Journal of Cluster Computing manuscript No.
(will be inserted by the editor)

A Taxonomy of Peer-to-Peer Desktop Grid Paradigms

Han Zhao · Xinxin Liu · Xiaolin Li[⋆]

Received: date / Accepted: date

Abstract Desktop grid systems and applications have generated significant impacts on science and engineering. The emerging convergence of grid and peer-to-peer (P2P) computing technologies further opens new opportunities for enabling P2P Desktop Grid systems. This paper presents a taxonomy for classifying P2P desktop grid implementation paradigms, aiming to summarize the state-of-the-art technologies and explore the current and potential solution space. To have a comprehensive taxonomy for P2P desktop grid paradigms, we investigate both computational and data grid systems. Moreover, to ease the understanding, the taxonomy is applied to selected case studies of P2P desktop grid systems. The taxonomy is expected to be used as a survey of the state-of-the-art, a design map, a guideline for novice researchers, a common vocabulary, or a design space for simulation and benchmark, and to be extended as the technologies rapidly evolve.

Keywords P2P desktop grid · taxonomy · scientific computational grid · data grid · scheduling · resource management

1 Introduction

The last decade has witnessed the rapid evolution of grid computing, which delivers high performance and high throughput for large-scale applications in science, engineering, business, and military [38,37]. However, grid systems are confronted with significant challenges on increasing scalability and complexity issues. On the other hand, Peer-to-Peer (P2P) systems were conceived and designed for massive scales and users. To leverage the grid technologies with open standards and open architectures such as OGSA [39] and the elegant scalability and flexibility of P2P technologies, the convergence of grid and P2P technologies is imminent [36], particularly in the desktop grid environments. In this paper, we focus our investigation on P2P desktop grid systems.

⋆ Corresponding Author

Han Zhao and Xinxin Liu are with Department of Computer & Information Science & Engineering at University of Florida. Xiaolin Li is with the Department of Electrical & Computer Engineering at University of Florida.
E-mail: {han, xinxin}@cise.ufl.edu, andyli@ece.ufl.edu

Harnessing geographically distributed computing, communication, and storage resources, grid systems feature high performance, large scale, and relatively stable quality of service in well structured and trusted environments. The applications using grid services are typically large-scale and involve computation-intensive and data-intensive operations. Conventional grid systems administrate dedicated high-end workstations. For small lab researchers and common users, these resources are expensive and hard to access, thus largely confine the application of grid technologies. Recent trend of large-scale grid systems adopts cycle stealing or scavenging technologies, using low-end desktop machines for computationally intensive applications [52, 51]. Compared to traditional grid models, cycle scavenging achieves low cost and more flexibility, making it a good candidate for large-scale simulations and data analysis. Successful desktop grid projects such as Condor [57], BOINC [7], Entropia [26], and GridSAT [29] have emerged and gained immense popularity. However, they primarily operate in a relatively stable environment (e.g., university computer laboratory), where hosts are typically trustful and homogeneous. Moreover, these systems are organized in a centralized way that largely confines its expansion to a larger scale. As a result, while the centralized desktop grids achieve significant success in lab deployment, they are not fit to federate tremendous heterogeneous resources that are under-utilized in the Internet.

On the other hand, serving extremely large scale of concurrent users and large amount of data transfer, P2P computing is widely acknowledged as the most effective method for fluctuating resource sharing. In P2P computing resources are scattered at the edge of the Internet, usually have unstable connectivity and unpredictable IP addresses [36], and are coordinated without central intermediation. Therefore P2P computing potentially provide some complementary functionalities for managing grid systems, particularly for desktop grids. Grid systems are generally more secure and stable than P2P systems but need more improvement on handling heterogeneities, and dynamism in desktop computers. The idea of combining merits of desktop grids and P2P computing comes out naturally. However, merging P2P computing into desktop grids does not mean to simply construct one on top of the other. A variety of research issues, including but not limited to task scheduling, load balancing, data management and security, need to be thoroughly studied. This paper summarizes current research on constructing P2P based desktop grid systems. As this is a fast evolving field, this paper presents the snapshot of current efforts on integrating the two technologies, and aims at inspiring researchers for further development.

In general, introducing P2P techniques into the well-established desktop grid systems is attractive in the following aspects. First it allows for more sophisticated parallel model. Example project achieving this feature is Cohesion [69]. Typical desktop grids support only relatively simple parallel applications, whereas P2P expands the usage scenario to a greater extent. Second P2P introduces more heterogeneity and further extend user communities, for example, OurGrid [30]. Finally P2P adds more flexibility into desktop grids systems in terms of scheduling and task management, as we will present later in the taxonomy based on various prototypes and real system design schemes.

Related survey works have been conducted on general grid systems [12], P2P technologies [61], systems of resource management in grids [53], data grids [74] and desktop grids [28]. However, the convergence of P2P and desktop grids has not been comprehensively summarized in a systematic way yet. Our paper fills the gap, presenting a comprehensive taxonomy of P2P desktop grid implementation paradigms. Although P2P desktop grid is a nascent concept, it is rapidly growing and requires a comprehensive summary of the landscape to provide researchers with a better understanding of the solution space and a map to guide the future efforts. This taxonomy can also be used to formulate a common ground and vocabulary for comparison studies, design comprehensive simulators, benchmark suites, and foster community efforts to further extend the taxonomy to include evolving solutions.

In [53] grid systems are categorized into three themes depending on different design objectives: computational grid, data grid and service grid. Since the desktop grid systems are developed on top of the CPU cycle scavenging technology, most P2P desktop grid systems proposed so far fall into the category of computational grids, and systems self labeled as P2P data grids are systems providing data management functions for computational grids. As a result, we mainly focus on investigating P2P desktop computational grid systems (P2P-DCG for short), and confine our discussion on P2P desktop data grid (P2P-DDG) to systems storing and manipulating computational data for P2P-DCG. Due to the space limitation, a thorough survey of P2P based data sharing and service grid systems is beyond the scope of this paper, thus are excluded from the taxonomy. For a complete survey of the general data grid systems we refer the readers to Venugopal et al.'s work [74].

The distinguishing requirements of handling dynamism and heterogeneity in P2P desktop grids brew fundamentally different design approaches. In this paper we review the state-of-the-art technologies in this field and present a taxonomy for various solutions by summarizing published prototype and real system designs. Specifically our contributions are three folds: 1) we distill an abstract design model for general design of P2P desktop grids; 2) we categorize existing techniques in high level in order to convey a clear picture of the solution space to researchers and system designers; and 3) we map characteristics of representative systems into the taxonomy and provide concrete examples of usage scenarios.

The rest of the paper is organized as follows. Section 2 presents the taxonomy for general issues in P2P desktop grids (P2P-DCG) and Section 3 presents the taxonomy for P2P-based data management grid systems (P2P-DDG). Section 4 surveys representative P2P desktop grid systems using the proposed taxonomy. And finally, Section 5 concludes the paper.

2 Taxonomy of Peer-to-Peer Desktop Computational Grid

Grid technologies were inspired by the analogy to pervasive, open-standard based, economic power grids and were motivated to address the needs of scientific collaborations for large scale computations following open services and open architectures. The high demands of scientific computing attract world-wide efforts on the research of grid middleware for computationally-intensive applications, known as the computational grids. Computational grids are typically deployed in clusters or supercomputer centers. Recently, a large number of applications have emerged to collect computing power from idle CPU cycles of desktop machines. Successful examples such as SETI@home [8] deploy computational grids for the purpose of tackling computationally hard or even intractable scientific problems. However, due to the master/slave architecture used in these applications and the difficulties of dealing with highly dynamic and volatile resources, the solution provided by these examples is far from perfect. Applying P2P technologies to enable desktop grids is attractive in that P2P enables active participants rather than passive cycle-providers for users. P2P provides more efficient resource discovery and opens new opportunities for broader adoption of the desktop computational grid concept. The key challenge is how to integrate the two yet mature technologies into one efficient and effective system. This section presents an overview and the classification of the state-of-the-art research on this topic.

