Peer-to-peer Overlays for Resource Discovery

Filipe Rocha Paredes

Instituto Superior Técnico, TagusPark Campus
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Lisboa, Portugal
filipe.paredes@ist.utl.pt

***Abstract* - There are currently a variety of projects that try to improve the performance of applications by using spare cycles from other computers connected over the Internet and, sometimes in return of their own spare cycles in the future. Nonetheless, none of those projects allows, in large scale, home users to run unmodified desktop applications faster without an infrastructure for resource sharing that implies the use of a specific client-application in the computer host.**

To address such problem, this dissertation proposes mechanisms for resource discovery (e.g., CPU) to exploit different peer-to-peer network topologies that maximize the system performance metrics. This solution is part of an existent project, GINGER, that aims for the synthesis of three approaches: institutional grid infrastructures, popular cycle sharing applications and massively used decentralized P2P file-sharing applications.

The solution seek the development of a P2P middleware infrastructure based on the concept of a Gridlet, a semantics-aware unit of workload division and computation off-load. Several criteria, like bandwidth or resources availability are subject to analysis for the choice of neighbours in the peer-to-peer network for the routing of Gridlets.

I. INTRODUCTION

With the growing Internet access and increased capacity (processing, memory, storage, etc.) of personal computers, the computational power that can be obtained through the use of idle resources available in these machines should not be neglected. Aiming to exploit these resources, infrastructures and applications of Institutional Grids and Peer-to-Peer (P2P) overlays have been developed, which have allowed the use of such resources, including the performance improvement of parallel applications (Grids) or file sharing between multiple machines connected to the Internet (P2P).

Many communities in the Internet have witnessed a major expansion and popularization of P2P applications to share resources, either of processing cycles (SETI@home [1]) or from files.

The Grid Computing has been developed as a new generation computational model, both in the scientific world and commercial world. The spread of this technology encouraged the development of various tools in order to facilitate access to resources in Grids.

The use of distributed processing cycles emerged initially with applications such as SETI@home that follows a client-server model, where a central server distributes tasks to customers who voluntarily offer their cycles. After the execution of these tasks during idle periods of the machine, the results are sent to the central server.

Over the past few years a large number of proposals have been presented that attempt to establish a link between the institutional grid infrastructures (e.g. Globus [2]), popular cycle-sharing applications (e.g. SETI@home [1]) and decentralized P2P file-sharing applications. However, none of those infrastructures allows common users to exploit parallel execution for improved performance in popular applications, by using idle cycles from other users.

Given the increasing development of technologies such as institutional grids and P2P technologies, this work tries to fill the gaps left by the infrastructures that try to synthesize cycle-sharing applications to the previous mentioned technologies. The solution proposed in this report designs a system for resource discovery in a P2P overlay aiming to exploit different network topologies that maximize the system performance metrics. This system consists in a middleware P2P infrastructure based on the concept of a Gridlet, a semantics-aware unit of workload division and computation off-load. Popular applications, without any necessary modifications, have their tasks executed by other idle cycle's machines by sending the necessary data to the developed system that creates and submits the Gridlets to the network. These Gridlets are processed and returned as Gridlet-results to the original node.

The remaining of this paper is organized as follows. In section 2, it's presented an overview of some of the existing related works. In section 3, a system overview is presented. Later, in section 4 the system architecture is described. Then, in section 5, the mechanisms of resource discovery are explained. Performance evaluation of the proposed system is presented and discussed in section 6. Finally, in section 7 concludes this work and outline futures directions.

II. RELATED WORK

The P2P computing [3] has promoted a big change in the patterns of Internet usage in recent years. Its great advantage in relation to computing client / server, is the possibility of direct collaboration between users, without relying on centralized servers. Systems such as the Gnutella network [4], a virtual overlay network on the Internet, unstructured, totally decentralized, provides the advantages of P2P technology.

The system Chord [3] is an infrastructure for location and routing in P2P which performs a mapping of files identifiers. The location of data passing through implemented in Chord identification data (files) with keys and saving the pairs (key, data) mapped the keys in nodes.

