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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 January 2016 (28.01.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/015039 A1

(51) International Patent Classification:
H04L 29/06 (2006.01) *G06F 9/54* (2006.01)
H04L 1/16 (2006.01)

(21) International Application Number:
PCT/US2015/042181

(22) International Filing Date:
27 July 2015 (27.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/341,592 25 July 2014 (25.07.2014) US

(71) Applicant: **NETAPP, INC.** [US/US]; 495 East Java Drive, Sunnyvale, California 94089 (US).

(72) Inventors: **JOSHI, Sandeep**; c/o NetApp, 3rd Floor, Fair Winds Block, EGL Software Park, Off Intermediate Ring Road, Bangalore 560071 (IN). **MATHUR, Ankit**; c/o NetApp, 3rd Floor, Fair Winds Block, EGL Software Park, Off Intermediate Ring Road, Bangalore 560071 (IN). **PANDA, Sudip Kumar**; c/o NetApp, 3rd Floor, Fair Winds Block, EGL Software Park, Off Intermediate Ring Road, Bangalore 560071 (IN).

(74) Agents: **PAVENTO, Michael S.** et al.; Kilpatrick Townsend & Stockton LLP, Suite 2800, 1100 Peachtree Street, Atlanta, Georgia 30309 (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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, 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, ST, SZ,

[Continued on next page]

(54) Title: ASYNCHRONOUS COMMUNICATIONS HAVING COMPOUNDED RESPONSES

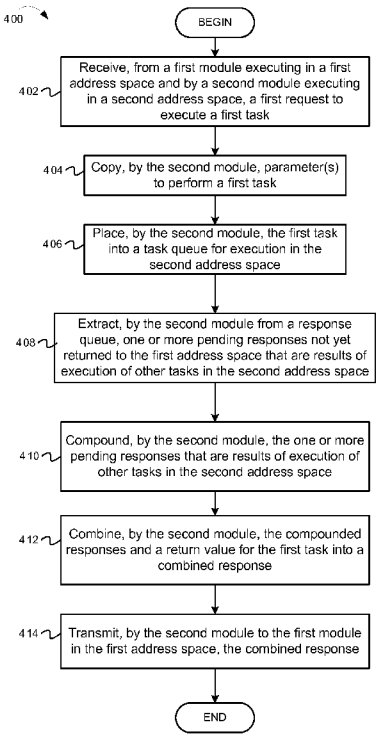

FIG. 4

(57) Abstract: A first request to execute a first task is received from a first module in a first address space and by a second module in a second address space. The first task is placed into a task queue for execution in the second address space. Pending responses not yet returned to the first module that are results of execution for other tasks in the second address space are extracted by the second module from a response queue. Requests for the other tasks were previously sent by the first module to the second module for execution in the second address space. The pending responses are compounded. The pending responses and a return value for acknowledgement the first request to execute the first task are combined, by the second module into a combined communication. The combined communication is transmitted by the second module to the first module in the first address space.

WO 2016/015039 A1

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

Published:

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

ASYNCHRONOUS COMMUNICATIONS HAVING COMPOUNDED RESPONSES

BACKGROUND

[0001] Features generally relate to the field of computers, and, more particularly, to asynchronous communications having compounded responses.

[0002] Remote Procedure Call (RPC) is an inter-process communication that enables a procedure to execute in a different address space. The different address space can be in a same or different computer. Also, the programmer of the procedure is not required to code the instructions for this remote interaction with the different address space. Therefore, the same code can essentially be written for the procedure executing locally or in the different address space.

[0003] Usage of RPC abstracts the programmer from network programming, network errors, different character encoding and word length of computers in a network. RPC also allows running a specialized task (as a service) on a computer such that the task can be used by other computers in the network.

[0004] RPC can be synchronous or asynchronous. For synchronous RPC, a remote procedure is called and the calling routine blocks until the remote procedure returns a result. In many scenarios, the remote procedure can be time consuming and this leads to the blocking of thread/ system-resources by the caller (or client). To avoid this blocking, many applications create two separate channels for asynchronous RPC. One channel is for sending the requests, and other channel is for receiving the responses. Each procedure can be identified by a unique identification. RPC that receives the request queues the request for processing and sends an acknowledgement to the client. When the procedure has completed execution, a response is sent to caller with the unique request identification along with the response.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The features of the disclosure may be better understood by referencing the accompanying drawings.

[0006] Figure 1 depicts a system having asynchronous communications that includes compounded responses, according to some features.

[0007] Figure 2 depicts an exchange of asynchronous communications that includes compounded responses, according to some features.

[0008] Figure 3 depicts an example compounded response that is included with an acknowledgement to a task request, according to some features.

[0009] Figure 4 depicts a flowchart of operations performed by a task server that includes compounded responses included with an acknowledgment to a task request, according to some features.

[0010] Figure 5 depicts an example compounded response that is included with a response to a task request, according to some features.

[0011] Figure 6 depicts a flowchart of operations performed by a task server that includes compounded responses included with a response to a task request, according to some features.

DETAILED DESCRIPTION OF EXAMPLE ILLUSTRATIONS

[0012] The description that follows includes example systems, methods, techniques, instruction sequences and computer program products that that include techniques of the features described herein. However, it is understood that the described features may be practiced without these specific details. For instance, although examples refer to RPC communication, communication based on other types of protocols can incorporate the features described herein. In other instances, well-known instruction instances, protocols, structures and techniques have not been shown in detail in order not to obfuscate the description.

[0013] As described above, RPC communications can include a request and a response. A request from a first module (e.g., process) executing in a first address space can be sent to a second module (e.g., process) executing in a second address space. Also, receipt of the request can be acknowledged. The request can be for execution of a task by the second module. A task can be a function, subroutine, method or some other unit of code. After the task is complete, the second module can return a response. The responses are typically small. For example, the responses can include a request identification and a response code. In a number of situations, the responses to multiple requests can accumulate together in the

second address space. The second module can be multi-threaded and executing on a separate server or computer. Accordingly, many of the requested tasks can get completed at or near the same time. Also, the requests and responses can be on two separate communication channels between the two address spaces. These two separate communication channels can communicate at differing speeds.

[0014] According to some features, responses in the second address space that have not yet been sent back to the first module that made the requests can be compounded together to create a compounded response. Also, this compounded response can piggyback onto a communication that includes an acknowledgement to a different request. In particular, the compounded response (that includes a number of responses) is combined with an acknowledgement for a request of a task that has not yet been completed. According to some features, the communications between the first address space and the second address space can be based on a communication protocol in an application layer of a multi-layered communication stack (e.g., RPC communications).

[0015] Therefore using some features described herein, there is no need to have a one-to-one relationship between an RPC request communication and an RPC response communication. In other words using some features described herein, there is no need to send a response for each request when pending responses for multiple requests are to be sent back to the first address space. Accordingly, these features can enable avoidance of the network latency for each request. Also, some features reduce the number of read/write context switches in the two address spaces, thereby improving performance of the system.

