Trans-Social Networks for Distributed Processing

Nuno Apolónia, Paulo Ferreira, and Luís Veiga

INESC ID Lisboa / Technical University of Lisbon, Rua Alves Redol 9, 1000-029 Lisboa, Portugal, nuno.apolonia@ist.utl.pt, paulo.ferreira@inesc-id.pt, luis.veiga@inesc-id.pt

Abstract. A natural succeeding process for the Internet was to create Social Networks (e.g. Facebook, among others), where anyone in the World can share their experiences, knowledge and information, using personal computers or mobile devices. In fact, Social Networks can be regarded as enabling information sharing in a peer-to-peer fashion. Given the enormous number of users, sharing could also be applied to the untapped potential of computing resources in users' computers.
By mining the user friendship graphs, we can perform people (and resource) discovery for distributed computing. Actually, employing Social Networks for distributed processing can have significant impact in global distributed computing, by letting users willingly share their idle computing resources publicly with other trusted users, or groups; this sharing extends to activities and causes that users naturally tend to adhere to.
We describe the design, development and resulting evaluation of a web-enabled platform, called Trans-SocialDP: Trans-Social Networks for Distributed Processing. This platform can leverage Social Networks to perform resource discovery, mining friendship relationships for computing resources, and giving the possibility of resource (not only information) sharing among users, enabling cycle-sharing (such as in SETI@home) over these networks.

Keywords: social networks, distributed processing, cycle-sharing, resource discovery

1 Introduction

In the past few years the computing power has been increasing. However, many computational problems requiring enormous amount of computer resources still exist; e.g. applications for scientific research, multimedia video or image encoding.

One of the earliest initiatives identifying idle computing cycles suitable for distributed computing, regards the WORM computing project at Xerox PARC [Sho98], proposed in 1978. Years after, the scientific community realized the potential benefits, mainly in physics, since the supercomputers employed for heavy-duty calculations at each institution, would often be under utilized. Thus,

by enabling the harvesting and allowing the sharing of such idle processing times, grid computing emerged.

Projects such as SETI@Home [ACK+02], Folding@Home,[1] Distributed.net[2] gather a gigantic pool of resources on the Internet, by using desktop computers from any house hold (also known as global distributed computing), allowing them to process data much quicker than in traditional supercomputers.

Nowadays, Social Networks allow individual distributed computing projects to be easily proposed, and promoted across friendship relationships, as well as making more effective the publicity of their motivation and results.

Motivation: The Internet has made it possible to exchange information more rapidly in a global scale. With the creation of Social Networks, anyone in the world can share their experiences and information using only his Internet enabled personal computer or mobile device.[3] Under this scope, there are many Social Networks such as Facebook, Orkut, and others still being actively created (such as Google+), each one exporting its own API to interact with their users' and groups' databases as well as allowing to gather idle resources scattered across the World; examples of such APIs are Facebook API[4] and OpenSocial.[5]

Shortcomings of Current Systems: The public-resource sharing and cycle-sharing systems that are widely used today, are not concerned with the common users' needs. They are mostly used for intensive computational projects (and proprietary) such as Folding@Home, PluraProcessing.

Some systems are beginning to use technologies previously unavailable to other projects, in order to cover more Internet users. However, they use the users' Browsers to do cycle-stealing not addressing the needs of the common users. Moreover, these systems use remote code embedded on Web sites and games (i.e. Adobe Flash based games) to gain access to potential idle resources.

Contribution: The main contribution of this project is the Trans-SocialDP platform with its architecture, messaging protocol and client application. This platform can perform resource and service discovery on top of Social Networks for third-party applications. Furthermore, Trans-SocialDP is able to gather idle cycles from users' computers and communities that are willing and capable of processing tasks, in order to achieve cycle-sharing on Social Networks. It also allows common users to make use of this paradigm to speedup their own (or common) tasks' execution without having to create their own networks.

Trans-Social Networks for Distributed Processing (Trans-SocialDP) is a Web-enabled platform, which was developed and evaluated to interact with Social Networks, and thus being able to mine the Social Networks for users' information which includes their idle resources. It leverages an existing middleware, Ginger [VRF07,SFV10] for task (called Gridlets) creation and aggregation.

