Gridlet Economics: Resource Management Models and Policies for Cycle-Sharing system

Pedro Filipe Oliveira
pedro.f.oliveira@ist.utl.pt

Instituto Superior Técnico

Abstract. Nowadays the home computers connected through the Internet are a huge potential source of computational resources. Some systems have already been developed to take advantage of that potential, by allowing the home users to create huge networks, where they can share those resources. However they have been mostly focused on file-sharing and less attention has been given to the sharing of computing resources, namely idle computer cycles. The creation of a system that allows home users to share their idle computer cycles, raises the issue of how those resources are going to be managed.
This paper presents the advantages of using an economic model for that management and an overview on the following characteristics of economic models: what can be used as currency, how the prices can be defined and how to choose between different price options.
Another characteristic that is also analyzed is the reputation. Reputation can have a great influence on the success or failure of an economy, it has the ability of influencing prices, generate trust, attract more users and induce good-behavior.
This paper presents as well an analysis on the resource discovery of the existing types of P2P systems (unstructured, structured and hybrid) and how they can be used in the management of resources in cycle-sharing. We also propose the development of an economic model for the management of the resources and its implementation on a cycle-sharing peer-to-peer system.

Keywords: resource management, peer-to-peer, cycle-sharing, economic models, reputation, resource discovery.

1 Introduction

The increasingly bigger computation power and cheaper prices of the commodity computers made them more and more popular. When connected through a high speed network these computers have the potential of providing bigger computational capability then a supercomputer and at a cheaper price. So, the Grids appeared to take advantage of that potential. Institutions with tens or hundreds of computers brought them together to solve computationally intensive problems. However this type of systems only uses the computers of institutions, leaving a huge pool of computational resources unused, the home computers that nowadays are connected by the Internet.

The BOINC system, with projects like SETI@home and Folding@home [3, 31], take advantage of those resources by using them to perform the CPU-intensive calculations necessary for scientific investigation. However these systems traditionally follow a rigid client-server model, with a centralized server that is the only one that can create the jobs that are executed. This means that the home users cannot take advantage of the resources of the system for which they contribute.

The P2P paradigm is based on the principle that every component of the system has the same responsibilities, acting simultaneously as a client and as a server. This means that a user that contributes to the system can also take advantage of it. There has been done considerable research on peer-to-peer systems and several successful applications were developed. However the main focus of these systems has been on file-sharing [5] and less attention has been given to the sharing of other resources, such as idle processor cycles.

One problem that arises of the use of a peer-to-peer system to do cycle-sharing is how to do the management of the resources, since the models of resource management used for file-sharing cannot be applied. The bit torrent [4], the most popular peer-to-peer content distribution system (represented 35% of Internet's traffic in 2005) uses for resource management a tit-for-tat mechanism where a user exchange uploads for downloads. But this mechanism is based on the fact that a user can contribute to the system by uploading the chunks that he already possesses while downloading those that he is missing, thus exploiting the demand to increase the supply. This model is not applicable to cycle-sharing, because in this type of systems the moments of contribution and usage can be far apart in time.

The model of resource management used in BOINC also cannot be employed, because in that type of systems there is no concurrency in the access to the resources, since the only one that can use them is the central server. The resource management model that is more similar to resource management in a peer-to-peer cycle-sharing system, is the one used in the Grids. However, in those systems it is assumed that all computers are trustworthy, that the components are relatively static and that there is no need to encourage contributions. Those assumptions cannot be made in a peer-to-peer system. So, in this paper we propose an economic model for the resource management in a peer-to-peer cycle-sharing system.

The rest of this paper is organized as follows. In the next section we is described the objectives of the economic model. In section 3 the related work is presented, with a description of the economic models in section 3.1. The reputation systems and their impact are described on section 3.2 and the section 3.3 will analyze the resource discovery. Section 4 shows the proposed architecture for the model and in section 5 is presented how it is going to be evaluated.

This work is part of a larger project called GINGER [35], an acronym for Grid In a Non-Grid EnviRonment, a peer-to-peer infrastructure intended to ease the sharing of computer resources between home users

2 Objectives

In a peer-to-peer system for cycle-sharing there are resources available to be used and jobs submitted to be executed on those resources. The main objective of this work is to create an economic model capable of managing the resources of this type of system, by deciding to which resource goes each job. The model must inform the system to make the decision in order to ensure that if there is a job to be executed and resources available to execute it, then the job is executed. That means that it has to have also a mechanism for finding available resources and map them to the jobs. The resource discovery mechanism must be able to deal with multiple requirements, such as: CPU speed, network bandwidth, etc.

The decisions of where one or a set of jobs submitted by a certain user are executed have to be made automatically. But the users must have the possibility to specify if they want that decision to try to comply with a certain deadline or to use the cheapest resources available. Because this is a cycle-sharing system the decisions must also take into to consideration multiple requirements that have to be fulfilled to execute a certain job, such as: CPU speed, memory size, applications installed, OS, etc. And because the requirements may change, it must be possible to add new requirements.

In order to keep the good functioning of the system, the model must be able to make the distinction between different types of users according to their contribution and consumption of resources. This implies that the model has to have a mechanism that controls how much a peer contributes and uses the resources of the system. That control must be flexible and take into consideration multiple aspects when evaluating how much a peer contributes and consumes. Contribution in a time when there is a lot of demand should be more valuable than contributions when there is more offer than demand. Contribution of more powerful and popular resources also should be more valuable. On the other hand, consumption in times of less usage or of less popular application should be considered of a minor importance. The mechanism that controls the contribution and consumption of the peers should enable the creation of incentives, such as better quality of service to peers that have good ratings, in order to encourage behavior which is in the system's overall best interest.

One problem that appears in economics is when someone asks for more money then what he should receive by claiming that he did more work than what was actually done. This is a problem that also can appear in cycle-sharing because is hard to know exactly how much processing time a task is going to take to be executed. This problem is called *overpricing*. The model should be able to detect and avoid this problem. Another problem that happens in economics if *fraud*. In cycle-sharing, it is considered *fraud* when a peer instead of executing a job and returning the result, it does not executed the job and simply sends an answer that simulates the result in order to receive credit. The last objective of the model is to be able to isolate peers that commit *fraud* from the system.

3 Related work

3.1 Economic models

When there is demand and supply of resources there has to be some management of who uses what and when. Economic models are a good way of doing that since in its basis they are supposed to do that in real life scenarios. Also, they have already been used for many centuries and proved to be a successful and sustainable way of regulating the exchange of resources, goods and services.

The use of an economic model provides a scalable option for the management of resources, especially when they are not all in the possession of the same entity, because each user regulates the use of its own resources. This means that there is no need for a central coordination, which would create a bottleneck, and speeds up the decision process, because the problem is distributed across all resource owners. Also, it offers incentives for resource owners to contribute their resources for others to use, since they profit from it, contributing to the growth of the system. Moreover, it provides a simple method for defining the priority order of the jobs, by establishing that the ones for which the users are willing to pay more have the highest priority. Likewise, it encourages the users that have jobs with a lower priority to back off and let the more time critical ones be executed first, since that means they will pay less.

Other advantage of the use of an economic model is flexibility since it allows a uniform treatment of all kinds of resources, from CPU time to application version. Also, both resource owners and consumers want to maximize their profit, i.e. the owners wish to earn more money and the users wish to solve their problems with the minimum cost possible. Economic models have the flexibility to allow the users to express their own requirements and objectives, enabling the development of scheduling policies that are centered on the users instead of the system, placing the power on the user's hands.

