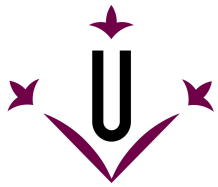


Universitat de Lleida

Development of an incentive and scheduling mechanism for a Peer-to-Peer computing system

Josep Maria Rius Torrentó

Escola Politécnica Superior
Departament d'Informàtica i Enginyeria Industrial

Development of an incentive and scheduling mechanism for a Peer-to-Peer computing system

Memòria presentada per obtenir el grau de
Doctor per la Universitat de Lleida
per

Josep Maria Rius Torrentó

Dirigida per

Dr. Fernando Cores Prado **Dr. Francesc Solsona Tehàs**
Universitat de Lleida

Programa de doctorat en Enginyeria

Lleida, novembre de 2011

Abstract

Peer-to-Peer (P2P) computing offers new research challenges in the field of distributed computing. This paradigm can take advantage of a huge number of idle CPU cycles through Internet in order to solve very complex computational problems. All these resources are provided voluntarily by millions of users spread over the world. This means the cost of allocating and maintaining the resources is split and assumed by each owner/peer. For this reason, P2P computing can be seen as a low-cost alternative to expensive super-computers.

Obviously, not every kind of parallel application is suitable for a P2P computing environment. Those with high communication requirements between tasks or with high QoS needs should still be performed in a Local Area Networking (LAN) environment. Otherwise, problems with huge computational requirements that can be easily split into millions of independent tasks are suitable for P2P computing, especially as solving these problems with a super-computer would be extremely expensive.

One of the most critical aspects in the design of P2P systems is the development of incentive techniques to enforce cooperation and resource sharing among participants. Incentive policies in P2P distributed computing systems is a new research field that requires specific policies to fight against malicious and selfish behavior by peers. Encouraging peers to collaborate in file-sharing has been widely investigated but, in the P2P computing field, this issue is still at a very early stage of research. Furthermore, the dynamics of peer participation are an inherent property of P2P systems and critical for design and evaluation. This further increases the difficulty of P2P computing.

Another critical aspect of P2P computing systems is the development of scheduling techniques to achieve an efficient and scalable management of the

computational resources. Unlike file-sharing, based on such immutable resources as files, the mutable ones, such as CPU and Memory are the principal resources involved in P2P computing. Inside the scheduling field, P2P computing can be seen as a particular variant of Grid computing. In a similar way as with the incentive polices, an extensive list of publications can be found that study the scheduling problems for distributed computing, such as Clusters or Grid computing, but few of these focus on P2P computing. For this reason, the scheduling problem in this kind of network is a field that still requires research in depth.

In this thesis we propose a Distributed Incentive and Scheduling Integrated Mechanism (DISIM) with a two-level topology and designed to work on large-scale distributed computing P2P systems. The low level is formed by associations of peers controlled by super-peers with major responsibilities in managing and gathering information about the state of these groups. Scalability limitations on the first level are avoided by providing the mechanism with an upper level, made up of super-peers interconnected through a logical overlay.

Regarding incentives, we propose a mechanism based on credits with a two-level topology designed to operate on different platforms of shared computing networks. One of the main contributions is a new policy for managing the credits, called Weighted, that increases peer participation significantly. This mechanism reflects P2P user dynamics, penalizes free-riders efficiently and encourages peer participation. Moreover, the use of a popular pricing strategy, called reverse Vickrey Auction, protects the system against malicious peer behavior. Simulation results show that our policy outperforms alternative approaches, maximizing system throughput and limiting free-riding behavior by peers.

From the scheduling point of view, the low-level scheduler takes user dynamism into account and is almost optimal since it holds all the status information about the workload and computational power of its constituent peers. Our main contribution at the upper level is to propose three criteria that only use local information for scheduling tasks, providing the overall system with scalability. By setting these criteria, the system can easily, dynamically and rapidly adapt its behavior to very different kinds of parallel jobs in order to

achieve an efficient performance. The results obtained proved the efficiency of the overall model and the convergence with the best assignment, achieved with an ideal centralized policy with global information.

Acknowledgements

I would like to acknowledge to all the people who supported me during the course of this thesis. First and foremost, I want express my gratitude to my supervisors, Dr. Fernando Cores Prado and Dr. Francesc Solsona Tehàs, for their continued encouragement and invaluable suggestions during this work. They have patiently supervised every little issue, always guiding me in the right direction. Without their help, I could not have finished my thesis successfully.

Specially thanks are given to all the seniors from the Group of Distributed Computing (GCD) from the University of Lleida (UdL). I must say that has been a great experience working closely with such good people: Concepció Roig, Francesc Giné, Fernando Guirado, Josep Ll. Lèrida, Josep M. Solà, Valentí Pardo, Albert Saiz and Xavier Faus.

During my research, I have shared great moments with all my colleagues: Ivan Teixidor, Miquel Orobitg, Damià Castellà, Héctor Blanco, Anabel Usié, Alberto Montanyola and, foremost, Ignasi Barri, the person with whom I have studied and worked since setting foot in this University. Many thanks to all of them for the wonderful times we shared.

I also want to thank all the members of the Edinburgh Data-Intensive Research group of the University of Edinburgh. During three month last summer, I had the chance to visit this group and work with great people who made me feel at home. It was a great experience that marked me in many positive aspects. I would especially like to acknowledge Malkolm Atkinson and Jano van Hermet, for all their support, and Gary McGilvary, for welcoming me as if I were one more.

Last but not least, I would like to dedicate this thesis to all my friends and, foremost, to my closest family. Especially to my parents Jaume and Maria,

who have given me the freedom to make my own decisions. To my girlfriend Soraya for her emotional support. And to my aunt Montserrat, the person who, without knowing, introduced me into the wonderful world of computers.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Over recent years, Peer-to-Peer (P2P) systems have become an important part of Internet. Millions of users have been attracted to their structures and services. The popularity of P2P systems has speeded up academic research bringing together researchers from systems, networking and theory. The most popular P2P networks support file-sharing and content distribution but, most recently, many new applications are emerging in different fields, such as P2P computing, the research area of interest in this thesis.

In this chapter, the concept of distributed computing is introduced to the reader, explaining the most popular distributed systems with special emphasis on P2P computing networks. First, the difference between parallel computing and distributed computing -two concepts frequently confused- is introduced in section 1.1. Next, the most commonly used distributed systems are summed up, with special emphasis being placed on those that take advantage of Internet to reach as many resources as possible or to make these widely available. Among all these distributed systems, this thesis is based on P2P networks. For this reason, section 1.2 explains this kind of network in detail, highlighting their main features and classifying them according to three different criteria: the degree of decentralisation, the overlay architecture and their purpose.

The P2P distributed paradigm has been widely used in the research field of sharing files over Internet. Such well-known applications as Napster [nap06], Gnutella [KC04], Bittorrent [bit], Kazaa [ZF], eDonkey [edo], Emule [emu], Freenet [CSWH01], Skype [sky], Groove [Gro09] are examples of this. Nowa-

days, a similar research field in the P2P paradigm is arising, but where computational resources are shared instead. This new research field is called P2P computing and researches such fields as scheduling, collaborating incentives, fault tolerance, security, and so on. The most important resource shared among the nodes making up the P2P network is the CPU, although Memory and Network bandwidth are also being investigated. Generally speaking, we are interested in building cheap and efficient P2P systems by grouping computational resources across Internet to be used in the execution of parallel/distributed applications.

Once the reader is focused on the field of P2P computing, section 1.3 introduces the two critical aspects in the design and construction of these networks, which are also the research fields where the main contributions of this thesis are made: the incentive policies and the scheduling mechanisms. Next, once the main terminology and concepts that make up the context of this thesis are explained, we present its main challenges and motivations. Finally, this chapter ends with the main objectives of the dissertation and details of how this manuscript is structured.

1.1 Parallel and distributed computing

The terms "parallel computing", "distributed computing" or even "concurrent computing" have a lot of overlap and there is no clear distinction between them. Very often, the same system may be classified as both "parallel" and "distributed". Despite this, distributed systems can be defined as groups of computers connected through a network. Distributed computing refers to the means by which a single computer program is executed in more than one computer at the same time. The different elements and objects of a program are executed or processed using different computer processors in parallel. Parallel computing may be seen as a particular tightly-coupled form of distributed computing, and distributed computing may be seen as a loosely-coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria:

- In parallel computing, all processors have access to a shared memory.

Shared memory can be used to exchange information between processors.

- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.

Taking these definitions into account, the systems analysed on this thesis belong to the distributed computing group.

1.1.1 Distributed computing

Distributed computing is formally defined as "a computer system in which several interconnected computers share the computing tasks assigned to the system" [IEE91]. Such systems include computing clusters, Grids and global computing systems gathering computing resources from individual PCs over the Internet. There are many reasons to explain the continuous growth of distributed systems and distributed computing. For example, there are applications that require connecting several computers to exchange data across a network. Imagine an application where all the data is produced in one physical location but it is needed in some other locations.

The use of a distributed system in other applications would be beneficial for many reasons. For example, in comparison with a single high-end computer, it may be more cost-efficient to obtain better performance by using a cluster of several low-end computers to avoid Memory contention. As a distributed system has no single point of failure, it can also be more reliable than a non-distributed system. A distributed system may be even cheaper, easier to expand and manage than a supercomputing system with the same performance.

Nowadays, efforts have been directed towards the design of systems with low power consumption.Many studies [BGDG+10] show that energy savings can be achieved by taking advantage of the infra-utilized computational resources connected to Internet. This will be one of the most important challenges of this dissertation.

1.1.2 Distributed systems

Distributed systems are those in which components located in networked computers communicate and coordinate their actions only by passing messages [DKC05]. Those computing technologies have recently emerged as new paradigms for solving complex computational problems. These systems enable large-scale aggregation and sharing of computational, data and other geographically distributed computational resources. In recent years, many researchers have reported numerous advances and innovative techniques for such paradigms, from theoretic to application aspects. Another fact that has contributed strongly to the rapid development of large-scale applications in many fields of science and engineering is the continuous development of high-speed networks and more especially, Internet.

Inside the distributed systems category, one can find a wide range of systems. Next, those that are most popular and closely related to the field of interest of this thesis, i.e. distributed computing, are briefly explained.

- **Cluster computing:** A computing cluster or simply cluster is a local computing system consisting of a set of independent computers and a network interconnecting them. All the component subsystems of a cluster are locally supervised within a single administrative domain, usually residing in a single room and managed as a single computer system. The interconnection network employs a local area network (LAN), which links the cluster nodes and isolates them from the outside world. Moreover, the cluster nodes are interconnected through a single point, often using a switch. This kind of environment operates in a local manner. Taking into account that we will profit from the endless computing resources of Internet, we must continue investigating more distributed systems.

- **Grid computing:** Grid is a distributed paradigm which appeared many years ago as a real alternative to the expensive supercomputers for HPC, High Performance Computing (Distributed Supercomputing, High-Throughput Applications, Data-Intensive Applications, etc.). Grid computing is defined as "coordinated resource sharing and problem solving in large, multi-institutional virtual organizations" [KF98]. Grid com-

puting joins virtually a large number of separate computers, clusters or LANs of workstations connected by a network (usually the Internet) in order to provide the ability to perform many more computations at once than would be possible on a single computer, by splitting the serial application into smaller and replicable pieces. One feature of Grid environments that often limits their use in universities, companies or institutions, is the huge software (such as Globus Toolkit [Fos05]) behind expensive hardware and which is costly to maintain and manage. We are interested in cheaper solutions to provide a means to take advantage of Internet to form distributed environments for parallel/distributed computation. The term "grid computing" has been extended to embrace different forms of computing.

Grid research is one of the most important sources of inspiration for this dissertation. As we shall see, the literature in this field has been widely used for reference. More precisely, advances in Grid computing, Enterprise Grid and Economic Grid have been used as guidelines in this thesis for the proposal for new scheduling and incentive mechanisms to increase resource collaboration by permitting the use of the computing resources of companies, organizations, public institutions and individuals, and so on, connected through Internet.

- **Cloud computing:** Cloud computing has rapidly spread in recent times. One of the first known definitions of the term Cloud Computing was by Prof. Ramnath K. Chellappa in [Che97]. He suggested that this would be a new "computing paradigm where the boundaries of computing will be determined by economic rationale rather than technical limits". More recently, Armbrust et al. [AFG+09] presented some key concepts of this paradigm, such as the illusion of infinite computing resources available on demand and the ability to pay for the use of computing resources on a short-term basis as needed. This allows companies to save costs by having a limited set of resources that can be increased according to their needs. In the same line, Jha et al. [JMF09] established how Clouds can be viewed as a logical continuation from Grids

by providing a higher-level of abstraction. In other words, Cloud Computing is a multi-purpose paradigm that enables efficient management of data centres, timesharing, and virtualisation of resources with a special emphasis on business models. Although rather interesting work has been developed in this field, its strong entrepreneurial and centralized style impedes its applicability in the construction of cheap collaborating systems, like the ones in which we are interested in.

- **Internet computing:** Internet Computing is the basis of all large scale distributed paradigms; it has grown very fast into a vast flourishing field with enormous impact on today's information societies. Internet technologies and applications are evolving and growing every day. Despite there being no closed taxonomy, most authors include Web computing inside this group, to service complex web applications with a lot of functionalities, the Internet of Things, referring to uniquely identifiable objects (things) and their virtual representations in an Internet-like structure, and Volunteer computing, maybe the most popular form of Internet computing. The Volunteer computing paradigm is based on the idea that most privately owned computers are idle most of the time, and could be used during their idle time to solve scientific or engineering problems that require large amounts of computer power. The basic model is that the volunteers download software that will do the scientific calculations from the web. This software typically works as a screensaver program, and every so often the program will ask the application to upload results and download more data to be processed. There are many ongoing volunteer computing projects nowadays, but by far the most popular are SETI@home [Pau02] and BOINC [And04].

    1. **SETI@home.** SETI@home is a volunteer computing project that analyses the data from the Arecibo radio telescope in Puerto Rico for signs of extraterrestrial intelligence. SETI@home has been downloaded to more than 5 million PCs around the world, and has already used the equivalent of more than 1 million years of PC processing power.

2. **BOINC.** In 2003, the team behind SETI@home launched BOINC (Berkeley Open Infrastructure for Network Computing), a general-purpose open-source platform that allows scientists to adapt their applications to volunteer computing. BOINC is a program that makes it possible to subscribe users to one or more projects. This platform allows several different applications to be run, allocating different percentages of the users' idle computer time to each one. There are many applications running on BOINC, including ClimatePrediction.net, studying climate change, and Predictor@home, investigating protein-related diseases.

- **P2P computing:** Although P2P computing can be found classified inside the Internet computing category in the literature, many authors define that kind of computing as a different group. This may be because of its increasing popularity and the fact that many researchers have focused strongly on that kind of distributed system in recent years. P2P Computing originated as a new paradigm after the traditional client-server computing. In its beginnings, P2P systems became very popular for file sharing among Internet between users. Systems like Napster, Gnutella, FreeNet, BitTorrent and many others with the same purpose entered strongly into our lives. In fact, it can be said that the P2P structure was popularized by those file-sharing systems, inspiring new structures and philosophies in many areas of human interaction. Peer-to-peer networking is not restricted to technology, but also covers social processes with a peer-to-peer dynamic. In such a context, social peer-to-peer processes are currently emerging throughout society. Unlike centralized or hierarchical models of Grid systems, in P2P computing, the users have equivalent responsibilities and capabilities, so they can be both servers and clients at the same time. Resource sharing and cooperation between users is a key point in this kind of networks. Peers make a portion of their computational resources directly available to other network participants, without the need for central coordination by servers or stable hosts. In this kind of distributed paradigm, it may be possible to construct cheap distributed systems on Internet for parallel/distributed processing with

management and energy consumption at almost nil cost. We focus our efforts mainly on these.

The aim of this thesis is to present innovative research results, methods and development techniques from both theoretical and practical perspectives related to P2P Computing. This project seeks original contributions in the field of incentive mechanisms on the sharing of resources and the scheduling of distributed tasks across Internet.

1.2 Peer-to-Peer paradigm

Conceptually, the P2P paradigm is an alternative to the client-server model, where there is typically a single or small cluster of servers[1]. Typically, there are only one or just a few servers and many clients[2] (see Figure 1.1). Client-Server systems represent single-unit solutions, including single- and multi-processor machines, as well as high-end machines, such as supercomputers and mainframes. On the other hand, a Peer-to-Peer model represents the execution of entities with the roles of clients and servers. Any entity in a system can play both roles. Similarly, an entity can be a server for one kind of request and client for others.

In its purest form, the P2P model lacks server nodes since all participants are treated equally in the system. Despite this, peers can perform different roles depending on the system needs. P2P models enable peers to share their resources (information, processing, presence, etc.) with at most a limited interaction with a centralized server. The peers need to deal with various kinds of network devices (wireless, unreliable modem links, etc.), support possibly independent naming, and be able to share the role of the server. There are many reasons that explain the increasing popularity of those systems. Some of the most important advantages are explained below [MKL+03].

[1] A server is formally defined as an entity that serves requests from other entities, but it does not initiate requests.

[2] A client is defined as an entity (node, program, module, etc.) that initiates requests but it is unable to serve requests.

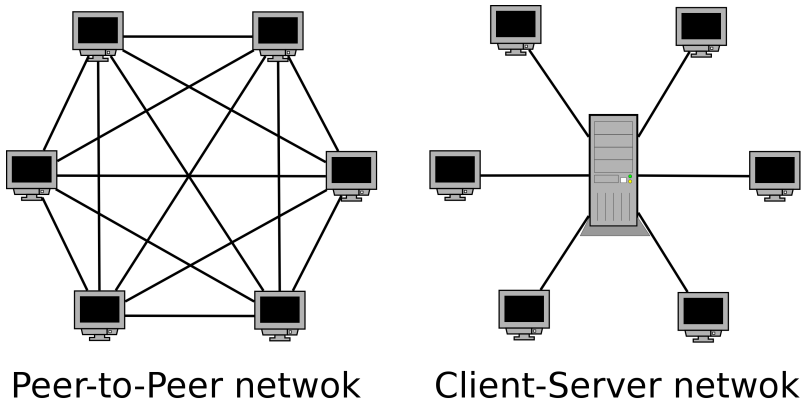


Figure 1.1: Peer-to-Peer versus Client-Server network.

- **Cost/work sharing/reduction.** When the number of clients in a Client-Server network becomes too large, the server has to deal with a huge workload, leading to bottlenecks. On the contrary, a P2P architecture solves this problem by spreading the load over all the peers in a balanced way when possible.

- **Resource aggregation and interoperability.** Interoperability in P2P networks is defined as the ability of the users to work together (interoperate), while the concept of resources aggregation refers to the capacity of such systems to collect a huge number of resources. Each node in the P2P system collaborates with its own resources, such as computing power or storage space. Applications that benefit from huge amounts of these resources, such as compute-intensive simulations or distributed file systems, naturally lean towards a P2P structure to aggregate computational resources to solve larger problems.

- **Scalability/reliability.** Scalability is defined as the system's capacity to handle growth on demand (load, users, requests, etc.), and reliability is defined as the ability of a system to handle node failures or data losses. Thanks to the lack of a strong central authority for autonomous peers, another important goal of the P2P networks is to improve system scalability and reliability.

- **Autonomy.** In many cases, the users of a distributed system are unwill-

ing to rely on any centralized service provider. Instead, they prefer that all data and work to be located and performed locally. Nodes are more autonomous in making decisions about collaboration with their resources in other works or jobs.

However, in order to take advantage of such characteristics, it is necessary to deal with several features that are specific to P2P systems. These features represent the main obstacles to the design of P2P architectures. Once solved, they provide great performance advantages to P2P systems that hardly can be achieved by other architectures using such few resources. The main challenging features of P2P systems are:

- **Heterogeneity.** Peers' resources and capabilities can vary sharply between peers. In this case, the challenge for the design of P2P systems is to provide such resources on a bigger scale, taking advantage of any peer's resources independently of its particular characteristics.

- **Distributed Management.** P2P avoids centralized resources and management, because these can limit the scalability. Also, distribution of management responsibilities among all the peers increases fault-tolerance.

- **Dynamism.** P2P systems assume that the computing environment is highly dynamic, that is, resources, such as peer nodes, will enter and leave the system continuously. This phenomenon is also known as **churn**. When an application is intended to support a highly dynamic environment, the P2P approach is a natural fit. Dealing with the dynamic behavior of P2P components provides the system with built-in fault-tolerance and an excellent adaptability.

Besides these features, which are implicit from any P2P network, there are some other important aspects/problems that are the targets of any P2P network designer.

- **Robustness.** In general, this concept refers to the ability of a computer system to cope with errors during execution or the ability of an algorithm

to continue to operate despite abnormalities in input, calculations, etc. A robust system should not break down easily when affected by a single failure and it should either recovers quickly from or holds up well under exceptional circumstances. Taking this definition into account, a P2P system will be robust if it is capable of dealing with any problem that may arise during the system lifetime.

- **Security.** In computer science, security usually consists of the provisions and policies adopted to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority. In a centralized system, security policy can be dictated from a single location, and can be enforced with firewalls, monitoring and intervention. On the other hand, centralization offers a single point of failure. Both direct malicious attacks and lax or negligent administration at the heart of a centralized network can undermine the security of an entire system. Otherwise, in a decentralized P2P system, bad behavior has a locality impediment. Malicious attacks need to occur at or near every part of the system it wishes to affect. However, P2P systems lack the tools available to a centralized administrator, so it can be much more difficult to implement security protection on a deployed P2P system.