[0016] Figure 1 depicts a system having asynchronous communications that includes compounded responses, according to some features. In Figure 1, a system 100 includes a first address space 106 and a second address space 126. Each of the first address space 106 and the second address space 126 can include a range of addresses that are accessible by a process through an operating system. The addresses for an address space can be virtual or physical. Also, multiple address spaces can be within a same or different computer. In the example depicted by the system 100 illustrated in Figure 1, the first address space 106 and the second address space 126 are in different computers. The first address space 106 is in a computer 102. The second address space 126 is in a computer 104.

[0017] The computer 102 includes a processor 108, machine-readable media 112, and a network interface 114 that are communicatively coupled together through a communication bus 116. The machine-readable media 112 can be volatile or non-volatile storage media. In this example, the machine-readable media 112 is configured to store a first module 110. The first module 110 can be software, hardware, firmware, or a combination thereof. For example, the first module 110 can be software executing on the processor 108. The network interface 114 can be software, hardware, firmware, or a combination thereof used for communication.

[0018] The computer 104 includes a processor 128, machine-readable media 132, and a network interface 134 that are communicatively coupled together through a communication bus 136. The machine-readable media 132 can be volatile or non-volatile storage media. In this example, the machine-readable media 132 is configured to store a second module 130, a task queue 160, and a response queue 162. The second module 130 can be software, hardware, firmware, or a combination thereof. For example, the second module 130 can be software executing on the processor 128. The network interface 134 can be software, hardware, firmware, or a combination thereof used for communication.

[0019] In the system 100, two channels of communication are created for communication between the computer 102 and the computer 104. A channel 150 is used for communication from the computer 102 to the computer 104. A channel 152 is used for communication from the computer 104 to the computer 102. According to some features, the computer 102 and the computer 104 can communication with each other using asynchronous RPC communications through the channel 150 and the channel 152. In this example as further described below, the first module 110 can transmit a request over the channel 150 to the computer 104. The request can be a request to perform execution of a task in the second address space 126. The second module 130 can receive the request and store the request in the task queue 160. Also in response to receiving the request, the second module 130 can transmit back an acknowledgement of receiving the request. Because the communications are asynchronous, the first module 110 can transmit additional requests for execution of tasks prior to receiving a response for the request. Also, after a task for a request has completed execution, the second module 130 can store the response (e.g., result) of the task in the response queue 162.

[0020] According to some features, responses in the response queue 162 in the second address space 126 that have not yet been sent back to the first module 110 in the first address space 106 can be compounded together to create a compounded response. Also, this compounded response can piggyback onto a communication that includes an acknowledgement to a different request. In particular, the compounded response (that includes a number of responses) can be combined with an acknowledgement for a request of a task that has not yet been completed. According to some features, this compounded response can piggyback onto a response to a different task request. Accordingly, multiple responses can be included with an acknowledgement to a task request not yet executed and/or with a different response after the task has been completed.

[0021] To illustrate, Figure 2 depicts an exchange of asynchronous communications that includes compounded responses, according to some features. Figure 2 depicts example asynchronous communications between the computer 102 and the computer 104 depicted in Figure 1. As described above, the computer 102 and the computer 104 are in different address spaces. In this example, the computer 102 includes a module or process (e.g., the first module 110 of Figure 1) that is transmitting requests for execution of tasks in the address space that includes the computer 104. The computer 104 receives and processes the requests (as now described).

[0022] As shown, the computer 102 transmits a send_task_ request 206 to the computer 104. With reference to Figure 1, the first module 110 transmits the send_task_ request 206 over the channel 150 to the computer 104.

[0023] After receiving the send_task_request 206, the computer 104 executes a send_task_request function 210. With reference to Figure 1, the second module 130 executes the send_task_request function 210. Execution of the send_task_ request function 210 can cause the computer 104 to transmit an acknowledgement (an acknowledgement of send_task_request 208) back to the computer 102. With reference to Figure 1, the second module 130 transmits the send_task_ request function 210. In this example, the acknowledgement is the acknowledgement of a send_task_request 208. The return value can be an alphanumeric value that uniquely identifies this task request (the send_task_request 206) relative to other tasks received by the computer 104. One or more responses that are in the response queue 162 in the second address space 126 that have not yet been sent back to the first module 110 in the first address space can be compounded together to create a

compounded response. The second module 130 can include this compounded response with the acknowledgement. Thus, the acknowledgement of the send_task_request 206 can include the acknowledgement along with a compounded response. An example of the acknowledgement along with a compounded response is depicted in Figure 3, which is described in more detail below.

[0024] Execution of the send_task_request function 210 can also cause the computer 104 to initiate execution of a task that is being requested by the send_task_request 206 – task execution 212. With reference to Figure 1, the second module 130 can initiate another process in the computer 104 to perform the task. The task can any type of operation that occurs in the address space 126. For example, the task can include moving data, creating new data, deleting data, performing an Input/Output (I/O) operation, etc. Although not shown in Figure 2, execution of the send_task_request function 210 can also cause the computer 104 to store the task into the task queue 160.

[0025] After the task execution 212 is complete, the computer 104 transmits a send_task_response 214 back to the computer 102. With reference to Figure 1, the second module 130 can transmit the send_task_response 214 back to the computer 102. One or more other responses that are in the response queue 162 in the second address space 126 that have not yet been sent back to the first module 110 in the first address space can be compounded together to create a compounded response. Thus, the first module 110 can execute the send_task_response 214 to extract responses from the response queue 162 and compound them together. An example of the response along with a compounded response is depicted in Figure 5, which is described in more detail below.

[0026] After receiving the send_task_response 214, the computer 102 executes a send_task_response function 218. With reference to Figure 1, the first module 110 can execute the send_task_response function 218. Execution of the send_task_response 218 can cause the first module 110 to transmit an acknowledgement (an acknowledgement 216 of send_task_response 214) back to the computer 104. Execution of the send_task_response function 218 can also cause the first module 110 to initiate one or more other processes in the first address space 106. For example, one or more other processes can perform operation in response to completion of the tasks in the second address space 126 that are identified by the one or more responses that are included in the send_task_response 214. Figure 2 depicts communications between the first address space 106 and the second address space 126 for

one task request/response. These communications can occur for multiple task requests/responses.

[0027] Figure 3 depicts an example compounded response that can be included with an acknowledgement to a task request, according to some features. Figure 3 depicts an acknowledgement communication 300 that includes an acknowledgement to receiving a task request. The acknowledgement communication 300 can be an example of the acknowledgement of send_task_request 208 depicted in Figure 2. As shown, the acknowledgement communication 300 includes an acknowledgement return value for acknowledgement of a received request for task execution 302 (the acknowledgement return value 302). According to some features, in addition to the acknowledgement return value 302, responses to completed tasks can be piggybacked onto the acknowledgement communication 300.