The Pastry [5] is a basis for scalable routing and location of objects distributed P2P applications for large-scale. The Pastry

plays the routing in the implementation and location of objects in a vast network overlay of nodes connected through the Internet. This can be used to support a variety of P2P applications, including data storage, data sharing and communication between groups.

In the Grid technology is distinguished from conventional distributed computing by its focus on shared resources on a large scale, innovative applications and in some cases, high-performance orientation [6].

The Globus project [7] is a project that caused great impact in the area of Grid Computing. Its system is called Grid Computing in Globus Toolkit and provides a series of features that allow the implementation of systems in Grid Computing as well as the development of applications for such systems.

Currently, we can find a vast computational power of the hundreds of millions of personal computers around the world. The computing resources from public gets huge computations distributed through the collection of resources on idle computers connected to the Internet.

The BOINC [8] is a distributed computing platform developed at Berkeley. Exceeded his original project, the SETI @ home, and now incorporates a large number of related projects. Its operation is based on the notion of units of work but is not flexible. All units of work are defined as having the same computational cost and bandwidth, determined in each project.

III. System overview

In the context of the Ginger project [9], this work emerges from the creation of a platform, capable of synthesize Grid infrastructures, P2P applications and cycle-sharing applications, exploring different P2P network topologies that maximize certain system performance metrics (e.g. bandwidth consumption or a task's processing time) intended for the choosing of neighbours in the network. The importance of network topology stems from the fact that resource discovery mechanisms follow the links formed by the P2P overlay network topology. Thus, the main goal of the system will be the correct routing of requests that must have into account the associated computation cost and various performance criteria that define the best choice, as the bandwidth of the connection or the available resources in the node.

The system developed is a middleware platform on a structured P2P overlay network that bases its operation around the concept of a Gridlet. A Gridlet is a fragment of data, capable of describing all aspects of a work task, as well as the necessary changes for processing the data. When a work is submitted by an application for processing, it is partitioned into small tasks that are used to generate Gridlets, which will be submitted into the overlay where will be processed by other nodes. When the computation is complete, the results can be sent, in the form of Gridlet-results directly to the sender node or become available in the overlay.

A. The Overlay

The desired solution requires a robust peer-to-peer overlay. Pastry [5] is a scalable and efficient peer-to-peer overlay routing. This P2P network represents a perfect structured overlay over the Internet for the proposed system since it contributes with a good quality of P2P properties such as self-organization of nodes, completely decentralization and fault-tolerant. Moreover, the Pastry overlay provides with a neighbour set for each created with a heuristic proximity that includes a limited number of the geographically nearest nodes.

B. Submission and processing of requests

Each node can submit requests in the form of Gridlets. These Gridlets will carry the necessary data for the task computation and the cost associated to it. Since the goal of this system does not address the interaction with the desktop application, or the division of tasks and its processing, the contents of the data that Gridlets transport are irrelevant. Thus, the cost of computing a Gridlet is pre-set. The processing of a Gridlet should only result in a reduction of the local resources indicated by the Gridlet cost and in the consumption of the Gridlet process estimated time.

C. Resource Discovery

In order to make accessible resources shared by other machines connected to the overlay,, an implementation of a resource discovery mechanism is needed so that resources are found and engaged efficiently. The main functionality of the solution consists of discovering and to manage the information related to the resources of a limited number of nodes in the network, for example, the whole neighbour set provided by Pastry. By sending update-type messages, each node will announce its resources, only to those nodes that belong to the node's neighbour set. When a request is submitted by a node, it checks the information provided by its neighbours and forwards the Gridlet to the node that seems more capable of process the Gridlet. The selection process of the best node to forward is a delicate and crucial process that guaranties an effective and efficient resolution of the tasks.

D. Retrieving the results

Sending the processed results for a cache of files on the P2P overlay (like PAST [10]) is the most flexible method for retrieving the results and ensures privacy, since no identifier is needed in the original Gridlet. However, it presents some drawbacks, like the latency obtained by the insertion into the cache and, additionally, the system should provide with an estimated time of the transmission delay, Gridlet's process time and insertion of the result delay to schedule a retrieval of the results from the cache. For replication purposes, the result is already stored in cache.

