

Incentive Mechanisms in Peer-to-Peer Networks

Pedro Dias Rodrigues

Carlos Ribeiro

Luís Veiga

INESC ID Lisboa / Instituto Superior Técnico – Universidade Técnica de Lisboa
Rua Alves Redol, 9
1000 Lisboa, Portugal
pedrorodrigues@ist.utl.pt, carlos.ribeiro@ist.utl.pt, luis.veiga@inesc-id.pt

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 absence of a central authority and the fact that each user can act both as a client and as a server. Thus, a need for self-regulation arises in order to guarantee that every user contributes to the system. 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 incentive mechanisms take into account the attacks these systems are subject to, as well as the system's structure, 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, most 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: *Incentive Mechanisms, Reputation, Currency, Sybil Attack, Computational Puzzles*

1. Introduction

The exchange of information between users has always been the main purpose of any computer network. While applications using Peer-to-Peer architectures have long been used in laboratories and other controlled environments to share computational resources, they have yet to achieve widespread utilization, mainly due to the difficulty to effectively control user behavior. Our work aims to contribute to the faster dissemination of these systems, allowing ordinary users outside controlled environments to take advantage of their benefits.

A central server is always a target for attacks and cause for failures; this is why decentralization is vital. But decentralizing also increases 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 takes place. 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.

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.

Several incentive mechanisms have been developed, according to immediate preoccupations and specific objectives. A categorization of reputation systems is presented in [13], and identifies three main components in these systems, namely: 1) information gathering, 2) scoring and ranking, and 3) response.

The main issues when considering effective incentive mechanisms applied to peer-to-peer networks, according to [8], are: Self-policing, which means that there is no central authority and that peers themselves must enforce 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.

Peer-to-peer systems have tried to minimize the risk of system disruption motivated by an attack to a central authority, enabling each user to make all necessary decisions and guaranteeing that the system remains in operation, even if most of the nodes fail. The downside is obvious: without a central authority, trust becomes a fundamental problem to deal with.

In this paper we present a new approach, combining robust incentive mechanisms resistant to common attacks with an architecture designed to support those mechanisms while optimizing the performance of the network. We demonstrate, simulating user interactions in PeerSim, that it is possible to enforce resource-sharing policies and to guarantee resources are only available to users who contribute to the system, without a significant impact on performance, over a large scale network. The possibility to have a complex resource economy, with fluctuating resource prices, is also introduced to better satisfy specific needs from application developers.

1.1. Pastry Overlay Network

The Pastry overlay [18] provides the basis for the development of our peer-to-peer network. 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.

The expected number of hops to propagate a message is $O(\log N)$, where N is the number of nodes in the network but, 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 only needs to know the address of a Pastry node, to which it sends a "join" 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 are no longer able to contact it.

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

1.2. PeerSim

PeerSim [15] is a simulator, capable of reproducing the operation of a large P2P network, with several thousand users exchanging resources simultaneously, as well as constant arrival and departure of peers. We chose to use PeerSim since it 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 used.

1.3. Organization of this paper

Section 2 presents an overview of peer-to-peer systems and their main characteristics, using some of the most well known applications as examples. It also encompasses information about relevant contributions to this field and discusses the contributions brought by different approaches to this problem

We present our solution in Section 3, highlighting major features and architectural decisions, detailed and explained. We then proceed to simulate possible implementation scenarios and analyze the effectiveness of our solution from different viewpoints, plus considerations about the impact on performance in Section 4. Finally, in Section 5, we present our conclusions and improvement possibilities.

2. Peer-to-Peer Systems

Designed and implemented in 2001, the BitTorrent protocol [2] is used to distribute large amount of data without requiring enormous resources from the hosts. 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 [3], have proven to be fragile and inefficient [1, 16, 19].

Also developed in 2001, KaZaA [9] adopted a different strategy. 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 [12]. 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 these nodes to act as small servers to organize ON in groups and compile information about the resources being shared by ON assigned to them. 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.

2.1. Currency and Reputation

Resource exchange systems can be as simple barter economies, with exchanges between users without any form of payment, the resources themselves being the currency. However, this approach is too limiting as introducing currency translates into users being able to share resources and accumulating currency for future use. Another aspect can be brought up: should every resource have the same value? Some resources are scarcer than others, and there are "rush hours", when the resource pool is insufficient to cope with every request. 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.

