Social Clouds: Crowdsourcing Cloud Infrastructure

Kyle Chard and Simon Caton

Abstract Software crowdsourcing is becoming an increasingly viable model for creating production software addressing every aspect of the software development lifecycle. However, as software development processes become yet more complex requiring dedicated systems for development, testing, and deployment, software crowdsourcing projects must also acquire considerable infrastructure in order to facilitate development. We propose the use of an infrastructure crowdsourcing model, termed a Social Cloud, to facilitate a user-contributed cloud fabric on which software development services and systems can be hosted. Social Clouds are motivated by the needs of individuals or groups for specific resources or capabilities that can be made available by connected peers. Social Clouds leverage lessons learned through volunteer computing and crowdsourcing projects such as the willingness of individuals to make their resources available and offer their expertise altruistically for "good causes" or in exchange for other resources or payment. In this chapter we present the Social Cloud model and describe how it can be used to crowdsource software infrastructure.

1 Introduction

Software crowdsourcing [33, 44] is a new approach to software engineering in which individual tasks of the software development lifecycle such as coding, deployment, documentation and testing are outsourced to a potentially unknown group of individuals from the general public. Building upon the wide-spread adoption of crowdsourcing, which focuses on unskilled or semi-skilled participants collectively achieving a given goal, software crowdsourcing further extends this model by requiring sophisticated coordination of expert users who together–knowingly or

K. Chard (✉)
Computation Institute, University of Chicago and Argonne National Laboratory, Lemont, USA
e-mail: chard@uchicago.edu

S. Caton
Karlsruhe Service Research Institute, Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: simon.caton@kit.edu

W. Li et al. (eds.), *Crowdsourcing*, Progress in IS,
DOI 10.1007/978-3-662-47011-4_11

unknowingly–participate in an orchestrated workflow. This model moves away from traditional software development approaches towards a decentralized, peer-based model in which open calls for participation are used to accomplish tasks.

Software crowdsourcing shows great promise as a model for developing production software without requiring dedicated teams of developers. Software crowdsourcing projects cover a wide array of efforts ranging from contribution to AppStores (e.g. Apple AppStore or Google Play) through to individual fine grained contests in tools such as TopCoder.[1] In TopCoder users set challenges (e.g., programming, design or testing) along with a prize, deadline and requirements for completing the challenge. Other users can then discover and accept challenges. Upon completion a winning contribution is selected and the winner is rewarded. In this case, participants are motivated by monetary rewards; however in other crowdsourcing scenarios they may also be motivated by other factors such as altruism or standing in the community.

Like any software development model, software crowdsourcing requires significant computing infrastructure for all stages of the software development lifecycle. For instance, storage resources are required for code repositories, databases, test datasets, and documentation. Compute resources are required for testing, continuous integration, and hosting of services. These requirements are in stark contrast to the infrastructure requirements of other crowdsourcing models which typically require only minimal infrastructure to track tasks and results across many thousands of users. Existing crowdsourcing approaches will not scale to the sophisticated requirements of software crowdsourcing. In a crowdsourced environment questions arise over who should operate and maintain such infrastructure. Due to the collaborative nature of most software development processes it is infeasible to operate all infrastructure on individuals' machines, rather, collaborative infrastructure is required so that all users can access these systems.

Recent work has explored the use of cloud based approaches to support the requirements of software development activities. In fact, systems such as TopCoder offer the ability to provision a cloud virtual machine (VM) for a given challenge. Cloud based systems allow access by all participants and enable use of virtualized resources. This, for example, allows participants to create sophisticated infrastructure with the same ease at which they could do it locally. However, it does not solve the problem of how these cloud resources are procured. And due to the predominant pay-as-you go model of cloud usage it does not solve the problem of how these resources are paid for.

In this chapter we present the use of a Social Cloud [9] as a model for providing such shared infrastructure via an infrastructure crowdsourcing-like model. Originally developed as a resource sharing framework for sharing resources between connected individuals in a social network we suggest here that such approaches can also be applied to form a crowdsourced resource fabric for software development. The Social Cloud model is built upon the premise of virtualized resource contribution from semi-anonymous (crowd) and socially connected users. Virtualization techniques enable contributed resources to be used securely (from both the consumer's and provider's

[1] http://www.topcoder.com/.

perspectives) and to also provide an intuitive and sandboxed environment on which to construct services. We present an overview of a social storage cloud, social content delivery network (S-CDN) and social compute cloud to be used as the basis for supplying shared infrastructure for software crowdsourcing and describe the use of a currency-based, social network-based and matching-based model for allocating resources in such environments.

2 Social Clouds

A Social Cloud is "*a resource and service sharing framework utilizing relationships established between members of a social network.*" [9]. It is a dynamic environment through which cloud-like provisioning scenarios can be established based upon the implicit levels of trust represented between individuals in a social network. Where a social network describes relationships between individuals and social networks exist in, and can be extracted from, any number of multi-user system, including for example, crowdsourcing systems.

In the remainder of this section we present an overview of Social Clouds and describe the crowdsourcing calls that can be employed. We then present implementations of a Social Storage Cloud, Social Content Delivery Network (CDN), and Social Compute Cloud. This will enable us to describe a general Social Cloud model for crowdsourcing Cloud infrastructures (in Sect. 4.1). In each of these settings, we have also investigated different mechanisms for managing exchange using credit-based, social network-based and preference-based models, each of which can be viewed as a proxy for handling different types of crowdsourcing calls.

2.1 Motivation and Overview

The vision of a Social Cloud is motivated by the need of individuals or groups to access resources they are not in possession of, but that could be made available by connected peers. Later, we describe a Social Storage Cloud, a Social Content Delivery Network and a Social Compute Cloud; however additional resource types (such as software, capabilities, software licenses, etc.) could also be shared. In each case a Social Cloud provides a platform for sharing resources within a social network. Using this approach, users can download and install a middleware, leverage their personal social network, and provide resources to, or consume resources from, their connections.

The basis for using existing online social networks is that the explicit act of adding a "friend" or deriving an association between individuals implies that a user has some degree of knowledge of the individual being added. Such connectivity between individuals can be used to infer that a trust relationship exists between them. Similar relationships can be extracted between software crowdsourcing participants due to the reliance on coordinated and collaborative development practices. In both situations there may be varying degrees of trust between participants, for instance family members have more trust in one another than acquaintances do. Likewise, close collaborators in a software development process may have more trust in one another than in members they do not know. The social network model provides a way to encode this information as basic social relationships and to augment the social graph with additional social or collaborative constructs such as groups (e.g. friend or project lists), previous interactions, or social discourse.

Another way to think about a Social Cloud is to consider that social groups are analogous to dynamic Virtual Organizations (VOs) [15]. Groups, like VOs, have policies that define the intent of the group, the membership of the group and sharing policies for the group. Clearly, in this model Social Clouds are not mutually exclusive, that is, users may be simultaneously members of multiple Social Clouds. Whereas a VO is often associated with a particular application or activity, and is often disbanded once this activity completes, a group is longer lasting and may be used in the context of multiple applications or activities. While Social Clouds may be constructed based on Groups or VOs, we take the latter view, and use the formation of social groups to support multiple activities. In addition, different sharing policies or market metaphors can be defined depending on the group, for instance a user may be more likely to share resources openly with close collaborators without requiring a high degree of reciprocation, however the same might not be true for friends or more distant collaborators.

