Social Networks for Cycle-Sharing

Nuno Apolónia
INESC-ID / IST
Rua Alves Redol No 9, 1000-029 Lisboa, Portugal
nuno.apolonia@ist.utl.pt
Supervisor: Prof. Luís Veiga Co-supervisor: Prof. Paulo Ferreira

***Abstract*—The growth of the Internet, and consequently the number of interconnected computers is the basis for the global distributed computing and public-resource sharing. Meaning that, these resources have been used for computation intensive projects that could not be completed in a short time frame, sometimes not even in supercomputers which are not widely available.**

Furthermore, the Internet is overwhelmed by social connectivity. Internet users make use of Social networks to interact and share information, knowledge and services with each other.

This paper presents an overview of Peer-to-Peer networks and Grids, to understand their advantages and problems. So that, we can grasp the fundamental ideas that sprout the global distributed computing and the problem of locating resources and services efficiently.

We also analyze Social networks and social interactions to understand how they can be explored for other uses rather than what they were initially created for.

In the last sections we explain the development and resulting evaluation of a web-enabled platform, called Social Networks for Cycle-Sharing (SNCS), that uses Social networks as a starting point for resource and service discovery and integrating it with the Ginger Middleware for distributed computing of tasks.

Also, to conclude that using a Social network for public-resource sharing can give common users the possibility of releasing resources for other programs usage.

***Index Terms*—Social networks, resource discovery, distributed systems, Peer-to-Peer, Grids, public-resource sharing, global distributed computing.**

I. Introduction

The computing power has significantly increased in the past few years (more or less like Moore's Law [18]), but there are still many computational problems that need an enormous amount of computing resources, e.g. applications for scientific research, financial risk analysis or multimedia video or image rendering and encoding. These resources are composed by computing elements like CPU, memory or data storage and all of them can be found on every house hold or in offices and even in our daily devices, such as notebooks or mobile phones.

With the Internet, the available resources for projects such as Seti@Home [4], Folding@Home[1], Distributed.net[2] were extended, by gathering the gigantic potential of using millions of desktop computers from as many house holds as possible (also known as global distributed computing), allowing them to process their data much quicker than in traditional super-computers.

The Internet also enabled information and content sharing by using Peer-to-Peer (P2P) networks. Which can be categorized in terms of their topology as being structured or unstructured. Unstructured systems are characterized by having a underlying topology unrelated with the placement of the contents, as opposed to Structured systems where it is attempted to place the contents in specific locations. Furthermore, Hybrid systems are optimizations to leverage the performance for locating contents and networks' scalability (in terms of traffic load). They highlight two types of users, the users that have more bandwidth are called super-peers and those with low bandwidths are called peers and the last ones are connected to the super-peers [21].

These networks have some challenges, such as efficient resource discovery. That is, when a peer needs a resource it will have to ask other peers for it. Some approaches try to minimize the message traffic that can be generated, either by contacting fewer peers (when information is spread to others) or by creating central nodes that have all or partial information for locating the exact content.

Moreover, the Internet has made it possible to exchange information more rapidly on a global scale. One of the natural steps was the creation of Social networks where anyone in the world can share their experiences and information using only their Internet enabled personal computer or mobile device[3]. Under this scope there are many Social networks such as Facebook[4], Orkut[5] or Youtube[6] each one exporting their own *APIs* to interact with their users and groups databases, e.g. Facebook API[7] and OpenSocial.[8] Moreover, these networks have great potential for financial benefits, such as Advertising.

Studies done on these networks show that they follow some properties like the Small-World property, meaning that there is a small group of users with high connectivity to others and a much larger group with low connectivity. Besides that, even the highly connected users only interact (on a daily

This paper is an extended version of N. Apolónia, P. Ferreira and L. Veiga. Social Networks for Cycle-Sharing. Proceedings of 10th Conferência sobre Redes de Computadores, Braga, Portugal, 11 e 12 de Novembro de 2010.

[1]Folding@Home Website: http://folding.stanford.edu on 05/01/2010
[2]Distributed.net Web site: http://www.distributed.net on 05/01/2010
[3]Facebook Mobile: http://www.facebook.com/mobile on 19/08/2010
[4]Facebook Website: http://www.facebook.com on 05/01/2010
[5]Orkut Website: http://www.orkut.com on 05/01/2010
[6]Youtube Website: http://www.youtube.com on 05/01/2010
[7]Facebook Developers:http://developers.facebook.com on 05/01/2010
[8]OpenSocial: http://code.google.com/apis/opensocial on 05/01/2010

basis) with a restrict group of users [24]. Considering that these networks could be regarded as enabling peer-to-peer information sharing (albeit mediated by a centrally controlled infrastructure), employing them for cycle-sharing should be a great improvement for global distributed computing, by allowing public-resource sharing among trusted users and within communities.

