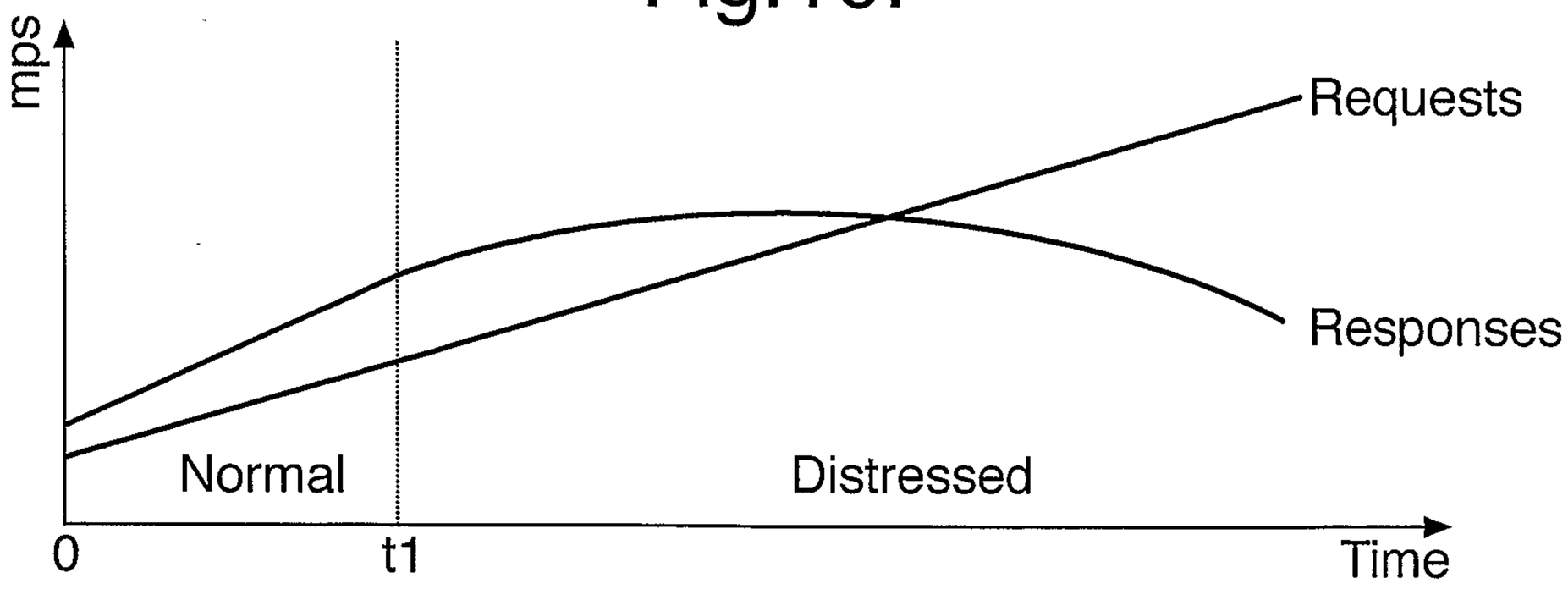

mps
Requests
Responses
Normal
Distressed
0
t1
Time

Fig.11.

