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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 March 2014 (27.03.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/047269 A2

(51) **International Patent Classification**:
*G06F 9/00* (2006.01)

(21) **International Application Number**:
PCT/US2013/060591

(22) **International Filing Date**:
19 September 2013 (19.09.2013)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
13/622,546 19 September 2012 (19.09.2012) US

(71) **Applicant**: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, California 94065 (US).

(72) **Inventors**: **KAEMMERER, Jens**; 254 Tyrella Avenue, Mountain View, California 94043 (US). **SRIVASTAVA, Ashish**; 3500 Granada Avenue, Apt. 349, Santa Clara, California 95051 (US).

(74) **Agents**: **MEYER, Sheldon, R.** et al.; Fliesler Meyer LLP, 410 Pacific Avenue, San Francisco, California 94133 (US).

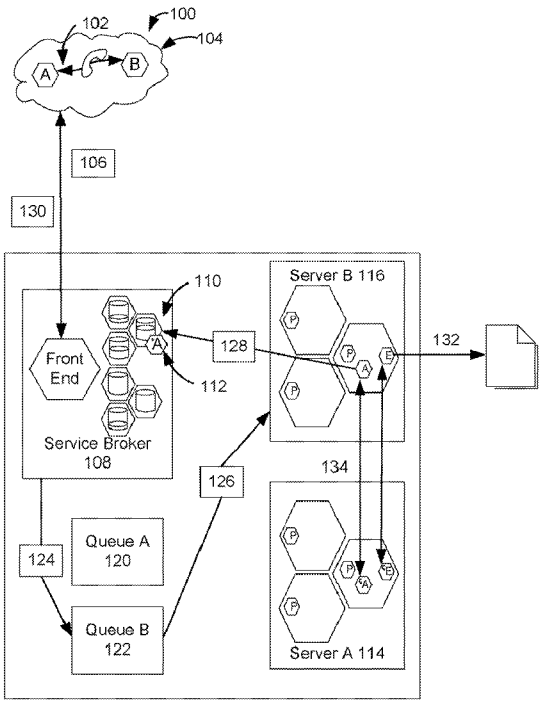
(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, 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,

*[Continued on next page]*

(54) **Title**: SYSTEM AND METHOD FOR SMALL BATCHING PROCESSING OF USAGE REQUESTS

Figure 1




(57) **Abstract**: In accordance with various embodiments, systems and methods that provide unified charging across different network interfaces are provided. A system for small batch processing of usage requests, can include a service broker, a plurality of servers wherein each server includes customer data, and a plurality of queues, each associated with a different server. When a usage request is received from a network entity, the service broker is configured to determine an internal ID associated with data requested by the usage request, determine on which particular server of the plurality of servers the data requested by the usage request is stored, enqueue the usage request in a particular queue associated with the particular server, and upon a trigger event, send all requests in the particular queue to the particular server in a batch.

WO 2014/047269 A2

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published**:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

# SYSTEM AND METHOD FOR SMALL BATCHING PROCESSING OF USAGE REQUESTS

## COPYRIGHT NOTICE:



## FIELD OF THE INVENTION:

**[0002]** The current invention relates to online charging systems in telecommunications networks and in particular to a system and method for batch processing of requests in a telecommunications system.

## BACKGROUND:

**[0003]** Typically, high volumes of usage requests are sent in a continuous network stream to an Online Charging System (OCS) entry point from an Intelligent Network node (IN). A usage request is any form of usage processing which requires customer data for charging purposes including, for example: a kilowatt of electricity used by a commercial customer during peak hour; a call from one subscriber to another; or a download request. Each usage request is processed separately and a response is returned to the originating IN.

**[0004]** Processing of usage requests is typically accomplished asynchronously: while one usage request is being processed, the next usage request can already be read from the network connection. The chronological order of incoming usage requests and outgoing usage responses can be different as a result of this asynchronous processing. Inside the OCS, processing of an individual usage request is typically accomplished synchronously: the usage request is sent from the OCS entry point to the OCS business logic nodes for processing. One OCS entry point typically serves many OCS business logic nodes.

**[0005]** Sending a small usage request (typically 100-200 bytes in size) results in costly network IO operations, context switches and transmission latency. If the time spent to process the individual usage request is very short (e.g., below 1 ms) – this cost can become a limiting factor for the OCS throughput and increase the Total Cost of Ownership (TCO) of the OCS. The OCS latency requirements for usage request processing is: 99.9% of all usage requests should be processed in less than 50 ms.

## SUMMARY:

**[0006]** In accordance with various embodiments, systems and methods that provide small batch processing of usage requests are provided. A system for small batch processing of usage requests, can include a service broker, a plurality of servers wherein each server includes

customer data, and a plurality of queues, each associated with a different server. When a usage request is received from a network entity, the service broker is configured to determine an internal ID (Identification) associated with data required for processing the usage request, determine on which particular server of the plurality of servers the data requested by the usage request is stored, enqueue the usage request in a particular queue associated with the particular server, and upon a trigger event, send all requests in the particular queue to the particular server in a batch. In accordance with other various embodiments, a program is provided for causing one or more computers to execute a method for small batch processing of usage requests, comprising: providing a service broker executing on one or more microprocessors; providing a plurality of servers, wherein each server can include customer data; providing a plurality of queues, wherein each server is associated with a different queue; receiving a usage request from a network entity; determining an internal ID (Identification) associated with data requested by the usage request; determining on which particular server of the plurality of servers the data requested by the usage request is stored; enqueuing the usage request in a particular queue associated with the particular server; and upon a trigger event, sending all requests in the particular queue to the particular server in a batch.

## BRIEF DESCRIPTION OF THE DRAWINGS:

**[0007]** **Figure 1** shows a system for small batch processing of usage requests, in accordance with an embodiment of the invention.

**[0008]** **Figure 2A** shows an individual request system.

**[0009]** **Figure 2B** shows a small batch system in accordance with an embodiment of the invention.

**[0010]** **Figure 3** shows a sequence diagram of small batch creation, in accordance with an embodiment of the invention.

**[0011]** **Figure 4** shows a sequence diagram of small batch request processing, in accordance with an embodiment of the invention.

**[0012]** **Figure 5** shows a method for small batch processing of usage requests, in accordance with an embodiment of the invention.

**[0013]** **Figure 6** shows in detail an illustrative service broker in accordance with an embodiment of the invention.

## DETAILED DESCRIPTION:

**[0014]** In the following description, the invention will be illustrated by way of example and not by way of limitation in the figures of the accompanying drawings. References to various embodiments in this disclosure are not necessarily to the same embodiment, and such references mean at least one. While specific implementations are discussed, it is understood that this is provided for illustrative purposes only. A person skilled in the relevant art will

recognize that other components and configurations may be used without departing from the scope and spirit of the invention.