The Ginger project [23] serves as a middleware for deploying distributed processing using a P2P network for work dissemination (*Gridlets*) within its peers. The main idea behind the Ginger project is that any user may need processing time for common applications to be executed.

This paper explains the development and resulting evaluation of a web-enabled platform, called Social Networks for Cycle-Sharing (SNCS), that interacts with a Social network (Facebook), for resource and service discovery and *Gridlet* dissemination, while using the Ginger Middleware for *Gridlet* creation and aggregation.

The rest of this paper is organized as follows. In the next section we will present some relevant related work. Section 3 is dedicated to present the architecture of the SNCS client. In Section 4, we will address some of the implementation issues. Section 5 presents the results obtained in the evaluation, and finish the paper with conclusions and future work.

II. Related Work

A. *Peer-to-Peer networks and Grids*

Peer-to-Peer (P2P) networks and Grids are the most common types of sharing systems, yet they evolved from different communities to serve different purposes [21].

Grid systems interconnect clusters of (super)computers and storage systems. Also, they can be dynamic and may vary in time. Grids 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 also vary in time with more volatility than in 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. distributing tasks as Seti@Home [4], data streaming for watching TV).[9]

These two distributed systems have different resources, which may indicate a different level of computing power of the nodes comprising each one. However, it is easier to leverage more desktop computers than to have large supercomputers at our disposal. This can make P2P systems aggregate more computing power than the Grid systems.

Resource discovery in P2P networks: The term resource is used to include hardware, software, licenses, Grid services, and others alike [10].

Distributed computing raises the problem of finding resources for given tasks, and P2P file sharing systems have always been dealing with such problems [13]. Since there is a lack of a central administration in P2P networks, the search for files may have to include all its peers and has to be redone every time any node requests a resource. This happens because those resources might be different among peers and may not be available indefinitely or always in the same location.

Flooding [16] is one of the earliest techniques used regarding resource discovery and to overcome problems of excessive traffic, alternatives include other *blind* methods such as *Random walks* and *multiple random walks* [22], hybrid methods that combine flooding with random walks such as *direct searches* [11] and *forwarding indices* [8].

With Structured P2P systems, the attempt to always control where the contents should be, lead to explore the alternative of using DHTs [12]. However, this approach can only operate when the resources are well known. Some other approaches to resource discovery have contemplated the solutions of integrating the idea of using a Social network within a Grid [9], to better guide the queries to the right resources.

B. *Distributed Computing Projects*

The subject of distributed computing has been previously addressed by several projects. And the first relevant were distributed.net[10] and GIMPS.[11]

Distributed.net uses computers from all around the world to do brute-force decryption of RSA keys, and attempt to solve other large scale problems.

The GIMPS project uses the same concept of distributed computing to search for *Mersenne* prime numbers, these numbers are of the form $2^P - 1$ where P is a prime. Both projects use their own Client and Server applications, following the same idea as the BOINC projects [3].

There are many other projects for distributed computing.[12] Although all of them have only one research topic (for each project), meaning that each system does not have the flexibility of changing its own topic of research. This has been addressed in nuBoinc [20].

SETI@Home System: Some work in the area of global distributed computing (the use of home and office computers for distributed computing) has been already addressed. As we can see in projects like SETI@Home [4], where they use these kind of resources to analyze radio wave signals that come from outer space. For this project, having more computing power means they could cover a greater range of frequencies to process.

The wave signals were divided in small units of fixed sized, to be able to distribute among the BOINC clients (that would be located in all the users' computers), then the client would compute the results in their idle time and send it to the central server asking for more work to do. In this system the clients

[9]PPStream: http://www.ppstream.com/ on 05/01/2010

[10]Distributed.Net: http://www.distributed.net on 05/01/2010

[11]GIMPS: http://www.mersenne.org on 05/01/2010

[12]List of Distributed Computing Projects: http://en.wikipedia.org/wiki/List_of_distributed_computing_projects on 05/01/2010

would only need to be able to communicate with the server when they finished the computations (or for asking more data).

Moreover, users had a ranking system to compete against other users, to motivate them to use this system. Thus, adding that the most important lesson of SETI@Home project was that to attract and keep users, such projects should explain and justify their goals, research subject and its impact.

Grid Infrastructure for Non-Grid Environments (Ginger) middleware: The main concept of the Ginger project [17], [19], [23] is that any home user may take advantage of idle cycles from other computers, much like SETI@Home. Donating idle cycles to other users to speedup other users' applications and by doing so, they would also take advantage of idle cycles from other computers, to speedup the execution for their own applications. 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*. By these means, the resources globally would always be occupied taking advantage of all idle resources, and giving home users the opportunity of executing their own tasks with acceptable performance.

C. *Analysis on Social networks*

Studies of Social networks such as [1], [14], [24] focus their attention into how users and groups interact with each other in the course of time and to quantify it so we can learn how these networks evolve in time.