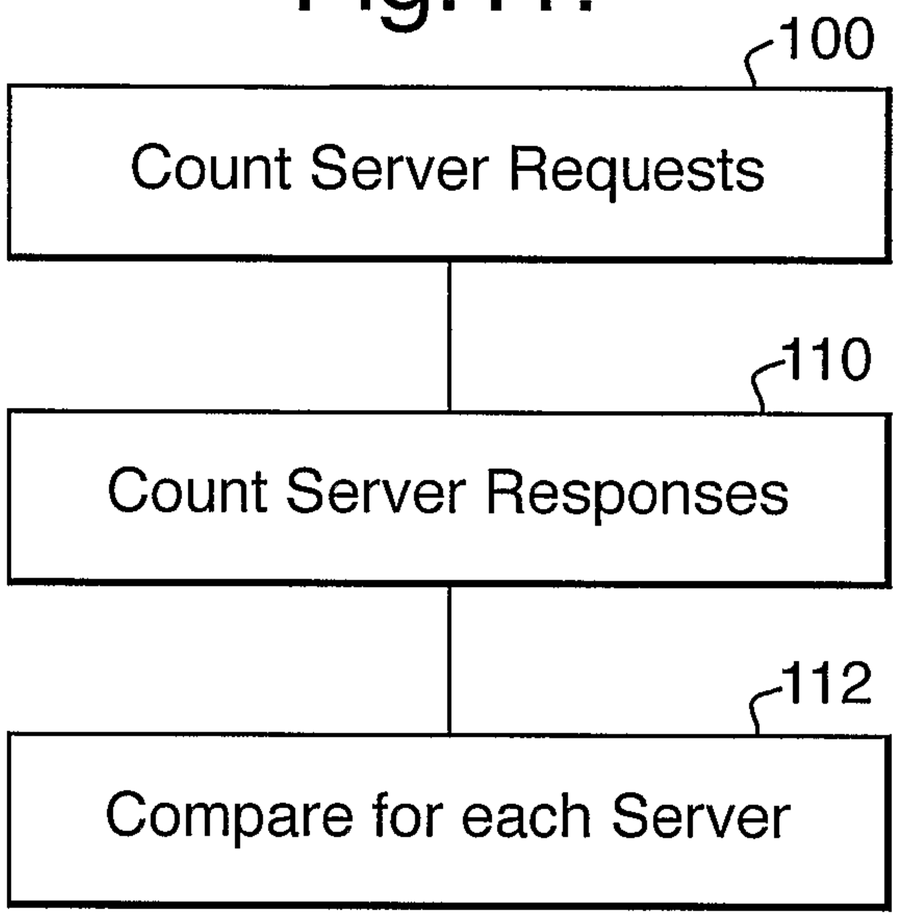


Fig.12.