[0028] In this example, the acknowledgement communication 300 includes a compounded response 314. The compounded response 314 can include one or more responses (for tasks that have completed execution) that have not yet been sent back to the first address space to notify the first module in the first address space that task(s) have been completed in the second address space. For example with reference to Figure 1, the second module 130 can retrieve one, some, or all of the responses residing in the response queue 162. The second module 130 can combine these responses to form the compounded response 314. In this example, the compounded response 314 includes five responses – a response 304, a response 306, a response 308, a response 310, and a response 312. As shown, each of the response 304, the response 306, the response 308, the response 310, and the response 312 include a request ID and a response code. The request ID includes an identification of the request that was transmitted to the second address space to initiate execution of a task. Thus, the response that is returned to the first address space can be associated with the request that causes execution of a task that created this response. The response code can include a value that includes a result of execution of the task. For example, one value of a response code can notify that the task was successfully executed. In another example, another value of a response code can notify that the task was not successfully executed. In another example, another value of a response code can be some data returned as a result of execution of the task.

[0029] To further illustrate, Figure 4 depicts a flowchart of operations performed by a task server that includes compounded responses included with an acknowledgment to a task request, according to some features. Figure 4 depicts a flowchart 400. The operations depicted in the flowchart 400 are described in reference to Figures 1-3. Operations of the flowchart 400 start at block 402.

[0030] At block 402, a first request to execute a first task is received from a first module executing in a first address space and by a second module executing in a second address space. With reference to Figure 1, the second module 130 executing in the second address space 126 can receive a first request to execute a first task from the first module 110 in the first address space 106 over the channel 150. With reference to Figure 2, the first request can be the send_task_ request 206. Operations of the flowchart 400 continue at block 404.

[0031] At block 404, parameter(s) to perform the first task are copied from the first request by the second module. With reference to Figure 1, the second module 130 can copy the parameters to perform the first task. For example, the first request can be a function that includes one or more parameters. Below is example pseudocode for the first request:

```
SendReqReturnVal_t    send_task_request (int cookie,  parameter1,
parameter2, …)
{
    Copy the parameters to perform the task.
    Place the task into task queue.
    Extract response(s).
    Compound the response(s).
    Combine the return value and the compounded response(s).
    Return compounded response(s) and along with the return value.
}
```

[0032] In this example pseudocode for the first task request, two parameters are passed with the function – parameter1 and parameter2. As shown in the first line of the pseudocode above, execution of the function send_task_request causes the parameters to perform the first task to be copied. The second module 130 can copy the parameter(s) for the first task request into machine-readable media in the second address space 126. The pseudocode shown above for the function send_task_request can also pass a "cookie" into the second address space 126. The "cookie" can be a value that uniquely identifies the task request sent by the

computer 102. The use of the "cookie" is further described below. Operations of the flowchart 400 continue at block 406.

[0033] At block 406, the first task is placed into a task queue for execution in the second address space by the second module. With reference to Figure 1, the second module 130 can place the first task (identified by the first task request) into the task queue 160 in the second address space 126. Accordingly, tasks can be removed from the task queue 160 for execution according to an order that the tasks are placed into the task queue 160. With reference to the pseudocode above for the send_task_ request, this operation is depicted by the second line of code. Operations of the flowchart 400 continue at block 408.

[0034] At block 408, one or more pending responses not yet returned to the first address space are extracted from the response queue by the second module. With reference to Figure 1, the second module 130 can extract one or more pending responses from the response queue 162. For example, the second module 130 can retrieve one, some, or all of the responses residing in the response queue 162. A pending response can be stored in the response queue 162 after the associated task has completed execution in the second address space 126. Operations of the flowchart 400 continue at block 410.

[0035] At block 410, the one or more pending responses that are results of execution of other tasks in the second address space are compounded together by the second module. With reference to Figure 1, the second module 130 can compound the one or more pending responses together. With reference to Figure 3, an example of the compounded responses is depicted by the compounded response 314, which includes five responses. Operations of the flowchart 400 continue at block 412.

[0036] At block 412, the compounded response and a return value for the first task is combined into a combined response. With reference to Figure 1, the second module 130 can combine the compounded responses with the return value. Returning to the pseudocode above for send_task_request, a new data type (new typedef) has been defined SendReqReturnVal:

[0037]

```
typedef struct cookieResp {
        int cookie;
        int response;
```

```
}cookieResp_t;


typedef struct SendReqReturnVal {
        int retVal;
        list<cookieResp> compoundedResp
} SendReqReturnVal_t;


SendReqReturnVal_t   send_task_request (int cookie, parameter1,
parameter2);
```

[0038] Because of the inclusion of the compounded responses with the return value for the acknowledgement of a task request, a new data type for the send_task_request has been defined. In particular, a new data type has been defined - SendReqReturnVal_t, which defines a new type definition for what is returned as the response for execution of send_task_request. As shown by the new data type defined above, the data type SendReqReturnVal_t returns a return value (retVal) that is the acknowledgement return value and a compounded response (compoundedResp) which includes a list of one or more responses. Thus, instead of the return value being an integer value as a response to execution of the send_task_request, a new data type has been defined to return the return value and a compounded response with one or more responses that are a result of completion of execution of other tasks in the second address space (which have not yet been returned to the first address space). Operations of the flowchart 400 continue at block 414.

[0039] At block 414, the combined response is transmitted by the second module to the first module in the first address space. With reference to Figure 1, the second module 130 can transmit the combined response back to the first address space 106 through the channel 152. Operations of the flowchart 400 are complete.

[0040] In addition to sending back combined responses with an acknowledgement, some feature can send back combined responses with a response that is queued to be sent back to the first address space. To illustrate, Figure 5 depicts an example compounded response that is included with a response to a task request, according to some features. Figure 5 depicts a compounded response 500. The compounded response 500 can be an example of the

send_task_response 214 depicted in Figure 2. With reference to Figure 1, the compounded response 500 can include one, some or all of the pending responses in the response queue 162 that have not yet been returned to the first address space 106.

[0041] In this example, the compounded response 500 includes five responses that have not yet been returned to the first address space 106. Accordingly, instead of returning one response for each communication from the second address space 126 to the first address space 106, according to some features multiple responses can be combined as a communication. The compounded response 500 includes a response 504, a response 506, a response 508, a response 510, and a response 512. Assume that the response 504 is at the top of the response queue 162. Conventional RPC operations typically would just return the response 504. However, some features combine other responses to be included with the response 504 that is returned to the first address space 106. In this example, four additional responses are included in the compounded response 500 that are returned – the response 506, the response 508, the response 510, and the response 512. For example, the response 506, the response 508, the response 510, and the response 512 can be the remaining responses in the response queue 162. In another example, the response 506, the response 508, the response 510, and the response 512 can be the next four responses in the response queue 162 to be returned, thereby leaving some responses in the response queue 162.