**[0015]** Furthermore, in certain instances, numerous specific details will be set forth to provide a thorough description of the invention. However, it will be apparent to those skilled in the art that the invention may be practiced without these specific details. In other instances, well-known features have not been described in as much detail so as not to obscure the invention.

**[0016]** Typically, high volumes of usage requests are sent in a continuous network stream to an Online Charging System (OCS). A usage request is any form of usage processing which requires customer data for charging purposes including, for example: a kilowatt of electricity used by a commercial customer during peak hour; a call from one subscriber to another; or a download request. In accordance with various embodiments, systems and methods that provide small batch processing of usage requests are provided. A system for small batch processing of usage requests, can include a service broker, and a plurality of servers forming the OCS. Each server includes customer data, and a plurality of queues, each associated with a different server. When a usage request is received from a network entity, the service broker is configured to determine an internal ID associated with data required for processing the usage request, determine on which particular server of the plurality of servers the data requested by the usage request is stored, enqueue the usage request in a particular queue associated with the particular server, and upon a trigger event, send all requests in the particular queue to the particular server in a batch.

**[0017]** In accordance with various embodiments of the invention, instead of sending each individual usage request directly from the OCS entry point to the OCS business logic layer, usage requests can first be sorted based on their OCS business logic node destination. Each node of the OCS business logic layer carries out the same business logic. However, customer data is partitioned across all OCS business logic. The OCS business logic node destination is determined based on the location of the customer data.

**[0018]** All usage requests with the same destination are then placed in the same 'small batch' container (i.e., queue). The maximum size of each 'small batch' container can be set to an arbitrary number depending on the particular needs and features of a system. Although examples shown herein use a maximum size of 20, this is not intended to be limiting in any way. The 'small batch' container is sent once the maximum size has been reached. The cost of sending the 'small batch' container in terms of network IO operations, context switches and transmission latency is significantly less on a per usage request basis than the cost of sending each usage request individually.

**[0019]** Overall OCS throughput can be increased in exchange for increasing individual usage request latency. Latency of the individual usage request is now a function of the size of the 'small batch' and increases to 30 ms (assuming 1 ms processing time for an individual

usage request). The OCS latency requirement of less than 50 ms is still fulfilled. Additional triggers can also be provided which cause the small batch to be sent. For example, during times of low usage request traffic volumes, a 'small batch' timeout mechanism can trigger sending of an incomplete 'small batch' to guarantee less than 50 ms latencies. Additional trigger can be provided for prioritization of requests. Upon arrival of a specific type of request, the small batch is sent immediately. For example, upon arrival of a long running request (ie. If executing the logic takes >25ms of processing time) the request can be sent immediately in a small batch all by itself.

**[0020]** **Figure 1** shows a system for small batch processing of usage requests, in accordance with an embodiment of the invention. At 100, Subscriber A 102 makes a call to subscriber B 104. Although a call is shown in Figure 1, this can be anything that the system views as an event, e.g., an SMS from A to B, a request by A to download media for which there is a charge, or any other action that results in a usage request. The request 106 is sent to a service broker 108 which can translate the request from a network-centric protocol into an internal native protocol, for example, an internal native protocol of the OCS. The service broker 108 can then analyze the request 106 to determine where the requested data is located. In accordance with an embodiment, the service broker can determine the data location using a cache 110 which stores internal (customer) IDs 112 of data corresponding to external IDs used in the requests.

**[0021]** Data is located on one or more servers, such as server A 114 and server B 116. Customer data is distributed across multiple server machines (first dimension) and across multiple processes (here called ‘server’) within a single server machine (second dimension). A typical process ‘hosts’ multiple partitions. Thus, each server can include a plurality of partitions, which each include a plurality of customer objects. Once the process including the data required to process the request is located (including identifying on which partition of which server the data is stored), then the request can be put into a queue associated with that process. Each queue is a per-server queue, so in this case there would be two queues, queue A 120 and queue B 122. Traffic that is received for a particular server accumulates in the particular server’s associated queue. For example, after the request 106 has been translated, the translated request 124 is placed in queue B 122. Once a trigger event occurs, a batch of requests 126 from queue B are sent to server B 116 for processing.

**[0022]** In accordance with an embodiment of the invention, the server can process the batch of requests 126 sequentially, taking each request in turn and generating a response. When all requests in the batch have been processed by the server the entire batch of responses is sent back 128 to the service broker 108. The service broker 108 then translates the responses from their internal native representations to the network-centric protocol of the requester, and then the translated responses are sent back to the requestor. For example, the service broker 108 can identify the response to translated request 124 and translate the

response back from the internal native protocol to the network-centric protocol of the request. The translated response 130 is then returned.

**[0023]** Additionally, as shown in Figure 1, data from the servers can be synchronously serialized and backed up on another node. In this way the data can be replicated 134 on another node to provide high availability. The data can also be asynchronously persisted 132 in a database. The data which is replicated and/or persisted can include, for example, one or more of the data required to process the request, customer data, and/or event-related data. In the case of node failure, where a server crashes midway through processing a batch, requests included in that batch are quarantined while the backup server comes online. Once the quarantine ends, requests are processed again.

**[0024]** **Figures 2A and 2B** show a comparison of individual request processing and small batch processing, in accordance with an embodiment of the invention. Figures 2A and 2B compare a typical individual request system 200 with a small batch request system 202 in accordance with an embodiment of the invention.

**[0025]** As shown in Figure 2A, the client side 204, for example a service broker (such as a service broker 108), of the individual request system 200 can include a charging processor client 208. When a usage request 212 is received from a network entity 214 (e.g., when a first subscriber calls a second subscriber, or when a subscriber requests to download media), that request 212 is handed off (or request 212 is requested to be handed off) to a separate thread running the charging processor client 208. (Note that there would typically be hundreds of such threads running concurrently in the client 204) The charging processor can translate the request as necessary before forwarding the request to the server side 216 of the individual request system 200.

**[0026]** On the server side 216, an entry processor 218 receives the request 212 and forwards it on to the appropriate charging service 220. A response is then returned in a similar fashion. The response is returned synchronously. Thus, in the individual request system 200 described above with respect to Figure 2A, each request is received, processed and sent to the server as it comes in. Prior systems found this arrangement workable because typical transit time is 1 ms, while server processing time is closer to 50 ms. However, improvements in architecture have reduced server processing time to much less than 1 ms, making transit time a substantial factor. By submitting requests in batches, this transit time can be spread over many requests, greatly reducing the amount of time per message spent on transit.

