INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Incentive Mechanisms in Peer-to-Peer Networks

Pedro da Silva Dias Rodrigues

Dissertação para a obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente: Professor Doutor Nuno João Neves Mamede

Orientador: Professor Doutor Carlos Nuno da Cruz Ribeiro

Co-Orientador: Professor Doutor Luís Manuel Antunes Veiga

Vogal: Professor Doutor Pedro Miguel dos Santos Alves Madeira Adão

October 2009

Acknowledgements

In order to accomplish the objectives defined for this thesis, the contributions from Prof. Carlos Ribeiro and Prof. Luís Veiga were crucial, pointing my work in the correct direction and providing valuable insight with their expertise.

I must also mention all the professors I encountered during my years at IST, each one contributed to the knowledge I acquired throughout my academic path.

I express my gratitude to Engº José Isabelino Coelho for his trust and for making possible the coexistence of my academic work with the start of a professional career.

I learnt a lot from colleagues I worked and studied with, namely António José da Silva, Daniel Correia, João Pereira, Nuno Correia and Sérgio Silva.

All my friends are important to me, but I specially greet Catarina Venâncio, João Assunção, Luís Duarte, Nuno Ferreira, Pedro Samorrinha and Sérgio Serra.

For all the patience, friendship and love, I admire my girlfriend, Cristiana Filipa Ferreira. You are the best!

Finally, I am grateful for my parents and their unconditional support in the course of my life, regarding all the choices I made.

Abstract

In the last few years, peer-to-peer systems became well known to the general public by allowing fast and simple exchange of resources between users. The complexity of these systems resides in the fact that each user can act both as a client and a server and, due to the absence of a central authority, the need for self-regulation. There is also the need to guarantee that every user contributes to the system as, if that isn't ensured, the performance of the system will decay due to the disparity between demand and offer of resources.

This paper describes our work developing incentive mechanisms, which enable the correct operation of peer-to-peer systems, imposing a balance between demand for resources and the existing offer. All the incentive mechanisms take into account the attacks these systems are subject to, as well as the structure of the system, so that they do not pose an unnecessary burden, slowing down the system excessively.

We explore concepts such as reputation and currency, which are used in other systems and, more importantly, in our everyday life, enabling a coherent scheme to detect untrustworthy users and reward truthful peers with faster access to the resources.

Our work is part of a larger project called GINGER, an acronym for Grid In a Non-Grid Environment, a peer-to-peer infrastructure intended to ease the sharing of computer resources between users.

Keywords

Peer-to-Peer System, Incentive Mechanisms, Reputation, Currency, Sybil Attack, Computational Puzzles

Resumo

Nos últimos anos, os sistemas peer-to-peer tornaram-se conhecidos do grande público por permitirem uma partilha de recursos entre utilizadores de forma rápida e simples. A complexidade destes sistemas é causada pelo facto de cada utilizador actuar como cliente e servidor e também pela necessidade de existir auto-regulação, já que não está presente qualquer entidade reguladora central. Caso não seja possível garantir que todos os utilizadores contribuem para o sistema, a diferença entre procura e oferta de recursos vai aumentando, levando a uma situação de ruptura.

Este documento descreve o nosso trabalho de desenvolvimento de mecanismos de incentivos para redes peer-to-peer, que criam um equilíbrio entre procura e oferta de recursos. Estes mecanismos levam em conta todas as preocupações de segurança necessárias neste tipo de sistemas, bem como o seu impacto na velocidade da rede, para que não se tornem um problema ainda maior do que o que pretendem resolver.

Exploramos conceitos como reputação e moeda, que são utilizados não só em outros sistemas semelhantes, mas também no nosso quotidiano, permitindo uma forma coerente de detectar utilizadores desleais e, por outro lado, premiar os que têm um comportamento correcto garantindo-lhes acesso rápido aos recursos.

O nosso trabalho está incluído no contexto de um projecto denominado GINGER, acrónimo do inglês “Grid In a Non-Grid Environment”, que pretende permitir uma infra-estrutura para facilitar a partilha de recursos computacionais entre utilizadores.

Palavras-chave:

Sistema *peer-to-peer*, Mecanismos de Incentivos, Reputação, Moeda, Ataque *Sybil*, Puzzles computacionais

Table of Contents

List of Figures

List of Tables

List of Acronyms

CPU – Central Processing Unit

DNS – Domain Name System

GINGER – Grid In a Non-Grid Environment

IP – Internet Protocol

P2P – Peer-to-Peer

SETI – Search for Extra-Terrestrial Intelligence

1

Introduction

The exchange of information between users has always been the main purpose of any computer network. While P2P systems have long been used in laboratories and company offices to share resources and information, they became more popular as the number of Internet users and the bandwidth available to them increased. The traditional Client-Server architecture could not offer the flexibility demanded and centralized servers became the major source of complications, not only due to poor performance but also as targets for attacks that could easily disrupt the correct functioning of several applications.

The possibility to communicate directly with other users, along with the increased performance, proved to be the key feature of P2P networks and, although they have been mainly associated with file exchange applications, the structure of the network allows the exchange of any resource between the users.

These complex systems have, until now, only been put into operation in controlled contexts, such as laboratories, mainly due to the difficulty to control user behavior. Our work contributes to the faster dissemination of these applications, allowing access to ordinary users, outside controlled environments.

Our main concern is related to the overall fairness of the system, guaranteeing that vicious users will not be able to corrupt the entire system, by not contributing to the resource pool or delivering false results. We developed an incentive schema taking into account these possible attacks to the system, already confirmed in other peer-to-peer networks, as well as every concern with the security of communications between users.

This research is encompassed in a larger project, GINGER, focused in grid computation on the Internet. However, the incentive mechanisms developed in this project can be used in complex systems, regardless of the resources being shared. The implementation of our work can be easily adapted to other applications, since it is developed on top of the Pastry overlay network [28], which can be used by other peer-to-peer applications.

1.1 Motivation

The reason we felt urgent the development of incentive mechanisms for peer-to-peer networks is mainly related to current fragilities of these applications. The intention to implement resource-sharing communities in highly demanding contexts calls for very strict policies in the network, and the problems verified in large file sharing communities have to be resolved.

A central server is always a target for attacks and cause for failures; this is why decentralization is vital. But decentralizing also expands the complexity of the system, making difficult the task of identifying users in order to prevent abuse and attacks. In controlled environments, it is possible to request user authentication via smartcards or biometric data because direct physical contact with the users exists. But in a scalable peer-to-peer network, users are geographically distributed and anonymous, hence the necessity to develop incentive mechanisms compatible with this network structure and capable of identifying attackers and prevent system collapse.

The most common attack in peer-to-peer networks is the Sybil attack [7], consisting in a single user deploying multiple virtual identities in order to obtain resources without ever contributing to the system. Another common attack is collusion, where two or more peers arrange resource exchanges between them in a way to create the illusion of fair sharing, to improve their reputation in the community. The presence of malicious users in the system can lead to total disruption, therefore the main concern must be to guarantee that these users are promptly detected and eliminated or otherwise rendered harmless.

1.2 Objectives

Our work aims to provide a solution capable of guaranteeing balance between resources demand and supply in the network, by ensuring that cooperating users are rewarded and malicious users punished. The resource sharing applications using our incentive mechanisms ensure that only cooperating users have access to the resource pool and attackers are excluded from the system. We will develop our solution on top of the Pastry overlay, so that it can then be adopted by all resource sharing applications also using Pastry.

Our solution will be tested using a simulator and we will assess the impact on performance to guarantee that it is minimal. High overhead is unacceptable as it most definitely means the incentive mechanisms would not be implemented, no matter how reliable they are.

We are confident that, with the implementation of these incentive mechanisms, the dissemination of applications for sharing computational resources is possible; allowing the access to ordinary users, with all the benefits these applications can accomplish.

1.3 Organization

The central focus of this proposal is the design of effective incentive mechanisms for peer-to-peer networks. In particular, we intend to prevent abuse from malicious users, ensuring that each peer must contribute with resources to the community, in order to have access to the pool of resources.

Throughout this document, we detail the most important aspects of peer-to-peer applications, describe well-known attacks and introduce our solution.

The document is organized according to the following scheme:

- Peer-to-Peer Architecture – In this chapter we present an explanation of the main features of P2P systems and compare them with the Client-Server architecture. We present an overview of the Pastry framework over which our work is developed, list and analyze successful systems with different approaches to the P2P philosophy and describe the PeerSim simulator, used to assess the effectiveness of the incentive mechanisms.
- Related Work – Here we present a description of current research in this field, improvements made to P2P systems throughout the last years the opinion of several authors about the most important aspects of resource sharing applications, namely reputation and currency.
- System Design – This chapter contains information about the development of the incentive mechanisms, the solutions we designed, our main concerns and the most important features.
- Implementation Details – Here we provide specific information regarding application parameters and interaction between the different components.
- Simulation and Evaluation – In this chapter we analyze the effectiveness of the implementation and report our findings, the impact of the incentive mechanisms on performance and the effectiveness of the solution developed.
- Conclusions & Future Work – In the final chapter we make an analysis from a critical point of view, mentioning the main features but also possible adjustments to improve the system.

2

Peer-to-Peer Architecture

The characteristic that defines the P2P architecture is the ability the participants have to communicate with each other and share resources between them, acting as clients and servers, without the necessity for a central coordination entity. These entities can be integrated to maximize the performance and make possible a structure, which can be desirable when the number of members is very high.

Several dimensions of Peer-to-Peer systems can be classified and taken into account. All the aspects considered have an impact on future development and usability of the system. The purpose determines most of the system's characteristics, along with security and integrity concerns, but certain aspects deserve a careful plan, since a small decision can turn into a big advantage or disadvantage in terms of performance and scalability.

A number of incentive mechanisms have been developed, according to immediate preoccupations and specific objectives. A categorization of reputation systems is presented in [21], and identifies three main components in these systems, namely: 1) information gathering, 2) scoring and ranking, and 3) response. We believe that decomposing the system will improve the final quality by allowing us to focus in each aspect and agree with the categories suggested.

The main issues when considering incentive mechanisms for peer-to-peer networks, according to [15], are:

- Self-policing, which means that there is no central authority and peers themselves must enforce the existing policies;
- Anonymity of users;
- No profit for newcomers, so that users are not encouraged to change identity;
- Minimal overhead to the application;
- Robustness to malicious users, even if they collude to achieve advantage over other users.

Any incentive mechanism has to fulfill these requirements in order to be effective.

2.1 Comparison with a Client-Server Architecture

Traditional computer systems are hierarchical, with different levels of responsibility. This happens, for instance, with the DNS used to translate domain names into IP addresses. This approach is perhaps the most evident, since almost all of our society is structured hierarchically, from Governs to Companies. The key decisions are made at the top of the pyramid, where the responsibility lies. This provides trust and accountability, but results in a bottleneck and single point of failure, meaning that if something happens to the entity responsible for the decisions, the whole system halts.

Peer-to-peer systems have tried to minimize this risk by enabling each user to make all necessary decisions and, by eliminating the central point of failure, guaranteeing that the system remains in operation, even if most of the nodes fail. This flexibility reflects also in the scalability of the system, allowing the users to communicate directly with each other without the need of powerful servers to route the messages and control active users.

The downside is obvious: without a central authority, trust becomes a fundamental problem. This “trust no one” environment is harmful, especially when we are developing a system to share resources, and so, unless trust issues can be worked out, users will not be inclined to share their resources and the system collapses. One of the deciding factors in deciding if users should be trusted is the reputation of those users, i.e., information about their past behavior.

2.2 Pastry Overlay Network

The Pastry overlay [28] provides the basis for the development of various peer-to-peer systems, with purposes ranging from file sharing to naming systems.

The network is structured as a ring of nodes, each node in Pastry having a unique 128-bit numeric identifier, known as nodeID, keeping a list of its neighbors and communicating the arrival or departure of nodes to the application running on the Pastry overlay. When a node receives a message and the nodeID of the target node, it routes that message efficiently to the node with a nodeID closest to the target. The expected number of hops is O(log N), where N is the number of nodes in the network.

An example of the routing process is illustrated in Figure 1.

The nodeID is assigned randomly when a node joins the network. Besides the neighbor list, each node also maintains a routing table with the IP addresses of nodes with the same nodeID prefix. Using a proximity metric, such as the number of IP routing hops in the Internet, it is possible to improve the message’s route.

To enter the system, a new node needs only to know the address of a Pastry node, to which it sends a “join” message, becoming its neighbor. The new node initializes its routing and neighbor tables and requests more information about other nodes, increasing its knowledge of the network and contacting nodes to announce the new arrival.

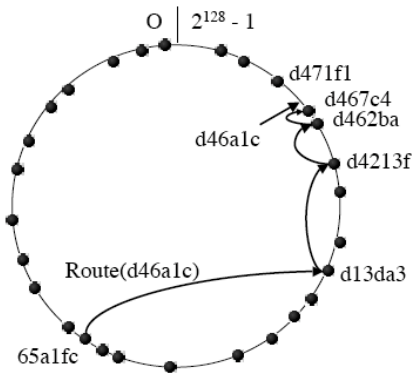

Figure 1 – Pastry network and the route of a message

The failing or unannounced departure of nodes in the Pastry network is also easily detected and treated. A node is said to have departed when its neighbors can no longer contact it. At that point, a new node must be inserted into the routing table to replace the departed node and to maintain the table's integrity. To obtain a new node, a request is made to a random neighbor. That neighbor sends its neighbors table and a node, which is not already in the routing table, is selected as a replacement.

To cope with malicious nodes, Pastry uses a random route to deliver a message to its destiny, instead of a deterministic routing scheme. With this mechanism, a series of messages sent from node A to node B would almost certainly use different routes, making it impossible for a malicious node to intercept those messages.