2.1 Design of P2P-DCG: Architecture and Role Model

In this section we outline existing research projects on P2P-DCG systems and develop a generalized architecture for the P2P desktop computational grids. The abstract architecture is shown in Fig. 1. The proposed architecture is highly summarized, presenting core functions and key design issues for P2P-DCG. From the figure we observe that the general system for P2P-DCG is built with four layers (bottom-up): (1) P2P Network Management, (2) Computing Resource Management, (3) Task Management, and (4) User Management. Each layer encapsulates core functions and exposes services to its upper layer. For example, at the bottom level P2P network management organizes geographically scattered desktop nodes into virtual P2P network topology, and presents an abstract computing resource pool for computing resource management. Meanwhile, each layer makes service request to its lower layer. Fig. 1 also displays a typical job execution cycle for P2P-DCG, shown in the dashed rectangle throughout the four layers. After a job is submitted, it is inserted into the waiting queue and is taken over by the task management layer. The top-down flow of job execution following downward arrows exhibits the process of service request handling until proper resources are found. Then each layer provides on time services to upper layer, ensuring the correct result is sent back to end user following upward arrows. Security management is addressed throughout all the four layers, and reliability management functions are implemented in the lower three levels for robust system design.

The P2P-DCG architecture generalized above not only summarizes system design aspects, but also identifies essential entities in a typical P2P-DCG system: users, tasks and resources (computing and networking). On the other hand, the P2P-DCG system is proposed to provide computational services with low-end resources. In that sense we can further identify different roles related to the computational service model: service consumer, service provider and service manager. Fig. 2 visualizes the corresponding relationship of entities abstracted from the P2P-DCG architecture and service model roles, and clarifies layer duties for design and implementation. As users of the desktop computational grid are connected via P2P network, they also contribute resources for other users, making them both service consumer and provider. In some systems, service providers are highly specialized computing nodes contributing resources, thus are differentiated from the common users. Service manager is in charge of managing entities of tasks and resources.

2.2 P2P-DCG Taxonomy

We summarize the taxonomy of P2P-DCG systems in this section. For each layer generalized in Fig. 1 we systematically classify key design aspects with mutually exclusive taxonomy. For building blocks across multiple layers such as security management, we identify commonly used technologies in current and ongoing projects at each layer, and taxonomize the technology if necessary.

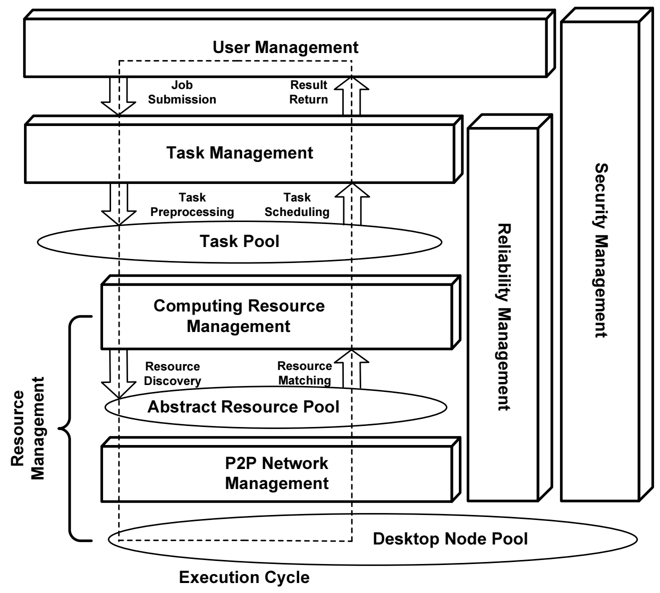

Fig. 1 P2P desktop computational grids general architecture

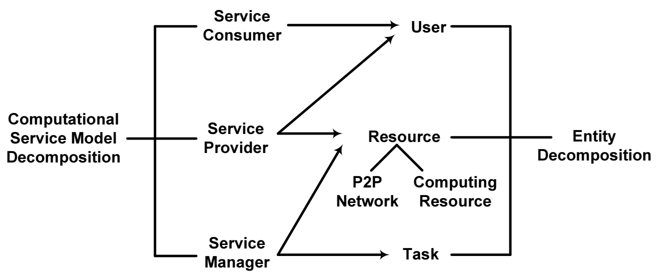

Fig. 2 Abstract model of service role and entity correspondence for P2P desktop computational grid

2.2.1 User Management

User management in P2P-DCG determines end user operational model. Fig. 3 shows the taxonomy for user management. Generally there are two ways to organize users in a P2P-DCG system. In a logical view, end users are abstracted from physical entities. A flat user organization does not impose any grouping restriction. On the other hand, some systems group logical users into multi-level communities. Job submission is done by multi-level processing through hierarchical schedulers on gateway peers. In a functional view, symmetric grouping indicates that the system does not have any role division. For example, in Cohesion [71] and OurGrid [30] nodes can simultaneously submit job and execute job submitted by others. In that sense each node is both resource provider and consumer, which conforms to the design principle of pure P2P. On the contrary some systems such as XtremWeb [19] rigorously differentiate users from

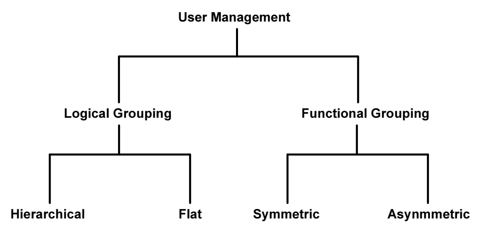

Fig. 3 Taxonomy for user management

worker nodes. Depending on different functional roles software modules are separately implemented and deployed in these systems.

2.2.2 Task Management

In P2P-DCG task management handles research issues related to user submitted jobs[1], including job characterization, interaction, decomposition and scheduling. The taxonomy is summarized in Fig. 4. Target applications for P2P-DCG can be unitary or composite [43]. Unitary applications, as its name indicates, use single parallel computational model such as Bag-of-Tasks or SPMD with MPI. Composite application consists of multiple unitary applications, in which inter-application dependency is taken into account for scheduling. Examples of composite applications include parameter sweeping and workflow applications. User job can be submitted to P2P-DCG either in batch, or in an interactive manner. Another issue concerning task characterization is the communication/computation ratio. Existing researches on P2P-DCG mainly focus on low-communication (e.g. parallel search [20]) or embarrassingly parallel (e.g. independent Bag-of-Tasks) task scheduling. One possible extension of low-communication is to exploit parallel asynchronous formulations for inter-task communications [56]. Another approach is to logically abstract programming environment into multiple virtual machines. For example, P2P-DVM [64] proposes a distributed VM middleware implementation to support MPI-based parallel model. Another class of systems such as Cohesion [71] and OurGrid [30] can flexibly adapt to various parallel models, thus we classify them as using adaptive inter-task communication. The detail description of these systems is presented in Section 4.