**[0027]** Figure 2B shows the small batch request system 202. On the client side 222 of the small batch request system 202, requests are similarly received from network entities. However, rather than immediately processing and forwarding these requests to the server side 224, a request batch service 226 aggregates the requests into per-server queues (e.g., queues 120 and 122 as shown in Figure 1). When a triggering event occurs, such as a timeout or the batch is full, a batch of requests is created from the contents of the queue, and charging invocable

client 228 is used to submit the batch of requests to the server side 224. A server side charging invocable module 230 can then process the batch of requests sequentially, forwarding each individual request to an entry processor 232 which can then forward the request to the appropriate charging service 234. When each request in the batch has been processed, the server side charging invocable module 230 can send the batch of requests, now populated with response data, back to the charging invocable client 228 and on to the request batch service 226. The request batch service 226 can then return each response to its requestor.

**[0028]** **Figure 3** shows a sequence diagram of small batch creation, in accordance with an embodiment of the invention. As described above, in accordance with an embodiment of the invention, a plurality of different triggers can be used to send a batch of requests from a queue to its associated server. These triggers can include, but are not limited to, a timeperiod, a maximum queue size, or priority information associated with a request. Once the trigger has been satisfied, then a batch request is created out of the contents of the queue which is sent to the associated server.

**[0029]** As shown in Figure 3, a client 300 (for example a service broker, such as service broker 108) can submit 302 a translated request to a batch request service 304. The batch request service can determine 306 the node on which the data required for processing the usage request is stored and return 308 the name of the node on which the requested data is stored. If the node information returned is invalid e.g. null, for example because the node has crashed, then the system can enter a suspend 310 mode and the requests in its associated queue are quarantined (i.e., taken out of circulation) during the recovery process. Otherwise, as shown at 312, the individual request is inserted into the request queue for the identified node. When a trigger event occurs 314, a batch request 316 is made from the contents of the request queue 318. Once the batch request has been created 320, the batch request is handed off to a 'dedicated thread' which subsequently sends 322 the batch request to the queue's associated server. The 'dedicated thread' is pre-allocated as part of a large thread pool. The dedicated thread handles the entire lifecycle of the batch request from this point on. A batch response is then returned 324.

**[0030]** **Figure 4** shows a sequence diagram of small batch request processing, in accordance with an embodiment of the invention. Figure 4 shows batch request service 404, request queue insert thread 400 and request queue dequeuing thread 412. There are typically many Request Queue insert threads but only one Request Queue dequeuing thread per node. A request queue insert thread 400 can be used to add 402 requests from a batch request service 404 to a request queue. When a request is added, the request queue insert thread 400 can determine if the requested node still exists, if not 406 (e.g., because the server has crashed) suspense management can begin.

**[0031]** The request queue insert thread 400 includes a loop 408 which waits for a trigger to occur. The Request Queue insert thread and Request Queue Dequeue thread can use the

monitor pattern to ensure mutually exclusive access to the request queue. The Request Queue insert thread will wait only if one of several condition variables (triggers) is true. Examples of triggers can include if the batch is full, if a timeout condition is reached, or if an error occurs. If the batch full trigger occurs, then the request queue insert thread sets a condition to batch full 410 and processing continues to the request queue dequeuing thread 412. The request queue dequeuing thread 412 can include a plurality of nested loops, including a first loop 414 monitoring for any error conditions and a second loop 416 waiting for a timeout condition. If an elapsed time is greater than or equal to a timeout time, then the timeout trigger 418 has occurred. Once either the timeout trigger or the batch full trigger have occurred 420, then the requests in the queue are dequeued 422 and a batch request 424 is created. The batch request 424 is then sent 426 to a dedicated batch request thread which sends the batch to the server, waits for a response batch, and then returns the response batch to the service broker.

**[0032]** **Figure 5** shows a method for small batch processing of usage requests, in accordance with an embodiment of the invention. At step 500, a service broker executing on one or more microprocessors is provided. At step 502, a plurality of servers are provided in which each server contains customer data (each of the plurality of servers contains a subset of the data of the customer base). At step 504, a plurality of queues is provided. Each server is associated with a different queue. At step 506, a usage request is received from a network entity. At step 508, an internal ID associated with data requested by the usage request is determined. At step 510, a particular server of the plurality of servers, on which the data requested by the usage request is stored, is determined. At step 512, the usage request is enqueued in a particular queue associated with the particular server. At step 514, upon a trigger event, all requests in the particular queue are sent to the particular server in a batch.

**[0033]** **Figure 6** shows in detail an illustrative service broker 600, in accordance with an embodiment of the invention. The service broker 600 may be one specific embodiment of the service broker 108 as shown in **Figure 1**. However, as understood by those skilled in the art, implementations of a service broker according to the invention are not such limited. For example, the service broker 600 is communicating with a plurality of servers (such as N servers, and in the case of **Figure 1**, two servers, i.e., server A 114 and server 116 B). Each server may include customer data. For example, the service broker 600 is configured to perform small batch processing of usage requests in corporation with the plurality of servers.

**[0034]** As illustrated in **Figure 6,** the service broker 600 comprises a plurality of queue containers (6001-1, 6001-2…6001-N), which are configured to include a corresponding plurality of queues, wherein each server is associated with a different queue. For example, if there are two servers such as server A 114 and server 116 B shown in Figure 1, then there may be two queue containers 6001-1 and 6001-2 (i.e., N=2), wherein the queue containers 6001-1 includes a queue that is associated with server A 114 and the queue containers 6001-2 includes a queue that is associated with server 116 B.

**[0035]** The service broker 600 may further comprise a first receiving interface 6003, configured to receive a usage request from a network entity. The receiving interface 6003 can be any interface that well known in the art or will be known in near future. In one embodiment, the usage request can include a call from one subscriber to another or a download request.

**[0036]** As described with respect to Figure 1, a service broker can determine the data location e.g., using a cache which stores internal (customer) IDs of data corresponding to eternal IDs used in the request. As shown in Figure 6, the service broker 600 further comprises a first determining unit 6004, which is configured to determine an internal ID associated with data requested by the usage request; and a second determining unit 6005, which is configured to determine on which particular server of the plurality of servers the data requested by the usage request is stored. For example, the usage request may request date located on e.g., server A 114, thus the first determining unit 6004 determines an internal ID associated with data requested by the usage request and the second determining unit 6005 determines the data requested by the usage request is stored on server A 114. The first determining unit 6004 and the second determining unit 6005 can be separate units or integrated into one unit. As known to those skilled in the art, the internal (customer) IDs can be stored in other storages other than cache.

**[0037]** Additionally, as shown in Figure 6, the service broker 600 further comprises a queue processing unit 6007, which is configured to enqueue the usage request in a particular queue associated with the particular server. In the above example wherein it is determined the data requested by the usage request is stored on server A 114, the queue processing unit 6007 may enqueue the usage request in the queue 6001-1 associated with the server A 114.