Resources in a Social Cloud may represent a physical or virtual entity (or capability) of limited availability. A resource could therefore encompass people, skills, information, computing capacity, or software licenses—hence, a resource provides a particular capability that is of use to other members of a group or community. Resources shared in a Social Cloud are by definition heterogeneous and potentially complementary, for example one user may share storage in exchange for access to compute. Or in the case of software crowdsourcing, a user may contribute compute resources to a particular group of users associated with a particular project while using storage resources associated with that same project.

In order to manage exchange, a Social Cloud requires mechanisms by which resource sharing can be controlled and regulated. Like crowdsourcing applications, in which contributors are rewarded for their contributions in different ways, a Social Cloud must also support different models for reward. Crowdsourcing applications often make use of monetary rewards for contribution (e.g., Amazon Mechanical Turk) or leverage reputation-based rewards and altruism (e.g., Wikipedia). Similarly, crowd workers have different motivations for participating (see: [43]), and we argue that many of these are also present in a Social Cloud context. In a Social Cloud we use the notion of a social marketplace as a model for facilitating and regulating exchange between individuals. A marketplace is an economic system that provides a clear mechanism for determining matches between requests and contributions. Importantly, a marketplace need not require monetary exchange, rather non-monetary protocols such as reciprocation, preference matching, and social graph based protocols can be used to determine appropriate allocations. If we compare possible non-monetary incentives for participating in a Social Cloud (see: [21]) to the motivation of participating in crowdsourcing platforms as defined in [43], there is a resounding overlap.

2.2 Crowdsourcing Calls

As Social Clouds are a form of *infrastructure crowdsourcing* they can be constructed with various calls depending on the purpose, and intent of the call. By leveraging [40]'s definition of call types, we refer to the following types of call for a Social Cloud:

- **Open Hierarchical** a call for resources to construct a personal Social Cloud. Here the call initiator may specify policies that define which resources are accepted for use. Where this could include specific relationship types, interaction histories or competencies etc. The important thing to note in this case, however, is that the call itself, is open; implying that anyone can offer to participate, but their participation may be subjected to user-specific policies.
- **Open Flat** a call for platform resources in the management of a Social Cloud. In this call, a Social Cloud platform asks for computational resources to facilitate its basic functionality. Resources can be contributed by any member of the community. We refer to this type of platform as a co-operative platform see: [22].
- **Closed Hierarchical** a call for resources from a specific social group. Here a call is only visible to a specific (sub)set of a user's friends and/or collaborators. Final selection, as with the open hierarchical call, may still be subject to user-specific policies.
- **Closed Flat** a call for platform resources from a specific (sub)community. Here a user or set of users define the social boundaries of a Social Cloud, for instance friends of friends, or a given social group or circle. Anyone within this community may provide resources for the Social Cloud platform in a similar manner to the Open Flat call.

From these call types, a given Social Cloud may enable contributions from a tight group of participants or more widely across a social network (e.g., friends of friends). Similarly, Social Clouds face the same difficulties as crowdsourcing applications with respect to quality, however, unlike typical crowdsourcing applications simple approaches such as task redundancy are not applicable. For this reason, we rely on interpersonal trust as a model for establishing reputation and predicting quality. Software crowdsourcing approaches, given their requirement for expert user contributions and complex tasks, may also leverage such approaches for establishing contribution quality.

2.3 *Social Storage Cloud*

In [9, 10] we present a Social Storage Cloud designed to enable users to share elastic storage resources. The general architecture of a Social Storage Cloud is shown in Fig. 1. We implement a Social Storage Cloud as a service-based Facebook application. Where a social network (Facebook) provides user and group management as well as the medium to interact with the Social Cloud infrastructure through an embedded web interface (Facebook application) which in turn exposes the storage service interfaces directly. To participate in a Social Storage Cloud users must deploy and host a simple storage service on their resources. Consumers can then interact directly with a specific storage service when allocated via a social/market protocol. The social marketplace is responsible for facilitating sharing and includes components for service registration and discovery, implementing and abstracting a chosen market protocol, managing and monitoring provisions, and regulating the economy. In this case we implement two economic markets: a posted price and a reverse auction. Both markets operate independently and are designed to work simultaneously.

Storage services are implemented as Web Services Resource Framework [12] (WSRF) services and provide an interface for users to access virtualized storage. Contributors must install this service on their local resources and register the service with the Social Storage Cloud application to participate in the market. This service exposes a set of file manipulation operations to users and maps their actions to operations on the local file system. Users create new storage instances by first requesting an allocation from a Social Storage Cloud and then passing the resulting service level agreement (SLA) (formed as the result of allocation) to a specific storage service, this creates a mapping between a user, agreement, and a storage instance. Instances are identified by a user and agreement allowing individual users to have multiple storage

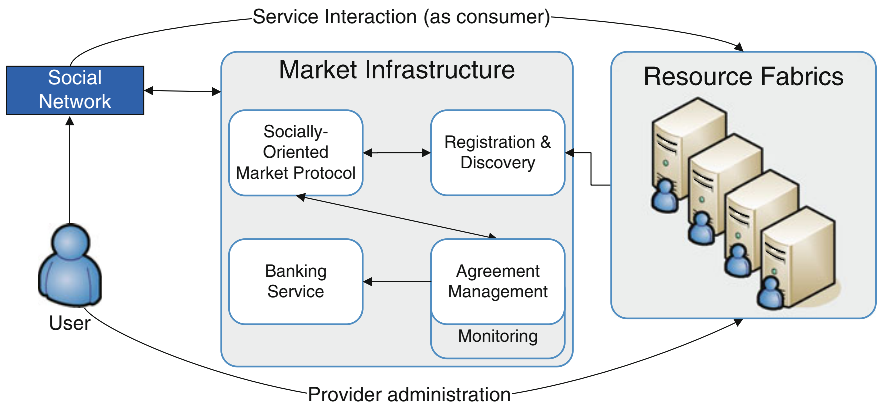
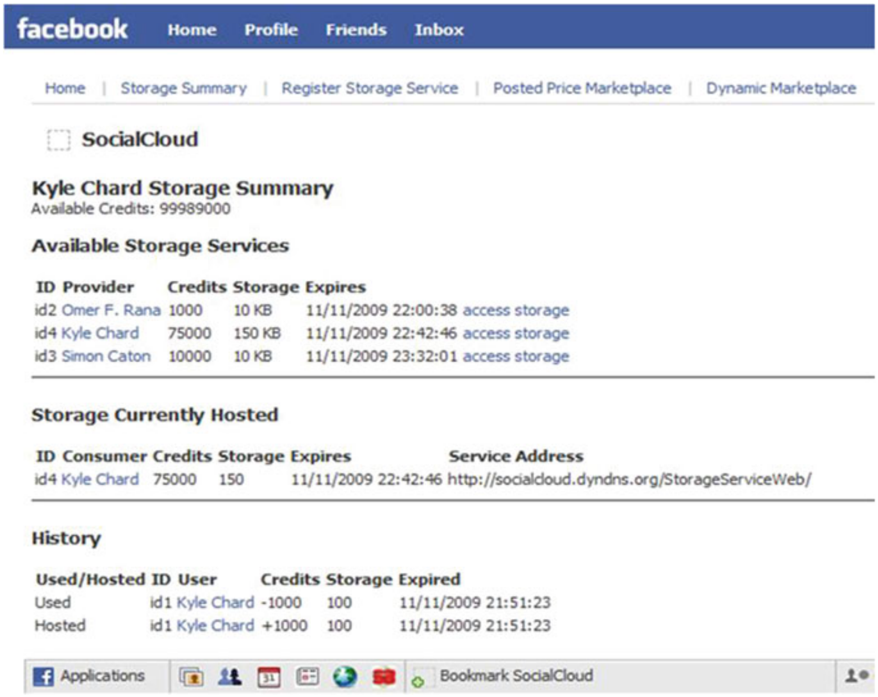


