Simulation Modelling Practice and Theory xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Using a novel message-exchanging optimization (MEO) model to reduce energy consumption in distributed systems

Nik Bessis [a], Stelios Sotiriadis [a,*], Florin Pop [b], Valentin Cristea [b]

[a] School of Computing & Maths, University of Derby, Derby, United Kingdom
[b] University "Politehnica" of Bucharest, Bucharest, Romania

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Distributed systems
Message exchange optimization
Energy consumption in distributed systems
Inter-cloud

ABSTRACT

The concept of optimizing energy efficiency in distributed systems has gained particular interest. Most of these efforts are focused on the core management concepts like resource discovery, scheduling and allocation without focusing on the actual communication method among system entities. Specifically, these do not consider the number of exchanged messages and the energy that they consume. In this work, we propose a model to optimize the energy efficiency of message-exchanging in distributed systems by minimizing the total number of messages when entities communicate. So we propose an efficient messaging-exchanging optimization (MEO) model that aims to minimize the sum of requests and responses as a whole rather than only the number of requests. The view is to optimize firstly the energy for communication (e.g. latency times) and secondly the overall system performance (e.g. makespan). To demonstrate the effectiveness of MEO model, the experimental analysis using the SimIC is based on a large-scale inter-cloud setting where the implemented algorithms offer optimization of various criteria including turnaround times and energy consumption rates. Results obtained are very supportive.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, users have increased the scale of the distributed systems due to the increased resource utilization. Clearly, the increased sum of messages that are exchanged between users and system entities have led to an increased energy consumption. Current solutions are focused on the optimization of the performance measures (e.g. resource discovery and scheduling) without focusing on the benefits that may derive from the introduction of a new message exchanging approach. This affects the amount of messages sent and received with regards to the energy consumption of nodes. This area emphasizes an increasing trend that shifts the focus from improving performance to optimizing the energy efficiency and performance of the system as a whole [5,8]. By planning an energy efficient computational model along with performance optimization we can reduce the consumption rates while at the same time increase the user satisfaction. In distributed systems, the performance includes the computational metrics (e.g. execution times) while the energy efficiency includes consumption of the interacting nodes.

Our contribution is by improving the energy consumption rates based on the minimization of the sum of messages that are exchanged during the resource management phase. We define as message exchange in distributed systems the communication among entities (nodes). To achieve this, we introduce a novel message-exchanging optimization (MEO) model in [18] to optimize performance of distributed systems. In addition, we focus on the energy efficiency and we present the performance evaluation of our approach in this direction. Current efforts include various nodes communicating with each

* Corresponding author.
E-mail addresses: n.bessis@derby.ac.uk (N. Bessis), s.sotiriadis@derby.ac.uk (S. Sotiriadis), florin.pop@cs.pub.ro (F. Pop), valentin.cristea@cs.pub.ro (V. Cristea).

1569-190X/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2013.02.003

other in order to request for services by sending messages, however without considering the number of messages. We start with a related works section (Section 2) to present current approaches to and challenges in message exchanging in distributed systems. Based on this analysis, we conclude with a critical discussion of the key characteristics for messaging.

To demonstrate the message approach in a real-case scenario we implement an inter-cloud facility, where various services are submitted from users to clouds for execution. In Section 3, we present our message-exchanging optimization (MEO) model with a specific focus on how to optimize the performance and the energy efficiency of job distribution. We illustrate the mathematical representation based on graph theory in order to determine relationships and structures of nodes (entities of the distributed system). In Section 4, we present the algorithm pseudo-codes to show the processes of message exchanging along with the performance measures. The rest of the paper is organized as follows; the configuration of the setting and the inter-cloud cases (Section 5.1) are included in the simulation toolkit (SimIC) that implements the algorithms (Section 5.2) and produces the experimental results (Section 5.3). The study concludes with a critical evaluation and discussion of future work (Section 6).

2. Related works

Currently, most message solutions are focused on the number of packets that are moved among processors of clusters and grid settings. In such cases the Message Passing Interface (MPI) [11] has been introduced as a portable message passing standard. The MPI is related with the point-to-point communication and the collective communication approach. The first case (point-to-point) includes that the processor of a node sends a message to another along with some data. MPI processes are independent and they communicate to coordinate a job submission, so messages are sent between two processes. The actual operation includes that one process sends a message to another one that receives it, and then it replies to the sender. A typical rule is that for a point-to-point MPI case the number of messages that are sent and received should match (one receive per send).

Each message contains a number of properties that include the actual data, the data type of each element, the number of elements, a message tag, and the ranks of the source and destination process [1]. This kind of exchange could occur in synchronous or asynchronous mode. For instance, the synchronous mode includes the sending of complete information while the asynchronous contains data regarding the time that the message left from the first process. In addition, asynchronous allows high dynamicness of the system, so it could be applied successfully for cases of variable workloads [12]. Point-to-point has been considered as a flexible method for messaging, however for a large number of processes the sum of messages will be high; a fact that affects the overall system performance. This shortcoming is based on the request to response model and includes that for each request a process should always reply.

A different approach is the collective communication that involves the transfer of many processes at a time; so, we could have coordinated communication of a group of processes. The solution increases the design complexity as it encompasses the synchronization of processes. Nevertheless, it is a more advanced approach which in operation could be applicable for larger scale distributed infrastructures. The collective communication is always synchronous in the sense that collection will not be completed until all MPI nodes reach the same point [19]. In general, by using broadcasting named as "broadcast call" one node sends a message to all nodes of the group. The "reduce call" procedure is called by the MPI at the end of the process for collecting information from all nodes' processors and stores the result on one node [2]. The collective functions follow the basic MPI requirement that denotes that the amount of data sent should match the amount of data as specified by the receiver [15], thus this makes it an inflexible solution.

For the collective communication procedures, a variety of different routines have been introduced to implement different communication patterns [11]. This includes AllToAll, AllGather, BCast, Scatter and Gather, AllReduce and other functions. Initially, the AllToAll model allows complete information distribution among all the node processes of a group [19]. This model forms the basic communication pattern that is used from most distributed systems, see, [22–25]. This is because the focus of such systems is on the scheduling aspects rather than on communication. Based on that, we also focus on utilizing the AllToAll solution for our experimental analysis. AllGather, on the other hand, collects processes and then broadcasts (BCast) to each conducted node. However, as the number of nodes increases, the performance is compromised due to the congestion of network resources. Finally, the Scatter and Gather method allows collective processes to be distributed in a different process and gather again back to the origin processor. A different view of messaging could include the network algorithms for minimizing the network bottleneck.

A number of theoretical models have been developed in order to avoid network congestion [1], however from the perspective of packets that are exchanged among network entities. For example, the spreading simple algorithm allows a node to send data to node as node $(p + i)$ where p is the process and i the iteration and, receive data from node $(p - i + N)$ mod N (where mod is the division modulo) [1]. A different approach called the ring/bucket/circular algorithm is presented in [3]. Specifically, at each iteration i a process p sends data to a node with an index $(p - i + 1 + N)$ mod N to the right neighbour on the list. The recursive doubling algorithm [19] requires less time as the number of total transfers is reduced. The MPI makes use of the MPICH [11] (MPI Chameleon) to recursively reduce the number of messages by utilizing a criterion: when the number of processes is a power of 2 it uses recursive doubling for small message sizes. Next, for the rest of the messages (large size) it uses the ring algorithm to achieve message dissemination. However, this solution is aimed at small-scale [9] based parallel computing systems. Steffenel and Jeannot [2] propose that most of these algorithms have been designed for homogeneous and tightly coupled systems.

In the case of high heterogeneous and de-coupled settings (e.g. grids and inter-clouds) solutions divide network into hierarchies. The MPICH-G2 [14] presents algorithms to gather data up the hierarchy using recursive doubling and then broadcast these data by binomial broadcast (according to a probability factor). Similarly, the MagPIe presented in [4] includes a three-stage algorithm to first AllGather data at coordinators; second gather data among coordinators and, third broadcast data by coordinators using, again, a binomial broadcast. The major drawback is that it follows static network hierarchical schemes in modelling decisions [10]. In addition, data transmission is repeated at coordinators thus keeping bandwidth values in high layers of hierarchy in low levels [1].

Gupta and Vadhiyar [1] illustrates an algorithm that is network topology aware and adaptive to various network loads. This solution follows the transient clustering of nodes based on network characteristics. In contrast, Steffenel and Jeannot [2] focuses on an alternative algorithm for minimizing the number of steps through a wide-area network. They also claim that the reduction has a direct impact on the performance modelling by minimizing the factors that directly interfere with the wide-area communication. Although efficient algorithms have been developed for specific networks, a generic model for heterogeneous and decoupled nodes has been proven to be complex to design. This is because of the dynamic nature of the resources. An important requirement to be considered is the time-centric information processing and the levels of elasticity and scalability that are required from the jobs. Fig. 1 demonstrates the relationships between various algorithms and approaches discussed.