**[0038]** The service broker 600 may further comprise a first sending interface 6009, which is configured to, upon a trigger event, send all requests in the particular queue to the particular server in a batch. As described above, the trigger event may be reaching a predetermined number of requests in a queue, arrival of a specific type of request, arrival of a long running request, and the like. For example, upon the number of usage requests in the queue container 6001-1 reaches a predetermined number, all the requests in the queue container 6001-1 are sent in a batch.

**[0039]** As described above, the service broker can translate the request from a network-centric protocol into an internal native protocol, for example, an internal native protocol of the OCS. As shown in Figure 6, the service broker (600) may further comprise a translator (6011) configured to translate the usage request from a network-centric protocol to an internal protocol.

**[0040]** In the embodiment, each server may process each request in the batch and return to the service broker the batch which has been populated with response data for each request. In such a case, the service broker 600 may further comprise a second receiving interface 6013 configured to, when the particular server has processed each request in the batch and returns the batch, which has been populated with response data for each request, to the service broker,

receive the batch from the particular server. In one embodiment, the translator 6011 may be further configured to translate the response data for each request from the internal protocol to the network-centric protocol, to create translated response data. Correspondingly, the service broker 600 may further comprise a second sending interface 6015 configured to return the translated response data to each requestor.

**[0041]** Those skilled in the art can understand that any of the units in the broker server 600 can be implemented by software, hardware or the combination thereof. The units in the broker server 600 can be separately implemented or integrated in any combination.

**[0042]** Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of application specific integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

**[0043]** The various embodiments include a computer program product which is a storage medium (media) having instructions stored thereon/in which can be used to program a general purpose or specialized computing processor(s)/device(s) to perform any of the features presented herein. The storage medium can include, but is not limited to, one or more of the following: any type of physical media including floppy disks, optical discs, DVDs, CD-ROMs, microdrives, magneto-optical disks, holographic storage, ROMs, RAMs, PRAMS, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs); paper or paper-based media; and any type of media or device suitable for storing instructions and/or information. The computer program product can be transmitted in whole or in parts and over one or more public and/or private networks wherein the transmission includes instructions which can be used by one or more processors to perform any of the features presented herein. The transmission may include a plurality of separate transmissions. In accordance with certain embodiments, however, the computer storage medium containing the instructions is non-transitory (i.e. not in the process of being transmitted) but rather is persisted on a physical device.

**[0044]** The foregoing description of the preferred embodiments of the present invention has been provided for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations can be apparent to the practitioner skilled in the art. The modifications and variations include any relevant combinations of the disclosed features. Embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the relevant art to understand the invention. It is intended that the scope of the invention be defined by the following claims and their equivalents.

**CLAIMS**

What is claimed is:

1. A system for small batch processing of usage requests, comprising:

a service broker executing on one or more microprocessors;

a plurality of servers, wherein each server can include customer data;

a plurality of queues, wherein each server is associated with a different queue;

wherein when a usage request is received from a network entity, the service broker is configured to:

determine an internal ID (Identification) associated with data requested by the usage request,

determine on which particular server of the plurality of servers the data requested by the usage request is stored,

enqueue the usage request in a particular queue associated with the particular server, and

upon a trigger event, send all requests in the particular queue to the particular server in a batch.

2. The system of Claim 1 or 2, wherein the usage request can include a call from one subscriber to another or a download request.

3. The system of Claim 1 or 2, wherein the service broker can further translate the usage request from a network-centric protocol to an internal protocol.

4. The system of any preceding Claims:

wherein the particular server is configured to return the batch to the service broker when the particular server has processed each request in the batch, wherein the batch has been populated with response data for each request.

5. The system of Claim 4, further comprising:

wherein when the service broker receives the batch from the particular server, the service broker is configured to:

translate the response data for each request from an internal protocol to a network-centric protocol, to create translated response data, and

return the translated response data to each requestor.

6. The system of any preceding Claims, wherein data on each server is persisted.

7. The system of any preceding Claims wherein data stored on a first server is made highly available by storing it on a second server.

8. A method for small batch processing of usage requests, comprising:

providing a service broker executing on one or more microprocessors;

providing a plurality of servers, wherein each server includes customer data;

providing a plurality of queues, wherein each server is associated with a different queue;

receiving a usage request from a network entity;

determining an internal ID (Identification) associated with data requested by the usage request;

determining on which particular server of the plurality of servers the data requested by the usage request is stored;

enqueuing the usage request in a particular queue associated with the particular server; and

upon a trigger event, sending all requests in the particular queue to the particular server in a batch.

9. The method of Claim 8, wherein the usage request includes a call from one subscriber to another or a download request.

10. The method of Claim 8 or 9, wherein the service broker further translates the usage request from a network-centric protocol to an internal protocol.

11. The method of any of Claims 8 to 10, further comprising:

when the particular server has processed each request in the batch,

populating the batch with response data for each request, and

returning the batch to the service broker.

12. The method of Claim 11, further comprising:

when the service broker receives the batch from the particular server,

translating the response data for each request from an internal protocol to a network-centric protocol, to create translated response data, and

returning the translated response data to each requestor.

13. The method of any of Claims 8 to 12, wherein data on each server is persisted.

14. The method of any of Claims 8 to 13 wherein data stored on a first server is made highly available by storing it on a second server.

15. A computer program comprising instructions that when executed on a computer system cause the computer system to perform all the steps of the method of any of Claims 8 to 14.

16. A computer program product comprising a non-transitory computer readable storage medium having the computer program of Claim 15 stored thereon.

17. A non-transitory computer readable storage medium including instructions stored thereon which, when executed by a computer, cause the computer to perform the steps of:

providing a service broker executing on one or more microprocessors;

providing a plurality of servers, wherein each server can include customer data;

providing a plurality of queues, wherein each server is associated with a different queue;

receiving a usage request a network entity;

determining an internal ID (Identification) associated with data requested by the usage request;

determining on which particular server of the plurality of servers the data requested by the usage request is stored;

enqueuing the usage request in a particular queue associated with the particular server; and

upon a trigger event, sending all requests in the particular queue to the particular server in a batch.

18. The non-transitory computer readable storage medium of Claim 17, wherein the usage request can include a call from one subscriber to another or a download request.

19. The non-transitory computer readable storage medium of Claim 17 or 18, wherein the service broker can further translate the usage request from a network-centric protocol to an internal protocol.

20. The non-transitory computer readable storage medium of any one of Claims 17 to 19 wherein when the particular server has processed each request in the batch, the particular server is configured to return the batch to the service broker, wherein the batch has been populated with response data for each request.