Our main concerns on the development of Trans-SocialDP were with the resource discovery and the manner with which a user could submit his own Job,

[1] Folding@Home: `folding.stanford.edu` accessed on 16/02/2012
[2] Distributed.net: `www.distributed.net` accessed on 16/02/2012
[3] Facebook Mobile: `facebook.com/mobile` accessed on 16/02/2012
[4] Facebook Developers: `developers.facebook.com` accessed on 16/02/2012
[5] OpenSocial Web site: `code.google.com/apis/opensocial` accessed on 14/02/2012

processed on others' computers, while also being able to mine any Social Network for user's information.

Document Road-map: The rest of the paper is organized as follows. In the next section, we address the relevant literature related to our work. Section 3 describes the architecture of Trans-SocialDP and its implementation detailed in Section 4. In Section 5, we offer an extensive evaluation of the platform in the context of two (interconnected) Social Networks. Section 6 finishes the paper with some conclusions and lines for future work.

2 Related Work

This section offers a review on relevant works and technologies more related to our work, addressing: i) Social Networks and mining on Social Networks, and ii) peer-to-peer networks, Grids and Distributed Computing.

Social Networks:

Social Networks are popular infrastructures for communication, interaction and information sharing on the Internet. Anyone with a desktop computer and a Browser can access such Web sites, like Facebook, MySpace, Orkut, Hi5, YouTube, LinkedIn and many more.[6] They are used to interact with other people for personal or business purposes, sending messages, posting them on the Web site, receiving links to other Web sites or even sharing files between people, among other uses.

In Social Networks, the basic (real life) behaviors or interaction patterns still apply [Sco88]. By grouping people in the same areas or topics, it is easier to exploit those interactions, because people understand better what the distributed tasks will accomplish and are willing to participate. Social Networks have already began to sprout new ideas to exploit them for uses other than human interactions, such as using it for enhancing Internet search [MGD06] and leveraging infrastructures to enable ad-hoc VPNs [FBJW08].

We focus on Facebook and OpenSocial, because of their size and possibility of access to users' databases by means of the APIs provided. Furthermore, Facebook claims to have reached 845.000.000 users (as of January of 2012) and MySpace (one Web site that uses the OpenSocial API) claiming to have more than 130.000.000 registered users. The potential of these networks for global distributed computing is best compared to other networks.

The Facebook and OpenSocial APIs enable Web applications to interact with the Social Networks servers using a *REST*-like interface[7] or, in case of Facebook, also a *Graph* interface.[8] This means that the calls from third-party applications are made over the Internet by sending HTTP GET and POST requests.

Social Cloud [CCRB10] is described as being a model that integrates Social Networking, cloud computing [AFG+10] and volunteer computing. In this model, users can acquire the resources (the only resource considered is disk

[6] List of Social Networks on Wikipedia.org

[7] Representational State Transfer: `tinyurl.com/6x9ya` accessed on 14/02/2012

[8] OpenGraph Protocol: `ogp.me` accessed on 14/02/2012

space) by exchanging virtual credits, making a virtual economy over the social cloud computing. Users can gather resources from their friends (either by virtual compensation, payment, or with a reciprocal credit model [MBAS06]), allowing this project to approach the objectives for public-resource sharing. Furthermore, there are a number of advantages gained by leveraging Social Networking platforms, such as gaining access to a huge user community, exploiting existent user management functionality, and rely on pre-established trust formed through user relationships. However, the trusting relationship of friends, may not be always the case[9] in Social Networks such as Facebook.

Peer-to-Peer networks, Grids and Distributed Computing: Peer-to-Peer (P2P) networks and Grids are the most common types of sharing support systems. They evolved from different communities to serve different purposes [TTP+07].

Grid systems interconnect clusters of supercomputers and storage systems. Normally they are centralized and hierarchically administrated, each with its own set of rules regarding resource availability. Resources can be dynamic and thus may vary in amount and availability during time, and have to be known beforehand among the network. Grid systems were created by the scientific community to run computation intensive applications that would take too much time in normal desktops (without being distributed) or on a single cluster, e.g. large scale simulations or data analysis.

P2P networks are typically made from house hold desktop computers or common mobile devices, being extremely dynamic in terms of resource types and whose membership can vary in time with more intensity than with Grids. These networks are normally used for sharing files, although there are a number of projects using those kinds of networks for other purposes, such as sharing information and streaming (e.g. Massive Multi-player Online games using P2P [KLXH04] to alleviate server load, distributing tasks as SETI@Home [ACK+02], data streaming for watching TV[10]). The nodes (or peers) are composed by anonymous or unknown users unlike in Grids, which raises its own problems with security or even with forged results [TTP+07].