These studies have reinforced the idea that those networks follow a power-law graph and that there are more users with few links than users with many (Small-world property). A user having many links (to other users), which can be in the thousands, does not mean that he/she will interact with everyone most of the time, these interactions are confined to a small group of users from all of those that the user is linked to. It is also assumed that users tend to have more links to others, rather than the ones they frequently interact with.

Small-world networks can be categorized by the possibility of connecting any two vertices in the network through just a few links [2]. Furthermore, growing networks can be hindered by two factors: *Aging of the vertices* and *Cost of adding links to the vertices or the limited capacity of a vertex*.

Many Social networks also have ways of connecting users, without being linked as friends, these connections are called groups, where knowledge is exchanged within a specific topic of interest. The creation for such groups and their evolution over time is inherent by people's tendencies of coming together to share knowledge of a particular theme [6].

Facebook and OpenSocial: There are many Social networks in the Internet.[13] The focus on Facebook and OpenSocial based networks is explained by having access to the databases, by means of the APIs they export. Moreover, Facebook claims to have 500.000.000 (as of July 21 of 2010) users and MySpace claiming to have more than 130.000.000 registered users. Which makes them well known within the common users. Also, the potential of these networks for global distributed computing is untapped compared to other networks.

Furthermore, the Facebook API[14] and OpenSocial API[15] enables Web applications to interact with the server using a REST-like interface[16] or in case of Facebook a Graph interface.[17] This means that the calls from the applications are made over the Internet by sending HTTP GET and POST requests and using XML or JSON messages.

An example of a Facebook application is Progress Thru Processors,[18] where the application shares the users' contributions to a BOINC system, with their friends through Facebook.

Social Cloud: Cloud computing [5] derives from resource-sharing environments, and work with the intent of bringing those environments to Internet users. Also, it was created a relation between the resources given and received, meaning that in order to acquire resources a user can buy, sell or exchange them in "marketplaces", which provide lists of resources to be used (according to a virtual transaction) by any user.

Social Cloud [7] is introduced as being a model that integrates social networking, cloud computing and "volunteer computing".

They also refer that it is a scalable computing model, where users' resources are dynamically provisioned amongst a group of friends. Also, adding that the model is similar to a Volunteer computing approach, where friends share resources amongst each other for little to no gain.

Their idea is that users can gather resources from their friends (either by virtual compensation, payment, or with a reciprocal credit model [15]), which makes this model approaching the public-resource sharing objectives.

Furthermore, they state that there are a number of advantages gained by leveraging Social networking platforms, such as gaining access to a huge user community, and rely on pre-established trust formed through user relationships. However, the trusting relationship of friends, may not be always the case in Social networks such as Facebook.

III. Architecture

This work proposes to use Facebook, to be able to locate resources for the execution of Jobs (which are composed of *Gridlets*) submitted by the users. Also, to discover computers' informations and users' profiles, such as the groups which they belong to and their friends.

[13]List of Social Networks: http://en.wikipedia.org/wiki/List_of_social_networking_websites on 05/01/2010

[14]Facebook dev. Wiki: http://wiki.developers.facebook.com/index.php/Main_Page on 05/01/2010

[15]OpenSocial Specs: http://www.opensocial.org/specs on 05/01/2010

[16]ReST: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm on 05/01/2010

[17]OpenGraph Protocol: http://opengraphprotocol.org on 23/08/2010

[18]Progress Thru Processors: http://www.facebook.com/progressthruprocessors on 05/01/2010

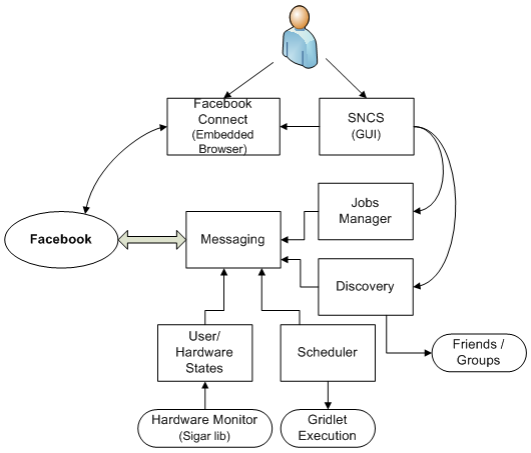

Fig. 1. SNCS architectural view

The SNCS platform is able to interact with the Social network server, meaning that it intercepts/sends messages from/to other users or groups, while also discovering users' computer profiles by contacting the Graph server.[19] It also gives the user the ability to initiate a Job, by using the client application' user interface.

To actually locate resources through the Social network, SNCS has the ability of searching the local resources, such as processors status, memory available, number of processors, processors frequency. And such information is sent to other users upon request, or it can also be sent to the users' Wall, in order for everyone (that has the ability to see the Wall) to retrieve it. Also, this information may contain the programs that can be executed by the computer to process the *Gridlets*, it is a configuration parameter that the user can deal with.