This solution fits better with the project goals, giving more primacy to flexibility and privacy, rather than time efficiency during the retrieval of results.

IV. architecture

Similarly to the architecture of the project witch this system is based on, Gigi [9], the architecture of the GiGi application proposed is structured in layers. The running environment for this system is controlled by an additional component, the GiGiSimulator, responsible for establishing and monitoring

the overlay. The proposed system consists in a GiGi application, an overlay network and a simulator. The interaction between each component determines how the system works.

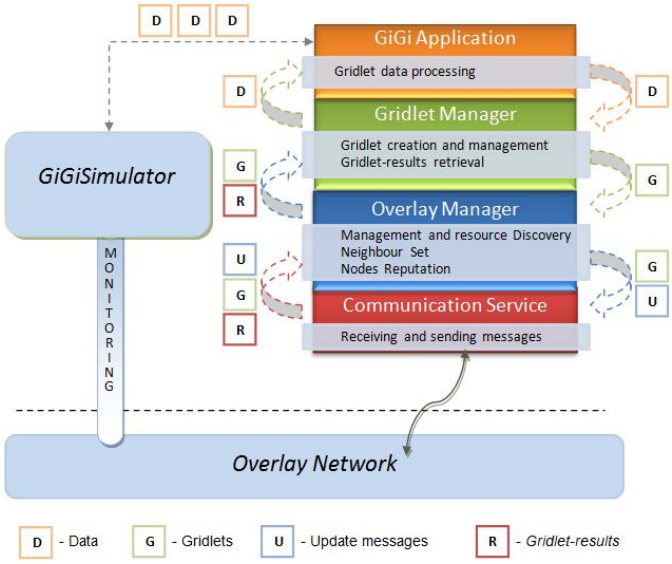

Fig. 1. System Architecture

Fig. 1 shows the architecture of the system illustrating its components, their interactions and a description of the main features of each layer. The GiGiSimulator is responsible for the simulation of the system, creating an overlay network with a customizable number of nodes and for each node of the network a Gigi application, which includes the four layers represented in Fig. 1. Each layer of the application interacts with the layer immediately above and below.

Following, we have a description of the components that make part of the system's architecture.

A. GiGiSimulator

This component simulates the operation of the whole system and overlay, serving as a support for the network and enables an interaction through a command interface with the Gigi application. Its role extends from generation of events on the network and on the GiGi application that is executed in each node, until the monitoring of messages transmitted between the nodes and a gathering of statistics relating to the activities monitored in the network.

The establishment of the network is launched in the simulator with the creation of the first node. When a node starts up, it can either join an existing ringed type network, or start a new one. If a ring doesn't exist yet, then the node will start its own ring. But once the first node has started a ring, all new nodes will bootstrap off of any one node in the existing ring, joining the network. After the network is complete, the Gigi applications are launched on each node of the overlay. Through a command-line interface in GiGiSimulator, it's possible to generate events on the network such as entry or exit of nodes or in applications such as the submission of requests.

Since this system only concerns about the network level of the original project [9] some of the interactions between the GiGi application and desktop applications or the aggregation of the results into one complete result are not taken into account.

B. GiGiApplication

The GiGi application layer has just a representative role in the structure of the Architecture since there are no interactions with real desktop applications. This layer was created to convey an idea of completeness in the GiGi application structure. Its functionalities are: transmission of data received from the simulator to the layer of Gridlet Manager and reception of Gridlet-results from the Gridlet Manager to notify the simulator of the completion of a task.

Since Gridlet's data computation is simulated, this process is executed in this layer and it's accomplished by reducing the resources in accordance to the cost of the Gridlet and by the consumption of time needed to process the Gridlet. No computation is actually applied to the data.

C. Gridlet Manager

The Gridlet Manager deals with all operations being carried out with Gridlets. It is in this layer that Gridlets are created from the data received from the previous layer. After their production they are sent to the bottom layer, the Overlay Manager.