2.3 Successful Peer-to-Peer Systems

The first decade of this century saw the growth of several applications using P2P networks, and the birth of the most famous ones. We take a closer look upon three emblematic P2P applications.

One of the most well-known protocols today, BitTorrent, responsible for more than 30% of all Internet traffic and as much as 70% of all P2P traffic in 2004, according to an extensive internet traffic study [23].

KaZaA was also a well-accepted application, used by millions around the world, with a different approach regarding the system architecture, based on a two-level hierarchy model, we find particularly interesting.

SETI@Home is the oldest of the three and the only one focused on sharing computational resources, namely CPU cycles, since both BitTorrent and KaZaA are devoted solely to file sharing.

2.3.1 BitTorrent

Designed and implemented in 2001, the BitTorrent protocol [2] is used to distribute large amount of data without requiring enormous resources from the hosts. The principle is quite simple and uses viral propagation to distribute files in a large network. The first user makes a file available to the network so that anyone can download it and, as multiple users obtain parts of that file, they can start to share them, removing the burden from the original provider. The more users with that file available, the easier and faster it gets to download the file, as the number of sources reduces the saturation of the network.

To address the problem of users who only downloaded content, without contributing to the sharing community, BitTorrent deployed basic incentive mechanisms to encourage resource sharing. However, these basic procedures, such as tit-for-tat [6], have proven to be fragile and inefficient. For instance, in [30] authors demonstrate that the Large View Exploit is effective against the incentive mechanisms used in BitTorrent. The exploit consists in connecting to a large number of peers instead of being restricted to the default swarm size. This way the user increases the chances of being unchoked and discovering seeders, without ever contributing to the system. The authors of [25] even state that the only reason BitTorrent performs well is because most people use the default settings of client software, since, as this paper shows, it is extremely easy to improve performance while reducing contributions to the system with just minor modifications in the settings, namely carefully selecting peers as well as contribution rates.

In [1] authors explore ways to improve fairness in BitTorrent by making simple changes to the tracker and tit-for-tat policy. They conclude that BitTorrent's rate-based tit-for-tat policy is ineffective but it is possible to achieve fairness using a combination of pairwise block-level tit-for-tat and a bandwidth-matching tracker. The behavior of the seed nodes and Local Rarest First policy are also included in the solution to improve performance.

On the other hand, some authors have sustained that simple algorithms, with low impact on system performance, are the best solution. In [18] authors support that Rarest First and Choke algorithms are effective and that there is no need for complex policies such as bit level tit-for-tat. Nevertheless, we find that a complex economy-based peer-to-peer application will always demand a more sophisticated incentive system.

2.3.2 KaZaA

Also developed in 2001, KaZaA [16] represents one of the most successful peer-to-peer applications ever, considering both the number of participating users and traffic volume. The main difference between the BitTorrent protocol and the FastTrack protocol used in KaZaA is that not all peers are equal; there are ordinary nodes (ON) and super nodes (SN), with different responsibilities [20]. Super Nodes are not dedicated servers and the lifetime of a SN is no longer than a few hours, avoiding potential problems related to single points of failure. The idea is that nodes with higher resources can act as small servers to organize ON in groups and compile information about the resources being shared by ON assigned to them.

When ON join the network, they upload the metadata of shared files to their assigned SN, which then acts as a small hub. Ordinary users then query their SN to obtain about information about the resources they need and, if necessary, the SN relays the query to other Super Nodes. It could be decided to have super nodes know not only the resources shared by their group of ordinary nodes, but by all the ordinary nodes in the network. However, this would impact the performance of super nodes, which are not dedicated servers as we have pointed out, and slow down the system.

KaZaA benefits from this architectural approach and shows that, even without central points of failure, structuring peer-to-peer networks is possible and can have a positive impact on reliability and performance, improving scalability.

2.3.3 SETI@Home

Launched by the University of California in 1999, SETI@Home [29] is probably the most recognized application for P2P distributed computing, with over 5.2 million participants worldwide. In order to find intelligence outside our planet based on electromagnetic transmissions analysis, scientists collect large amounts of data, which require large computational power to process. Not having the financial capability to invest in super computers, and knowing the number of people interested in this area, the Space Sciences Laboratory developed this application so that anyone connected to the Internet could donate spare CPU cycles from their personal computers to process data and then convey that information back to the servers.

Since the resources were being shared with no kind of payment and the users did not expect anything in return, incentive mechanisms were not necessary. However, to guarantee the accuracy of the computations made, the same tranches of data were dispatched to multiple users and the results compared among them.

2.4 Peersim

The Peersim [24] application is a complex simulator, capable of reproducing the operation of a large P2P network. Since a large-scale P2P network can have several thousand users exchanging resources simultaneously, as well as constant arrival and departure of peers, to evaluate the effectiveness of our incentive mechanisms we have to simulate, as accurately as possible, the behavior of a P2P network, with random actions from the peers. This is where Peersim comes in, allowing for extensive testing to assess the dependability of the developed mechanisms.

The Peersim simulator supports multiple configurations, the most important ones being the number of nodes in the network, the protocols used and the controls to monitor the state of the network. Varying the parameters, we can simulate the presence of malicious nodes in the network and evaluate the effectiveness of the incentive mechanisms.

Each simulation starts loading the prototype node, the first super node, and then continues to perform join requests and all other actions, making decisions according to the specified parameters. The major components of the simulator are:

- The simulator engine, responsible by invoking every other component in the correct sequence;
- The network, where the information about all nodes is stored;
- Nodes, with a common implementation and operation.

All the parameters are passed to the simulator engine in the beginning of the execution and then passed on to the nodes as instructed.

We built our incentive mechanisms on top of an implementation of the Pastry Protocol for Peersim maintained by Manuel Cortella and Elisa Bisoffi and available at the Peersim website.

3

Related Work

3.1 Making Resources Available

In a system designed to assist the exchange of resources between users, the manner in which the resources are made available and to whom they are available is of the utmost importance.

When an application is used only by a restricted number of people and the demand for resources is not greater than the supply, it is possible to satisfy every request. However, when the demand escalates, there is a necessity to prioritize requests and the efficiency of the system relies deeply on the quality of the solution.

3.1.1 Virtual Machines

One of the most straightforward manners to share computational resources is presented in [12]. The authors recommend a system based on the operation of a Laundromat, called Evil Man. The requesting party is responsible for providing all the information as a Virtual Machine, which is then loaded by the provider. The interaction between users is very simple, with low impact on performance. Every time a request is fulfilled currency changes hands, from the requesting party to the service provider, therefore obliging every peer to share resources in order to obtain the currency it needs to pay for services. Although this model is undoubtedly efficient, it does not account for previous interactions, with no information being given about the quality of the service a user provides.

3.1.2 Overbooking and Queues

A way to deal with resource allocation, considering overbooking and queues, is explained in [9]. When the demand for resources surpasses the offer, peers will have to wait to obtain resources. Since we are mostly considering computational resources and not specific content, it is not crucial for a peer to obtain the resources from one determinate user. Therefore it will have the tendency to ask for resources from multiple peers, guaranteeing that the waiting period is as narrow as possible. SHARP offers a solution to cope with this situation, distinguishing between claims and leases. A claim simply represents the request made by a peer, while a lease corresponds to a promise to make a resource available. Peers are then evaluated according to their actions, seeing that they have to comply with the promises made to other peers.

3.2 Currency and Reputation

3.2.1 Currency

As all devices man has developed throughout centuries of history, computers are as complex as a person can master their use. Economic systems have become more and more dense and what was a simple evolution from bartering has become what we now know as currency, with all the concepts involved, such as inflation, deflation, interest and exchange rates.

The economy in a resource exchange system can be as complex as we want it to be. We can have a barter economy, where a user asks for a resource from another user and gives a resource he owns in return, or we can go as far as various types of currencies, each one with a purpose and exchange rates between them.

In a barter economy resources are a form of currency and, when peers exchange resources between them, the trade is mutually beneficial hence there is no need for other types of payment. However this would mean that users could only consume resources is they had something the other party needed in that precise moment. As we know, the demand fluctuation renders this option ineffective, posing unnecessary limitations to the sharing community. Introducing currency translates into users being able to share resources even when they do not require any in return, accumulating currency to use when needed.

Some argue that there should be a payment in every exchange, and that payment should have nothing to do with the reputation measure we have talked about before. Another approach is to consider only reputation – good and bad members of the network. Good members should have access to the resources and pay nothing for them, except the obligation to participate in the reputation system, praising the member who shared his resources.

Then another aspect can be brought up: should every resource have the same value? Some resources are scarcer than others, and there are “rush hours”, where the resource pool is insufficient. This is the difference between a token economy, where a token equals a resource, and a currency economy, where the value of a resource fluctuates.

3.2.2 Reputation

As we have pointed out, knowing which users you can trust is crucial. To address this problem many solutions have been proposed, but the most used is, by far, the reputation of the user. The reputation of an entity can be described as the result of the trust that every peer places on that entity.

EBay [8] is probably the most notorious example. The system works as follows: when you buy a product from another user you are asked to classify the seller according to the quality of the product, communication, packaging and delivery time. If a user gets good reviews, it adds to his reputation and he becomes a renowned seller, attracting more potential buyers.

The major problem is convincing users to cooperate, by accurately reporting the result of their resource exchanges. Without any incentive mechanisms, the user has no advantage in cooperating

[9]. By reporting positive ratings, it creates advantages to other peers, who increase their reputation and, by reporting fake negative ratings, the user increases its reputation comparatively to other users.

In [34], the authors present a reputation system intended for peer-to-peer file sharing systems, Credence, which can be adapted to other types of resources. The main purpose of the system presented is to avoid contaminating the network with mislabeled or dangerous content, such as viruses, by encouraging peers to classify the resources obtained from other peers. This is accomplished by linking the reputation of every user not only to the quality of the resources it shares, but also to quality of its evaluation on other peers' resources. Using a system like Credence, it is possible to motivate users to share their insight on other peers and distinguishing untrustworthy users.

With a solid reputation mechanism it is possible to distinguish good service providers from bad. However, assembling this mechanism is far more complicated than it seems, and we will talk about the most prominent problems ahead.

3.2.3 Reputation vs. Currency

As we have seen in the last sections, reputation and currency are different concepts that can be used separately or combined. A single representation is a common approach, as can be seen in KARMA [33], which uses a single scalar value that can be seen as the reputation of a peer but also the currency used to pay for resources. The KARMA of each peer is incremented or decreased every time a transaction occurs, constituting the payment for that transaction, but the amount of KARMA users possess is also an indicator for their reputation.

This approach, also the basis for the Eigentrust algorithm [15], is simpler and can very well be effective in most environments, but is ultimately too shallow for more complex applications, since it cannot distinguish the value of the resource nor the efficiency demonstrated in the sharing process.

If we think of eBay, where currency and reputation are used as separate concepts, we can take into account several aspects when we are choosing a seller, for instance comparing the prices for the product with other sellers and knowing if the seller has positive feedback from other transactions.

In [14] authors focus on the problem of trust and support the existence of reputation agents (R-agents), from whom users can buy information about the reputation of other users and, after they have participated in resource exchanges, they can also sell information they have to the same R-agents. The currency used to pay reputation agents is different from the one used to pay other peers for resources, so that users are compelled to participate with information gathered from their experience, in order to use the reputation system.

Even if the system consists only of one kind of resources and a fixed value is used for every transaction, an adequate reputation representation can distinguish the quality of the resources provided. As an example we can think of computational resources sharing, and the difference between sharing a single core processor or a quad core processor, admitting that the resource is shared in time slots, which is the most obvious.

3.2.4 Reputation Distribution

After a solution to identify cooperative and trustful users has been achieved, a manner to propagate that information must be developed.

Once again, had we a central authority mechanism and it should be relatively easy, since every user could ask the “market regulator” which users he could trust. Still, since a P2P system must not be centralized, things get a little more complicated.

Pragmatically, a user can only trust itself and users he has interacted with. However, this is very limited, and the next step is to trust in users that are trusted by the users we trust, and so forth. This process is called Reputation Distribution or Trust Distribution and consists in each user having a set of trusted users and sharing it with others. If I have interacted with a user and I decide he is trustworthy, then maybe I should ask that user information about other users, expanding my trust network. However, this process has to be approached with caution, since my trust in another user cannot be total. Even if I have a positive association with another user, maybe he will fail in the future, and so the problem of how to distribute reputation comes up.

We can adopt two methods: either we prune the network, saying that we only accept a certain level of trust distribution or we weight that distribution.

The first option is more restrictive. For instance, we trust only the users we have had direct contact with and those user’s direct contacts. This only propagates trust one level, becoming a 2 level network tree.

The second option is a bit more inclusive, although it demands the users to weight each association. For instance, if I have several resource exchanges with a user and most of them are successful, I can say that I trust that user to a certain amount, say 90%. Then, when I ask that user for a list of trustworthy “business partners”, I must weight that list, taking into account that I only trust the source to 90%. This method allows a user 7 levels down the tree to be trusted almost fully, if the path is trustworthy, on contrary to the pruning option, while still being reluctant when classifying users, taking the distortion factor into account.

When we are considering a network of peers who classify each other by sharing their experience with previous transactions, we immediately come upon the need to introduce a method for reporting untrustworthy users – negative reputation. In [10], authors develop and test a framework for propagating trust and distrust. If we consider only positive reputation, a malicious user will have the chance to perform several faulty transactions before its reputation reflects those actions; it can be seen as a slow death. The upside is that a framework with only positive reputation has less traffic circulating between peers and the security issues are simpler, since there is no need to worry about false negative votes and coalitions of peers to expel fair users.

The framework presented in [10] changes the focus of trust from the number of voters to trust relationships developed over time, much like in real world, where you trust the opinion of a long acquaintance more than that of ten strangers. Then there is the question of how to model distrust in the system: should votes range from 0 to 10, for example, or should 0 be a neutral vote and introducing negative numbers instead as a solution?