Therefore it is considered that an economic model is suitable for the management of resources in a decentralized environment where resources can vary and are owned by different entities. There already exist some systems that implement economic models for resource management [8, 9, 17]. However, none addresses the particular aspects of resource management in a peer-to-peer cycle-sharing system. Next, we will present three fundamental aspects of economic models: currency, price definition and price selection.

3.1.1 Currency

When using an economic model there has to be of some type of currency which is exchanged in the transactions. In a peer-to-peer system there are many options that can be used as currency. The types of currencies can be divided into two categories: non-monetary and monetary.

The non-monetary systems are simpler and easier to build but, have some limitations. Next, are presented the non-monetary currencies.

Resources: the use of the resources that we have in exchange for what we need in a direct exchange, as used in [2, 10]. For example, exchange disk space for processor time. The direct exchange of resources is also called bartering. It is used mainly to start economies because of the simplicity associated with the fact that it is a non-monetary system. Monetary systems need to give value to something that by itself does not have any value, the money, independently of what is used as money, and that fact makes it much more complex. Also, bartering is safer, because it is harder to deceive someone if the good has to be delivered on the moment of the trade. But it has some disadvantages: one can only trade if one has possession of the resource at the time of the trade. This means that if it is a resource that is consumed but never leaves the possession of the owner, it is impossible to purchase it for later use. It is also difficult to make the conversion of value between different types of resources. Besides, some resources cannot be traded only half, so the thing that it is traded for it as to have the same value.

Although traditionally bartering is an operation solely between two actors, there is the possibility of making a barter transaction with more than two actors [2]. In that type of transactions a ring of direct exchanges is formed where someone sends a resource to someone that passes a resource of the same value to other; the other sends to someone else that eventually gives some resource to the original sender.

Multiple virtual currencies: multiple virtual currencies is an approach where any user can issue a currency that can be traded, is used in [13, 34]. This type of economies is the next step after bartering. Instead of trading goods directly there are traded tickets that have the value of the good. This type of economy solves some of the bartering problems since there is no longer the need to have possession of the resource at the time of the trade; it allows the possibility of buying a resource to use later. Also, it is more flexible, because if one has a ticket or coupon that gives him the right to use the resource x, he can trade it instead of using it. Still, this type of currency has some disadvantages, the ticket it is only valid for a specific type of resource, so the problem of conversion between different types of goods remains. Also, the value of the ticket is based on the trust that one have that the entity that issued the ticket is going to fulfill the right that it grants. So one can only trade the ticket with users that also have trust in the person that issue the ticket. Besides, this makes it easier to deceive someone, since it is simpler to trade tickets that do not correspond to real resources, then it is to fake resources.

Monetary systems solve most of the problems that exist in non-monetary systems. The conversion between different goods is now simple because they are priced based on the value of the money. Also, they have the advantage of being more practical. But there is still the need to decide what is going to be used as money:

Real Money: the use of real money has many advantages. The problem of giving value to this type of currency does not exist, since it is already valuable outside of the system. Another advantage is that there exist already many payment systems [12] that store the money of the user and make the transactions. But there are some legal issues

[24] and implications that raise many problems. The security issues are much more important because if someone is able to subvert the system the repercussions are much greater. Moreover, in a peer-to-peer system the paradigm is that there is no overseeing entity, so it raises the problem of who is responsible for preventing frauds from happening. Also, the use of real money makes the system vulnerable to users that have a huge wealth, because puts them in position in which they can monopolize the system.

Virtual money (micro-payments systems): virtual money is a simplification of the use of real money. This type of money has no value outside of the system, so if someone subverts the system there are not very big repercussions. That also implies that the security issues of how to store the virtual money are less important. Moreover, with virtual money users are more likely to allow the usage of that money to be automated. But it has some problems, since initially this type of currency has no value; there is the need to make it have value. People only value money because they believe that, later on, they will be able to trade it for something that they want. So, in order to make the virtual money valuable, one has to make its users trust that the system will have things that they want in the future and that they will be able to trade the virtual money for them. Another problem is that all the virtual money that a user manages to save in that system can only be used in that system. If it was real money that was being used, it could be cashed-out and inserted into another system. In [16, 36, 22] is presented some work that has already been done in peer-to-peer systems using this type of currency.

Renewable money: renewable money is a special type of virtual money that self re-charges over time, depending on the system policy. The type of mechanism is mostly used when a single entity has control of all of the resources and the users have no way of contributing to the system [19, 14]. It can also be used as a simplification of the virtual money, because it has fewer problems. In a system that uses virtual money if there is more offer than demand, the new users will have difficulty in selling their resources to win credits to use the system; in a self charging system that problem does not exist. To have flexibility, the users can be divided into classes, based on the resources they have, and use that to condition the rate at which their money re-charges.

The choice of the currency should be a careful one, because the type of currency used, due to the implication that it has on the economy, can determine the success or failure of the economic system.

3.1.2 Price definition

The price definition is an important factor of economic models, as it happens with the currency the set of rules used to define prices has a great influence on the success or the failure of the system. The alternative selected defines how the resource owners can make their choices and maximize their objectives. There are many different mechanisms for defining the price in a normal economy. Next we will present the models for price definition presented in the GRACE framework [8] along with some critics:

Commodity Market: commodity market is the traditional type of market. In this type of market the resource owners specify the price and the consumers only have the

choice of buying or not. That happens because, unlike other mechanism such as auction or bargaining, the consumers have no direct influence on the price definition. The price definition policy can be *flat*, where the price does not change for a certain amount of time, or *variable*, where price changes very often based on the amount of supply and demand. The *flat* policy is often used in markets where the supply and demand have very small variation over time. Whereas the *variable* policy is used when there are large discrepancies between the low and high peak of demand or the supply varies widely with some regularity. For example in a cycle-sharing environment probably the supply will increase highly during the night in a given set of time zones.

This type of market is relatively simple to implement, the most complex part is to define how the prices are calculated, and it is mostly used when the market is at equilibrium, because it is very stable. But it has the disadvantage of not achieving an optimal solution, since it does not take into consideration the value that the consumers give to the resource (i.e. their utility). The systems in [9, 17] consider this type of market.

Posted Price: posted price is not exactly a type market, but more of a mechanism that is used by the resource owners to advertise special offers. This mechanism can be used to attract new customers, which normally would not buy or use that service, in order to establish or increase the market share. Also, it can be used to encourage users to use the service in a time of less demand, because the posted prices are cheaper but can have usage conditions associated to them. Although the tactic of reducing the prices can seem to go against the objective of maximizing the supplier profit, because it reduces the profit margin, in some cases it might be better to have a smaller profit than having no profit because nobody is buying or using the service.

Bargaining: bargaining is a mechanism used to set prices that works in the following way: the resource owner establishes a price and consumer makes a bid with a value that is lower than the price asked. If the resource owner does not accept the bid, adjust the price in order to make it closer to the value of the bid. If the consumer does not accept the new price, raises slightly the value offered. They both continue negotiation until they reach a mutually agreeable price or one of them is not willing to negotiate any further. Like the posted price this mechanism can be used to increase the market share or encourage the use of the service in times of less demand. The bargaining method is better for selling objects or services whose value is difficult to define beforehand. Also, it is better in maximizing the profit margin of the resource owner, but the negotiation makes it more expensive (particularly in terms of communication) than the posted price method, so if the value sold is very low it might not justify the use of bargaining [37].