Today, large-scale systems, e.g. inter-clouds, utilize the notion of the resource manager in order to achieve the dynamic capabilities. Specifically [16] presents meta-scheduling solutions by allowing communication to be implemented in terms of message exchanges. However, most of these approaches aim at decentralized, heterogeneous, decoupled, and transparent systems, where entities exchange messages in an AllToAll fashion. Others, like [16], utilize a common negotiator as an interface where jobs are placed in a kind of advertisement and, resources bid by sending back messages. In our work, we introduce a novel message-exchanging optimization (MEO) model for inter-cooperated infrastructures. We suggest that by minimizing the sum of messages sent and received we can enhance both the performance measures and energy efficiency.

Our solution highlights the shortcomings of the point-to-point MPI, e.g. for a large scale of processes the number of messages will be extremely high. This affects the overall system performance because for each request a process should always reply. Also the MPI collective approach leads to a similar limitation: the number of sent and received data should be the same at the collection point. Other solutions, like BCast, AllGather, etc., represent approaches to send messages and not how to receive. Finally, the messaging as applied to meta-scheduling systems includes heuristic criteria [21]. Such systems encompass AllToAll methods that send messages or use heuristic algorithms to reduce the number utilizing opportunistic criteria. In such cases the responders collect requests and reply back with positive or negative replies. Other works, as in [13], focus on interoperable agents, which travel in settings and capture resource information regarding their action domain so they update the internal data of nodes. However, this is considered as a heavy solution due to the increasing agent size.

In contrast, we propose the MEO model to minimize the sum of requests and responses as a whole rather than only the number of requests. Our work is related to message exchanging in large-scale infrastructures (grids and inter-clouds) so we consider the method of collective pattern solution as the most appropriate. This is because it involves complete exchange for one-to-many nodes as happens in a distributed system. Specifically, we propose that number of responses could have a vital role in the overall optimization of energy efficiency. The next section presents our proposed MEO model.

3. MEO: the energy-aware message-exchanging optimization model

In distributed systems, each resource has a component named as a distributed resource manager that optimally and efficiently assigns jobs to local resources for allocating computational units. The jobs are exchanged in terms of messages that

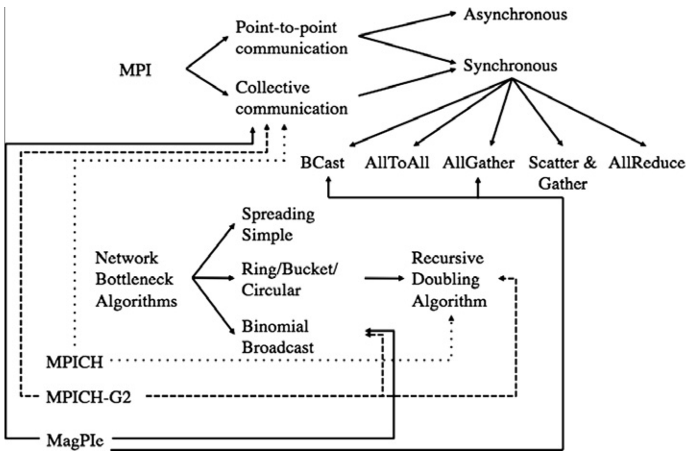

Fig. 1. The relationships of message passing and network bottleneck algorithms.

are sent and received among inter-connected nodes in order to find capable resources for execution. In contrast with most of the current research focused on the optimization of resource management aspects (e.g. performance of resource discovery and scheduling) we focus on the communication process.

We focus on the effect that could be achieved when reducing the sum of messages in terms of performance and energy consumption. A vital concern to address is the actual topology of the system and the level of the depth that topologies could have. This issue emerges new challenges that are associated with the number of resources that constitute an interoperable system [7]. We characterize the whole message model to base upon a time-centric solution; where a decentralized topology of distributed resource management components allows extensive message requests (messages sent) however during regular time intervals. This allows us to address the NP-complete distribution problem. MEO comprises of two phases namely, the Centralized Message Phase (CMP) and the Decentralized Message Phase (DMP). Fig. 2 demonstrates the centralized and decentralized phases as executed during the message distribution.

The DMP allows us to model a realistic large-scale setting that increases the opportunity for resource discovery and allocation as more and more resource managers communicate within specific time frames. In this way, MEO is expected to assist the overall resource management phase (resource discovery, scheduling and allocation). This involves messages that are exchanged at different time instances, as well as considering deadlines for job executions with regard to the problem specification. MEO defines the complete group of processes that illustrate both phases (CMP and DMP) and we demonstrate each as follows:

- *Step 1:* The CMP starts when a number of jobs are submitted to an entity that becomes the requester. Each job contains a set of requirements that are organized as properties regarding time intervals (e.g. waiting time, interval, etc.) and computational units (CPU, memory, bandwidth, etc.).
- *Step 2:* The requester stores each job received in a list. Each list row has a message with key characteristics including the deadline and the length of job as a measurement for calculating resource availability on remote resources.
- *Step3:* The requester defines the deadline that its related interval limit and the size of the list (e.g. for big lists the deadline could be higher as the time needed to be transferred is higher). This also defines the cost of communication among entities. So a small deadline could result in a small number of job submissions, while a large one could lead to a heavy submission.
- *Step 4:* The requester collects addresses of inter-connected nodes from a catalogue. These nodes will become the responders in the communication. The study assumes that these are stored in a local profile.
- *Step 5:* The requester sends the list with jobs as a message along with data requirements (e.g. deadline, job length). In addition, the message includes the ranking criteria (e.g. turnaround, energy consumption level); so all the responders will use the same classification levels for fair selection. It should be mentioned that identifications and tags define each message. During communication, the tags are set to unique values in order to characterize the group of messages.
- *Step 6:* The responder collects a single request (i.e. the list along with data) and performs an internal resource availability operation according to the ranking criteria. Then it classifies jobs and generates a new temporary list.
- *Step 7:* Each job is ranked according to a function and a decision is taken with regards to job availability in requester resources.
  Step 7a: In case of full job availability (CMP includes that each job on the list can be executed locally) the responder generates a list with jobs.
  Step 7b: In case of non-availability (e.g. responder can execute none or few of the jobs contained in the list) the requester initializes DMP. This implies a further job distribution, however in an updated time frame. This will loop steps 1–6.
  Step 7c: In case of non-availability because of an empty responder profile (e.g. resource is the actual terminal) the message is finished instantly. Therefore, responses are not sent back.
- *Step 8:* The new list is created with jobs and rankings in a descending performance order of rankings. This forms the criteria for selecting jobs at the next resource management level. In the case that the DMP responder acknowledges that the job(s) will be executed on a remote machine, it redirects messages to inter-connected nodes. All messages are assigned with updated time deadlines.
- *Step 9:* The lists are collected from the requester that compares and classifies jobs by using the desired ranking criteria. This includes unique jobs that are now assigned with an identification of the selected resource for allocation. The decision defines whether a remote resource will be accepted as the host for job execution or not.

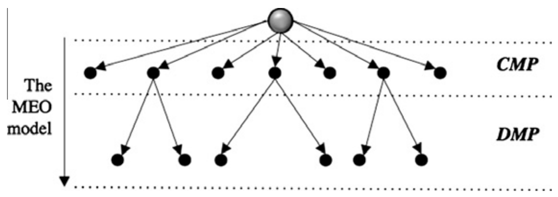

Fig. 2. The CMP and DMP phases.

- *Step 10:* The procedure ends and each job is sent to the local or remote resource.

The next sections detail the CMP and DMP communication method for message distributions by focusing on messaging tagging.

3.1. Centralized Message Phase (CMP)

This is the first level of message exchanging. Let us assume that a central entity collects job requests in the form of messages from all other sources (e.g. a user). Each job request contains a list of requirements. To identify resource availability for job allocation and execution, the central entity must send job messages to any inter-connected entity. Inter-connected entities collect the job messages, check for availability and in cases where more than one job requests are contained in the same message, inter-connected entities also rank (in terms of preference) each job contained in the message list. The ranking list is sent back to the central entity if there is resource availability. In case where no availability exists in a remote entity the process is terminated instantly and responses are not sent back. In this way we reduce the number of messages sent back with the central entity receiving ranked positive responses only.

The central entity sends jobs by (a) waiting for an interval call (a time frame to elapse) or (b) reaching the number of jobs in the list. Collected jobs are ranked according to a performance criterion and jobs are ready for allocation to different resources. Fig. 3 shows the CMP and the interactions of a central entity with three responders. We characterize CMP as acting similar to a centralized computing system where the nodes have complete knowledge of all resources. This includes that jobs are about to be executed only from the inter-connected resources. In other words, the CMP decision-making becomes the centralized component for identifying resources and planning resource management.

Fig. 3 shows that the Entity$_{Req}$ (central entity) forms a list that will be forwarded to entities Entity$_{Res}$ (remote inter-connected entities). Then it assigns a tag and a value (e.g. tag = u) that is the same for each submission. Inter-connected entities (e.g. Entity$_{Res}$) receive a message with the specific tag and perform an availability check for each of the jobs in the list. This includes the ranking of jobs so a new list is formed containing the jobs in a descending of performance measures (e.g. highest performance rate is placed first). In case of non-availability (e.g. list is empty) the request is terminated, thus a response is not sent back. In any other case the Entity$_{Res}$ reforms the list and sends back its response. The Entity$_{Req}$ collects jobs by identifying the tag value (tag = u) and performs the global ranking (based on different performance functions, e.g. energy cost). Finally, jobs are assigned to inter-connected Entity$_{Res}$ using a different tag denoting the next resource management step (job assignment and allocation).