SETI@Home [ACK+02] aims at using globally distributed resources to analyze radio wave signals that come from outer space, hoping to find radio signals originated from other planets on our galaxy. For this project, having more computing power means it is possible to cover a greater range of frequencies, instead of using supercomputers [ACK+02]. Thus, the authors found a way that lets them use computers around the world to analyze such wave signals.

The wave signals are divided in small units of fixed size and distributed among the clients (that would be located in any user computer operating as a screen saver when there are idle cycles). Then, each client computes the results in its spare time and sends it to the central server asking for more work to do. The most important lesson of SETI@Home project was that to attract and keep

[9] How Facebook could make cloud computing better: `tinyurl.com/237ddem` accessed on 14/02/2012
[10] PPStream: `ppstream.com` accessed on 14/02/2012

users, such projects should explain and justify their goals, research subject and its impact.

BOINC (Berkeley Open Infrastructure for Network Computing) [And04] is a platform for distributed computing through volunteer computers; it emerged from the SETI@Home project and became useful to other projects.[11] Although each project has its own topic and therefore their own computational differences, the BOINC system used for each project (client application) has to be unique.

There are many other projects for distributed computing computing.[12] However, all of them have only one topic of research (for each project), meaning that each system does not have the flexibility of changing its own research topic. With **BOINC Extensions for Community Cycle Sharing** (**nuBOINC** [SVF08]), users without programming expertise may address the frequent difficulties in setting up the required infrastructures for BOINC systems and subsequently gather enough computer cycles for their own project. The nuBOINC extension is a customization of the BOINC system, that allows users to create and submit tasks for distributed computing using available commodity applications. They try to bring global distributed computing to home users, using a public resource sharing approach.

The main concept of **Ginger** (Grid Infrastructure for Non-Grid Environments) [VRF07], from which our proposal is derived, is that any home user may take advantage of idle cycles from other computers, much like SETI@Home, given the right incentives [RRV10]. However, by donating idle cycles to other users to speedup their applications, they would also take advantage of idle cycles from other computers, to speedup the execution for their own applications, with arbitration based on users classes [SFV10], reputation and possibly subject to a virtual currency economic model [Oli11].

To leverage the process of sharing, Ginger introduces a novel application and programming model that is based on the Gridlet concept. Gridlets are work units containing chunks of data and the operations to be performed on that data. Moreover, every Gridlet has an estimated cost (CPU and bandwidth) so that they can try to be fair for every user that executes these Gridlets.

Discussion: While there has been work done to approach volunteer computing using Social Networks, as the communication overlay, they do not either have the same objectives, as public cycle-sharing, or give the users the possibility of using others' resources (idle cycles) for their own work. Comparing these projects based on the communication's latencies, can be volatile and misleading because of the unstable conditions, either by the servers' latencies, network latencies or even the type of computers used.

Furthermore, for our project we assume that the user may not want to give or have idle cycles to spare which would add a small communication time to the overall process.

[11] BOINC projects: `boinc.berkeley.edu/projects.php` accessed on 14/02/2012
[12] List of Distributed Computing projects on Wikipedia.org

3 Architecture

Our work makes use of Social Networks, such as Facebook and MySpace to mine these networks locating users' information and their resources in order to execute tasks (*Gridlets* [VRF07]) on their computers. Note that, this information may include user's profile, such as friends and groups.

Trans-SocialDP uses the SIGAR library[13] to acquire local information about resources (i.e. processor information, memory available). Such information can be sent over the Social Network when requested, while also using it to decide whether it should accept a new Job (from someone else).

The main approach for Trans-SocialDP is to have the platform split into two parts: one that interacts with the Social Networks; and another to interact with the users, the local resources and the Ginger Middleware (which is out of the scope of this work [VRF07]).

Design Requirements: The client application interacts with the Social Networks (Facebook, MySpace) through Web Protocols named Graph and REST (which are an added layer to the HTTP protocol). As Social Networks are still developing their own systems, the operations available within the client application may change over time. We use libraries such as *RestFB* library,[14] *myspace-id* library[15] and *OpenSocial-java* library[16] to ease the communication to and from the Social Networks.

Moreover, in order for Trans-SocialDP not to interfere with the users' normal usage of their computers, the client application can schedule processing to another time, while also preventing its overuse by stopping its activities, i.e. the processing of requests and Gridlets only happens when there are idle cycles to spare.

Trans-SocialDP Architecture: The Trans-SocialDP architecture relies on an interaction with the Social Networks through the Social Network's API (Graph or REST protocols) for the purpose of searching and successfully executing Jobs; with the Ginger Middleware for Gridlet creation and aggregation; and also the user's operating system to acquire the information and hardware states that are needed.