As we have pointed out, knowing which users one can trust is crucial and the most popular solution to address this problem is attributing reputation to each user. The reputation of an entity can be described as the result of the level of trust peers place on that entity.

The website eBay [5] is probably the most famous example. When you buy a product from another user you are asked to rate the seller according to the quality of the service. If a user receives good reviews, it adds to his reputation and he becomes a renowned seller, attracting more potential buyers.

Pragmatically, a user can only trust itself and users he has interacted with. However, this is very restrictive, therefore a process of Reputation Distribution or Trust Distribution is necessary. In [6], authors develop and test a framework for propagating trust and distrust, changing the focus from the number of voters to relationships developed over time, much like in real world, where you trust the opinion of a long acquaintance more than that of ten strangers. The burden of the trust propagation mechanism must be taken into account when designing the system, so that its impact does not hinder the positive effects of having an incentive mechanism. The work in [17] presents a mechanism for lightweight and distributed trust

propagation, with the intent of implementation in low capacity devices, such as mobile phones, also providing good insight on this subject.

2.2. Robustness and Security

Thinking about the robustness and security of a system leads to immediate considerations regarding the capacity to identify its users, but the main problem with electronic identities is that a physical user can create as many virtual identities as he wishes.

The Sybil attack, as described by Douceur [4], 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. Various attacks against computer systems, such as spam and denial of service, are conducted creating multiple virtual identities, the low cost of creating virtual identities being the major incentive for attackers.

Preventing Sybil attacks has been the motivation for countless works and [11] presents an interesting survey of how existing applications cope with the existence of such attacks. It shows that, although the attack is prominent, several applications are completely vulnerable and did not adopt adequate security mechanisms.

Collusion is also a major problem when used by attackers to improve each other's reputation in the system, gaining the trust of other users and profiting from that situation.

In his PhD Thesis, Jesi [7] 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. In [14] authors develop a mechanism to prevent collusion based on peer auditing. The idea is to compel users to publish signed records of their actions, for example using cryptographic keys, which can then be audited by other peers.

The robustness of the system can only be ensured if selfish and untrustworthy users, who do not contribute to the system and take advantage of other peers, are detected and banned. In [23], authors describe the design and implementation of a solution to identify fraudulent behavior from users in cycle sharing systems. This application focuses on situations where nodes make available only a part of the promised resources, namely using only a fraction of the computational power promised.

Users joining the network have no past interactions to prove their commitment to the system and therefore their reputation data is non-existent.

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 [10]. The idea is to compile information on interactions with recently joining peers, determining if the system has a high rate of whitewashing, to decide how to cope with them.

To counter attempts of Sybil attack computational puzzles can be used. If every user has to solve a puzzle in order to enter the network, it creates a cost that not all attackers are comfortable with. Puzzles must be difficult to solve, requiring large amount of computation, but easy to verify.

One of the most straightforward solutions is presented in [22] 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.

3. System Design

Our aim was always the balance between a currency and reputation-based system capable of delivering proper incentives to users, and the impact that system and all implemented security measures have on the network performance. These mechanisms are intended to work in a decentralized application, 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 not become a massive burden to users, incapacitating them from producing useful computation and, to achieve this compromise, we had to restrict the number of users assigned to each broker.

In our solution, developed on top of the Pastry framework and Ginger [21], 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 having an assigned broker. 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 manage and control by a single broker and, on the other hand, a group with a too small number of peers cannot satisfy those users' needs. Additionally, the existence of brokers means that our architecture can be classified as partially hierarchical and structured.

To bootstrap the system, a small set of trustworthy users is needed, to act as super nodes. 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 new users to enter the network they have to contact a user already in the system, meaning that some sort of contact outside the application is required.

After acceptance into the network, a new unique ID is created and assigned to the user; the broker responsible for that group contacts the new user, presenting its credentials, and sends its neighbors list so that the new node can start contacting other peers.

Figure 1 exemplifies the architecture of the system, with four virtual groups of nodes, each one with an assigned broker or Super Node.