3.2. The Decentralized Message Phase (DMP)

The second phase is triggered when messages during CMP could not lead to the central Entity$_{Req}$ assigning all jobs to any inter-connected Entity$_{Res}$. In this case, messages are forwarded to a second level of remote resources. These are resources that are inter-connected with the initial requesters. Fig. 4 shows the interactions between the two levels of entities. Specifically, the first level is the entities that are directly related with the requester (centralization), and the second level defines the possible remote sites (decentralizations).

DMP details the distribution of jobs based on contacted entities decision-making processes. This implies that in case of non-availability of first contacted entities, a request could be further distributed to other remote resources. By assuming that there could be different levels of responding nodes, the possibility for further dissemination is increased. This is also based on the fact that entities could be part of different virtual organizations or collaborating groups. In general, decentralization offers a variety of advantages such as interoperability and heterogeneity management along with increased resource availability (elasticity factor). However, this involves complex topologies of entities. Nevertheless, we use timing and timestamps in order to control the job spreading.

DMP extends CMP. In DMP Entitiy$_{b\_Res}$ is transformed to an Entity$_{b\_Req}$ and forwards a request to its inter-connected resources (e.g. Entity$_{c\_Res}$) that is initially unknown and unreachable to the Entity$_{a\_Req}$. In this way we ensure that jobs are forwarded in terms of messages that are exchanged in a total decentralized approach. It should be noted that MEO model is

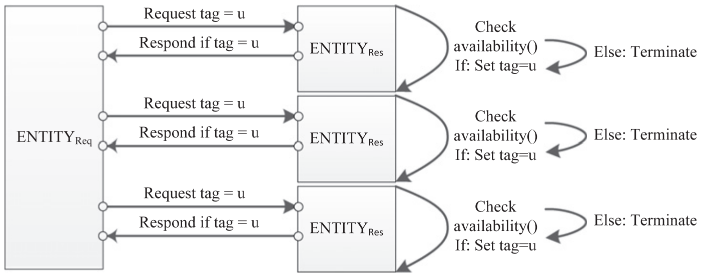

Fig. 3. The Centralized Message Phase.

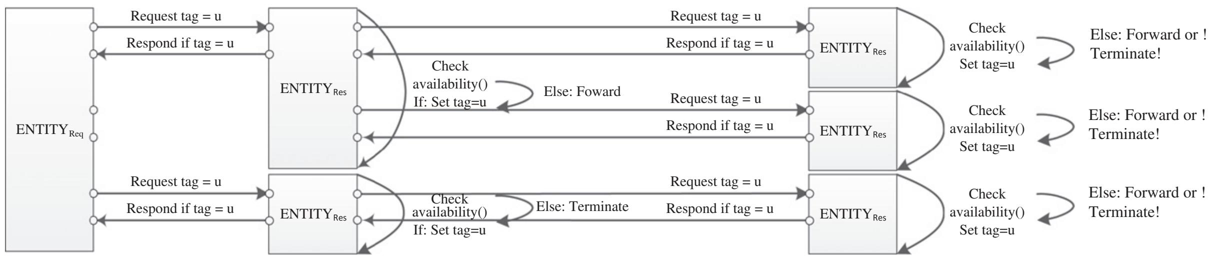
ENTITY_Req
Request tag = u
Respond if tag = u
ENTITY_Res
Check availability() If: Set tag=u
Else: Foward
Else: Terminate
Check availability() Set tag=u
Else: Forward or ! Terminate!

Fig. 4. The Decentralized Message Phase.

based on time-centric criteria, thus each message has a set of intervals. When the initially defined time frame has elapsed an instant termination of message is denoted. Based on Fig. 4, a list of jobs is distributed to the first level of entities as happened with CMP. If availability exists, then the updated list is sent back to requester using the defined tag (tag = u). In case of non-availability of jobs execution the contacted entity ($\text{Entitiy}_{\text{b\_Res}}$) forwards the request to remote resources (e.g. $\text{Entitiy}_{\text{b\_Res}}$) and sets a new tag.

At this time instance the procedure is forwarded to the remote entity ($\text{Entitiy}_{\text{b\_Res}}$) that executes initially the CMP to identify local resources, and the DMP in case of non-availability. In a similar vein, the same procedure is followed until the end of the initially defined interval. For each further distribution we set a new interval that is lower than the initial requested. This is because we take into account the communication delays and the time needed for decision making at the first level of entity. During exchange of the rankings, criteria are configured to the same performance measures of the requester in order to have a fair resource selection strategy. Next we detail the mathematical representation of the aforementioned discussion.

4. Mathematical representation of MEO

We detail the mathematical representation of the energy efficient MEO model that includes the CMP and DMP using graph theory. Let us assume that there is $v_1 \in V$, where $V = \{v_1, v_2, \ldots, v_n\}$ are entities of a distributed system that constitute the message-exchanging nodes. Each node has an uptime value υ_{v_n} and $\{\upsilon_{v_n} \in \mathrm{R} | \upsilon_{v_n} > 0\}$ that defines its operational period (in ms). The graph $G = (V,E)$ is an undirected graph with nodes $v_n \in V$ and $v_n \neq v_{n+1}$, etc. If v_n communicates with v_{n+1} a trail is created between nodes called w_1. In our case we aim at a directed graph as nodes communicate with different orders, thus $w_{n:n+1}$ is considered as a walk that connects v_n and n_{v+1}. In a similar way, $w_{n+1:n}$ is considered as a walk that connects n_{v+1} and v_n. Each v_n contains a profile f_n where $v_{n+1} \in f_n$, so n_v and n_{v+1} are inter-connected nodes.

- We define a list $f_n = \begin{bmatrix} v_{n+1} \\ \vdots \\ v_w \end{bmatrix}$ to contain the entities $v_{n+1}, \ldots, v_w$ as the addresses of nodes to send messages.

Suppose that a request $r_a \in Rq$, and $Rq = \{r_1, r_2, \ldots, r_p\}$ is submitted at a time instance t_i, and $\{t \in R | t \geqslant 0\}$in an entity v_a. The entity defines an interval int_a that expresses the waiting time (deadline) and $\{\text{int}_a \in R | \text{int}_a > 0\}$. The entity also defines a maximum size of the file as s_a and $\{s \in R | s > 0\}$. Each list that is generated by v_a contains a number of data $d_a \in D$ and $D = \{d_1, d_2, \ldots, d_q\}$ that are organized in an array with respect to the message identification. The d_a contains the job specification and the interval that it could vary for each job. Thus, each message $m_a \in M$, and $M = \{m_1, m_2, \ldots, m_i\}$ where $m_a = [L_a, tag_a]$ and L_a is the list of the message with tag_a as the tag value of the message.

- We define a list $L_a = \begin{bmatrix} d_1 \\ \vdots \\ d_s \end{bmatrix}$ to contain the data of a message where s defines the size of the list and $d_1 = \begin{bmatrix} j_1 int_1 \\ \vdots \\ j_s int_a \end{bmatrix}$, where j_s is the last job with interval int_a.
- We define as $j_f = [clocks_f\, cpi_f\, cores_f\, bw_f, h_f]$ as the basic metrics for calculating performance (h represent the requested hours).

Let us suppose that the message sent to a recipient v_{n+1} with a delay dl_a and $\{dl \in R |\ 0 < dl_a < int_a\}$. The node v_{n+1} executes a ranking function according to a performance measure. For example, this is related with information extracted from j_1 and dl_a, and for the case of the total time taken between the submission of a job for execution and the return of the complete output it is defined by the turnaround ranking formula. A more detailed discussion on ranking measures will be presented in the next sections that include the variety of measures and metrics.

$$TurnaroundRanking(job_i) = \left\{ \frac{instr_{job_i} cpi_{job_i}}{clocks_{job_i} cores_{job_i} 10^5} \right\}_{cloud} + int_{job_i}$$

The response message is $m_b \in M$ where $m_b = [L_b, tag_b]$.

- We define as list $L_b = \begin{bmatrix} j_k \\ \vdots \\ j_l \end{bmatrix}$ where j_k is the first ranked job and j_l is last ranked job of the L_b.

- We define as $j_k = [perf_k, udl_a]$ as the basic metrics for calculating performance where udl_a is the updated delay $\{udl_a \in R | 0 < udl_a < dl_a\}$ that includes the decision making time frame of the requester.

In this case the trail $w_{n:n+1}$ defines the distance of the requester and responder thus $w_{n:n+1} = dl_a$. The requester collects messages and ranks jobs according to a function called *RankReq(L_b)* function. Then each job j_k is associated with a resource, e.g. v_{n+1}. Specifically, the *RankRes(Lb)* defines the minimization of the performance criteria. Usually, this is related to the origin of the use case, e.g. turnaround times or energy efficiency. The latter implies that the initial performance parameter is related to the energy consumption of a resource manager. Table 1 details the list of notations.