Jobs are considered to be tasks initiated by the users, and containing more than one Gridlet to be processed in someone else's computer; all Jobs state what they require in order to execute those Gridlets, so that the client application can search for users or groups (computer information) that could help on their processing.

A Gridlet contains the information necessary to process it, meaning that it has the data file(s) to be transferred to another user and the arguments to be given to the executable program. The process of creating and aggregating the Gridlets is managed by the Ginger Middleware and is outside the scope of this work [VRF07].

[13] SIGAR library: `hyperic.com/products/sigar` accessed on 14/02/2012
[14] RestFb Web site: `restfb.com` accessed on 14/02/2012
[15] MySpaceID: `developer.myspace.com/MySpaceID` access on 14/02/2012
[16] OpenSocial Java: `code.google.com/p/opensocial-java-client` access on 14/02/2012

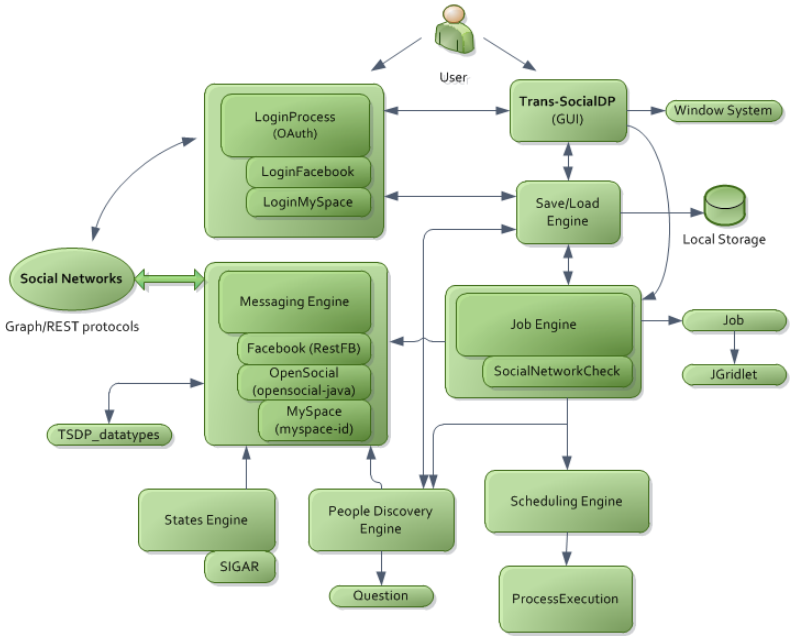

Fig. 1. Trans-Social Networks for Distributed Processing module view.

The architecture for Trans-SocialDP is comprised of a set of modules depicted in Fig. 1, and described as follows.

Trans-SocialDP (GUI): this is the main module, which contains the graphical user interface for the user to interact with the client application. It is responsible to initiate the interaction with the Social Networks and local computer.

Login Process: this module includes the OAuth[17] call definition for each Social Network. It provides to the user the Log-in web site for each Social Network in order for the client application to interact with each one; it uses an embedded web browser (using the JDIC library) for that effect. Note that each Social Network is responsible for implementing its own OAuth system (or other authentication systems); for such the client application needs to implement submodules with each specification for each connection (LoginFacebook, LoginMySpace).

Messaging Engine: this is a main module to the client application; it is required for communication with all the Social Networks, containing all the necessary functions to do so. The communication between the client application and the Social Networks chosen are made by each submodule, according to the used protocol of the Social Network. In particular, for the Facebook submodule it uses the Graph protocol to send/retrieve Posts/Comments to/from this Social Network.

[17] OAuth Web site: `oauth.net` access on 27/02/2012

MySpace submodule sends/receives key-value pairs to/from a global map used by third-party applications, which is provided by MySpace servers. Furthermore, MySpace is being constructed as if it was an OpenSocial network, thus we make use of the OpenSocial-java library to allow for OpenSocial communication protocol.

The Messaging Engine also contains its own data types, which are converted to the message schemas applied to the messages sent/retrieved for each Social Network, because they may block some types of messages.

Job Engine: this module is divided in two parts, where the first is responsible to start the chain of events for processing a Job (submitted by the user), such as sending search messages, sending Gridlets to other users.