[0042] As shown, each of the response 504, the response 506, the response 508, the response 510, and the response 512 include a request ID and a response code. As described above, the request ID includes an identification of the request that was transmitted to the second address space to initiate execution of a task. Thus, the response that is returned to the first address space can be associated with the request that causes execution of a task that created this response. The response code can include a value that includes a result of execution of the task. For example, one value of a response code can notify that the task was successfully executed. In another example, another value of a response code can notify that the task was not successfully executed. In another example, another value of a response code can be some data returned as a result of execution of the task.

[0043] To further illustrate, Figure 6 depicts a flowchart of operations performed by a task server that includes compounded responses included with a response to a task request, according to some features. Figure 6 depicts a flowchart 600. The operations depicted in the

flowchart 600 are described in reference to Figures 1-2 and 5. Operations of the flowchart 600 start at block 602.

[0044] At block 602, a first request to execute a first task is received from a first module executing in a first address space and by a second module executing in a second address space. With reference to Figure 1, the second module 130 executing in the second address space 126 can receive a first request to execute a first task from the first module 110 in the first address space 106 over the channel 150. With reference to Figure 2, the first request can be the send_task_request 206. Operations of the flowchart 600 continue at block 604.

[0045] At block 604, execution of the first task is completed to create a first response. With reference to Figure 1, the second module 130 or some other module can execute the first task. For example with reference to Figure 1, the second module 130 can initiate another module in the computer 104 to perform the task. The task can be any type of operation that occurs in the address space 126. For example, the task can include moving data, creating new data, deleting data, performing an Input/Output (I/O) operation, etc. After execution of the first task is completed, operations of the flowchart 600 continue at block 606.

[0046] At block 606, one or more pending responses not yet returned to the first address space are extracted from the response queue. With reference to Figure 2, the computer 104 can execute a send_task_response 214 in the second address space. With reference to Figure 1, the second module 130 can execute the send_task_response 214. Below is example pseudocode for the send_task_response 214:

```
int send_task_response(list<cookieResp_t>)
   {
    Remove responses from response queue
    Compound responses together
    Return Success/Failure.
   }

typedef struct cookieResp {
    int cookie;
    int response;
    }cookieResp_t;

int send_task_response(list<cookieResp_t>);
```

[0047] As shown by the first two lines of pseudocode of send_task_response provided above, one or more other responses that are in the response queue 162 in the second address space 126 that have not yet been sent back to the first module 110 in the first address space can be extracted from the response queue 162 and compounded together to create a compounded response. Thus, the second module 130 can remove the responses from the response queue 162. Operations of the flowchart 600 continue at block 608.

[0048] At block 608, the one or more pending responses that are results of execution of other tasks in the second address space are compounded together by the second module. As described above with reference to Figures 1-2, the second module 130 can compound the one or more pending responses together based on execution of the send_task_response. Execution of the send_task_response 218 can also cause the first module 110 to initiate one or more other processes in the first address space 106. For example, one or more other processes can perform operation in response to completion of the tasks in the second address space 126 that are identified by the one or more responses that are included in the send_task_response 218. Operations of the flowchart 600 are complete.

[0049] As will be appreciated by one skilled in the art, some features may be a system, method or computer program product. Accordingly, some features may be entirely hardware, entirely software (including firmware, resident software, micro-code, etc.) or some combination of software and hardware that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, some features may take the form of a computer program product in one or more computer readable medium(s) having computer readable program code included thereon.

[0050] Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage

device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

[0051] A computer readable signal medium may include a propagated data signal with computer readable program code included therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

[0052] Program code included on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

[0053] Computer program code for carrying out operations for some features may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[0054] Some features are described with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other

programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0055] These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

[0056] The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0057] While features are described with reference to various implementations and exploitations, it will be understood that these features are illustrative and that the scope of the features is not limited to them. In general, techniques for asynchronous communications having compounded responses as described herein may be implemented with facilities consistent with any hardware system or hardware systems. Many variations, modifications, additions, and improvements are possible.

[0058] Plural instances may be provided for components, operations or structures described herein as a single instance. Finally, boundaries between various components, operations and data stores are somewhat arbitrary, and particular operations are illustrated in the context of specific illustrative configurations. Other allocations of functionality are envisioned and may fall within the features described herein. In general, structures and functionality presented as separate components in the exemplary configurations may be implemented as a combined structure or component. Similarly, structures and functionality presented as a single component may be implemented as separate components. These and other variations, modifications, additions, and improvements may fall within the features described herein.

CLAIMS

1. A method comprising:

receiving, from a first module executing in a first address space and by a second module in a second address space through a first channel, a first request to execute a first task;

placing, by the second module, the first task into a task queue for execution in the second address space;

extracting, by the second module from a response queue, pending responses not yet returned to the first module that are results of execution of other tasks in the second address space, wherein requests for the other tasks were previously sent by the first module to the second module for execution in the second address space;

compounding, by the second module, the pending responses that are the results of execution of the other tasks in the second address space to form a compounded response;

combining, by the second module, the compounded response and a return value for acknowledgement of the first request to execute the first task into a combined communication; and

transmitting, by the second module to the first module in the first address space, the combined communication.

2. The method of claim 1, further comprising:

copying, by the second module, at least one parameter to perform the first task, after receiving the first request to execute the first task.

3. The method of claim 1 or 2, wherein transmitting the combined communication occurs prior to completion of the first task.

4. The method of any preceding claim, wherein transmitting the combined communication is through a second channel.

5. The method of any preceding claim, further comprising completing, by the second module, execution of the first task.

6. The method of claim 5, further comprising:

extracting from the response queue other pending responses that are results of execution of additional other tasks in the second address space; and

compounding the other pending responses and a response for the first task that is the result of execution of the first task by the second module.

7. The method of any preceding claim, wherein the first request and the combined communication are based on a communication protocol in an application layer of a multi-layered communication protocol stack.

8. A non-transitory machine readable medium having stored thereon instructions comprising machine executable code which, when executed by at least one machine, causes the at least one machine to:

receive, from a first module executing in a first address space and by a second module in a second address space through a first channel, a first request to execute a first task;

place, by the second module, the first task into a task queue for execution in the second address space;

extract, by the second module from a response queue, pending responses not yet returned to the first module that are results of execution of other tasks in the second address space, wherein requests for the other tasks were previously sent by the first module to the second module for execution in the second address space;

compound, by the second module, the pending responses that are the results of execution of the other tasks in the second address space to form a compounded response;

combine, by the second module, the compounded response and a return value for acknowledgement of the first request to execute the first task into a combined communication; and

transmit, by the second module to the first module in the first address space, the combined communication.