Based on this discussion we present lemmas to address the following:

- The trail calculation in a bidirectional graph.
- The message number calculation of MEO.
- The energy-aware message-exchanging model optimization.
- The message timing justification.
- The message distribution.
- The energy-aware message exchanging costs (message size and delays).
- The energy-optimization of MEO.

The lemmas show the performance of the energy-efficient MEO model is always equal or better when compared to the AllToAll approach for the specification discussed in the lemmas. In addition, we have defined the costs of communication in terms of message size and delay.

4.1. Lemma of trail calculation

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and v_{n+1} is the ending point with $\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$. We assume that there is at least one path from v_n to v_{n+1} if and only if $v_{n+1} \in f_n$. For

Table 1
List of message-exchanging notations.

Notation symbol	Description
V	A set of nodes
$v_n \in V$	A node
$w_{n:n+1}$	A trail between nodes
υ_{v_n}	A node uptime value
f_n	A profile of v_n
Rq	A set of requests
$r_m \in Rq$	A request
t_a	A time instance
int_a	An interval of node$_a$
s_a	The size of the message
D	A set of data
d_a	A dataset with job specification
M	A set of messages
m_a	A message
L_a	A list of data
w	The size of the L_a
k	The index value of the best job (j_k)
tag_a	A tag value of the message
j_f	The job representation
$clocks_f$	The job clocks (mhz)
cpi_f	The job cpi
$cores_f$	The required cores
$memory_f$	The memory
$storage_f$	The storage
bwf	The bandwidth
dl_a	The delay to reach a recipient
$RankRes(L_a)$	The ranking function at receiver
$perf_k$	The performance value
udl_a	The updated delay
$RankReq(L_b)$	The ranking function at requester
l	The size of L_b

each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_{n+1} a trail is created with weight $w_{v_n:v_{n+1}}$. Then for all $w_{v_n:v_{n+1}}$ in a directed graph formation there is a bi-directional weight that is calculated as follows:

$$w_{v_n:v_{n+1}} = \begin{cases} dl_a + udl_a : \ if\, RankRes(L_a),\ \nexists L_b = \varnothing \\ (2 \times dl_a) + udl_a \leqslant int_a : \ if\, RankRes(L_a) \to L_b \neq \varnothing \quad \text{and } w_{v_n:v_{n+1}} = w_{v_{n+1}:v_n} \\ \sum_{z=v_n}^{z=v_{n+1}} dl_z + udl_a \leqslant int_a : \ if\, RankRes(L_a) \to L_b \neq \varnothing \quad \text{and } w_{v_n:v_{n+1}} \neq w_{v_{n+1}:v_n} \\ \sum_{z=v_n}^{z=v_j} dl_z + \sum_{z=v_n}^{z=v_j} udl_a \leqslant int_a : \ if\, RankRes(L_a) \to L_b \neq \varnothing \quad \text{and } w_{v_n:v_j} \neq w_{v_j:v_n} \end{cases}$$

Proof. Let dl_a to be the distance between v_n and v_{n+1} and $\upsilon_{v_n}, \upsilon_{v_{n+1}} \neq 0$. We assume that there is at least one path if $v_{n+1} \in f_n$ then the sum of the delay dl_a and the udl_a is the total weight of the path as the return of the $RankRes(L_a)$ as defined by the L_b is an empty set (in other words there is no job execution availability). In contrast if the $RankRes(L_a)$ returns an L_b set, then the weight equals to the dl_a multiplied by 2 (if $dl_{v_n:v_{n+1}} = dl_{v_{n+1}:v_n}$) as it includes a request and a response message transfer. Finally, we add the value of the udl_a the decision making delay of the v_{n+1}.

In case where $dl_{v_n:v_{n+1}} \neq dl_{v_{n+1}:v_n}$, the sum of the delays and the udl_a define the weight of the path. In particular the int_a is always greater or equal to the weight as it represents the deadline for the message distribution where $\upsilon_{v_n} > int_a$ and $\upsilon_{v_{n+1}} > int_a$. In other words, if the int_a becomes greater than the weight of the path the message is terminated. At last, the addition of the sum of the delays (dl_z) and internal decision making (udl_z) for a multi-level distribution case will give us the value of the total delay that is always lower or equal to the int_a for a non-message termination case.

4.2. Lemma of message number calculation of MEO

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and v_{n+1} is the ending point with $\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$. Suppose that $w_{v_n:v_{n+1}} \neq 0$ then the number of sent messages (SM) during exchanging is calculated as $MS_{MEO} = \eta + \frac{\eta}{\theta}$ and, $\theta \neq 0$ where η is the maximum number of messages forwarded from requester and θ is the factor of availability defined by the v_{n+1} as derived from $RankRes(L_a)$. Specifically the last operator generates the L_b that is the list of job availability in v_{n+1}. The θ value is calculated as $\theta = \frac{\eta}{e}$ where e is the size of the received messages if the list L_b is not an empty set thus $l \neq 0$.

Proof. Let $w_{v_n:v_{n+1}} \neq 0$, $\theta \neq 0$, $l \neq 0$ and $\upsilon_{v_n}, \upsilon_{v_{n+1}} \neq 0$ then the size of the list L_b defines the capacity of the v_{n+1} to reply back that is formed according to the $RankRes(L_a)$. In this case the maximum number of jobs is set to l. Then the θ factor equals the number of messages sent divided by the number of messages received (based on the l value). Then the overall value of messages is the sum of sent messages from the requester to the division of sent from responder expressed by the v_{n+1} capability and described by the θ factor. We define $l \neq 0$ as in any other case (e.g. $l = 0$) the message is terminated, as there is no job availability.

4.3. Lemma of energy-aware message-exchanging model optimization

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and v_{n+1} is the ending point and $\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$. Suppose that $w_{v_n:v_{n+1}} \neq 0$ then the number of sent message (SM) by using MEO is always lower or equal to the AllToAll collective approach. Thus, the performance of the MEO is always equal or better when compared to the AllToAll approach. In case of $\theta < \eta$ and $l \neq 0$ and the value of $\theta > 0$, MEO performance is continually improved than the AllToAll approach.

Proof. As $w_{v_n:v_{n+1}} \neq 0$ a trail among v_n and v_{n+1} ($\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$) thus this designates that at least one message has been sent. The AllToAll MS is calculated $MS_{AllToAll} = 2 \times \eta$ while the SM of MEO is $SM_{MEO} = \eta + \frac{\eta}{\theta}$. As the hypothesis includes that $\theta \neq 0$ then if $l = \eta$, for all contacted $v_{n+1}, v_{n+1}, \ldots, v_{n+w}$ (w is the last node of the f_n list that can offer availability) $SM_{AllToAll} = SM_{MEO}$. However, if $l < \eta$ and $l \neq 0$ the value of $\theta > 0$ thus according to lemma 4.2 the $SM_{AllToAll} > SM_{MEO}$. So, we define $Perf_{MS_{all\text{-}to\text{-}all}} = \frac{1}{MS_{all\text{-}to\text{-}all}}$ and $Perf_{MS_{MEO}} = \frac{1}{MS_{MEO}}$ then we conclude that $Perf_{MS_{MEO}} > Perf_{MS_{all\text{-}to\text{-}all}}$.

4.4. Lemma of message timing justification

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and v_{n+1} is the ending point and $\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$. For each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_{n+1} and vice versa, after the ranking operation where $l \neq 0$, the total delay time is formulated as $int_{v_n} \geq d_{v_n} + d_{v_{n+1}} + udl_{v_{n+1}}$. We propose that in a MEO model the interval time is always greater or equal to the sum of the delays in order to have message replies.

Proof. According to lemma 4.1, $w_{v_n:v_{n+1}}$, represents the weight of the path with respect to the delay of the communication link. Thus, in this case $d_{v_n} \leq int_{v_n}$, $d_{v_{n+1}} \leq int_{v_n}$ and $udl_{v_{n+1}} \leq int_{v_n}$. If the sum $(d_{v_n} + d_{v_{n+1}} + udl_{v_{n+1}})$ is greater than int_{v_n} then a message is terminated immediately. So, as $l \neq 0$ the sum of delays is required to have a lower value than the total interval time based on our hypothesis that includes that a response is sent if and only if there is availability and the time interval has not been surpassed.

4.5. Lemma of message distribution

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and connected with v_{n+1} that is further connected with v_{n+2} as a finishing point and $\upsilon_{v_n} \neq 0$, $\upsilon_{v_{n+1}} \neq 0$ and $\upsilon_{v_{n+2}} \neq 0$. For each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_{n+1} and forwarded to v_{n+2} the probability of succeeding availability is decreasing through time. That is to say that $P(v_{n+1}) > P(v_{n+2}) > \cdots > P(v_i)$ for a given interval when $uld_{v_n} < uld_{v_{n+1}} < uld_{v_{n+2}}$, etc., so the probability to find a resource on time tends to decrease with respect to the initial chosen interval of the requesting entity.