21. The non-transitory computer readable storage medium of Claim 20, further comprising:

when the service broker receives the batch from the particular server,

translating the response data for each request from the internal protocol to the network-centric protocol, to create translated response data, and

returning the translated response data to each requestor.

22. The non-transitory computer readable storage medium of any one of Claims 17 to 21, wherein data on each server is persisted, and wherein data stored on a first server is made highly available by storing it on a second server.

23. A service broker (600) for communicating with a plurality of servers each including customer data to perform small batch processing of usage requests, comprising:

a plurality of queue containers (6001-1, 6001-2 and 6001-N), configured to include a corresponding plurality of queues, wherein each server is associated with a different queue;

a first receiving interface (6003), configured to receive a usage request from a network entity,

a first determining unit (6004), configured to determine an internal ID associated with data requested by the usage request,

a second determining unit (6005), configured to determine on which particular server of the plurality of servers the data requested by the usage request is stored,

a queue processing unit (6007), configured to enqueue the usage request in a particular queue associated with the particular server, and

a first sending interface (6009), configured to, upon a trigger event, send all requests in the particular queue to the particular server in a batch.

24. The service broker (600) of Claim 23, wherein the usage request can include a call from one subscriber to another or a download request.

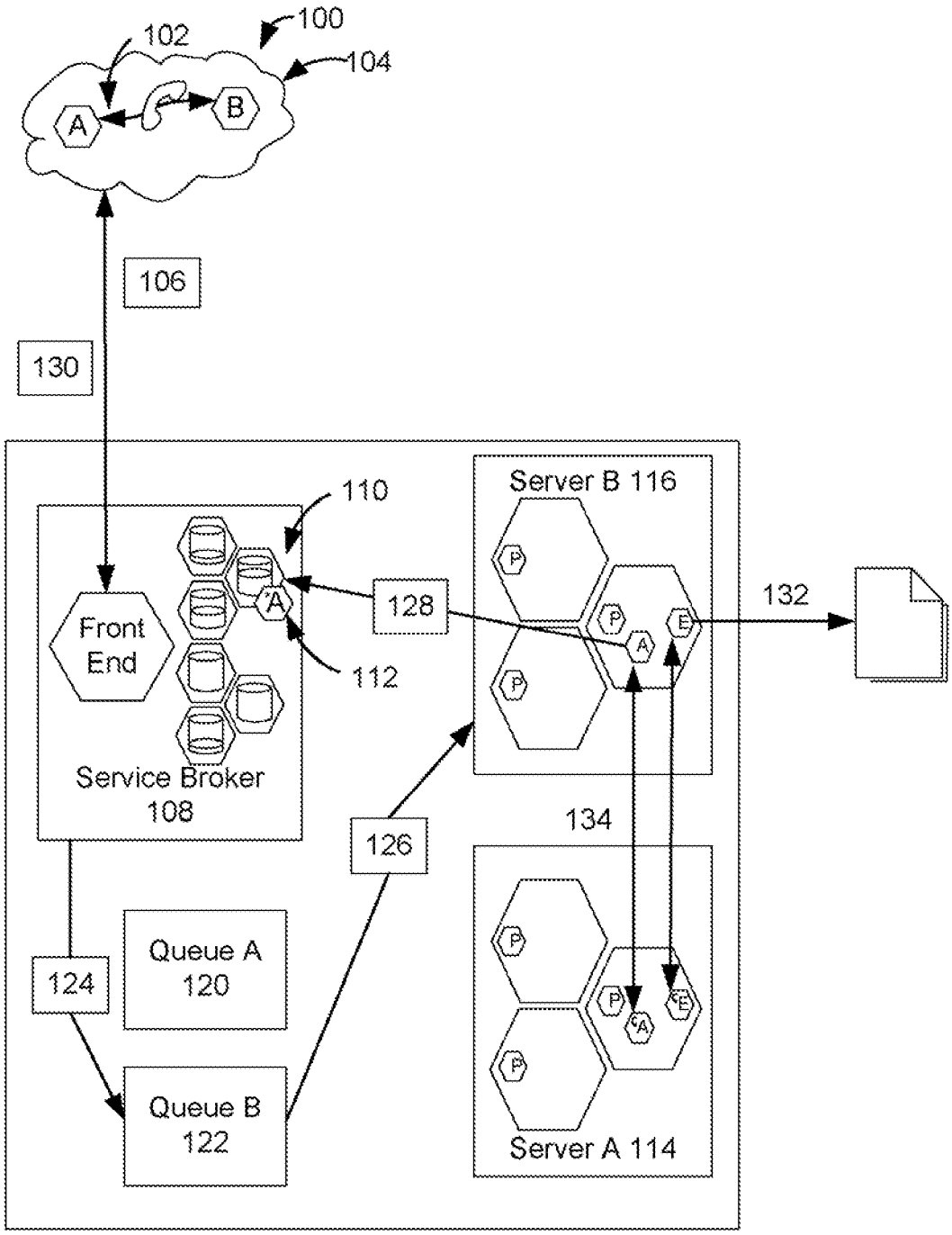
25. The service broker (600) of Claim 23, wherein the service broker further comprises a translator (6011) configured to translate the usage request from a network-centric protocol to an internal protocol.

26. The service broker (600) of Claim 23, further comprising a second received interface (6013) configured to, when the particular server has processed each request in the batch and returns the batch, which has been populated with response data for each request, to the service broker, receive the batch from the particular server.

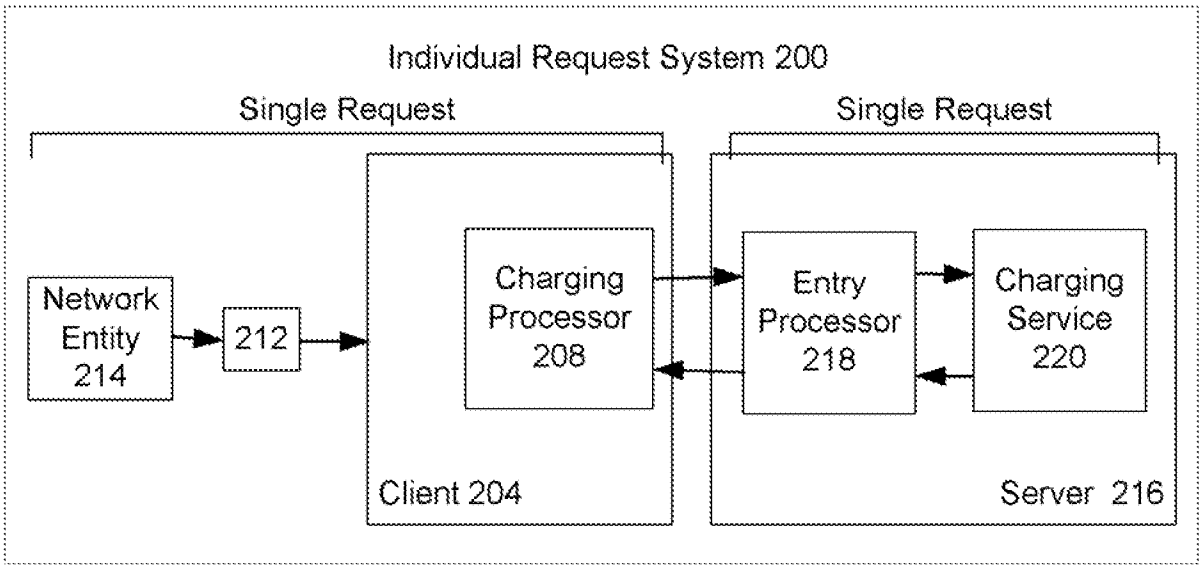
27. The service broker (600) of Claim 23, wherein the translator (6011) is further configured to translate the response data for each request from the internal protocol to the network-centric protocol, to create translated response data, and