The following trust propagation schemes are the most common and effective:

1. Direct propagation, the most straightforward propagation scheme, can be explained by a simple example: if John trusts Hannah and Hannah trusts Karl, then John also trusts Karl. If a node builds a trust relationship with a neighbor after multiple successful transactions, it may be inclined to ask the opinion of that neighbor about other peers, expanding the information stored about users in the network.

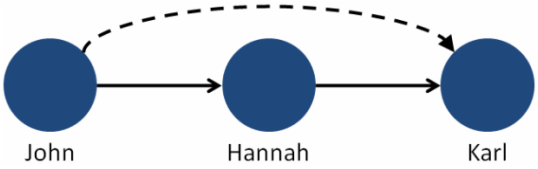

Figure 2 – Direct Trust Propagation

2. Co-citation means that, if John trusts Hannah and Karl, and Rudolph trusts Karl, then Rudolph also trusts Hannah. This trust propagation method is more inclusive and builds on current knowledge about the neighbors to infer a trust measure on unrecognized peers by trusting peers who have positive feedback about nodes already on the trust network. Analyzing the opinion of other nodes about peers we trust can be an effective way to determine the reliability of such nodes.

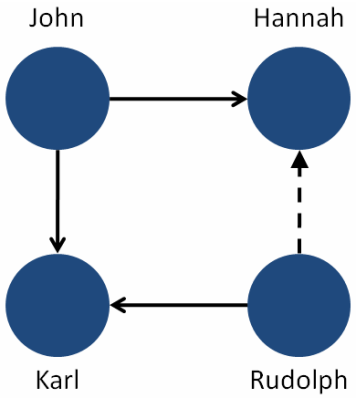

Figure 3 – Co-Citation Trust Propagation

3. Transpose trust is a mutual trust, if John trusts Hannah then Hannah also trusts John. This simple trust propagation method implies trusting in users who trust us, which can be unreliable since fraud is a strong possibility.

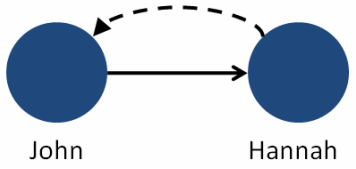

Figure 4 – Transpose Trust Propagation

4. Trust coupling is a propagation of trust by common ideas – if John and Hannah trust Karl, Rudolph trusts Karl, then, if Rudolph trusts John, then he also trusts Hannah.

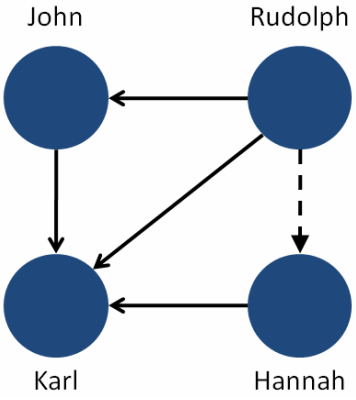

Figure 5 – Trust Coupling Propagation

Distrust can be propagated in the same manner as trust, or the system can opt for one-step distrust. This means that, when a peer does not trust another peer, he simply disregards those peer's opinions.

Results of the experiments with this framework show that co-citation performed quite well as a method for propagating trust and, as expected, the introduction of distrust was helpful, except when using only direct trust propagation. The authors recommend therefore using co-citation and one-step distrust for achieving better results.

As we have pointed out in the objectives of our work, the weight of the trust propagation mechanism must be taken into account when designing the system, so that its impact does not minimize the positive effects of having an incentive mechanism. Regarding this subject, the work in [26] presents a mechanism for lightweight and distributed trust propagation, with the intent of implementation in low capacity devices, such as mobile phones. The authors advocate that direct propagation is only meant for applications with centralized servers and does not scale on distributed environments; therefore the idea is that a user only stores a limited subset of the trust web and applies a machine learning technique for propagating trust as well as distrust.

3.3 Robustness and Security

When we think about the robustness of the system we intend to develop, we have to take into consideration all the limitations inherent to remote interaction between users. We have to establish identities for the users of the system, so that we can discriminate who has access to the resources and identify users who fail to comply with established rules.

The problem with electronic identities is that a physical user can create as many virtual identities as he wants. In a controlled environment, such as a company department, the number of nodes is static, and therefore the access can be limited to the known terminal nodes. User identification can be achieved at the terminal level, with a mandatory login. However, this cannot be applied in a distributed environment, where nodes are constantly entering and exiting the network and there is no central authority to authenticate users.

There are also other threats to consider besides user identification, such as collusion between users to obtain advantages over other users and bribery, where users pay for privileged access to resources. A cryptographic key infrastructure is essential to provide for accountability, since almost every security solution uses this method to authenticate users.

3.3.1 The Sybil Attack

The Sybil attack, as described by Douceur [7], consists in a single user operating multiple virtual identities to abuse his permission to access resources. With the ability to create multiple virtual identities, a user needs not to worry about sharing resources and reputation. When one of the identities is considered to have had poor behavior, the attacker simply uses another. If the system is permeable to Sybil attacks, it cannot attract good resources, and falls apart.

Various attacks against computer systems are conducted creating multiple virtual identities, classifying those attacks as Sybil attacks. We all suffer from this type of attacks every day, mostly spam. The low cost of creating virtual identities is the major incentive for attackers and another example are the denial of service attacks, where a large number of requests are made to a server, flooding it and degrading the performance. It is curious that the same kind of attack can be used against peer-to-peer systems, since a great motivation for these systems is avoiding the congestion and security problems of a central server.

Preventing Sybil attacks has been the motivation for countless works and [19] presents an interesting survey on how existing applications cope with the existence of Sybil attacks. It shows that, although the attack is prominent, several applications are completely vulnerable and did not adopt any security mechanisms.

In this section we focused on attacks where a single physical user creates multiple virtual identities and uses them at will, to avoid the necessity of contributing to the system with resources. In fact, Sybil attacks can be used in a more strategic manner, with extended presence of cooperating attackers. We distinguish this kind of complex attacks in the next section.

3.3.2 Collusion

Collusion is also a major problem and, as we have mentioned above, coalition between users may not involve multiple physical users, but just multiple virtual identities for a single physical user. This phenomenon can be described as a coalition between users to improve each other's reputation in the system, gaining the trust of other users and profiting from that situation. Since the system uses the reputation of users to determine access to resources, if the reputation of a user can be manipulated to unfaithfully reflect the behavior of that user, it will lead to unfair exchanges, where a user does not share as much as his reputation implies.

Mechanisms to detect collusion have to be effective, determining if a group of users is forging reputation to get ahead in resource queues and access resources.

In his PhD Thesis, Jesi [13] focused on techniques to avoid the possibility of system disruption by a group of coordinated attackers exploring frailties in the neighbor's list propagation methods. The example used by the author is a system where users gather neighbors for their own list from their neighbors' lists. If a group of cooperating attackers list only each other in their respective lists, and tamper the timestamps so that the list appears to be fresher, those lists are propagated in the network and legitimate users progressively replace their neighbors with links to the attackers. After a pre-determined amount of time attackers would leave the system and the remaining users would have no connections between them, resulting in a completely partitioned network.

In [22] authors develop a mechanism to prevent collusion based on peer auditing. The idea is that users are compelled to publish signed records of their actions, which can then be audited by other peers. However, the incentives in this paper are designed for a remote file storage application, where the sharing of resources is a lengthy process. On the contrary, if we are considering a system where shared resources are immediately consumed the incentives have to suffer some adjustments. The solution proposed also demands the implementation of anonymous communications to protect the identity of the auditors and preventing not only collusion but also bribery attacks. The incentive for users to perform audits on their peers is that the system, as a whole, benefits from cheaters being discovered and ejected.

3.3.3 Malicious Users

The robustness of the system can only be increased if selfish users, who do not contribute to the system, are detected and banned. We also have to consider untrustworthy users, who give the impression of sharing resources, while in fact they just introduce entropy in the system. A good example is the proliferation of fake content in the system, or responding with false results to requested calculations.

There are a number of situations where this strategy was used to cause degradation of Peer-to-Peer systems such as Napster and KaZaA, to name the most known. If the contents of a system become untrustworthy, users are less inclined to use that system. The quality of the resources being shared is as essential to the system as the behavior of the users who share them.

The detection of fake content is simpler when we consider file-sharing applications. In systems where the shared resource is computational power, the used method is to send the same data to be

processed by multiple peers and then comparing the results. The SETI@Home application uses this approach to guard against malicious actions that could compromise the ability to achieve the wanted results. However, due to the increased cost of replicating requests, peers may choose to trust the result, requiring verification only if the importance of the result is important enough.

Another important aspect, one that can also be used to measure the quality of service, is the time elapsed between the request and the actual response from the resource provider. However, in order to consider a maximum amount of time allowed for a request, it is necessary to have a central authority capable of providing accurate timing information.

In [36], authors describe the design and implementation of a solution to identify fraudulent behavior from peers in cycle sharing systems. This application focuses in situations where nodes make available only part of the promised resources, namely using only a fraction of the computational power promised.

3.3.4 Newcomers

Users joining the network have no past interactions to prove their commitment to the system and therefore their reputation information is non-existent. Two simple strategies can be followed, namely optimistically trust new peers or, on the other hand, refusing to interact with those peers. If the first strategy is chosen, the system is encouraging whitewashing attacks and users with a rather negative reputation would prefer to exit the network and rejoin with a clean slate. If the system has a pessimistic approach newcomers face great difficulties to start exchanging resources and building up reputation and are limited to interactions with other recently joined peers.

While a slow start can in fact be a solution and a guarantee that users really intend to contribute to the system, an adaptive solution based on the behavior of newcomers can prove to benefit the application [17]. The idea is to compile information on interactions with recently joined peers on a specific group, determining if the system has a high rate of whitewashing, to decide how to cope with new users. The information is constantly being updated by the broker, according to received reports, so that changes in the conduct of newcomers reflect on the policy adopted.

3.3.5 Computational Puzzles

There have been several solutions to deal with Sybil attacks, not only in peer-to-peer networks but also to combat spam and denial of service attacks. The most common solutions opt for artificially raising the computational cost of using the service, creating human time delays or imposing financial costs.

In the peer-to-peer context, the most adequate is the use of computational puzzles. If every user has to solve a puzzle in order to enter the network, or even to request a resource if that is the policy enforced, it creates a cost that not all attackers are comfortable with. The creation of multiple virtual identities would then have an associated cost so great that the profit from using the Sybil attack would not pay off.

The major difficulty in using computational puzzles is guaranteeing that attackers do not reuse puzzle solutions and, in [3], a fully decentralized scheme to continually distribute challenges is proposed. Other authors [35, 4] have suggested the use of a public source of random challenges, but it would create a central point of failure, reason because a decentralized scheme is far more suitable.

Halderman and Waters [11] present a framework for deriving challenges using data from various online sources. The main concerns are the freshness of the challenges, the capability of verification by multiple agents and resilience against attackers who try to inject their own input as part of the challenges. The policies for gathering data can be customized based on the goal of the application. In some contexts it may be more important to use fresh information, from newspaper sites for instance, and in others it may be more useful to use more stable information, such as stock price history. Users must then harvest data from multiple online sources, according to the application's policy and perform computation over that data. Verifiers can validate the challenges by checking just a random subset of the original sources. Authors present Sybil attacks as an example of problems this framework can mitigate, along with remote storage and auditing applications.

The complexity of the puzzles has to be the result of careful consideration, as it is the vital condition for the successful protection against Sybil attacks. Puzzles must be difficult to solve, requiring large amount of computation, but easy to verify. As we have said, puzzles must incorporate constantly modified challenges, to avoid the possibility of precomputation. If the puzzle challenge is constant, e.g. generating a pair of public/private keys according to a certain pattern, an attacker can develop a manner to precompute the solution, decreasing the effectiveness of the security mechanism. On the other hand, we have to take into account that often attackers have at their disposal a computational power far greater than legitimate users, and therefore creating excessively complex puzzles could then pose enormous difficulties to computational constrained users. Since we are focusing our research in finding appropriate incentives to implement in computational resource sharing networks, severely computationally constrained users are not our main targets. We will, nevertheless, take into account the limitations of users and maintain the effort required to solve computational puzzles between adequate boundaries.

The scheme proposed in [3] consists in creating random challenges based in inputs from all the participants in the network. That way, whenever a user A contacts another user B, B can verify not only that A has solved a puzzle but also that the puzzle included the input provided by B and is therefore valid. This solution introduces an all-to-all broadcast of challenges, which constitutes a burden to the network. However, the authors support the efficiency of the solution and show that the mechanisms can be implemented on top of the Chord overlay.

Rivest, Shamir and Wagner [27] propose a different approach, using puzzles with time restrictions, which they refer to as Time-Lock Puzzles. The idea is to create a Puzzle that cannot be solved until a pre-determined amount of time has passed, be it because the puzzle is so complex that it takes a considerable amount of time to solve or using trusted agents to keep information secret for a period of time. We are, evidently, more interested in the first option. The algorithms to solve the puzzles should

ideally be non-parallelizable, so that multiple machines must take as much time to solve the puzzles as simple, single processor computers. This approach can introduce a delay to requests in the network, and has the advantage of possible adjustment, since it is easier to control the complexity of the puzzle, different complexities can be used according to the situation. For example, if a user is flooding the network with requests, a puzzle with a longer solution time can be sent. Although this scenario is not simple, since not every user can decide on the complexity of the puzzles, it is nevertheless possible.

Tsang and Smith [32] argue that crypto puzzles have fundamental problems, namely their imprecision, since it is impossible to calculate solving times due to the extreme heterogeneity of computing devices, and the environmental impact of using innumerous CPU cycles without useful outcome. The authors propose a completely different approach based on trusted computing, which relies on security provided by physical hardware. Even though we acknowledge the relevance of problems presented, it is our goal to develop universal mechanisms, which can run on every system, without specific hardware requirements.