Fig. 1 Social Storage Cloud Architecture. Users register shared services, their friends are then able to provision and use these resources through a Facebook application. Allocation is conducted by an underlying market infrastructure

instances in the same storage service. The storage service creates a representative WSRF resource and an associated working directory for each instance to sandbox storage access. The resource keeps track of service levels as outlined in the agreement such as the data storage limit. Additionally the service has interfaces to list storage contents, retrieve the amount of storage used/available, upload, download, preview and delete files. These interfaces are all made available via the integrated Facebook application.

The two market mechanisms (posted price, and reverse auction) operate similarly, allowing consumers to select and pay for storage resources hosted by their friends. In a posted price market users select storage from a list of friends' service offers. In the reverse auction (tender) market, consumers outline specific storage requirements and pass this description to the Social Cloud infrastructure; providers then bid to host the storage. Both mechanisms result in the establishment of an SLA between users. The SLA is redeemed through the appropriate storage service to create a storage instance. An example summary user interface is shown in Fig. 2. This summary view shows user allocations both on others' resources as well as others' allocations on contributed resources.


facebook
Home
Profile
Friends
Inbox
Home | Storage Summary | Register Storage Service | Posted Price Marketplace | Dynamic Marketplace
SocialCloud
Applications
Bookmark SocialCloud

Kyle Chard Storage Summary
Available Credits: 99989000

Available Storage Services

ID	Provider	Credits	Storage	Expires	
id2	Omer F. Rana	1000	10 KB	11/11/2009 22:00:38	access storage
id4	Kyle Chard	75000	150 KB	11/11/2009 22:42:46	access storage
id3	Simon Caton	10000	10 KB	11/11/2009 23:32:01	access storage

Storage Currently Hosted

ID	Consumer	Credits	Storage	Expires	Service Address
id4	Kyle Chard	75000	150	11/11/2009 22:42:46	http://socialcloud.dyndns.org/StorageServiceWeb/

History

Used/Hosted	ID	User	Credits	Storage	Expired
Used	id1	Kyle Chard	-1000	100	11/11/2009 21:51:23
Hosted	id1	Kyle Chard	+1000	100	11/11/2009 21:51:23

Fig. 2 Social Storage Cloud user summary interface. This interface displays available storage services in a user's network. It also lists other users consuming storage resources as well as historical reservations

The Social Storage Cloud is the simplest model with respect to issuing a call, and it uses the most common approach for regulating exchange–a monetary model. Calls in this case are closed. Only established social relationships (or friend in Facebook terminology) are permitted. The call is also hierarchical in that requesting users set the rules for the exchange through the use of an institutionalized (economic) mechanism, which also performs the allocation process. In [9, 10], we explore the use of a credit-based system that rewards users for contributing resources and charges users for consuming resources. The use of a credit model requires the implementation of services to manage the exchange of credits securely. We use a Banking service to manage users' credit balances and all agreements a user is participating (or has participated) in. Credits are exchanged between users when an agreement is made, prior to the service being used. The two concurrent economic markets, posted price and reverse auctions, are designed to model different forms of exchange: a posted price index, and a reverse Vickrey auction. Following an allocation, communication between consumer and provider is through the establishment of an SLA (represented using WS-Agreement [3]).

In a **posted price** market a user can advertise a particular service in their Social Cloud describing the capabilities of the service and defining the price for using this service. Other users can then select an advertised service and define specific requirements (storage amount, duration, availability, and penalties) of the provision. This approach is analogous to the process followed by crowdsourcing applications such as Amazon Mechanical Turk where requesters post tasks and advertise a stated price for accomplishing the task. The Social Storage Cloud uses a simple index service to store offers and provides an interface for other users to then discover and select these offers. When a user selects a service offer they also specify their required service levels, a SLA is created defining the requirements of the provision such as duration and storage amount. Before using the service, the generated SLA must be passed to the appropriate storage service to create an instance. The storage service determines if it will accept the agreement based on local policy and current resource capacity. Having instantiated storage the agreement is passed to the Banking service to exchange credits. A copy of the agreement is stored as a form of receipt.

In a **reverse auction** (tender) market, a requesting user can specify their storage requirements and then submit an auction request to the Social Storage Cloud. The user's friends can then bid to provide the requested storage. We rely on auction mechanisms provided by the DRIVE meta-scheduler [8]. In particular, we use a reverse Vickrey auction protocol as it has the dominant bidding strategy of truth telling, i.e., a user's best bidding strategy is to truthfully bid in accordance to their (private) preferences, making the Vickrey auction more socially centric. It also means that "anti-social" behavior such as counter speculation is fruitless. In a reverse auction providers compete (bid) for the right to host a specific task. The DRIVE auctioneer uses the list of friends to locate a group of suitable storage services based on user specified requirements; these are termed the bidders in the auction. Each bidder then computes a bid based on the requirements expressed by the consumer. The storage services include a DRIVE-enabled interface that is able to compute simple bids based on the amount of storage requested. The auctioneer determines the auction winner and creates an SLA between the consumer and the winning bidder. As in the posted price mechanism, the agreement is sent to the specified service for instantiation and the bank for credit transfer. In this model the consumer is charged the price of the second lowest bid, as the Vickrey auction is a second-price mechanism.

When considering this market approach for crowdsourcing there are clear advantages and disadvantages: The use of a virtual currency provides a tangible framework for users to visualize their contribution and consumption rates in a way they can relate to: if they run out of credits, they cannot consume resources; users can manage their credits independently with respect to their personal supply and demand; any mechanism or means of executing a call can be designed to facilitate exchange and achieve specific design intentions; and, the design of the mechanisms have obvious parallels to classic crowdsourcing call structures. Despite these advantages, however, there is an obvious challenge: the need to manage and maintain economic stability, i.e., inflation/deflation over time and the dynamics of context: users, like workers can come and go which can aggravate inflation/deflation. These challenges cannot be understated.

2.4 Social Content Delivery Network

In [11, 32] we present the notion of a Social Content Delivery Network (CDN) which builds upon the idea of a Social Storage Cloud to deliver scalable and efficient distribution of shared data over user contributed resources. The Social CDN is designed to enable data-based scientific exchanges via relationships expressed in social or community networks. Figure 3 illustrates the Social CDN model. Here a user has produced a data artifact for a collaborative project (e.g. the results of a scientific experiment, a new software component, bundle or library, or new data set for analysis), which the user wishes to share with their collaborators. In the Social Storage Cloud setting presented previously this would constitute a backup action being performed by the user using the storage resources of a their social peers, similar to that of a Dropbox storage action. In a Social CDN, however, the backup action is secondary to the action of sharing and distributing data amongst collaborators.