For task-level scheduling, we consider the problems of task distribution and workload balancing. Strictly speaking all the scheduling issues discussed in this taxonomy is cross-level. For example workload balancing requires resource status monitoring and task migration across the network substrate. Here we simplify the classification problem. Conventionally tasks are distributed directly to computing resources using HTTP or FTP protocol. A new way for data-intensive task distribution utilizes collaborative data transfer, which is widely used in P2P applications such as BitTorrent [55]. In BitTorrent data are truncated into small pieces and a user performing data downloading as well as contributing simultaneously. This method can be incorporated into P2P-DCG for highly efficient task distribution [15, 76, 44]. However it also introduces

[1] The word job and the word task are sometimes interchangeable, in some other situations task is of smaller granularity than job, e.g., one user submitted job can be divided into multiple identical tasks.

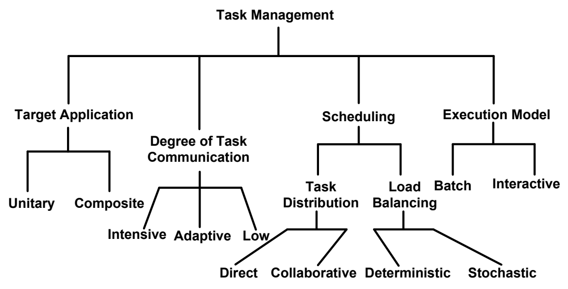

Fig. 4 Taxonomy for task management

security threats [32], which can be leveraged by using coding schemas with partial knowledge, which will be discussed later in Sect. 3.

Load balancing is also important in task-level scheduling. Deterministic load balancing methods, such as diffusive algorithms [31], require full or partial knowledge of the system. According to the knowledge each task is assigned to its host following certain predefined rules. In P2P-DCG deterministic load balancing can use either standard grid service [17] or underlying DHT-based overlay networks such as CAN [47]. Another approach is stochastic load balancing, where system knowledge is optional and task dispatching happens in a stochastic process. Example application for stochastic load balancing is Anthill [62], in which an agent-based load balancing algorithm called Messor [63] is proposed. Messor mimics ant behaviors and achieves uniformly distributed workload through swarm intelligence.

2.2.3 Resource Management

Resource management plays a critical role in P2P-DCG. Krauter et al. [53] presented taxonomy and conducted a comprehensive survey of resource management systems for general distributed environment. Here we focus on P2P-DCG system and summarize state-of-the-art research from different perspectives. Resource management in P2P-DCG covers both computing resource management and P2P network management, as indicated in Fig. 1. For computing resources we generalize four core functions: resource discovery, resource monitoring, resource identification and resource utilization. The taxonomy is presented in Fig. 5.

Resource Identification In P2P-DCG physical resources are presented as logically abstract units. Designers of the P2P-DCG system can use a global namespace to differentiate the virtual entities, where each resource is identified using a global unique identifier. Another option is to use hierarchical namespace analogous to DNS. In this method computing resources are typically clustered into different communities and resources are locally unique within each community.

Resource Monitoring Scheduling decision making heavily relies on resource monitoring. Based on the timing of resource status collection, resource monitoring can be classified into periodic reporting and on-demand reporting [53]. In periodic resource monitoring, scheduler regularly contacts individual node for state information (pull), or each node periodically reports its state (push) [17]. The drawback of periodic monitoring is that

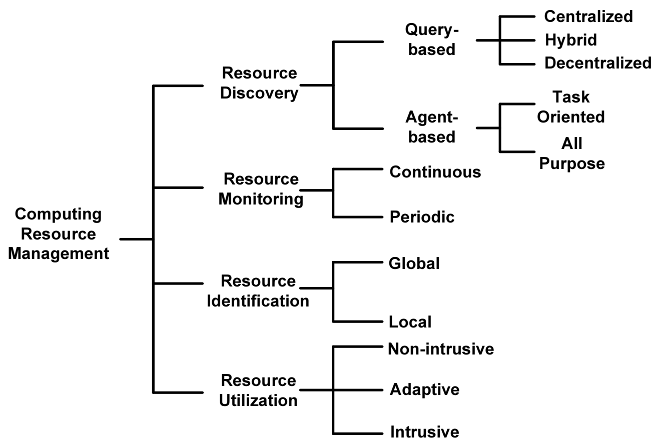

Fig. 5 Taxonomy for computing resource management

higher information accuracy aggravates communication overhead. One solution to this problem is to use predictions exploiting spatial and temporal correlations of resource information [23]. Another approach is to report state information upon requesting. For example CCOF [82] uses an on-demand pull scheduler, which collects state information from time-zone organized CAN network.

Resource Discovery In [53] resource discovery was categorized into query-based and agent-based. Query-based methods are most common in P2P-DCG schedulers [14]. In some systems the scheduler resorts to the central registry, or the regional catalog (e.g. GridSearch [50]) for resource discovery. A different approach is purely distributed, taking most advantage of flexibility of P2P network [24], but incurs higher scheduling complexity. Recently agent-based resource discovery becomes popular due to its scalability and simplicity. Unlike query-based methods, Example projects such as A-peer [73], Ontogrid [4], and Organic Grid [21], achieves resource discovery through dynamically dispatched codes performing simple tasks in agent-based methods. In some systems such as AgentTeamwork [40], agents are task-oriented, meaning that each agent carries on one specific task. In contrast some mobile agents are all-purpose; each agent works towards its goal with continuous state transitions.

Another research problem for resource discovery is matchmaking [48,49] that locates resources subject to certain constraints. The taxonomy of matchmaking in P2P-DCG is presented in Fig. 6. Recently approximate or rough matchmaking [11,10] gains favor of some researchers due to its ability to handle vagueness and uncertainty in resource queries.

Resource Utilization Resource utilization concerns how the management functions affect resource providers. In some systems such as XtremWeb [6] or GPU [59], resource management functions constantly run as background applications. Another kind of resource utilization is non-intrusive. Management functions are either activated at screen saver applications when resources are completely idling. Example projects using this method are OurGrid [30] and Organic Grid [21]. Some other systems such as Cohesion [71] achieves non-intrusive adaptively that computing cycle contribution is performed through utilization sensing.

Network Management Network management in a P2P-DCG system is in charge of administrating underlying P2P substrates and offering communication mechanisms. The taxonomy is summarized in Fig. 7. Most systems are proposed to operate on a particular form of P2P network organization, either with or without a logical structure. In unstructured P2P networks communications for resource query are conducted mainly using flooding [66]. An alternative approach logically organizes peers into DHT-based overlay network. A number of researches propose resource and task scheduling mechanisms on the structured network such as Chord [42], CAN [81,46] and Pastry [24], and address problems of unbalanced resource distribution due to hot zone and churn effect [33]. Lastly there are systems built on multiple overlays mixing unstructured and structured networks, denoted as hybrid. In contrast to systems built on particular P2P network, systems such as P2PGrid [18], aim at providing universal communication management functions without particular restriction on logic network organization.

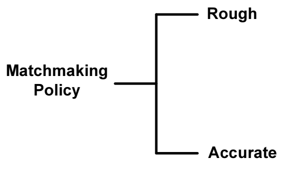

Fig. 6 Taxonomy for computing resource matchmaking

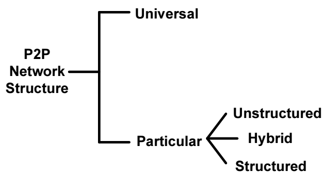

Fig. 7 Taxonomy for P2P network structure

2.2.4 Security and Reliability Management

P2P desktop grids extend traditional desktop grid environment to the global scale. However, this extension also introduces distrustful and highly volatile features. Hence, maintaining security and reliability is of paramount importance in P2P-DCG systems. Security management techniques are involved in all the vertical levels of the P2P-DCG architecture shown in Fig. 1, while reliability is mainly addressed at system design aspects apart from users. In this part we categorize security protection and reliability maintenance technologies applied to various levels in a typical P2P-DCG system. The taxonomy is summarized in Fig. 8 and Fig. 9 respectively.