One of the most straightforward solutions is presented in [35] and consists in reversing a cryptographic hash, given the original random input with a number of bits deleted. The challenge is very simple to create and verify but demands intense computation from the challenged node to perform a brute force attack in order to identify the deleted bits. The downside of this approach resides on the fact that the solution can be parallelized and, therefore, the workload can be distributed provided the attacker possesses several processors.

We summarize the major benefits and disadvantages of these approaches in Table 1.

Puzzle	Advantages	Disadvantages
Gathered from Online Sources	• Freshness	• Dependency from sources outside the network • Difficulty to create puzzles and verify solutions
Public Source of Challenges	• No burden to the nodes in the system	• Dependency from sources outside the network • Central point of failure • Scalability issues
Puzzles with Contribution from Peers	• Distributed verification of solution	• Increased complexity • Increased network traffic
Time-Lock Puzzles	• Precise definition of elapsed time until the puzzle is soluble	• Multiple inputs necessary in different occasions • Larger burden to challengers
Hash Reversal	• Simple creation and verification	• Parallelizable solution

Table 1 – Computational Puzzles Comparison

4

System Design

After an extensive survey of existing systems and solutions to problems we're bound to encounter during this project, we could anticipate some pitfalls and utilize the knowledge gathered on the subject.

Our focus was always the balance between a currency and reputation system capable of delivering proper incentives to users, and the impact that system and all the security measures have on the performance of the network. As we have said when introducing the problem in hand, our mechanisms are intended to work in a decentralized application, therefore rendering any solution based on central servers unfit. Nevertheless, we have arrived at the conclusion that a good incentive-based application must rely on both reputation and currency and, as a result, the existence of brokers presents several advantages, as the FastTrack protocol used in KaZaA demonstrated. The responsibility of being a broker should be shared among users, as a manner to keep the principle of decentralization, but it is important that this responsibility does not become a giant burden to users, incapacitating them to produce useful computation. To achieve this compromise, we had to restrict the number of users assigned to each broker and develop architecture to efficiently cope with communication between brokers, in order to maintain coherence throughout the system.

4.1 Network Architecture

Our solution is built on top of the Pastry overlay network and, as a result, the network is represented as a ring of peers connected to their immediate neighbors. We implemented virtual groups of users to structure the network. Each group has an assigned broker and functions as a system within the system. These groups are self-regulated; the users classify their peers and the broker, a user responsible not only for the emission of currency but also for guaranteeing good performance, acts according to that classification.

All nodes are based on the same prototype and capable of performing the duties of the broker, although some components remain inactive until they are promoted to that status. During regular operation, every node maintains information about its neighbors, such as public cryptographic keys and reputation. Upon promotion that information is refreshed and completed and the new broker starts to update it as well.

The major differences between data processed by a broker and regular nodes are the need to collect the public cryptographic keys and reputation from all peers in the group instead of only from

neighbors, monitor transactions to detect fraudulent behavior and, most importantly, information about the amount of currency each node possesses. A broker must also maintain a list of neighbor brokers.

The groups are highly dynamic, but the number of users in one group is always kept inside defined thresholds, since a very large group would become almost impossible to control by a single broker, resulting in a bottleneck, and, on the other hand, a group with a small number of peers cannot satisfy those users needs.

The existence of brokers means that our architecture can be classified as partially hierarchical and structured. In the following sections we describe the processes of creation, dissolution and fusion of groups in further detail.

4.2 Groups

The process of adjusting the number of groups according to the needs of the system must be as straightforward and effective as possible so as to maximize the performance of the application. We describe the situations when changes to the groups are necessary and how the processes develop.

4.2.1 Initialization

To bootstrap the system, a small set of trustworthy users is needed, to act as super nodes. In fact it is possible to initialize the system with just one super node, however in this situation the initial user would be vulnerable to attacks, which could lead to a total failure in bootstrapping the system. These users only need to reside in the system long enough for other users to be accurately classified as trustworthy. From that point on the brokers will be elected from the pool of peers, according to their reputation.

In order for a new user to enter the network, it has to contact a user already in the system, meaning that some sort of contact outside the application is required, for instance publication of IP addresses in public lists. This is the common method all P2P applications use to perform the initial peer discovery, the most prominent example being the publication of torrent files for BitTorrent in web pages.

After acceptance into the network, a new unique ID is created and assigned to the user, who can be allocated to the same group of the peer contacted, if the objective is to reduce the number of messages between groups. If the network traffic is not a critical factor, the user can be assigned to a random group, to avoid the possibility of malicious users choosing which group they want to join. The broker responsible for that group will then contact the new user, presenting its credentials, and send its neighbors list so that the new node can start contacting other peers. There is no point in establishing a relation between the node ID and the group that node belongs to, since the groups are dynamic and, as we explain in the following sections, users can be transferred to other groups transparently.

4.2.2 Group Partition

The users of the system are very heterogenic; therefore the threshold for group size must be set accordingly to the computational capability of the broker responsible for that group, which can be

assessed using computational puzzles. When a group is growing and the size threshold is about to be attained, the most reputed user should be appointed as the broker for a new group. Half of users, chosen randomly, are transferred to the new group.

4.2.3 Group Merge

When a group is not performing as expected, with the demand for resources far greater than the supply inside that group, there is a choice to be made. Either the group is dissolved and the users scattered among other groups, or the group is merged with another group. While the first option is undoubtedly simpler, the second can be much more effective, provided that the brokers exchange information, conducting the stalled group towards a group with a large amount of unused resources.

4.3 Currency

The brokers are the only users capable of introducing currency in the system and are responsible for controlling the amount of currency inside their group.

When a user enters the system and is integrated in a group, the broker of that group registers an additional amount of currency, distributed not to the new user but to the most reputed users in that group. This mechanism prevents users entering a group from using all the initial currency and then exit the group giving nothing in exchange.

Inversely, when users exit a group, the broker must make sure that the amount of currency in the group remains unchanged. If the exiting user has more currency than introduced when he entered, that amount must be distributed, otherwise currency is removed from the users in that group. This process guarantees that there is no shortage of currency and, more importantly, that even highly reputed users cannot retain such a quantity of currency that allows them to stop sharing resources and still having enough to acquire other user's resources for a long period, which is to say that no user can be allowed to obtain and maintain a dominant position.

4.4 Reputation

As we have mentioned in section 3.2, currency and reputation are essential to the well functioning of the network. Through our research, we have come to the conclusion that using both currency and reputation as two separate concepts is the most adequate manner to cope with all the problems present in peer-to-peer applications. Reputation is not only necessary to deal with problems such as users accumulating currency and then make the system imbalanced, but can also provide basis for a more complex resource economy, where prices fluctuate and the quality of the resources may vary.

The differences between the role of currency and reputation emerge immediately when we consider a resource transaction. Currency is passed from the buyer to the seller and, as a result, the seller becomes wealthier and the buyer poorer. However, if the transaction was carried out correctly, both users see their reputation rise, since every correct transaction is positive for the system. The buyer provided payment and the seller kept his promise to deliver the resource, so the position of both users becomes stronger within the system.

On the other hand, a user with little need for resources who is a great contributor to the system will see the amount of currency in his possession rise indefinitely. The reputation of a user must not follow this line of thought and a limit has to be set, as a measure to guarantee the balance of the system and the availability of resources. If users who have accumulated large amounts of currency decide to stop sharing resources and use that currency only to consume resources from their peers, it creates an unbalanced situation and the system will undoubtedly be harmed.

The reputation of users is periodically adjusted downwards to promote activity and, additionally, when a broker records a bulk of transactions with the same buyer, that user also has his reputation decreased. We find this latter adjustment necessary to maintain the system's throughput, since users with large amounts of currency can flood the network with requests, degrading the experience of other users. However, to avoid excessive penalization of idle users, the reputation adjustment can never lower the reputation below the initial value.

Using these precautions when awarding and removing reputation will balance the system and prevent unfair behavior from users.

4.5 Security

As we have mentioned before, the basis for a fair sharing system is the ability to detect and cope with malicious users, therefore our solution based on currency and reputation will only be as effective as it is trustworthy.

Since there are no means to establish a direct connection between physical and virtual identities, the main threat to the system are attackers who create multiple bogus identities for their own profit, subverting the classification parameters. This type of attack is known as Sybil attack, as described in section 3.3.1, and can provide attackers with unfair advantage over legitimate users. Moreover, it can be a vehicle for further attacks, mainly Denial-of-Service attacks. In this section we describe the security mechanisms included to deal with Sybil attacks and other possible threats to the system.

4.5.1 Cryptographic Key Infrastructure

As the cornerstone of the security mechanisms, we deem vital the existence of a cryptographic key infrastructure. To circumvent the problems presented by remote operations and communications using anonymous networks, the usage of cryptographic mechanisms to prove the identity of users has become customary in many distributed systems and applications.

The outline and complexity of these cryptographic mechanisms vary according to the specific necessities and capabilities of the systems using them. The most common option to provide security assurance against malicious users is to employ digital certificates, which identifies the entity signing the data, or cryptographic keys, symmetric or asymmetric, capable of guaranteeing integrity, authenticity and privacy of the data.

For the type of system we are considering, the digital certificate option does not seem appropriate, since it requires a trusted entity, a certificate authority, capable of validating these certificates, which is not present in a P2P network.

The utilization of symmetric keys is more efficient than using a pair of public and private keys; however it implies a way to share a key between two peers in a secure manner. The reliability of this process cannot be guaranteed without an additional effort and, since our architecture entails the dynamic allocation of peers in groups and, consequently, constant changes in the super nodes pool, the use of asymmetric cryptography is more appropriate. If each node possesses a key pair, there is only a need to maintain one of the keys private, while the public key can be transmitted freely. The responsibility of creating the pair of cryptographic keys relies on each user, relieving the broker of the group from that rather computational expensive task.

The size of the cryptographic key is determining to the reliability of the encryption. Using the RSA algorithm, which allows both signing and encryption, a key length of either 1024 or 2048 bits is considered secure. Since the average time a user spends in the system is relatively low, we find acceptable the use of 512-bit keys to minimize the computational cost of generating the keys and signing the messages.

It is essential to the correct operation of the system that basic properties are guaranteed:

- Authenticity of Origin, so that there is no doubt on the identity of the sender;
- Non-Repudiation, forcing the sender to assume the authorship of the message;
- Integrity, to guarantee that the message was not adulterated.

Users must sign each message sent using their private key, so that the properties above are maintained. Although the broker of the group is responsible for providing nodes the public key necessary to verify the authenticity and integrity, this duty has no significant impact on performance since the nodes are able to store that information for future interactions with the same peer.

4.5.2 Sybil Attack

When a malicious user resorts to the Sybil attack in the context of peer-to-peer application, usually the objective is to obtain some advantage over other users in order to consume resources without contributing to the system. Our currency allocation policy deals with this issue and, since we do not provide new users with currency, they are obliged to contribute to the system in order to acquire means to pay for resources. Attackers may also use their multiple identities to praise each other or criticize legitimate users, undermining the reputation system. The restrictions we implement regarding multiple transactions from the same user in a small time frame in addition to the certification of messages between users guarantee that it is not possible for attackers to tamper with their own reputation or to discredit legitimate users.

Nonetheless, the Sybil attack may just be a preparation for a more extensive attack, namely Denial-of-Service attacks against brokers.

To protect the application from Sybil attacks we use computational puzzles, described in section 3.3.5. The complexity of the puzzles was tested and adjusted accordingly to benchmarks in a simulated environment, so that legitimate users are able to use the application as intended, while

possible attackers have to deal with a heavy burden and cannot profit from using multiple virtual identities. The detailed analysis of the results of the simulations is in chapter 6.

4.5.3 Selfish and Malicious Users

The detection of users who are not contributing to the system or are failing in their responsibilities is based on the reputation of those users. If a user is only consuming resources from other peers, it will eventually have no currency to make payments and the necessity to share resources will arise. However, if that user has a large amount of currency, collected along an extensive period of time, there is the possibility he would stop contributing and still be able to pay for resources. That is why the use of reputation is also important. While a user can retain as much currency as it intends, there is a limit for the amount of reputation, so that each user must continue to put in its resources, even if in a smaller amount than it uses other peers' resources.

The problem of untrustworthy users is more complicated. Those are users who pretend to share their resources but then fail to deliver them, deliver false results of computation or fake content. Untrustworthy users are a liability to the application not only because of the entropy they introduce in the network, but also due to the decrease in performance and throughput they cause and must therefore be dealt with efficiently. Peers can report a user and the broker of that group acts so as to expel the malicious user.

In order to protect the system from badmouthing attacks, wrongful claims that a peer is untrustworthy, the peer who files the report presents proof that a promise was made and not fulfilled. Since every message exchanged between nodes must be signed with the private key of the sender, the log of those messages can be used as proof that a resource was promised but not delivered, or that the resource provided did not match the promise made.

The mandatory signature of every message also guarantees that it is impossible for users to create bogus messages praising themselves, as the broker will always validate the identity of the node responsible for the message.

So that every user is encouraged to report bad peers, the broker, upon confirming the information, awards reputation from the reported peer to the legitimate user. Without this incentive, users could decide to keep the information only for their personal use, without taking the time to contact the broker.

4.5.4 Newcomers

A delicate problem occurs every time a user joins the network, due to the lack of information the community has about the reliability of the new peer. Since our solution features super users, the brokers of each group, every new user must be registered and recognized by the broker of the group he enters.

The only option for a newcomer to build up reputation and accumulate currency is providing resources to other users but those reputed users will not be inclined to acquire resources from unknown peers, as the probability of defection is higher.