All messages of the type Gridlet received from the network are automatically routed to this layer which will be analysed. According to the current availability of the local node, a decision is taken about a Gridlet to be processed by the node in the layer of Gigi Application or to be sent back to the Overlay Manager forwarding to another node in the network.

This layer is also responsible for retrieval of results from the submitting Gridlets. As there is no entity that simulates the desktop application, the results are just collected and sent to the layer of Gigi Application which will notify the simulator about the completing the task.

D. Overlay Manager

The Overlay Manager is responsible for routing and addressing in the overlay network. The discovery and management of resources in the network are also made in this layer. The resources of the local node are controlled by this component, which performs operations to reduce or increase their values. When changes in resources occur, they are announced to the nodes of the local node neighbour set throughout update messages.

The neighbour set is established and managed at this level. This set is created based on physical proximity that separates the nodes. Moreover, this layer maintains all the information about the availability of resources that each node of his neighbour set has. During the selection of a node as destination of a Gridlet, this information will be assessed according to a set of criteria that aims to obtain the better available node.

When a Gridlet is received from the Gridlet Manager, a node with sufficient available resources to address this Gridlet

is selected as target for its routing. The statistics results of choosing that node for routing a request are stored in a table of reputation regarding that node. The Overlay Manager uses this table when, on the whole neighbour set there are no availability that meet the cost of a Gridlet. In this case, the selection of a node to route the Gridlet is based on the reputation table that as information regarding previous statistics results, as cases of failure and who had less delay in processing the applications.

E. Communication Service

The Communication Service layer deals with all communications between the Gigi application and overlay network. The effective transmission and receiving of messages to and from the overlay is in this layer. All messages received from the Overlay Manager are sent to the network. The network contacts the Communication Service layer when there is a message addressed to the node associated with the application. This message is received and forwarded to the Overlay Manager.

The overlay also notifies this layer about any joins or exits from nodes, all of which occur in the vicinity of that node. The actions to these changes are performed by the Overlay Manager layer.

F. Overlay

The DHT network overlay used is Pastry [5]. The network nodes are connected in a ring-based topology in the order of its identifiers. In the Ids space a node links to other two nodes, one with the previous node that has the Id immediately lower and the later node that has the Id immediately higher. The assignment of identifiers is done randomly meaning that nodes closed in the Id space can be geographically dispersed.

This type of structure provides a set of properties key to the sustainability of the system: decentralization, the nodes self-organize amongst themselves without the need for any kind of central coordination or the a super-node; scalability, the network will operate properly for large numbers of nodes; faults tolerant, the network will be reliable even with the constant input, output and failures of nodes.

G. Messages Types

In this system there are two main types of messages that are sent across the network: the Gridlets and the update messages. The Gridlets form the basic unit of work requests generated by the simulator. Later, these messages are put on the network, to be processed by nodes with sufficient availability, depending on the cost that is associated with each Gridlet. In turn, update messages, whose goal is the dissemination of resources and the availability of a node, are transmitted when changes occur in a node's availability. The message contains the available resources of the sender node and, optionally, the duration of that availability (as described in [11]). Each node sends only information about their own resources and propagates these messages only to nodes listed in its set of neighbours.

There are more two types of messages propagated by the Gigi application, the Gridlet-results and ContentResult messages. The Gridlet-results are Gridlet-type messages, varying only the purpose of its content. The data field have results of the computation on the data from the original Gridlet and the cost associated refers to the effective cost of the processed task. Messages from ContentResult are messages that encapsulate the Gridlet-results so they can be sent through the system cache used, the PAST [10].

V. Resource Discovery

The Gigi's Overlay Manager layer is primarily responsible for the discovery and management of available resources in the overlay. The concept of Gridlets, used this system, cannot be reduced to the simple injection of such messages on the network and expect their propagation throughout the nodes until it finds a node available to handle the job. Thus, it is necessary to know forehand the availability of the network before tasks are submitted.