Proof. Let $w_{v_n:v_{n+1}} \neq 0$ and $w_{v_{n+1}:v_{n+2}} \neq 0$, with $\upsilon_{v_n} \neq 0$, $\upsilon_{v_{n+1}} \neq 0$ and $\upsilon_{v_{n+2}} \neq 0$, so there is a trail with $w_{v_n:v_{n+2}} = w_{v_n:v_{n+1}} + w_{v_n:v_{n+2}}$ and $w_{v_n:v_{n+2}} \neq 0$. d_{v_n} defines the delay of the channel to reach v_{n+1} and the $d_{v_{n+1}}$ the delay of the channel to reach v_{n+2}. For a non-message termination case, and if we assume that $d_{v_n:v_{n+1}} = d_{v_{n+1}:v_n}$ we have $int_{v_n} \geqslant (d_{v_n} + d_{v_n+1}) \times 2$. We define as possibility of an entity v_{n+1} the division of $\frac{d_{v_n+1}}{coef}$ where coef is a coefficient value defined by the entity (e.g. int_{v_n}). The coef is set to the same value by the entire pool of entities. However, as time increases, and $d_{v_{n+1}} \neq 0$, the $d_{v_n} > (d_{v_n} + d_{v_n+1})$. Thus we conclude that $P(v_{n+1}) > P(v_{n+2}) > \cdots > P(v_i)$, so $P(\frac{d_{v_n+1}}{coef}) > P(\frac{d_{v_n+2}}{coef}) > \cdots > P(\frac{d_i}{coef})$; this encompasses that as the number of further disseminations increase the possibility to meet initial deadline is decreased as well.

4.6. Lemmas of energy-aware message-exchanging costs

4.6.1. Message size cost

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and v_{n+1} is the ending point and $\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$. For each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_{n+1} and vice versa after the ranking operation the size cost of communication is related to the size of each message sent and received divided by the capacity of the communication channel (c-the bandwidth) as given by the following formula.

$$CostSize_{MEO} = \frac{s_s + s_r}{c_{v_n:v_{n+1}}}$$

We propose that the $cost_{MEO}$ of the MEO operation offers always lower or equal cost results when compared with the $cost_{AllToAll}$.

Proof. Let $w_{v_n:v_{n+1}} \neq 0$ $(\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0)$ thus there is a trail from v_n to v_{n+1}. The cost operation defines the sum of the file sizes s_s and s_r where $s_r \leqslant s_s$ divided by the channel bandwidth. Thus if $s_r = s_s$ as happened in the AllToAll case, the $cost_{all\text{-}to\text{-}all} = 2 \times s_s$. In contrast, if $s_r > s_s$ then $2 \times s_s > s_s + s_r$. We conclude that $\frac{2 \times s_s}{c_{v_n:v_{n+1}}} > \frac{s_s + s_r}{c_{v_n:v_{n+1}}}$. In this case $cost_{AllToAll} > cost_{MEO}$ and this validates that if $s_r > s_s$ then according to lemma 4.4 MEO offers always lower costs results.

4.6.2. Message delay cost

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and v_{n+1} is the ending point with $w_{v_n:v_{n+1}} \neq 0$, $int_{v_n} \neq 0$, $\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$. For each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_{n+1} and vice versa after the ranking operation the delay cost of communication is related with the delay of each message to reach and return from a recipient divided by the interval time int_{v_n} as defined by the v_n entity where $\upsilon_{v_n} > int_{v_n}$ and $\upsilon_{v_{n+1}} > int_{v_{n+1}}$.

$$CostMessageNumber_{MEO} = \frac{\eta + \frac{\eta}{\theta}}{int_{v_n}}$$

We propose that as the interval increases the cost of the delay function decreases and based on lemma 4.2 it offers lower costs if $\eta \neq \eta + \frac{\eta}{\theta}$ and $\theta \neq 0$. Especially, if $dl_{v_n:v_{n+1}} = dl_{v_{n+1}:v_n}$, the $\sum_{i=n}^{\eta} dl_{v_i} > \sum_{i=n+1}^{\theta} dl_{v_i}$, thus the delay cost function can be represented as follows.

$$CostMessageDelay_{MEO} = \frac{\sum_{i=n}^{\eta} dl_{v_i} + \sum_{i=n+1}^{\theta} dl_{v_i}}{int_{v_n}}$$

We propose that the delay cost function can represented in terms of the sum of delays where the MEO solution always offers better performance with lower cost values compared with the AllToAll approach if $\sum_{i=n}^{\eta} dl_{v_i} > \sum_{i=n+1}^{\theta} dl_{v_i}$.

Proof. Let $w_{v_n:v_{n+1}} \neq 0$ ($\upsilon_{v_n} \neq 0$ and $\upsilon_{v_{n+1}} \neq 0$) thus there is a trail from v_n to v_{n+1}. The cost operation defines the sum of the delays to reach recipients in addition to the sum of delays for messages returned (in case of availability) divided by the interval that represents the coefficient value of the requesting entity. Thus if $\eta \neq \eta + \frac{\eta}{\theta}$ there is at least one responder that does not reply thus $\eta > \theta$ and if $dl_{v_n:v_{n+1}} = dl_{v_{n+1}:v_n}$, the $\sum_{i=n}^{\eta} dl_{v_i} > \sum_{i=n+1}^{\theta} dl_{v_i}$. If we compare this with the AllToAll message-exchanging, we conclude that in the former case $\sum_{i=n}^{\eta} dl_{v_i} = \sum_{i=n+1}^{\theta} dl_{v_i}$, and since $\sum_{i=n}^{\eta} dl_{v_i} > \sum_{i=n+1}^{\theta} dl_{v_i}$ (lemma 4.2) the cost of the delay using MEO is always lower.

4.7. Lemmas of energy optimization

4.7.1. Message path energy optimization

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and connected with v_{n+1} that is connected to v_{n+2}, etc. The connection continues to node v_b that forms the ending point and $\upsilon_{v_n} \neq 0, \upsilon_{v_{n+1}} \neq 0, \upsilon_{v_{n+2}} \neq 0, \ldots, \upsilon_b \neq 0$. For each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_b and forwarded among intermediate entities and $int_{v_n} > \sum_{i=v_n}^{v_x} d_i$ the best message distribution path is defined by the minimum of the sum of the costs for paths from v_n to v_{n+1} to v_{n+2}, etc. as defined by lemma 4.6. In other words, the minimum function $f(x)$ that calculates the cost $c_{v_n:v_b}$ defines the best path where $c_{v_n:v_b}$ defines the bandwidth of the channel from v_n to v_b.

Proof. Let $w_{v_n:v_x} \neq 0$ so there is a path among v_n to v$_x$. In particular due to the decentralized nature of the setting this could include multiple paths of the same message to reach the final destination where each message m$_y$ has a total delay less or equal to int_y according to lemma 4.1. Thus, different requests from remote resources are ranked (to the host location) according to a best path selection of the cost measures. In the case of $int_{v_n} > \sum_{i=v_n}^{v_x} d$ there is a possibility for job allocation to a remote location, thus the lowest cost value defines the best trail based on the selection of the cost function f(x) that it is either related to the cost$_{\mathrm{MEO}}$ or the total delay as follows.

$$\min[f(x)] = \begin{cases} cost_{MEO} = \frac{s_s + s_r}{c_{v_n:v_{n+1}}}, & if\ x = size \\ \sum_{z=v_n}^{z=v_{n+1}} dl_z + udl_a, & if\ x = delay \end{cases}$$

So, we conclude that the best path optimization for multi-level messaging is defined by the minimum cost operation.

4.7.2. Energy consumption optimization

Let $G(V,E)$ be a directed graph with non-negative edge weights, and suppose that v_n is the starting point and connected with v_{n+1} that is connected to v_{n+2}, etc. The connection continues to node v_b that forms the ending point and $\upsilon_{v_n} \neq 0, \upsilon_{v_{n+1}} \neq 0, \upsilon_{v_{n+2}} \neq 0, \ldots, \upsilon_b \neq 0$. For each message $m_i \in \{m_1, m_2, \ldots, m_n\}$ that is sent from entity v_n to v_b and forwarded among intermediate entities the energy consumption of the MEO approach is related with the cost message delay operation and given by the following formula.

$$ConsKW = \frac{watts \times CostMessageDelay}{1000}$$

We propose that the consumption rates in MEO are related with the cost of the delay function, as this is the consumption frequency in terms of timing, so based on lemma 4.5 MEO offers optimized consumption rates.

Proof. Let $w_{v_n:v_x} \neq 0$ so there is a path among v_n to v$_x$. If $\eta \neq \eta + \frac{\eta}{\theta}$ and $\theta \neq 0$ we assume there is at least one node that it does not reply back thus $\eta > \theta$. If we compare this with the AllToAll message-exchanging, we conclude that in the former case $\sum_{i=n}^{\eta} dl_{v_i} = \sum_{i=n+1}^{\theta} dl_{v_i}$, and since in MEO $v_{i=n}^{\eta} dl_{v_i} > \sum_{i=n+1}^{\theta} dl_{v_i}$ (lemma 4.2) the cost function is MEO consumption $\left(\frac{\sum_{i=n}^{\eta} dl_{v_i} + \sum_{i=n+1}^{\theta} dl_{v_i}}{int_{v_n}}^{MEO} > \frac{\sum_{i=n}^{\eta} dl_{v_i} + \sum_{i=n+1}^{\theta} dl_{v_i}}{int_{v_n}}^{all-to-all} \right)$. Thus the energy consumption rates will be optimized as well and that forms what we wanted to show.