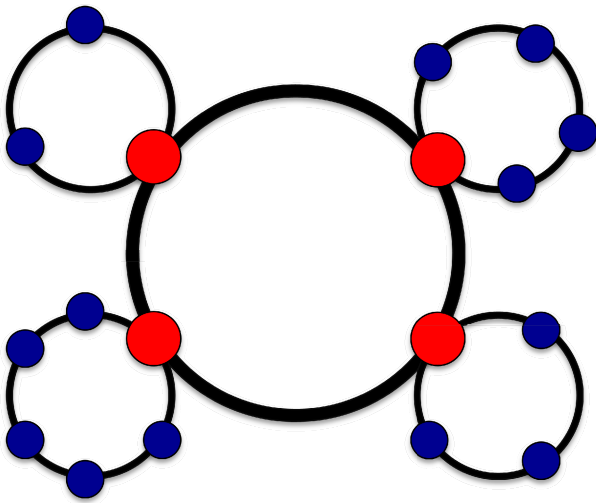


Figure 1 – Architecture of the system

The threshold for group size can be set accordingly to the computational capability of the broker or as a constant value, considered reasonable. When a group is growing and the size limit is about to be attained, the most reputed user should be appointed as the broker for a new group. Half of the users in that group, chosen randomly, are transferred to the new group.

When a group is not performing as expected, with the demand for resources far greater than the supply inside that group, either the group is dissolved and the users scattered among other groups, or the group is merged with another group. The second option is more effective, with the possibility of brokers exchanging information, conducting the stalled group towards a group with a large amount of unused resources.

3.1. Currency and Reputation

The brokers are the only users capable of introducing currency in the system and are responsible for registering an additional amount of currency to the most reputed users upon the entrance of a new peer. This 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, guaranteeing that there is no shortage of currency and no illegitimate transfers of currency, allowing users to achieve and maintain a dominant position.

Currency and reputation are essential to the well functioning of the network and we have come to the conclusion, based on existing systems, that using both as two separate concepts is the most adequate manner to cope with all the problems presented. In a resource exchange, currency is passed from the buyer to the seller and, as a result, the seller becomes wealthier and the buyer poorer. But, if the transaction was carried out

correctly, both users see their reputation rise, since every correct transaction is positive for the system. A user with little need for resources and who is a great contributor to the system will see the amount of currency in his possession rise indefinitely, and so a limit to the reputation is set to guarantee the balance of the system and the availability of resources, thus fostering collaboration.

The reputation of users is periodically adjusted downwards by the brokers to promote activity; both the time interval between adjustments and the reputation points withdrawn can be defined according to the nature of the application and the number of users using it. 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.

3.2. Robustness and Security

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, however it implies a way to share a key between two peers in a secure manner, consequently the use of asymmetric cryptography is more appropriate. The responsibility of creating the pair of cryptographic keys relies on each user and the size of the cryptographic keys is set as 512-bit for performance reasons, acceptable since the average time a user spends in the system is relatively low.

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.

Our currency allocation policy deals with the issue of users using initial funds to acquire resources without sharing, but 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. It is also possible for users to report malicious peers, allowing the brokers to fine them, as long as they present proof of that claim, namely signed messages to corroborate the accusation.

Even so, to protect the application from Sybil attacks we use computational puzzles, forcing users to perform a hash reversal. The complexity of the puzzles was adjusted so that legitimate users are able to access the network, while possible attackers have to deal with a heavy cost of entrance.

When a user joins the network, the only option 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. We rely on the broker to maintain a record of recent transactions involving recently arrived users so that other peers can use that information to decide the best course of action.

3.3. Interactions

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 Super Node contacts the new node to convey its ID and also includes the cryptographic puzzle, described in section 3.4, to be solved. The joining user becomes a member of the network only when the correct solution is sent back to the broker and verified.

When a user intends to acquire resources from other members of the application, he starts by selecting a neighbor in the same group and, if it is the first interaction with that peer, he can ask the broker for information about the reputation and the public cryptographic key. The request for resources and promise of delivery are cryptographically signed to guarantee authenticity and non-repudiation. 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 already have information about each other from previous interactions, the broker needs only to get involved in this last step, with minimal impact on performance.