- **Fault-tolerance.** Fault-tolerance is the property that enables a system to continue operating properly with an overall or partial system failure. In a P2P system, this aspect becomes crucial since the system itself has to deal with the unpredictable behavior of millions of users, a fact that can easily compromise the good performance of the overall system.

- **Availability.** In computer systems and networking, availability is a general term that is used to describe the amount of time over a one-year period that the system resources are available in the wake of component failures in the system. Availability of a system over its life-cycle is typ-

ically measured as a factor of its reliability. As reliability increases, so does availability. However, no system can guarantee 100% reliability; and as such, no system can assure 100% availability. In a P2P system, the most widely formula to increase the availability of the shared resources is based on replication techniques.

- **Cooperation.** In general, this term is defined as the process by which the components of a system work together to achieve the global objectives. This aspect becomes crucial in P2P networks. With no cooperation among peers, the network will reach a point where it will be difficult to obtain resources. In this kind of network, users tend to exploit the maximum resources they are able to obtain, offering minimum resources in response. For this reason, efficient mechanisms and techniques oriented towards increasing cooperation between peers must be developed.

- **Peer reputation/trust.** The reputation of the users in a P2P network is an aspect that has to be taken into consideration each time a peer has to perform an action that involves the participation of other peers. Formally, the reputation can be defined as the aggregation of all the recommendations from the third-party agents about the quality and the previous behavior of a user. Moreover, the trust concept does not differ greatly from the reputation term and both of them are considered important issues in P2P systems. Transactions in this kind of networks can cross domains and organizations, and not all domains can be trusted to the same level. A flexible and general-purpose trust management system can maintain current and consistent reputation information for the different entities in a distributed system.

- **Quality of Service (QoS).** Quality of service is the ability to provide different priorities to different applications, users, or data flows, or to guarantee a certain level of performance or response time. In P2P systems, QoS guarantees are especially important because the network capacity tends to be insufficient to service all the requests at one time.

- **Load balancing.** In P2P systems, not all the content is always available

from anywhere on the network. In such cases, the system may degenerate to a client-server network with all network traffic and computing requirements being directed to a small number of hosts. For this reason, a good algorithm for balancing the load in the system can greatly increase the overall performance.

- **Free-riding problem.** Free-riding is considered one of the most serious problems encountered in P2P systems. The presence of free-riders has a significant impact on the overall system performance. A free-rider is a peer that consumes resources from the network with no compensation/contribution in return. This definition includes those peers that only want to take advantage of the system but without sharing anything and those that simply share less than they should. Free-riding was recognized as a major problem in the very early days of peer-to-peer networks. Studies of real networks indicate that most users tend to behave selfishly and do not share their resources. For example, according to [AH00] "almost 70% of Gnutella users share no files, and nearly 50% of all responses are returned by the top 1% of sharing hosts". Taking this into account, one can easily understand how important the incentive mechanism that fight for minimizing the effect of this kind of selfish users is.

1.2.1 Degree of Decentralization

Although it would seem contradictory, not all peer-to-peer networks are completely decentralized.

1.2.1.1 Centralized peer-to-peer architectures

Like the popular Napster [nap06], these networks contain a central server that executes vital functions for the system. This central server stores an overview of the available nodes and resources in the network. In this way, the central server makes it possible for peers or nodes to rapidly find, locate and share resources with other peers. Despite this good feature, the whole system stops functioning if the central servers cannot be reached for whatever reason. Figure 1.2 shows an example of a centralized P2P architecture. In this example, Peer

A requests *"ccc.mp3"* from the Server, which is responsible for locating it in the system and ensuring that the file will be served.

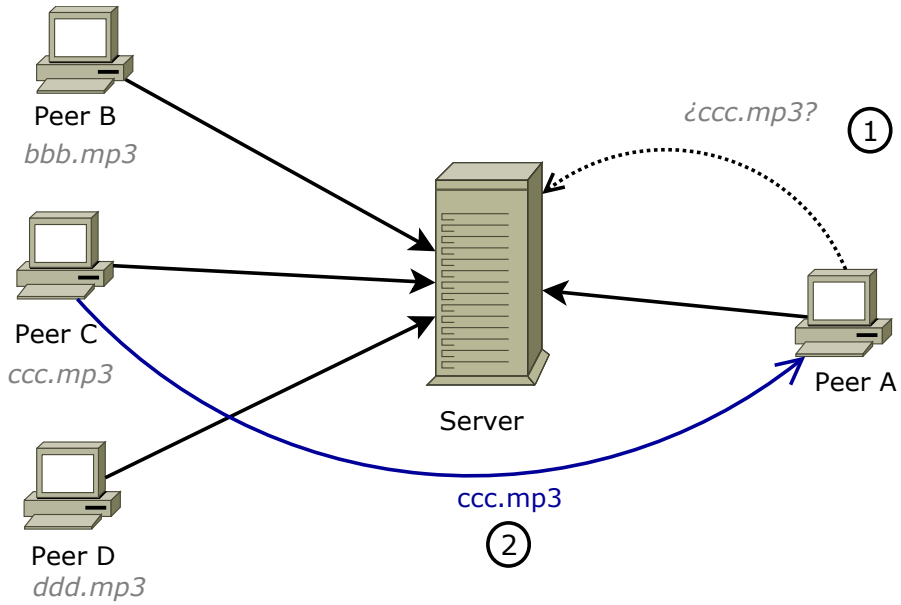


Figure 1.2: Example of a Centralized Peer-to-Peer network.

1.2.1.2 Pure decentralized architectures

The nodes perform functions without the intervention of centralized components. These types of architectures have theoretically unbounded scalability and a high level of fault tolerance. In addition, these systems are autonomous and self-organizing in a sense that the peers are responsible for the functioning and viability of the network. In practice, a great deal of these systems have limited scalability because self-organization implies a lot of traffic to keep the network running. Examples of this kind of network are Gnutella [KC04], Freenet [CSWH01] and Ares Galaxy [are]. Figure 1.3 shows an example of a pure decentralized P2P architecture. In this example, Peer A asks for *"ccc.mp3"* to its neighbours, which will propagate the request in the system until the file is located and directly served to Peer A.

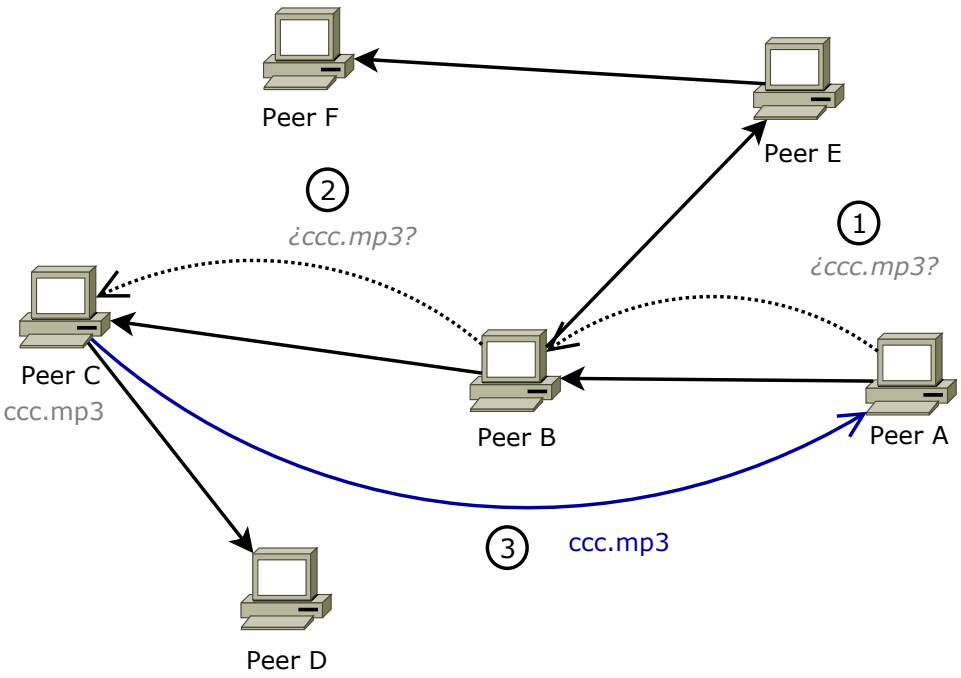


Figure 1.3: Example of a Pure Peer-to-Peer network.

1.2.1.3 Hybrid systems

These architectures are often hierarchical networks that adopt elements of both centralized and pure decentralized architectures, which combine the advantages of both. In hybrid peer-to-peer systems, some peers have more capacities than others. These nodes, that perform more functions in the network, are known as super-peers and are selected taking into account their computational capacity and bandwidth. Managing traffic requests inside the network is one of the main objectives this kind of peer. This kind of network has the advantages of the centralized P2P architectures but solves the main drawbacks of the latter, like the reduced scalability and low fault tolerance. In other words, the hybrid systems take the main advantages of both architectures explained before, i.e., the centralized P2P networks and the pure decentralized ones. To achieve that, when peers connect to the system, they have to inform the super-peer about the exact type and amount of resources they are going to share. This way, each request in the system can be delegated to the appropriate super-peer in order to optimize process of searching for the resources. Because of its great advantages,

many popular networks that use this kind of structure can be found. Some examples are eDonkey2000 [edo], BitTorrent [bit], Kazaa [ZF] and Gnutella2 [gnu]. Figure 1.4 shows an example of a hybrid P2P architecture. In this example, Peer A requests *"ccc.mp3"* from its super-peer, which will delegate the request to the super-peer controlling the owner of this file. Finally, the request is directly served by Peer C.

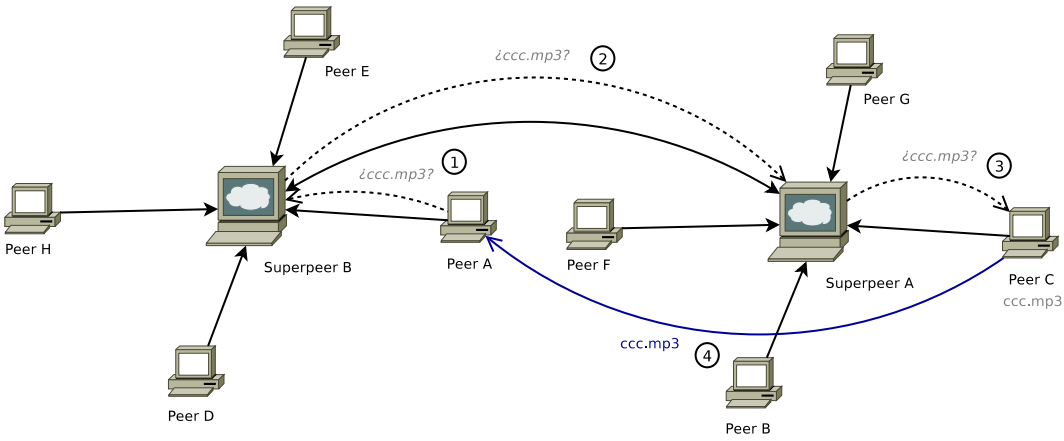


Figure 1.4: Example of a Hybrid Peer-to-Peer network.

1.2.2 Overlay architecture

Depending on how the nodes are organized in the network, it is possible to speak of two different kinds of networks, unstructured and structured. Next, both concepts are explained in detail and two popular networks are presented as an example of each type.

1.2.2.1 Unstructured

A system is unstructured (see Figure 1.5) when nodes and data are positioned without certain rules and in an ad hoc manner in the network. The location of data is not connected to the topology of the network which results in cumbersome and rather inefficient search methodologies - such as the "query flooding model" of Gnutella [KC04] - that hamper scalability. An advantage is that these systems – e.g. Napster, Gnutella, KaZaA [nap06, KC04, ZF] –

mostly support keyword-based searches. Freenet ([CSWH01]) is often called a "loosely structured" network because it is not rigidly structured in that the location of the data is not totally specified.

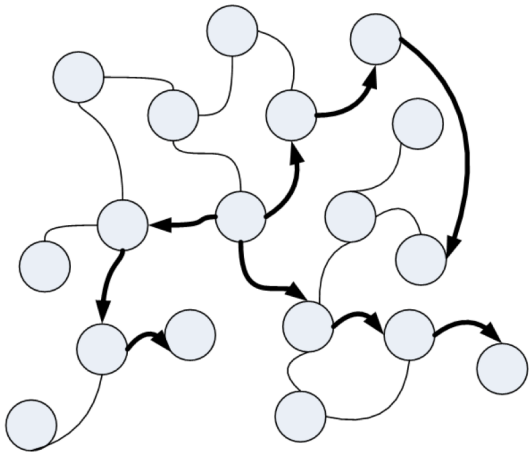


Figure 1.5: Example of an unstructured Peer-to-Peer network.

1.2.2.2 Structured

In this type of network (see Figure 1.6), nodes and data are placed in a structured way in the network so as to be able to locate data efficiently, which

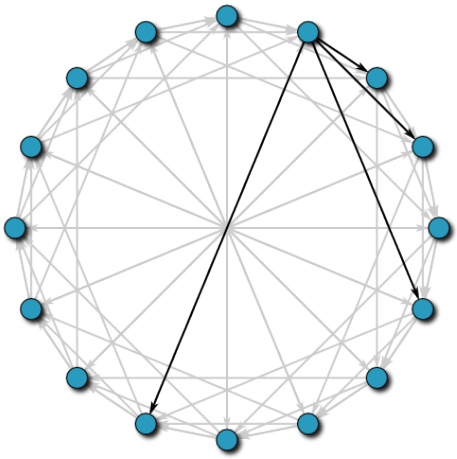


Figure 1.6: Example of a structured Peer-to-Peer network.

increases the possible scalability. The nodes, data or other resources are connected to specific locations. Distributed routing tables make it possible to acquire search results efficiently, i.e. in a smaller number of hops. Structured systems are, in comparison with unstructured systems, more scalable, reliable and fault tolerant. A shortcoming is that these systems laboriously handle the transient connectivity of nodes whereby the system needs to reconfigure the structure constantly. Examples of structured systems are Chord, CAN, and Tapestry [SMK+01, RFH+01, ZKJ01].

1.2.3 Network purpose

Nowadays, there are numerous applications of peer-to-peer networks. The most important and popular ones are briefly explained and classified into three different fields: communications, file-sharing and computing.

1.2.3.1 Communications

This category consists of applications designed to connect people around the world. We are talking about phone calls, chats, instant messaging, videoconferences, e-mail, etc. These applications can be divided into two main groups:

- **Instant Messaging (IM).** These applications allow the users to exchange text (and even some kinds of files) in real time. With the continuous growth and propagation of Internet, IM applications become hugely popular and millions of users were attracted to their networks. That fact, and the well-known technical limitations of the traditional client-server IM applications, motivated the appearance of new IM applications implemented over P2P networks. Using this topology, the system's scalability can be dramatically increased because each participating peer is responsible for a subset of the system's load. Additionally, the overall reliability and security of the IM system is not dependent on any particular peer. The effect of a peer's failure or compromise is localized in such a peer. Thus, P2P IM solves the main issues associated with traditional Client-Server IM systems.

- **IP telephony.** IP telephony, also called VoIP, is used to conduct voice conversations across Internet Protocol (IP) based networks. This technology provides important advantages in the fields of both individuals and companies. For the former, it can be said that an IP telephone service is often cheaper than traditional public telephone network service and can remove geographic restrictions to telephone numbers. For the second group, IP telephony separates the voice and data pipelines of the company's networks, channelling both types of traffic through the IP network while giving the telephone user a range of advanced capabilities. Moreover, besides voice, VoIP software applications also include other services such as conferencing or virtual FXOs[3]. As an example, Skype [sky] is one of the most popular P2P applications that allows users to make voice and video calls and chats over the Internet.

1.2.3.2 File-sharing

File-sharing is certainly the most popular use of this kind of network. Whenever someone talks about P2P, the first thing that comes to people's mind is a long list of software (kazza [ZF], eMule [emu], bittorrent [bit], etc.) for sharing files, especially films and music. Formally, file sharing can be defined as the practice of distributing or providing access to digitally stored information, such as computer programs, multimedia (audio, images, and video), documents, or electronic books. Users can use software that connects to a peer-to-peer network to search for shared files on the computers of other users connected to the same network. Files of interest can then be downloaded directly from other users on the network. Typically, large files are broken down into smaller chunks, which may be obtained from multiple peers and then reassembled by the downloader. This is done while the peer is simultaneously uploading the chunks it already has to other peers.

Although this use of the P2P networks is the most popular and by far the most commonly used, those systems have to deal with the important and

[3]Foreign eXchange Office, or FXO, designates a telephone signaling interface that receives POTS, or "plain old telephone service". Analog telephone handsets, fax machines and (analogue) modems are examples of FXO devices.

controversial problem of sharing files with *copyright*. Moreover, since these networks usually have millions of users and everyone can share whatever they want, this problem is difficult to solve.

1.2.3.3 Computing

Nowadays, there is a clear trend in which personal computers improves much faster than the requirements of the common users. That improvement includes CPU, Memory and storage capacity, and goes in parallel with a continuous drop in price of that hardware. Moreover, the steady development of Internet, especially in terms of bandwidth and price reduction, makes it very easy to provide access for a large part of the world population. These two facts have motivated the idea of harnessing the millions and millions of small amounts of idle computational resources that, grouped together into a virtual network, could overshadow some of the most powerful supercomputers in the world. Furthermore, if we take into account that the cost of maintaining all of these resources is shared between the millions of users forming the network, P2P computing becomes a real alternative to the expensive clusters and supercomputers.

However, it must be said that not everything in this kind of distributed computing is positive. There are some problems and issues that still need to be solved and improved. For example, the quality of service in those system can hardly be guaranteed since the computational resources are not stable in the system and it is very difficult to predict how many there will be in the near future. For this reason, not all the parallel applications fit properly into this kind of networks. That does not mean that not all applications can be executed on P2P computing system, but there are a group of them whose natural features are less appropriate for such kind of system. To round up, the optimal applications to run on a P2P computing system are those in which the computational requirements are so huge that it will be tremendously slow and expensive to run on a supercomputing. In fact, since several years ago, there are some applications with these characteristics running on this kind of system. For example, SETI@home [Pau02] is a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence

(SETI). Everyone can participate by running a free program that downloads and analyzes radio telescope data. Another famous example is distributed.net [dis], which takes advantage of idle computational resources from millions of desktop machines to decipher encryption codes.

1.2.4 Summary

Among all of these kinds of P2P networks, obviously the computing oriented ones are the focus of interest of this thesis. Regarding the degrees of structure and decentralization, we deal with both structured and unstructured platforms, and we apply our models to hybrid topologies. Specifically, we focus on those networks composed of a variable number of interconnected super-peers that act as the front-end of the forming nodes. At the same time, those nodes can be clusters, local-area networks or simply isolated workstations connected to Internet. This choice guarantees the interoperability between different groups of distributed environments on Internet. Scalability is another important feature that this choice provides because the whole system load is split and managed by the super-peer, so we avoid collapsing the network with too much traffic. Moreover, the super-peers will have the most important responsibilities such as starting, controlling and managing the system.

1.3 Peer-to-Peer Computing

P2P computing systems harness CPU idle cycles from thousands of computers connected through Internet and aggregate their computational resources to execute huge and massively-parallel distributed applications that cannot be executed by standard supercomputing centers due to their costs and long execution times. P2P computing can potentially provide access to a huge volume of cheap computer resources, but still has to deal with important challenges related to peer cooperation and trust, efficient use of resources, security, fault-tolerance and availability to widen its deployment.

In the literature, many formal definitions of the P2P computing concept can be found, so we cannot hope for a single well-fitting one. However, two of

the most complete and academically accepted are introduced below:

"Peer-to-peer computing has been envisaged to solve computing scenarios which require spatial distribution of computation, spatial distribution of content, real-time collaboration, ad-hoc networking[4]*, scalability or fault-tolerance at reduced costs. P2P systems provide higher storage and access capacity via distribution of resources across peers, improved reliability due to the availability of multiple peer machines and distributed security achieved by distributing partial secrets across peers. Unlike the client-server computing paradigm, where all the computation cycles and data are to be had from a single source, in P2P the participating peers contribute with CPU cycles and storage space."*

"A P2P computing network is characterized by direct connections using virtual namespaces, it describes a set of computing nodes that treat each other as equals (peers) and supply processing power, content or applications to other nodes in a distributed manner, with no presumptions about a hierarchy of control."

P2P computing architectures allow for decentralized application design, moving from centralized server models to a distributed model where each peer, independently of software and hardware platforms, can benefit and profit from being connected to millions of other peers. In such architectures, clients and servers have a lateral relationship rather than the traditional vertical relationship, giving the whole peer group tremendous processing power and storage space.

It is worth mentioning that P2P computing is still evolving and much needs to be done to overcome complex issues. Some of these issues are security, network bandwidth, fault tolerance, node searching algorithms, p2p application design and development, network architecture designing and so on. Among these, in this thesis we deal with the development of incentive policies and scheduling mechanisms and algorithms.

[4]Ad-hoc networking is defined as a system that enables communication to take place without any preexisting infrastructure in place, except for the communicating computers. These computers form an ad-hoc network. This network and associated computers take care of communication, naming and security. P2P systems can be used on top of an ad-hoc communication infrastructure.