The act of sharing resources with a peer who does not yet possess reputation information is known as optimistic unchoking. This principle is used in every P2P application to allow progressive integration of a new user in the network, otherwise newcomers would never be able to start exchanging resources. This optimistic unchoking is, in most of the cases, random, which is to say that the process of integration of new users is not analyzed and bears no consequences to the system.

We pretend to use a different approach and rely on the broker to maintain a record of recent transactions involving recently arrived users and posting the average cooperation rate from these users. There is still a random factor to optimistically unchoke a node, but the probability is continually adjusted using this record of past experiences. If the number of defections increases, this method also provides protection against whitewashing – users leaving the system and rejoining with a new identity to avoid penalties in their reputation. Every node can choose to consult the broker asking for advice or decide whether to accept the exchange partner based on its own experience.

We consider a newcomer to be a peer that has not yet had the opportunity to show its reliability or had the time to communicate with an acceptable number of nodes to build trust relationships. The threshold defined to decide if a node is considered to be a newcomer could be adjusted depending on the size of the group, to better integrate the impact untrustworthy newcomers could have on the network. However, we do not find the additional complexity and consequences to the performance of brokers justified, and prefer to adopt a common sense attitude and settle a permanent value.

After simulation, we considered that 10 interactions, either as asker or provider of resources, are plenty to start using the reputation points as a viable indicator of the behavior a node will show; consequently, after these interactions a user is no longer considered a newcomer by the system. We arrived at this conclusion by verifying that, simulating the presence of erratic nodes in the system, the probability of a malicious node behaving correctly in the first 10 interactions is less than 5%. There is, obviously, the possibility of a peer discovering this method and simulate a different behavior precisely in the initial interactions. We feel that possibility is not a threat to the system, as long as the malicious nodes are later detected, and that this approach will be efficient in the large majority of situations.

4.5.5 User Exclusion

After the decision to exclude a user from the system has been taken, the broker contacts that user to communicate the situation and proceeds to correct its neighbor's list to reflect the decision. Nodes that enter the system after that moment no longer have contact with the expelled user and, while other peers can continue to contact the expelled node using stored information, no more exchanges involving that node are authorized by the broker. As some attempts fail, the users will remove the expelled node from their own neighbor's list, progressively eliminating any record of that node.

This information can take some time propagating to all peers and some interactions will have to be interrupted, resulting in unfruitful effort for some nodes. However, the other possible approach consists in flooding the network with messages notifying each user of the eviction, which is clearly worse in terms of performance and would no doubt have a higher impact on the system.

5

Implementation Details

5.1 Message Structure

Every message exchanged between users in the network has the same structure, being composed of two fields: type and body.

We summarize the contents of the body of the most important messages in Table 2.

Message Type	Message Body				
Join Request	Destination	Source	Public Key	Signature	
Computational Challenge	Destination	Source	Nounce	Hash	Signature
Challenge Response	Destination	Source	Solution	Signature	
Join Reply	Destination	Source	Neighbors Set	Signature	
Leave Notification	Destination	Source	Signature		
Service Request	Destination	Source	Request	Signature	
Service Reply	Destination	Source	Value	Signature	
Service Demand	Destination	Source	Details	Signature	
Service Supply	Destination	Source	Resource	Signature	

Table 2 – Message Contents

5.2 Interactions

In this section we detail the message flow between peers in each major situation. We do not regard as necessary presenting low-level information in the sequence diagrams, such as representing cryptographic signature details, but it will be referred to when considered relevant.

5.2.1 New User Joining the Network

To enter the network, a user must possess the address of a node that is already active and send a join request to that node. Since only Super Nodes can accept new members, any other nodes forward the request to the broker of the group they are in.

The broker assigns the 128-bit ID of the new node resorting to a Uniform Random Generator, which is given a random seed as parameter. Constant synchronization between brokers is unfeasible due to the scalability limitation it would introduce, so the solution adopted is an acceptable compromise, since there is a probabilistic guarantee that each ID generated is different.

When the Super Node communicates the UID to the new user, it also includes the cryptographic puzzle to be solved by that user. This challenge, described in detail in section 5.4, has the purpose of posing a barrier against attacks, but can also be used as a benchmark for the computational capacity the new node possesses. The joining user becomes a member of the network only when the correct solution is sent back to the broker and verified. The Super User then conveys the acceptance information to the new member and also information about other users in that group, the neighbors, so that the newcomer can start participating in the resource exchange.

The entire process is exemplified in Figure 2.

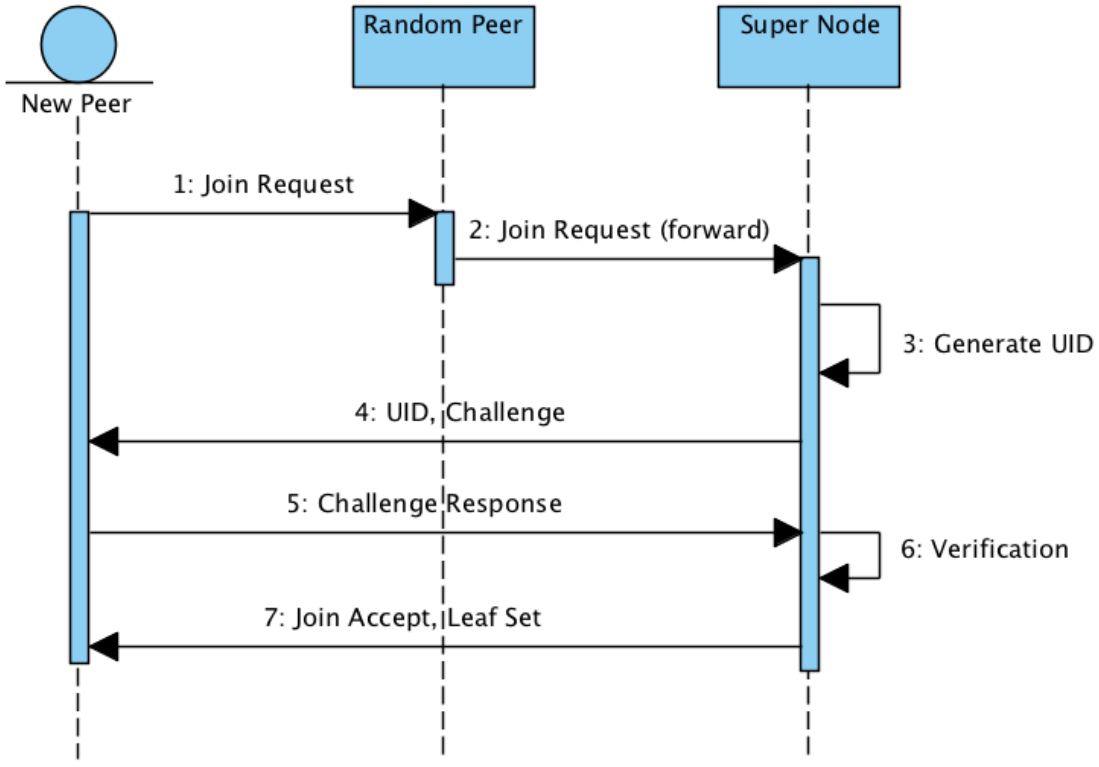

Figure 6 – Join (Sequence Diagram)

5.2.2 Successful Exchange of Resources

When a user intends to acquire resources from other members of the application, he starts by selecting a neighbor, who must belong to the same group. He can then ask the broker for information about the selected peer, if he deems it necessary, however this step can be averted if the user has had previous successful interactions with that neighbor. The broker can provide information about the reputation of another user, as well as the public cryptographic key needed to verify the authenticity of the messages received from the exchange partner, as mention in section 4.5.1.

The request for a resource must also be signed to guarantee authenticity and non-repudiation since a large number of requests can be considered a denial-of-service attack to the network, therefore it is necessary to identify the attacker without any doubts.

The provider has to keep information about the resources he has supplied to provide proof in case the asking peer tries to deny the exchange. The same applies to the payment made.

After both parties confirm the exchange and notify the broker, the broker awards a reputation bonus to both users to reward their correct behavior. If the peers have already had previous interactions and have the information needed already stored, the broker needs only to get involved in this last step. By allowing the major part of the operation to be carried out by the peers, without participation from the broker, we keep the broadcast of messages to a minimum, benefiting the performance of the network and relieving the super node from additional chores.

Figure 3 represents the steps in a normal resource exchange process, including a request to the broker about the reputation of the selected neighbor and the reward process after the exchange. As we have discussed before, the implicated peers may have had previous interaction and, as a result, have information stored that eliminates the need to contact the broker prior to the exchange. In that case, steps 2, 3, 5 and 6 are not necessary and do not occur.

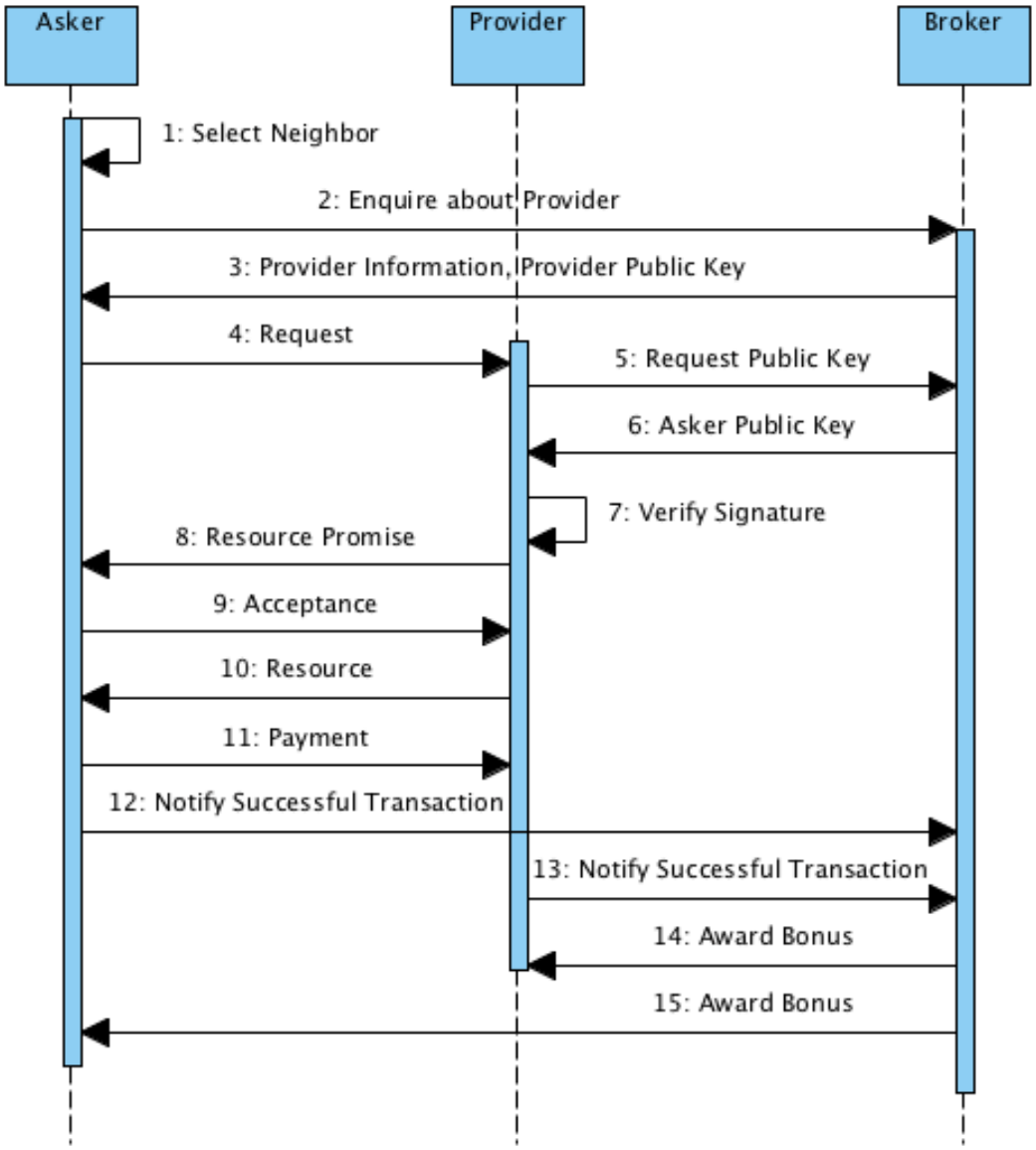

Figure 7 – Successful Exchange (Sequence Diagram)

As a protection against collusion between peers and also to prevent abuse from peers with a dominant position, the broker only awards reputation if certain assumptions are verified. We tested the probability of a node being wrongfully accused of collusion by performing simulations with correct nodes only. It is acceptable to the broker to maintain information about the last 50 interactions and, if a pair of nodes exchanged resources at least 10 times in that timeframe they are accused of collusion and penalized. In Table 3 we summarize the results of our simulations, presented the averages values resulting from 10 runs for each simulation, with an average of approximately 8500 transactions. We repeated the same tests with faulty nodes in the system and confirmed that setting the threshold at 10 exchanges guarantees virtually no false detections. However, the determinant factors to the accuracy of collusion detection are the average number of nodes in each group and, for that reason, the detection mechanism is enabled only when the group has, at least, 15 nodes, otherwise it is impossible to maintain the same efficiency determining collusion penalties.

Threshold	False Positives										
	1	2	3	4	5	6	7	8	9	10	Average
3	29	36	35	33	34	33	29	36	31	32	32,8
5	13	12	15	13	14	14	13	15	14	13	13,6
7	6	3	4	4	5	4	6	4	5	5	4,6
9	0	0	1	0	0	0	0	0	1	0	0,2
10	0	0	0	0	0	0	0	0	0	0	0

Table 3 – Collusion Detection Efficiency

Using the same line of thought, and the same measurements, a node is also penalized if it participates in more than 10 transactions, considering 50 consecutive transactions, as the buyer, because it is considered to constitute an unbalancing factor capable of reducing the performance of the system and the ability for other nodes to have access to resources. If a user is considered to be taking advantage of a dominant position and accumulated currency, the broker acts as to reduce the reputation of that node and, consequently, to reduce the probability of another peers accepting that user as a partner.