Tender/Contract-Net: tender/contract-net is one of the most widely used models for price negotiation of contracts of great value in the real economy. Most of Portuguese government contracts have to be done by this model in order to ensure transparency. In this model the first step is not done by the resource owner, but instead is the consumer that publishes the requirements and sends a request for proposals. Then the interested resource owners answer by stating what they have to offer and how much it would cost. When all proposals are gathered or the established deadline for receiving

proposals is reached, the consumer evaluates the proposals and awards the contract to the most appropriate resource owner.

This model is the one that best maximizes the consumer utility, i.e. solving the problem with the minimum cost possible. However, due to the high cost and time consumption required for the negotiation, its usage is only justified if the value of the contract that is being negotiated is very large. The system proposed in [9] also considers this type of market.

Auctions: as the tender/contract-net model, auctions support many-to-one negotiations. The difference is that, instead of being many sellers and one buyer, in auctions there is only one seller and many buyers.

The three key players involved in an auction are: resource owners, auctioneers (mediators) and buyers. Many e-commerce portals such as Amazon and eBay serve as mediators and act as a trusted third party that regulates the auction process. But in a peer-to-peer environment it is considered that trusted third party do not exist a and if the auctioneer is malicious it can subvert the result of the auction. In [28] is presented a protocol for decentralized ascending auctions where the auctioneers are groups of peers. This protocol is able to ensure that the auctioneer is not capable subverting the result of the auction as long as one of the peers in each group is not malicious.

The auction process can have different rules, depending on this rules the auction can be classified into four types: English auction (first-price open cry), First-price sealed-bid auction, Vickrey auction (second-price sealed bid) and Dutch auction.

The English auction is the traditional type of auction. It opens with a minimum bid and after, each buyer successively increases their offer in order to exceed the other bids. When none of the other bidders is willing to raise the price anymore, the auction ends and the highest offer wins the item at the price of its last bid. In principle this type of model is the one that optimizes the resource owner objective, since the iterative raise of the bids forces the buyers to give the real value to the resource. But in [26] they noticed that the users followed the following strategy: when there was little demand the buyers instead of bidding the real value of the resource offered a much smaller value, reducing the seller profit. Other strategy was to bid just before the deadline of the auction, not giving the other buyers the possibility to make a counter-bid.

In the First-price sealed-bid auction the buyers submit their bids without knowing the other's bid. That means that each user only submits one bid, since there is no way of knowing if there is the need to make counter-bid. The highest bidder wins the item at the price of its bid. In [26] is said that in this type of auction the strategies used in the English auction cannot be used since the buyers do not know if the demand is low and there are not counter-bids.

The Vickrey auction is very similar to the First-price sealed bid auction. The only difference is that the highest bidder wins the item at the price of the second highest bidder. This type of auction is used to encourage the buyers to offer a value slightly larger than the real value since they will not have to pay that price but always a smaller amount.

In the Dutch auction the auctioneer starts with a high bid and continuously lowers the price until one of the bidders takes the item at the current price. It is similar to the First-price sealed bid because in both the buyers have no relevant information about each other. This type of auction has the advantage that it allows the auctioneer to control the speed of the reduction and therefore the time that the auction takes.

Auctions are very popular because they require little information about the item real value or demand. Also, it optimizes the resource owner objective of making the most profit possible. However, this type of model has the costs associated with the auctioneer and it is time consuming.

Bid-based Proportional Resource Sharing/Share Holders: the bid-based proportional resource sharing/share holders model is mostly used in cooperative concurrency to resources. In this model the resources allocated to each user can either be a percentage of the total resources proportional to the bid made in comparisons to the other bids. Or each user can get a share of the system and then the time that he can use the resources is proportional to his share. One advantage of this type of model is that ensures that there is not starvation of any user and that all users, independently of their importance, are given is fair share of access to the resources. However, this model is made to work in a cooperative environment and in a peer-to-peer system is considered that the users are rational and work in a competitive way [7].

Collective/Cooperative: a collective/cooperative is a business which is run, and often owned, by a group of people who take an equal share of any profits. This model is used by smaller business to meet their economic objectives and be able to compete in an environment where there competitors are very large, for example the competition in a tender/contract-net. An example of this model is the small vine farmers which form a cooperative and sell the wine under the same brand to compete with the big companies.

Monopoly: a monopoly is the case where a single entity is in possession of all of the resources or is the single provider of a determine type of service and dominates the market. In this model the resource owner determines the terms on which the consumers shall have access to the resources. The BOINC system [3, 31] is an example of this type of model.

The model chosen to define the prices influences the ability of the resource owners and consumers to maximize their objectives. Also it can be a factor to attract new consumers or cause them to leave.

3.1.3 Price selection

In an economic model normally the resource consumer has many options of services from which he can choose. The alternatives can offer exactly the same service for the same cost, in that case the selection of the resource owner that is going to provide the service is pretty straightforward. But usually that is not the case; normally each alternative offers a different type of quality of service, which can fully or partially satisfy the requirements of the resource consumer, and a different price. In this case the selection of the right service provider is crucial to achieve the user objective.

In economics, utility is a measure of the relative satisfaction of the consumer with the consumption or purchase of a determined good. Usually the utility is represented through a function, called the utility function. That function can take into consideration many different parameters and it is very important to the price selection, because the right selection can be made by choosing the option that maximizes the value of the function that corresponds to the consumer perceived utility.

However the work that a user submits is normally not executed as single job, but as set of smaller units. So, the selection of where each job is going to be executed must not only take into consideration the factors or utility that makes that single choice the best. But also the factor that globally reach the objectives of the user for that work, as for example the ability to meet a deadline. In [1] it is presented a broker that selects the service provider based on one of two user policies: budget or time. With the budget restriction the broker tries to spend as little money as possible. The time restriction has two modes of functioning: either the user sets a deadline and the broker tries to meet that deadline spending as little as possible or the broker tries to execute the job as fast as possible without taking into consideration the costs.

In an economic environment the risk is another important aspect to take into consideration in the price selection. More important in a peer-to-peer system where there is no regulating authority, which means that both the resource owners and consumer can try to take advantage of one other without punishment [20]. So, the reputation (section 3.2) can be used as factor for price selection, since it is a measure of the risk of the transaction. In economics the riskier an investment is, the bigger the pay-off as to be in order to justify the possibility of losing money. That means that if the reputation of a peer is low it cannot ask for a high price, because it will not make up for the risk.

While the price definition is the most important factor to achieve the objective of the resource owner, since it defines the price and the buyers have few control over it. The price selection is the most important factor to achieve the objectives of the resource consumer.

3.2 Reputation

Reputation is important in an economic model because trading involves always a certain amount of asymmetric risk (that one of the parts will not fulfill its obligations once the other has committed the resources or currencies) and the reputation is used to reduce that risk. There is the assumption that the behavior of someone in the past is a relatively good predictor of their future behavior. However, in order to use that assumption it is necessary to have some knowledge of the past behavior of that person, which one might not have. The reputation is considered to be the overall opinion of the system about someone or something and can used as an indicator of the past behavior.