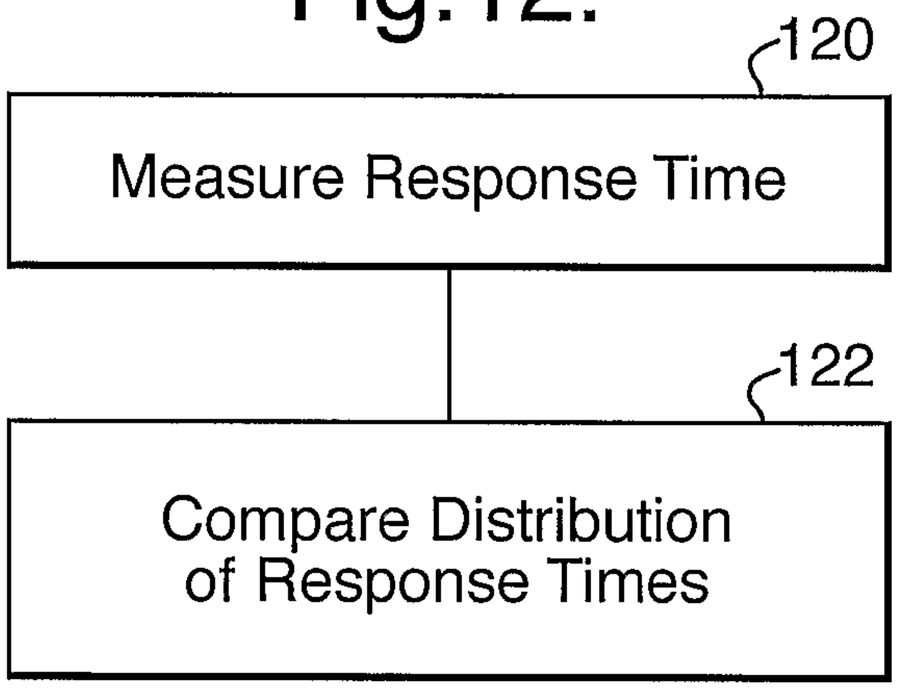
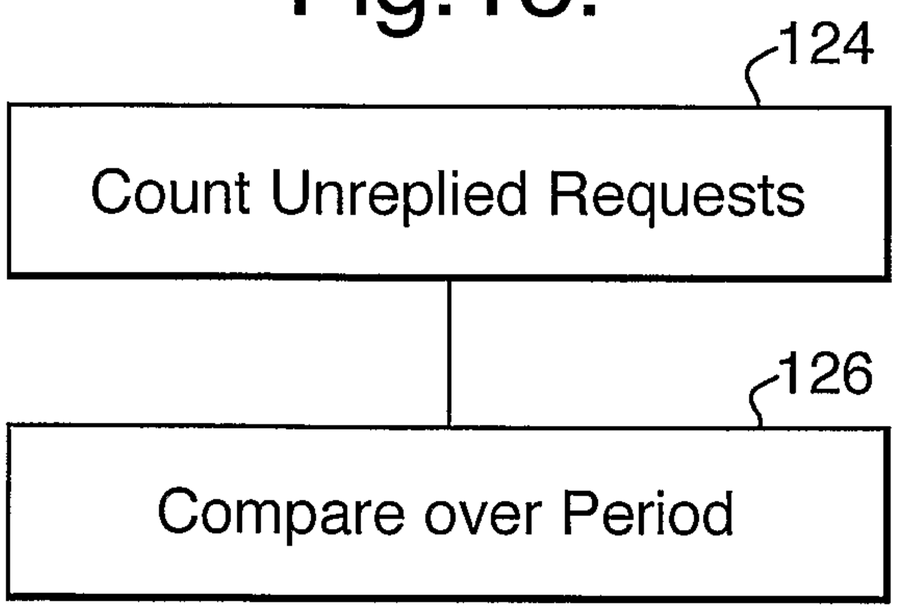
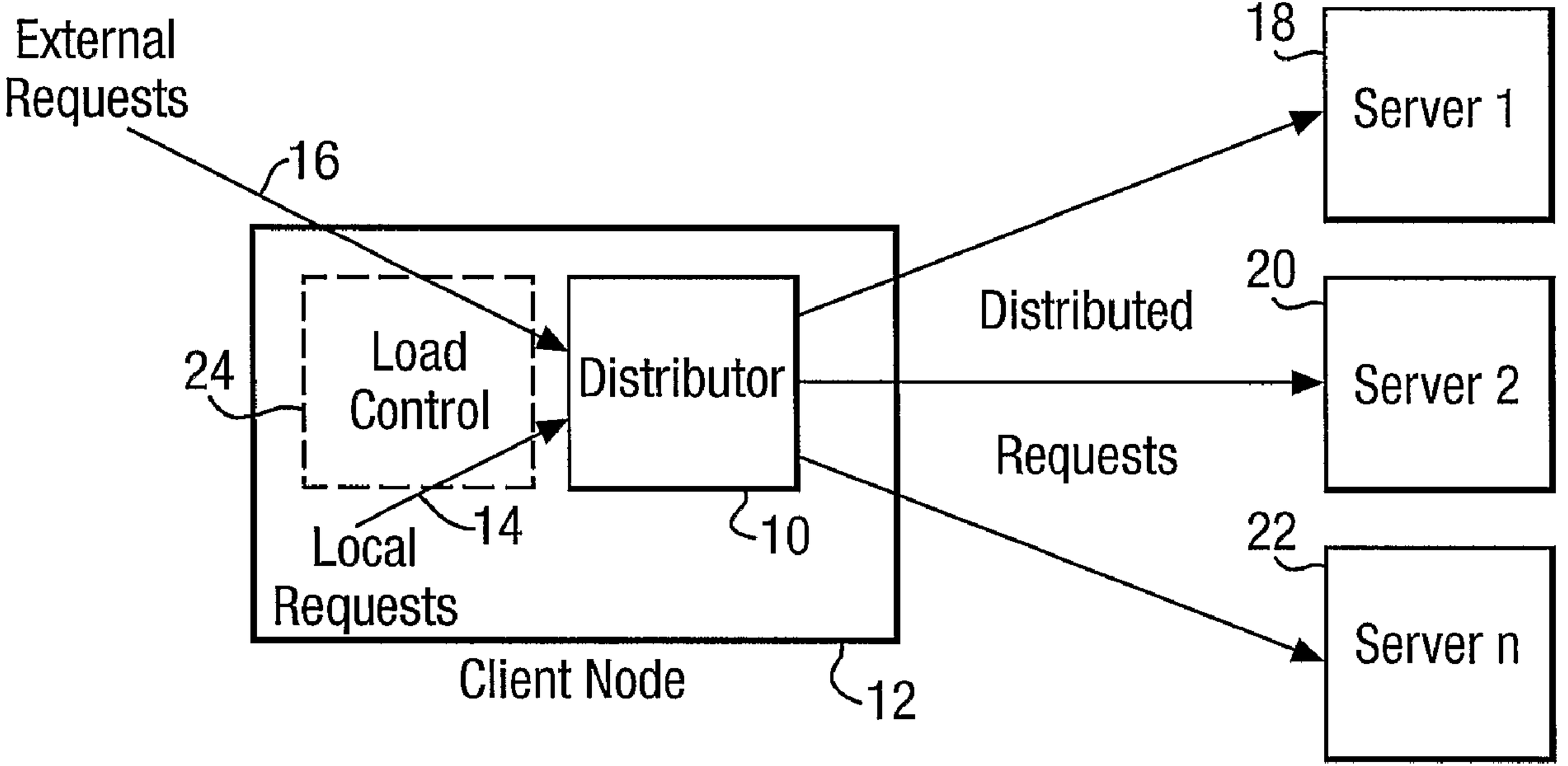


Fig.13.




External
Requests
16
18
Server 1
20
Server 2
22
Server n
Distributed
Requests
24
Load
Control
Distributor
14
10
Local
Requests
Client Node
12