In Figure 4 we represent a situation where two nodes collude to improve their reputation and, because the limit of transactions has been exceeded, the broker penalizes both users. In the case of a single user taking advantage of a dominant position, the provider of resources is still rewarded, while the offender is penalized. The modification is in step 8, which becomes a reward instead of a penalization.

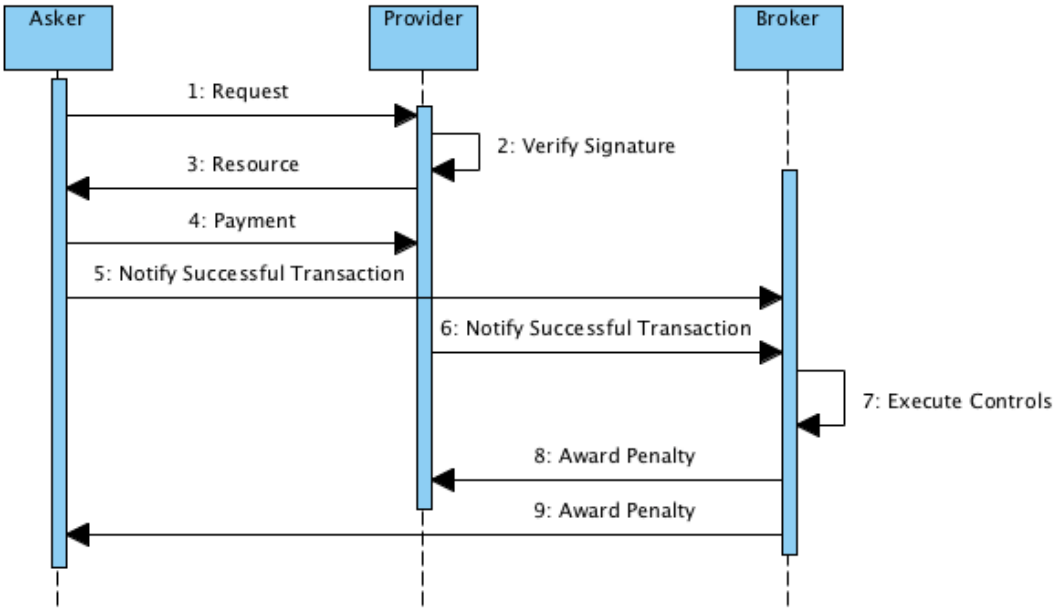

Figure 8 – Collusion Detection (Sequence Diagram)

An additional interaction between the asker and the broker or the provider and the broker may exist if one of the peers has only recently arrived to the network. In this case, the broker is able to provide information about newcomers in the system, information used to decide if the transaction continues with the selected neighbor. Figure 9 represents a situation when the providing node is a newcomer to the system, but the interactions are exactly the same if the new user in the network is the provider. Since the information about the reputation of newcomers is inconclusive, the broker includes information about the compliance rate of newcomers in the system and, in this case, the asker continues with the transaction.

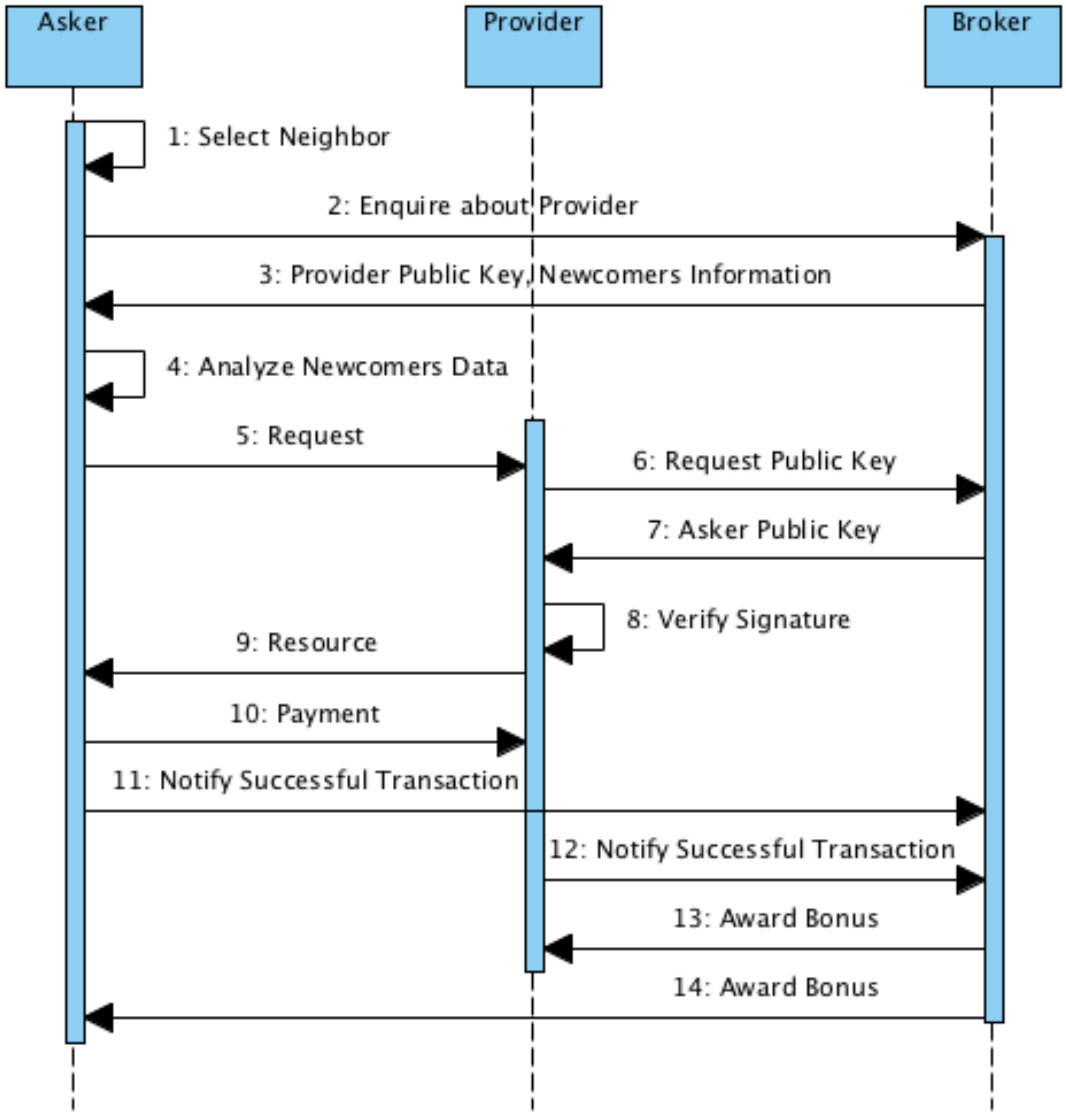

Figure 9 – Newcomer Situation (Sequence Diagram)

5.2.3 Corrupt Resources

To maintain the quality of resources in the system and accurately detect malicious users, it is possible to denounce peers if they fail to deliver the resources promised or deliver corrupt resources, described in section 3.3.3. In Figure 10 an example of one such episode is given.

The offended user, who is obliged to provide evidence of that claim, can then contact the broker. Since every message exchanged between users is signed, the messages sent by the malicious user, both during the negotiation phase and then the delivery of results, can be used as evidence to corroborate the claim.

We leave the implementation of specific means to detect fraudulent behavior to the developers of applications using these incentive mechanisms, since these can vary immensely depending on the purpose and environment in which the application is run. Our implementation makes it possible to report those actions and the actions performed by the broker as a consequence are also in place.

To encourage users to report malicious peers, a bonus is awarded in case the claim is verified, besides the obvious penalty for the malicious user.

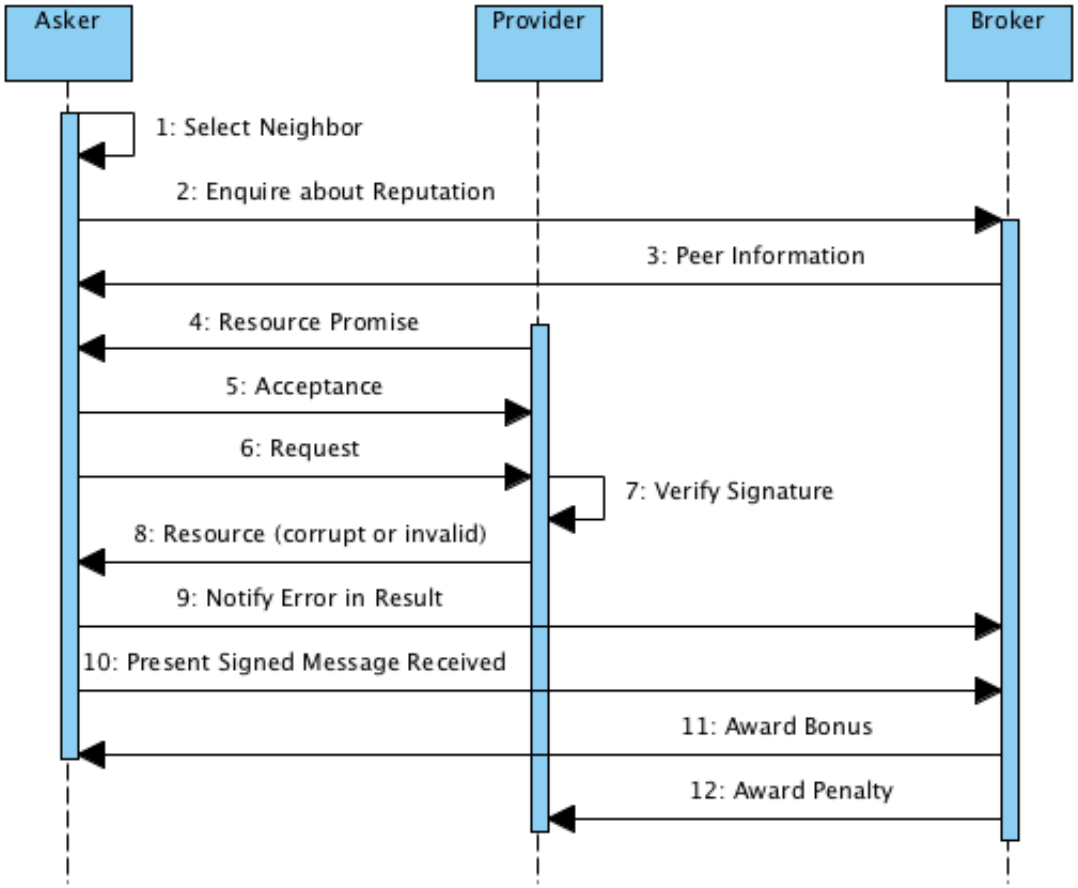

Figure 10 – Delivery of Corrupt Resources (Sequence Diagram)

5.3 Computational Puzzles

As we mentioned in section 3.3.5, computational challenges are one of the most efficient ways to combat Sybil attacks. This method, also used sometimes to determine the computational resources of a node, is based on intensive computation, specifically mathematical problems.

After considering several approaches, described in section 3.3.5, and bearing in mind that both efficiency and complexity are vital variables, we opted by using Hash Reversal challenges. In this approach, a challenger produces a cryptographic hash and, after erasing *n* bits from the input, sends that information to the challenged nodes so that they determine the complete original input, also called *nounce*.

The challenged nodes must perform a brute force search, exemplified in Figure 11, and multiple hashing actions to determine the correct value. The complexity of the computation can be easily adjusted by varying the parameter *n*. Both generation and verification are simple procedures, which guarantees that the Super Nodes, the challengers, will not be burdened.

```
findNounce(nounce)
      if nounce is complete
            newHash ← hash(nounce)
            return (newHash == outputHash)
      else
            index ← first null bit in nounce
            for i ← 0..1
                  nounce[index] ← i;
                  newNounce ← nounce;
                  result ← findNounce(newNounce)
                  if result is TRUE
                        return newNounce
                  end
            end
      end
end
```

Figure 11 – Hash Reversal Algorithm

The downside of using hash reversal challenges lies in the method of solving these challenges, which can be parallelized. As a consequence, some authors have made efforts to develop non-parallelizable puzzles. An interesting approach is described in [31], where authors compare several approaches to the use of computational puzzles, their positive and negative aspects and ways to enhance them. Nevertheless, and as pointed out in that paper, even parallelizable puzzles are effective against the majority of attacks, with the advantage of simplicity in both construction and verification.

We find the hash reversal solution balanced and adequate to the objectives in hand, allowing for relatively low costs for the Super Nodes and mitigation of existent threats. The results of our simulations suggest that the computational puzzles are efficient and do not pose an excessive burden to the system and to legitimate users. However, considering possible future developments, our framework can be easily adapted to different challenges and fine tuning of puzzle complexity.

We chose to use an alphanumeric *nounce* with a length of 512 bits, and defined *n* as 40 bits. The simulation values that lead to this choice are detailed in Table 4, with all the values expressed in seconds. The simulations were run on a MacBook Pro laptop with an Intel Core 2 Duo processor running at 2.2GHz, 4GB of DDR2 667MHz RAM and Java version 1.5.0_19.

n	**Simulation 1**	**Simulation 2**	**Simulation 3**	**Simulation 4**	**Average**
24 bits	0,55 s	0,57 s	0,59 s	0,50 s	0,55 s
32 bits	0,79 s	1,21 s	1,23 s	1,18 s	1,10 s
40 bits	13,4 s	8,3 s	10,8 s	11,6 s	11 s
48 bits	295 s	313 s	504 s	176 s	322 s

Table 4 – Hash Reversal Simulations

Considering the values obtained through simulation, we believe that *n* should be set to 40 bits. The necessary computation is reasonable for a legitimate node, since it is only performed once, when entering the network. However, for an attacker wishing to deploy multiple identities in the application, the required time to satisfy enough computational puzzles is very high, even if the attacker has access to high-end technology and fast processors.