The Social CDN model builds upon a network of *Data Followers*, analogous to to Twitter followers, users that follow data status updates and data posts of a given user. Note that users do not automatically reciprocally follow their data followers. Therefore, like circles in Google+, a follower-followee connection is not considered bilateral. However, the relationship between these users must be bilaterally authorized for our assumptions on pre-existent trust to hold. In order to share data, a user appends an artifact (via a social network application) to a status message or Tweet. This action tells the Social CDN which dataset should be shared, in which data follower circle, and invokes the Social CDN's data transport and management algorithms to execute the sharing action. Likewise, when users access data status messages they may also be published to enable social interactions around data usage.

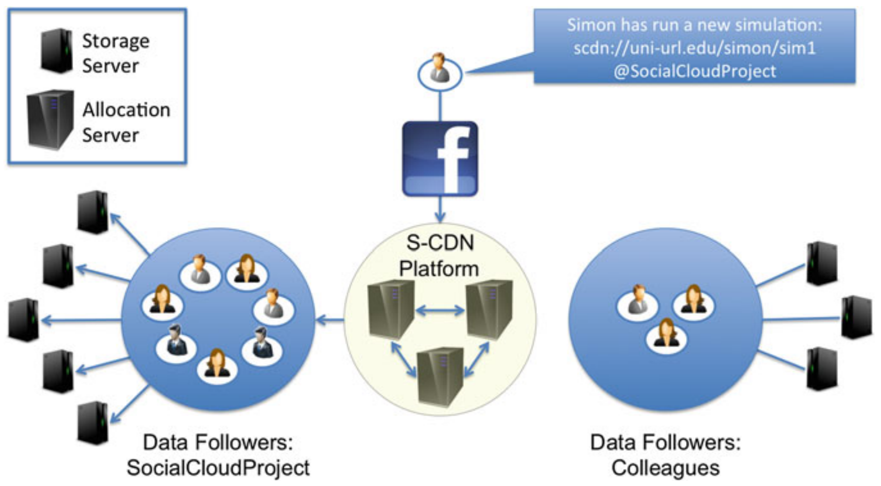

Storage Server
Allocation Server
Simon has run a new simulation: scdn://uni-url.edu/simon/sim1 @SocialCloudProject
S-CDN Platform
Data Followers: SocialCloudProject
Data Followers: Colleagues

Fig. 3 Social CDN: usage and overview. Data followers of a user receive data updates when the user shares a dataset with the Social CDN. Allocation servers determine the most efficient placement and transfer of the dataset

As in the the Social Storage Cloud setting, users contribute storage resources to the Social CDN, i.e., Social Cloud platform. This enables the Social CDN to use storage resources located on the edge devices owned (and therefore contributed) by members of data follower circles, including the user sharing the data. These servers are used to share and access data, and also as temporary hosts for others' data within the network. Users who provide resources to the Social CDN can also have the data encapsulated by a data post pushed to their resources in the form of replicas by the Social CDN architecture.

By building on the notion of a Social Storage Cloud, storage resources contributed to the Social CDN are used by the central Social CDN application to distribute data across replica nodes. In this sense, the system constructs a distributed data warehouse across resources networked in accordance to the social network of their owners. However, rather than using proprietary storage services the Social CDN builds upon Globus [14], a provider of high performance data transfer and sharing. Globus uses'endpoints' installed on users' resources to enable transfer and sharing. The Social CDN application uses the Globus APIs to alter sharing properties on an endpoint to enable replicas to be stored and accessed. It then uses these same APIs to transfer data between a replica and the requester's endpoint. The Social CDN web application is implemented as a Django application, it uses a social network adapter to connect to Facebook for authentication and access to the user's social graph, and a local database to store and manage users and allocations.

In this setting, the main crowdsourcing-like call is in the construction of the Social CDN infrastructure—an open flat call in that anyone can take part and provide resources to the Social CDN. How these resources are then used, is another matter, as we can assume that data is not shared universally. Therefore, there are a myriad of ways in which to move data around the Social CDN. Unlike the Social Storage Cloud, the incentives to take part in and contribute to a Social CDN are more closely tied with personal and collective benefit. For this reason, we have not implemented a market-like setting as we expect the Social CDN to be autonomically [28] self-managed over time based on how interactions and data shares emerge over time.

Instead, we leverage the social basis of a Social Cloud to use network structure and network analysis algorithms to select appropriate replica locations on which to distribute data. This approach attempts to place replicas of each dataset on selected available endpoints of the dataset's owner's friends (followers), where the replication factor (the number of nodes selected for replication) is a configurable system parameter. The approach attempts to predict usage based on relationships between users, for example by placing replicas on friends with the strongest connection to the data contributor.

In the Social CDN we use social network analysis algorithms to rank potential replica locations, for example determining important, well connected individuals in the network. We use graph theory metrics such as centrality, clustering coefficient, and node betweenness to determine nodes that are important within a network. In each case the network analysis algorithms identify a ranked list of nodes (social network members) that are used to place replicas. To address availability constraints we construct a graph that has edges between nodes if the availability of two nodes overlaps, and a "distance" weighting is assigned to each edge that describes the transfer characteristics of the connection. When selecting replicas, we choose a subset of nodes that cover the entire graph with the lowest-cost edges. In [11] we show that using such approaches can improve the accessibility of datasets by preemptively allocating replicas to nodes that are located "near" to potential requesters.

In a crowdsourcing context the use of social network based allocation strategies is appropriate as it presupposes quality driven through matching individuals that know one another, or have worked together in the past. For example, such approaches could be used to select users (and their resources) that are central in a network and therefore have strong bonds with other members of the network (or project). The use of a social network-based allocation approach upon the premises and observation of previous as well as existing collaborative actions suits the establishment of Social Clouds in scenarios like software crowdsourcing. Such contexts are inherently network based, and even if workers do not know each other in the physical world, they may have digital (working) relationships in (semi)anonymous networks that can be seen as a forms of pre-existent trust. Thus fulfilling the basic premises of a Social Cloud.

In the Social CDN, we do not adopt an SLA-based approach. The reason is quite simple: replicas and their resources are assumed to be transient and, naturally, also replicated. Similarly, there is no way of determining for how long a replica should be available, or how long the follower-followee relationship will be required or maintained. Furthermore, the Social CDN is responsible for the placement of replicas and their migration (as appropriate) around the network, therefore it would also be cumbersome to create and sign SLAs in this setting. This means a Social

CDN using the allocation method presented is constructed on a best-effort basis, and inherently reliant on the collaborative impetus of its users.

2.5 Social Compute Cloud

In [5] we present a Social Compute Cloud designed to enable access to elastic compute capabilities provided through a cloud fabric constructed over resources contributed by users. This model allows users to contribute virtualized compute resources via a lightweight agent running on their resources. Consumers can then lease these virtualized resources and write applications using a restricted programming language.