Each node announces its resources, by sending update messages, only to nodes that belong to the node's neighbour ser. When a node has pending requests for submission, it checks the information about its neighbours and forwards the Gridlet to the node that find it more suitable for the job. If none of the neighbours have necessary availability to process the request then it is sent to the node in the neighbour set that will have better chance (based on previous records) from forwarding the request to other nodes with availability.

A. Neighbours set

Right from the beginning, crucial elements to allow resource discovery are initialized in the Overlay Manager. In the final phases of network creation the nodes announce their presence and their resources through update messages and each one build its neighbour set. This set of nodes is provided by Pastry and is built on a metric of proximity between nodes, including the *n* (value varies depending on the configuration and size of the network) geographically closest nodes, that is, with the lowest values of RTT.

However, there may be situations where a given node has in its neighbour set nodes that do not see him as a neighbour. For example, to sets where *n* equals 20, the node A can see the node B as one of its 20 closest and node B can have 20 nodes which are closer than A. These situations become very commonly for large networks. To ensure a minimum of symmetry in the relationship between the neighbours a method was defined that when the node A announce its availability to node B, B will accept A as his neighbour adding it to his neighbour set and announces his availability to node A. To limit the uncontrolled growth of the sets, a criteria for choosing a node as a neighbour was defined: B will only accept A as a neighbour if A possesses an identifier numerically smaller than B; this ensures convergence to stability. Otherwise node B returns the update message to A. The node A, upon receiving his own message it realizes the rejection of B and excluded him from his set.

B. Select best available node

The definition of a node with better availability is one that has higher availability according to a weighted measure of the defined metrics (proximity, CPU, memory and bandwidth).

Each metrics used to define the available resources contribute, in general, with similar weight in the weighted calculation of a node's availability. These metrics represent the resources available in this node, so it is the preferred a choice to a node capable of meeting the demands and continue with available resources. The factor of proximity may also have great importance on this choice, to the extent that we nodes placed relatively close and with availability to handle the tasks, avoids the spread of such requests or long transmissions over the network and thus restricting the allocation of resources into the closest nodes.

C. Select best unavailable node

When there are no nodes with sufficient availability to handle a task, the selection of a node should is carried based on the node's ability to forward the request to other nodes that are able to process the task. This ability can be found on the historical records of results statistics from requests previously sent.

Historic records of statistics results from past routes through a certain node are maintained in a reputation table. These records indicate the number of failed results, the number of times that a request back to sender node and the number of attempts used to retrieve the results. These measures, being measures of failures or inefficiencies allows setting a level of rejection on a node through the following weighted calculation:

- Failures x 0.7 + Back to origin x 0.05 + Retries x 0.25

The bigger the result value, the higher the level of rejection for that node. For this reason, greater weight is given to the number of failures. It would be obvious from the outset, to exclude those nodes from where they had found many failures, however, this is not advisable since these are measures based on past actions and the actual performance does not depend on the node in question, but on its neighbours where the entry of a new node neighbour or an increase in availability in the vicinity of the node allows the recovery of reputation on it. As for the other measures, both ensure that the work will be properly done and returned with only implications of time or excessive number of retransmissions, in the case of Return to the Source with less importance on this last one.

VI. SIMULATION AND PERFORMANCE EVALUATION

Various tests have been executed for measuring and evaluating the performance of the created system. Such tests consist on simulating the flow of messages throughout the network on distinct scenarios that the simulator allows through the modification and combination of network parameters and variations in the system configuration parameters. For this objective, we used the Freepastry's simulator were a Pastry P2P overlay network can be simulated with the platform developed in this report.

During the simulations, the nodes in the system will be divided into 2 groups: a group of host nodes, which provides its processing cycles for performing work from others and another group of client nodes, who will submit requests to prey on idle cycles available in the overlay.

Each test shall have control over the following configuration parameters:

1) The number of nodes in the network;
2) The percentage of the types of nodes, host or client;
3) The total availability in the network;
4) The number of Gridlets the submit;
5) The submitted Gridlet computing cost (in units);
6) The number of client nodes, that is send requests.