SNCS advertises users' availability to others, sending messages and scheduling tasks (i.e. search for informations, *Gridlet* acceptance) on other users (Friends, Friends of Friends, Groups) in order to execute the tasks when users can spare their idle cycles.

SNCS starts listening for requests that can appear on the users' Wall, friends' Wall or Registration post on the Applications' Wall. As Facebook does not allow people to interact with each other without being friends, the latter option was added to circumvent this inability, making it possible to gather resources from people outside the friends' domain.

SNCS Architecture: The SNCS architecture (depicted in Fig.1) relies on a number of SNCS components running locally in each user's machine that interacts with the Social network through its API (Graph or REST protocols) for the purpose of searching and successfully executing Jobs; with the Ginger Middleware for *Gridlet* creation; and also the user's operating system to acquire the informations and hardware states, that are needed.

Jobs are considered to be tasks initiated by the users, and containing *Gridlets* to be processed in someone else's computer, all Jobs should state what they require to execute them, in order for SNCS to discover specific users or groups.

The *Gridlets* should contain the data file(s) to be transfered to another user and the arguments to be given to the executable program. The process of creating and reassembling the *Gridlets* is managed by Ginger Middleware and is outside the scope of this work [23].

The architecture is comprised of modules, depicted in Fig. 1. Each module has its own function as follows.

SNCS (GUI) is the main module to interact with the users, containing the graphic interface. It is responsible for establishing the connection to Facebook, by starting the Facebook Connect module. It also loads all the necessary information onto SNCS, such as the configuration of priorities. Also, the user can start a new Job submission by using the interface presented.

Facebook Connect (Embedded browser) is the module that serves to authenticate the user to Facebook, it displays the web page given by Facebook for that purpose, by means of the *JDIC* library.[20] Afterwards, it extracts the necessary *access token* for consequent access to the Facebook server. This token is given by Facebook to everyone that accepts this Facebook application, and has to be renewed within a determined time frame.

Messaging is the main module for interacting with the Social network. It makes use of the *RestFb* library,[21] that creates the JSON[22] or XML objects, which are required to access Facebook Graph/REST functions. This module also contains the options necessary to read and write to the users/groups/Application Wall Posts or Comments and removing them as well. Furthermore, some Facebook restrictions may apply to it, such as limiting the size of the messages. This module also contains the Schemas applied to the messages sent and retrieved, to specify what actions should be taken.

Jobs Manager is the module that runs a continuous "checking" cycle, which verifies submitted Jobs that the user has in progress; checks for new Jobs from the users' Wall, groups' Wall or Registration Post that can be processed on the users' machine; verifies *Gridlets* that have been sent to the user after accepting a Job; checks for submitted Jobs or *Gridlet* completion; checks for messages that SNCS needs to redirect to its friends; checks for messages that have been redirected to the user, so that it can answer them on the Registration Post. Moreover, this module hands the acquired *Gridlets* over to the scheduler module for later execution.

Discovery this module serves as an addition to the previous. Meaning that it searches for friends and groups, in order to reach as many people as possible, to complete a Job. It sends messages to friends in order for them to redirect those to their own friends (FoF method), while also sending messages to groups of interest for that specific Job. It is also responsible for registering the user in the Applications' Wall.

[19]Facebook API: http://developers.facebook.com/docs/api on 25/08/2010

[20]JDIC: https://jdic.dev.java.net/ on 13/09/2010

[21]RestFb: http://restfb.com on 13/09/2010

[22]JSON: http://www.json.org on 13/09/2010

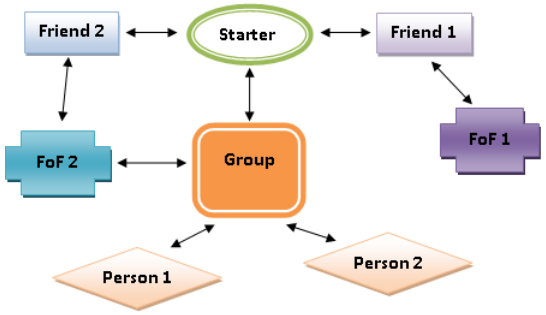

Fig. 2. Scenario 6 View

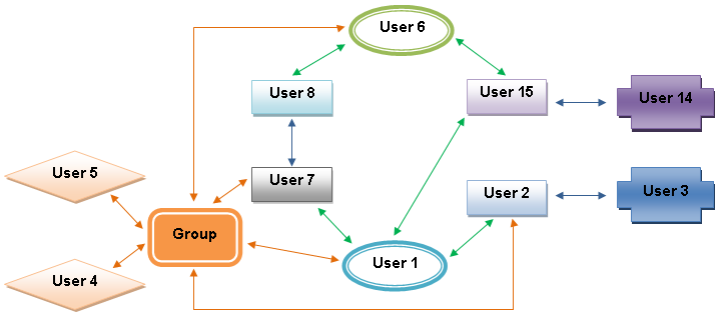

Fig. 3. Scenario 7 View

User/HW States this module takes in consideration the processors' idle times, the Internet connectivity and the users Facebook state, to yield all the modules until a later time, when the processor has idle cycles to spare. Also, it sends the state of the user's machine (*Online*, *Offline*, *Idle*) to the Social network, which can be retrieved by other users. This module uses a submodule, that is comprised of the *SIGAR* library,[23] that reports the system information needed to determine the availability of the resources.