As a protection against collusion between peers and also to prevent abuse from peers with a dominant position, the broker stops awarding reputation if a possible attack is detected. To achieve this, brokers maintain a limited record of past transactions and are able to identify both multiple consecutive transactions between users and massive resource requests from a single user. Thresholds can be defined taking into account the size of user groups, making sure the number of false positive detections is negligible.

An additional interaction with the broker may exist if one of the exchange participants has only recently arrived to the network. In this case, the broker is able to provide information about newcomers in the system, used to decide if the transaction continues with the selected neighbor.

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. The offended user can then contact the broker, using the signed messages exchanged with the malicious user as evidence to corroborate the claim. We provide a framework and 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. 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.

3.4. Computational Puzzles

Computational challenges based on intensive computation are one of the most efficient ways to combat Sybil attacks. 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*, through brute force search and multiple hashing actions. 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 will not be severely burdened.

The downside of using hash reversal challenges lies in the method of solving these challenges, which can be parallelized. Nevertheless, and as pointed out in [20], even parallelizable puzzles are effective against most attacks, with the advantage of simplicity in construction and verification.

We chose to use an alphanumeric *nounce* with a length of 512 bits, and defined n as 40 bits. The elapsed time to solve the challenges averages 11 seconds according to simulations on a MacBook Pro laptop with an Intel Core 2 Duo processor running at 2.2GHz, with 4GB of DDR2 667MHz RAM. We consider this value reasonable for a legitimate node, since it is only performed once. However, for attackers wishing to deploy multiple identities in the application, the required time is very high, even if the attacker has access to high-end technology and fast processors.

4. Simulation and Evaluation

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 and we used PeerSim to accomplish that. 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.

Depending on the parameters we define for the simulation, 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.

Table 1 lists all relevant parameters passed to the simulator and a short explanation for each one.

We decided to keep the initial reputation of each node constant, as well as the penalties for incorrect behavior and awards for successful exchanges, so that the number of detected infractions needed to expel a node is the same in both simulations.

Parameter	Purpose	Scenario 1	Scenario 2
Initial Budget	Sets the currency amount awarded to each node	100	1
Initial Reputation	Sets the initial reputation of each node	50	50
Maximum Group Size	The maximum number of nodes simultaneously in a group	N/A	500
Maximum Cost of a Resource	The cost of resources has a direct impact on the number of exchanges a user can accomplish	5	5
Probability of a Node Being Faulty	Adjusts the number of faulty nodes in the system	10%	10%
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 100%	10,5%	10,5%
Probability of a Node Leaving the Network		0%	0%
Probability of a Node Asking Other Peers for Resources		89,5%	89,5%
Probability of Faulty Node Making Fake Requests or Acceptances	Parameters to adjust the behavior of faulty nodes, making them more or less aggressive	10%	10%
Probability of a Faulty Node Badmouthing or Praising		10%	10%
Probability of Collusion Between Faulty Nodes		25%	25%
Probability of Optimistically Unchoking a New Peer	This parameter is used if a node chooses not to consult the broker	10%	10%
Existence of Super Nodes	Enables or disables the existence of super nodes and groups	No	Yes
Use of Signatures	Enables or disables the use of cryptographic signatures	N/A	Yes
Progressively Award Currency and Reputation	Enabling this option we can define a low initial budget, obliging nodes to share resources in order to accumulate currency	N/A	Yes
Periodic Reputation Adjustment	Periodically decreasing their reputation can force nodes to share resources, instead of just using accumulated currency	N/A	Yes
Collusion Detection	The broker of each group maintains a record of transactions to verify possible collusion between nodes	N/A	Yes

Table 1 – Simulation Parameters

4.1. Simulation Results

In this section we present the average results of our two most relevant simulations. The first scenario was designed to measure not only the impact malicious users have on the system, but mainly the advantages they have over complying nodes. Malicious nodes act in order to maximize their gain, not with the objective of disrupting the system or damage other peers. This situation seemed more realistic to us, as most users have nothing to gain from the destruction of the network.

On the other hand, the second scenario intends to measure the effectiveness of our incentive mechanisms,

comparing the results achieved by the malicious nodes with the ones from the first scenario. It is also very important to determine the reduction of performance, establishing a relation between cost of deploying these mechanisms and the profit obtained.