Nevertheless, *n* is just a parameter and can easily be adjusted if the person responsible for the application considers it unsuitable or if the computational requirements to solve reverse hash problems become easier to satisfy with technology development.

6

Simulation and Evaluation

Our solution was tested using a simulation framework, PeerSim, which provided us with the opportunity to tune certain parameters and to increase the accuracy of our incentive mechanisms. The parameters are passed to the simulator via a configuration text file.

After hundreds of simulations we have reason to believe our system is stable and can be successfully implemented in real applications.

6.1 Faulty Nodes

In order to evaluate the effectiveness of the incentive mechanisms developed, it is necessary to simulate network operation as close to reality as possible, including the presence of users who try to get an undeserved advantage over other users. We use the term faulty users to identify every member of the network who does not comply with established rules and aims to gain an advantage over other users exploiting weaknesses in the system.

Using information gathered during our research we could point out the major problems these systems suffer from and, consequently, deliberately instructed some of the nodes to adopt erratic behaviors, giving us the opportunity to access the stability of the application. We introduced random parameters so that multiple conducts are followed; therefore there is not a simple configuration for these faulty nodes but in fact several different approaches and levels of aggressiveness to the possible frailties in the system.

According to the parameters of the simulation we define, these nodes might try to collude with each other so that their reputation and amount of resources rapidly increases, perform praising or badmouthing attacks to manipulate reputation measures, or carry out overbooking of resources making promises and then failing to deliver the resources.

Not only it is possible to increase the aggressiveness of faulty nodes and preferred attack methods, but we can also control the approximate number of faulty nodes in the system. This is particularly important since we must measure the impact on the system of the faulty nodes.

6.2 Configuration Parameters

Table 5 contains a summary of all relevant parameters provided to the simulation and a simple explanation for each one. Further details and in-depth analysis are given in the next section.

Parameter	Purpose
Duration	Controls the duration of the simulation, making possible to analyze the effectiveness of the mechanisms in short and long simulations
Initial Budget	Sets the initial currency amount, which limits the possibility of a node acquiring resources from a peer
Initial Reputation	The amount of reputation, if the penalties are fixed, determines the number of faulty behaviors a user can have before being expelled
Maximum Group Size	The maximum number of nodes simultaneously on a group allows us to measure the impact on the performance of super nodes
Maximum Cost of a Resource	The variation on the average cost of acquiring resources has a direct impact on the number of exchanges a user can accomplish. This parameter is necessary and has a different objective than the initial budget
Probability of a Node Being Faulty	Adjusts the percentage of faulty nodes in the system
Probability of a Node Refusing to Provide Resources	Simulates a situation in which resources are scarce and therefore can be difficult to acquire
Probability of a Node Being Added to the Network	These parameters are used to control the behavior of nodes in the system, increasing and decreasing the rate of entrance and abandoning. The sum of these three probabilities must be 1
Probability of a Node Being Removed from the Network	
Probability of a Node Asking Other Peers for Resources	
Probability of Faulty Node Making Fake Requests or Acceptances	Parameters to adjust the behavior of faulty nodes, making them more or less aggressive
Probability of a Faulty Node performing Badmouthing or Praising Attacks	
Probability of a Faulty Node Colluding With Other Faulty Nodes	
Probability of a Node Optimistically Unchoking a New Peer	This parameter is used if a node chooses not to consult the broker to decide if newcomers are to be trusted
Use of Signatures	Enables or disables the use of cryptographic signatures, calculating the impact on performance
Existence of Super Nodes	Enables or disables the existence of super nodes and groups, allowing to access the impact of our architecture
Progressively Award Currency and Reputation	Enabling this option we guarantee that users do not enter the system possessing a large budget and must contribute with resources in order to accumulate currency
Periodic Reputation Adjustment	Periodically decreasing the reputation of nodes one can oblige nodes to continuously contribute, instead of just using accumulated currency
Collusion Detection	The broker of each group maintains a record of transactions to verify possible collusion between nodes by searching for multiple exchanges involving the same nodes in a short period of time

Table 5 – Simulation Parameters

We decided to keep two of the parameters constant throughout the simulations, namely the number of cycles in each simulation and the initial reputation of each node. The penalties for incorrect behavior and awards for successful exchanges are also stable so that the number of detected infractions needed to expel a node is the same in all simulations.

We chose to award 50 reputation points (kudos) to each new node. The penalties and awards are listed in Table 6 and Table 7, respectively. The ceiling for reputation is 100 points.

Reputation Penalties	
Insufficient Funds	-5
Denying Promised Resources	-5
Praising or Badmouthing	-25
Collusion	-50
Falsely Reporting other Node	-25
Periodic Adjustment	-1

Table 6 – Reputation Penalty Values

Reputation Awards	
Successful Exchange (Asker)	1
Successful Exchange (Provider)	2
Accurately Reporting a Faulty Node	2

Table 7 – Reputation Award Values

6.3 Simulation Analysis

In this section we present the results of our simulations and given parameters. Knowing that the number of possible configurations is almost unlimited, we selected what we consider to be the most interesting and relevant possibilities.

It was our objective not only to measure the effectiveness of our system, but also to determine the impact of all mechanisms in the performance.

All the simulations were run on the same hardware, a MacBook Pro laptop with an Intel Core 2 Duo processor running at 2.2GHz, 4GB of DDR2 667MHz RAM and Java version 1.5.0_19.

6.3.1 Simulation 1 - Zero Configuration

This initial simulation is made without any protection mechanisms, so that performance impacts can be measured. We assume that there are no faulty nodes to allow for future analysis of the throughput the application can attain.

Initial Budget	100
Existence of Super Nodes and Groups	False
Maximum Group Size	N/A
Resource Availability	100%
Maximum Price of Resources	5
Signature of Messages	False
Progressive Reputation and Budget	False
Periodic Reputation Adjustment	False
Collusion Detection	False
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	0%
Probability of Overbooking	N/A
Probability of Praising/Badmouthing	N/A
Probability of Collusion	N/A

Table 8 – Zero Configuration: Simulation Parameters

Joins	2378	Reputation (Correct Nodes)	Average	72
Departures	79		Standard Deviation	17
Expelled Nodes	0	Budget (Correct Nodes)	Average	100
Final Number of Nodes	2300		Standard Deviation	10
Super Nodes	0	Successful Exchanges (Correct Nodes)	Average	8.44
Faulty Nodes	0	Reputation (Faulty Nodes)	Average	N/A
Exchanges Attempted	20040	Budget (Faulty Nodes)	Average	N/A
Exchanges Failed	0	Successful Exchanges (Faulty Nodes)	Average	N/A
Elapsed Time: 46 seconds				

Table 9 – Zero Configuration: Simulation Results

6.3.2 Simulation 2 - Inexistent Incentive Mechanisms

In this simulation we intend to verify the consequences of a peer-to-peer application without any incentive mechanisms, where some of the nodes perform attacks in order to maximize their profits.

Initial Budget	100
Existence of Super Nodes and Groups	False
Maximum Group Size	N/A
Resource Availability	100%
Maximum Price of Resources	5
Signature of Messages	False
Progressive Reputation and Budget	False
Periodic Reputation Adjustment	False
Collusion Detection	False
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	10%
Probability of Overbooking	25%
Probability of Praising/Badmouthing	25%
Probability of Collusion	25%

Table 10 – Inexistent Incentive Mechanisms: Simulation Parameters

Joins	2344	Reputation (Correct Nodes)	Average	72
Departures	96		Standard Deviation	16
Expelled Nodes	0	Budget (Correct Nodes)	Average	100
Final Number of Nodes	2249		Standard Deviation	10
Super Nodes	0	Successful Exchanges (Correct Nodes)	Average	8.57
Faulty Nodes	211	Reputation (Faulty Nodes)	Average	74
Exchanges Attempted	21960	Budget (Faulty Nodes)	Average	109
Exchanges Failed	0	Successful Exchanges (Faulty Nodes)	Average	17.14
Elapsed Time: 45 seconds				

Table 11 – Inexistent Incentive Mechanisms: Simulation Results

6.3.3 Simulation 3 - Collusion Detection

This simulation includes collusion detection. To identify the possibility of collusion attacks it is necessary the presence of at least one Super Node, however we prefer to perform this simulation without the possibility of multiple groups, to separate the consequences of a different architecture. To achieve that goal, the maximum number of nodes in a group is very large so that there is only one pastry ring.

Initial Budget	100
Existence of Super Nodes and Groups	True
Maximum Group Size	500000
Resource Availability	100%
Maximum Price of Resources	5
Signature of Messages	False
Progressive Reputation and Budget	False
Periodic Reputation Adjustment	False
Collusion Detection	True
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	10%
Probability of Overbooking	0%
Probability of Praising/Badmouthing	0%
Probability of Collusion	25%

Table 12 – Collusion Detection: Simulation Parameters

Joins	2376	Reputation (Correct Nodes)	Average	71
Departures	144		Standard Deviation	17
Expelled Nodes	98	Budget (Correct Nodes)	Average	100
Final Number of Nodes	2135		Standard Deviation	10
Super Nodes	1	Successful Exchanges (Correct Nodes)	Average	8.80
Faulty Nodes	165	Reputation (Faulty Nodes)	Average	72
Exchanges Attempted	21345	Budget (Faulty Nodes)	Average	82
Exchanges Failed	21	Successful Exchanges (Faulty Nodes)	Average	10.75
Elapsed Time: 46 seconds				

Table 13 – Collusion Detection: Simulation Results

The collusion detection mechanism did not result in any false positive detection, which means that all 98 nodes expelled were faulty nodes. We cannot verify if the faulty nodes still in the system resorted to collusion, remembering that there is only a 25% chance of that happening, but nevertheless, their reputation was slightly reduced, their budget decreased substantially as well as the average number of exchanges.

6.3.4 Simulation 4 - Message Signing

With this configuration we intend to verify the benefits of introducing message signing to prevent attacks such as badmouthing and praising, as well as allowing nodes to report malicious peers who fail to deliver promised resources. We maintain the configuration regarding the size of groups.

Initial Budget	100
Existence of Super Nodes and Groups	True
Maximum Group Size	500000
Resource Availability	100%
Maximum Price of Resources	5
Signature of Messages	True
Progressive Reputation and Budget	False
Periodic Reputation Adjustment	False
Collusion Detection	True
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	10%
Probability of Overbooking	25%
Probability of Praising/Badmouthing	25%
Probability of Collusion	25%

Table 14 – Message Signing: Simulation Parameters

Joins	2330	Reputation (Correct Nodes)	Average	73
Departures	116		Standard Deviation	17
Expelled Nodes	293	Budget (Correct Nodes)	Average	100
Final Number of Nodes	1922		Standard Deviation	10
Super Nodes	1	Successful Exchanges (Correct Nodes)	Average	9.02
Faulty Nodes	64	Reputation (Faulty Nodes)	Average	48
Exchanges Attempted	20986	Budget (Faulty Nodes)	Average	94
Exchanges Failed	506	Successful Exchanges (Faulty Nodes)	Average	4.2
Elapsed Time: 102 seconds				

Table 15 – Message Signing: Simulation Results

In this simulation we can see the impact on the number of exchanges correct nodes accomplish and also that the reputation of faulty nodes denotes that their actions were detected. In combination with other mechanisms, the diminished reputation can be used to segregate faulty nodes, minimizing the impact on the rest of the network.

The most relevant result is the number of expelled nodes, approximately three times more than with just collusion detection, and the number of faulty nodes remaining in the network. This clearly shows that the effectiveness of the detection mechanisms was drastically improved and must be taken into account when analyzing the reputation and budget of the faulty nodes in the network. Since the penalties for incorrect behavior are severe, especially for collusion, those nodes are expelled and their data is not reflected on other indicators.

We must also take notice of the elapsed time of the simulation, which more than doubled, due to the computational requirements of cryptographic mechanisms. Although the elapsed time of this simulation is not especially high, the impact on performance must be taken into account when we consider the scalability of the system. The stage with higher computational requirements is the creation of the cryptographic keys but, after an extended period, that factor is softened and signing the messages, as well as verifying signatures, becomes a considerable burden, mainly for the Super Users. Distributing that responsibility is one of the predominant reasons for the change in architecture we proposed earlier and test in the next simulation.

It is very important to note that, in our simulations, we are using 512-bit keys, less than what is commonly recommended. We explain that decision in section 4.5.1 but we feel it is worth referring that the same simulation using 1024-bit keys takes considerably longer, 184 seconds in average.

6.3.5 Simulation 5 - Full Incentive Mechanisms

This simulation includes all our incentive mechanisms as well as the architecture described in section 4.1, with multiple groups. We set the maximum number of users in each group at 500 after simulations with different values, as it proved to be an equilibrate value considering the number of nodes in the simulation.