Security management in user level mostly relies on user identity verification. Allowing user identification provides a screening process for certified peer management. For example, in P2P-DCG systems where users are also resource providers, incentive mechanisms described in Fig. 10 are more easily to be implemented for non-anonymous systems. At task level security management concerns two aspects: (a) task data privacy protection, commonly achieved through data encryption algorithms; and (b) cheating avoidance. The second problem arises from suspectable computational service of individual peer or collusion in order to gain advantage over others. To avoid such behaviors current P2P-DCG systems mainly use result verification for protection.

In order to avoid polluting local environment, sandboxing is extensively used for resource providers in P2P-DCG systems. Sandboxing creates an isolated environment

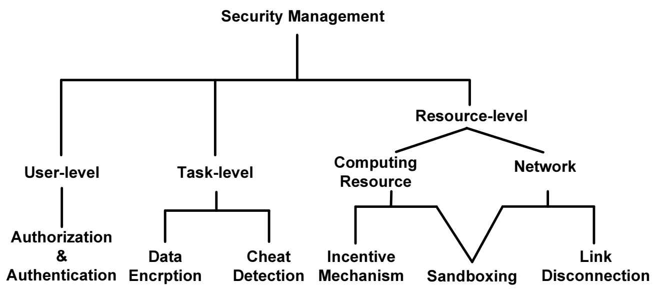

Fig. 8 Taxonomy for security management

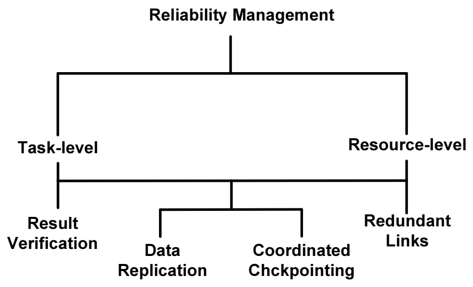

Fig. 9 Taxonomy for reliability management

for remote computational task execution. The encapsulation is commonly achieved using virtual machines. For example OurGrid [30] uses Xen technology to shield remote malicious attacks. In addition, link disconnection can be used if remote attacks from certain source are identified. Finally, P2P-DCG systems must provide effective incentive mechanisms to prevent free-riding and promote cooperation across distrustful peers. Typically two types of incentive mechanisms are used in P2P network management: market oriented and reciprocation [67]. Market oriented methods make use of economic models to regulate resource contributions. For example, Gupta et al. [42] proposed a lightweight P2P-DCG system called CompuP2P, which create a dynamic resource market for promoting collaborations across the resource providers. The other approach uses reciprocation that system rewards participants based on their contribution histories. Direct reciprocation applies to long-lasting applications as they provide enough reciprocation opportunities. Example project such as OurGrid [9] uses this method to restrain free-riding. On the other hand, indirect reciprocation methods use reputation or its variants [72]. The taxonomy for incentive mechanism is summarized in Fig. 10.

In addition to secure management, a successful P2P-DCG system should also provide reliable computational services. One chief technology to accomplish fault-tolerant computational services is data replication [27], which offers backup executions on multiple peers. Result verification is also useful for fault detection. It differs from result verification applied in secure management in detection motivations. Moreover, peers can build up coordinated checkpoints [64] for fault recovery and establish redundant links across the computational peers in case of network failures.

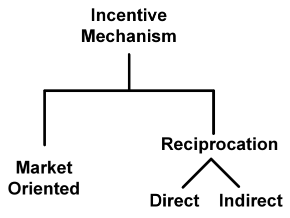

Fig. 10 Taxonomy for incentive mechanism design in P2P-DCG system

3 Taxonomy of Peer-to-Peer Desktop Data Grid

As we discussed in Sect. 1, the discussion of P2P-DDG systems in this section is limited to systems providing storage management functions for P2P-DCG. In fact, the development of these P2P-DDG systems are largely motivated by the needs of data-intensive applications, e.g., applications for bioinformatic and high-energy physics projects. The data size generated by these applications is on the order of Terabytes today and Petabytes in the near future. Hence to accommodate the large-volume data usage scenarios in P2P-DCG, a well-designed P2P-DDG system should be able to provide efficient data storage services as well as mechanisms to transfer, publish, replicate, share and analyze data. The major design barrier for conventional data grids lies in the high network latencies and expensive storage devices [54]. The integration of P2P resource sharing for high availability and low-cost desktop machines is a promising solution to break the barrier. In this section we survey and classify key features of storage management in P2P-DDG systems.

The P2P-DDG systems denoted in this paper differs from the P2P content sharing systems in that: (1) P2P-DDG systems handle immutable data (mostly read-only) and mainly support long-running applications; and (2) P2P-DDG systems provide an integrated computation and data distribution environment. Fig. 11 illustrates the taxonomy for storage management functions in P2P-DDG systems. Since P2P-DDG systems are highly relevant to the P2P-DCG systems, the overlapping features are already described in Sect. 2, thus are not discussed in details here. The taxonomy is classified into two categories: key features regarding data itself and operation related methods.

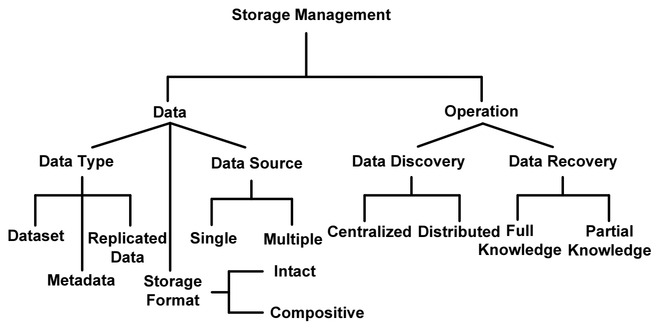

Fig. 11 Taxonomy for storage management in P2P desktop data grids

3.1 Taxonomy of Data Related Features in P2P-DDG

Data Type Two kinds of data are manipulated using storage management functions in P2P-DDG systems, namely *dataset* and *metadata*. Datasets are application generated data usually divided and stored as file instances in a P2P storage system, and metadata are information related to the storage system itself, including dataset description information, dataset contents, replica catalogs, etc. [25]. In addition, to improve data availability and accommodate fault-tolerance, a third data type, *replicated data*, is used in P2P-DDG systems. Depending on the data contents, data in P2P-DDG systems can also be categorized into public application data, personal data, service data, and etc.

Data Source The number of data sources in P2P-DDG systems largely depends on different application scenarios. For example in a P2P-DCG system solving computationally intensive problems in high energy physics, a particle detector might be involved for data generation. In that case the data can only be generated by a special instrument, making it a single source of data in this application. As an opposite example, many applications prompt desktop users to submit their own data for collaborative computation, forming multiple data sources. The number of data sources affects the design mechanisms for storage management in P2P-DDG systems, such as data consistency maintenance.

Storage Format Just like the taxonomy we made for task distribution in task management for P2P-DCG, data storage format in P2P-DDG can be either intact or composite. Intact data storage maintains each dataset or metadata unit in one piece. On the contrary, composite data storage takes advantage of BitTorrent-like technologies, trunking data into pieces for more delicate storage control, and merges the data for collaborative computation when necessary.

3.2 Taxonomy of Operation Related Functions in P2P-DDG

Data Discovery Similar to resource discovery in computing resource management for P2P-DCG, data discovery in P2P-DDG can be achieved through one central category, or several central categories maintained on super peers. Another way of data discovery uses P2P technologies to locate data distributively. For example, PeerStripe [60] stores data in a Pastry network, and retrieves data pieces using distributed hashed information (DHTs) of metadata.