1.3.1 Incentive Mechanisms

One of the most critical aspects in the design of P2P systems is the development of incentive techniques to enforce cooperation and resource sharing among participants. These systems enable massive resource and information pooling at low cost per participant and at scales that are difficult to achieve with traditional client-server systems, while local autonomy and network effects provide resilience to failures and attacks. However, one distinguishing characteristic of P2P systems compared with more traditional client-server designs is the widespread cooperation between participants by sharing resources and information. As all the benefits of these systems are deeply rooted in cooperation, they are inherently vulnerable to large-scale non-cooperative behavior. It is therefore necessary for the system to be designed so that participants generally cooperate. The mechanisms that are embedded in the system for this purpose are called incentive mechanisms.

Cooperation is highly appreciated because P2P network users tend to exploit the greatest amount of resources they can obtain, offering minimum resources in return. This behavior undermines the goal of P2P of spreading resources and imposes the concept of **free-riding**. Free-riding leads to the need for social and economic mechanisms to balance resource usage and efforts in P2P systems. Experience with peer-to-peer systems shows that in the absence of incentives for donation, a large proportion of the peers only consume the resources of the system. Free-riding leads to the need for social and economic mechanisms to balance resource usage and efforts in P2P systems.

Incentive aspects in P2P networks have been in the enhancement and development loop since the emergence of this class of systems. Theoretically, and to fully utilize the resources of a P2P network, the system architect must have a precise understanding of the payoff of each individual user joining the network in order to construct an appropriate incentive mechanism [ACM04]. However in practice, the architect is unable to have a clear vision of every user's requirements. Even if this were possible, it is difficult to implement the incentives that would actually work. The difference between what researchers design, and how users actually react to the proposed incentives depends on many factors.

- First, users in a P2P network perceive the proposed incentives in different ways depending on their own utility functions[5].
- In second place, most of the proposed incentive mechanisms do not adapt to changes in user behavior.
- Thirdly, there is no extensive and reliable measurement study that would help P2P network designers evaluate the real performance of their incentive mechanisms with the expected performance. Most of the measurement studies completed on existing P2P networks demonstrate the success or the failure of such systems with no analysis or explanation on why these systems succeeded or failed.

When researchers designate a P2P system as successful, they usually relate the success to the system's public acceptance and approval without providing detailed evidence supporting the evaluation. That is, there is no proof that these systems are successful because they have strong incentive mechanisms, robust against security attacks, or some other explanations. Moreover, some of these systems have not been thoroughly tested against security attacks for different reasons.

- First, there is only a vague understanding of what strategies the attackers may possibly follow.
- Second, users might not attempt to subvert the incentive mechanism because they already obtain a satisfactory level of performance even though the incentive mechanism of the P2P system is vulnerable to attack.

In general, a failure of a P2P system is related to the lack of incentive acceptance among users, who disregard the system if they sense that it does not provide them with satisfactory payoffs. With no cooperation among peers, the network will reach a point where it will be difficult to obtain resources. Incentive policies try to persuade peers to share their resources. What makes the incentive problem particularly difficult is the unique set of challenges that P2P systems pose [ZH08]:

[5]The utility function expresses utility as a function of consumption as opposed to the provided shared (files, CPU, Memory, etc.).

- **Large populations.** P2P computing networks are designed to allocate millions and millions of users. Obviously, managing that number of peers is far from being a trivial matter.

- **High turnover.** Managing so many users at the same time implies that the network has to deal with many unpredictable connections and disconnections.

- **Asymmetry of interest.** Not all the users in the system have the same needs or objectives. Figure 1.7 shows an example of asymmetry of interest, where the peer A wants service from B, B wants service from C and C wants service from A.

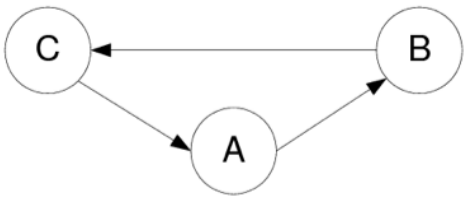


Figure 1.7: An example of asymmetry of interest.

- **Cheating/colluders and malicious.** Peers can be selfish colluders or cheaters, when they try to fool the system to gain unfair advantages over other peers, or malicious, when their purpose is to subvert the system or disturb other users.

- **Zero-cost identities.** Creating an account in this kind of network usually have no cost for the users and most of them switch identities in order, for example, to clean some previous bad behavior.

To deal with these challenges, a typical incentive system relies on a control mechanism that analyses the trustworthiness of the participating peers and combats selfish, dishonest and malicious peer behavior. In the literature, one can found different kinds of incentive mechanism. Overall they can be classified into two categories: Trust-based incentive mechanisms and Trade-based incentive mechanisms [OKRK03]:

1. **Trust-based incentive mechanisms.** Those mechanisms are proposed to establish trust among peers in P2P systems [HKC09, RY07]. In a trust-based system, each entity's behavior is used to predict how it will behave in the future. Trust is a straightforward incentive for cooperation, in which the agent simply executes the requested action if it trusts the principal entity. The agent may either share the same goals with the principal, or it may believe in increasing the cooperativeness of other entities. In the trust-inference mechanisms, the information about users' behavior can be propagated throughout the system [LS08, ZHC08]. At the same time, two categories of trust-based incentive mechanisms can be identified:

   - **The collective pattern.** A collective is a set of entities sharing the same goals with mutual trust. The agent in a collective does not need any remuneration, as its incentive for collaboration stems from being a member of the group. A peer is therefore allowed to fail to cooperate with the rest of the network for some reason. Agents however have to ensure that the principal entities are members of the same collectives. The mutual trust requirement makes collectives inherently not well-scalable. A real-life example of a collective may be a family, or a small group of families. Trust between the members of the family is unconditional, while individuals who may be unable to offer goods or services (e.g. due to illness) are still able to obtain goods or services from the other members of the group without facing any sort of exclusion or punishment.
   - **The community pattern (Reputation-based).** A community is a group of peers whose cooperation is based on the reputation earned by acting as agents. The principal entity remunerates the agent by increasing its local reputation according to the estimated value of the services offered or actions taken. As the reputation of an agent is mainly remembered by the principal entities it cooperated with, communities with stable or localized transaction patterns are expected to function better [OKRK03]. Various reputation-

management schemes whose goal is to disseminate local reputations throughout the community are being designed and used in P2P networks. As a result, entities will have access to the global reputation of other entities, which can be used to further reduce the degree of selfish or lavish behavior in the network. The dissemination of local reputation, however, requires careful design in order to avoid malicious behavior. The reputation-based community pattern is one of the predominant ones in P2P and ad-hoc networks. A real-life example of a community may be a group of people (e.g. in villages) who interchange goods as gifts.

2. **Trade-based incentive mechanisms.** In trade-based incentive patterns, agents are explicitly remunerated by the manager. The remuneration is in turn an action carried out by the manager on behalf of the agent. It can be either immediate after the agent's action, or deferred (like a promise).

   - **Immediate remuneration patterns.** Immediate remuneration is usually implemented in terms of "Barter Trade" as a direct exchange of actions. A prompt action is carried out by the manager entity as a return. The two actions are therefore coupled, and trust is not really required (the two entities may even remain anonymous). This approach scales better than other schemes where the two actions are temporarily uncoupled. The roles of the two participating entities are inherently symmetric, as the one entity will only act as an agent if it is interested in an action carried out by the other entity. In practice, "reference actions" that are required by all entities are very often exchanged in order for a specific action to be executed in return. Typical examples of such reference actions in P2P networks are packet forwarding. Real-life examples of immediate remuneration mechanisms may be the use of reference goods (such as salt, flour etc) in older trading communities, or the exchange of goods for other goods.
   - **Deferred remuneration patterns.** In general (and in real-life

as well) it is infeasible to couple two cooperation actions, as the agent entity may often not require an immediate action in return. Deferred or bond-based incentive patterns use the notion of a "bond" or "promise" by the principal entity to the agent that will carry out an action in return in the future. Those mechanisms are also known as credit-based systems and, together with the reputation-based ones, are the most commonly used in the P2P networks.

Tacking into account this taxonomy, the incentive mechanism presented in this thesis can be catalogued into the "Deferred remuneration patterns" group. Specifically, we propose a credit-based mechanism where the users are rewarded with credits when sharing their resources with others. For this reason, the remuneration is deferred in time since the users do not obtain directly any action in return, but only credits. We have chosen this system because is as scalable as the reputation ones but it is easier to implement policies to avoid malicious behavior.

1.3.2 Scheduling Policies

Scheduling policies (disciplines) are implicitly used everywhere where a resource needs to be allocated. Whenever a number of people want a service at the same time, a scheduling policy determines when each person receives the service. This happens almost everywhere we venture in our daily lives. From restaurants and supermarkets, to banks and amusement parks, we queue for service in a variety of ways. In many convenience stores there is a single cash register where people queue to be served in a First-Come-First-Served (FCFS) order. In large supermarkets, there are many checkouts, and some are dedicated exclusively to serving customers with a limited number of items. On the other hand, in restaurants everyone receives a little bit of service all the time, which can be approximately modeled as Processor-Sharing (PS). Beyond these everyday examples, we also see a variety of policies used in modern computer systems. Applications such as web servers, routers, operating systems, supercomputers, databases, and disks are all under a constant barrage of requests and need to determine how best to allocate resources to users.

In P2P computing, tasks are assigned to heterogeneous computing resources. In the simplest case, all tasks are independent from each other and can be scheduled to execute in any order. This kind of parallel application is referred to in the literature as bag-of-tasks (BoT) applications [CPC+03]. In other cases, a work flow can specify restrictions on the order in which the tasks need to be executed. Only the BoT applications are considered in this thesis. Despite their simplicity, BoT applications are used in a variety of scenarios, including data mining, massive searches (such as key breaking), parameter sweeps, simulations, fractal calculations, computational biology, and computer imaging. Assuming applications to be BoT simplifies the requirements in several important ways. In particular, fast execution of applications can be delivered without demanding any QoS guarantees. It also makes it easier to provide a secure environment, since network access is not necessary during the execution of a foreign task. Moreover, some popular projects, such as SETI@home [Pau02] and MyGrid [CFA+04], utilize Grid and P2P computing to schedule such compute-intensive applications to the available resources.

Efficient use of computing resources to minimize the execution time requires a scheduling mechanism that can appropriately determine the assignment of tasks. However, since communication over a wide area network and scheduling for many computing resources require long processing time, centralized scheduling methods may cause load concentration and a single point of failure [BCF+08]. Solving this issue can be said to be one of the most important challenges for the research community in the P2P computing field, one special case of distributed environment.

Task scheduling in a P2P computing system is a two-step process. In the first step the peer selection decision is made and in the second, the assigned peer decides the order in which the tasks are to be executed. Since the second step is similar to scheduling in non-distributed computing, it is not of interest for this thesis. Only the first step is considered for the rest of this study. More formally, scheduling in a P2P computing system is a problem of mapping a set of tasks (t1, t2, t3, ..., t_m) to a set of peers (p1, p2, p3, ..., p_n) as is shown in Figure 1.8.

Among others, the main goals of a scheduling policy can include:

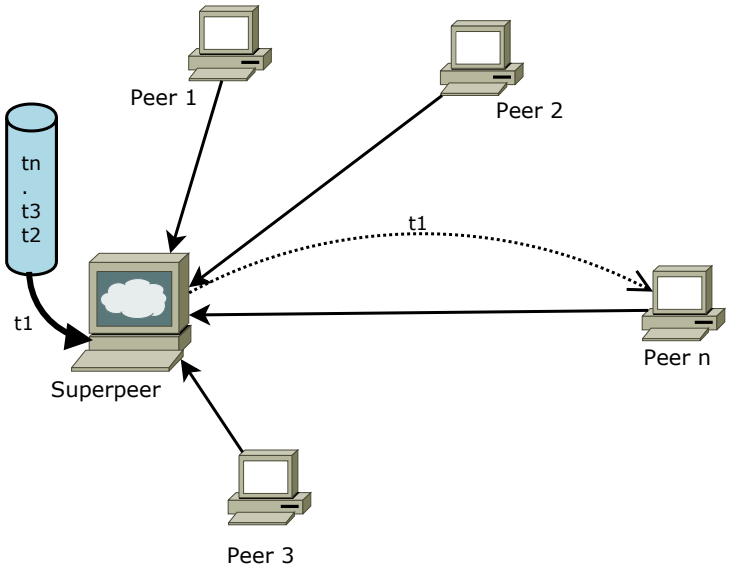


Figure 1.8: Task mapping.

- Reduced application completion times.
- Better resource utilization.
- Reduced hardware costs.

Given the characteristics of the P2P computing system, task scheduling for unreliable peers becomes a challenging issue. Available peers at scheduling time may not remain available for the length of the assigned task completion times and a task may have to be moved to other peers during execution. Without proper attention to task scheduling in the P2P computing environment, the application completion may not even be possible. Some of the characteristics of task scheduling in this kind of networks are:

- **Scheduler Organization:** there are many ways that a scheduler itself can operate in such distributed environments. Some possible options include dedicated peers devoted to scheduler execution, collections of related schedulers each with unique goals, and decentralized schedulers which may execute on every peer and communicate with others to make scheduling decisions.

- **Scheduling Policy:** depending on the information available for scheduling, policies may compute schedules at the start of the application (static) with no revisions, improve statically computed schedules with information that becomes available at run-time (hybrid) or only compute schedules at run-time (dynamic).

- **State Estimation:** the unreliable nature of peers and dynamic scheduling policies require methods for environment state estimation (run-time information) to make mapping decisions. These methods can be either predictive or non-predictive.

- **Rescheduling Approaches:** improvements to computed schedules are generally desirable due to continuous changes of peer states, and so approaches can be employed to initiate rescheduling. Some of the possible options include periodic or event-driven rescheduling.

- **Task Dependencies:** given the application's nature, tasks may depend (for data or coordination) on each other for execution. Therefore, scheduling policies need to account for these dependencies to avoid deadlocks and provide optimized schedules.

- **Scheduling Overhead:** time and memory consumptions for computing and executing schedules add to the overall overhead of application execution.

- **Fault Resilience:** unreliable peers require scheduling policies to consider peer failure while making scheduling decisions. For example, when selecting a peer for task execution, a dynamic scheduling policy needs to ensure that the target peer is still available. Furthermore, a policy may also be able to reschedule a task in case of peer failure.

- **Ease of Implementation:** this includes scheduling policies that can be easily implemented without requiring too much overhead.

- **Adaptability for Application Nature:** once a P2P computing environment is setup, it maybe required to serve as an execution platform

for more than one application. Due to the diverse type and size of applications, the ability of a scheduling policy to adapt to every application can further improve overall peer utilization.

- **Centralized/distributed:** In a centralized scheduling environment, a central machine (node) acts as a resource manager to schedule jobs to all the surrounding nodes that are part of the environment. In this scenario, jobs are first submitted to the central scheduler, which then dispatches the jobs to the appropriate nodes. Those jobs that cannot be started on a node are normally stored in a central job queue for a later start. Instead, distributed scheduling involves multiple localized schedulers, which interact with each other in order to dispatch jobs to the participating nodes. Distributed scheduling overcomes scalability problems, which are incurred in the centralized paradigm; in addition it can offer better fault tolerance and reliability. However, the lack of a global scheduler, which has all the necessary information on the available resource, usually leads to sub-optimal scheduling decisions.

Scheduling in distributed computing is a well-studied area which has resulted in a diverse set of scheduling policies for various paradigms of distributed computing, including Clusters, Grid, Cloud computing and so on. However, task scheduling for P2P computing environments is an open issue and requires more research.

1.4 Motivation

With the fast development of high-speed networks and powerful but low cost computational resources, P2P computing has emerged as an attractive distributed computing paradigm. P2P computing uses Internet to connect thousands or even millions of users into a single huge virtual computer area based on the sharing of computational resources [FI03]. By using computational resources scattered throughout Internet, P2P computing is able to deliver a computation power two orders of magnitude higher than the most powerful supercomputer [And07]. So, taking both its high performance and the sharing of

resources into account, we can assert that P2P computing represents a real and low cost supercomputing alternative for some kinds of parallel application[6].

P2P architectures take advantage of the under utilization of personal computers, integrating them into a platform based on the sharing of computational resources between geographically distributed equals. The participation of their component nodes (or peers[7]) is voluntary. Although cooperation is the key to the success of a peer-to-peer system, it is difficult to cultivate without an effective incentive mechanism. In fact, many P2P systems lack such a mechanism and consequently suffer from free riding [gnu]. Free-riders consume resources donated by other peers while not donating any of their own. Experience with peer-to-peer systems shows that in the absence of incentives to donate, a large proportion of the peers only consume the resources of the system [ABCM04]. Free-riding is a concern because it decreases the utility of the shared resources in the system, potentially to the point of system collapse.

The main aim of P2P computing networks is to aggregate the computational power of heterogeneous, geographically distributed and dynamic computational resources. These resources are usually administrated in a collaborative way by different entities and owned by various domains. Therefore, these systems have to deal with a high degree of resource volatility, which makes it difficult to exploit them efficiently. For these reasons, effective scheduling is of fundamental importance in P2P computing networks. However, due to the unique characteristics described above, scheduling in this kind of environment is particularly challenging. On one hand, some users in a typical P2P computing environment, also called peers, have computational applications, also called jobs, to execute, but they may lack computational resources for them. These peers are known as resource consumers. On the other hand, some other peers, known as resource providers, have relatively underused resources. It is highly desirable for consumers to schedule jobs across those resources, but the scheduling is significantly complicated by the distributed ownership of the

[6]Parallel applications, also called parallel jobs, are composed of a number of concurrently executing processes (called tasks), which collectively perform a certain computation. A rigid parallel job has a fixed number of tasks (referred to as the job's size) which does not change during execution [Fei97]. To execute such a parallel job, the job's tasks are mapped to a set of processors using a one-to-one mapping.

[7]A peer is defined as an entity with capabilities similar to other entities in the system.

computational resources. Consumers and providers are independent from each other, each having their own needs and objectives.

Over recent years, some researchers have been trying to solve this kind of problem by using economic methods. For example, [SPF03] and [BAGS02] started to tackle the problem by applying economic methods to Grid computing. Despite this, the related research is still in its beginnings and extensive research efforts are still required in this field. Obviously, P2P computing networks differ greatly from human society, where these economic models are often used. In fact, scheduling based on economic models for P2P computing is still a highly challenging problem.

1.5 Objectives

From the beginning of this thesis, the idea was to achieve two main objectives. The first was to design an incentive mechanism to motivate users to participate actively with the purpose of the network. The second was the design of a cooperative scheduling method for a large peer-to-peer computing system. Although these could be considered as two main research lines, both are closely interconnected to each other. This is because both mechanisms share some system information and are implemented and developed by the same agent in the network. Furthermore, decisions taken by one of them will usually have an influence over the other, and vice versa.

More specifically, this work proposes scheduling and incentive techniques to enforce cooperation and sharing among peers, but guaranteeing the efficient utilization of system resources. This distributed and cooperative scheduling mechanism also improves scalability for large-scale P2P computing.

1.5.1 Objectives of the incentive policies

A credit-based scheme to enforce collaboration in P2P computing systems based on the reinvestment of resource payments has been designed. Our approach differs from others in the sense that we distribute credits non-uniformly among peers based on their contribution to the system. The incentive mech-

anism is based on the reverse Vickrey auction strategy and it implements a non-negative credit function supported by a historic term to differentiate between newcomers and old collaborative peers. The main objectives of our incentive mechanism are the followings:

1. **Enforce collaboration.** Cooperation and collaboration are two crucial aspects in this kind of networks. Actually, the definition of P2P says that all peers in the system have to be servers and clients at the same time. If peers are not open to contribute on whatever the system needs from them, the overall network is condemned to fail. These system needs can be managing requests, organize other peers, report information, etc.

2. **Incentive resource sharing.** Another important aspect that determines the success of the system is the amount of resources shared by each user. Obviously, the more resources shared by them, the better the performance the system will have. For this reason, it is so important to encourage the users to share as many resources as possible by rewarding them for their contributions.

3. **Discourage free-riders.** As commented above, these users are a real problem for P2P computing networks, since they only want to use resources but without sharing anything. Logically, these users have a very negative effect on the system performance, so it is important to detect them as soon as possible and try to penalize them by giving them the lowest possible QoS.

4. **Avoid cheating.** Unfortunately, the free-rider behavior is not the only malicious and selfish behavior that can be found in a P2P computing network. The incentive mechanism presented in this thesis takes into account the most common kinds of cheating and tries to protect the system from them.

5. **User behavior.** Like in real live, not all the users in a P2P computing system behave the same and it is almost impossible to know how everyone will act once they have joined the network. Despite this, in the process of designing the incentive mechanism, some general behavior has to be

defined in order to predict how it will act depending on what the users do. For this reason, we defined different user behavior in this project in order to evaluate how the incentive mechanism responds to them.

6. **Increase system throughput.** In the end, all the objectives explained above are aimed at increasing the system throughput and performance given the same external conditions, i.e., number of peers, system workload, user behavior, etc. The incentive mechanism presented in this thesis has been provided with a set of features that can be set up by the system administrator in order to adjust the system to changes in external conditions.

1.5.2 Objectives of the scheduling policies

A two-level scheduling mechanism for distribute parallel applications in a peer-to-peer computing system has been designed. The goal of the two-level design is to limit the number of messages required to solve the overall scheduling. In this way, the cost, i.e. message exchanging, of mapping the tasks is significantly reduced, avoiding the saturation of the system network due to message flooding. More detailed objectives are listed below.