In addition, variables which influence the routing of tasks within the system can also be configured, with the aim of revealing the best option values. At the results of each test are analysed and evaluated.

All tests were performed on a single machine with the following characteristics: an Intel Core 2 Duo T8300 2.40GHz, 3070 MB of RAM and OS Windows Vista 32-bit. The system was implemented on the implementation of Freepastry version 2.0_01, running on the platform NetBeans IDE 6.0.1 with Java JDK 1.6.0_06.

A. Procedure

The gain provided by the resource discovery mechanisms in the process of nodes selection to forward the requests can be obtained by considering the difference between the normal execution of the system and the execution of the system changing configuration values or disabling the existent mechanisms.

Two aspects that influence the node selection to forward to were tested: a) measurement of the information about the neighbour's availability; b) the ability that nodes have to learn about their neighbours by keeping historical records of statistics results.

In the first test a), the information about the availability of a node is obtained from a weighted measure on the resource's node and its proximity to the local node. It were tested various executions with different weights in the calculation of this measure. For effective measurements, tests should occur in scenarios were the system provides with enough resources, were the availability in the whole network meets the demand (point of saturation) and situations of excessive demand that the network can not immediately provide. **Variables**: the number of Gridlets to submit range from 300, 500 (1st point of saturation), 700, 900, 1000 (2nd point of saturation) and 1200 Gridlets; and the variation of the weight of metrics (CPU, memory and bandwidth band) and proximity in the node selection calculation;

The second test b), concerns about the reputation system maintained in each node, where they acquire information about the statistics results of their neighbours in the past and learn the best ways as they send more requests to the network. The gain of using this mechanism can be obtained from the difference between the performance of the system with and without the execution of the reputation mechanism, given that each simulation should occur sequentially and should always be the same node to send requests to allow that the system

evolve and learn about the neighbours where the tasks were submitted. **Variables:** execution with and without the reputation mechanism enforcement.

TABLE I
PARAMETER OF RESOURCE DISCOVERY MECHANISM TEST

Parameters	Value	Parameters	Value
Number of nodes	1000 nodes	Node types	500 clients 500 hosts
Network availability	[4000, 4000, 4000] [CPU, memory, bandwidth]	Gridlets cost	High Cost [8,8,8]
Client nodes	100 nodes	-	-

The major goal of this test is to evaluate the performance and the extra efficiency obtained from the discovery and resources management methods used in this system.

B. Test results

For test a) on the measurement of availability, three different weighted calculations were set on the node‘s availability metrics. The first calculation only evaluates the availability in terms of resources, distributing the weight equally for the metrics: 33% for the CPU, 33% for memory and 33% for bandwidth. The second calculation only evaluates the proximity of the node. And finally, a last calculation weights the two measures, favouring resources: 40% to proximity and 60% to resources shared equally for each metric in 20%.

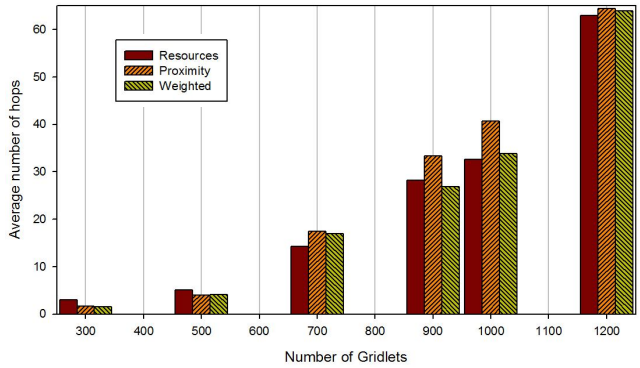

Fig. 2. Average number of hops in test a)

In Fig**.** 2 is possible to observe the resource discovery quality obtained by the three calculations. Analysing shows that for tasks for a number of Gridlets less than or equal to 500 the calculation based only in resources obtained the worst results, but above the 500 requests it has improved, compared with other calculations performance. Please note that until the point of saturation (500) there isn’t any lack of resources in the network, but from that point on the lack of resources is a constant. Therefore, we can infer that the calculation based on resources is favourable for situations of solicitation in excess of resources. The calculation based on the nodes proximity, has good quality efficiency as long as there are many resources available on the network and very low quality in situations of immediate lack of resources, since it is from 700 Gridlets submitted that gets a more growth on worse results.