9. The non-transitory machine readable medium of claim 8, wherein the machine executable code, when executed by the at least one machine, causes the at least one machine to:

copy, by the second module, at least one parameter to perform the first task, after receiving the first request to execute the first task.

10. The non-transitory machine readable medium of claim 8 or 9, wherein the transmit of the combined communication is to occur prior to completion of the first task.

11. The non-transitory machine readable medium of any of claims 8 to 10, wherein the transmit of the combined communication is through a second channel.

12. The non-transitory machine readable medium of any of claims 8 to 11, wherein the machine executable code, when executed by the at least one machine, causes the at least one machine to:

complete, by the second module, execution of the first task.

13. The non-transitory machine readable medium of claim 12, wherein the machine executable code, when executed by the at least one machine, causes the at least one machine to:

extract from the response queue other pending responses that are results of execution of additional other tasks in the second address space; and

compound the other pending responses and a response for the first task that is the result of execution of the first task by the second module.

14. The non-transitory machine readable medium of any of claims 8 to 13, wherein the first request and the combined communication are based on a communication protocol in an application layer of a multi-layered communication protocol stack.

15. The non-transitory machine readable medium of any of claims 8 to 14, wherein the combined communication is formatted according to a new data type that is defined to include the acknowledgement and a list of the pending responses, and wherein a return value of the function is defined as the new data type.

16. An apparatus comprising:

a processor; and

a computer readable storage medium having program instructions embodied therewith, the program instructions executable by the processor to cause the apparatus to:

receive, from a first module to execute in a first address space and by a second module in a second address space through a first channel, a first request to execute a first task, wherein the first request is part of a function that is to be executed in the second address space;

place, by the second module, the first task into a task queue for execution in the second address space;

extract, by the second module from a response queue, pending responses not yet returned to the first module that are results of execution of other tasks in the second address space, wherein requests for the other tasks were previously sent by the first module to the second module for execution in the second address space;

compound, by the second module, the pending responses that are the results of execution of the other tasks in the second address space to form a compounded response;

combine, by the second module, the compounded response and a return value for acknowledgement of the first request to execute the first task into a combined communication,

wherein the combined communication is formatted according to a new data type that is defined to include the acknowledgement and a list of the pending responses,

wherein a return value of the function is defined as the new data type; and

transmit, by the second module to the first module in the first address space, the return value of the function after execution of the function.

17. The apparatus of claim 16, wherein the apparatus is configured to copy at least one parameter to perform the first task, after receipt of the first request to execute the first task.

18. The apparatus of claim 16 or 17, wherein the apparatus is configured to transmit the combined communication prior to completion of the first task.

19. The apparatus of any of claims 16 to 18, wherein the apparatus is configured to complete, by the second module, execution of the first task.

20. The apparatus of claim 19, wherein the apparatus is configured to:

extract from the response queue other pending responses that are results of execution of additional other tasks in the second address space; and

compound the other pending responses and a response for the first task that is the result of execution of the first task by the second module.

21. The apparatus of any of claims 16 to 20, wherein the first request and the combined communication are based on a communication protocol in an application layer of a multi-layered communication protocol stack.

22. A computer program comprising instructions that when executed by one or more computer systems cause the one or more computer systems to perform the method of any of claims 1 to 7.

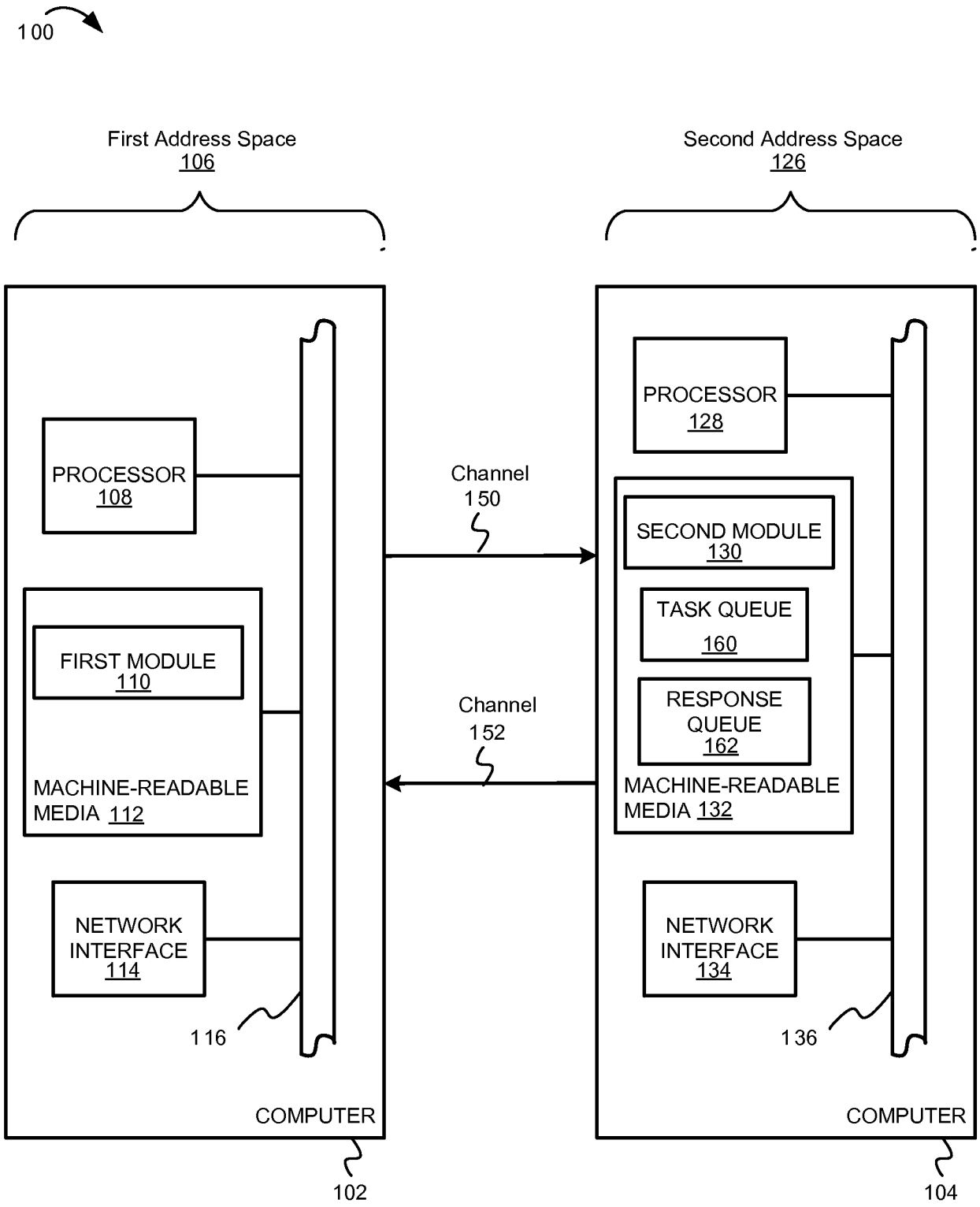
100
First Address Space
106
Second Address Space
126
PROCESSOR
108
FIRST MODULE
110
MACHINE-READABLE
MEDIA 112
NETWORK
INTERFACE
114
116
COMPUTER
102
Channel
150
Channel
152
PROCESSOR
128
SECOND MODULE
130
TASK QUEUE
160
RESPONSE
QUEUE
162
MACHINE-READABLE
MEDIA 132
NETWORK
INTERFACE
134
136
COMPUTER
104