The second part (SocialNetwork Check) is responsible for searching for new Jobs in each Social Network, in order to answer them according to the Job's requirements and the local resource's availability. In the specific case of Facebook the redirected messages (FoF method) are also taken into consideration, when starting or searching for Jobs. Thus, each Messaging Engine submodules need to be aware of the communication protocol previously established for each Social Network, so that the Job Engine only requires to know a generic way of send/retrieving information for any Social Network.

People Discovery Engine: this module mines the Social Networks to retrieve users, friends, groups and other types of information from them. In addition, it is also used to communicate with other client applications in order to establish relations between different users' accounts on different Social Networks (users that are connected to several Social Networks).

As an implementation issue, the client application needs to distribute the tasks (Gridlets) as fair as possible. Thus, the module uses the users' information to assess if the incoming messages (sent by users that accepted the Job), come from different users. In order to send a Gridlet for each user, even if they accepted the Job on more than one Social Network (may not happen when the number of users is less than the number of Gridlets).

Furthermore, this module contains a data structure for the questions that are sent to other client applications, i.e. in order to ask other users for a specific person from a Social Network.

Scheduling Engine: this module gives priorities to the Gridlet's processing, according to a specified criteria (e.g. users' friends tasks may have higher priorities than other users).

States Engine: this module determines the state of the user's computer, taking in consideration the processor's idle times, Internet connectivity (essential to all engines), the user's Social Networks states and the local state (i.e. when the client application has been halted by the user). It also uses the SIGAR library, which reports the system information needed to determine the availability of the resources.

Prototypical example: The prototypical example as depicted in Fig. 2 gives an idea of the platform's communication flow, from the creation of a new Job (by the Starter user), to using the Messaging Engine to send and retrieve

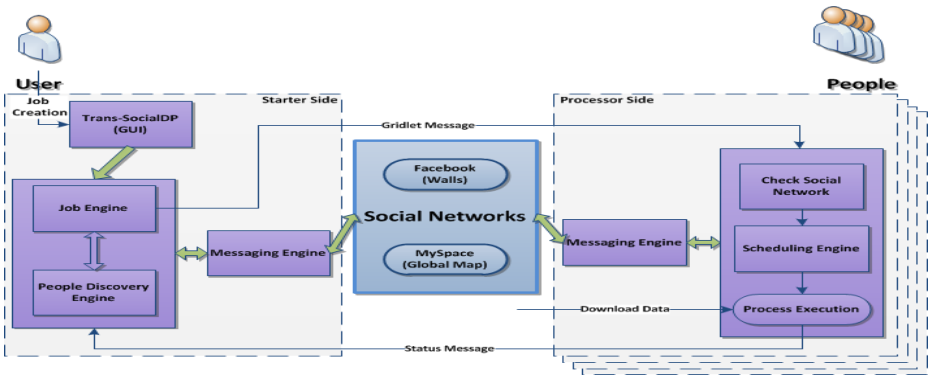

Fig. 2. Trans-SocialDP Prototypical example

messages from the Social Networks the application is connected to. In this example, the Gridlet message is sent by the Starter's Messaging Engine to a Social Network (Facebook), retrieved by the Messaging Engine on the Processor side (an application that accepted the Job), scheduling it to be processed when it has idle cycles to spare and sending the results (status) back to the Starter in the same manner.

4 Implemention Details

The implementation of Trans-SocialDP aims for a simple use by the end-users. Also, the different types of operating systems lead us to favor portability; therefore, we used Java as the main language. We primarly chose Facebook over other alternatives, because it has a higher number of registered users than any other Social Network, and MySpace because it is well-known by Internet users, while also utilizing some of the OpenSocial concepts.

This section gives an insight on how the technologies were used, such as Graph and REST protocols. It also explains the schemas used for the messages sent/received to/from the Social Networks chosen.

Technology Employed: For the purpose of interacting with the Graph and REST servers, the client application makes use of the RestFb library for Facebook, the OpenSocial-java and MySpaceID libraries for MySpace, which gives a simple and flexible way of connecting to them and conceal the use of XML or JSON objects.[18] However, the functions (using REST) or connections (using Graph) have to be known, in order to use these libraries, e.g. to read the Posts on a user's Wall on Facebook using the Graph protocol, we need a user's ID or Name for the library to access Facebook and retrieve that user's Wall.

As Trans-SocialDP also needs to gather the information about the local resources of the users' computers, we make use of the *SIGAR* library. This allows

[18] JSON: `json.org` accessed on 14/02/2012

us to easily access a list of local resources each time it is called, such as processors, cores, memory. Also, it gives us the ability to know the current states of those resources, i.e. it can give us the available memory at the requesting time, or even the current idle time for each of the available cores. This library is also useful for the fact that it can work in multiple environments, such as Windows, Linux, among others, making it possible the portability of Trans-SocialDP to other systems.