The general architecture of the Social Compute Cloud is shown in Fig. 4. The Social Compute Cloud is built upon Seattle [4], an open source Peer-to-Peer (P2P) computing platform. The Social Compute Cloud extends Seattle to use Facebook APIs to authenticate users and to associate and access a user's social graph. It uses Seattle's virtualized resource fabric to enable consumers to offer their resources to the cloud by hosting sandboxed lightweight virtual machines on which consumers can execute applications, potentially in parallel, on their computing resources. Consumers can access these virtualized resources (via the secure virtual machine interfaces) to execute arbitrary programs written in a Python-based language. While the concept of a Social Compute Cloud could be applied to any type of virtualization environment we use lightweight programming (application level) virtualization as this considerably reduces overhead and the burden on providers; in [42] we explore the use of a more heavyweight virtualization environment based on Xen, however the time to create and contextualize VMs was shown to be considerable.

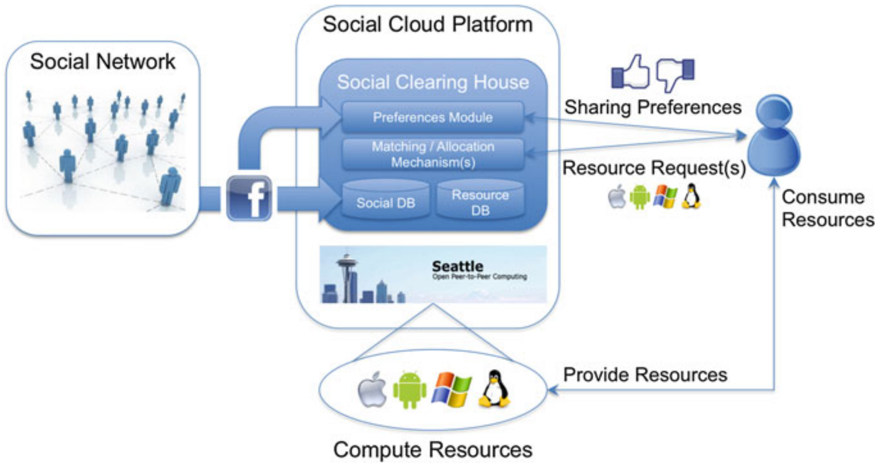
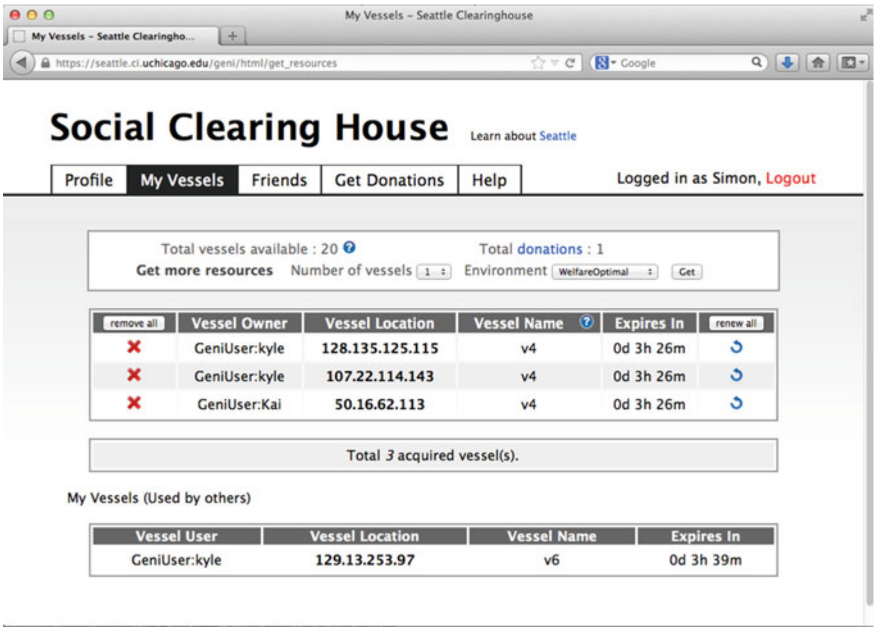


Fig. 4 Social Compute Cloud Architecture

We chose to base the Social Compute Cloud on Seattle due to its lightweight virtualization middleware and its extensible clearing house model which we extend to enable social allocation. Building upon Seattle we leverage the same base implementation for account creation and registration, resource contribution infrastructure, and resource acquisition mechanisms. We have extended and deployed a new social clearing house that leverages social information derived from users' Facebook profiles and relationships along with a range of different preference matching allocation protocols. We have implemented a service that enables users to define sharing preferences (e.g., a ranked order of other users) and new interfaces in Seattle that allow users to view and manage these preferences.

Figure 5 shows a user interface which presents the resources being used by a particular user as well as the users that are using this particular user's resources. This interface, extended from Seattle, provides a model to renew and cancel existing reservations.

The Social Compute Cloud poses one main type of call: a two-way closed hierarchical call for resources. Here, users specify preferences with whom they are willing to consume resources from and provide resources to. These preferences, as will be discussed in more detail later, are not binary, i.e., either provide or not provide, but also rank users against one another. In a similar manner to the previous two Social Cloud settings, an open/closed flat call for platform resources, could also be envisaged. Unlike the Social Storage Cloud, we move away from a credit model, but


My Vessels – Seattle Clearinghouse

My Vessels – Seattle Clearingho...

https://seattle.ci.uchicago.edu/geni/html/get_resources

Social Clearing House Learn about Seattle

Profile | My Vessels | Friends | Get Donations | Help

Logged in as Simon, Logout

Total vessels available : 20 Total donations : 1

Get more resources Number of vessels 1 Environment WelfareOptimal Get

remove all	Vessel Owner	Vessel Location	Vessel Name	Expires In	renew all
✗	GeniUser:kyle	128.135.125.115	v4	0d 3h 26m	↻
✗	GeniUser:kyle	107.22.114.143	v4	0d 3h 26m	↻
✗	GeniUser:Kai	50.16.62.113	v4	0d 3h 26m	↻

Total *3* acquired vessel(s).

My Vessels (Used by others)

Vessel User	Vessel Location	Vessel Name	Expires In
GeniUser:kyle	129.13.253.97	v6	0d 3h 39m

Fig. 5 Social Compute Cloud user summary interface showing current allocations for an individual user

retain user involvement in call management. Similarly, to the Social CDN, we place a heavier reliance on social network constructs, but instead of using social network analysis methods to determine "good" allocations, we involve user choice in the form of sharing preferences, and allocate (or match) based upon these preferences.

Preference matching algorithms allow consumers and producers to specify their preferences for interacting with one another. This type of matching is successfully applied in a variety of cases, including the admission of students to colleges, and prospective pupils to schools. In the Social Compute Cloud we capture the supply and demand of individual users through a social clearing house. As we use a centralized implementation we can derive the complete supply and demand in the market and therefore match preferences between all participants in a given Social Cloud. This of course is only after users define their preferences towards other users, that is, their willingness to share resources with, and consume resources from, other users in the Social Cloud.