FIG. 1

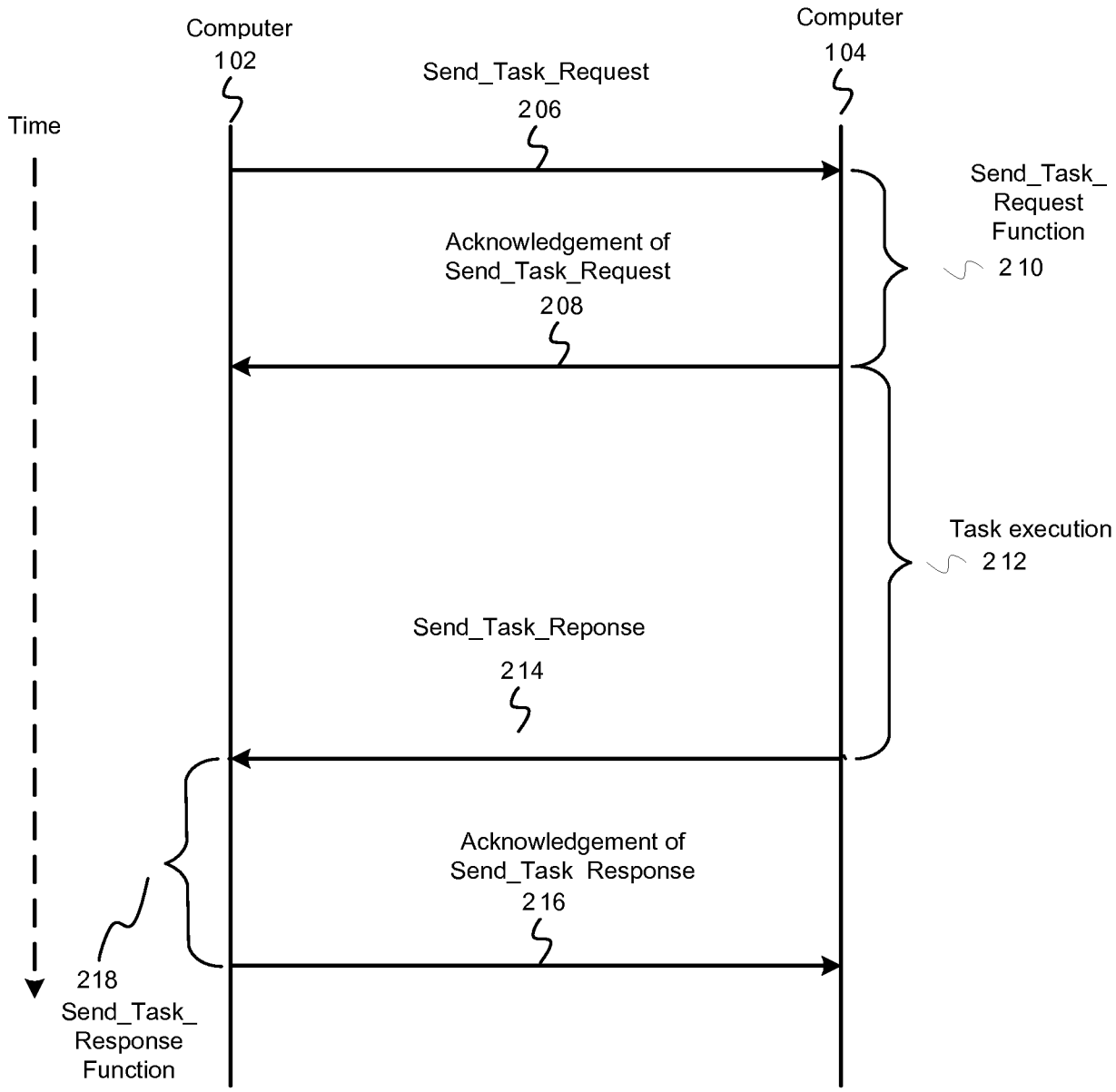
Computer
102
Computer
104
Send_Task_Request
206
Time
Send_Task_
Request
Function
210
Acknowledgement of
Send_Task_Request
208
Task execution
212
Send_Task_Reponse
214
Acknowledgement of
Send_Task Response
216
218
Send_Task_
Response
Function

FIG. 2

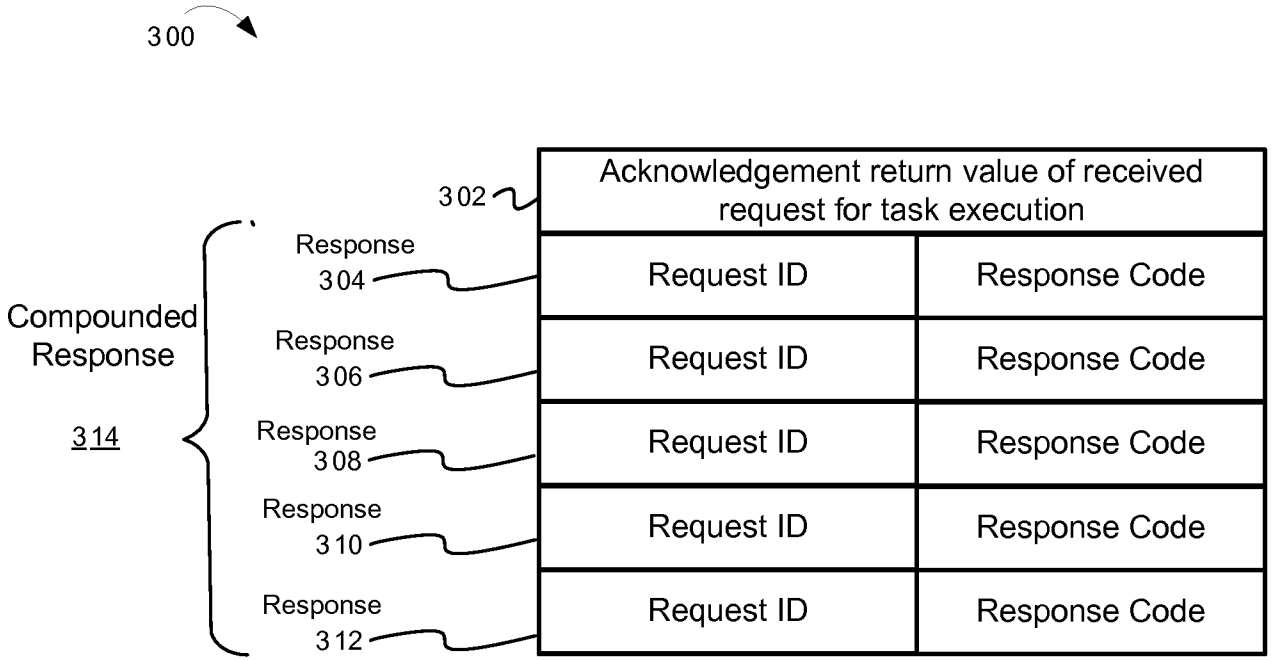
300
302
Acknowledgement return value of received request for task execution
Request ID
Response Code
Compounded Response
314
Response 304
Response 306
Response 308
Response 310
Response 312