1. **Efficient management of the dynamic resources.** It must be said that the resources shared in this kind of network have a specially feature that makes them very different from the static resources of the popular file-sharing networks: their dynamism. When someone is sharing a file on a P2P network, this file will have always the same features from the start to the end. In contrast, when someone shares the Memory or CPU of a personal computer, the computing capacity available will vary according to the user activity. In conclusion, dynamism is an important feature of this kind of resource that has to be taken into account in the scheduling mechanism.

2. **To avoid message flooding.** In order to achieve the previous objective, it is necessary to update the information needed by the scheduling

mechanism very frequently. Taking into account that there can be millions of users on those networks, keeping all these information up to date can be a real problem and can lead the system to a saturation point. For this reason, we propose a distributed scheduling mechanism composed of two levels that avoids the message-flooding problem.

3. **Load balancing.** Balancing the load in a P2P computing system is important to avoid collapsing a few parts of it. If load-balancing policies are not properly applied, all the load in a high heterogeneous system like the P2P always tends to be assigned to the most powerful nodes. This may transform the P2P computing system into a client-server or a few sets of client-server systems, with all the problems that this entails in terms of bottlenecks and single points of failures.

4. **To avoid wasting idle resources.** Another particular feature of P2P computing networks is their inability to store the resources shared on them. As happens with other resources, like the electricity or some natural resources, if they are not used at the moment, they are wasted. For this reason, the scheduling mechanism has to be much more flexible when the system is heavily underused.

1.6 Overview

This section briefly describes the chapters in this thesis.

- **Chapter 1: Introduction.** This chapter explains what the thesis topic is about, identifying the research questions and showing how they will be answered. First, some important concepts are introduced to provide the readers with some background. Secondly, the general P2P paradigm is briefly described and some classifications of these networks are explained. Third, the P2P computing concept is introduced before dealing with the problems of scheduling and incentives. After doing so, the main objectives of this thesis are mentioned, the chapter finishing with a quick overview of the following chapters.

- **Chapter 2: Related Work.** This chapter reviews work in various areas related to the material covered in this thesis, some of which is cited in other chapters. The chapter explains the main features and functionalities of some P2P computing networks but is mainly focused on two aspects of these systems: the scheduling mechanisms and incentive policies.

- **Chapter 3: DISIM.** This chapter presents a new Distributed Incentive and Scheduling Integrated Mechanism (DISIM). This is the main chapter in the thesis since it introduces its main contributions, namely the scheduling mechanism and the policies to encourage the peers to participate actively with the system's purpose.

- **Chapter 4: Experimental Results.** This chapter deals with the analysis of the data obtained from the measurements performed through simulation tests of the scheduling mechanism and its incentive policies described in Chapter 3. Moreover, the behavior of users from a P2P computing network is modeled in order to achieve more realistic results in the experimentation.

- **Chapter 5: Conclusions and Future Work.** This chapter reviews key points. The objectives and results of the entire research thesis are summarized, special emphasis being placed on the scientific contributions that have been introduced in this research work. At the end, some prospective points for the future work of this research are provided. Furthermore, the main publications obtained during the development process of this thesis are also enumerated.

Chapter 2

Related Work

2.1 Introduction

Recently, there has been a great impetus to P2P computing. Despite this, the term P2P is not new. In one of its simplest forms, it refers to communication among peers. For example, in telephony, users talk to each other directly once the connection is established. The concept of P2P computing is also not new. Many earlier distributed systems, such as UUCP [Now78] and switched networks [Tan81], followed a similar model.

In this chapter, the main works of the scientific community related to the incentive policies and scheduling mechanisms are collected and analyzed. Section 2.2 provides an overview of the incentive techniques most commonly used in some popular computing networks. Section 2.3 does the same but for the most widely used resource management and scheduling policies.

Section 2.4 studies the most important P2P computing systems in depth with special emphasis on their incentive and scheduling mechanisms. Furthermore, this section highlights their main strengths and weaknesses and how they inspired us in the development of the mechanism proposed in this thesis.

2.2 Incentive policies

The incentive mechanisms for P2P Computing systems have grown in interest recently. These techniques encourage trustworthy behavior and maximize social welfare. However, they are vulnerable to collusion and rely heavily on the users' history. To further efficiency, researchers extended the social mechanisms into economic ones, by applying game theory and monetary-based schemes, and into trust ones, by applying reputation-based schemes.

Next, the main popular incentive policies are briefly explained and cataloged into three main groups (Figure 2.1): game theory, pricing and reputation.

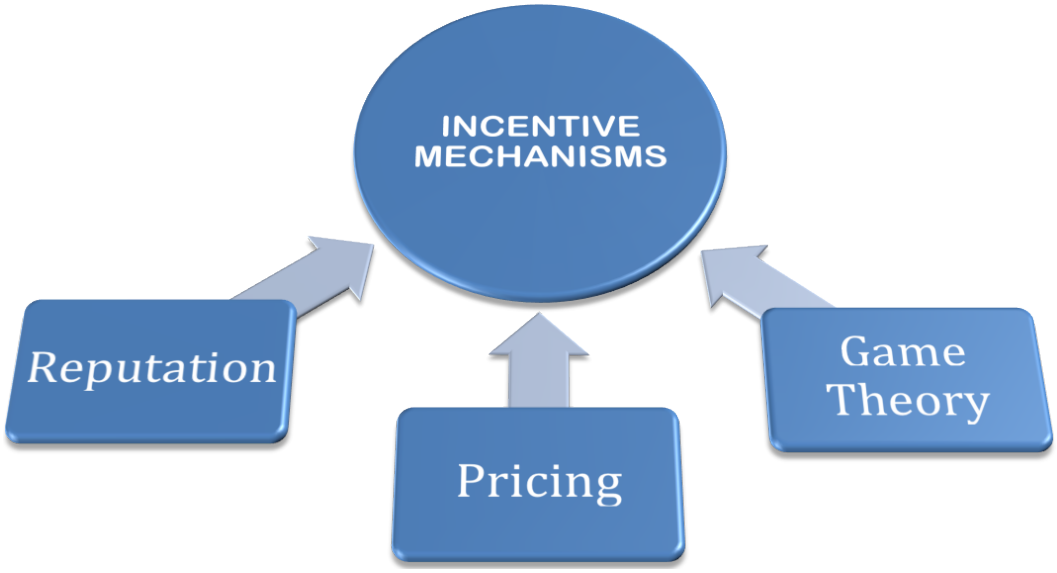


Figure 2.1: Incentive policies.

2.2.1 Game Theory

A considerable amount of research has focused on the exploitation of economic theories through a game-theoretical approach to analyze formally how selfish behavior affects the performances of P2P networks. Game theory models competitions as games and tries to work out the best strategies for the game players mathematically. The P2P systems consist of multiple nodes, each of which may carry out an infinite number of transactions. In any transaction, a node can

choose from a finite number of actions. The incentive problem is usually modeled using Game Theory to study the problems of whitewashing (nodes that change their identities to clean their history) and collusion. Most of the existing game theoretical analyses assume that all the nodes are rational, that is, they always try to maximize their utility and minimize the expense. Therefore, P2P systems can be represented as n-person repeated non-cooperative games that allow mixed strategies. According to Nash Folk Theorem [Osb04], in such strategic games, there is always at least one Mixed Strategy Nash Equilibrium (MSNE). The Nash Equilibrium is a strategy profile that consists of a strategy for every player who cannot increase its utility by choosing another strategy, given the strategies by the others. Applied to P2P systems, the equilibrium level (see Figure 2.2) is the point that optimizes the sharing capabilities and reduces free-riding behavior [Cou06, MCJI06, vdSMR08].

Extensive game theoretical analyses have been conducted to prove that incentive mechanisms can influence the nodes in the P2P systems to be more cooperative and hence to increase social welfare. However, these analytical models usually rely on some assumptions that simplify the systems, failing to estimate the real characteristics precisely. For example, although the rational nodes are mostly in the P2P systems, both the altruistic nodes and pure free

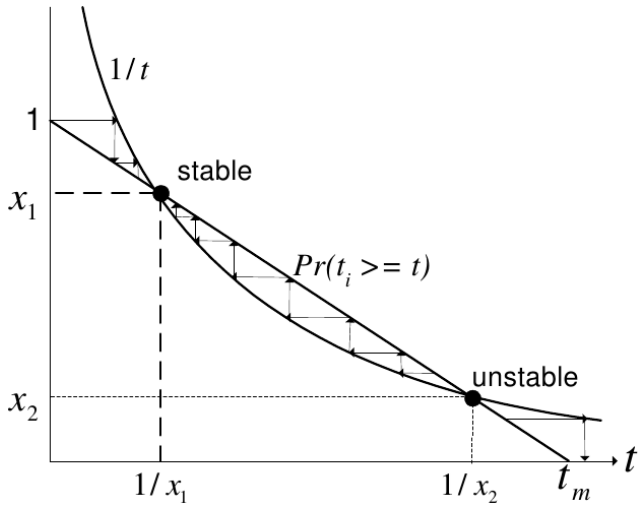


Figure 2.2: Equilibrium in game theory based incentive models.

riders can also be observed. In addition, the super-peers in the hybrid P2P systems take more responsibilities than the rest and thus their utility function is different from that of the others. The influence of such assumptions on the stability and reliability of the Nash equilibrium needs to be further evaluated. Despite this, the models can still produce useful indications of node behaviors [ZAM10].

2.2.2 Pricing

On the other hand, pricing has proved to be another effective means to incentive sharing and control computer networks. It helps in obtaining benefits for the network operators and providing incentives for users to cooperate [LLW09]. The pricing-schemes [GPM01, YGm03, pay] follow the principle that nodes pay the resource providers for the resources they consume with either real money or virtual tokens/credits. This kind of scheme can work effectively in a service-oriented environment as the service providers are rewarded with either money or virtual currency and free-riding behavior cannot obtain any profit from the nodes. Different generations can be found inside the pricing-schemes group:

- **Flat Rate** [Cou06, GPM01]: One likely payment model for peer to peer systems is some kind of flat rate membership fee per time period, for example as a way of recovering royalty costs or the overhead involved in running servers. This kind of approach is specially used in commercial applications of peer-to-peer networking but they have some important drawbacks that makes them inappropriate for the systems under study in this thesis. For example, the fact that flat fees are unrelated to agents' behavior implies that they still give rise to the problem of free-riding.

- **Token-based:** In such systems, a customer first has to check the service at the market before being allowed to buy tokens from the broker. Then, it can use the tokens to purchase the service on the market. The system then provides the customer with the service and finally redeems the tokens from the broker. The main limitation of the token-based micro-payment system is that every transaction will generate a new token (see Figure 2.3). The broker has to keep a record of all the tokens, which, in

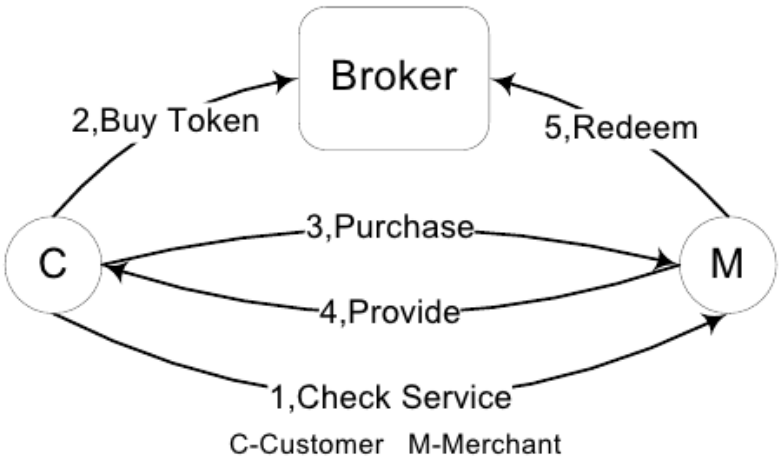


Figure 2.3: Transactions in Token-based Incentive Systems.

turn, leads to a scalability problem. PPal [YGm03] is an example of a system of this kind.

- **Account-based:** The second generation micropayment systems are account-based, in which every customer has an account with the broker and authorizes the broker to transfer money from their accounts. To purchase a service, a customer first checks the service on the market, before it informs the broker of its interests, as shown in Figure 2.4. The system then supplies the customer with the service. After checking the quality of the service, the customer confirms with the broker that he is willing to pay. The broker then takes the money from the customer's account. PayPal [pay] is probably the most popular example of such a mechanism.

A rarely used kind of incentive mechanism, but still present in the literature, is the fixed-contribution scheme. In such systems, a node has to contribute a fixed amount of resources before being allowed to participate in the network. This scheme normally requires a centralized entity to monitor the quality of contributions made by the nodes. However, it is not suitable for a majority of decentralized P2P networks. Direct Connect [dir] is a typical example of such schemes. It requires each node to contribute a minimum number of files and

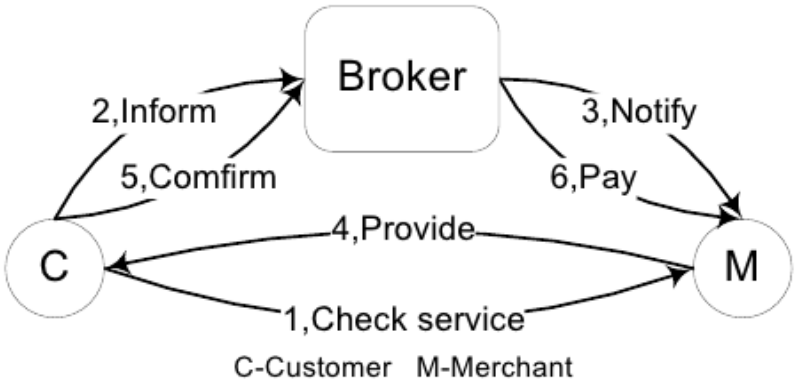


Figure 2.4: Transactions in Account-based Incentive Systems.

make a minimum upload bandwidth available.

2.2.3 Reputation

Going to the reputation-based schemes, Andrade [ABCM07] presents an incentive mechanism called the Network of Favors to assemble a large grid (OurGrid P2P System[CBA+06]), which makes it in the interest of each participating peer to contribute their spare resources. Besides, H-Trust [ZL08] is a group-trust management system inspired by the H-index aggregation technique. It offers a robust personalized reputation evaluation mechanism for both individual and group trusts with minimal communication and computation overheads for Peer-to-Peer computing systems. In a similar way, PowerTrust [RY07] is also a reputation system focused on P2P computing. In [MKP08], a distributed mechanism is presented which promotes fairness and inhibits free-riding through a shared-history reputation mechanism designed for use in dynamic Peer-to-Peer systems.

Finally, in [ZZM07], Zhang et al. define a mechanism in which each node is associated with two parameters: money and reputation. Peers exchange money for services and increase their reputations by doing so. Similarly to other schemes, they classify peers into three different types: honest, selfish, and malicious. There is a central authority that settles disputes between peers when one believes it has overpaid or not received enough service. Despite this,

this centralization scheme becomes one of the main drawbacks of the system since it limits its scalability. In another interesting work [PWL09], Ponce et al. define a credit system that is used to keep track of the interaction between peers. They introduce two new properties to consider: transferring credit and proximity. Transferring credit will promote proximity routing by allowing paths to be taken that were inaccessible before (because of debt).

2.2.4 Incentive Policies and Methods Comparison

Table 2.1 shows a comparison of the most relevant features of the main Incentive schemes (grey rows in the table). Moreover we have also analyzed the characteristics of one relevant policy for each method.

As we can see in the first row on the table 2.1, the main problem with most existing monetary/payment schemes (booth token-based or account-based) is the heavy load on the trusted, centralized broker. A broker is required to handle accounts, distribute and cash coins, provide security (such as double-spending detection), etc. Although payments need not be on-line, the broker must eventually take some action for every transaction; as a result, broker load is always $O(n)$ in the number of transactions. Brokers therefore present a scalability and performance bottleneck for any system using these payment schemes.

We analyzed three pricing schemes PPay [YGm03], SHARE [CNA+04] and TIM [ZLH07]. PPay [YGm03] is a token-based micropayment scheme that addresses the dual problem of improving performance while maintaining security, defining an internal transferable and self-managed currency called coins. SHARE [CNA+04] allocates resources using a centralized combinatorial auction. Solving the NP-complete combinatorial auction problem provides an optimally efficient allocation. TIM (Trust-Incentive service Management) [ZLH07] is a hybrid pricing-reputation scheme. It provides a price strategy based on resource supply and demand plus a weighted voting scheme to secure the P2P system.

All payment policies, as PPAY, SHARE and TIM, can be adapted independently of the structure or non-structure of the P2P system. Moreover, all

use money (virtual or real) as the main incentive to reinforce cooperation, that can be applied on either a local or global scope. To ensure the effectiveness of incentive mechanism, both PPAY and SHARE reinforce collaboration using authenticated identities that require a non-scalable and centralized certification/authentication authority. TIM avoids this requirement using a reputation mechanism to decline the joining request from malicious peers. QoS can be easily supported in such policies by adjusting the final resource price depending on the required QoS level. The main drawback of these policies, in addition to the poor scalability due to the centralized management, is the full penalization of free-riding behavior. PPAY and TIM support partially free-riders punishment and SHARE not at all.

The next scheme analyzed is the fixed contribution based incentive mechanisms, which also cannot effectively prevent free-riders in P2P systems. However, its principle can be incorporated into the real-time reciprocity-based scheme. This is when a node is consuming resources, it is required to make a minimum amount of resources available to either the resource provider or the whole community.

Taking reputation systems into consideration, most are impractical due to overhead introduced by calculating the peers' reputation. However, this scheme has great potential for preventing free-riding effectively, and the complexity of the mechanisms can be reduced if fewer peers are contacted. Another important consideration is that the metrics for trust value calculation should be context-aware and behavior-sensitive. Moreover, most of the reputation networks use user-specified pseudonyms, that is not resilient to the whitewash and sybil attacks. On the other hand, the service exchange ring-based incentive mechanisms can meet most of the requirements. However, the feasibility of this scheme largely depends on the search mechanisms of the underlying P2P networks.

PowerTrust [RY07] proposes a Look-ahead Random Walk algorithm in which all the nodes calculate the trust values based on their neighbors. A DHT is applied to assign every node a trust value manager, which collects all the ratings about this node and submits these ratings to the trust manager. This feature hinders the adaptation of PowerTrust to P2P systems without

a DHT-based search engine. PowerTrust significantly reduces the calculation complexity and the data redundancy problem associated with trust calculation and storing, improving the scalability. Feldman et al. [MCJI06] proposes a Reciprocative decision function, and introduces the notion of generosity. Generosity measures the benefit an entity has provided relative to the benefit it has consumed. Obviously, a peer is willing to cooperate with the collaborators who are more generous than itself, providing a barrier to free-riders. However, the management of reputation information is critical in such systems to provide a secure incentive mechanism.

Direct or Real-Time reputation systems, like BitTorrent [Coh03], use Tit-for-Tat as a method for resource allocation, where a user's upload rate dictates his download rate. This approach requires that both users share the resource at the same moment or manage historic information in order to detect free-riders and reward contributors.

Table 2.1: Comparison of incentive mechanisms and policies for P2P systems.

		Schemes							Features											
		Pricing				Reput.			Adaptability		Rewards		Free-Rider Penalization		Overhead Problem					
Method	Scheme/Policy	TB	AB	CM	AC	IND	DIR	GT	Topology	Search	Contribution Evaluation	Reward	Achieved	Method	Processing	Messages	Distributed	QoS Support	Enforcement	Scalability
Pricing	MONETARY								yes	yes	Local	Money, virtual token	yes	No Service	yes	yes	no	yes	System Identities	no
	Ppal	✓		✓					yes	yes	Local	Virtual Coins	Partial	No Service	No	yes	No	Yes	System Identities	no
	Share	✓			✓				yes	yes	Global	Virtual Currency	no		yes	yes	No	yes	System Identities	no
	Tim		✓			✓			yes	yes	Local	Money	Partial	No Service, No Admission	no	yes	Yes	no	weighted voting	no
	FIXED CONTRIBUTION								no	yes	Third Party	Access Rights	Partial	No Service	no	no	No	no	User-specified pseudonyms	yes
Reputation	INDIRECT								yes	yes	Global	Differentiated QoS	Partial	Differentiated QoS	yes	yes	Yes	Partial	System Identities	no
	Feldman					✓		✓	yes	yes	Local	Reputation	yes	No Service	no	yes	Yes	no	shared history	no
	Power-Trust					✓			no	no	Global	Reputation	no	None	no	no	Yes	no	System Identities	yes
	DIRECT						✓		no	no	Local	Differentiated QoS	Partial	Differentiated QoS	no	no	yes	no	User-specified pseudonyms	yes
	Bit-Torrent						✓		no	no	Local	Differentiated QoS	Partial	Differentiated QoS	no	no	yes	no	User-specified pseudonyms	yes
	SERVICE EXCHANGE RINGS					✓			yes	yes	local	Access Rights	ND	No Service	ND	ND	yes	yes	User-specified pseudonyms	ND

TB: Token-based; **AB:** Account-based; **CM:** Commodity Market; **AC:** Auction-based; **IND:** Indirect; **DIR:** Direct; **GT:** Game Theory.