Data Recovery Data recovery problem is a unique function in P2P-DDG systems if data is stored in composite format. Full knowledge data recovery denotes to data assembling methods that requires metadata for every data trunk. Conversely, another method only uses partial metadata information, therefore decreases the recovery costs and enhances security. For example PeerStripe uses data trunk coding for information recovery; the assembling of data pieces only need a fraction of the whole metadata knowledge.

P2P-DDG systems also share similar function designs with P2P-DCG. For example, incentive mechanisms for storage space contribution and security management. These topics are already discussed in details in Sect. 2, thus are not included in this section.

Table 1 Taxonomy Mapping of Desktop Computational Grid Projects Part 1

Project Name	Affiliation	User Management	Task Management	Computing Resource Management
XtremWeb	IN2P3 (CNRS) (FR) INRIA (FR) Paris-Sud 11 U (FR)	Flat Asymmetric	Unitary, Batch Low communication Direct dispatch Deterministic load balancing	Centralized query Global namespace Continuous monitoring Intrusive utilization
Cohesion	Tübingen U (GR)	Flat Symmetric	Unitary, Batch Adaptive communication Deterministic load balancing	Decentralized query Exact matchmaking Adaptive utilization
CCOF	U of Oregon (US)	Hierarchical Symmetric	Unitary, Direct dispatch Low communication	Hybrid query Adaptive utilization
Alchemi	U of Melbourne (AU)	Flat Asymmetric	Unitary, Direct dispatch Low communication	Decentralized query Adaptive utilization
OurGrid	Universidade Federal de Campina Grande (BR)	Hierarchical Symmetric	Unitary, Batch Low communication	Centralized query Non-intrusive utilization
GPU	N/A	Flat Symmetric	Unitary, Batch Low communication	Centralized query Intrusive utilization
Organic Grid	Ohio State U (US) Louisiana State U (US)	Flat Symmetric	Unitary, Direct dispatch Adaptive communication	Agent-based discovery Continuous monitoring Intrusive utilization

4 Case Study

This section studies several representative P2P desktop grid systems. Due to the space limitation we select projects that comprehensively cover previously developed taxonomy as much as possible. Table 1 and Table 2 combined taxonomize P2P desktop computational grid projects while table 3 identifies and classifies properties of the P2P desktop data grid projects.

Table 2 Taxonomy Mapping of Desktop Computational Grid Projects Part 2

Project Name	Network Management	Security	Reliability	Implementation
XtremWeb	Unstructured	Sandboxing, Data encryption Cheat detection	Replication Checkpointing	Java
Cohesion	Unstructured	Sandboxing Indirect Reciprocation	Replication	Java JXTA
CCOF	Structured	Sandboxing Incentive (Reputation) Quiz-based cheat detection	Replication	Simulation-based
Alchemi	Unstructured	User-level authorization User-level authentication	N/A	C# .NET framework
OurGrid	Unstructured	User-level authentication Direct reciprocation Cheat detection	Task-level verification	Java
GPU	Unstructured	User-level authentication Data encryption	N/A	Delphi
Organic Grid	Structured	N/A	Checkpointing	Java

Table 3 Taxonomy Mapping of Desktop Data Grid

Project Name	Organization	Data Type & Source	Storage	D Discovery	D Recovery	Implementation
PeerStripe	Virginia Tech (US)	All types Multiple	Composite	Distributed	Partial knowledge	Java
ADICS	Cardiff U (UK)	All types Multiple	Intact	Centralized	Full & Partial knowledge	Java
BitDew	INRIA (FR)	All types Multiple	Intact & Composite	Distributed	Full knowledge	Java

4.1 Selected P2P Desktop Computational Grid Projects

4.1.1 XtremWeb

XtremWeb [6, 19] is among the earliest research efforts to build the global-scale computing platforms for E-science applications. Strictly speaking XtremWeb does not fully implement a P2P desktop grid. Although XtremWeb claims to be designed towards P2P, its current architecture makes it closer to the conventional server/client model. In XtremWeb nodes are functionally classified into three categories: clients, workers and coordinators. However, Compared to other desktop grid systems such as SETI@home [8] or Condor [57], its implementation is inclined to P2P as computing nodes in XtremWeb are more autonomous. XtremWeb is capable of solving problems in a wide range of domains at the form of less correlated parallel tasks, e.g., MPI or BoT applications. XtremWeb relies on RPC (Remote Procedure Call) for communication management. Resource scheduling is achieved through resource management engine on coordination nodes. XtremWeb provides multiple levels of security protection. For example, for task management XtremWeb provides data privacy protection and result verification, and for resource and network management XtremWeb mainly uses sandboxing to shield the system from malicious attacks. Fault-tolerance is addressed in XtremWeb as well [34]. Replications and checkpointing are used in XtremWeb for fault-tolerant communications.

4.1.2 Cohesion

Cohesion [71, 69] is a modular P2P desktop grid system that highly adaptive to unstable and volatile distributed environments. In order to achieve fine grained resource control as well as mitigate scheduling complexity, Cohesion adopts a micro-kernel architectural design pattern, in which only minimum core functions are included, and user specific functions are pluggable to the system. Therefore Cohesion is lightweight, and can be highly customized for a variety of applications. Cohesion is quite flexible in handling various parallel models because of its highly adaptive feature for dynamic task decomposition. In addition, Cohesion allows for task migrations to utilize idling resources and balance workloads. Cohesion maintains a highly configurable unstructured P2P network using JXTA [3]. To better manage resources, Cohesion uses a tree structure for information aggregation [70]. Security is achieved mainly through sandboxing and trust management, the former isolates local codes from uncertified remote execution, while the latter protects user applications from hosts with bad reputation. Cohesion also uses local task replication to start over execution in case of task failure.

4.1.3 CCOF(Cluster Computing On the Fly)

CCOF [81] is a scalable community-based desktop grid project developed at University of Oregon. It offers a lightweight mechanism for community-based cycle sharing. In CCOF, hosts are loosely coupled and organized in a P2P overlay network. CCOF mainly aims at handling deadline-driven BoT style workload [81]. To schedule these embarrassingly parallel workloads CCOF employs two levels of schedulers. At local level, scheduler on each client is responsible for admission control and idle cycle tracking. For application-level meta scheduling a global wave scheduler is proposed [82] to harvest night time idle cycles. To accomplish this goal, CCOF uses a time-aware CAN-based DHT overlay to organize and to migrate hosts based on different time zones.

For security concerns CCOF uses sandboxing. In addition, CCOF addresses fault tolerance through workload replication. To tackle the insecure P2P environment and promote incentives, CCOF sets up a reputation system [80] and hands out quizzes to hosts for result verification. The quiz-based scheduling divides hosts into trusted and black list, thus effectively prevent individual and collude cheating behaviors.

4.1.4 Alchemi

Alchemi [58] is a .NET-based Grid framework developed as part of the Gridbus project in Australia. Alchemi aggregates idle desktop computers across the Internet for resource intensive applications. Since Alchemi is implemented using programming technologies such as C#, Web Services and .NET framework, it is suitable for clustering Windows desktop machines. Currently Alchemi only supports parallel communication-less tasks. Alchemi implements both object-oriented grid thread model and web service based grid job model for task abstraction. Tasks submitted by user nodes are dispatched to the manager node, and then they are scheduled to either dedicated or non-dedicated executor nodes in a simple FCFS manner. Alchemi uses a hierarchical architecture to support scalable application execution. Cross-platform interoperability is also supported through grid broker. Therefore Alchemi is capable of building a global-scale desktop grid not constrained to Windows platforms. For communication management Alchemi uses .NET remoting [65] for object transmission. Security issues are also concerned in Alchemi. Multiple levels of protection are provided by .NET defensive mechanisms, for example, suitable mechanisms are employed for user and contributor authorization. Alchemi does not provide any specific solution for failure management and fault recovery.