FIG. 3

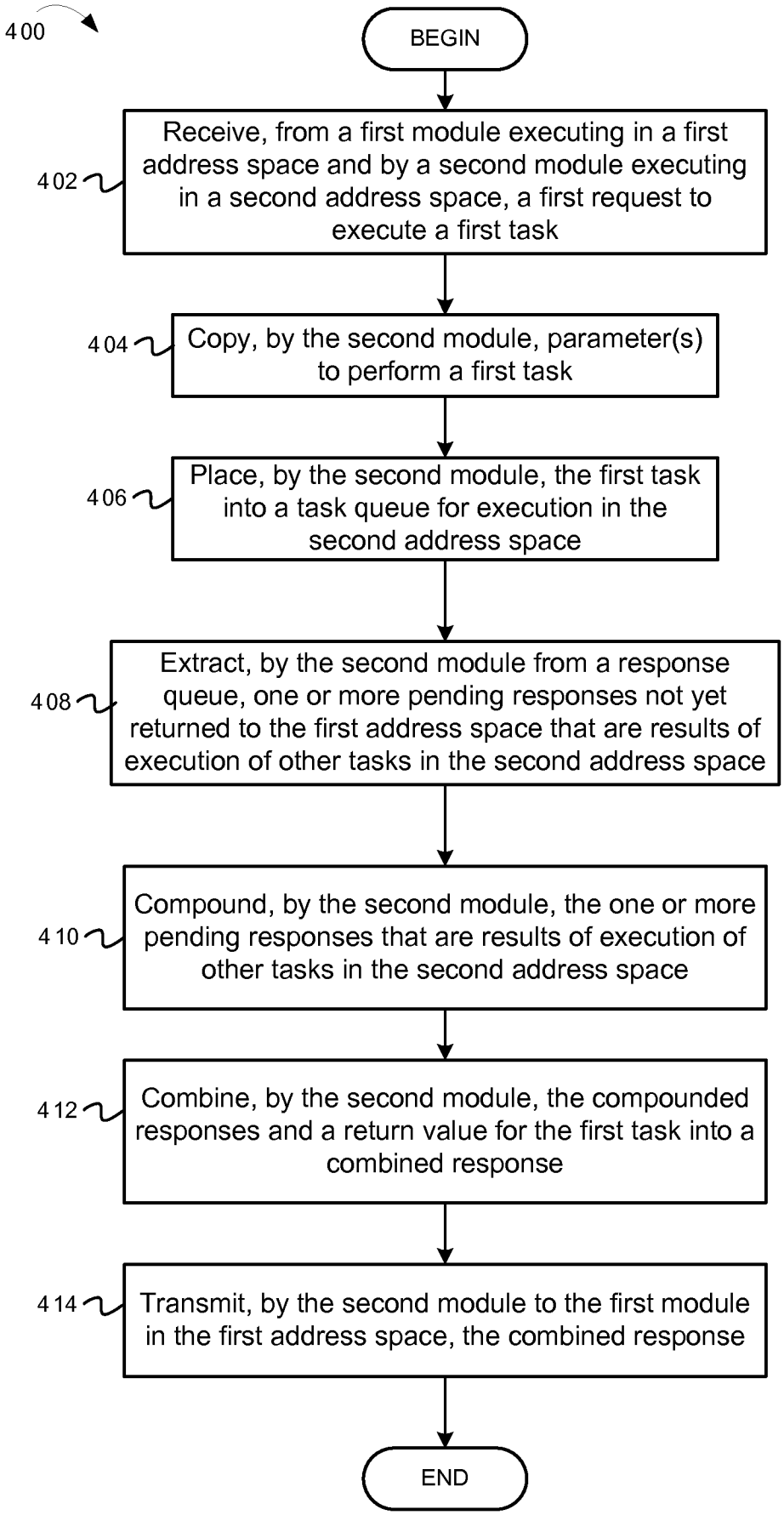
400
BEGIN
Receive, from a first module executing in a first address space and by a second module executing in a second address space, a first request to execute a first task
402
Copy, by the second module, parameter(s) to perform a first task
404
Place, by the second module, the first task into a task queue for execution in the second address space
406
Extract, by the second module from a response queue, one or more pending responses not yet returned to the first address space that are results of execution of other tasks in the second address space
408
Compound, by the second module, the one or more pending responses that are results of execution of other tasks in the second address space
410
Combine, by the second module, the compounded responses and a return value for the first task into a combined response
412
Transmit, by the second module to the first module in the first address space, the combined response
414
END