2.3 Scheduling policies

Scheduling policies have always been the focus of interest of many researchers in the distributed computing field. Over the last few years, the appearance of many popular systems for harvesting idle cycles from ordinary users, such as the BOINC project [And04], has raised many new and interesting challenges on this field. Those systems are oriented to CPU-intensive workpile applications and require donors of cycles to manually coordinate through a centralized web site. More general cycle-sharing systems, such as Condor [LMM88], provide automatic scheduling but require a centralized matchmaking service and are limited to members of participating institutions. To deal with these problems, in [LZZ+04], the authors present a scheduling infrastructure that supports automatic scheduling for different classes of parallel applications in an open environment. They claim that, depending on the application's features, the scheduling needs are clearly distinct. Given this, they organize P2P cycle-sharing applications into four classes: those which consume huge amounts of compute, those that require the results within a specified deadline, those with loose coordination among subtasks and those with minimal computational needs but which placement throughout the Internet.

In [BCF+08], the authors compare the performance of centralized and decentralized strategies for scheduling bag-of-tasks applications in desktop grids and conclude that, for large-scale platforms, a centralized scheduler with global coordination may be unrealistic. In Tycoon [LRA+05], a distributed market-based resource allocation system is measured. The job scheduling and resource allocation in Tycoon is based on a decentralized auction mechanism. Every peer runs its own auction independently for its local resources, without coordination. This feature can lead to too many scheduling messages being generated in the system for good allocation. The Shirako [ICG+06] system presents another mechanism for on-demand leasing of shared networked resources based on brokers that implement the resource selection. However, the system does not define how the different brokers are coordinated with each other. In the OurGrid P2P system [ABCM07], the load management protocol is based on complete broadcast messaging, flooding the system network with an uncon-

trollable number of messages.

Furthermore, in [NJAB08] the authors investigate the trade-off between two different techniques for mapping the parallel tasks onto the computational resources: bin-packing schedulers and replicated schedulers. The first approach requires complete and accurate information about the applications and the network environment. The second approach does not use any information but, instead, applies the principle of task replication to achieve good performance. Each of these approaches has its drawbacks; attaining accurate and complete information about resources and applications is not always possible in a peer-to-peer environment, while the redundancy of replication schedulers means an extra consumption of resources. Taking all of this into account, they propose scheduling heuristics that use any available information to perform efficient scheduling of BoT applications. Their results show that judicious use of whatever information is available leads to a reduction in resource consumption, without compromising the application's performance. Despite this, they do not consider information about the network, which can be a strong disadvantage if the scheduler has to deal with data-intensive applications.

The mechanism proposed in this thesis is designed to work on an open environment where any peer can, at the same time, share its resources and submit applications to the system. Moreover, according to [BCF+08], we also believe that a centralized scheduler with global coordination may be unrealistic. Unlike other approaches that may collapse the system with a huge number of messages ([LRA+05] and [ABCM07]), or authors that simply do not explain how the coordination with brokers is implemented ([ICG+06]), the scheduling mechanism proposed in this work achieves near-global assignment of tasks using only local information. Furthermore, in contrast to [NJAB08], different kinds of parallel job have been taken into account during the design process. That allows the network administrator to adapt the system to work properly with very different kinds of parallel jobs with the aim of ensuring efficient performance.

2.4 P2P computing platforms

In this section, some popular peer-to-peer systems of the literature are explained in detail. Their schedulers and incentive mechanisms are the most similar to the ones proposed in this thesis, so there is an special emphasis in detail their operation.

2.4.1 Tycoon

The purpose of Tycoon [LRA+05] is to allocate compute resources, like CPU cycles, memory, network bandwidth, etc. to users economically and efficiently. In other words, the resources are allocated to the users who value them the most. To give users an incentive to truthfully reveal how much they value these resources, users use a limited budget of credits to bid for resources. The form of a bid is (h; r; b; t), where h is the host to bid on, r is the resource type, b is the number of credits to bid, and t is the time interval over which to bid. This bid says, "I'd like as much of r on h as possible for t seconds of usage, for which I'm willing to pay b". This is a continuous bid in that it is in effect until cancelled or the user runs out of money.

The user submits this bid to the auctioneer that runs on host h. This auctioneer calculates b_i^r/t_i^r for each bid i and resource r and allocates its resources in proportion to the bids. This is a "best-effort" allocation in that the allocation may change as other bids change, applications start and stop, etc. Credits are not spent at the time of the bid; the user must utilize the resource to burn the credits. Note that the auctioneers are completely independent and do not share information. As a result, if a user requires resources on two separate hosts, it is his responsibility to send bids to those two markets. Also, markets for two different resources on the same host are separated.

This service model has two advantages. First, the continuous bid allows user agents to express more sophisticated preferences because they can place different bids in different markets. Specific auctioneers can differentiate themselves in a wide variety of ways. For example, an auctioneer can increase the amount of a resource (e.g. more CPU cycles), better quality-of-service (e.g., a guaranteed minimum number of CPU cycles), a favorable network location,

etc. A user agent can compose bids to satisfy user preferences whenever it sees fit. Secondly, since the auctioneers push responsibility for expressing sophisticated bids onto user agents, the core infrastructure can remain efficient, scalable, secure, and reliable. The efficiency and scalability are a result of using only local information to manage local resources and operating over very simple bids. The security and reliability are a result of independence between different auctioneers.

2.4.1.1 Architecture

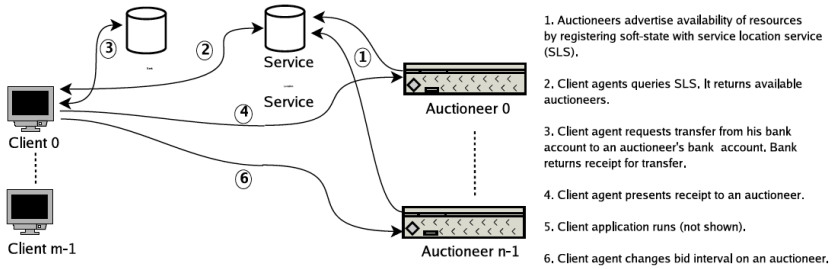


Figure 2.5: Overview of how the Tycoon components interact.

- **Service Location Service:** Auctioneers use this service to advertise resources, and agents use it to locate resources (steps 1 and 2 in Figure 2.5). The prototype uses a simple centralized server, which allows it to be robust against many forms of hardware and software failures.
- **Bank:** The bank maintains account balances for all users and providers. Its main task is to transfer funds from a client's account to a provider's account (step 3 in Figure 2.5). The assumption is that the bank has a well-known public key and that the bank has the public keys of all the users. These are the same requirements for any user to securely use a host with or without a market-based resource allocation system.
- **Auctioneer:** Auctioneers serve four main purposes: management of local resources, collection of bids, allocation of resources to users according to their bids and advertisement of the availability of local resources.

- **Agent:** The role of a tycoon agent is to interpret a user's preferences, examine the state of the system, make bids appropriately, and verify that the resources have been provided. The agent is involved in steps 2, 3, 4, and 6 of Figure 2.5.

- **Funding Policy:** Funding policy determines how users obtain funds. Two possibilities are open and closed loop-funding policies. In an open loop funding policy, users are funded at some regular rate. In a closed-loop (or peer-to-peer) funding policy, users themselves bring resources to the system when they join. They receive an initial allotment of funds, but they do not receive funding grants after joining. Instead, they must earn funds by enticing other users to pay for their resources.

Summarizing

The Tycoon system applies market-based principles, in particular an auction mechanism, for resource management. On one hand, the auctions are completely independent without any centralized control, since every resource owner in the system coordinates its own auction for local resources. On the other hand, the Tycoon system provides a centralized Service Location Service (SLS) for super-schedulers to index resource auctioneers' information. By doing so, Tycoon manages the shared resources efficiently, but at the expense of the scalability of the system.

Another weak point of the network can be found on its funding policy. As said before, the users are rewarded at the moment of joining the system with some initial funds. That policy can encourage users to behave selfishly by joining the system only to spend the initial funds. After that, they may reconnect again with a new identity just to take advantage of that.

Moreover, the bank mechanism proposed in Tycoon is a very interesting solution for managing all the users' accounts but it has some deficiencies, like scalability problems or the vulnerability of the bank. Despite this, we believe that a centralized bank-like entity is a good solution since it allows security policies to be applied to fight against cheating and malicious peers.

2.4.2 Shirako

Shirako's leasing architecture derives from the SHARP framework for secure resource peering and distributed resource allocation [FCC+03]. The participants in the leasing protocols are long-lived software entities (actors) that interact over a network to manage resources.

- Each guest has an associated service manager that monitors application demands and resource status, and negotiates to acquire leases for the mix of resources needed to host the guest. Each service manager requests and maintains leases on behalf of one or more guests, driven by its own knowledge of application behavior and demand.

- An authority controls resource allocation at each resource provider site or domain, and is responsible for enforcing isolation among multiple guests hosted on the resources under its control.

- Brokers (agents) maintain inventories of resources offered by sites, and match requests with their resource supply. A site may maintain its own broker to keep control of its resources, or delegate partial, temporary control to third-party brokers that aggregate resource inventories from multiple sites.

These actors may represent different trust domains and identities, and may enter into various trust relationships or contracts with other actors.

Shirako is a toolkit for constructing service managers, brokers, and authorities, based on a common, extensible leasing core. A key design principle is to factor out any dependencies on resources, applications, or resource management policies from the core. This decoupling serves several goals. First, the resource model should be sufficiently general for other resources such as bandwidth-provisioned network paths, network storage objects, or sensors. Second, shirako should support development of guest applications that adapt to changing conditions. Finally, shirako should also make it easy to deploy a range of approaches and policies for resource allocation in the brokers and sites.

Note that Shirako has no globally trusted core; rather, one contribution of the architecture is a clear factoring of powers and responsibilities across a dynamic collection of participating actors, and across pluggable policy modules and resource drivers within the actor implementations.

2.4.2.1 Architecture

Figure 2.6 summarizes the protocol interactions and extension points for the leasing system. An application-specific service manager uses the lease API to request resources from a broker. The broker issues a ticket for a resource type, quantity, and site location that matches the request. The service manager requests a lease from the owning site authority, which selects the resource units, configures them (setup), and returns a lease to the service manager. The arriving lease triggers a join event for each resource unit joining the guest; the join handler installs the new resources in the application. Plug-in modules include the broker provisioning policy, the authority assignment policy, and the setup and join event handlers.

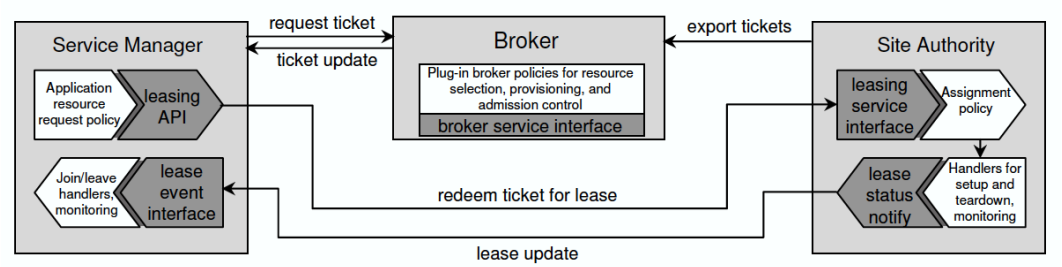


Figure 2.6: Shirako Architecture.

- **Resource Leases.** The resources leased to a guest may span multiple sites and may include a diversity of resource types in differing quantities. Each lease binds a set of resource units from a site (a resource set) to a guest for some time interval (term). In the current implementation, each lease represents a number of units of resources of a single type. For example, a lease could represent a reservation for a block of machines with specified processor and memory attributes (clock speed etc.), or

a storage partition represented by attributes such as capacity, spindle count, seek time, and transfer speed.

- **Brokers.** Figure 2.6 depicts a broker's role as an intermediary to arbitrate resource requests. The broker approves a request for resources by issuing a ticket that is redeemable for a lease from some authority, subject to certain checks by the authority. The ticket specifies the resource type and number of units granted, and the interval over which the ticket is valid (the term). Sites issue tickets to the brokers for their resources; the broker arbitration policy may subdivide any valid ticket held by the broker.

Summarizing

Shirako is a system for on-demand leasing of shared networked resources across clusters. Shirako's design goals include: autonomous providers, who may offer resources to the system on a temporary basis but retain final control over these; adaptive guest applications that lease resources from the providers according to changing demand; pluggable resource types, allowing participants to include various types of resources, such as network links, storage and computing; brokers that provide guest applications with an interface to acquire resources from resource providers; and allocation policies at guest applications, brokers and providers, which define the manner resources are allocated in the system.
The leasing abstraction provided by Shirako is a useful basis for coordinating resource sharing for systems that create distributed virtual execution environments of networked virtual machines. However, the complexity of its design and the rigidity of its agents (brokers) make it more appropriate for Cloud architectures than for P2P computing ones. Furthermore, it is not clear how the different brokers in the system interact with each other. Despite this, it has some interesting ideas that inspired the design of our scheduler. For example, the way the resources are allocated in shirako encouraged us to divide the system into small areas controlled by a manager.

2.4.3 OurGrid

OurGrid [ACBR03] is an opensource grid middleware based on a peer-to-peer architecture. It is mainly developed at the Federal University of Campina Grande (Brazil), which also runs an OurGrid instance named "OurGrid", in production since December 2004. Anyone can freely and easily join it to gain access to a large amount of computational power and run parallel applications. This computational power is provided by the idle resources of all the participants, and is shared in a way that ensures that those who contribute more, receive more when they so need. Currently, the platform can be used to run any application whose tasks (i.e., parts that run on a single machine) do not communicate among themselves during execution, like most simulations, data mining and searching.

OurGrid is an open, free-to-join, cooperative grid in which labs donate their idle computational resources in exchange for accessing other labs' idle resources when needed. It uses a peer-to-peer technology that makes it in each lab's best interest to collaborate with the system by donating its idle resources. OurGrid leverages from the fact that people do not use their computers all the time. Even when actively using computers as research tools, researchers alternate between job execution (when they demand computational power) and result analysis (when their computational resources go mostly idle).

For OurGrid to be useful, it must be fast, simple, scalable, and secure. These were the four major goals that guided the design of OurGrid. Achieving these goals was a very challenging task. In order to simplify the problem somewhat, at least for now, the developers reduce OurGrid's scope to supporting Bag-of-Tasks (BoT) applications. As stated above, BoT applications are parallel applications whose tasks are independent.

2.4.3.1 Architecture

Figure 2.7 shows the main components of OurGrid and how they interact between themselves.

- **OurGrid Broker.** The OurGrid Broker (originally called MyGrid) is the scheduling component of the OurGrid solution. A machine running

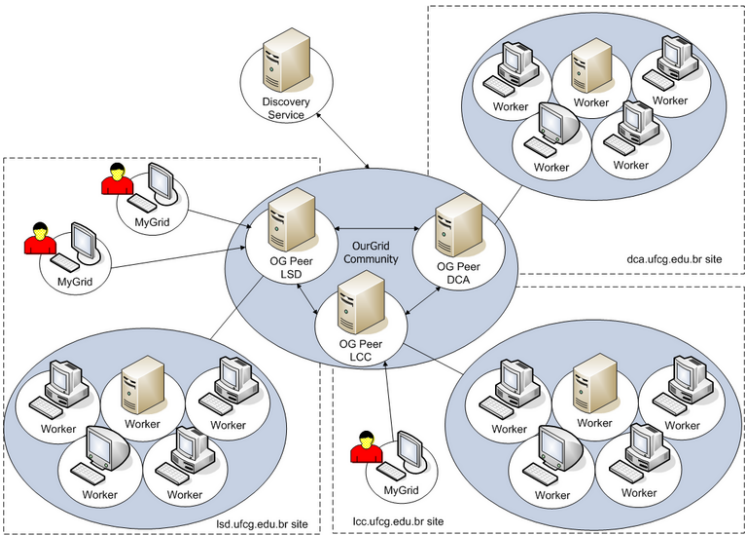


Figure 2.7: Ourgrid Architecture.

the Broker is called the home machine, which is the central point of a grid. During the processing of jobs, it acts as the grid coordinator, scheduling the execution of tasks and doing all the necessary data transfer to and from grid machines. Due to its central role, grid configuration and management, as well as job specification, is done on the home machine. For these reasons, users will likely use their desktop as the home machine for their grid. This approach decentralizes access to the grid allowing multiple users, each using their own installation of the Broker, to do concurrent processing. The Broker is OurGrid's user front end. It provides all the necessary support to describe, execute, and monitor jobs. Job processing is done by machines running OurGrid Workers. During the execution of a job, the Broker gets Workers on-demand from its associated Peer. It is the Broker's role to schedule the tasks to run on the Workers and to deploy and retrieve all data to/from Workers before and after the execution of tasks.

- **Peers.** An OurGrid Peer runs on a machine called the peer machine. The main role of a Peer is to organize and provide worker machines that belong to the same administrative domain. From the user's perspective,

a Peer is a Worker provider, i.e., a network service that dynamically provides Workers for task execution. From an administrative point of view, a Peer determines how and which machines can be used as workers. In Figure 2.7, there is one Peer for each administrative domain. This architecture allows different site administrators to enforce their own policies regarding the use of their Workers.

- **Workers.** The OurGrid Worker component runs on each machine that will be available for task execution. The Worker provides necessary access functionality to the home machine. It also provides some basic support for instrumentation and fault handling. Furthermore, combined with the OurGrid Peer, it allows for the use of machines in private networks. In practice, any computer connected to the Internet can be used as a worker machine, even if it lies in a different administrative domain or behind a firewall. In Figure 2.7, administrative domains, possibly using their own intranets, are illustrated as rectangles containing Workers.

2.4.3.2 Network of Favours

To encourage resource contribution to the network, OurGrid uses a resource allocation mechanism called Network of Favours. The Network of Favours is an autonomous reputation scheme that rewards Peers that contribute more. This way, there is an incentive for each Peer to contribute as much as possible to the system.

In this mechanism, when a participant A is doing a favor to participant B when A allows B to use its resources. The philosophy of the Network of Favours is that every participant does favors to other participants expecting the favors to be reciprocated. When a user has many requests to deal with, it prioritizes those who have done favors to it in the past. The more favors a participant does, the more it expects to be rewarded. The participants locally account their favors and cannot profit from them in another way than expecting other participants to do them some favors . Detailed experiments have demonstrated the scalability of the network of favors [ABCM07], showing that the larger the network becomes, the more fairly the mechanism performs.

Summarizing

OurGrid is a peer-to-peer computing architecture that implements a very consistent mechanism for managing the relations between the peers inside the network. This mechanism, called Network of Favors prioritizes the peers with high reputation, thus it motivates sharing. From the scalability point of view, the main advantage of OurGrid is that there is no need to store global reputation. The reputation of neighbors is maintained in each peer and a communicating overhead is avoided.

This scheme is focused on solving free-riding, although it limits the accuracy of job scheduling. This is because the reputation of a peer is not propagated through the system, so a master peer will only have objective information if a peer has collaborated with it before. Thus, scheduling success depends on the collaboration of the peers after the mapping of tasks. If the target peers cooperate with each other, their respective reputation will increase, so performance will be good. However, if the target peers do not cooperate, job performance will not be guaranteed.

Since the OurGrid was born, the authors have published many interesting works related to its incentive and scheduling mechanisms. All these work inspired us in many aspects on the design process of our proposal. Between others, we can highlight the choice of a non-negative credit system that protects the networks against malicious behavior, known as identity-changing cheating. This kind of behavior is quite common on P2P networks and happens when a user creates a new account in order to clean some previous bad behavior. However, there are some others typical kinds of cheating that are not taken into account in OurGrid and which are important enough to be considered in our proposal.

2.4.4 CompuP2P

CompuP2P [GSS06] uses a peer-to-peer architecture for creating markets to trade computing resources, such as CPU cycles, disk space, etc. CompuP2P creates different markets for different amounts of a computing resource, referred to as a commodity. Nodes that are responsible for running different commodity markets are termed "market owners" (MOs). MOs are dynamically re-assigned as nodes leave and join the network. A MO does the job of a matchmaker between sellers and buyers, and maintains information about the sellers. This information can include such things as available compute power and/or disk space, operating system type and version, platform type, price requirement, etc. Upon receiving a request from a client, the MO returns the information about the seller that best meets the client's requirements. A Chord-based protocol is used for market creation and lookup, and with a high probability of both sellers and buyers of a commodity converging on the same market, i.e., both sellers and buyers contact the same MO that is responsible for running the market for that commodity. It must be noted that a single physical node can be a MO for various commodities.

CompuP2P is designed to take users' selfishness into account, and uses ideas from game theory and microeconomics to price computing resources. CompuP2P allow users to define their policies regarding what, when, how, and by whom their resources can be used. Moreover, it allows users to specify their task requirements while accessing the system resources.

2.4.4.1 Architecture

CompuP2P presents a three-layer architecture.

- **Computing Resources layer.** This layer refer to various distributed resources, such as compute power, disk space, files, etc., that exist in any large Internet-scale system. These resources belong to different nodes that are part of the underlying P2P network. In CompuP2P, nodes joining the system are organized in a Chord ring [SMK+01].

- **Resource Trading layer.** The functionality of this layer can be further divided into three sub-layers. The market lookup protocol for ensuring

that sellers and buyers looking to trade a commodity converge on the same market. The resource pricing protocol for ensuring that both sellers and MOs are suitably compensated for the service they provide to clients. And the dynamic market creation protocol, used for selecting nodes that act as MOs for specific commodities.