All the simulations were run on a MacBook Pro with an Intel Core 2 Duo 2.2GHz processor and 4GB of DDR2 667MHz RAM.

Scenario 1 – To verify the consequences of introducing malicious nodes in a peer-to-peer application without any incentive mechanisms, we programmed 10% of the nodes to perform attacks in order to maximize their profits. These nodes reject all

requests for resources, collude to improve their reputation and utilize the initially awarded currency to obtain resources from other users, simply creating new identities when needed.

The results obtained are compiled in Table 2.

Number of Nodes	2249
Expelled Nodes	0
Super Nodes	0
Faulty Nodes	211
Exchanges Attempted	21960
Exchanges Failed	0
Reputation (Correct Nodes)	72
Budget (Correct Nodes)	100
Successful Exchanges (Correct Nodes)	8.27
Reputation (Faulty Nodes)	74
Budget (Faulty Nodes)	109
Successful Exchanges (Faulty Nodes)	17.14
Elapsed Time	45 seconds

Table 2 – Zero Incentive Mechanisms

Scenario 2 – We then proceeded to simulate the behavior of the network combining all the incentive mechanisms we propose, and defined a threshold of 500 users per group to verify if the variation of both demand and offer of resources had any effect on the overall throughput.

The average results are listed in Table 3.

Number of Nodes	2036
Expelled Nodes	308
Super Nodes	5
Faulty Nodes	27
Exchanges Attempted	19294
Exchanges Failed	850
Reputation (Correct Nodes)	60
Budget (Correct Nodes)	28
Successful Exchanges (Correct Nodes)	7.98
Reputation (Faulty Nodes)	36
Budget (Faulty Nodes)	3
Successful Exchanges (Faulty Nodes)	0.9
Elapsed Time	108 seconds

Table 3 – Full Incentive Mechanisms

4.2. Simulation Analysis

After analyzing the results from the various simulations, focusing mainly on the most significant ones described in the last section, we can draw several conclusions. The most important is that the throughput of the application is not reduced by the introduction of the incentive mechanisms. The higher elapsed time to achieve the same number of resource exchanges is caused by the operations with cryptographic keys, while other incentive mechanisms have little or no influence. However, if we take into account that the effort lies mostly on creating the cryptographic keys and that it will be divided among the peers, the impacts on performance do not seem severe.

The success of attacks from faulty nodes is greatly reduced when the incentive mechanisms are in place, reducing the completed exchanges of malicious users from twice as much as the accomplished by legitimate nodes to just 11%. Guaranteeing that defective users were detected and could not have an advantage over legitimate ones was the main objective of this work, and we consider the results highly pleasing. It is worth mentioning that one of the incentive mechanisms, a small initial budget and progressive introduction of currency, results in a slow start even for honest users. For that reason, the number of successful transactions may appear to be weak but, as the number of interactions increases and time advances, that initial impact is softened.

5. Conclusion

The main objective of our work was the development of effective and reliable incentive mechanisms for peer-to-peer applications, to promote the sharing of resources in uncontrolled environments. To achieve better results and a more balanced network, we opted for a solution based on groups of users, with a 2-tier hierarchy – Super Nodes and Regular Nodes. This architecture guarantees scalability by dividing users into groups, which are easier to manage by the Super Nodes, whose responsibility is to oversee the behavior of nodes in the network, acting as a judge to disputes and enforcing all the defined policies.

Bearing in mind that the most prominent threat comes from the Sybil Attack, we resorted to a computational test based on hash reversal and based the entire system upon two different but related concepts: currency and reputation. The first one consists in a payment method, used in resource transactions, while reputation intends to symbolize if the user is reliable based on past interactions. 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.

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.

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

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

[5] eBay, <http://www.ebay.com/>

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

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

[8] 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.

[9] KaZaA, <http://www.kazaa.com>

[10] 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.

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

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

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

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

[15] PeerSim, <http://peersim.sourceforge.net>

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

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

[18] 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).

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

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

[21] Veiga, L., Rodrigues, R., Ferreira, P. (2007). GiGi: An Ocean of Gridlets on a "Grid-for-the-Masses". 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007) (pp. 783-388), Rio de Janeiro, Brazil.

[22] 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.

[23] 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.