4.1.5 OurGrid

OurGrid [30] is an open source grid middleware that enables global research community collaboration. OurGrid accepts BoT applications from users and aggregates idle resources for execution speedup. OurGrid groups users into communities. Each user interacts with his or her OurGrid community through a client broker called MyGrid, which hides the heterogeneous grid structure and provides fast application turnaround for end user. Exploiting the features of BoT execution, scheduling in OurGrid is non-intrusive to local user, and designed to be explicit and adaptive for task assignment [68]. Since OurGrid is free to join, promoting resource donation and preventing free-riding become critical. OurGrid uses reciprocation methods called Network of Favors [9] to handle this problem. Compared to the market oriented method, the reciprocation schema is lightweight and more autonomous, but less flexible. Another critical issue addressed in OurGrid is security. OurGrid proposes an isolation technology based on Xen virtual machine, named SWAN (Sandboxing without A Name), for local data protection.

4.1.6 GPU

GPU (Global Processing Unit) [59] is a P2P desktop resource aggregation framework allowing users to build their own cluster for computationally intensive applications. The GPU framework is implemented in Delphi/Kylix. The target application for GPU is unitary, processed by an FD (File Distributor) and dispatched to task pools for execution using the GPU computational engine. GPU adopts methods from Gnutella [2] for resource management: resources are registered and taken care of through a coordinator. Computational requests routing in GPU is flooded. To improve scalability GPU also introduces ultrapeers (faster nodes) to organize a tree-structured network. However currently GPU cannot support execution in large magnitude, scaling only up to 60 nodes according to the project website. At run-time GPU loads a library of functions called plugins for interaction with the computational engine. For security concern GPU provides authentication for user applications as well as data privacy protection for plugins through PGP (Pretty Good Privacy).

4.1.7 Organic Grid

Organic Grid [21] is a bio-inspired desktop grid middleware that organizes heterogeneous geographically distributed nodes for data-intensive scientific applications. Unlike Xtremweb or GPU which relies on central coordination of scheduling, nodes in Organic Grid are functionally identical and autonomous. Therefore Organic Grid exploits the maximum degree of scalability. However, this approach also brings substantial difficulties for scheduling. Organic Grid solves this problem using mobile agents mimicking real world biological systems [22]. Data-intensive job is divided into sufficiently small chunks and are capsulated into mobile agents. Task and resource scheduling in Organic Grid is based on swarm intelligence approaches that each agent roaming around the network performs simple task without any global knowledge of the system. The network organization in Organic Grid is a structured tree overlay. Better-performing nodes are periodically checked and pushed upwards the tree for performance enhancement. Organic Grid does not provide any specific mechanism for malicious attack shielding. In case for encountering execution fault, the malfunctional node and its descendants will be disconnected from the tree, and unfinished tasks are resumed from recorded checkpoints.

4.2 Selected P2P Desktop Data Grid Projects

4.2.1 PeerStripe

PeerStripe [60] is a desktop grid system which transparently distributes data to storage spaces contributed by peers in a P2P network. PeerStripe mainly considers immutable data, and organizes the storage resource pool into a Pastry overlay network. Data generated by grid applications are injected into PeerStripe by certain data centers. In PeerStripe, data files are divided into varied-size chunks, and are encoded for fault tolerance. The metadata containing chunk information of each file is created and replicated in the system. To retrieve a file, PeerStripe will consult to the meta-data first, and pull each chunk according to the meta-data in Pastry. Peerstripe uses partial knowledge to retrieve data from chunks; after locating required chunks, PeerStripe decodes the chunks using erasure codes and assemble them for users.

4.2.2 ADICS

The peer-to-peer **A**rchitecture for **D**ata-**I**ntensive **C**ycle **S**haring (ADICS) [13,45] is a P2P-based storage architecture for grid applications. It is proposed by researchers at Cardiff University as part of the European CoreGRID project. ADICS is designed for handling mostly read-only scientific data. The data source of ADICS is multiple; a limited number of data providers produce input data. In ADICS peers are functionally asymmetric. Some nodes identified as data caching peers are in charge of storage management, and other nodes identified as worker peers take care of computational tasks. In ADICS, entire files are stored and replicated on data caching peers. Data recovery is directly achieved through original or replicated file. ADICS uses Peer-to-Peer Simplified (P2PS) middleware [75] to manage the underlying unstructured P2P infrastructure. Data discovery is achieved through centralized consultation to file and replication catalogues.

4.2.3 BitDew

Motivated by broadening the scope of parallel applications in P2P-DCG, the BitDew framework [35] implements a programming environment for P2P-based data management desktop computational grids. BitDew differentiate stable peers from volatile peers; the stable nodes are capable of running independent data management services and volatile nodes utilize or provide local storage resources. Dataset, replicated data and metadata coexist in BitDew for storage management and fault tolerance. For data discovery, distributed data catalogs are used and data look-up is achieved through DHT overlay protocols. BitDew is flexible in that it enables component plugin in accordance with user specification. For example, BitDew can specify either BitTorrent or FTP way for data transfer.

4.3 Status, Vision and Challenges

Despite its promising perspectives of delivering scalable computational and storage power to end applications and larger communities, most P2P desktop grids stagnate in their prototype design phase and lack successful stories of practical large-scale and stable deployments. Most systems achieve small to medium scale of applications used by hundreds of users worldwide. Take OurGrid as an example, as of April 29th, 2010, there are 248 machines on the grid, with 57 running at the same time [5]. While quite successful for computing power sharing within academic community, we expect P2P desktop grids expand greatly to accommodate applications of widespread public appeals.

We envision that P2P desktop grids have not fully unleashed its potential of utilizing globally distributed resources. Tremendous research efforts need to be put into incorporating well-tested solutions. Here we do not intend to make an exhaustive list of research challenges but just name a few to invite further discussions and proposals. First, most current P2P desktop systems as summarized in this taxonomy narrowly focus on embarrassingly parallel batch job executions, which is inadequate to support larger user communities who need to run more advanced parallel and distributed applications. The client side software can be made simpler, e.g. using a standard web browser with rich user interfaces. Second it is still problematic to provide QoS guaranteed services due to high churn rate of peers. Third to restrain free-riding and promote truthful peer behaviors. Market oriented approaches and other incentive mechanisms might be introduced to enable self-regulation, for instance, dispatching tasks and scheduling resources in the form of commodity market [16, 77] or auctions [41, 78, 79]. Last but not the least, as cloud computing becomes dominant for providing "pay-as-you-go" computing services, how to integrate key design choices and technologies of both grid and cloud platforms is nontrivial. Fortunately we have already seen such research efforts. For example, BitDew has been recently integrated into the Cloud@Home [1] project to provide data management services.

5 Conclusions

The convergence of desktop grid computing and P2P technologies has ushered in new opportunities for research and applications. In this paper we investigated the state-of-the-art research and implementations on this topic, identifying solution spaces and challenging issues in P2P desktop computational and data grid. A taxonomy was presented to highlight the key elements of system architecture and implementation paradigms. We have also investigated integration of agents in both computational and data P2P desktop grid. Applying the taxonomy, we further presented a survey on several representative systems. We expect the taxonomy to be further extended by researchers as the technologies rapidly evolve. This first taxonomy of P2P desktop grid systems can be used to guide many aspects of system research, design, and implementation, including, but not limited to: 1) designing gateway interfaces for grids to streamline the access to the underlying P2P services, 2) applying various existing overlay network structures in resource management and task scheduling, 3) implementing more efficient data management functionalities, and 4) studying more effective security mechanisms to protect systems and enhance security and privacy.