- **Service layer.** The service layer accepts service requests from a user. A service can be a computation task or a data storage request. A computation task is submitted by a user in the form of an XML task file. The task file is parsed and appropriate computing nodes in the network that can execute the associated sub-tasks are determined. The service layer allows a user to specify their task requirements while accessing the system resources. Moreover, a user interested in backing up the local data can also request the service layer to search for appropriate storage nodes in the network. The data is usually replicated at multiple remote nodes and is stored in either plain-text or encrypted format.

2.4.4.2 Resource Trading

This layer is responsible for creating and managing markets. Each node based on its current and past load estimates the average number of resources that would remain idle in future. Suppose a node determines that it has C cycles/sec available for the next T time units (where T a long enough time period) that it can provide or make available to others for processing.

Since different nodes have different amounts of compute power to sell and purchase, it is necessary to create suitable markets to permit buyers and sellers to come together and trade the amount of compute power they require.

The term commodity represents a range of idle CPU cycles/sec values. Each market deals in only one type of commodity (i.e., homogeneous markets). A single physical node may be responsible, i.e., be a market owner (MO), for more than one market.

Figure 2.8 depicts how nodes with different values of idle compute power C join different markets.

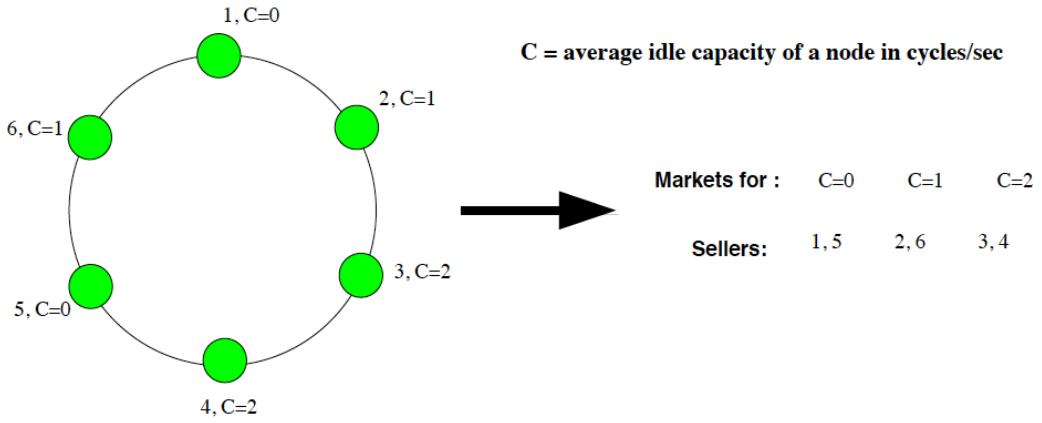


Figure 2.8: Creation of markets for CPU cycles in CompuP2P.

2.4.4.3 Pricing

Pricing is non-trivial when there are either multiple sellers from a buyer's point of view or when a buyer is trying to minimize its cost of processing (again assuming multiple sellers). Utilizing the model that a transaction involving the trading of compute power can be modeled as a one-shot game and using the results from game theory and microeconomics (the classical Prisoner's dilemma problem [Osb04] and Bertrand oligopoly [Bay05], respectively), it can be concluded that long-term collusion among compute power sellers (and MO) is unlikely to occur. In the one-shot Prisoner's dilemma game, non-cooperation is the only unique Nash equilibrium strategy for the players. In fact, the model of Bertrand oligopoly suggests that sellers (irrespective of their number) would not be able to charge more than their marginal costs for selling their resources (see [Osb04] for a game-theoretic derivation of this result). In the Bertrand oligopoly the sellers' strategy is to set "prices" (as opposed to "outputs" in the Cournot oligopoly), and is thus more reasonable to assume in the context of CompuP2P. In CompuP2P, all the sellers in a market sell the same amount of a computing resource. As a consequence, in CompuP2P, irrespective of how many there are in a market, sellers set prices equal to their marginal costs only. Among other things, the marginal cost of providing a computing resource can include listing price, bandwidth cost for message

exchange, etc., and is represented by MC_i for a node, i.

Since the best pricing strategy for sellers is to charge equal to their marginal costs, it results in zero profits for them. Therefore, sellers would not be motivated to sell their compute power unless some other incentive mechanisms were devised for them. To solve this problem, CompuP2P use the technique employed in the Vickrey auction ([Nis99, Vic61]). When a seller when joins a market, it provides the MO with its marginal cost information. A buyer, looking to minimize its cost, selects the seller with the least marginal cost, but the amount it has to pay to the seller is equal to the second lowest marginal cost value listed on the market. This selection scheme is called the reverse Vickrey auction. The above strategy provides non-zero profit to the selected seller and ensures that sellers state their correct marginal costs to the MO (see [Vic61] for the truth-eliciting property of the Vickrey auction). The strategy is also inherently secure because even if sellers learn about the posted marginal costs, they cannot take undue advantage of that information to post a lower marginal cost than their actual values.

Summarizing

CompuP2P is an architecture for sharing computing resources in peer-to-peer networks. It creates dynamic markets of network-accessible mutable resources in a completely distributed, scalable, and fault-tolerant manner. CompuP2P uses ideas from game theory [Osb04] and microeconomics to devise incentive-based schemes.

Thanks to the Chord-based structure, CompuP2P can boast of a better scalability than many other P2P computing alternatives. Otherwise, it has some important deficiencies, like not taking free-riders into account or omitting some kinds of cheating which are very common on P2P computing networks. Besides this, the main problem of the system is trying to manage the dynamic resources shared in the network as static resources. This happens when the markets are created and the nodes are classified depending on their computational resources. By doing so, everyone will be allocated to a group that will be the same until a task is assigned to them. The problem appears if the user changes the workload of the machine, which is as easy as opening an application or starting to reproduce a multimedia file. If that happens, the node will be allocated to the wrong market and the system may be expecting it to supply something out of reach.

Otherwise, it has inspired us in some aspects, like the use of the reverse Vickrey auction as the pricing strategy. This strategy protects the system against malicious behavior by users while ensures a profit for the peers sharing their resources, thus encouraging sharing. Another strong point of the system that helped us in the designing of our mechanism is the methodology used to reward the market owners, entities that are quite similar to the manager peers that we have in our mechanism. This methodology defines a payment policy that encourage the market owners to manage the shared resources efficiently, rewarding them for always choosing the best worker for each task execution.

2.4.5 CoDiP2P

CoDiP2P [CBR+08] is a decentralized mechanism for distributing computation and resource management that takes into account the system heterogeneity by using the P2P paradigm. The design of CoDiP2P is focused on hiding system complexity from the programmers and users. The hierarchical topology in managing and maintaining the system-growing capacity favors the good scalability of the system. Fault-tolerance has also been considered by providing self-organization of peers and avoiding centralized components. The system is able to manage its resources efficiently, independently of their heterogeneity, geographical dispersion and volatility.

2.4.5.1 Architecture

Figure 2.9 shows the CoDiP2P architecture. In this example, the tree has three levels and is made up of four areas. The peers that make up the system have two roles, as workers and managers, and are grouped together by areas. The architecture of CoDiP2P is a tree of areas. In each CoDiP2P area there is only one Manager and N Workers, where N depends on the network properties, bandwidth and latency. The main goal of the manager is to manage peers in the same area and schedule jobs to be executed over the workers. At the same time, these workers can also submit jobs to their corresponding manager.

The properties of CoDiP2P are the following: scalability to ensure that the system supports the massive entry of peers; distributed management to harness the computing resources offered by the nodes making up the system; fault tolerance to prevent the high possibility of a peer failure and replace it by ensuring the stability, robustness and performance of the global system; self-organization is the way that each peer can be a manager or a worker dynamically according to the needs of the system; heterogeneous resource management to manage the scheduling and load balancing of tasks among the peers efficiently, independently of their heterogeneity, geographical dispersion and volatility.

The main features of CoDiP2P involved in resource management are the followings: balanced peer *insertion* of new peers; *scheduling* of multitasking

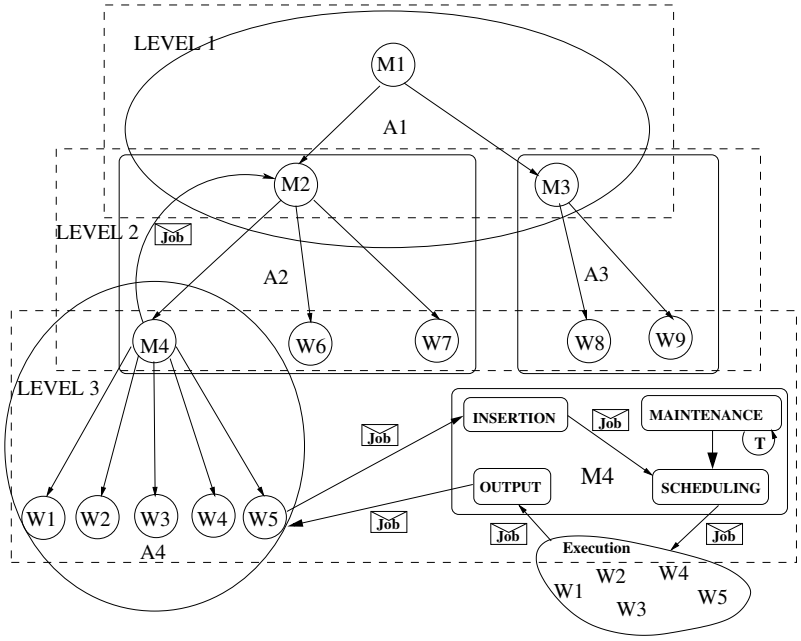


Figure 2.9: CoDiP2P, a tree-like architecture.

jobs; *maintenance* which keeps the system updated in periods of T seconds and peer *output*, also used to re-balance the tree when a peer leaves the system. The insertion and output cost is $\theta(\log_{Size(A_i)}(N))$, where $Size(A_i)$ is the area capacity and N is the number of peers in the system. The cost of the maintenance algorithm is hardly worse, $\theta(Size(A_i))$. These low costs justify the choice of a tree-like architecture.

Summarizing

CoDiP2P is a decentralized mechanism for distributing computation and resource management which takes the system heterogeneity into account by using the P2P paradigm. The design of CoDiP2P is focused on hiding system complexity from the programmers and users. The hierarchical topology in managing and maintaining the system-growing capacity favors the good scalability of the system. Fault-tolerance has also been considered by including self-organization of peers and avoiding centralized components. The system is able to manage its resources efficiently, independently of their heterogeneity, geographical dispersion and volatility.

CoDiP2P has been developed by our group over during the last few years. Providing this system with an efficient scheduling mechanism and an effective incentive mechanism was one of the main motivations for this thesis. For this reason, although technically it may not be catalogued as related work, it has been included in this chapter because this architecture is the basis of the present project.

However, it must be stated that the incentive and scheduling mechanism presented in this thesis are not exclusively for this architecture. It can be deployed on many other P2P computing systems, provided they fulfill certain requirements. Those requirements are explained in section 3.2, before detailing the proposed mechanism.

2.4.6 Summary of Architectures

Table 2.2 summarizes the analyzed P2P architectures. The summary is focused on the characteristics of scheduling and incentives policies. Also, the main features of the integrated incentive and scheduling mechanism proposed in this thesis, incorporated in the CoDiP2P architecture are detailed.

Our incentive policy is based on credits (virtual money) and an account-based scheme to enforce peer collaboration. With regards to the analyzed architectures, our proposal/policy suggests a differentiated distributed incentive method capable of discouraging free-riding and preventing the threats of cheating through whitewashing and dumping.

Table 2.2: Comparison of Distributed P2P architectures.

		P2P Computing Architectures				
	Feature	**Tycoon**	**Shirako**	**OurGrid**	**CompuP2P**	**CoDiP2P**
System	**Nodes**	Clusters	Clusters	Peers	Peers	Peers
	Resources Shared	CPU cycles, memory and network bandwidth	CPU cycles, storage, network bandwidth	CPU cycles, storage, network bandwidth	CPU cycles, storage, network bandwidth	CPU cycles, storage, network bandwidth
	Application Type	Parallel Dependent Tasks	Parallel Dependent Tasks	Parallel Independent Tasks	Parallel Independent Tasks	Parallel Independent Tasks
Incentive Scheme	**Scheme**	Pricing	Pricing	Reputation	Pricing	Pricing
	Policy	Auction-based	Token-based	Indirect	Game Theory	Account-based
	Organization	Distributed markets	Distributed brokers	Autonomous	Chord conected markets	Distributed
	Scope	Local	Local	Local	Local	Global
	Free-rider Penalization	No	No	Yes	No	Yes
	Whitewashing Penalty	No	No	Yes	No	Yes
	Dumping Cheating	Yes	No	No	Yes	Yes
Scheduling	**Algorithm**	Auction-Share	Leasing	Storage-affinity	Auction-based	Auction-based
	Allocation	Auction-Based	Leasing	Best-effort with Replication	Auction-based	Auction-based
	Organization	Distributed	Distributed	Distributed	Distributed	Hierarchical and Distributed
	Scope	Global	Global	Two-Tiers (local & global)	Local (market)	Three-Tiers (local, area, global)
	Admission Control	No	Yes	No	Yes	Yes
	QoS Support	Yes (latency, utilization and risk)	Yes	No	No	No
	Price model	Users Fixed Price	-	-	Marginal cost	Marginal Cost
	Aceptance criteria	First/Second Price	-	-	Reverse Vickery Auction	Reverse Vickery Auction
	Messages	O(N),N=aution agents	O(N),N=Brokers	O(N), N=local schedulers	O(log N), N=peers	O(log N), N=areas

The proposed scheduling algorithm stands for its hierarchical and distributed nature based on several levels of scheduling (peer, group of peers and global). The scheduler and resource allocation are both auction-based, taking their decisions according to the law of supply and demand of resources. The resource price construction is founded on the marginal cost defined by the peer and it uses the reverse Vickery auction method to match supply and demand requests. Other important feature of our algorithm is its scalability. It is designed to restrict the number of messages required to schedule a job. This feature avoids flooding the overlay network with messages and guarantees the scalability of the whole system.

Chapter 3

DISIM

3.1 Introduction

This chapter introduces a new distributed and cooperative two-level mechanism for scheduling jobs and incentivizing the users in a P2P computing network. This model improves scalability for large-scale P2P computing with super-peer topologies. In order to guarantee this scalability, the proposed mechanism takes a two-level P2P architecture into consideration. The high-level is composed of a set of super-peers interconnected by means of an overlay. Each super-peer is responsible for a group of peers. Together, all these groups compose the lower level of the system. At the same time, each of these groups is made up of a different number of peers. Peers are the smallest autonomous unit and own the computational resources shared in the system, mainly CPU, Memory and Storage.

At the low-level, the traditional problem of mapping a group of tasks into a set of well-known computational resources is dealt with. At this level, long-established algorithms from the literature could be employed in order to achieve an efficient scheduling. Despite this, we propose a new credit-based mechanism to enforce collaboration inside the system. The allocation of job tasks, performed by a super-peer is based on the reverse Vickrey auction, thereby achieving the optimum allocation of resources since it has complete information about the overall area. Moreover, this strategy also protects the system against dumping cheating.

Due to scalability issues involved in maintaining the global information, this mechanism is integrated into another one that works at a higher level, where the super-peers are connected to each other. With the aim of reducing the information requirements, a method that manages the most important system resource information is defined. It provides a value that reports about the computing capacity controlled by each super-peer.

Section 3.2 explains the main features of the overall P2P computing framework that was taken into account when designing the mechanism. Section 3.3 explains how the credits are used and introduces their main properties and benefits. Section 3.4 explains how the mechanism manages the low-level of the network, that is, how each super-peer controls the events generated within its own group of peers. In Section 3.5, the upper-level scheduling and incentive mechanism, jointly with its components and their operating details, is described.

3.2 P2P Computing Framework

As explained in Section 1.2, not all P2P networks are equally structured. There are many different kinds of overlays with very different properties and features. This section details the P2P network overlay that has been taken into account for the proposed incentive and scheduling mechanism. Although the mechanism is not designed for only one kind of network structure, the overlay must have some specific features in order to be deployed. For this reason, this section introduces and explains these features with the aim of identifying the kind of networks that can work properly with our proposals. First, the overall architecture is explained before detailing how the scheduling scheme is adapted to the structure presented. Finally, the incentive polices are introduced by describing the operation of the proposed credit-based mechanism.

3.2.1 Architecture

The system is made up of several associations of peers grouped together in an area. Figure 3.1 shows an example of a possible grouping of peers. These

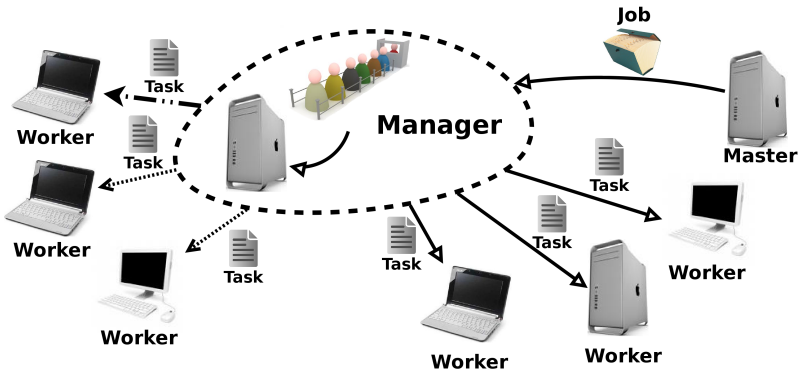


Figure 3.1: Example of an area with eight peers.

groups of peers are made up of one super-peer and n additional peers, where n depends on the network properties, bandwidth and latency. Basically, the system defines three kinds of roles that can be assumed by peers, Workers (W), Managers (M) and Masters (MS). Workers are those peers who provide their resources to execute tasks. Since several workers are grouped together, it is necessary for one of them to take the responsibility of managing the group. This entity is called the manager and is responsible for controlling the system and scheduling jobs. A master is a peer that has submitted a job to the system and monitors its execution to completion. When a peer, whether a manager or a worker, launches a job into the system, it will also become a master, while simultaneously maintaining its previous role.

Figure 3.1 also shows the application model taken into account in such system. The master peer starts the job main task, which splits the computational work into several independent tasks. Those tasks are submitted together to the manager, which is responsible for distributing them among the workers of the system. Finally, the job main task will also collect the results from the completed tasks and generate the final outcome. Note that the communication requirements in this process only include sending the program and the necessary data to execute such tasks and receiving the partial results from the workers. The ratio between those communications and the required execution time of each task will define the granularity of the job.

The system can have multiple areas interconnected through the managers by means of an overlay, making up the upper level. Figure 3.2 shows how the

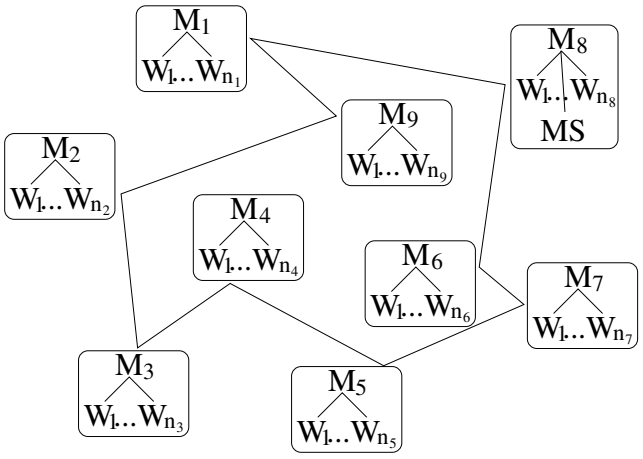


Figure 3.2: Example of system overlay where K=2.

managers are interconnect to each other. Most popular structured P2P overlays have this architecture (CAN, Chord, Kademlia, Pastry, Tapestry, Viceroy, Bruijn, etc.) [EAH05]. The common characteristic of these overlays is that each node has a fixed number of K neighbors (links), where K << N, with N being the number of peers in the system. Our system is even designed for environments where each peer can have different K values. All the policies proposed in this thesis are independent of the number of areas and the interconnection overlay.

The main goals of this kind of system architecture are the following:

- Scalability. Ensures that the system supports the massive entry of peers.
- Distributed management. Harnesses the computing resources offered by the nodes making up the system.
- Robustness. Deals with the high possibility of a peer failure and replace it without affecting the working of the global system.
- Self-organization. Allows peers to change their role dynamically according to the system needs.
- Heterogeneous resource management. In order to perform efficiently the scheduling and load balancing of tasks among peers.

3.2.2 Scheduling

In the upper level, managers deal with more scheduling information, although less accurate. At this point, each manager has a distributed scheduler for assigning computing tasks among areas. Thus, global scheduling involving all computing resources is achieved by collaborations among distributed schedulers on individual areas. Due to this two-level topology, we divide our mechanism into two parts: the Global Mechanism (GM) and the Local Mechanism (LM). The former is responsible for propagating the task through the overall system, always trying to allocate more tasks to the "best" super-peers, and the latter defines how the tasks are assigned to the peers inside each area.

To improve scalability, the amount of resource information used by the GM is limited to the knowledge of adjacent areas. We assume that information for doing local assignments is not very large and the manager has accurate information about the power capacity and occupancy of the area's peers. Furthermore, as an area is usually small in size, the LM can know the state of all peer resources and perform a near-optimal task assignment. The area scheduling policy is based on the reverse Vickrey auction strategy to avoid peers offering resources at a price lower than their real cost (dumping cheating). The LM is performed by each manager within its own scope, normally a local area network.