To determine the "best" matches between users, given the stated requirements, we use several different matching algorithms. The chosen algorithms differ with respect to their ability to satisfy different market objectives and performance. Commonly used market objectives include finding solutions to the matching problem which are stable (i.e., no matched user has an incentive to deviate from the solution) or optimizing the total welfare of the users, the fairness between the two sides of the market, or the computation time to find a solution. The choice of particular market objectives in turn affects which allocation and matching strategies can be considered. This can range from direct negotiation to a centralized instance that computes this matching; and both monetary and non-monetary mechanisms can be applied. Our approach considers non-monetary allocation mechanisms based on user preferences. Depending on the specific market objective, several algorithms exist that compute a solution to the matching problem, e.g. computing a particularly fair solution or one with a high user welfare.

Two-sided preference-based matching is much studied in the economic literature, and as such algorithms in this domain can be applied in many other settings. For this reason we developed a standalone matching service that enables clients to request matches using existing protocols. This service is used by the Social Compute Cloud (via the social clearing house) to compute matches for given scenarios. The matching service includes three algorithms from the literature, and a fourth of our own implementation. These algorithms have been selected to address limitations when preferences are incomplete or specified with indifference, that is when not all pairwise combinations have preferences associated or when users associate the same preference to many other users. Briefly, the matching algorithms offered in the Social Compute Cloud are:

- **Deferred-Acceptance (DA)** [16]: is the best known algorithm for two-sided matching and has the advantages of having a short runtime and at the same time always yielding a stable solution. However, it cannot provide guarantees about welfare, and yields a particularly unfair solution where one side gets the best stable solution and the other side gets the worst stable solution.

- **Welfare-Optimal (WO)** [26]: is a common matching algorithm that yields stable solutions with the best welfare for certain preference structures. The approach uses structures of the set of stable solutions and applies graph-based algorithms to select the best matches.
- **Shift** [24]: is designed to find stable solutions with consideration for welfare and fairness when indifference or incomplete lists are present. While DA and WO can still be used in such settings, they can no longer guarantee to find the globally best solution. In such settings, Shift can find a stable match, with the maximum number of matched pairs for certain special cases. However, these scenarios are in general hard to approximate, and consequently the standard algorithms are not able to provide non-trivial quality bounds with respect to their objectives. Finding the optimal solution for the matching problem with respect to the most common metrics: welfare or fairness, is NP-hard [24].
- **Genetic Algorithm with Threshold Accepting (GATA)** [23]: is a heuristic-based algorithm that yields superior solutions compared to the other algorithms. The GA starts with randomly created (but stable) solutions and uses the standard mutation and crossover operators to increase the quality of the solutions. GATA then uses this solution as input for the TA algorithm, an effective local search heuristic that applies and accepts small changes within a certain threshold of the current solution performance.

Each of these algorithms have their specific performance merits and we have studied their performance with respect to a Social Cloud in [5, 23]. The key findings, however, suggest that the GATA or similar GATA-like approaches perform well in larger problem sizes, with more complex preference structures and with stochastic supply and demand, rather than a batch allocation mode in which two-sided matching algorithms are often applied. This means for the purposes of facilitating crowdsourcing calls, that users can be more involved in terms of how the call clears.

Like the Social CDN approach, there is currently no handling of SLAs in the Social Compute Cloud. Rather, each compute node is reserved for a predetermined period of time, if during this time a node goes offline the resource consumer will be notified and as a form of "enforcement" may consequently update their sharing preferences. Given the social context of a Social Cloud, it is foreseeable that any issue of this sort be first discussed in order to find either a resolution or cause for the error. This social process is important so as to not damage the real world relationship underpinning the exchange. It is also similar to the feedback and discourse methods often used in crowdsourcing platforms (like Amazon's Mechanical Turk) when employers are not satisfied with the quality of a worker's results.

3 Quality Management, Trust, and Agreements

Having established the means to architect a Social Cloud, and allocate resources in various settings, the question of (collaboration) quality arises with respect to the actual infrastructure provided. This, for the moment, is even irrespective of how this infrastructure is used. Instead, we refer here to quality of infrastructure instances that are sourced from the community.

The crowdsourcing literature proposes many different means of assessing and defining (worker) quality, reliability and trustworthiness with respect to (task) solutions. Where examples include: redundant scheduling, gold standard questions, qualification tests, peer review and employer acceptance rate. Whilst not all of these approaches are relevant or meaningful in crowdsourcing computational infrastructures, it is still important to maintain a notion of quality that supports collaborative processes. In terms of a computational infrastructure, there are several aspects that can be used to denote and measure quality both quantitatively (e.g. availability, error rate, mean error recovery time, etc.) and qualitatively (non-functional parameters like owner's technical competence, trustworthiness, responsiveness to crowd sourcing calls, etc.).

Without delving into quality properties, the crowdsourcing literature does, in general, differentiate between different methods of assessing quality. Where up-front task design and post-hoc result analysis are the two main methods of controlling work quality [30]. Up-front task design typically involves methods of pre-selection: a means of ensuring a minimum ex-ante quality level of contributions [17] so that an employer mitigates the risk of poor quality solutions through pre-selection tests or processes. Typically, these are in the form of qualification tests, or thresholds for worker attributes. Post-hoc result analysis, however, allows a worker to perform a task before validating the result in some way. Here typical examples are the gold standard (micro)task[2] [37], the redundant scheduling of tasks to multiple workers to provide a basis for solution comparison and worker reliability [29], TopCoder-like competitions, and peer review (for example as in Wikipedia). In fact, we see similar approaches to these used in volunteer computing settings where compute nodes are inherently unreliable e.g. [2, 7].

Despite the level of research into quantitative methods of deriving and defining the quality and reliability of results in the crowdsourcing literature, it is hard to avoid the more subjective issues surrounding the perception of quality with respect to the qualifications and/or competence of a worker. We observed in [13] that although crowdsourcing platforms use several mechanisms to assess worker reliability and capabilities these methods can seldom be applied to identify actual worker abilities or competencies. Instead, they reveal only whether the worker is likely to posses the necessary abilities to perform a specific (micro) task and/or if they will do so diligently. If we consider the notion of a Social Cloud, where computational resources are crowdsourced within a specific (social) community, we can see that the "standard" means of assessing quality may not be sufficient. In [13], we attempted to disentangle quality and competence; where the latter is potentially influential on ex-ante expectations of quality, and thus in decisions related to resource allocations as well as requests.

An alternative method to assessing quality was proposed by [25]: to infer a level of trust in the worker via the accuracy of their solutions. For the purposes of a Social

[2]Tasks where the solution is known ex-ante to test worker accuracy.

Cloud, we can augment this approach by redefining accuracy as either perceived quality ex-ante (in a qualitative sense) or observed quality ex-post using predefined measures for quality of service (in a quantitative sense). This would capture the two methods of assessing quality mentioned above. We can also augment [25]'s inference of trust through the assumption of pre-existent inter-personal trust between members of a Social Cloud. In assessing quality in this manner, we are highlighting two artifacts of a Social Cloud: trust and some form of provisioning agreement.

To avoid a lengthy discussion on trust, we refer to [6] where we defined trust for the context of a Social Cloud as follows: "Trust is a positive expectation or assumption on future outcomes that results from proven contextualized personal interaction-histories corresponding to conventional relationship types and can be leveraged by formal and informal rules and conventions within a Social Cloud to facilitate as well as influence the scope of collaborative exchange". Two of the most relevant aspects in this definition from a crowdsourcing perspective are "interaction-histories" and "conventional relationship types". Where the former overlaps with ex-post measures of prior performance (potentially) in various collaborative contexts, the latter is somewhat abrasive in the context of crowdsourcing. We tend to view workers as a part of a large anonymous human crowd of workers. In a Social Cloud this is not the case (nor may it be the case in software crowdsourcing projects), and consequently, we can view trust at a more subjective and personal level, and (at least) assume that the existence of trust will be positively correlated to quality in the general sense.