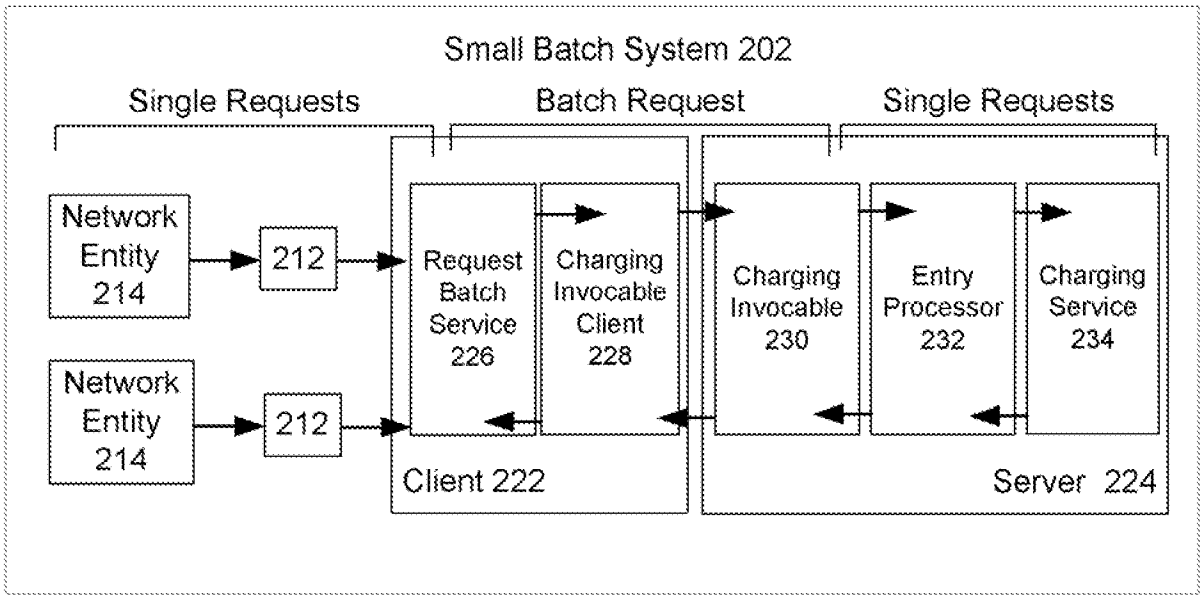
the service broker (600) further comprises a second sending interface (6015) configured to return the translated response data to each requestor.

# Figure 1


102
100
104
A
B
106
130
110
Server B 116
Front End
A
128
132
112
Service Broker 108
126
134
124
Queue A 120
Queue B 122
Server A 114