3.2.3 Operation

Referring to the incentive policies, our incentive scheme is based on credits and uses an account-based system to manage the transactions between users. The mechanism is located in each super-peer, which coordinates its own group of peers and has direct information about its neighboring super-peers. Figure 3.3 shows how the mechanism works when a user requests computational resources. In step 1, a user, called *master* throughout this process, can be seen submitting a job to be executed to the system. In step 2, the manager of the area will have to decide, according to the user's information, if the job can be launched or not. If the request is accepted, the manager will take into account its neighboring super-peers to see if it is better to delegate some tasks of the job

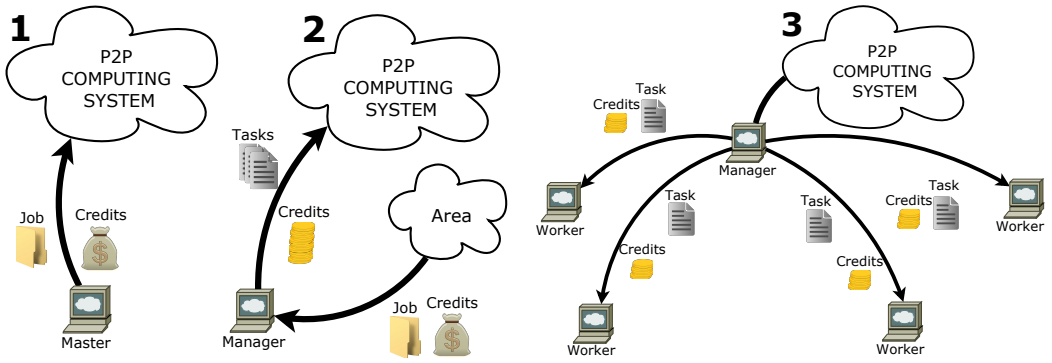


Figure 3.3: Operation of the credit-based mechanism.

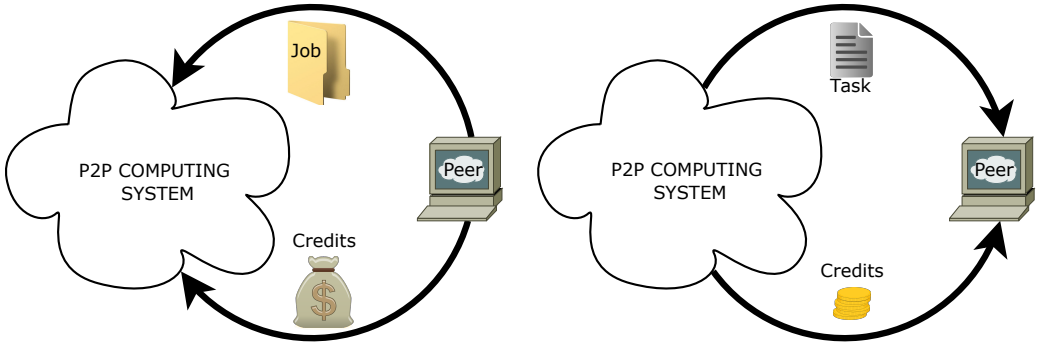
to them. Finally, step 3 shows how a manager assigns the tasks directly to its workers inside an area. Obviously, throughout this process, workers sharing their resources are rewarded with credits from the master. Figure 3.3 also shows how the manager is responsible for such transactions.

3.3 Management of Credits

This section explains how the credits are managed by the system, detailing the main features of the proposed credit-based mechanism and highlighting its most important advantages. Credits are used for accounting the consumption and cession of resources. As can be seen in Figure 3.4a, when a peer submits a job, it pays some credits for its execution. In the same way, when a peer executes a task, it is rewarded with credits as shown in Figure 3.4b. The credit system is the main instrument used by the incentive policy.

The system only imposes one restriction on these dealings: credits cannot be negative. If credits are not limited to zero, peers would be tempted to reconnect to the system with a new identify in order to be treated as new users. This action is known in the literature as ID-Changing cheating. There are two different reasons why a peer may be interested in having a new identity in the network:

- If the system implements a mechanism that rewards newcomer peers to encourage them to participate in the network, one may be tempted to



(a) Peer submitting a job. (b) Peer accepting a task.

Figure 3.4: Credit transaction.

join the system just to spend the initial credits given by the managers and reconnect again (with a new ID) once all these credits have been exhausted.

- If the system has a mechanism that allows users to have negative credits and penalizes them, e.g., with lower priorities in the job queue, a peer could use this kind of cheating to bypass this limitation.

For the same reason, the users have no credits when joining the network. Despite this, if the system is idle and there are no jobs in the manager's queue, a peer with no credits will still be allowed to submit an application. However, it must be said that under normal circumstances (the system being used by peers), newcomers will have to share their resources to earn some credits before being able to launch a job. Under this scenario, the system will create more credits dynamically depending on the network expansion. Next, how the system works at startup is explained:

- At system startup, none of the users have credits. As all the users have the same conditions, the job queue has a FCFS (First-Come, First-Served) scheduling policy. Thus, the first user to launch a job will be allowed to do it for free, but the workers executing the tasks of this job will be rewarded anyway with the corresponding credits.

- When the first worker finishes executing a task, it will be rewarded with

a number of credits. From this point on, the jobs launched by this worker (with credits) will be prioritized over others when competing for computational resources. Thus, the job scheduling policy will become a priority-based queue. In the absence of jobs with priority, jobs will be selected on a FCFS basis. Note that jobs from peers without credits will only be executed when the system is idle.

- If a peer launches a job without enough credits to pay the "bill", the request will be only accepted if there are no competing peers with more credits asking for resources at that moment. If this happens, it will pay with all its credits (leaving its account to zero) but the workers will receive the total number of credits they are entitled to. Note that during this process, some credits will be created by the system itself. This way, the amount of credit grows dynamically with the system and nothing happens if a peer runs out of credits.

Finally, although it is outwith the main scope of this thesis, we would like to give some ideas about how the credits could be underwritten to prevent malicious behavior by users, e.g., creating more credits. One can find many works in the literature about distributed authentication systems designed for P2P networks. For example, in [WZK05], the authors describe an authentication protocol for Peer-to-Peer systems that provides the operating peer with anonymity from other peers and any external users, except the managers. Additionally, the system identifies peers' actions when cooperating with the super-peer. This can serve to control malicious peers trying to violate the rules of the P2P application. In a similar but more recent work ([TT07]), the authors presented a model for data authentication in peer-to-peer storage networks, where data items are stored, queried and authenticated in a totally decentralized way. By extending fundamental cryptographic techniques to distributed environments, they ensure an efficient verification of membership and update operations over dynamic data sets and eliminate the threat of replay attacks. In the proposed mechanism, all the peers are involved in the accounting and authentication tasks that allow the incentive and scheduling policies to be implemented securely and confidently. For this purpose, any of the two pro-

tocols explained above could be used to prevent malicious behavior by users, in other words, to minimize the risk of cheating. Moreover, this last protocol can also be used to solve another common problem of P2P computing systems: the persistence of the data such as that referring to the accounting and the reputation of the users. Note that if some data is stored only in a peer, which if disconnected, will cause that information to be lost. For this reason, it is necessary to use a scheme to store the data in a distributed and replicated way, such as the one presented in [TT07].

3.4 Local Mechanism

This Section is contained in the following paper:

J. Rius, F. Cores and F. Solsona, *"Incentive Mechanism for Scheduling Jobs in a Peer-to-Peer Computing System"*, Submitted to the Journal of Simulation Modelling Practice and Theory, 2011.

3.5 Global Mechanism

This Section is contained in the following paper:

J. Rius, F. Cores and F. Solsona, *"Cooperative Scheduling Mechanism for Large-Scale Peer-to-Peer Computing Systems"* Submitted to the Journal of Parallel and Distributed Computing. Special Issue: NovelArch for HPC, 2011.

Chapter 4

Experimental Results

4.1 Introduction

This chapter presents the experimentation conducted to demonstrate the feasibility and good performance of the proposed incentive and scheduling mechanism. The results are presented and extensively analyzed in Sections 4.3, 4.4 and **??**.

Section 4.2 proposes a model to represent the behavior of the users in a P2P computing network. It specifically defines two kinds of peers, free-riders and collaboratives, and studies the probabilities of moving from one state to another. It must be taken into account that, in order to evaluate (through simulation tests) a mechanism like the one proposed in this thesis, it is necessary to define and model the behavior of peers. Furthermore, this behavior must be as real as possible in order to make our proposals applicable.

4.2 User Behavior

Before evaluating the incentive mechanism proposed in this thesis, the user behavior of peer-to-peer networks was studied. Although there are many works in the literature aimed at modeling the behavior of users in different P2P systems, most of them focus on file sharing.

Studies in [SR06, HWS07] have shown that user behavior is largely invari-

ant across P2P systems. User behavioral patterns are in constant transition, although the broad characteristics are comparable in different systems. Hence, it is pointed out that some of them are very close to observed behavior, e.g., Weibull distributions, and others reflect worst-case or utopian scenarios, e.g., exponential or uniform distributions.

In [AAF+08], the authors studied P2P user behavior in a simulation environment and concluded that the Weibull and Pareto distributions represented realistic behavior for the quantity of resources shared by a peer in the system. Both distributions assumed that a large number of peers share a small number of computational resources while only a few share large amounts of them. They also concluded that the Poisson distribution represents a hypothetical utopian scenario where every peer shares a significant number of resources. Furthermore, the presence of a large number of free-riders has been confirmed by extensive measurement studies [BCC+06, ZSR06, FHKM04]. While the Weibull and Pareto cases represent realistic behavior (i.e., a large number of free-riders), the Uniform case is used as a base for comparison, and the Poisson case represents a scenario where every peer shares a constant number of resources.

In [SR06], Stutzbach showed how session lengths are fitted by Weibull or Log-normal distributions, but not by the exponential or Pareto models. While most sessions in a P2P system are short (minutes), some sessions are very long (days or weeks). This differs from exponential distributions, which exhibit values closer together in length, and heavy-tailed distributions, which have more pronounced extremes (years).

This section presents a model used to represent the user dynamic behavior based on two kinds of peer, collaborative and free-riders, and their changing probabilities. Just as a reminder, free-riders are those peers that do not collaborate with the system at all. Free-riders launch their own tasks for execution but do not execute foreign ones. Collaborative peers are those that share some of their computational resources.

Figure 4.1 shows the peer state diagram. The labels in the arcs represent the changing probability between states. Thus, a collaborative peer can become a free-rider with a probability of ρ if the system allows it and a free-rider

can also become a collaborative peer with a σ probability if it is properly encouraged.

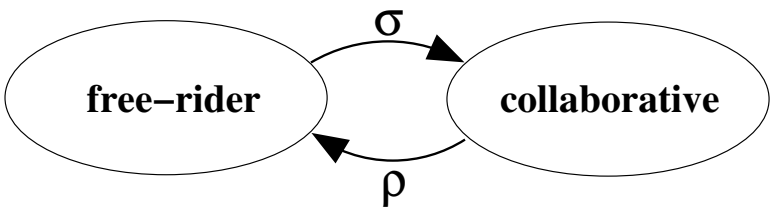


Figure 4.1: Peer state diagram.

One of the main challenge of an incentive mechanism is to encourage free-riders to become collaborative, but without forgetting the encouragement of collaborative peers to stay so as long as possible. Previous works modeled the amount of resources shared or session lengths, but state changing was not studied. For this reason, a previous work to the experimental study consisted in modeling the state changes, that is, defining new probability functions for both ρ and σ. These parameters were derived via careful sensitivity analysis until a representation was found that reflects observed user behavior (in [SR06, HWS07]) within the limitations of a simulation environment. In the following sections the methodology used to define these probabilities is described.

4.2.1 Collaborative to free-rider

The probability that a collaborative peer becomes a free-rider depends on the collaborative ratio (ColRatio) expressed in equation 4.1. This ratio gives the relationship between the work performed for the requesting peer and the foreign ones performed by the requesting peer. $LaunchedTasks$ is the number of tasks launched by the peer for execution and $RejectTasks$ the discarded ones. $ExecutedTasks$ are the tasks accepted and executed by the peer and submitted by other peers.

$$ColRatio = \frac{LaunchedTasks - RejectTasks}{ExecutedTasks} \tag{4.1}$$

The probability distribution ρ behaves exponentially with the form $\Upsilon^{ColRatio}$ ($\rho = \Upsilon^{ColRatio}$), where Υ is a constant in the range [0..1]. Fig-

ure 4.2 shows different ρ probability distributions, defined as $F_\rho(ColRatio) = 1 - \Upsilon^{ColRatio}$.

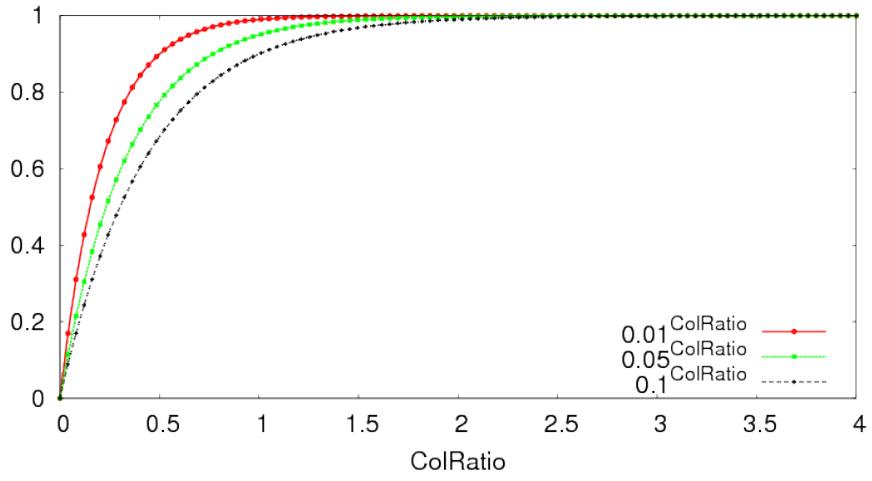


Figure 4.2: Probability distribution of stopping collaborating ($F_\rho(ColRatio) = 1 - \Upsilon^{ColRatio}$).

Summing up, the ColRatio indicates the profit that the peer is obtaining from the system. Thus, the higher the ColRatio, the lower the probability of ceasing to collaborate since it means that the peer is being compensated for its kind behavior by the system with a good QoS.

4.2.2 Free-rider to collaborative

In this case, the probability that a free-rider peer becomes collaborative depends on the free-rider ratio (FreeRatio) expressed in equation 4.2. This ratio is the percentage of rejected jobs per peer (i.e. peer penalization).

$$FreeRatio = \frac{RejectTasks}{LaunchedTasks} \tag{4.2}$$

The σ probability has a polynomial behavior with the form $FreeRatio^\phi$, where ϕ is a constant in the range $(0..\infty)$. Fig. 4.3 shows different σ probability distributions $F_\sigma(FreeRatio) = FreeRatio^\phi$.

To sum up, if the system detects that the peer is a free-rider and rejects

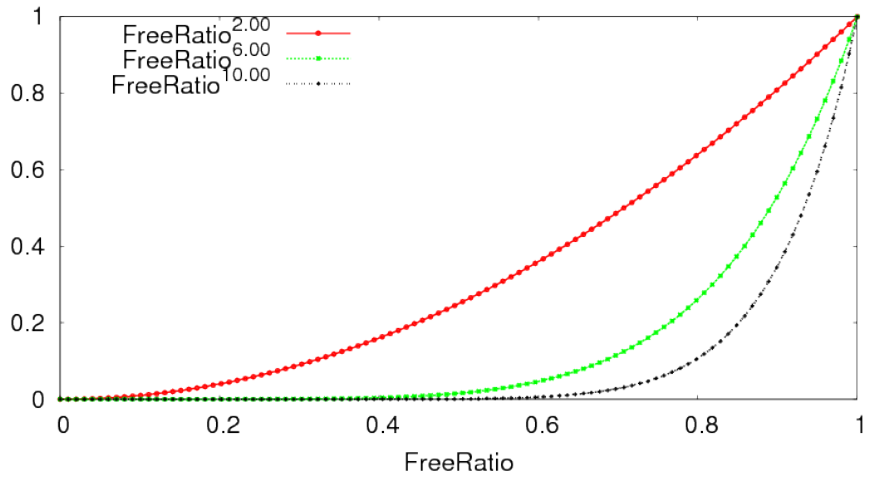


Figure 4.3: Probability distribution of starting to collaborate ($F_\sigma(FreeRatio) = FreeRatio^\phi$).

almost all its requests (in this case the FreeRatio will have a high value) the probability of the peer changing its behavior will increase. This is quite obvious since the main aim of a free-rider peer is to benefit from the network and, if that does not come about, it only has two options: start collaborating in order to increase the QoS it receives or leave the system for good.

4.3 Global Mechanism Evaluation

This Section is contained in the following paper:

J. Rius, F. Cores and F. Solsona, *"Cooperative Scheduling Mechanism for Large-Scale Peer-to-Peer Computing Systems"* Submitted to the Journal of Parallel and Distributed Computing. Special Issue: NovelArch for HPC, 2011.

4.4 Local Mechanism Evaluation

This Section is contained in the following paper:

J. Rius, F. Cores and F. Solsona, *"Incentive Mechanism for Scheduling Jobs in a Peer-to-Peer Computing System"*, Submitted to the Journal of Simulation Modelling Practice and Theory, 2011.

Chapter 5

Conclusions and Future Work

This chapter describes the conclusions reached in the development of this work, together with the main contributions and publications. It also explains the main open lines that should be developed in the near future.

5.1 Conclusions

Despite the continuous increase in computer power, the computational requirements of many applications have increased even more. In addition, improvements in the connectivity features of computer systems have motivated the growth of a set of applications operating under a new distributed computing paradigm. This paradigm is called Peer-to-Peer (P2P) computing and is the field of interest of this thesis. P2P computing architectures take advantage of the underutilization of computers, integrating them into a virtual network for resource sharing and allowing the use of the idle resources of thousands of computers connected via the Internet. For this reason, it can be claimed that P2P computing is an emerging low-cost alternative to the expensive supercomputer since it provides access in a very scalable way to millions of idle CPU cycles from all over the world.

Despite this, P2P computing systems are far from perfect. Due to their short life, they still need to be improved in many aspects. However, there has been an increase in the number of the publications in this field over recent years. Many researchers are trying to improve those networks in different ways.

As P2P systems involve the participation of the nodes of a network without central administration, the following particularities have to be considered: a rapid network reconfiguration, since nodes can connect and disconnect from it in a continuous, fast and arbitrary way; a dynamic allocation of services and resources; self-organization, where each node regulates its level of participation in the network; and a highly dynamic and heterogeneous environment.

Before developing DISIM, we were working on a project for a first prototype of a P2P computing system. This system is introduced in Section 2.4.5 under the name of CoDiP2P but it was named CompP2P in its origins. It can be said that such a P2P paradigm was the starting point for this and other theses. For this reason, we deem it appropriate to cite the following publications obtained with this work.

[BRG+07] I. Barri, J. Rius, I. Goiri, D. Castella, Fernando Cores, Francesc Solsona. *CompP2P: A P2P Computing System.* CEDI - XVIII Jornadas de Paralalelismo, 2007

[CRB+08] D. Castella , J. Rius, I. Barri, A. Guim, M. Orobitg, H. Blanco and F. Gine. *CoDiP2P: a Distributed Computing Architecture Oriented to P2P Environments.* XIX Jornadas de Paralelismo, 2008.

[CBR+08] D. Castella, I. Barri, J. Rius, F. Solsona, F. Gine and F. Guirado. *CoDiP2P: a Peer-to-Peer Architecture for Sharing Computing Resources.* International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2008), Advances in Soft Computing (Springer), volume 50, pages 293-303, 2008.

[CRB+09] D. Castella , J. Rius, I. Barri, F. Gine and F. Solsona. *CoDiP2P: a New P2P Architecture for Distributed Computing.* Conference on Parallel, Distributed and Network-based Processing (PDP 2009), pages 323-329, 2009.

One of the most critical aspects of P2P computing which needs to be extensively improved is the development of incentive policies to increase the participation of the users. In such systems, all the resources are provided voluntarily

by the users, so cooperation is a key point to success. Another important aspect of P2P computing systems is the mapping of tasks to the computational resources. The high dynamism and heterogeneity of such resources makes the scheduling process a really complex issue.

In this thesis, an incentive and scheduling mechanism is presented that can be used in many different kinds of shared computing networks provided they can be sub-grouped into areas controlled by a super-peer. We defined a two level overlay in order to guarantee the scalability of the system. Moreover, the decision to put both actions together (incentivizing and scheduling) in the same mechanism is based on the aim of minimizing the information to be sent inside the system in order to avoid its collapse. Most of the users' information may be necessary to perform both actions so, under this scenario, the fewer the messages exchanged between users, the better the performance of the proposed mechanisms will be.

Among all the possible options -game theory, reputation-based, token-based, etc.- we decided to implement a credit-based incentive scheme with a non-negative credit function (to prevent ID-changing cheating) and with a historic term used to differentiate newcomers from old collaborative peers. This decision was taken because a credit-based mechanism ensures the scalability of the system and allows the implementation of policies to avoid malicious behavior. The main objective of this mechanism is to motivate users to participate actively in the system, protect it against selfish and malicious behavior and discourage free-riding peers.