FIG. 4

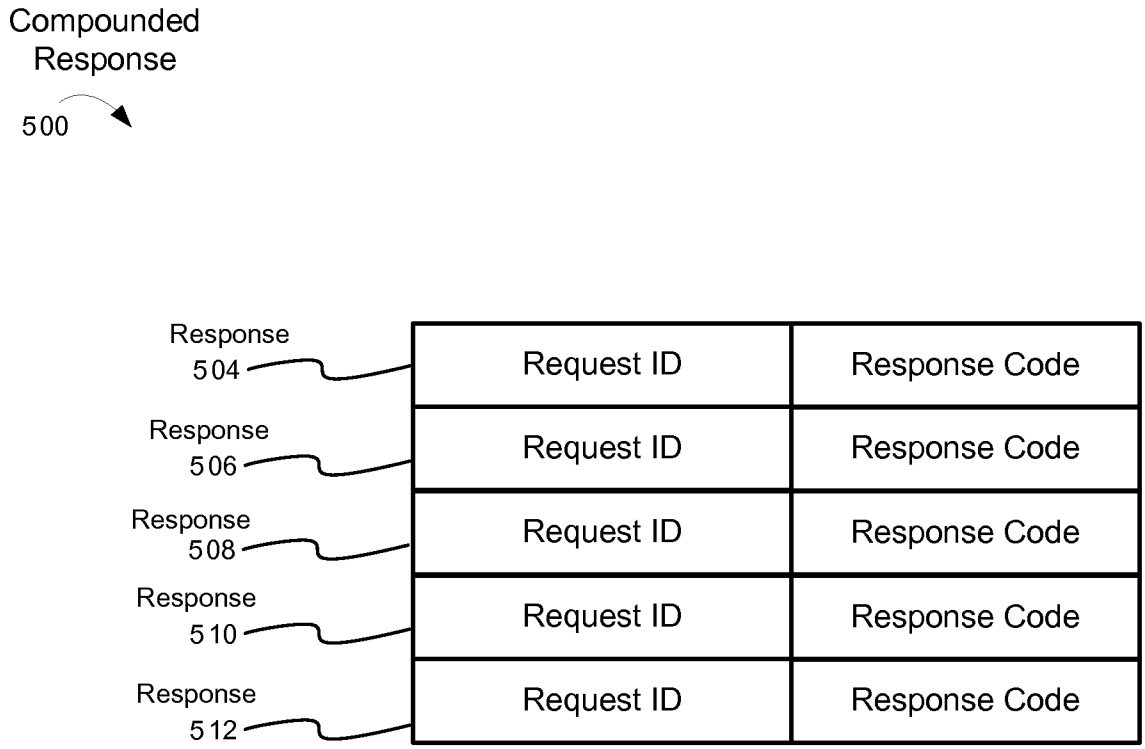
Compounded Response
500
Response 504
Response 506
Response 508
Response 510
Response 512
Request ID
Response Code
Request ID
Response Code
Request ID
Response Code
Request ID
Response Code
Request ID
Response Code

FIG. 5

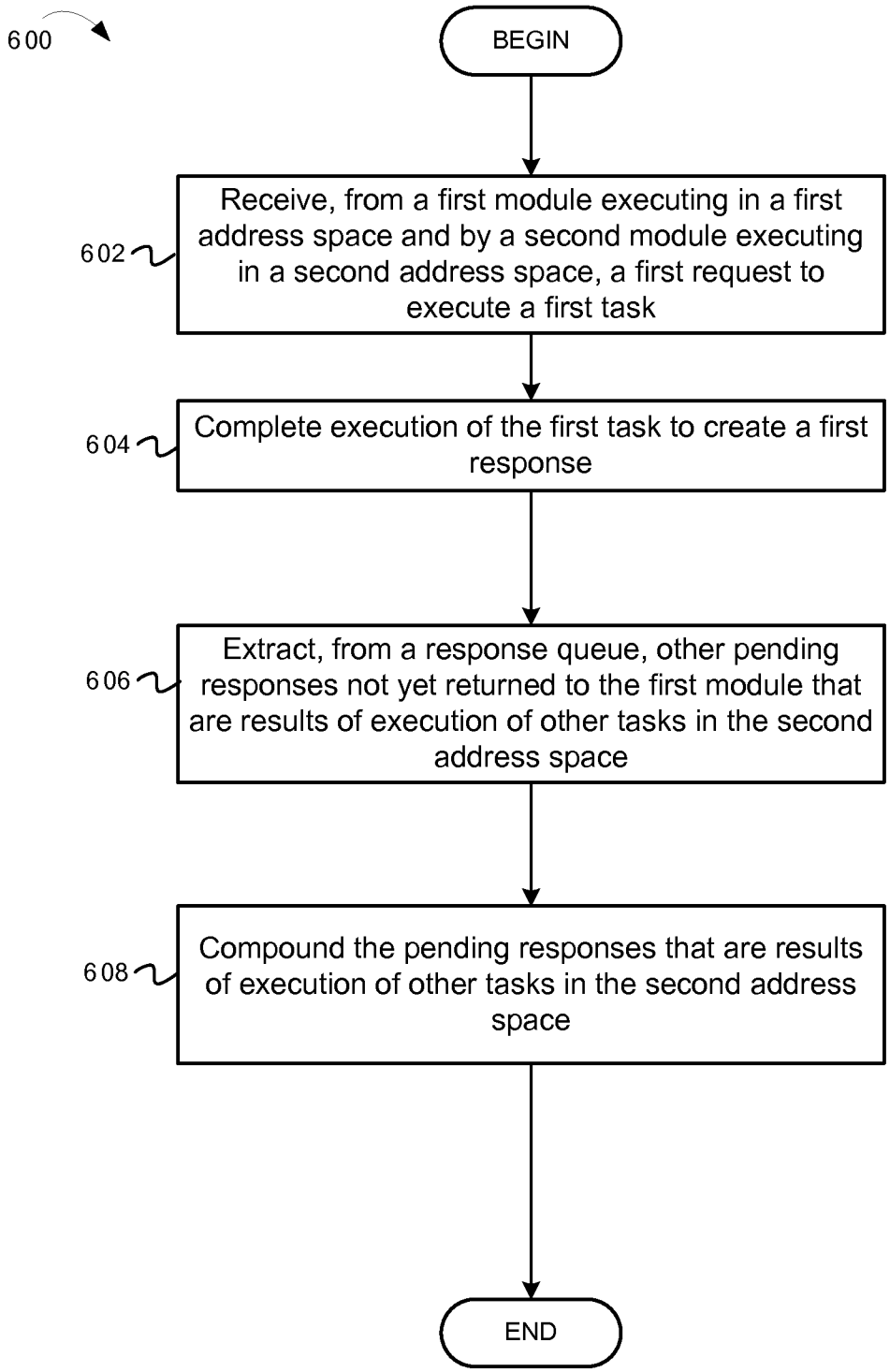
600
BEGIN
602
Receive, from a first module executing in a first address space and by a second module executing in a second address space, a first request to execute a first task
604
Complete execution of the first task to create a first response
606
Extract, from a response queue, other pending responses not yet returned to the first module that are results of execution of other tasks in the second address space
608
Compound the pending responses that are results of execution of other tasks in the second address space
END

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/042181

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06 H04L1/16 G06F9/54
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2007/171919 A1 (GODMAN PETER J [US] ET AL) 26 July 2007 (2007-07-26) page 2, paragraph 0023; figure 1A page 3, paragraph 0033 page 1, paragraph 0006 page 4, paragraph 0050-0051 page 7, paragraph 0077	1-22
A	US 2007/115821 A1 (SIM SANG HYUN [KR] ET AL) 24 May 2007 (2007-05-24) page 3, paragraphs 0033,0047; figure 2	1-22

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

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 October 2015	19/10/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Hoareau, Samuel

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2015/042181

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007171919 A1	26-07-2007	NONE	
US 2007115821 A1	24-05-2007	NONE	