However, the ability of users to rely on trust alone is dependent on the type of relationship [41]. This differentiated view of trust means that in some circumstances some form of collaborative agreement that clearly defines measures of quality is needed. In [9, 10] we explored the implementation details of formal agreements or SLAs (Service Level Agreements). However, this in retrospect was an over-engineered approach. Reference [41] observed that in the relationship contexts (close) friend and family, agreements are not perceived necessary by users. However, in relationship contexts such as acquaintances and colleagues (the arguably more likely relationship types in crowdsourcing contexts) some formalization of an agreement as well as some form of explicit incentive to contribute is necessary. This does not, however, imply the formal representation of a collaborative action using a standard like WS-Agreement [3], but rather a leaner representation capturing: the minimum details of the exchange (actors, and definition of instance); basic measures of quality (e.g. availability); implications of failure; and (when applicable) a definition of reward be it tangible, e.g., monetary payment, or intangible, e.g. reputation points.

4 A Social Cloud for Software Crowdsourcing

In this section we present a general Social Cloud architecture and describe how the principles of Social Cloud Computing can be leveraged in software crowdsourcing applications. As software development processes are inherently collaborative, they involve groups of contributors who are in some cases unknown and in others based on strong social ties between one another; they require sophisticated resources and software to develop, test, deploy, integrate, and perform other common software lifecycle processes; and they rely on contributions from various people to achieve these goals. Thus, there are two areas in which Social Cloud principles can be applied to software crowdsourcing: (1) as a model for supporting software crowdsourcing infrastructure requirements, and (2) as a means of using social network analysis to derive competency and quality.

4.1 General Social Cloud Architecture

Based on our previous experience, we now present a unified architecture that supports Social Storage, Social CDN and Social Compute Clouds. Like any Cloud model, and as discussed above, a platform is required to coordinate and facilitate the basic functionality of a Social Cloud (user management, resource allocation, etc.). The resources to support this platform, can either be provided by a third party, or crowd-sourced from the Social Cloud user base, as a form of co-operative platform [22]. Figure 6 shows the high level architecture for a Social Cloud and its key components, which are as follows:

A **Social Marketplace** is an institutionalized microeconomic system that defines how supply is allocated to demand. In other words, it is responsible for the facilitation,

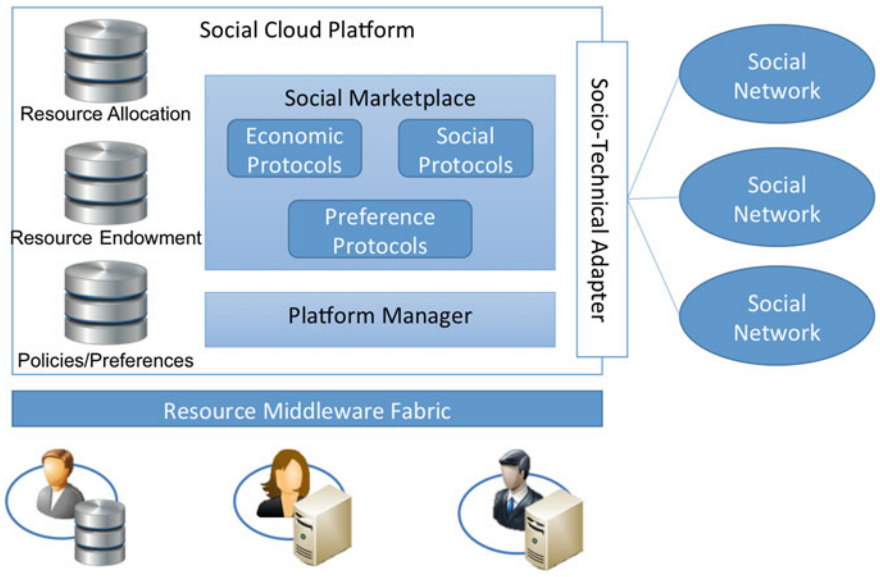


Fig. 6 General Social Cloud Architecture

and clearing of calls. A social marketplace captures the following: the protocols used for distributed resource allocation, the rules of exchange, i.e. who can take part and with whom may they exchange, and the formalization of one or more allocation mechanisms. A social marketplace is therefore the central point in the system where all information concerning users, their sharing policies, and their resource supply and demand is kept. For this reason, the social marketplace requires data stores: to capture the participants, the social graph of its users, as well as their sharing policies; and a resource manager to keep track of resource reservations, availability, and allocations.

A **Platform Manager** administrates the basic functionality of the Social Cloud. The platform manager is a (semi-)autonomic co-operative system managing its resources either through the creation of calls for platform resources, or syphoning off parts of contributed computational resources. It is responsible for ensuring that the Social Cloud is responsive and available. Such approaches may also be applied more widely as a means of supporting other crowdsourcing platforms.

A **socio-technical adapter** provides access to the necessary aspects of users' social networks, and acts as a means of authentication. The socio-technical adapter could leverage any source for social information such as an existing social network platform or a software crowdsourcing application. Once a user's social network has been acquired via the socio-technical adapter, the social marketplace requires the preferences and policies of the user to facilitate resource allocation.

Data stores record state for the Social Cloud such as users, social graphs, resource endowments, policies and preferences, and current and historical allocations. These data stores are used by the social marketplace to influence allocation and by participants to manage their interactions with the social cloud.

A **resource middleware fabric** provides the basic resource fabrics, resource virtualization and sandboxing mechanisms for provisioning and consuming resources. In the examples above the middleware includes mechanisms to access the storage and compute resources of participants. It defines the protocols needed for users and resources to join and leave the system. The middleware is also responsible for ensuring secure sharing of resources by implementing interfaces and sandboxed environments that protect both providers and consumers.

Resources are the technical endowment of users that are provided to, and consumed from, the Social Cloud. These resources could include personal computers, servers or clusters and specifically the storage and compute capabilities that these resources make available.