One way of using the reputation that is used in the real world are the references, which means that someone already has some knowledge about the past behavior of a

certain person and passes that information along in the form of a reference. In [39] it is proposed a system that uses this mechanism.

Besides reducing the risk, reputation can also be used as a mechanism to induce good behavior in markets with asymmetric information. That concept is not new, in fact several economist have already published work analyzing its proprieties. One of the major auction sites in the world, eBay [15], relies almost exclusive on a system like this to reduce the risk and induce good behavior on the part of its members.

The system works by encouraging the buyers and sellers to rate each other after each transaction. It is a binary reputation mechanism which means that the rating can be one of two values, "praise" (i.e. positive) or "complaint" (i.e. negative, problematic), together with a short text. These classifications are then made publicly available to all users. For a system like this to work it is important that two properties hold:

1. It is optimal to sellers to settle down to a steady-state pair of real and advertised qualities, rather than oscillate, successively building up and milking their reputation.
2. The quality of sellers as estimated by buyers before the transaction is equal to their true quality.

The first property is important because if the seller is allowed and chooses to oscillate, because that would result in additional profits, that profit would be made at the expenses of the buyers, which in the presence of competitive markets, would eventually leave. That would lead to the collapse of the system.

The second property is needed to do expectation management, because the classification given by a peer to a transaction is not based on the real quality of the item. Instead, it is given based on the relation between the quality expected and the real quality item. This means that if a buyer receives an item that is in good condition, but for some reason is expecting it to be even better; he would rate the transaction as a complaint, regardless of the fact that he/she received an item in good condition.

Over the years the reputation system of eBay has proven itself capable of providing a remarkable stability. Still it is requires human interaction to give and evaluate the classifications.

3.2.1 Reputation functions

The reputation system collects the classification that the users gave to the transaction. But that information is only useful if it can be used to compare the reputation of one peer with another. So, there is the need to convert it into a value that can be quantified and compared. In the next paragraph there will be a description of function used to convert the classifications given into a comparable value.

First we have to choose which factors we are going to consider in the function. In [38] the authors identify three major factors to consider when evaluating the reputation:

1. The classification that a peer obtains through transactions.
2. The number of interactions the peer had with other peers.
3. A balance factor of trust, which reflects the trust that there is in the classification given by that peer.

Then the reputation value of a peer *u* can be calculated by the sum of all the classification given by other peers to *u* multiplied by the balance factor of the peer that gives the classification, divided by the total number of interactions of *u*.

The balance factor makes the given function asymmetric. Asymmetric functions consider that some peers are not trusted and that trust is transient. That is important because in [11], it is said that symmetric function are vulnerable to Sybil attacks and badmouthing since they treat every classification as the same. Also the fact that trust is considered transient makes it easier to develop trust in peers to which there has been no prior interaction.

3.2.2 Reputation storage and calculation

The calculation of the reputation value of one peer depends on where that information is stored. In centralized systems, like eBay, the reputation is stored in the server, so it is only logical that it is the server that calculates the reputation value of each peer. This is a simple but effective solution, and since the server is a trusted third party there is no risk of adulteration of the value stored.

However in a decentralized peer-to-peer systems there is no central server or any other trusted third party that can store and calculate the reputation. One option is to store in the neighbors, like in KARMA [36]. The feedback is stored by the adjacent nodes in the DHT[1], this solution is considered relatively safe because the distribution of nodes over the DHT is random, and so it is very difficult to know who is going to be one peer's neighbor in order to have it change the peer's reputation. Nevertheless, the nodes storing the reputation are not trustworthy so the values are replicated into the more than one neighbor, when a peer wants to know someone's reputation it asks all of the neighbors, and in case of receiving different values, uses the majority to reach a decision.

In the EigenTrust Algorithm [21] the reputation is stored locally. Then the algorithm computes a global reputation value for every peer based on the local reputation values assigned to each peer by other peers. This method of calculating the reputation takes into consideration the entire system's history of a peer.

Calculating the global reputation value makes the algorithm more accurate, yet it also makes it more complicated, requiring long periods of time to compute that value. In [25] the authors propose a system that instead of considering the entire history uses only limited information about the peer. The authors show that it is possible still to achieve good results, while reducing the overhead of calculating the reputation.

[1] DHT (Distributed Hash Table) is an infrastructure used in structured peer-to-peer systems to organize the peers. A more detailed description is made in section 3.2.2.

3.3 Resource discovery

The basis of an economic model is the possibility of selecting the option that maximizes the utility of the resource consumer. In order to make the selection there is the need to know what options are available, for that there is the need to discover the resources available at a certain time. The resource discovery in peer-to-peer systems is divided into three main categories, depending on how the nodes are organized: unstructured, structured and hybrid.

3.3.1 Unstructured

In an untrusted peer-to-peer system the nodes are randomly distributed, these types of systems are generally more appropriate for accommodating a highly-transient population due to the low cost of entering and leaving the system. But because there is no information about the organization of the peers, these systems need to use uninformed search techniques that are very inefficient.

Gnutella[2] was one of the biggest file-sharing peer-to-peer systems, it was decentralized and unstructured. In this system each peer was connected to a small number of other peers, the neighbors. The search method was basic flooding, which works by sending the query to all the neighbors and when the neighbors receive it, they forward the same query to all of their neighbors. This search mechanism is not scalable because each query generates a huge amount of traffic that increases exponentially with the number of nodes, so as the number of peer increases the amount of traffic generated by the queries saturates the system. In result, many queries are dropped making the search unreliable. Additionally, due to the high disconnection rate of the peers the network never stabilizes.

In order to prevent the search mechanism blocking the system there are several techniques, such as the use of a Time-To-Live (TTL). The TTL indicates the number of hops away from its source that a query should be propagated to. This limits the amount of traffic generated by each query. However it has some drawbacks, it limits the search to only a part of the system, so the item searched may not be found although it exists in the system. Also, the value of the TTL must be carefully chosen, if too small most queries will return incorrect or not optimal results, if too large the amount of traffic generated by each query continues to be too much. Iterative deepening [23] solves this problem by starting with a small TTL and increasing it in each attempt.

Another technique is performing random walks [23, 18]. In this technique instead of sending the query to all of the neighbors, the peer sends the query only to some randomly chosen neighbors. If the query does not return successfully is then sent to some of the other neighbors; this continues until the query was sent to all neighbors or returns a positive result. The method reduces in many cases the traffic overhead imposed on the network with the tradeoff of a highly variable performance and, in

[2] http://wiki.limewire.org/index.php?title=GDF [last checked: 2009-12-30]

contrary to flooding, the number of query replicas does not increase with the hop distance.

Kazaa[3] uses a hierarchic approach with two classes of nodes, super peers and ordinary nodes. The super peers connect between themselves and form an unstructured overlay network while the ordinary nodes are each on connected to one of the super peers. Each super peer maintains an index of the files present in the ordinary node that are connected to it and the searched is made only among the super peers. This technique allows the use of flooding while maintaining scalability, since the number of nodes search is much smaller. Also, it has been demonstrated that nodes with low bandwidth and less computational power were the ones being saturated and slowing down the search process; this approach take that fact into consideration and therefore it isolates the slower nodes from the search process.