It should be mentioned that the cost of communication could be calculated either by the size of the message or the delay of the communication link according to the configuration of the requesting entity. Next, we focus on the algorithmic structure and the ranking operations.

5. Use case scenario of inter-clouds

To demonstrate the messaging model we develop an inter-cloud case study. Specifically, an inter-cloud is an inter-collaborated setting of sub-clouds that exchange services in order to increase service elasticity [20]. In inter-clouds, a distributed manager is named as meta-broker and is responsible for service distribution and decision-making by having spontaneous knowledge of the environment. Here we employ this infrastructure in order to implement the MEO approach. Specifically, we associate meta-brokers with entities that exchange messages in order to allocate resources as in [6]. We present the conceptual design of the inter-cloud and its association with the MEO approach based on the implementation of algorithmic pseudo-codes. We employ the whole model in a novel inter-cloud simulator called SimIC [17] that automates the entire algorithmic process.

5.1. The algorithmic structure

To demonstrate the inter-cloud we detail two algorithms to demonstrate the MEO model. For each distribution, the meta-broker follows the Algorithm I for requesting resource availability.

Algorithm I: Message-Exchanging of requester

Require:	job_spec$_i$:	the job specification
	job_list:	the list of jobs formed during waiting
	ack	an acknowledgement message with the id of the responder
	req_mbr	the requester meta-broker
	res_mbr	the responder meta-broker from the meta-registry
	interval$_a$	an interval value for the job formation
	interval$_b$	an interval value for the message distribution
Methods:	send(job_spec$_i$, res_mbr, tag)	the method to send information to responder meta-broker
	get(ack)	the method to get acknowledgement from responder
	wait(interval$_a$)	the method to wait for responses
	rank(res_mbr)	the method to rank the inter-connected meta-broker
	put_sel_list(res_mbr)	the method to store selected responders into a list

```
1: wait(interval$_b$)
2: **for all** jobs $\in$ job_list
3:    send(job_spec$_i$, 0) to res_mbr
4: **end for**
5: **while** wait(interval$_b$)
6:    get(ack, res_mbr)
7:    rank(res_mbr)
8:    sel_list(res_mbr)
9: **end while**
10: **if** (sel_list(0))
11:    send(job_spec$_i$, 1) to res_mbr
12: **end if**
```

Algorithm II represents the responder meta-broker that collects the requests by waiting for an interval and executes a ranking function to classify requests.

Algorithm II: Message-Exchanging of responder

Require:	job_spec$_i$:	the job specification
	job_list:	the list of jobs formed during waiting
	ack	an acknowledgement message with the id of the responder
	req_mbr	the requester meta-broker
	res_mbr	the responder meta-broker from the meta-registry
	Interval$_c$	an interval value for waiting messages
Methods:	send(job_spec$_i$, res_mbr, tag)	the method to send information to responder meta-broker
	get(ack, req_mbr)	the method to get acknowledgement from responder
	wait(interval$_a$)	the method to wait for responses
	rank(job_spec$_i$)	the method to rank the inter-connected meta-broker

(*continued on next page*)

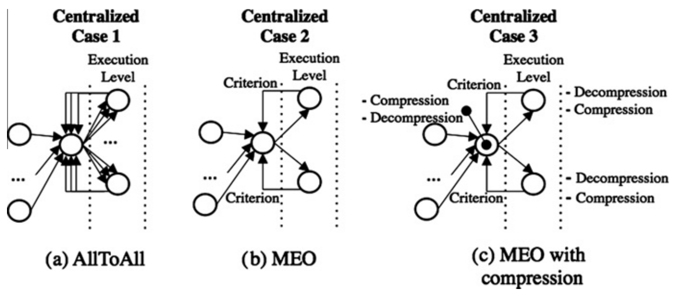

Fig. 5. The three centralized simulation cases include: (a) the AllToAll model, (b) the message-exchanging optimization model, (c) the message-exchanging optimization model with compression.

(*continued*)

Algorithm II: Message-Exchanging of responder

```
1: wait(interval_b)
2: while wait(interval_c)
3:   get(ack)
4:   for all jobs ∊ job_list
5:   rank(job_spec_i)
6:   end for
6: end while
7: if(match ← TRUE)
8:   send(ack,req_mbr)
9: else
10:   terminate()
11: end if
```

In Algorithm I, the meta-broker collects the jobs by waiting for an interval and creates a deferred queue. Then, for all the jobs in the queue it sends a unique request and waits for a response during a second interval that is set with regards to the experimental case. During that time it collects responses and executes a ranking function to select the best resource according to the ranking criteria. Finally, the algorithm sends the classified jobs to the selected resources by attaching a tag to notify the origin of the request. In Algorithm II, the ranking function decides whether the responder meta-broker could execute the request or not. The algorithm generates a flag value that is set to true if the meta-broker can execute the job, thus a message can be sent back to the requester. In any other case the flag is set to false and the algorithm terminates the job.

5.2. The simulation configuration

This section presents the description of the topologies that are utilized of this study. We have implemented various inter-cloud simulation cases to compare the AllToAll and the MEO model. The comparison is mainly based on the fact that AllToAll forms the default message exchanging policy for most of the distributed systems. Thus, the study uses the AllToAll solution as the benchmark for our comparison. We use SimIC [17] to configure a diversity of inter-clouds in terms of datacentre hosts and software policies where desired number of users could send single or multiple requests for computational power (cores, CPU, memory, storage, bandwidth), software resources (measured empirically in clock cycles per instruction and millions of instructions per second) and uptime of Virtual Machines (VMs) utilizations.

SimIC includes a variety of entities that have been modelled to achieve a diversity of meta-computing inspired requirements. These include large-scale distribution of job requests among meta-brokers similar to distributed management systems (e.g. grids). Meta-brokers decide the sub-cloud to execute services and distribute messages based on MEO. In SimIC the cloud (local-broker) is dynamically aware of the current computational capacity so it contacts the meta-broker. The former decides whether to execute jobs locally or to forward the request to other meta-broker. To demonstrate the effectiveness of our approach we have implemented two topologies in SimIC as follows.

(a) Centralized topology: The single level message distribution depicts a centralized system (e.g. a sub-cloud of an inter-cloud or a local resource management system-LRMS). In such case a collection of nodes (typically the sources) submits jobs to the centralized component (e.g. a local-broker) that forwards requests to internal resources for resource availability. In this topology we integrate three cases to simulate the AllToAll approach, the MEO model and the MEO model

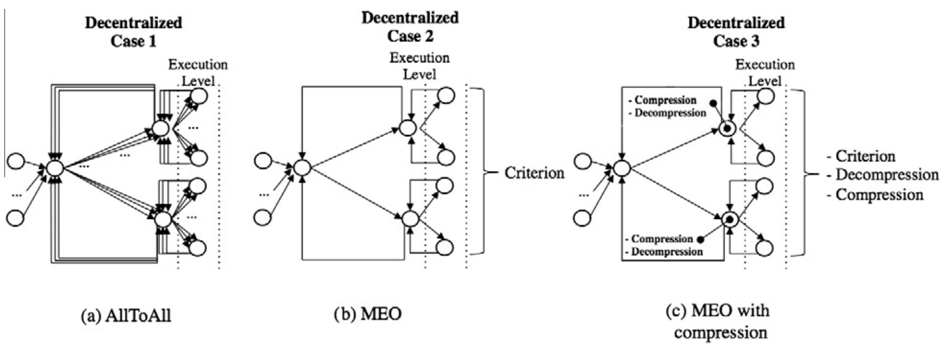

Fig. 6. The three decentralized simulation cases include: (a) the AllToAll model, (b) the message-exchanging optimization model, (c) the message-exchanging optimization with compression.

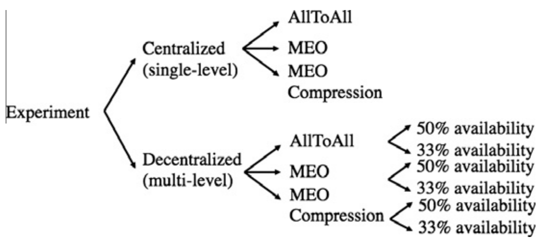

Fig. 7. The experimental map in SimIC.

with file compression. The ranking criteria define the cost of communicating that is measured from the total delay. In addition, we present the improvement factor of the MEO model when compared with the AllToAll approach. Fig. 5 shows the topology of the three cases the AllToAll, the MEO model and the MEO model with compression.