In situations of massive demand for resources, the nodes that we choose with the increased availability will have greater chances of being successful, since sending a request for a node with a capacity to treat only one Gridlet, may occur that another node also has sent a request at the same time and consequently, who arrives first will be the chosen, relaying the other. Choosing the node with increased availability will have more chances to process an application, and a greater likelihood of this node, if fully occupied, finishing a task and be able to process the request as soon as possible. In such a situation, the proximity between nodes does not affect in any way the node’s ability to handle or not a request, and in some cases may even hinder the discovery of resources if the only available ones are distant.

In a situation of abundant availability, the calculation based on resources is irrelevant. Then, the calculation based on proximity gains efficiency since we chose the closest available.

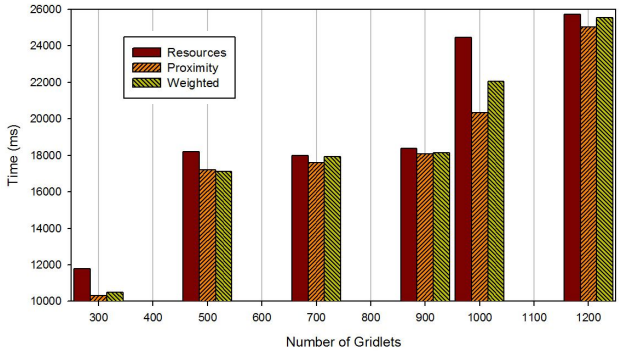

Fig. 3. Total simulation time in test a)

In terms of time required for completion of the 700 Gridlets, we can see in Fig. 3 that the calculation based on resources can be time consuming, in all situations. Even in situations of lack of resources this calculation that promotes efficiency (see Fig. 2) has higher time consumption. This cost corresponds to the wasted time in wrong choices due to favouring of nodes with more resources that can be geographically more distant (higher latency). But the calculation by proximity, as expected, has the best times in all situations cause gives primacy to the selection of the closest nodes. The weighted calculation gets intermediate values comparing the two previous calculations. Also with great resource consumption the weighted decision also produces the lowest time.

Thus, the weighted calculate based on resources and proximity (favouring the resources) provide the best results in most cases, as it takes the best of the two measures in both situations with and without availability. We can thus conclude that the calculation would be ideal with dynamically adjust the values of the weights on the resources and proximity according to the availability in the network. When the network abounds in resources, should be given a greater or total weight to proximity, taking advantage of its speedy completion of requests over the time lost in the discovery of nodes with

greater availability for selection based on resources. In situations of scarcity or lack of resources in the network should be assigned a greater weight to resources and a significant weight to the proximity, taking the selection on nodes with more resources and taking advantage of the time factor in the selection by proximity.

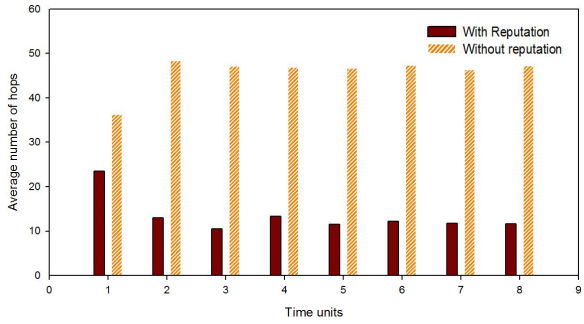

Fig. 4. Average number of hops in test c)

In test c) the reputation system, the simulation is executed always with the same network nodes and the tasks are all submitted from the same node. The rate of tasks is always the same, 700 Gridlets in order to test the behaviour of the system when there is a lack of resources on the network. Just for these situations the mechanism target can operate in a relevant and influence the results.