Scheduler this module is an addition to the *Gridlet* processing, making use of the priority lists, while also stopping the process when the computer does not have idle cycles. The priority lists consists of friends and other people added by the user, in order for SNCS to use the idle cycles on *Gridlets* belonging to the people with the highest priorities. The module starts a submodule that is responsible for processing the *Gridlet*, it transfers the necessary data file(s) between the client applications, and upon completion it informs the originator the *Gridlet* state.

Discovery Mechanism: The discovery mechanism searches for people that have the capability to do the work, by either retrieving their computers' information or by requesting it, via messaging on top of Facebook. Furthermore, it verifies if there are any users capable of accepting a Job on the users' Group list in the same manner. To cover as much work as possible, SNCS also searches among its friends' Walls if they have any Jobs that can be fulfilled. The information gathered contains a list of resources and properties of the users' computers, such as number of processors, processors clock speed, total memory, list of user defined programs.

SNCS attempts to match the needs of a Job to the information gathered, although it is out of the scope of this work, the matching of the information should not constrain the execution or acceptance of a Job, meaning that a semantics should be taken into account in order to approximately match these properties to what is required [19].

User Interface: SNCS was designed to provide a simple Graphical User Interface (GUI), in order for any user to utilize it without much burden. In order for SNCS to function correctly, a user needs to have an account on Facebook, and *Log in* into it via the client application. While on the background it gets the *access token* needed for future communications. SNCS is also able to configure some of the aspects needed to better suite the users, such as prioritize the incoming *Gridlets*.

After the *Log in* process, users have access to the main interface, where options like creating a new Job, sending their computers' informations to the Social network, or even making the client application *Offline* are available. For example, the new Job interface allows a user to easily start a Job, by inserting all the required information in the fields presented.

IV. Implementation Details

SNCS was implemented in Java for its portability purposes, it uses Facebook as its Social network for interactions between users' client applications. This Social network was chosen because it provides access to many features, and it is well-known within the common users.

For the purpose of interacting with the *Graph* and *REST* servers, SNCS messaging module makes use of the *RestFb* library, that gives a simple and flexible way of connecting to the Facebook servers and conceal the use of JSON objects. The Facebook Graph protocol gives the possibility to access any public object, such as users, feeds (or Walls), comments, either using their unique identifiers or by their names. However, Facebook is still developing this technology and for that purpose the use of the *REST* server is still an option.

For the communication between the SNCS clients, we use our own message schemas (see appendix), much because Facebook does not allow some types of message, such as plain XML. These schemas are very simple and human readable, in order for Facebook to allow them on their web site, and not consider as any type of blocked messages.

SNCS Constraints: The decision of using Facebook as the Social network, has brought some constraints due to the limitations that it imposes, either with the Use Terms[24] or their API. In order to interact between users SNCS normally sends messages between them by posting on their Walls, which can not be guaranteed between users that are not friends. As such, we use the method of redirecting messages, by sending it to a friends' Wall, so that they can direct the messages to the proper Wall, meaning their friends (FoF method).

Facebook has also limited the size of the messages that can be sent by outside applications, and the method used to