# Figure 2A




# Figure 2B

# Figure 3

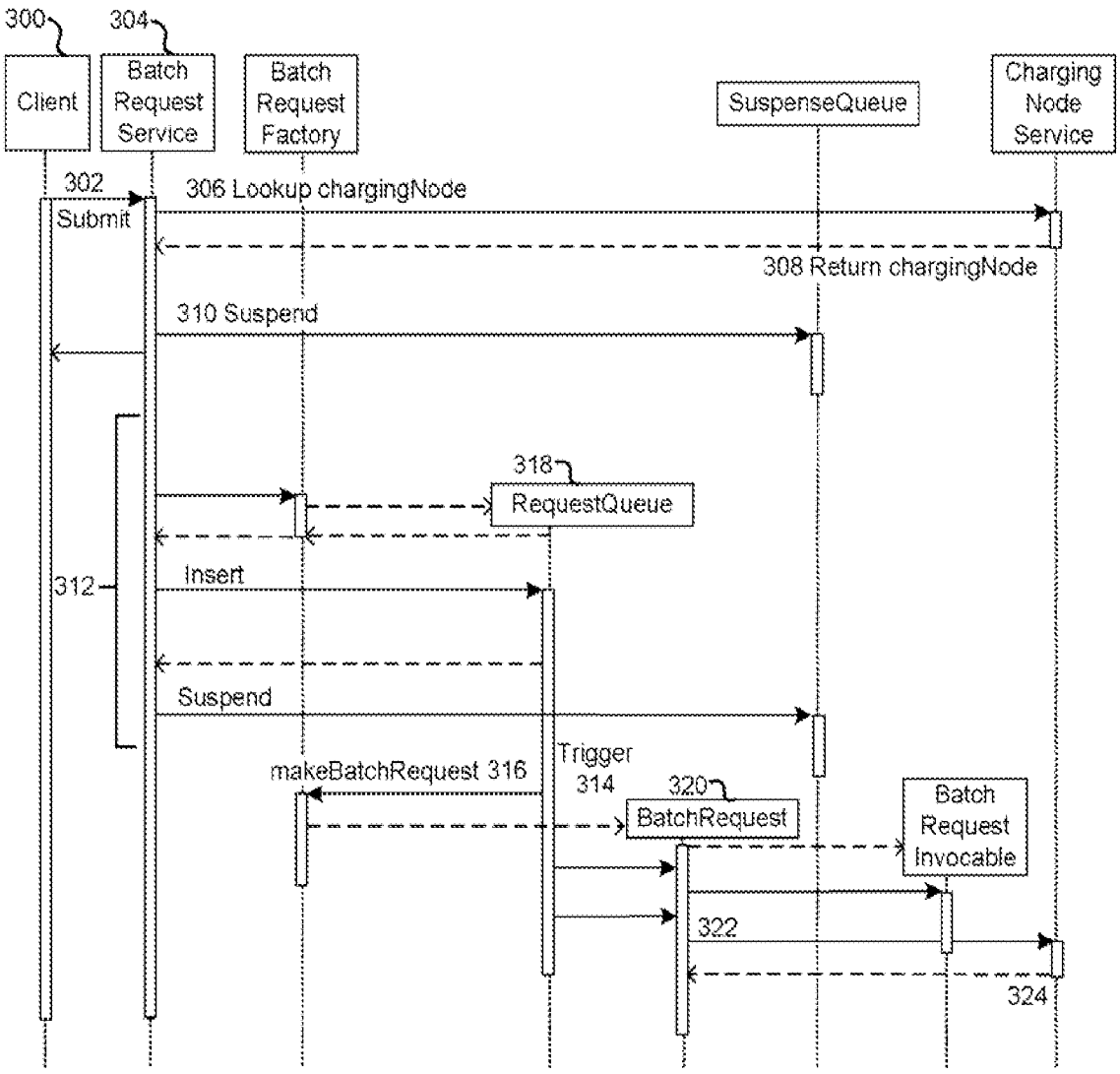
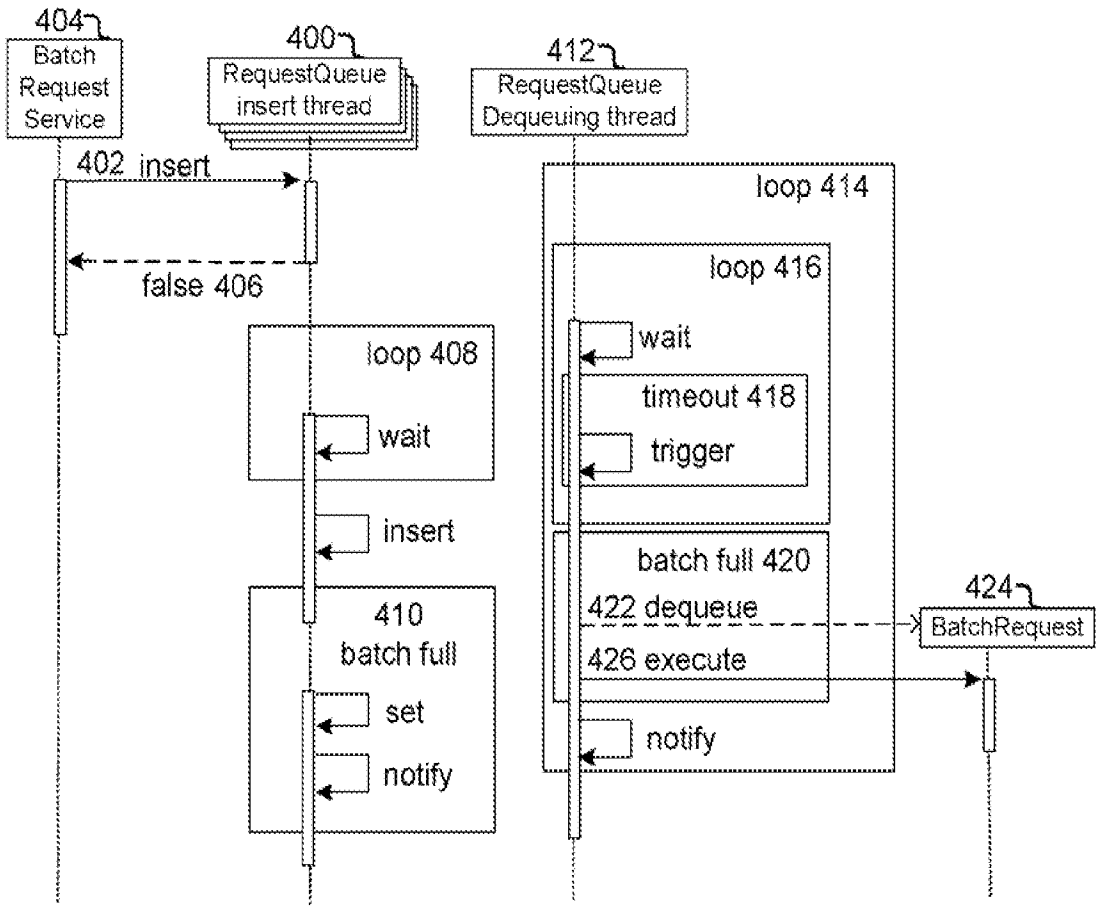

300
304
Client
Batch Request Service
Batch Request Factory
SuspenseQueue
Charging Node Service
302
Submit
306 Lookup chargingNode
308 Return chargingNode
310 Suspend
318
RequestQueue
312
Insert
Suspend
Trigger 314
makeBatchRequest 316
320
BatchRequest
Batch Request Invocable
322
324

**Figure 4**


404
Batch Request Service
400
RequestQueue insert thread
412
RequestQueue Dequeuing thread
402 insert
false 406
loop 414
loop 416
wait
timeout 418
trigger
loop 408
wait
insert
batch full 420
422 dequeue
426 execute
424
BatchRequest
410
batch full
set
notify
notify