3.3.2 Structured

In a structured peer-to-peer system the peers' distribution follows a rigid organization over an indexing service based on hashing, known as Distributed Hash Table (DHT). Peers and keys are mapped through a hash function which allows the finding of the peer corresponding to a key in a very efficient manner. The number of messages grows smoothly with the number of peers. In this way they solve the scalability problem of unstructured systems, but have the disadvantage of being less suitable for highly transient populations due to their strict organization of the nodes which implies high costs for entering and leaving the system.

In Chord [33] the keys and nodes are mapped over m-bit key space and the key is assigned to the first node whose ID is equal or greater than the value of the key. Chord distributes its nodes over a one-dimensional ring according to their ID, so in order for it to work a node needs to know his successor. To locate the node corresponding to a certain key the queries travel through the ring always in the same direction (clockwise or counter-clockwise) until it reaches the desired node. However this mechanism requires O (N) messages to locate the node, which is highly inefficient. To speed up the lookup process, each node maintains a table with m entries. In the first entry of the table the node stores a pointer to the node that is responsible for the key half-ring away from it, in the second entry to the node responsible for the key one-quarter away and on the third entry the one one-eighth away and so on. So before sending the query to its successor, a node checks its finger table and if one of the entries points to a key that is lower than the one it is searching, it sends the query directly to that node further away as possible, skipping the nodes between. This emulates a binary search, thus requiring O (log N) messages to locate a node corresponding to a certain key. In order to maintain its rigid structure, Chord periodically runs a stabilization protocol, where if one node fail its successor becomes responsible for its keys. In order for it to work each node needs to maintain a pointer to its predecessor. The update of the systems in case of a node join, leave or fail requires O ($\log^2 N$) messages.

[3] Kazaa, http://www.kazaa.com/us/index.htm [last checked: 2010-01-02]

The CAN (Content Addressable Network) [27] uses a virtual *d*-dimensional Cartesian coordinate space to map the keys onto to nodes. The coordinate space is divided into zones, which are segments of the space, and each node is responsible for a certain zone. The keys consist of *d* numbers and correspond to a point in the coordinate space and the node responsible for that zone is the one responsible for that key. Each peer is connected to its next and previous peer in each dimension. In order to do the lookup for a certain key the CAN uses a greedy strategy where it sends the message to the neighbor that is closer to the key until it reaches the desired location. This way, the lookup will follow a straight line through the Cartesian space from source to destination. The lookup requires O ($d \cdot N^{1/d}$) messages and a routing table with 2d entries. However, in order for the lookup mechanism to function it requires a continuous coordinate space without any unassigned zones. So when a node leaves the system, the zone of one or more neighbors is enlarged in order to contain the zone that was left unassigned. The creation of a new zone to allow the join of a new node is made by splitting one the existing ones. Nodes joining or leaving have a localized effect and require only O (*d*) messages to update the system.

3.3.3 Hybrid

Unstructured systems allow a random distribution of nodes, which have the advantage of being suitable to highly transient populations, but make the search very inefficient. Structured systems organize the distribution of the peers in a strict way, which enables a very efficient search, but does makes joining and leaving of peers very costly. Hybrid systems try to maintain the efficiency of the search of the structured systems, while employing a less strict organization, thus making them more suitable for transient populations.

In Pastry [29] each peer has a unique 128-bit node identifier (nodeId) that is used to indicate the node position in circular space. The nodeId is assigned randomly when a node joins the system, so with high probability the nodes with adjacent nodeIds are diverse in geography or jurisdiction. However Pastry takes into account network locality and seeks to minimize the distance messages travel, according to a scalar proximity metric like the number of IP routing hops. Each node in Pastry maintains the following state: a *routing table*, a *neighbor set* and a *leaf set*. The routing table has $\lceil \log_{2^b}(N) \rceil$ rows with $2^b - 1$ entries each. The entries at row *n* of the routing table refer to nodes whose nodeId shares the present node's nodeId in the first *n* digits. This means that each entry has many possible nodes, the choice of the node is made based on the proximity metric. The *neighbor set* contains the *M* nodes which are closest to the node according to the proximity metric. The *leaf set* is the set of nodes with the $|L|/2$ numerically closest larger nodeIds, and the $|L|/2$ nodes with numerically closest smaller nodeIds, relative to the present node's nodeId. The routing in Pastry is made by first checking if the key falls into the range of the nodes in the *leaf set*. If so, the message is sent directly. If not, then the *routing table* is used to do the routing in a tree-like manner. The routing process is done in less than $\lceil \log_{2^b} N \rceil$ steps and will succeed unless either half of the *leaf set* nodes fail simultaneously. The *neighbor set* is used to aid new nodes join the system. The nodes that are closest to the new node are likelier

to have a similar state, thus providing a good starting to point for the initialization of the entering peer.

3.3.4 Complex resource discovery

So far the resource discovery described only considered exact match queries of a single value, the key, that are mostly used in file-sharing systems. But in a cycle-sharing environment the search has to take into consideration multiple parameters that might have a range of values which are acceptable, more like in Grid environment. For example, one peer may want to discover the nodes with a processor speed between 2 and 3GHz and from 1 up to 2 GB of RAM memory.

In [18] Iamnitchi et al. porpose a fully decentralized architecture for resource discovery that deals with requests for a set of desirable attributes rather than a global unique identifier. In this architecture, participants are called Virtual Organizations (VO) and can be home users or institutions. Each VO publishes information about one or multiple of its resources on one or more local servers, called nodes or peers. To discover a resource the user sends a query to a known node. If the query request matches locally, the node responds with a matching description. Otherwise it forwards the request to another node, the other node does the same until the query matches or the TTL expire. The forwarding of requests is made based on one of the following strategies: random walk, learning-based, best-neighbor, learning-based + best-neighbor. In the random walk the request is sent to a randomly chosen node. In the learning-based the node remembers the past answers and sends the request to a node that had already answered to similar query, if there is no relevant experience a node is chosen randomly. In the best-neighbor strategy the node registers how many answers each neighbor has provided and sends the request to the neighbor with the highest number. The learning-based + best-neighbor is identical to the learning based, but when no relevant experience exists the query is forwarded to the best neighbor. Nevertheless, this architecture is based on unstructured peer-to-peer systems, so it has the same disadvantages.

In [6] Andrzejak et al. propose an extension to CAN that allows ranged queries. In the extended CAN the “objects” are pairs (attribute values, resource id), where the attribute value is a real number. Only a subset of nodes participates, called the interval keeper (IK), and is responsible for storing the pairs of a certain interval. Each server in the Grid reports its current attribute value to an IK with the appropriate interval. To support multiple attribute queries the system must have one DHT for each attribute, depending on the characteristics of the attribute it will be used the standard or extended version of CAN. The query is split into multiple smaller queries, one for each attribute, that are solved separately and in the end the results are concatenated in a join-like operation.

In XenoSearch [32] it is proposed an extension to the Pastry system. As in the extended CAN, each attribute is mapped into a different DHT and the query for each attribute is resolved separately. Ranged queries are possible due to the fact that the information is conceptually stored in a tree, where the leaves are the XenoServers and

the intermediate nodes are aggregation points (AP). Each AP summarizes the values of the nodes below it in the tree, the key of the AP is a prefix of the child nodes IDs, so it possible to determine the range of values of the child nodes.