[23]Sigar Library: http://www.hyperic.com/products/sigar on 13/09/2010

[24]Facebook Use Terms: http://www.facebook.com/terms.php

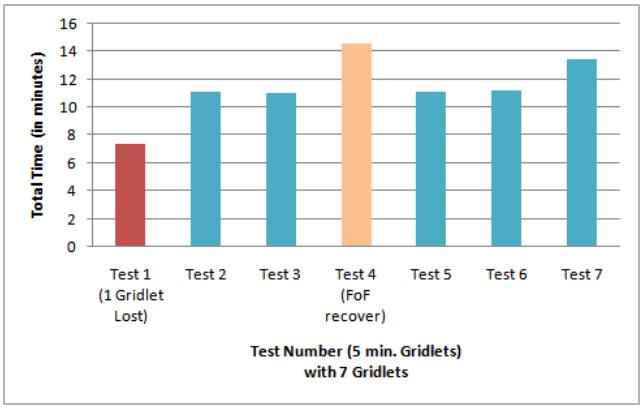

Fig. 4. Total time for 7 Tests in scenario 6

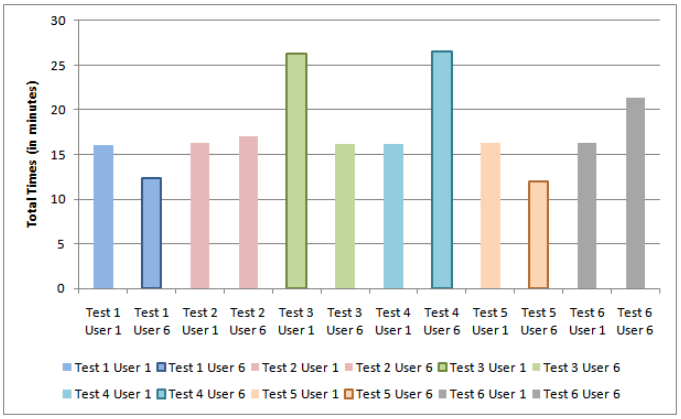

Fig. 5. Total times for Scenario 7

circumvent it was to split messages in smaller ones, making SNCS search for all the parts of those messages.

As it has been said before the Graph protocol is still under development, and as such reading and writing of Comments on groups' Walls is dealt with the REST protocol and also their removals.

The most important constraint is that Facebook also limits the number of requests that can be sent by the client application each day per user. This limit can be changed by Facebook, and is based on the affinity users show for the Facebook application's use of Facebook Platform through their interactions, also *"values will change over time depending on how users interact with your application"*.[25] However, we cannot consider that every Social network has the same limits, and therefore a specialized limitation would have to be reviewed for each case.

V. Evaluation

The evaluation of SNCS addresses its performance, stability and viability to the usage of a Social network. Our focus is to know the achievement of resource and service discovery, by recruiting as many computers as possible to execute *Gridlets*. While also, integrating with the normal usage of the Social network, meaning the amount of information sent to the Social network should be kept minimal. Finally, SNCS should leverage idle cycles to be used.

Our evaluation included several scenarios, where the environment for each changes as follows. First we have only one *Gridlet* between two friends, and the *Gridlet* processing time is 1 or 5 mins. The consequent scenarios encompass more than one friend, a friend of friend (FoF) and a group with other people. The scenario 6, as depicted in Fig. 2, that we consider in this paper is comprised of 2 friends, 2 FoF and a group with 3 people, where one of them is a FoF, with 7 *Gridlets* that are processed in 5 mins each.

[25]Facebook Allocations: http://www.facebook.com/insights/?sk=ao_123798840981469#!/business/insights/app.php?id=123798840981469&tab=allocations on 27/08/2010 (can only be accessed by Facebook applications' Administrators)

Scenario 7, as depicted in Fig. 3, is an attempt to test SNCS in a more realistic environment, having a more complex network of users. Thus, we have two users who start a Job (User 1 and 6), where User 1 has three Friends (User 2, 7 and 15), User 6 has two Friends (User 8 and 15). User 2 and 15 have each a FoF not connected to anyone else. Also, we created a group with six people (User 1, 2, 4, 5, 6 and 7). The layout of this network is made in a attempt to maximize the diversity of the users' roles, making it possible for a Job request to reach any of the users.

In this scenario, User 1 and 6 start a Job each, that contain 8 and 7 *Gridlets* respectively, making a total of 15 *Gridlets* to be processed by any of the users in this network. Furthermore, the client application does not restrain itself to gather only one *Gridlet* for each Job, however it only accepts a Job request per user for each "channel" (Group, Wall, Applications' Wall) that the Job request appears in, i.e. User 7 can accept Jobs from the Group it is connected to, from its friend (User 1), and its friend (User 8), where the latter connection is of FoF to User 6, meaning that (in this network) it could acquire four *Gridlets*.

For all scenarios, we assume that the number of *Gridlets* are suitable to complete the Jobs and that they would take exactly the time spent (1 or 5 mins); that all users would have their client application running prior to the start of the Job; that every user can only process 1 *Gridlet* at a time; and that all computers would have the capabilities of processing the *Gridlets* at that time. For the purpose of simulating the processing time, we used a timed count down program for each of the *Gridlets*, that each processing client application would have to *download* from a web site.

Scenario 8 was made in order to evaluate the performance with a real program that renders images. In this scenario we have one friend, one FoF and two users in a group (not counting with the starter) where one of them is the friend. The goal of this scenario is to know if SNCS can function with a real processing program, such as Pov-Ray,[26] which is used in the tests. For each test the number of *Gridlets* to be