Message Schemas: Trans-SocialDP uses Social Networks to send and retrieve messages via their external interfaces. In Facebook, it reads Posts (messages that are contained in the users' Walls, groups' Walls) and Comments (messages contained within the Posts), and writes other messages on users' Walls (which is a space that contains messages) either as Posts or Comments.

As for MySpace, Trans-SocialDP uses a global map of key-value pairs to send and retrieve messages to and from other applications. These keys, are generated by the client application depending on the type of message plus a random number in order for the keys to be different (e.g. for the Job search request message the key becomes transSocialDP.JobSearch.Random Number). The values for each key contains the message to be sent to another user, which is the same as we use on Facebook.

These Schemas are very simple and human readable, in order for Facebook (or other Social Networks) to allow them on the Web site, and not consider them as Spam or other type of blocked messages. They are also human readable to assure the users what information is being sent to other users.

5 Evaluation

The evaluation of Trans-SocialDP is comprised of a scenario that assesses each Social Network and ultimately the combination of interacting with several Social Networks, in order to know the effects each can carry to the processing times, while also evaluating our works' goals.

Scenario Mixed: This scenario was designed to evaluate the performance of Trans-SocialDP on both Social Networks (Facebook, Myspace) with a complex system of connections. As depicted in Fig. 3 the starter has a connection to Facebook (FB) and to MySpace (MS) where it can send/retrieve messages from other client applications, adding that some users may have accounts on both Social Networks and are identified by their UIDs of each Social Network. The starter needs to know if a user is the same on both networks, even though the UIDs are different. This network is composed by 2 Friends in Facebook, each one having a Friend (FoF), a group (Facebook) with 4 users including the starter, and 6 users connected to MySpace.

The number of *Gridlets* to be processed are 8 in total, and the processing time is 5 minutes each, in order for all users in the network to take at least one Gridlet. The first three tests were made without a local database of users' information, and the later three with it.