(b) Decentralized topology: The multi-level message distribution shows a decentralized system (e.g. an inter-cloud). Here, the source submits jobs to a node (decentralized meta-broker) that forwards the request to inter-connected meta-brokers. The latter redistribute the request to other meta-brokers, etc. We have configured a two level topology where we have implemented the three cases as previously. For each of the cases we have implemented two sub-experiments where we configure the resource availability to standard levels as follows. The first sub-case details that a request that arrives in the execution level (third level) will have 50% resource availability. The second sub-case includes that the resource availability (again at third level) will be 33%. In this way, we can demonstrate how the metrics are affected when the system has a medium or low resource availability. Fig. 6 illustrates the topologies of the inter-cloud configuration that includes decentralized meta-broker communication [6] for MEO.

5.3. The experimental analysis

This section presents the experimental analysis for comparing the centralized and decentralized cases of Section 5.2. Fig. 7 demonstrates the experimental map that includes comparison of centralized and decentralized simulations.

Table 2 shows the centralized topology specification that characterizes the experimental case of a typical cloud. We define as job size the MIPS (Million of Instructions Per Second). We also delimit the compression rate for both experiments to 29%, and the compression delay is set to 1.8 ms and the decompression in 3.8 ms.

Table 3 shows the centralized topology specification that characterizes the experimental case of a typical cloud. We define as job size the MIPS. The availability is set to 0% for the second level resource, 50% (sub-case a) and 33% (sub-case b) for all resources of the third level.

Fig. 8 demonstrates the energy consumption rates of the AllToAll approach and of the proposed MEO model for the centralized experiment. It is shown that for 100% availability the value of energy spent for the AllToAll approach is increased as more users enter the system.

Fig. 8 shows the high-low lines that represent the optimized degree of MEO approach. This is because 10 messages are transferred for each new user that submits 10 jobs. In contrast the number of messages is lower, so MEO indicates improved rates with a low-increased trend line. Fig. 9 shows the total delay of the AllToAll and MEO model for both cases. It is apparent that the increasing tendency of lines offer lower time delays for MEO as depicted in second *y*-axis (the scale for the MEO model). In addition, the increasing rate of the MEO trend line is lower than the AllToAll approach.

Table 2
The centralized use case specification.

Clouds	Users	Average delay	Number of jobs	Size of each message	MIPS	CPU	CPI	Bandwidth	Power (Watts)	Availability
3	1–5	10 ms	10	5 kb	100	2	3	512 kbps	300–500	100%

Table 3
The decentralized use case specification.

Clouds	Users	Average delay 1st–2nd levels	Average delay 3rd levels	Number of jobs	Size of each message	MIPS	CPU	CPI	Bandwidth	Power (Watts)	Availability
7	1–10	10 ms	30 ms	2	10 kb	1000	2	4	512 kbps	100	0%, 33%, 55%

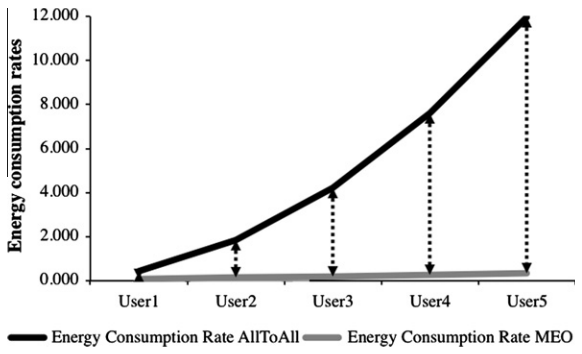

Fig. 8. Comparison of the energy consumption rates for AllToAll and MEO.

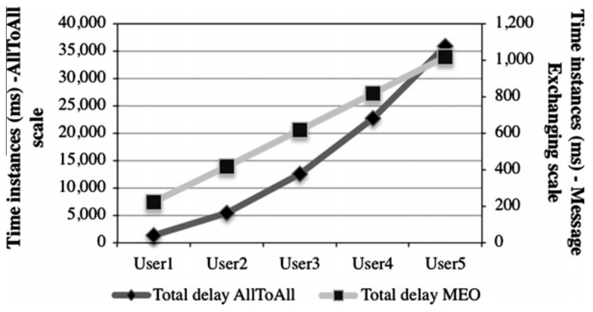

Fig. 9. Comparison of total delay times and polynomial trend lines for AllToAll and MEO.

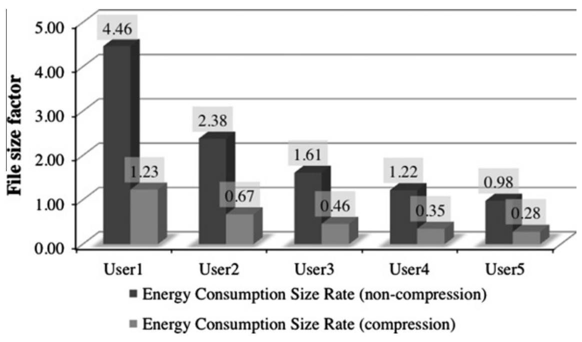

Fig. 10. Comparison of energy consumption rates of MEO with regards to size cost.

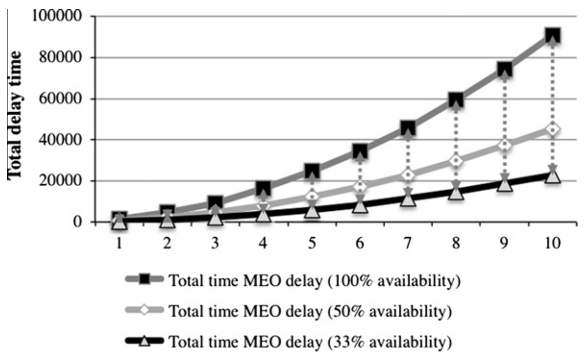

Fig. 11. MEO total delay times in decentralized multilevel submissions (high low lines for 33%, 50%, 100%).

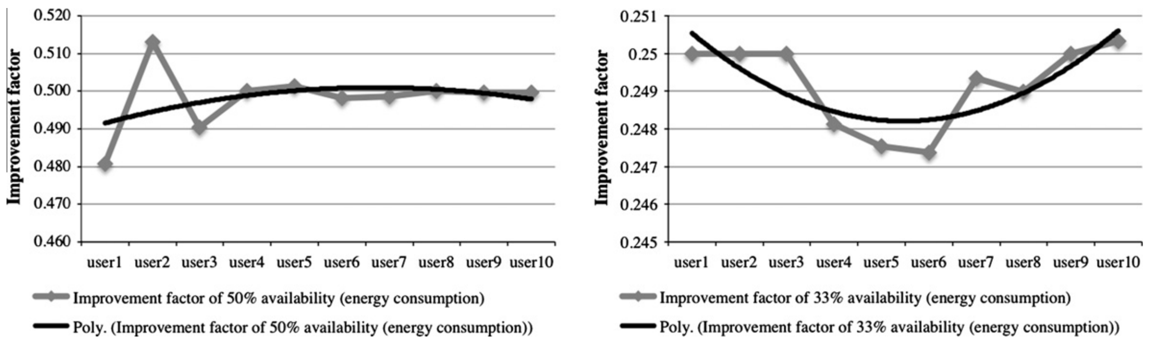

Fig. 12. Comparison of the 50% and 33% resource availability energy consumption MEO improvement rates.

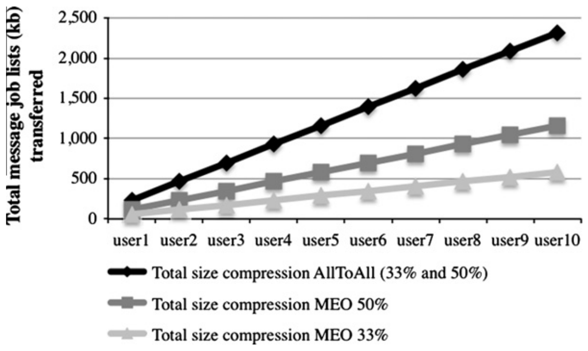

Fig. 13. Comparison of the total KBs transferred in cases of AllToAll and MEO (33% and 50%).

Fig. 10 illustrates the energy consumption rates for the MEO model file compression case. The same rates are selected for the AllToAll experiments as well.

We compare non-compression against compression cases in a centralized system. We conclude that as the number of users increases, the number of messages that contain compressed lists of job descriptions is optimized. The file size factor (y-axis) describes how the list size (of collected job descriptions in KB) affects the energy consumption of the communication channel in terms of available bandwidth. For the decentralized case, Fig. 11 demonstrates the MEO delay times in multilevel job distributions. The high-low lines show that as the number of users increases (thus their job submissions, e.g. 2 per user) the total delays for low resource availability (33%) has been optimized.

Based on these results, we conclude that MEO model optimizes its delay rates for low resource availability cases. This is particular useful for large-scale dynamic systems where multiple users request job allocations. So as the allocation number increases, MEO will offer low delays to remaining job submissions. Fig. 12 shows the comparison of the 50% and 33% resource availability energy consumption improvement rates of MEO. Specifically, for the case of 50% availability, the polynomial trend line decreases over the time however it remains in the range of 0.490 to 0.5. However, the second sub-case of 33% availability shows increasing rates for high number of users, but remains under the 50% rate (lower than 0.251), so offers improved results. It should be mentioned that this particular output is for the inter-cloud of 7 sub-clouds where 10 users submit 2 services one after the other with a delay of 10 ms as illustrated in Fig. 6.