Acknowledgment

The authors would like to thank the editors and referees for their valuable suggestions, which have significantly helped improve the quality and presentation of this paper. The research presented in this paper is supported in part by US National Science Foundation (grants CNS-0709329, CCF-0953371, and OCI-0904938).

References

1. Cloud@Home. URL: `http://clouds.gforge.inria.fr/pmwiki.php`.

2. Gnutella. URL: `http://www.gnutelliums.com`.
3. JXTA (Juxtapose): an open source peer-to-peer protocol. URL: `https://jxta.dev.java.net/`.
4. Ontogrid. URL: `http://www.ontogrid.net`.
5. Status of OurGrid. URL: `http://status.ourgrid.org/`.
6. XtremWeb Project website. URL: `http://www.xtremweb.net/`.
7. D. P. Anderson. Boinc: a system for public-resource computing and storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.
8. D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.
9. 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*, 67(8):957–966, 2007.
10. I. Ataollahi and M. Analoui. Resource matchmaking algorithm using dynamic rough set in grid environment. *CoRR*, abs/0909.1397, 2009.
11. A. Azab and H. Kholidy. An adaptive decentralized scheduling mechanism for peer-to-peer desktop grids. In *ICCES '08: Proceedings of the International Conference on Computer Engineering and Systems*, pages 364–371, 2008.
12. M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Softw, Pract. Exper*, 32(15):1437–1466, 2002.
13. D. Barbalace, D. Talia, I. Kelley, I. Taylor, and C. Mastroianni. A data sharing protocol for desktop grid projects. Technical Report TR-0165, CoreGRID Report, 2008.
14. S. Basu, S. Banerjee, P. Sharma, and S.-J. Lee. Nodewiz: peer-to-peer resource discovery for grids. In *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid*, volume 1, pages 213–220, 2005.
15. C. Briquet, X. Dalem, S. Jodogne, and P.-A. de Marneffe. Scheduling data-intensive bags of tasks in p2p grids with bittorrent-enabled data distribution. In *UPGRADE '07: Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, pages 39–48, 2007.
16. R. Buyya, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling. In *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, pages 1507–1542. Wiley Press, 2002.
17. J. Cao, O. M. K. Kwong, X. Wang, and W. Cai. A peer-to-peer approach to task scheduling in computation grid. *International Journal of Grid and Utility Computing*, 1:13–21, 2005.
18. J. Cao and F. B. Liu. P2PGrid: Integrating P2P networks into the grid environment. In *GCC '05, 4th International Conference on Grid and Cooperative Computing*, volume 3795, pages 871–883, 2005.
19. F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. N?i, and O. Lodygensky. Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21(3):417 – 437, 2005.
20. D. Caromel, A. di Costanzo, and C. Mathieu. Peer-to-peer for computational grids: mixing clusters and desktop machines. *Parallel Computing Journal on Large Scale Grid*, 33(4-5): 275–288, 2007.
21. Chakravarti, Baumgartner, and Lauria. The organic grid: Self-organizing computation on a peer-to-peer network. *IEEE Transactions on Systems, Man, and Cybernetics*, 35, 2005.
22. A. J. Chakravarti, G. Baumgartner, and M. Lauria. Application-specific scheduling for the organic grid. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 146–155, 2004.
23. C. Chapman, M. Musolesi, W. Emmerich, and C. Mascolo. Predictive resource scheduling in computational grids. In *IPDPS'07: IEEE International Parallel and Distributed Processing Symposium.*, pages 1–10, 2007.
24. A. S. Cheema, M. Muhammad, and I. Gupta. Peer-to-peer discovery of computational resources for grid applications. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 179–185, 2005.
25. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200, 2001.
26. A. A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: architecture and performance of an enterprise desktop grid system. *J. Parallel Distrib. Comput*, 63(5):597–610, 2003.

27. S. Choi, M. Baik, J. Gil, C. Park, S. Jung, and C. Hwang. Group-based dynamic computational replication mechanism in peer-to-peer grid computing. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, page 7, 2006.
28. S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang. Characterizing and classifying desktop grid. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 743–748, 2007.
29. W. Chrabakh and R. Wolski. GridSAT: a system for solving satisfiability problems using a computational grid. *Parallel Computing*, 32(9):660–687, 2006.
30. W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the world, unite!!! *Journal of Grid Computing*, 4(3):225–246, September 2006.
31. A. Corradi, L. Leonardi, and F. Zambonelli. Diffusive load-balancing policies for dynamic applications. *IEEE Concurrency*, 7(1):22–31, 1999.
32. F. Costa, L. Silva, I. Kelley, and I. Taylor. Peer-to-peer techniques for data distribution in desktop grid computing platforms. In *Making Grids Work*, pages 377–391, 2008.
33. V. Darlagiannis, N. Liebau, O. Heckmann, A. Mauthe, and R. Steinmetz. Caching indices for efficient lookup in structured overlay networks. *Agents and Peer-to-Peer Computing*, pages 81–93, 2006.
34. S. Djilali, T. Herault, O. Lodygensky, T. Morlier, G. Fedak, and F. Cappello. Rpc-v: Toward fault-tolerant rpc for internet connected desktop grids with volatile nodes. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 39, 2004.
35. G. Fedak, H. He, and F. Cappello. Bitdew: a programmable environment for large-scale data management and distribution. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, 2008.
36. I. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *IPTPS '03: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, 2003.
37. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 2nd edition, 2004.
38. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.
39. I. T. Foster. Globus toolkit version 4: Software for service-oriented systems. *J. Comput. Sci. Technol*, 21(4):513–520, 2006.
40. M. Fukuda, K. Kashiwagi, and S. Kobayashi. Agentteamwork: Coordinating grid-computing jobs with mobile agents. *Applied Intelligence*, 25(2):181–198, 2006.
41. D. Grosu and A. Das. Auction-based resource allocation protocols in grids. In *Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems*, 2004.
42. R. Gupta, V. Sekhri, and A. K. Somani. CompuP2P: An architecture for internet computing using peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, PDS-17(11):1306–1320, 2006.
43. A. Iosup, , R. Iosup, D. H. J. Epema, J. Maassen, and R. V. Nieuwpoort. Synthetic grid workloads with ibis, koala, and grenchmark. In *Proceedigs of the CoreGRID Integrated Research in Grid Computing*, pages 271–283, 2005.
44. A. Kaplan, G. C. Fox, and G. von Laszewski. Gridtorrent framework: A high-performance data transfer and data sharing framework for scientific computing. In *SC'07: GCE07 Workshop*, 2007.
45. I. Kelley and I. Taylor. Bridging the data management gap between service and desktop grids. *Distributed and Parallel Systems*, pages 13–26, 2008.
46. J.-S. Kim, B. Bhattacharjee, P. J. Keleher, and A. Sussman. Matching jobs to resources in distributed desktop grid environments. Technical report, UMIACS-TR-2006-15, 2006.
47. J.-S. Kim, P. J. Keleher, M. A. Marsh, B. Bhattacharjee, and A. Sussman. Using content-addressable networks for load balancing in desktop grids. In *HPDC'07: Proceedings of the 16th International Symposium on High-Performance Distributed Computing*, pages 189–198, 2007.
48. J.-S. Kim, B. Nam, M. Marsh, P. Keleher, and B. Bhattacharjee. Creating a robust desktop grid using peer-to-peer services. In *IPDPS'07: Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pages 1–7, 2007.
49. J.-S. Kim, B. Nam, M. Marsh, P. Keleher, B. Bhattacharjee, and A. Sussman. Integrating categorical resource types into a p2p desktop grid system. In *GRID '08: Proceedings of*