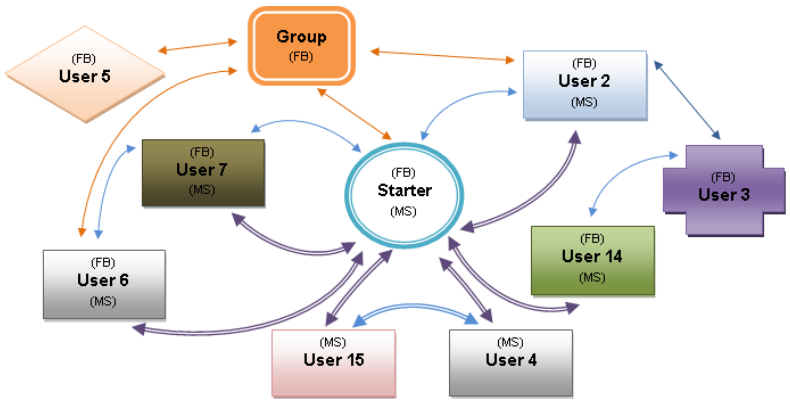

Fig. 3. Trans-SocialDP Scenario Mixed View

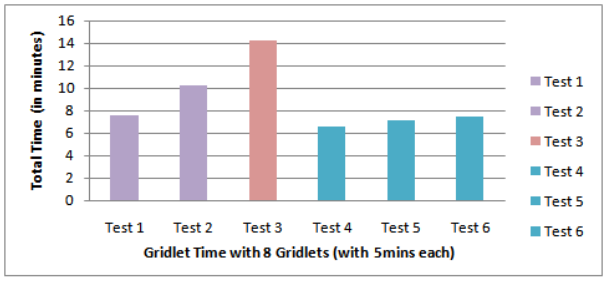

Fig. 4. Rendering Test Times for Scenario Mixed

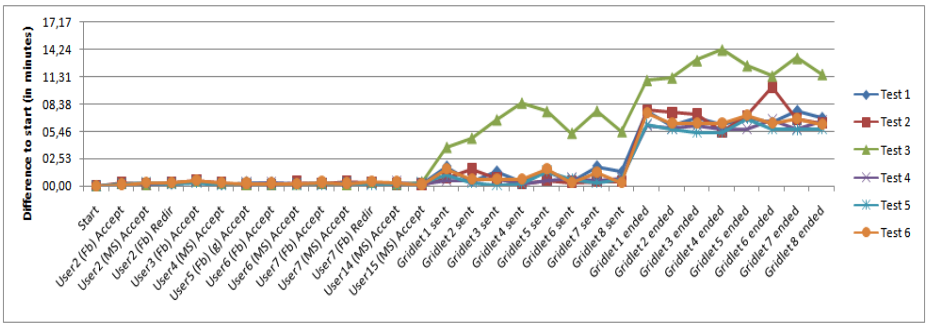

Fig. 5. Communication Times for Scenario Mixed

For this scenario we assume that i) all client applications are running, and connected to their respective Social Networks; ii) that the client applications can retrieve a *Gridlet* message from any place that they have accepted, e.g. User 7 can accept a Job from Facebook (Friend connection) or MySpace and that User

14 can only accept a Job from MySpace, although it has a connection to User 3 (on Facebook), it goes beyond the 2nd degree of friendship we imposed.

The results for scenario Mixed, as depicted in Fig. 4, describes the total time for a Job for each test. Some situations, as in test 3, are caused by latency related problems with the communication between client applications. Also, for test 2 the creation of the local database might not have been completely accurate, thus the starter sent more than one *Gridlet* to a user (the same user on different Social Networks).

This case can happen when the starter asks for information about a user to the network (or friends/other users) and does not get an answer within a pre-defined time period; it then assumes that the user may not be the same in both Social Networks, where in fact they were. However, besides all the latency caused by the communication, the total process times in all tests still gain speedups against local execution (where it should have been about 40 minutes).

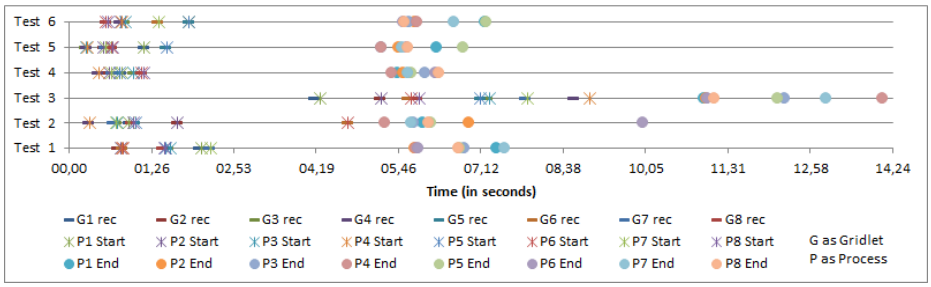

Fig. 6. Wait times for Gridlet process on Scenario Mixed

In Fig. 5 we can see the time each task takes to be completed, and the spikes the connection to the Social Network can cause on the overall process; e.g. the third test took much more time to send the *Gridlets* than in the other tests, because of the added times for mining information about other users and some latency in the servers.

In Fig. 6 we can notice some delay to receive a *Gridlet* message in test 3, which can occur when the network communication is having more latency than usual. We also see in test 2 that a *Gridlet* message was received and its processing was delayed in relation to the other *Gridlets*, when a user received it from another Social Network.

Discussion: We can state that the overhead that Trans-SocialDP imposes on the overall process is minimal compared to the time it takes to process a *Gridlet*, which in realistic terms can be more than 1 hour. However, these times can be hindered by other tasks such as searching for resources that do not return positive results, or even that the total resources available are lesser than the number of *Gridlets* that have to be processed.

Furthermore, we can state that the search for users' information may hinder the total process, because the information on both Social Networks may not be compatible, and thus situations like sending more than one *Gridlet* for the same user via different Social Networks can occur.

When comparing with local execution, Trans-SocialDP decreased the total processing time, compared to what it would have consumed in the users' computers. Trans-SocialDP achieves overall speedups on Jobs, and the added support of other Social Networks may benefit this speedup. By finding capable users, that either the search did not reached on one of the Social Networks, or that the users are only connected to the other Social Network.

Moreover, we can confirm that the users can donate their resources (processors' time) for other users' consumption. While also, taking advantage of other users' resources, that have the same interests (or in the same groups), to further speedup their own tasks.

6 Conclusion

Our project exceeds the boundaries of a common use of Social Networks, allowing any user to consider donating their idle processing cycles to others (while also making use of others) that may need to finish their tasks faster than they would in their own computers. In this project we presented a new method of resource and service discovery through the use of Social Networks and the interaction between users.