[26]Pov-Ray: http://www.povray.org accessed on 15/10/2010

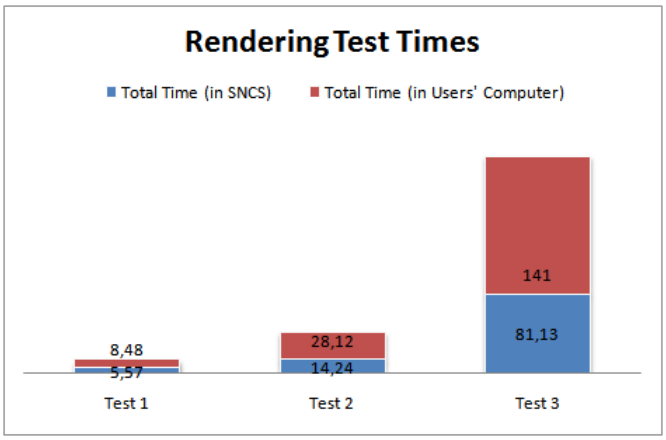

Fig. 6. Rendering Test Times for Scenario 8

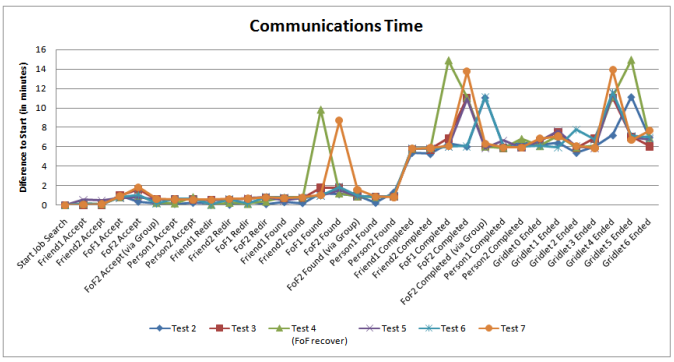

Fig. 7. Times of each action for each test

completed is four and their consuming times in the processing computers are defined by each data file used.

Moreover, in the first test (in scenario 8) we used a small data file, which would render a medium quality image (1280x720 of size, with anti-aliasing at 0,3). In the second test we used a larger data file, and with the same quality as the previous test. In the third test, we used the previous data file, but with properties that would render an high quality image (3921x2767 of size, maximum anti-aliasing and quality).

The resulting times from scenario 6, as depicted in Fig. 4, gives insight to how SNCS behaves in Facebook. Meaning that, in each test the times to complete a Job were in the order of 11 mins. Although, in Test 1 the FoF2 did not received the last *Gridlet* as it was supposed to, and in Test 4 the FoF2 crashed and recovered the last *Gridlet* in time to complete it. These situations proved that the total times, can be hindered by the fact that people are not always in a *Online* state and also by giving more than one *Gridlet* to the same user the Job will have higher total times. Furthermore, when comparing with traditional processing, SNCS decreased the total processing time by approximately 68% in time w.r.t. what it would have consumed in the users' computer.

In Fig. 7 is explained in detail how much time each task takes in relation with the starting point, i.e. it can take less than 1 min for users' client applications to find and accept new Jobs, and that the higher spikes are caused by the fact that the client application only found the *Gridlet* some minutes later due to its *Offline* state.

The results for scenario 7, as depicted in Fig. 5, brings us closer to understand how SNCS performs in a realistic environment. In this scenario we can see that the total times can vary depending on factors such as number of *Gridlets*, users states (*Offline* versus *Online*), number of users/groups involved, Social network latency and use of concurrent *Gridlets* (or Gridlet queue).

The times on this scenario are around 16 minutes to complete both Jobs, however we can see that in Test 1 and 5 the Job initiated by User 6 was completed 5 minutes earlier than in the other tests, this is due to the fact that the *Gridlets* were evenly distributed through the available users.

The results for scenario 8 confirmed that SNCS can gain speedups against local execution, as depicted in Fig. 6. Where we have the total times in Test 1 around 6 minutes, Test 2 around 14 minutes and Test 3 with 81 minutes. Furthermore, in all the tests the friend user processes 2 *Gridlets*, meaning that it queues one to be processed when it has idle cycles to spare.

Test 3 demonstrates that with longer running *Gridlets* the variables that hinder the overall performance, can be amortized by the difference that it would take to process all the data in the user's computer.

We can conclude that the overhead which SNCS gives to the overall process can be minimal compared to the time it takes to process a *Gridlet*. However, times can be hindered by the fact that searching for resources may not return positive results or that the total resources available are less than the number of *Gridlets* to be processed, or even that latency of Facebook servers may vary with their global traffic load.

We can also conclude that the number of messages will vary with the number of users (friends, FoF and groups) that comes in contact with the Job while varying with the number of *Gridlets* comprising the Job. Meaning that in this scenario the number of messages in total were 41, considering that between friends and groups there are 5 messages for each user that accepts a Job, and that for FoFs there are 8 messages for each.

We can state that the number of messages sent to Facebook are proportionally increased by the number of users in the network, meaning that a Job may receive as many accepts and denies messages as users in the network. Although, the user may not be aware of this in the long run, because those messages are erased when they are not needed, making a clean environment in Facebook, meaning that we can accomplish our goal of making SNCS viable to use Facebook without hindering the usage of the Social network.