# Figure 5

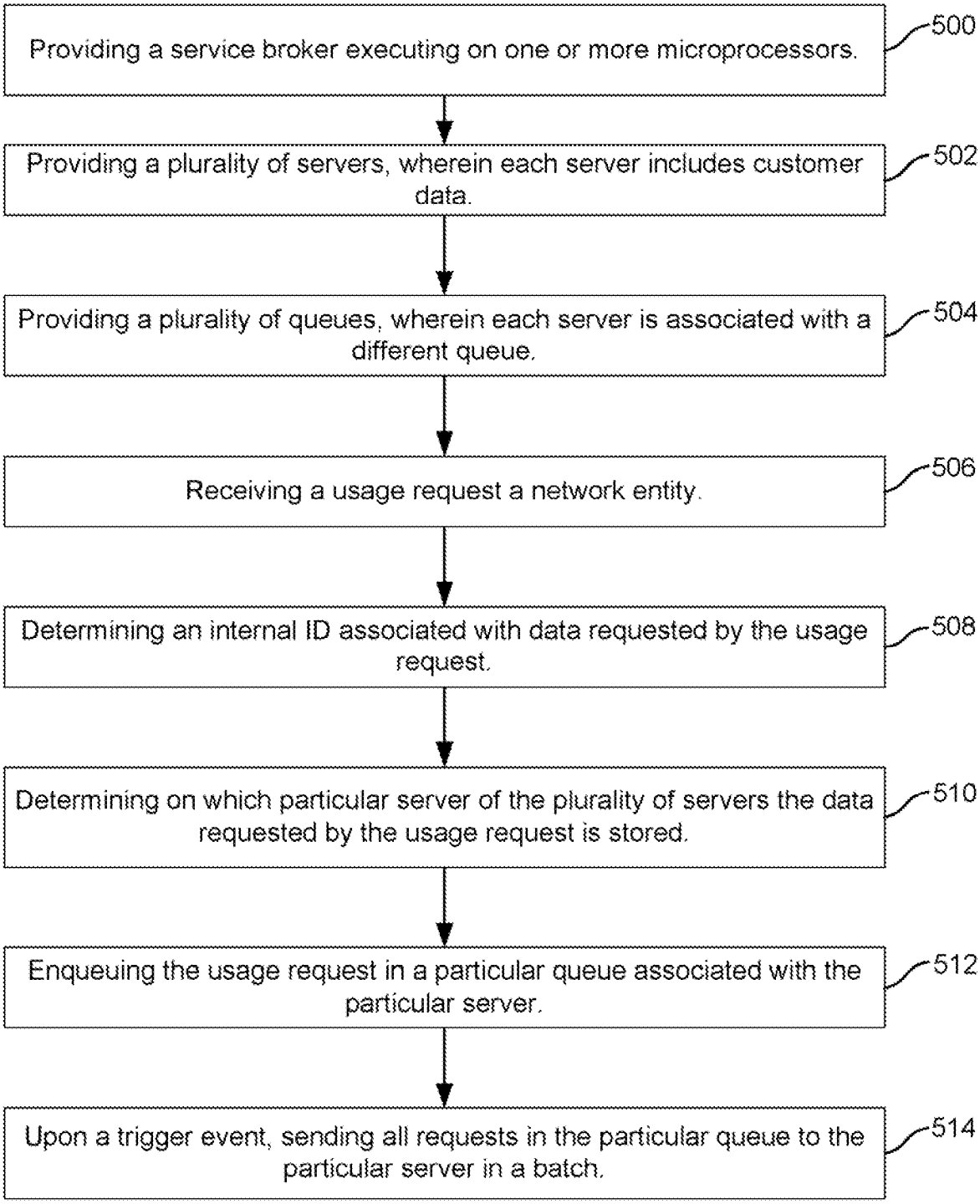
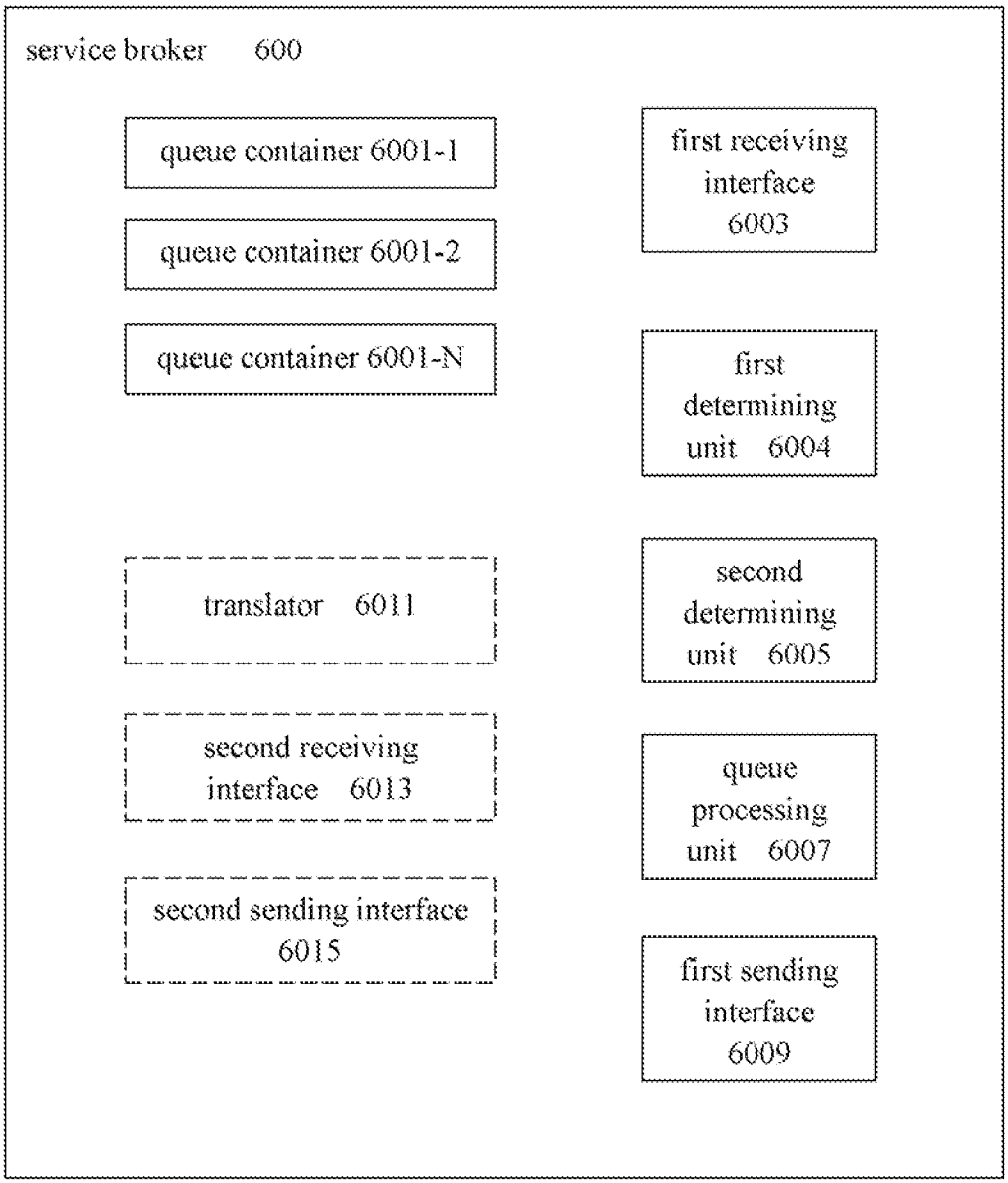

Providing a service broker executing on one or more microprocessors.
500
Providing a plurality of servers, wherein each server includes customer data.
502
Providing a plurality of queues, wherein each server is associated with a different queue.
504
Receiving a usage request a network entity.
506
Determining an internal ID associated with data requested by the usage request.
508
Determining on which particular server of the plurality of servers the data requested by the usage request is stored.
510
Enqueuing the usage request in a particular queue associated with the particular server.
512
Upon a trigger event, sending all requests in the particular queue to the particular server in a batch.
514

**Figure.6**


service broker 600
queue container 6001-1
queue container 6001-2
queue container 6001-N
translator 6011
second receiving interface 6013
second sending interface 6015
first receiving interface 6003
first determining unit 6004
second determining unit 6005
queue processing unit 6007
first sending interface 6009