The system presented above needs multiple DHT to support multiple attribute queries and has the associated overhead that maintaining all of them requires. In [30] Schmidt et al. propose a system that supports multiple attribute queries using a single one-dimensional DHT. The system uses a space filing curve which maps all possible *d*-dimensional attribute values to a single dimension. Attributes are mapped onto nodes by interleaving their binary representation. For example, a resource For example, a resource containing three attributes with values (3,2,1) is represented in binary as (**1**1,**1**0,**0**1), so it will stored in the node with the ID **110**101. Notice that as the number of attribute increases so does the size of the ID. Ranged queries are possible by leaving some undefined bits that can have any value. For example the query of resources with attribute values (1, 2, 0–3) is represented as 01*00*. However the range query sizes can only be powers of two and can only start from values that are also powers of two.

4. Architecture

This paper proposes the design of an economic model that will be applied on top of a peer-to-peer system for cycle-sharing. In this system the users make their computational resources available to be used by the other participants. So that when a user wants, he can submit jobs that are executed on the resources of the other users of the system. The jobs submitted are split into smaller work units, the *Gridlets*, in order to be distributed over the resources available in the system. In this work it is assumed that an existing layer is responsible for doing the creation of the *Gridlets* and aggregation of the results.

In the economic model the users will buy the right to use the resource of the system. For that, it will be used a system of credits, which also will serve as a control mechanism of the relation between how much a user contributes and consumes the system resources. A reputation system will be used as well, and according to the user reputation and history in the system (the number of *Gridlets* executed, hours in the system, etc.) the users will be divided into classes. Depending on the class that a user is in, a different quality of service will be provided. For example: the *Gridlets* of the users in a higher class will have a higher priority or the users from a lower class might not have access to a part of the system resources.

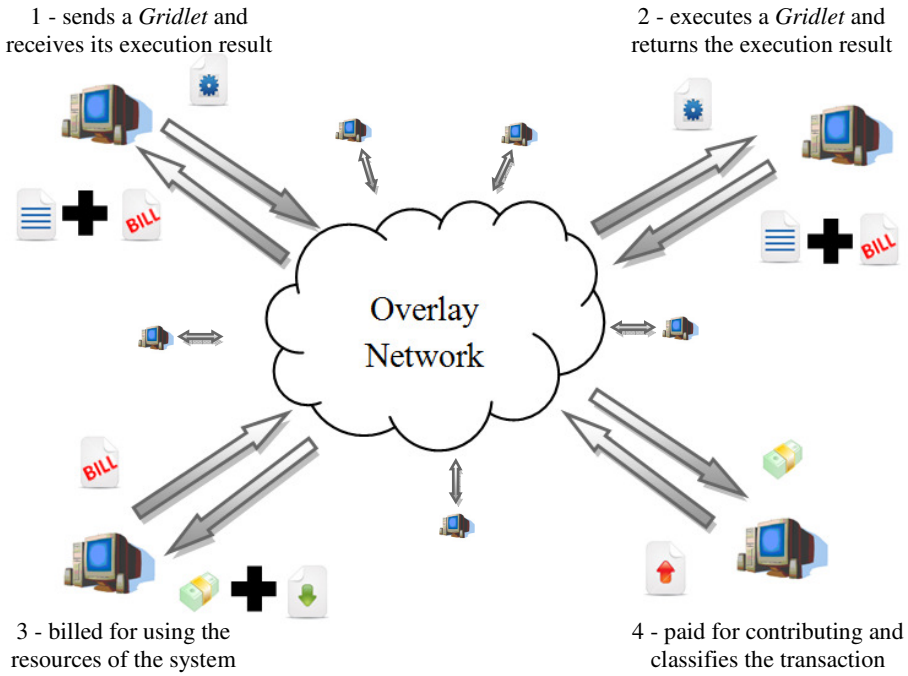

Figure 1 – Overview of the system

In Figure 1 is presented an overview of the system where the nodes execute transactions that involve *Gridlets*, credits and classifications. Next it will be presented a more detailed description of how the system will work

For the model to operate there will necessary two modules that will work together. The first module will be used when the user wants to consume the resources available on the system, so it will be called the *consumer*. The second module is the *contributor*, because it will be the one responsible for contributing to the system the resources that the user specifies.

The *consumer* module work starts when it receives the set of *Gridlets* that are going to be submitted into the system. First it will ask the user to specify the resources (CPU speed, network bandwidth, application, etc.) required to execute *Gridlets*. That information can also be supplied by the layer that created the *Gridlets*. It will also ask the user the ideal values of the unit of cost to execute it. The unit of cost will be a set of pre-defined characteristics (CPU, bandwith, price, etc.) that will be used to select the node that maximizes (or at least fulfills) the user utility. The components of the unit of cost will be compared, so they will have to be scalars that share a relation among them. This means that characteristics such as CPU speed or network bandwidth can be used as components in the unit of cost, but characteristics like the OS or applications installed cannot. Since the exact match might not exist in the system, it will ask thresholds for that unit. The last factor that the user will need to specify is the policy for the *Gridlets* execution (deadline / cheapest possible). To help the user, the application may display some information about the overall state of the system.

Afterwards, the *consumer* module will wrap each *Gridlet* in a message with the information specified and inserts the messages into the system, so that the *Gridlets* are executed. Then it waits for the results. When the results are received, it is also received an invoice from the node that executed the *Gridlet*. (Figure 1 - nr. 1) Then, depending on the price asked and the validity of the result (if it is possible to check), the *consumer* module will pay the full or a partial amount of the price asked and classify the transaction (Figure 1 - nr. 3).

The *contributor* module will have two main functionalities: making the user resources available and the execution of the *Gridlets* inserted by the *consumer* module into the system. For the first functionality, the user must specify when and which resources are going to be made available and how they are going to be priced. That information is necessary because prices will be defined automatically, depending on the demand and supply. Then, according to the characteristics of the resources made available (including the price) the node will register itself in the indexing overlay.

The indexing overlay is the mechanism used to discover the node that best matches the requirements of the *Gridlet*. This mechanism uses a hierarchic approach with two classes of nodes, *brokers* (similar to super peers) and ordinary nodes. All the nodes will be distributed over a Chord ring, which will be used for communication. In the Chord ring, all of the nodes will work in the same way, but the *brokers* will be the successor node of a number of predefined keys (e.g. 1000, 2000, etc.), so that they can be easily found.

The *brokers* will act as an index of the ordinary nodes and store a pair with their characteristics (CPU speed, network bandwidth, price, etc.) and the identifier in the Chord. The characteristics will have to include the values of the unit of cost, since those are the ones used to determine the node position in the index, and might also include others such as OS or applications installed. In order to distribute the index among them, the brokers will be organized in a d-dimensional CAN, where each dimension corresponds to a parameter of the defined unit of cost (CPU speed, network bandwidth, price, etc.). Then each *broker* will have the part of the index which corresponds to the characteristics that match the zone for which it is responsible. This means that the node responsible for the zone between 1950 and 2050 in the dimension of the CPU speed will have the index of the ordinary nodes that have the CPU speed in that range. However, the values of the capacity of a computer normally are round numbers; there are CPU's with 2000 Hz but not with 2005 or 2010 Hz, which would mean that some zones would be full and others empty. To prevent this from happening, when a node is inserted into the index a random offset is added to its characteristics. That offset will be small, below a predefined quantum (e.g., 200 Mhz), so that it does not have a major impact on its location in the index, but large enough to spread the nodes across the index. The entries on the index will also have a lease, so when the lease expires is the corresponding node responsibility to re-register itself in the index.