At last, Fig. 13 illustrates the comparison of the value in kilobytes (KBs) that are transferred in the multilevel decentralized inter-cloud.

Specifically, it is shown that the AllToAll approach includes the highest number of data transfers, while MEO with 33% availability offers the lowest. In other words, in a busy inter-cloud the MEO model does not increase the transfer rates and the load of the channel. To conclude, this section presented the experimental analysis of the MEO model in both centralized and decentralized topologies. By implementing our solution in an inter-cloud system, we have shown that MEO outperforms the traditional AllToAll message exchanging in terms of energy efficiency rates (delay, power and size cost).

Further, we have focused on the decentralized topology and we have integrated two sub-cases where the inter-cloud offers medium (50%) and low (33%) resource availability. The aim here is to demonstrate that the MEO model offers optimized results in a highly demanding inter-cloud setting. That is to say that the comparison of the 50% and 33% resource availability energy consumption rates shows improvement for the second case. At last, we detailed the comparison of the total kilobytes transferred for the experimental case of AllToAll (33% and 50%), message-exchanging 50% and message-exchanging 33% resource availability. Results shows that the MEO does not add load to the communication channel.

6. Remarks and future work

This work proposes the MEO model in order to optimize the energy efficiency of communication in distributed systems. The proposed solution forms a simple but significant approach in terms of both energy and performance efficiency. By not receiving negative requests from remote entities we optimize fundamental metrics, e.g. total delays. Nevertheless, this implies a variety of costs and concerns with regards to the size of messages, the definition of intervals, the topologies of the system and the throughput of jobs. To answer these issues we presented a MEO model that optimizes a number of time-centric performance criteria as well as energy-aware measures (e.g. the energy consumption rates based on the uptime of resources).

The actual approach includes a mathematical representation of a graph theory model. To demonstrate the MEO approach in a real-case scenario we have implemented an inter-cloud simulation, where various services are submitted from users to meta-brokers for extracting resource availability. The simulation experiments draw a number of considerations as follows.

(a) The diversity of message exchanging latencies shows increased performance in terms of energy consumption rates.
(b) The collective model (operating in synchronous standards) optimizes the number of messages performance (e.g. for the configuration of the centralized experimental case the improvement factor is 3.5).
(c) The ranking procedure is considered as first come first served fashion, and for this case the energy consumption levels are improved as well.
(d) Both experimental cases show high adaptive-ness to various workloads and topologies.
(e) The decentralization offers high dynamic-ness (e.g. for cases of low resource availability) by slightly affecting performance due to meta-brokering message exchanging delays.

The future steps of this work include the exploration of different ranking techniques (based on job performance measures) for achieving a further optimization of our approach. In addition, we aim to implement a Message Passing Interface system for queuing host processors for information processing during run-time; thus achieving real-case solution. In addition different variations of VMs (number and configuration) could be included to demonstrate the heterogeneity of the system. With regards to energy consumption, measurements required to be validated in various workloads as in [12] and a variety of topologies for identifying supplementary optimization criteria. In terms of simulation we have utilized simple scheduling algorithms thus a more advanced solution could further improve results with regards to resource allocation, and service execution processes. Finally, a further extension of will include the utilization of historical data in the form of past job submission to optimize our model.

References

[1] R. Gupta, S.S. Vadhiyar, An efficient MPI_allgather for grids, in: Proceedings of the 16th International Symposium on High Performance Distributed Computing (HPDC '07). ACM, New York, NY, USA, 2007, pp. 169–178.

[2] A.L. Steffenel, E. Jeannot, Total Exchange Performance Prediction on Grid Environments: modeling and algorithmic issues, Towards Next Generation Grids, Springer, US, 2007. pp. 131–14.

[3] E. Chan, R. Van de Geijn, W. Gropp, R. Thakur, Collective communication on architectures that support simultaneous communication over multiple links, in: Proceedings of the 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2006, pp. 2–11.

[4] T. Kielmann, H. Bal, S. Gorlatch, K. Verstoep, R. Hofman, Network performance-aware collective communication for clustered wide-area systems, Parallel Computing 27 (11) (2001) 1431–1456.

[5] C.X. Mavromoustakis, D.H. Karatza, Quality of service measures of mobile ad-hoc wireless network using energy consumption mitigation with asynchronous inactivity periods, Simulation 83 (1) (2007) 107–122.

[6] S. Sotiriadis, N. Bessis, A. Antonopoulos, Decentralized Meta-brokers for Inter-Cloud: Modeling brokering coordinators for interoperable resource management, in: Proceedings of the 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'12), IEEE Computer Society, Chongqing, China, 2012, pp. 2475–2481.

[7] N. Bessis, S. Sotiriadis, F. Xhafa, F. Pop, V. Cristea, Meta-scheduling issues in interoperable HPCs, grids and clouds, International Journal of Web and Grid Services 8 (2) (2012) 153–172.

[8] F. Owusu, C. Pattinson, The current state of understanding of the energy efficiency of cloud computing, in: Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM ‘12). IEEE Computer Society, Washington, DC, USA, 2012, pp. 1948–1953.

[9] H. Tang, T. Yang, Optimizing threaded MPI execution on SMP clusters, in: Proceedings of the 15th International Conference on Supercomputing (ICS ‘01), ACM, New York, NY, USA, 2001, pp. 381–392. Mpich2 home page. <http://www-unix.mcs.anl.gov/mpi/mpich2> (accessed 20.06.12).

[10] MPICH-G2. <http://www3.niu.edu/mpi> (accessed 20.06.12).

[11] The message passing interface (MPI) standard. <http://www.mcs.anl.gov/research/projects/mpi/> (accessed 20.06.12).

[12] D.H. Karatza, Simulation study of multitasking in distributed server systems with variable workload, Simulation Modelling Practice and Theory 12 (7-8) (2004) 591–608.

[13] I. Foster, R.N. Jennings, C. Kesselman, Brain meets brawn: why grid and agents need each other, in: P. Ritrovato, S.A. Cerri, S. Salerno, M. Gaeta, C. Allison, T. Dimitrakos (Eds.), Proceedings of the 2005 conference on Towards the Learning Grid: Advances in Human Learning Services, IOS Press, Amsterdam, The Netherlands, 2005, pp. 28–40.

[14] Y. Cotronis, Composition of message passing interface applications over MPICH-G2, International Journal of High Performance Computing Applications 18 (3) (August 2004) 327–339.

[15] S. Sotiriadis, N. Bessis, N. Antonopoulos, Towards inter-cloud schedulers: a survey of meta-scheduling approaches, in: Proceedings of the 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC ‘11). IEEE Computer Society, Washington, DC, USA, 2011, pp. 59–66.

[16] A. Auyoung, B. Chun, A. Snoeren, A. Vahdat, Resource allocation in federated distributed computing infrastructures, in: OASIS ’04: 1st Workshop on Operating System and Architectural Support for the On-demand IT Infrastructure, Boston, MA, October 2004, 1–10.

[17] S. Sotiriadis, N. Bessis, N. Antonopoulos, SimIC: Simulating the Inter-Cloud, in: The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013), March 25–28, 2013, Barcelona, Spain, 2012.

[18] N. Bessis, S. Sotiriadis, F. Pop, V. Cristea, Optimizing the energy efficiency of message exchanging for job distribution in interoperable infrastructures, in: Proceedings of the 4th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS-2012), IEEE Computer Society, Bucharest, Romania, 2012, pp. 105–112.

[19] L. Adhianto, B. Chapman, Performance modeling of communication and computation in hybrid MPI and OpenMP applications, Simulation Modelling Practice and Theory 15 (4) (2007) 481–491.

[20] N. Bessis, S. Sotiriadis, F. Xhafa, F. Pop, V. Cristea, Meta-scheduling issues in interoperable HPCs, grids and clouds, International Journal of Web and Grid Services, InderScience 8 (2) (2012) 153–172.

[21] F. Pop, C. Dobre, C. Stratan, A. Costan, V. Cristea, Dynamic meta-scheduling architecture based on monitoring in distributed systems, International Journal of Autonomic Computer 1 (4) (2010) 328–349.

[22] K. Leal, E. Huedo, I.M. Llorente, ‘A decentralized model for scheduling independent tasks in federated grids’, Future Generation Computer Systems 25 (8) (2009) 840–852.

[23] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, B. Hirsbrunner, Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm, Future Generation Computer Systems (2011) 402–415.

[24] D. Jackson, Q. Snell, M. Clement, Core algorithms of the Maui scheduler, in: Job Scheduling Strategies for Parallel Processing, Springer, 2001, pp. 87–102.

[25] A. Iosup, T. Tannenbaum, M. Farrellee, D. Epema, M. Livny, Inter-operating grids through delegated matchmaking, Scientific Programming 16 (2) (2008) 233–253.