With functionality and quantitative evaluation, we can conclude that the results are encouraging, despite the overheads introduced by the variable Facebook latency, and the intermediate messaging among FoFs. In fact, with SNCS, Jobs are completed faster than in the user's computer, releasing it

for other tasks. The performance gains would increase with longer running *Gridlets* (more realistically about 1 hour) by amortizing overheads attributable to Facebook.

VI. Conclusions

In this paper, we presented a new method of resource and service discovery through the usage of a Social network. It is also considered that by making use of a Social network already established, we can involve more people donating their computers' idle cycles.

We also describe a platform (SNCS) designed to use Facebook, to search for potential resources available on this Social network, that can execute *Gridlets*.

We evaluated SNCS with scenarios that resulted in leveraging idle cycles and faster execution. Also, the total time of a Job can vary depending on the availability of the resources on the Social network. Furthermore, we can state that longer running *Gridlets* would amortize the variables which hindered the SNCS performance. Moreover, we demonstrate that it is possible to make use of a Social network to perform generic distributed computing, and not only for a single problem.

In the future, we plan to further SNCS addressing the issues of having a realistic environment and complementing with the results of real peoples' usage. While also, making use of other known programs to process *Gridlets* (such as image/video generation). Furthermore, we believe adding a Jobs topics' ontologies would increase the chance to direct the *Gridlets* to people that can be more appropriate to process them.

References

[1] Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th international conference on World Wide Web*, page 844. ACM, 2007.

[2] L. Amaral, A. Scala, M. Barthelemy, and H. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States of America*, 97(21):11149, 2000.

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

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

[5] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. 2009.

[6] L. Backstrom. Group formation in large social networks: membership, growth, and evolution. In *In KDD 06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM Press, 2006.

[7] 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.

[8] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *International Conference on Distributed Computing Systems*, volume 22, pages 23–34. IEEE Computer Society; 1999, 2002.

[9] L. Gao, Y. Ding, and H. Ying. An adaptive social network-inspired approach to resource discovery for the complex grid systems. *International Journal of General Systems*, 35(3):347–360, 2006.

[10] F. Heine, M. Hovestadt, and O. Kao. Towards ontology-driven P2P grid resource discovery. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 76–83. IEEE Computer Society Washington, DC, USA, 2004.

[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95. ACM New York, NY, USA, 2002.

[12] G. Manku. Routing networks for distributed hash tables. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 133–142. ACM New York, NY, USA, 2003.

[13] E. Meshkova, J. Riihijrvi, M. Petrova, and P. Mhnen. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer Networks*, 52(11):2097–2128, 2008.

[14] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, page 42. ACM, 2007.

[15] 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. Sixth IEEE International Conference on*, pages 193–202. IEEE, 2006.

[16] C. Papadakis, P. Fragopoulou, E. Athanasopoulos, M. Dikaiakos, A. Labrinidis, and E. Markatos. A feedback-based approach to reduce duplicate messages in unstructured Peer-to-Peer networks. In *Integrated Workshop on Grid Research*. Springer, 2005.

[17] P. 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.

[18] R. Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[19] J. 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.

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

[21] 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.

[22] D. Tsoumakos and N. Roussopoulos. A comparison of peer-to-peer search methods. In *Proceedings of the Sixth International Workshop on the Web and Databases*. Citeseer, 2003.

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

[24] C. Wilson, B. Boe, A. Sala, K. Puttaswamy, and B. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 205–218. ACM, 2009.

APPENDIX

Schemas used in the messages on Facebook

```
//Schema for Searching People to do a job
SNCS;Job;MyId;<hardware/software required>;

//Schema to answer to a job / JobGridlet
SNCS;<Job/JobGridlet>;MyId;<Accepted/Denied/Completed>;<JobId>
//Special versions: only on comments (App Wall Post) adds a JobId

//Schema to give a GridletJob to someone
SNCS;JobGridlet;JobId;GridletNumberX;MyId;YourId;<Program>;<File>;<Commands
needed to execute>;<Other comments needed>;
//The file should be the place where the client application can download it,
//MyId and YourId should be here so that if someone else reads this
//they should skip it if its not for them, and security reasons

//Schema for GridletJob Status Update (completed example)
SNCS;JobGridlet;MyId;Completed;<JobId>;<GridletNumber>;<where to download>;

//Schema for Redirect Messages
SNCS;Redir;<PersonId to redirect to>;<PostId>;<Type>;<rest of the message>;

//Schema for Redirect Request
SNCS;Redir;Request;<RedirToID>;<Type>;<UserId that started the redirection>;
//the last is optional

//Schema for Splitted messages
SNCS;PartX-Y;UserID;<Rest of the message>
//X is the number of the part, Y is the total number of parts, UserID must be
//placed because Facebook sometimes forgets from whom the message belongs
```

Times for each action for the tests in scenario 6 (larger figure):