So, when the *consumer* module inserts a message, containing the *Gridlet* and the requirements for its execution, into the system, it will actually send it to the closest *broker* in the Chord. Then, the CAN will be used to route the message to the *broker* that is responsible for the zone that matches the ideal values of the unit of cost (Figure 2). If a node that fulfills all the *Gridlet* requirements is present in the index of that

broker the message is sent to be executed to the selected node. Otherwise a search is made through the neighbors of the broker, which zones are within the threshold specified in the message, using the expanding ring technique. As it happens with the node characteristics, the values searched will normally be a round number too; so, to prevent the routing in the CAN to be directed at the same set of nodes (the ones that matches a round number) an offset will also be added to the search characteristics.

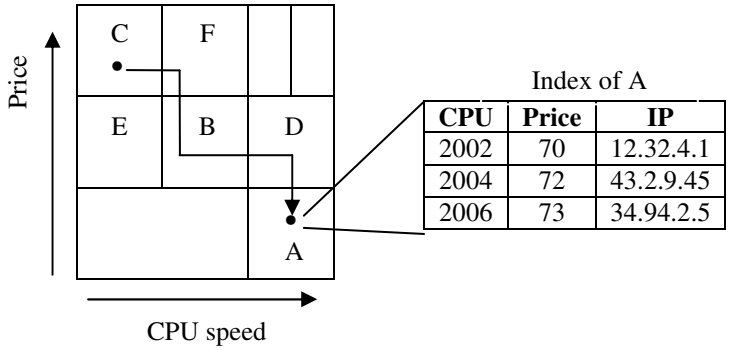

Figure 2 - example of the routing of a message with a unit of cost with the following ideal values: CPU - 2003; Price - 71.

Once the *Gridlet* reaches its destination, if the resources are available, it is immediately executed. If the resources are busy, the module will either put the *Gridlet* in a queue or send it to be executed in another node. This decision will be made based on the state of the system at that moment, the policy specified in the message and the user class. After the *Gridlet* have been executed, the result and the invoice are sent to the source (Figure 1 - nr. 2). Optionally the result can be stored in the Chord and the invoice will have a reference to where it is cached. The invoice will include the amount to be paid to the node that executed the *Gridlet*, a percentage to the *brokers* involved in the routing process and, in the case where the result is cached, a percentage to the node or nodes that stored it. When the payment is received the *contributer* module classifies the node that inserted the *Gridlet* into the system (Figure 1 - nr. 4).

All the operations of the *consumer* and *contributor* modules are going to be executed automatically, the user interaction will only be necessary to define the initial specification (policies, price, threshold, etc.) and submit the job. Also, both modules will execute as a single application in user machine.

4.1 Components to be built

The following components will have to be implemented in order for the *consumer* and *contributor* modules to function (Figure 3):

- The *Controller* component is the main component of the *consumer* module. This component will receive the *Gridlets*, from the layer that creates them, wrap them in messages and insert the messages into the system. It will also receive the results,

decide how much will be paid to the nodes in the invoice and give them a classification.

- The *Market Analysis* component will analyze the usage of the system and the prices being asked by the other nodes of the system. It will do a statistical analysis of that information (correlation of the price with the characteristics or reputation of the peer) in order to make it useful. The information supplied by this component will be mainly used by the *contributor* module to define the price of the resources.
- The *Advertising* component is responsible for making the user resources available to the system. It will receive from the user the information of when and which resources are made available, define the price of those resources and register itself in the index of the *brokers*. The prices will be defined based on the user specification and the information supplied by the *Market Analysis* component.

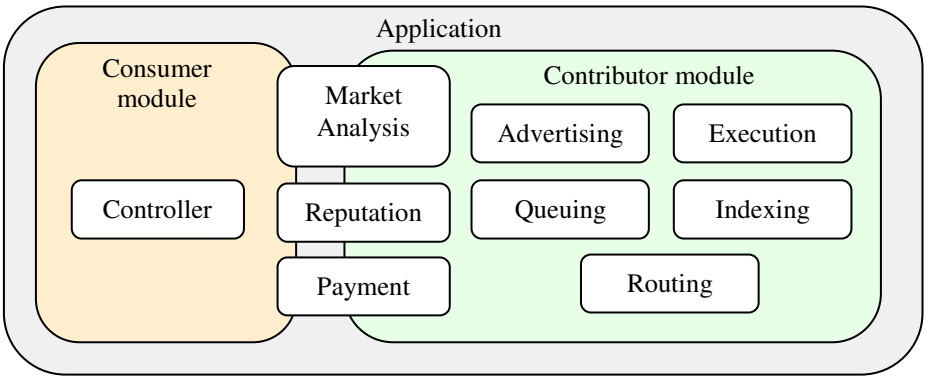

Figure 3 – Component architecture of the economic model implementation

- The *Execution* component main responsibility will be to execute the *Gridlets* received. It will also monitor how much of the resources were used, calculate the value to be charged and send the invoice. After the payment has been received this component will classify the node which submitted the *Gridlet*.
- The *Queuing* component is only necessary when one or more *Gridlets* arrive to be executed and the resources are busy, so it will probably just function in times of great demand. The *Queuing* component is responsible for deciding if the *Gridlet* is queued or sent to another node to be executed. It will also decide the order of execution of the *Gridlets* in the queue, based on the reputation or class of the node that submitted them.
- The *Routing* component will be responsible for routing of the messages. This component will be responsible for routing the messages in the Chord ring and in the CAN overlay. It will also do the necessary communication to stay in those DHTs.
- The *Indexing* component will be used only by nodes that are brokers. This component will be responsible for maintaining the index of nodes in the system and selecting the node that best fulfills the values defined in the unit of cost.

- The *Payment* component is responsible for storing the user credits and managing the transactions. This component will not be the main focus of this work, considering the possibility of being an interface that makes the bridge to another payment system. Only the basic functionalities will be built.
- The *Reputation* component is responsible for collecting the nodes classification, store it and treat that information so that can be used for comparing the peers. It will also store other relevant information, such as the class of the node or number *Gridlets* that it executed. This component will also not be the main focus of this work, considering the possibility of being an interface that makes the bridge to another reputation system. Only the basic functionalities will be built.

5 Evaluation

The evaluation of the economic model will focus on how well the jobs are mapped into the available resources. To do that, measurements will be made on how much the user is satisfied with the system performance, by using the two following metrics:

- The first metric is the time that it takes since a job is submitted until the result of the execution is collected, compared against the time it takes the local execution
- The second metric is how well the requirements of the job execution are met. Each job will have a threshold of requirements in which can be executed, this metric will evaluate how far from the ideal requirements the jobs were executed.

The first evaluation will start with more resources than jobs and analyze how the model reacts to a scenario where the number of jobs grows and overcomes the resources available. In this scenario the nodes that have jobs to be submitted will start with enough credits to execute them.

The second evaluation will consider the same metrics and scenario, but will consider peers that make different levels of contribution to the system. The results will be analyzed by classes of users and compare the different levels of satisfaction according to their levels contribution.

The third evaluation will analyze how well the model is able to satisfy the user policy by measuring how many credits are spent or how well the deadlines are met. This evaluation will consider the same scenario as the second evaluation.

The next point is to evaluate which factors for calculating the prices are more effective, different aspects and weight will be considered to calculate the value asked for the job execution. The metric used for comparison will be the credits accumulated in the end. For this evaluation we will use the scenario of the first evaluation, but with a constant number of jobs.