Initial Budget	1
Existence of Super Nodes and Groups	True
Maximum Group Size	500
Resource Availability	100%
Maximum Price of Resources	5
Signature of Messages	True
Progressive Reputation and Budget	True
Periodic Reputation Adjustment	True
Collusion Detection	True
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	10%
Probability of Overbooking	25%
Probability of Praising/Badmouthing	25%
Probability of Collusion	25%

Table 16 – Full Incentive Mechanisms: Simulation Parameters

Joins	2444	Reputation (Correct Nodes)	Average	60
Departures	101		Standard Deviation	14
Expelled Nodes	308	Budget (Correct Nodes)	Average	28
Final Number of Nodes	2036		Standard Deviation	27
Super Nodes	5	Successful Exchanges (Correct Nodes)	Average	7.98
Faulty Nodes	27	Reputation (Faulty Nodes)	Average	36
Exchanges Attempted	19294	Budget (Faulty Nodes)	Average	3.0
Exchanges Failed	850	Successful Exchanges (Faulty Nodes)	Average	0.9
Elapsed Time: 108 seconds				

Table 17 – Full Incentive Mechanisms: Simulation Results

Using our incentive mechanisms it was possible to accurately detect nodes misbehavior. A large number of faulty nodes were expelled, 90% of all detected. Even the nodes that were not withdrawn could only perform, in average, approximately one exchange, one third of the amount correct nodes achieved. The elapsed time allows us conclude that the only significant impact on performance caused by our incentive mechanisms is due to the cryptographic signature of messages, all the other security measures have a negligible impact on the system.

6.3.6 Simulation 6 – High Number of Faulty Nodes

With this simulation we intend to measure the impact malicious users have over the network, determining the consequences in throughput to correct users. Not only will the number of faulty nodes be higher, but those users will also be more aggressive, attacking the network more frequently.

Initial Budget	1
Existence of Super Nodes and Groups	True
Maximum Group Size	500
Resource Availability	100%
Maximum Price of Resources	5
Signature of Messages	True
Progressive Reputation and Budget	True
Periodic Reputation Adjustment	True
Collusion Detection	True
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	50%
Probability of Overbooking	50%
Probability of Praising/Badmouthing	10%
Probability of Collusion	50%

Table 18 – High Number of Faulty Nodes: Simulation Parameters

Joins	2382	Reputation (Correct Nodes)	Average	62
Departures	105		Standard Deviation	14
Expelled Nodes	531	Budget (Correct Nodes)	Average	39
Final Number of Nodes	1746		Standard Deviation	27
Super Nodes	5	Successful Exchanges (Correct Nodes)	Average	8.6
Faulty Nodes	621	Reputation (Faulty Nodes)	Average	39
Exchanges Attempted	17823	Budget (Faulty Nodes)	Average	4.8
Exchanges Failed	822	Successful Exchanges (Faulty Nodes)	Average	2.2
Elapsed Time: 107 seconds				

Table 19 – High Number of Faulty Nodes: Simulation Results

We verify that the presence of malicious users does not have a major impact on the performance of the system, resulting only in a decrease of the total number of exchange attempts and an increase of failures during transactions. The number of successful exchanges by correct nodes even increases as a result of the low reputation values faulty nodes can obtain. Taking advantage of the fact that it is easier to detect malicious nodes, accurate users can select trustworthy neighbors to interact with, avoiding potential failures during the transaction.

The bonuses awarded to nodes that report incorrect behavior are also reflected on the final budget and reputation of correct nodes.

6.3.7 Simulation 7 – Scarce Resources

By simulating resource scarcity we can test the process of dissolving groups and the effects of this situation on the number of successful exchanges. To better analyze the behavior of the application, we reduce the maximum size of groups.

Initial Budget	1
Existence of Super Nodes and Groups	True
Maximum Group Size	100
Resource Availability	30%
Maximum Price of Resources	5
Signature of Messages	True
Progressive Reputation and Budget	True
Periodic Reputation Adjustment	True
Collusion Detection	True
Probability of Adding a Node	10,5%
Probability of Removing a Node	0,5%
Probability of Exchange of Resources	89%
Percentage of Faulty Nodes	10%
Probability of Overbooking	25%
Probability of Praising/Badmouthing	10%
Probability of Collusion	10%

Table 20 – Scarce Resources: Simulation Parameters

Joins	2344	Reputation (Correct Nodes)	Average	60
Departures	111		Standard Deviation	14
Expelled Nodes	73	Budget (Correct Nodes)	Average	29
Final Number of Nodes	2160		Standard Deviation	28
Super Nodes	24	Successful Exchanges (Correct Nodes)	Average	7.53
Faulty Nodes	156	Reputation (Faulty Nodes)	Average	38
Exchanges Attempted	17266	Budget (Faulty Nodes)	Average	3.0
Exchanges Failed	747	Successful Exchanges (Faulty Nodes)	Average	2.18
Elapsed Time: 109 seconds				

Table 21 – Scarce Resources: Simulation Results

By reducing the availability of resources, we impose greater effort on the users, requiring them to perform multiple requests until they can satisfy their objectives. The decreased number of successful exchanges, accompanied by the decrease on the available budget, allows us to measure the impact on the system of the scarcity of resources, intensified by the constraints of smaller groups. The difficulty to find suitable partners for transactions drives users to demand resources from peers with lower reputation, increasing their throughput.

Looking at the final results, we consider them to be positive, considering that we simulated a situation where 70% of the requests for resources are turned down coupled with fewer nodes in each group, resulting only in a 9% drop of attempted exchanges and average transactions.

7

Conclusions & Future Work

As the computational power of personal computers grew and tasks were performed faster, the amount of time users make full use of their resources diminished. Realizing this, efforts were made to optimize the usage of resources and minimize the need for expensive high-end computers. While in controlled environments policies can easily be enforced to guarantee the correct behavior of users, it has been a challenge to develop a reliable system to share resources in the Internet, among thousands of anonymous peers.

With this problem in mind, the main objective of this thesis was the development of effective and reliable incentive mechanisms for peer-to-peer applications. After careful and comprehensive research, we strived not to ignore any relevant contribution made by other researchers in this area of study. To achieve better results and a more balanced network, we decided to implement a solution based on multiple groups of users, with a 2-tier hierarchy – Super Nodes and Regular Nodes.

The architecture of the system guarantees scalability by dividing users into groups, which are easier to manage by the Super Nodes. The existence of these nodes is necessary to oversee the behavior of nodes in the network, acting as a judge to disputes and enforcing all the defined policies, indispensable to maintain the correct functioning and high throughput of the system.

Preventing attacks to the system that could compromise its stability and drive users away was the major objective. Bearing in mind that the most prominent threat comes from the well-known Sybil Attack, we focused on this problem and came upon documented processes to thwart it. We resorted to a computational test based on hash reversal and, after several simulations and comprehensive analysis, defined the exact parameters adequate to an effective initial cost which signifies an acceptable burden to legitimate users while it is also a considerable barrier to malicious ones.

The entire system is based upon two different but related concepts: currency and reputation. The first one consists in a payment method, obtained by providing services to other peers and consumed when the node requests resources from neighbors on the network. The value of resources can be adjusted dynamically, accounting for rush hour effects and other fluctuations on demand and offer of resources.

On the other hand, reputation intends to symbolize if the user is reliable, based on past interactions between that user and all other nodes. The reputation of nodes becomes a central indicator for neighbors when choosing transaction partners, allowing them to overlook and avoid potential hazards and complications, which would delay their access to the resources needed.

Super Nodes play a vital role handling these two values, acting as brokers to the group they are managing and assigning themselves the rewards and penalties when necessary. These nodes maintain all the information needed to simplify procedures and thwart any attempts to deceive the reputation and currency schemes.

We developed our incentive mechanisms on top of the Pastry overlay network, making it very easy to adapt to other applications already using the same framework, and hoping that, by resorting to a well-known structure, we can benefit the performance and widespread of these systems.

Each portion of our solution was tested individually and then as a whole using the Peersim simulator and fine-tuned to achieve the best results possible. The many configurable parameters we could pass on to the simulation engine made possible complex and detailed simulations as well as the construction of diverse possible scenarios to replicate known attacks. After this process, we feel comfortable stating that our solution is reliable and correctly addresses the major problems posed to every peer-to-peer network.

7.1 Future Work

We have established that the Super Nodes are responsible for guaranteeing the correct functioning of the system by identifying potential threats and making sure all nodes comply with the requirements. However, we have not yet defined the entity responsible for controlling the correct operation of the Super Nodes. On other words, as soon as any user becomes a Super Node, the system has no way to assure those users are not taking advantage of the position attained to benefit them or collude with others.

The approach we consider to be more correct is to develop a way by which Super Nodes can judge one another and, when needed, achieve the necessary quorum to expel a malicious Super Node.

Mimicking a principle some countries apply in their political system, a limitation on mandates can also be set, which would mean that Super Nodes could only execute that post during a limited amount of time, being automatically substituted once their validity expired.

References

[1] Bharambe, A. R., Herley, C., & Padmanabhan, V. N. (2006). Analyzing and improving a BitTorrent networks performance mechanisms. 25th IEEE International Conference on Computer Communications (INFOCOM 2006). Orlando.

[2] BitTorrent, http://www.bittorrent.com/

[3] Borisov, N. (2006). Computational Puzzles as Sybil Defenses. Sixth IEEE International Conference on Peer-to-Peer Computing. Cambridge: IEEE Computer Society.

[4] Castro, M., Druschel, P., Ganesh, A., Rowstron, A., and Wallach, D. (2002). Secure routing for structured peer-to-peer overlay networks. In Fifth Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, Massachusetts.

[5] Cheng, A., & Friedman, E. (2005). Sybilproof Reputation Mechanisms. ACM SIGCOMM workshop on Economics of peer-to-peer systems.

[6] Cohen, B. (2003). Incentives Build Robustness in BitTorrent.

[7] Douceur, J. R. (2002). The Sybil Attack. 1st International Workshop on Peer-to-Peer Systems.

[8] eBay, http://www.ebay.com/

[9] Fu, Y., Chase, J., Chun, B., Schwab, S., & Vahdat, A. (2003). Sharp: An architecture for secure resource peering. 19th ACM Symposium on Operating Systems Principles (SOSP).

[10] Guha, R., Kumar, R., Raghavan, P., & Tomkins, A. (2004). Propagation of Trust and Distrust. International World Wide Web Conference.

[11] Halderman, J. A., & Waters, B. (2007). Harvesting Verifiable Challenges from Oblivious Online Sources. Conference on Computer and Communications Security. NY: ACM.

[12] Hansen, J., Christiansen, E., & Jul, E. (2006). The Laundromat Model for Automatic Cluster Computing. 3rd IEEE International Conference on Autonomic Computing.

[13] Jesi, G. P. (2007). Secure Gossiping Techniques and Components. PhD Thesis, University of Bologna, Department of Computer Science, Bologna.

[14] Jurca, R., & Faltings, B. (2003). An Incentive Compatible Reputation Mechanism. IEEE Conference on E-Commerce.

[15] Kamvar, Sepandar D., Schlosser, Mario T., Garcia-Molina, H. The Eigentrust algorithm for reputation management in P2P networks. Proceedings of WWW2003, May 20-24, 2003, Budapest, Hungary.

[16] KaZaA, http://www.kazaa.com

[17] Lai, K., Feldman, M., Stoica, I., Chuang, J. Incentives for Cooperation in Peer-to-Peer Networks. Workshop on Economics of Peer-to-Peer Systems, 2003.

[18] Legout, A., Urvoy-Keller G., Michiardi, P. Rarest First and Choke Algorithms Are Enough. Proceedings of ACM SIGCOMM/USENIX IMC'2006, October 25--27, 2006, Rio de Janeiro, Brazil.

[19] Levine, B. N., Shields, C., Margolin, N. B. (2006). A Survey of Solutions to the Sybil Attack

[20] Liang, J., Kumar, R., Ross, K. (2003) Understanding KaZaA.

[21] Marti, S., Garcia-Molina, H. (2005). Taxonomy of trust: Categorizing P2P reputation systems.

[22] Ngan, Tsuen-Wan “Johnny”, Wallach, Dan S., Druschel, Peter. Enforcing Fair Sharing of Peer-to-Peer Resources.

[23] Parker, A. The True Picture of Peer-to-Peer File-Sharing. IEEE 10th Int’l. Workshop on Web Content Caching and Distribution, 2005.

[24] PeerSim, http://peersim.sourceforge.net

[25] Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2007). Do incentives build robustness in BitTorrent? In NSDI'07, Cambridge, MA.

[26] Quercia, D., Hailes, S., & Capra, L. (2007). Lightweight Distributed Trust Propagation. 7th IEEE International Conference on Data Mining (ICDM).

[27] Rivest, R. L., Shamir, A., and Wagner, D. A. (1996). Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684.

[28] Rowstron, A., & Druschel, P. (2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science* (2218).

[29] Search for Extraterrestrial Intelligence (SETI), http://setiathome.berkeley.edu/

[30] Sirivianos, M., Park, J. H., Chen, R., & Yang, X. (2007). Free-riding in BitTorrent Networks with the Large View Exploit. *IPTPS.* Bellevue.

[31] Tritilanunt, S., Boyd, C., Foo, E., Nieto, J. (2007). Toward Non-Parallelizable Client Puzzles. 6th International Conference , CANS 2007: Cryptology and Network Security.

[32] Tsang, P. P., & Smith, S. W. (2008). Combating Spam and Denial-of-Service Attacks with Trusted Puzzle Solvers. Information Security Practice and Experience, 4th International Conference. Sydney: Springer.

[33] Vishnumurthy, V., Chandrakumar, S., & Sirer, E. (2003). KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. Workshop on the Economics of Peer-to-Peer Systems. Berkeley, California.

[34] Walsh, K., & Sirer, E. G. (2006). Experience with an Object Reputation System for Peer-to-Peer Filesharing. Symposium on Networked System Design and Implementation (NSDI).

[35] Waters, B., Juels, A., Halderman, J. A., & Felten, E. W. (2004). New client puzzle outsourcing techniques for DoS resistance. 11th ACM Conference on Computer and Communications Security (pp. 246 - 256). Washington, DC: ACM.

[36] Zhang, Z., Hu, Y., Midkiff, S. (2006). CycleMeter: Detecting Fraudulent Peers In Internet Cycle Sharing. Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. Tampa, Florida.