However, with a credit-based mechanism, some entities, over time, may become large credit hoarders. These entities can be peers rewarded for performing some functionalities in the system or simple peers sharing their resources without spending anything since they have no computational needs. One of our main contributions is to propose a reinvestment policy for credits. This policy, called Weighted, has been proven to increase peer cooperation enormously.

From the scheduling point of view, we propose a two level mechanism composed of the low-level scheduler, which operates in an area space and the high-level scheduler, which manages inter-area information in scheduling tasks. Area scheduling processes are performed by their managers by using the reverse

Vickrey Auction strategy, achieving a near-optimal task assignment at this level. In the inter-area level, managers exchange computational information with their neighbors. Based on that information, each manager is responsible for deciding how many tasks to allocate to its own area and how many will be better allocated to its neighboring ones.

Before evaluating our proposal, we modeled the behavior of two different users, free-riders and collaborative and we studied the probabilities of moving from one role to another. This behavior must be as real as possible in order to achieve applicability of our proposals. After that, this was implemented in a simulator in order to evaluate the system performance according to the user behavior. We showed that our proposed incentive mechanism is tolerant of user behavior changes and and reduces the number of free-riders significantly. This contribution leads to the following publications:

[RCS09b] J. Rius, F. Cores and F. Solsona. *A Reinvestment Mechanism for Incentive Collaborators and Discouraging Free Riding in Peer-to-Peer Computing.* International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 09), pages 905-906, 2009.

[RCS09a] J. Rius, F. Cores and F. Solsona. *A New Credit-Based Incentive Mechanism for P2P Scheduling with User Modeling.* The First International Conference on Advances in P2P Systems (AP2PS 2009), pages 85-91, 2009.

After modeling user behavior, the low-level mechanism was implemented in a simulator to evaluate the proposed algorithms. An extensive battery of interesting results comparing the proposed reinvestment policies was obtained. Simulation results show that our proposed incentive mechanism outperforms alternative approaches, maximizing system throughput and limiting free-riding, and is also tolerant of changes in user behavior. The results were compared with a basic credit-market mechanism with a uniform investing. The proposed Weighted incentive policy can improve the system performance by an average of up to 50%, depending on the system features (free-riders, total amount of credits in the system, etc.). All these results have proved to be statistically significant and a mathematical model was provided in order to

formalize our scheme. These results were presented with the following publication:

[RBCS10] J. Rius, I. Barri, F. Cores and F. Solsona. *A Formal Credit-Based Incentive Model for Sharing Computer Resources.* Euro-Par 2010 - Parallel Processing, 16th International Euro-Par Conference, Proceedings, Part I, pages 497-509, 2010.

At the high level, the performance of the proposed mechanism was evaluated by simulating a large-scale P2P computing system. The results showed that our proposed architecture can achieve appropriate scheduling irrespective of the number of peers, while it demonstrates a high scalability. Distributed scheduling with limited resource information can perform a near-optimal centralized scheduling. Furthermore, the mechanism was tested with very different kinds of parallel jobs and it was concluded that, thanks to the three criteria system for weighting the areas, it can always manage the shared resources efficiently. These results were submitted to the following journal and are being reviewed:

Submitted J. Rius, F. Cores and F. Solsona. *Cooperative Scheduling Mechanism for Large-Scale Peer-to-Peer Computing Systems.* Journal of Parallel and Distributed Computing. Special Issue: NovelArch for HPC, 2011.

To sum up, we proposed a new decentralized incentive and scheduling mechanism for distributing computation and resource management that takes the system heterogeneity into account by using the P2P paradigm. Our architecture implements a global credit-based incentive scheme for many P2P computing overlays. It allows the efficient scheduling of different kinds of multitasking distributed applications. The scheduler manages updated information from the system, and can thus take the resource mutability into account. Moreover, the mechanism is also designed to encourage users to be collaborative peers, i.e., sharing as many resources as possible, while protecting the system against malicious and selfish behavior. The overall work was recently submitted to the following journal:

Submitted J. Rius, F. Cores and F. Solsona. *Incentive Mechanism for Scheduling Jobs in a Peer-to-Peer Computing System.* Simulation Modelling Practice and Theory, 2011.

5.2 Future Work

At this point, we can say that this thesis has covered all the objectives. From a research perspective, based on the extensive experience gained during its development, this thesis has also opened a large number of challenges to be tackled.

1. **Provide the system with QoS.** It is well known that the Quality of Service (QoS) is a potential weakness of peer-to-peer computing networks. It is also true that the use of efficient techniques to encourage peer collaboration increases quality of service throughout the system. Otherwise, the main aim of the incentive mechanisms is to increase the participation of peers and discourage some bad behavior by the users, like selfish (free-riders) or malicious (cheating), rather than ensuring QoS. However, QoS is really appreciated by users in any kind of system. Going into the literature, one will realize than one of the most accepted methods for ensuring QoS in this kind of network is based on the replication of tasks or some basic information to increase the performance of the system. For this reason, we believe that providing the mechanism with a task replication method would be a challenging project.

2. **Churn rate.** This issue has not explicitly been analyzed since the real impact of the churn rate on the mechanism strongly depends on how the network prevents this. As pointed out in the manuscript, our mechanism is designed to work on many kinds of network provided they can be sub-grouped into areas controlled by a manager. So, the impact on the scheduling algorithm will vary depending on how the network responds to churn. However, some improvements could be made in the mechanism to minimize this impact. For example, introducing a new entity that keeps a copy of the information state of the managers could prevent the

system from losing important information if a manager is unexpectedly disconnected.

3. **Trustworthy of the results.** The trustworthiness of participating peers is one of the main weaknesses of P2P computing networks. In the work presented, we take the reputation of the users into account based on their previous contributions to the system. Despite this, no one can guarantee that a very well reputed peer starts to fake its results at any given time. Thus, any contribution in this field would be very welcome.

4. **Credit underwriting.** This issue can be considered as one of the most important weakness of the P2P networks in general. How credits are underwritten is a real problem in such systems. In other words, what is to stop a self-interested worker or master node from simply creating more credits and lying about how many it really has? In our proposal, we are thinking about a kind of authentication system were each user has an ID with an associated account. This system should prevent the users from increasing their credits while ensuring anonymity of the operating peer from other peers and any external users while keeping all its personal data safe. Despite this proposal, more work is still required in this field and another security mechanism should be developed.

5. **New reinvestment policies.** This thesis has proved the importance of good management of the credit system. An appropriate reinvestment policy can improve the system utilization significantly. This is because some entities may become huge credit hoarders, but their computational needs are really low, so they spend much less than they earn. For this reason, new reinvestment polices should be developed in order to increase the system performance even more.

6. **Nash equilibrium.** Another interesting work could be the study of the stability and robustness of the system. In order to achieve that formally, the Nash equilibrium strategy of the overall system should be found, i.e., the point where every user cannot increase its utility by choosing another

strategy, given the strategies of the others. Once found, it will provide the mechanism with a notorious credibility due to the high acceptance of the Nash equilibrium strategy.

7. **Mathematical model.** As stated in Section **??**, we really believe on the potential of the proposed mathematical model. However, in order to take the most of it, it is necessary to introduce user dynamism and to model user behavior. This would open a new and extensive range of possibilities for testing the proposed policies and methods formally.

Bibliography

[AAF+08] V. Aggarwal, O. Akonjang, A. Feldmann, R. Tashev, and S. Mohr. Reflecting p2p user behaviour models in a simulation environment. *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2008.

[ABCM04] Nazareno Andrade, Francisco Brasileiro, Walfredo Cirne, and Miranda Mowbray. Discouraging Free Riding in a Peer-to-Peer CPU-sharing Grid. In *13th IEEE Symposium on High Performance Distributed Computing (HPDC'04)*, pages 129–137, 2004.

[ABCM07] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray. Automatic grid assembly by promoting collaboration in peer-to-peer grids. *Journal of Parallel and Distributed Computing*, 2007.

[ACBR03] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro, and Paulo Roisenberg. Ourgrid: An approach to easily assemble grids with equitable resource sharing. 2003.

[ACM04] Panayotis Antoniadis, Costas Courcoubetis, and Robin Mason. Comparing economic incentives in peer-to-peer networks. *Comput. Netw.*, 46:133–146, September 2004.

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing.* Number UCB/EECS-2009-28. EECS Department, University of California, Berkeley, Feb 2009.

[AH00] E. Adar and B. A. Huberman. Free Riding on Gnutella. Technical report, Xerox PARC, August 2000.

[And04] David P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.

[And07] David P. Anderson. Volunteer computing: Planting the flag. In *Proceedings of PCGrid 2007, workshop held at the IPDPS conference*, 2007.

[are] Ares galaxy. http://sourceforge.net/projects/aresgalaxy/.

[BAGS02] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. Economic models for resource management and scheduling in grid computing. pages 1507–1542. Wiley Press, 2002.

[Bay05] M. Baye. *Managerial Economics & Business Strategy + Data Disk*. McGraw-Hill, 2005.

[BCC+06] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. *IEEE ICDCS*, 2006.

[BCF+08] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, L. Marchal, and Y. Robert. Centralized versus distributed schedulers for bag-of-tasks applications. *IEEE Trans. on Par. and Dist. Syst.*, 19:698–709, 2008.

[BGDG+10] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Q. Dang, and Kostas Pentikousis. Energy-Efficient Cloud Computing. *The Computer Journal*, 53(7):1045–1051, September 2010.

[bit] Bittorrent. http://www.bittorrent.com/.

[BRG+07] I. Barri, J. Rius, I. Goiri, D. Castella, F. Cores, and F. Solsona. Compp2p: A p2p computing system. 2007.

[CBA+06] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the world, unite!!! *Journal of Grid Computing*, 2006.

[CBR+08] D. Castella, I. Barri, J. Rius, F. Gine, F. Solsona, and F. Guirado. Codip2p: A peer-to-peer architecture for sharing computing resources. 50:293–303, 2008.

[CFA+04] Lauro Beltrão Costa, Loreno Feitosa, Eliane Araújo, Gustavo Mendes, Roberta Coelho, Walfredo Cirne, and Daniel Fireman. Mygrid: A complete solution for running bag-of-tasks applications. In *In Proc. of the SBRC 2004 – Salao de Ferramentas (22nd Brazilian Symposium on Computer Networks – III Special Tools Session*, 2004.

[Che97] Ramnath Chellappa. Intermediaries in cloud-computing: A new computing paradigm. *INFORMS*, 1997.

[CNA+04] Brent N. Chun, Chaki Ng, Jeannie Albrecht, David C. Parkes, and Amin Vahdat. Computational resource exchanges for distributed resource allocation. Technical report, 2004.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent. *In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[Cou06] R. Courcoubetis, C.; Weber. Incentives for large peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 24(5):1034 – 1050, 2006.

[CPC+03] W. Cirne, D. Paranhos, L. Costa, E. Santos-Neto, F. Brasileiro, J. Sauve, F. Silva, C. Osthoff, and C. Silveira. Running bag-of-tasks applications on computational grids: The mygrid approach.

Proceedings of ICCP 03 - International Conference on Parallel Processing (2003), 2003.

[CRB+08] D. Castella, J. Rius, I. Barri, A. Guim, M. Orobitg, H. Blanco, and F. Gine. Codip2p: a distributed computing architecture oriented to p2p environments. 2008.

[CRB+09] D. Castella, J. Rius, I. Barri, F. Gine, and F. Solsona. Codip2p: a new P2P architecture for distributed computing. 2009.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *International Workshop on Designing Privacy Enhancing Technologies: Desing Issues in Anonymity and Unobservability*, pages 46–66. Springer-Verlag New York, Inc., 2001.

[DCKM04] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, Oregon, August 2004.

[Dev02] IBM Developers. Charming python: Simpy simplifies complex models. In *Simulate Discrete Simultaneous Events for Fun and Profit*. 2002.

[dir] Dc++. http://dcplusplus.sourceforge.net/.

[dis] Distributed.net. http://www.distributed.net/.

[DKC05] Jean Dollimore, Tim Kindberg, and George Coulouris. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science Series)*. Addison Wesley, May 2005.

[EAH05] S. El-Ansary and S. Haridi. An overview of structured overlay networks. In *Theo. and Alg. Aspects of Sensor, Ad Hoc Wireless and P2P Net.* 2005.

[edo] edonkey2000. http://www.eDonkey2000.com/.

[emu] emule. http://www.emule-project.net/home/.

[EZ01] T. S. Eugene and H. Zhang. Predicting internet network distance with coordinates-based approaches. 2001.

[FCC+03] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: An architecture for secure resource peering. In *In Proceedings of the 19th ACM Symposium on Operating System Principles*, pages 133–148, 2003.

[Fei97] Dror G. Feitelson. Job scheduling in multiprogrammed parallel systems, 1997.

[FHKM04] F. Fessant, S. Handurukande, A. Kermarrec, and L. Massoulie. Clustering in p2p file sharing workloads. *IPTPS*, 2004.

[FI03] Ian Foster and Adriana Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pages 118–128, 2003.

[Fos05] Ian T. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In Hai Jin, Daniel A. Reed, and Wenbin Jiang, editors, *NPC*, volume 3779 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2005.

[gnu] Gnutella2. http://g2.trillinux.org/.

[GPM01] Mironov I. Golle P., Leyton-Brown K. and Lillibridge M. Incentives for sharing in peer-to-peer networks. *Lecture Notes in Computer Science*, 2232:75–87, 2001.

[Gro09] Groove Virtual Office Website. http://www.groove.net/, January 2009.

[GSS06] R. Gupta, V. Sekhri, and A.K. Somani. Compup2p: An architecture for internet computing using peer-to-peer networks. *IEEE Trans. Parallel Distrib. Syst.*, 2006.

[HKC09] Zage D. Hoffman K. and Nita-Rotaru C. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 41(4):31 pages, 2009.

[HWS07] K. Ho, J. Wu, and J. Sum. On the session lifetime distribution of gnutella. *International Journal of Parallel, Emergent and Distributed Systems*, 2007.

[ICG+06] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K.G. Yocum. Sharing networked resources with brokered leases. In *ATEC '06*, 2006.

[IEE91] *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries.* IEEE Computer Society Press, New York, NY, USA, January 1991.

[JMF09] Shantenu Jha, Andre Merzky, and Geoffrey Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurr. Comput. : Pract. Exper.*, 21:1087–1108, June 2009.

[KC04] S. H. Kwok and K. Y. Chan. An enhanced gnutella p2p protocol: A search perspective. *Advanced Information Networking and Applications, International Conference on*, 1:599, 2004.

[KF98] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, November 1998.

[KHvM97] Wolfgang Kreutzer, Jane Hopkins, and Marcel van Mierlo. Simjava—a framework for modeling queueing networks in java. pages 483–488, 1997.

[LLW09] Oksana Loginova, Haibin Lu, and X. Henry Wang. Incentive schemes in peer-to-peer networks. *The B.E. Journal of Theoretical Economics*, 9(1), 2009.

[LMM88] M. Litzkow, M.Livny, and M.W. Mutka. Condor - a hunter of idle workstations. *Eighth International Conference on Distributed Computing Systems (1998)*, 1988.

[LRA+05] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent Grid Systems*, 1(3):169–182, 2005.

[LS08] Zhengqiang Liang and Weisong Shi. Analysis of ratings on trust inference in open environments. *Perform. Eval.*, 65(2):99–128, 2008.

[LZZ+04] Virginia Lo, Daniel Zappala, Dayi Zhou, Yuhong Liu, and Shanyu Zhao. Cluster computing on the fly: P2p scheduling of idle cycles in the internet. *Proceedings of the IEEE Fourth International Conference on Peer-to-Peer Systems (2004)*, 2004.

[MCJI06] Feldman M., Papadimitriou C., Chuang J., and Stoica I. Free-riding and whitewashing in peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 24(5):1010 – 1019, 2006.

[MKL+03] Dejan Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. *HPL-2002-57 (2003)*, 2003.

[MKP08] Peter Merz, Florian Kolter, and Matthias Priebe. Free-riding prevention in super-peer desktop grids. In *ICCGI '08: Proceedings of the 2008 The Third International Multi-Conference on Computing in the Global Information Technology (iccgi 2008)*, pages 297–302, Washington, DC, USA, 2008. IEEE Computer Society.

[nap06] Napster. http://www.napster.com/, 2006.

[Nis99] Noam Nisan. Algorithms for selfish agents: Mechanism design for distributed computation. In *In Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–15. Springer, 1999.

[NJAB08] Nelson Nóbrega-Júnior, Leonardo Assis, and Francisco Brasileiro. Scheduling cpu-intensive grid applications using partial information. In *ICPP '08: Proceedings of the 2008 37th International Conference on Parallel Processing*, pages 262–269, Washington, DC, USA, 2008. IEEE Computer Society.

[Now78] D. Nowitz. Uucp implementation description. *UNIX Programmer's Manual, Bell Laboratories (1978)*, 1978.

[OKRK03] Philipp Obreiter, Birgitta König-Ries, and Michael Klein. Stimulating cooperative behavior of autonomous devices – an analysis of requirements and existing approaches. In *IN SECOND INTERNATIONAL WORKSHOP ON WIRELESS INFORMATION SYSTEMS (WIS2003*, pages 71–82, 2003.

[Osb04] Martin J. Osborne. *An introduction to game theory*. Oxford Univ. Press, New York, NY [u.a.], 2004.

[Pau02] Pragyansmita Paul. Seti @ home project and its website. *Crossroads*, 8:3–5, April 2002.

[pay] Paypal. https://www.paypal.com/.

[PWL09] V. Ponce, J. Wu, and X. Li. Improve peer cooperation using social networks. *Int. J. Parallel Emerg. Distrib. Syst.*, 24(3):189–204, 2009.

[RBCS10] Josep Rius, Ignasi Barri, Fernando Cores, and Francesc Solsona. A formal credit-based incentive model for sharing computer resources. In *Euro-Par 2010 - Parallel Processing, 16th International Euro-Par Conference, 2010, Proceedings, Part I*, pages 497–509, 2010.

[RCS09a] J. Rius, F. Cores, and F. Solsona. A new credit-based incentive mechanism for p2p scheduling with user modeling. pages 85–91, 2009.

[RCS09b] J. Rius, F. Cores, and F. Solsona. A reinvestment mechanism for incentive collaborators and discouraging free riding in peer-to-peer computing. pages 905–906, 2009.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, 31:161–172, August 2001.

[RY07] A. Rahbar and O. Yang. Powertrust: A robust and scalable reputation system for trusted p2p computing. *IEEE Trans. on Par. and Dist. Syst.*, 18(4):460–473, 2007.

[sky] Skype. http://www.skype.com/intl/es/home/.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31:149–160, August 2001.

[SPF03] Shashank Shetty, Pradeep Padala, and Michael P Frank. A survey of marketbased approaches to distributed computing (cise tr03013. Technical report, 2003.

[SR06] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *IMC 06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, New York, NY, USA, 2006. ACM.

[Tan81] A. S. Tanenbaum. Computer networks. *Prentice-Hall International (1981)*, 1981.

[TT07] Roberto Tamassia and Nikos Triandopoulos. Efficient content authentication in peer-to-peer networks. In *Proceedings of the 5th international conference on Applied Cryptography and Network Security*, ACNS '07, pages 354–372, Berlin, Heidelberg, 2007.

[vdSMR08] Turaga D.S. van der Schaar M. and Sood R. Stochastic optimization for content sharing in p2p systems. *IEEE Transactions on Multimedia*, 10(1):132–144, 2008.

[Vic61] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.

[WZK05] Adam Wierzbicki, Aneta Zwierko, and Zbigniew Kotulski. Authentication with controlled anonymity in p2p systems. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, PDCAT '05, pages 871–875, Washington, DC, USA, 2005.

[YGm03] Beverly Yang and Hector Garcia-molina. Ppay: Micropayments for peer-to-peer systems. pages 300–310. ACM Press, 2003.

[ZAM10] Kan Zhang, Nick Antonopoulos, and Zaigham Mahmood. A taxonomy of incentive mechanisms in peer-to-peer systems: Design requirements and classification. *International Journal on Advances in Networks and Services*, 2010.

[ZF] Niklas Zennstrom and Janus Friis. Kazaa. http://www.kazaa.com/.

[ZH08] Manaf Zghaibeh and Fotios C. Harmantzis. A lottery-based pricing scheme for peer-to-peer networks. *Telecommunication Systems*, 37(4):217–230, 2008.

[ZHC08] R. Zhou, K. Hwang, and M. Cai. Gossiptrust for fast reputation aggregation in peer-to-peer networks. *IEEE Trans. on Know. and Data Eng.*, 20:1282–1295, 2008.

[ZKJ01] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, EECS Department, University of California, Berkeley, Apr 2001.

[ZL08] H. Zhao and X. Li. H-trust: A robust and lightweight group reputation system for peer-to-peer desktop grid. In *The 28th Int. Conference on Distrib. Comp. Syst. Workshops*, pages 235–240, Washington, DC, USA, 2008. IEEE Computer Society.

[ZLH07] Y. Zhang, L. Lin, and J. Huai. Balancing trust and incentive in peer-to-peer collaborative system. *International Journal of Network Security*, 2007.

[ZSR06] S. Zhao, D. Stutzbach, and R. Rejaie. Characterizing files in the modern gnutella network. *MMCN*, 2006.

[ZZM07] Chen S. Zhang Z. and Yoon M. March: A distributed incentive scheme for peer-to-peer networks. In *Proceedings of IEEE INFOCOM'07*, 2007.