The main approach for Trans-SocialDP is to have a client application split in two parts. One that interacts with the Social Network using REST or Graph protocols; and another to interact with the users' computers for local resource discovery, and the Ginger Middleware for creation and aggregation of Gridlets.

We evaluated Trans-SocialDP with several scenarios to determine the viability and the changes it would take by contacting different Social Networks. Thus, we created a scenario that interacts with both Social Networks to regard Trans-SocialDP performance, stability and viability on such networks.

In addition, the scenario described was created with several users' roles in mind, meaning the users could be regarded as Friends, Friends of Friends, group members, or other roles, in order to fully test the communication system between the users' client applications.

With the obtained results, we can conclude that while the total times for processing a Job gained speedups against local execution in the users' computers, this can be hindered by some variables: latency of Social Network servers, the fact that searching for resources among Social Networks users may not return positive results, that the total number of available resources is less than the number of Gridlets that comprises a Job, while also the distribution of tasks among the available users may not completely parallelize them.

In summary, we find all our goals to have been met, in the sense that our proposed platform can leverage Social Networks, by mining users' friendships,

relationships, and affiliation to groups and communities, in order to gather computational resources. Such resources can be employed to speedup jobs in a global distributed computing platform. By bringing in the concept of resource sharing to global infrastructures such as Social Networks, we allow virtually any common user to make use of idle resources scattered across the Internet, within a framework they are familiar with. Within, Trans-SocialDP, global cycle sharing can become widely employed as many other features supported by Social Networks.

Future work: For the future, we believe that Jobs completion and the search for resources would benefit with requirements' semantics, increasing the chance to direct Gridlets to peoples' computers that would satisfy those. In addition, a reputation method would assure users that the distributed tasks are given to those that can actually undertake the responsability of processing the tasks.

Acknowlegments This work was partially funded by FCT projects PTDC/EIA-EIA/102250/2008, PTDC/EIA-EIA/113993/2009 and by PIDDAC Program funds (INESC- ID multiannual funding).

References

[ACK+02] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[AFG+10] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[And04] D.P. Anderson. BOINC: A system for public-resource computing and storage. In *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10. IEEE Computer Society, 2004.

[CCRB10] K. Chard, S. Caton, O. Rana, and K. Bubendorfer. Social Cloud: Cloud Computing in Social Networks. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 99–106. IEEE, 2010.

[FBJW08] R. J. Figueiredo, P. Boykin, P. Juste, and D. Wolinsky. Integrating overlay and social networks for seamless p2p networking. In *WETICE*, pages 93–98, 2008.

[KLXH04] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *IEEE INFOCOM*, volume 1, pages 96–107. Citeseer, 2004.

[MBAS06] M. Mowbray, F. Brasileiro, N. Andrade, and J. Santana. A reciprocation-based economy for multiple services in peer-to-peer grids. In *Peer-to-Peer Computing, 2006. P2P 2006. 6th IEEE International Conference on*, pages 193–202. IEEE, 2006.

[MGD06] A. Mislove, K.P. Gummadi, and P. Druschel. Exploiting social networks for internet search. *BURNING*, page 79, 2006.

[Oli11] Oliveira, P. G. and Ferreira, P. and Veiga, L. Gridlet Economics: Resource Management Models and Policies for Cycle-Sharing Systems. In *International Conference on Grid and Pervasive Computing (GPC 2011), Lecture Notes in Computer Science (LNCS)*. Springer, May 2011.

[RRV10] P.D. Rodrigues, C. Ribeiro, and L. Veiga. Incentive mechanisms in peer-to-peer networks. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE, 2010.

[Sco88] J. Scott. Social network analysis. *Sociology*, 22(1):109, 1988.

[SFV10] JN Silva, P. Ferreira, and L. Veiga. Service and resource discovery in cycle-sharing environments with a utility algebra. In *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–11. IEEE, 2010.

[Sho98] S. Shostak. *Sharing the universe- Perspectives on extraterrestrial life*. Berkeley Hills Books, 1998.

[SVF08] J. Silva, L. Veiga, and P. Ferreira. nuboinc: Boinc extensions for community cycle sharing. In *SASO Workshops*, pages 248–253, 2008.

[TTP+07] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. Peer-to-Peer resource discovery in Grids: Models and systems. *Future Generation Computer Systems*, 23(7):864–878, 2007.

[VRF07] L. Veiga, R. Rodrigues, and P. Ferreira. GiGi: An Ocean of Gridlets on a "Grid-for-the-Masses". In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, pages 783–788. IEEE Computer Society, 2007.