The last point to be evaluated is the impact of malicious users on the system. For this scenario there will be inserted peers that commit *fraud* and others that will practice *overpricing*. While doing this, the malicious peers will submit jobs. The first two

metrics will be used to evaluate the success of the system in isolating this type of nodes.

6 Conclusions

The growth of the Internet and the increasingly higher number of idle resources held by home users enabled the creation of peer-to-peer networks to use those resources. These types of system are very flexible, but do not have any central authority that is responsible for the management of those resources. In this paper it is proposed an economic model capable of that management in a cycle-sharing environment.

For the elaboration of that model we have surveyed the characteristics of economic models in the real world and some examples of computer systems that use economic models for the management of their resources. It also was made an analysis of the existing reputation systems and resource discovery mechanisms in peer-to-peer networks.

In the model proposed the user will be able to contribute with resources to the system and submit jobs that will be executed on the resources available on other nodes. The nodes will be distributed over a DHT according to the characteristics of the resources made available. When a job is submitted, the economic model will select where it is going to be executed. It is expected that the model provides an efficient management of the resources and maximizes the objective of both the users that contribute and use the system. The metrics that will be used in simulation to assess whether those expectation are held are also presented in this paper.

References

1. D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. Future Generation Computer Systems, 18(8):1061–1074, 2002.
2. K.G. Anagnostakis and M.B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing, in the Proceedings of the 24th International Conference on Distributed Computing (ICDCS04), 2004.
3. 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):61, 2002.
4. N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on cooperation in bittorrent communities. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, page 115. ACM, 2005.
5. S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys (CSUR), 36(4):371, 2004.
6. A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In Proc. Second IEEE Int'l Conf. on Peer to Peer Computing, 2002.
7. C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in P2P systems. In Proceedings of the 3rd International Conference on Peer-to-Peer Computing, page 48. IEEE Computer Society, 2003.

8. R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic models for management of resources in grid computing. In Technical Track on Commercial Applications for High-Performance Computing, SPIE International Symposium on The Convergence of Information Technologies and Communications (ITCom 2001), 2001.
9. R. Buyya and S. Vazhkudai. Compute power market: Towards a marketoriented grid. In The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001), 2000.
10. L.G.L.M. Carlo, E.T.O. Felipe, and MG França. The Use of Reciprocal Trade as a Model of Sharing Resources in P2P Networks. In Proceedings of the 2009 Fifth International Conference on Networking and Services-Volume 00, pages 91–96. IEEE Computer Society Washington, DC, USA, 2009.
11. A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, page 132. ACM, 2005.
12. Y. Chou, C. Lee, and J. Chung. Understanding m-commerce payment systems through the analytic hierarchy process. Journal of Business Research, 57(12):1423–1430, 2004.
13. B. Chun, Y. Fu, and A. Vahdat. Bootstrapping a distributed computational economy with peer-to-peer bartering. In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, 2003.
14. B.N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D.C. Parkes, J. Shneidman, A.C. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors, pages 19–28, 2005.
15. C. Dellarocas. Analyzing the economic efficiency of eBay-like online reputation reporting mechanisms. In Proceedings of the 3rd ACM Conference on Electronic Commerce, pages 171–179. ACM New York, NY, USA, 2001.
16. M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, pages 144–152. ACM New York, NY, USA, 2003.
17. R. Gupta and A.K. Somani. Compup2p: An architecture for sharing of computing resources in peer-to-peer networks with selfish nodes. In Second workshop on the Economics of Peer-to-Peer systems, 2004.
18. A. Iamnitchi, I.T. Foster, A Peer-to-Peer approach to resource location in Grid environments, in: J. Weglarz, J. Nabrzyski, J. Schopf, M. Stroinski (Eds.), Grid Resource Management, Kluwer, 2003.
19. D. Irwin, J. Chase, L. Grit, and A. Yumerefendi. Self-recharging virtual currency. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, page 98. ACM, 2005.
20. R. Jurca and B. Faltings. Reputation-based pricing of P2P services. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, pages 144–149. ACM New York, NY, USA, 2005.
21. S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In Proceedings of the 12th international conference on World Wide Web, pages 640–651. ACM New York, NY, USA, 2003.
22. N. Liebau, V. Darlagiannis, A. Mauthe, and R. Steinmetz. Token-based accounting for p2p-systems. In Proceeding of Kommunikation in Verteilten Systemen KiVS, pages 16–28. Springer, 2005.
23. 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.
24. R.J. Mann. Regulating Internet Payment Intermediaries. In Proceedings of the 5th international conference on Electronic commerce, pages 376–386. ACM New York, NY, USA, 2003.

25. S. Marti and H. Garcia-Molina. Limited reputation sharing in P2P systems. In Proceedings of the 5th ACM conference on Electronic commerce, pages 91–101. ACM New York, NY, USA, 2004.
26. C. Ng, P. Buonadonna, B.N. Chun, A.C. Snoeren, and A. Vahdat. Addressing strategic behavior in a deployed microeconomic resource allocator. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, page 104. ACM, 2005.
27. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, page 172. ACM, 2001.
28. D. Rolli, M. Conrad, D. Neumann, and C. Sorge. An asynchronous and secure ascending peer-to-peer auction. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, page 110. ACM, 2005.
29. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), volume 11, pages 329–350, 2001.
30. C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. In Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, page 226, 2003.
31. M.R. Shirts, C.D. Snow, E.J. Sorin, and B. Zagrovic. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. Biopolymers, 68:91–109, 2003.
32. D. Spence and T. Harris. Xenosearch: Distributed resource discovery in the xenoserver open platform. In Proceedings of HPDC, pages 216–225. IEEE Computer Society, 2003.
33. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, page 160. ACM, 2001.
34. D.A. Turner and K.W. Ross. A lightweight currency paradigm for the P2P resource market. In Proc. 7th International Conference on Electronic Commerce Research. Citeseer, 2004.
35. 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.
36. V. Vishnumurthy, S. Chandrakumar, and E.G. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In Workshop on Economics of Peer-to-Peer Systems. Citeseer, 2003.
37. R. Wang. Bargaining versus posted-price selling. European Economic Review, 39(9):1747–1764, 1995.
38. L. Xiong and L. Liu. Building trust in decentralized peer-to-peer electronic communities. In Fifth International Conference on Electronic Commerce Research (ICECR-5). Citeseer, 2002.
39. B. Yu and M.P. Singh. Incentive mechanisms for peer-to-peer systems. In Agents and peer-to-peer computing: first international workshop, AP2PC 2002, Bologna, Italy, July 15, 2002: revised and invited papers, page 77. Springer-Verlag New York Inc, 2003.

Appendix A: Work plan

Task	Description	Duration
Routing	Implementation of the mechanism of routing in both DTHs.	2 weeks
Index	Implementation of the index, the mechanism for registration and search.	2 weeks
Simple economic model	Implementation of the basic economic model with fixed prices.	3 weeks
Dynamic prices	Design and implementation of the mechanism for automatic price definition.	4 weeks
Reputation	Design and implementation of the automatic classification mechanism and the division in classes of users.	4 weeks
Policies	Optimization of the node selection to comply with policy that the user specified.	2 week
Evaluation	Evaluation of the system in a simulation.	4 weeks
Report	Writing the report	3 weeks