the 2008 9th IEEE/ACM International Conference on Grid Computing, pages 284–291, 2008.
50. M. Koh, J. Song, L. Peng, and S. See. Service registry discovery using gridsearch p2p framework. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, page 11, 2006.
51. D. Kondo, G. Fedak, F. Cappello, A. A. Chien, and H. Casanova. Characterizing resource availability in enterprise desktop grids. *Future Generation Computer Systems*, 23(7):888–903, 2007.
52. D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien. Characterizing and evaluating desktop grids: an empirical study. In *IPDPS'04: Proceedings of the 18th International Parallel and Distributed Processing Symposium*, pages 26–36, 2004.
53. K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software–Practice and Experience*, 32 (2):135–164, 2002.
54. H. Lamehamedi, Z. Shentu, B. Szymanski, and E. Deelman. Simulation of dynamic data replication strategies in data grids. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 100.2, 2003.
55. A. Legout, G. Urvoy-Keller, and P. Michiardi. Understanding bittorrent: An experimental perspective. Technical report, HAL - CCSd - CNRS, 2005.
56. Z. Li and M. Parashar. A computational infrastructure for grid-based asynchronous parallel applications. In *HPDC '07: Proceedings of the 16th International Symposium on High-Performance Distributed Computing*, pages 229–230, 2007.
57. M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - A hunter of idle workstations. In *ICDCS '88: 8th International Conference on Distributed Computing Systems*, pages 104–111, 1988.
58. A. Luther, R. Buyya, R. Ranjan, and S. Venugopal. Alchemi: A .NET-based enterprise grid computing system. In *ICOMP'05: 6th International Conference on Internet Computing*, 2005.
59. T. Mengotti. GPU, a framework for distributed computing over gnutella, 2004. URL: `http://gpu.sourceforge.net/gpu_p2p/index.html`.
60. C. Miller, P. Butler, A. Shah, and A. R. Butt. Peerstripe: a p2p-based large-file storage for desktop grids. In *HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing*, pages 221–222, 2007.
61. D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical Report HPL-2002-57R1, Hewlett Packard Laboratories, 2003.
62. A. Montresor. Anthill: a framework for the design and analysis of peer-to-peer systems. In *Proceedings of the 4th European Research Seminar on Advances in Distributed Systems*, 2001.
63. A. Montresor, H. Meling, and O. Babaoglu. Messor: Load-balancing through a swarm of autonomous agents. In *Proceedings of 1st Workshop on Agent and Peer-to-Peer Systems*, pages 125–137, 2002.
64. L. Ni, A. Harwood, and P. J. Stuckey. Realizing the e-science desktop peer using a peer-to-peer distributed virtual machine middleware. In *MCG '06: Proceedings of the 4th international workshop on Middleware for grid computing*, page 3, 2006.
65. P. Obermeyer and J. Hawkins. Microsoft .NET remoting: A technical overview, 2001. URL: `http://msdn.microsoft.com/en-us/library/ms973857.aspx`.
66. T. V. Pham, L. M. Lau, and P. M. Dew. An adaptive approach to p2p resource discovery in distributed scientific research communities. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, page 12, 2006.
67. M. R. Rahman. A survey of incentive mechanisms in peer-to-peer systems. Technical report, CS-2009-22, University of Waterloo, 2009.
68. C. A. F. D. Rose, T. Ferreto, R. N. Calheiros, W. Cirne, L. B. Costa, and D. Fireman. Allocation strategies for utilization of space-shared resources in bag of tasks grids. *Future Generation Computer Systems*, 24(5):331–341, 2008.
69. S. Schulz and W. Blochinger. An integrated approach for managing peer-to-peer desktop grid systems. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 233–240, 2007.
70. S. Schulz, W. Blochinger, and H. Hannak. Capability-aware information aggregation in peer-to-peer grids – methods, architecture, and implementation. *Journal of Grid Computing*, 2008.

71. S. Schulz, W. Blochinger, M. Held, and C. Dangelmayr. Cohesion - a microkernel based desktop grid platform for irregular task-parallel applications. *Future Generation Computer Systems*, 24(5):354–370, 2008.

72. S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok. Trusted P2P transactions with fuzzy reputation aggregation. *IEEE Internet Computing*, 9(6):24–34, 2005.

73. L. Tie-Yan, Z. Zhi-Gang, and Y. Si-Zhen. A-peer: an agent platform integrating peer-to-peer network. In *CCGrid '03: Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 614–617, 2003.

74. S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, 38(1):3, 2006.

75. I. Wang. P2PS (Peer-to-Peer Simplified). In *Proceedings of 13th Annual Mardi Gras Conference - Frontiers of Grid Applications and Technologies*, pages 54–59. Louisiana State University, 2005.

76. B. Wei, G. Fedak, and F. Cappello. Collaborative data distribution with bittorrent for computational desktop grids. In *ISPDC '05: The 4th International Symposium on Parallel and Distributed Computing*, pages 250–257, 2005.

77. R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. G-commerce: Market formulations controlling resource allocation on the computational grid. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium (IPDPS)*, pages 46–53, 2001.

78. H. Zhao and X. Li. Efficient grid task-bundle allocation using bargaining based self-adaptive auction. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 09)*, volume 0, pages 4–11, 2009.

79. H. Zhao, X. Liu, and X. Li. Hypergraph-based task-bundle scheduling towards efficiency and fairness in heterogeneous distributed systems. In *the 24th International Parallel and Distributed Processing Symposium (IPDPS)*. Atlanta, USA, 2010.

80. S. Zhao, V. Lo, and C. GauthierDickey. Result verification and trust-based scheduling in peer-to-peer grids. *IEEE Fifth International Conference on Peer-to-Peer Computing*, 0: 31–38, 2005.

81. D. Zhou and V. Lo. Cluster computing on the fly: resource discovery in a cycle sharing peer-to-peer system. In *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, pages 66–73, 2004.

82. D. Zhou and V. Lo. Wavegrid: a scalable fast-turnaround heterogeneous peer-based desktop grid system. In *IPDPS '06: 20th International Parallel and Distributed Processing Symposium*, 2006.

Han Zhao is currently a PhD student in Department of Computer & Information Science & Engineering at University of Florida. He received the BE degree in Software Engineering from Jilin University, China. His research interests include parallel and distributed computing, autonomous systems and P2P systems. He is a student member of ACM and IEEE.

Xinxin Liu is currently a PhD student in Department of Computer & Information Science & Engineering at University of Florida. She received her BE degree in Software Engineering from Jilin University, China. Her research interests include wireless sensor networks and resource management.

Xiaolin Li is Associate Professor in Department of Electrical and Computer Engineering at University of Florida. His research interests include Parallel and Distributed Systems, Cyber-Physical Systems, and Network Security. His research has been sponsored by National Science Foundation (NSF) , Department of Homeland Security (DHS), and other funding agencies. He is an associate editor of several international journals and a program chair or co-chair for over 10 international conferences and workshops. He is on the executive committee of IEEE Technical Committee on Scalable Computing (TCSC) and served as a panelist for NSF. He received the PhD degree in Computer Engineering from Rutgers University, USA. He is directing the Scalable Software Systems Laboratory (`http://www.s3lab.ece.ufl.edu`). He received the National Science Foundation CAREER Award in 2010. He is a member of IEEE and ACM.