According to Fig. 4, it is possible to observe the routing quality obtained using the reputation mechanism. The smallest difference occurs during the first iteration since the reputation system has not yet acquired the information about their neighbours. In the second iteration is already visible a large reduction in the number of retransmissions made, remaining at that level from that point. So we can say that this mechanism converges very quickly to his best performance.

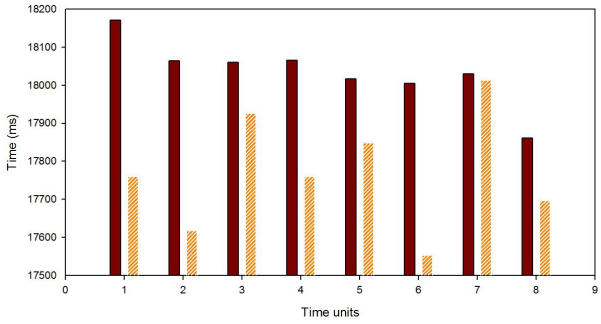

Fig. 5. Total simulation time in test c)

However, the great routing efficiency with reputation loses the time required for completion of tasks (see Fig. 5) for the execution without reputation. But it is important to note that the difference is very low, for example, for the first iteration, the difference is only 400 milliseconds.

VII. CONCLUSION

The great achievement of this work resides in the fact that popular applications can improve, in a transparent manner, its performance through parallel execution of their tasks using processing cycles of surplus from other machines belonging to the same overlay. In contrast to previous approaches in this area, the proposed solution enabled successfully exploitation of idle resources on the network by users with common generic applications without the need of any modifications or use of API, Libraries, or specific programming languages.

The execution of tests on the system developed has shown the success of the proposed features (discovery of resources and efficient delivery of Gridlets) in operating the network topology based on performance metrics (CPU, memory, bandwidth and vicinity). The use of reputation mechanisms also allowed to achieve more efficient with regard to the delivery of Gridlets.

VIII. FUTURE WORK

With the future pointing to a greater interconnect between all types of machines and devices the attention to the level of security should not be neglected for such applications. From shared data protection by the network to the control of access to resources from other machines, we must prevent the system from an array of threats that exist in networks today.

The tests show us various performances for different weights values in the calculation of the selection criteria. The study of the state of the network and consequent configuration of variables at run time could improve performance in abnormal conditions. Variables like the weights assigned calculate the performance metrics are examples of changes that would raise significant variations the final results.

REFERENCES

[1] D.P. Anderson et al., “SETI@ home: an experiment in public-resource computing,” *Communications of the ACM*, vol. 45, 2002, pp. 56-61.

[2] I. Foster e C. Kesselman, “Globus: a Metacomputing Infrastructure Toolkit,” *International Journal of High Performance Computing Applications*, vol. 11, 1997, p. 115.

[3] S. Androutsellis-Theotokis e D. Spinellis, “A survey of peer-to-peer content distribution technologies,” *ACM Computing Surveys (CSUR)*, vol. 36, 2004, pp. 335-371.

[4] Y. Chawathe et al., “Making Gnutella-like P2P Systems Scalable.”

[5] A. Rowstron e P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, vol. 11, 2001, pp. 329-350.

[6] I. Foster, C. Kesselman, e S. Tbecke, “The anatomy of the Grid,” *Grid Computing: Making the Global Infrastructure a Reality*, 2003.

[7] I. Foster e C. Kesselman, “Globus: a Metacomputing Infrastructure Toolkit,” *International Journal of High*

Performance Computing Applications, vol. 11, 1997, p. 115.

[8] D.P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," *5th IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 365-372.

[9] L. Veiga, R. Rodrigues, e P. Ferreira, "GiGi: An Ocean of Gridlets on a" Grid-for-the-Masses"," *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, 2007, pp. 783-788.

[10] A. Rowstron e P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," *ACM SIGOPS Operating Systems Review*, vol. 35, 2001, pp. 188-201.

[11] I. Filali, F. Huet, e C. Vergoni, "A Simple Cache Based Mechanism for Peer to Peer Resource Discovery in Grid Environments," *Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)-Volume 00*, 2008, pp. 602-608.