4.2 Crowdsourcing Infrastructure for Software Crowdsourcing

Figure 7 shows how the unified Social Cloud architecture can be used to crowdsource infrastructure required to facilitate a software development project. In this case, various users–some with existing relationships between one another and some

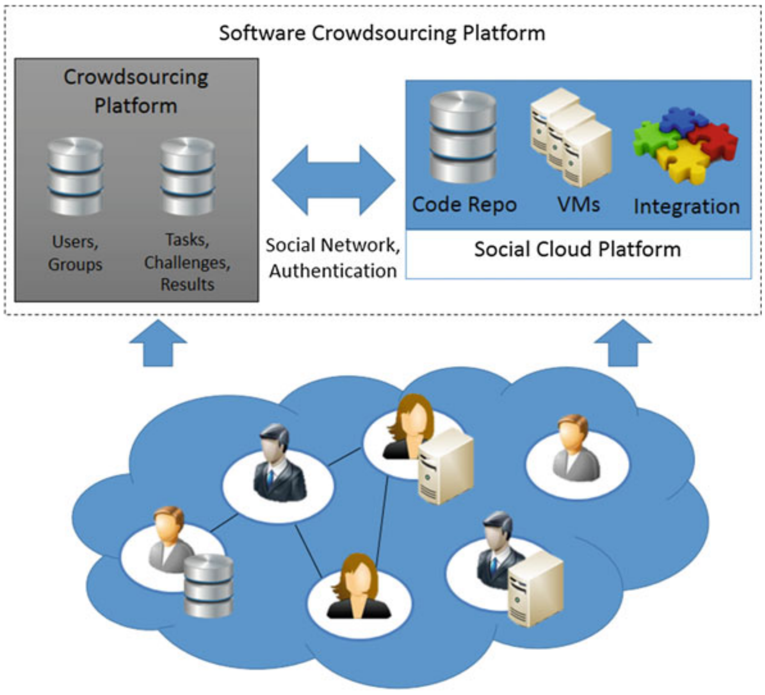

Software Crowdsourcing Platform
Crowdsourcing Platform
Users, Groups
Tasks, Challenges, Results
Social Network, Authentication
Code Repo
VMs
Integration
Social Cloud Platform

Fig. 7 A software crowdsourcing model that leverages a Social Cloud contributed by participants to facilitate operation of important software development services

without–are participating in a software crowdsourcing project. Not only are these users contributing their software engineering capabilities to the software crowdsourcing project but some are also contributing resources via an associated Social Cloud that allows the hosting and execution of various important software development services for the project.

The Social Cloud leverages the social network derived from the software crowdsourcing platform or potentially from other social networks that exist between users. We expect that in most cases the crowdsourcing platform will provide the authentication and authorization model as well as providing user and group information to the Social Cloud. The Social Cloud uses this information (e.g., groups) to enable user contributed resources and capabilities to be shared with other participants. By establishing a network (based on groups) between users we can also enable other techniques to improve collaboration including, for example, the ability to provide communication between members and to facilitate rapid feedback. It may also provide mechanisms to more easily discover capable participants and motivate contribution [27].

When selecting a software project for participation, participants will be able to optionally contribute resources to the project. Users will be presented a Social Cloud interface that enables the selection of local resources, and by using existing storage and compute services they will be able to make their resources accessible to the Social Cloud. At this stage agreement on the market model must be made (e.g., how are users rewarded for their contribution) at the same time as reaching agreement regarding their participation in the project (e.g., rewards associated with their software engineering contributions). Policies will need to be developed regarding who can access a contributed resource and for what reason. For example, perhaps resources will be contributed for short term leases to support testing; or alternatively they might be contributed for longer periods to support project wide services such as code repositories.

Other project members can then access and use resources contributed to the Social Cloud through published interfaces and adhering to the market protocols selected. Depending on the contribution and market model, they may be able to consume resources at will by virtue of being a member of the project or they may require some form of matching between their requirements and others capabilities. Resource allocation will follow the general Social Cloud model in which virtualized and sandboxed resources are offered and used by other participants.

There are several different market approaches that can be applied to a software crowdsourcing Social Cloud: (1) The market may be tied to the model employed by the crowdsourcing application, or (2) the market may be independent from the crowdsourcing application and operate only within the context of the Social Cloud.

In the first model, the Social Cloud will use the credits (virtual or real) that are associated with the software crowdsourcing platform. In this case, contributors will associate a credit value with their resource contributions using an economic model. For instance, a contributor may charge $1 credit per day for using their compute resources. Similarly, consumers (or the project) will be charged credits when accessing or using these resources. If resources are used for the project itself, these credits may be taken from the overall reward associated with the project. We expect that such capabilities are present in most crowdsourcing applications and that we can leverage these workflows via published APIs or other means.

In the second model, the Social Cloud will apply an independent economic model such as a credit model, social network model, or preference matching model. The difference between an independent credit model and the credit model tied to the crowdsourcing application is that all credits and exchanges will be managed entirely by the Social Cloud. The social network model will allow project participants to share resources with a group, enabling, for example specification of restrictions based on relationships between individuals. Such restrictions may include allowing access to only close friends or collaborators who have worked together on a previous project, or who share a connection in an external social network. The preference model provides a more regulated mechanism for members to control their contributions and use of resources. In this model users will be able to contribute resources and optionally define preferences for which members can use these resources. Consumers, when requesting resources, may define preferences for which members they use resources from.

4.3 Establishing Trust and Competency via Social Networks

Online social networks such as Facebook, Google+ and Twitter provide a model in which relationships between individuals are encoded digitally and can be accessed programmatically to develop socially-aware applications. Even implicit social networks formed via user accounts, groups, publications and online actions can be extracted and used to establish linkages between individuals. The premise of a Social Cloud builds upon these relationships to infer a level of trust between individuals. This trust can be leveraged to encourage higher levels of quality of service in sharing settings.

As discussed in Sect. 3 such networks provide a powerful model for inferring capabilities and competencies. In traditional crowdsourcing models such requirements are less prevalent as tasks are often oriented around unskilled activities. However, in software crowdsourcing projects it is important to establish levels of proficiency. This is increasingly valuable as it is often difficult to determine the accuracy of self-reported competency, and equally challenging to construct suitable measures to capture, in general, transferable notions of competence for crowdsourcing [13]. While reputation measures provide a method to address these requirements, they require bootstrapping and also reliability that the reputation model cannot be subverted. We believe that the ability to leverage trusted connections between individuals following a Social Cloud model is of particular value in the general crowdsourcing domain for these reasons.

To do this, we first need to define the frames of reference when discussing trust for a Social Cloud or in settings where social structures are used in crowdsourcing. We identified in [6] three frames of reference for trust: (1) trust as an intrinsic (subjective) attribute of an inter-personal social relation, i.e. trust as a basic foundation of social actions; (2) trust in the competence of an individual to be able to deliver a given resource or capability, i.e. a belief in the self-awareness and personal evaluation of competence; and (3) trust in an individual to deliver, i.e. keep their promises, adhere to any (informal) agreements etc. While it is easy to conceive the first and third frame of reference being captured by the social network structure and any associated SLA, soft agreement or "gentleman's agreement" respectively, it is however, difficult to interpret the second frame of reference without a basic definition of competence.

In [13] we provide an overarching summary of competence in crowdsourcing scenarios. Where competence refers to an inflected action with respect to "practical knowledge" and is characterized by "being able to", "wanting to", "being allowed to", and "being obliged to" do something [39]. In crowdsourcing literature the term of competence is often misused to refer to a capability or some form of (domain) authority [13].

We argue that the premises of a Social Cloud could provide a basic framework for (in)formally observing competence as a consequence of network effects: the social ascription of competence by peers. For the ascription of competence a "social" element comes into play as a reciprocal situational ascription of appropriateness to specific actions by the actor themselves and another person or persons (e.g. a collaborator) who are participating in the relevant situation [13]. Consider, for example, a system similar to LinkedIn endorsements. While such a system would be particularly apt in a software crowdsourcing model issues surround the interpretation of reputation. Social networks, however, provide a mechanism to apply transitive inference, that is, to traverse the graph of endorsements based on previous interactions with a particular entity or project. Such approaches can be used to further enhance the reliability of competence measures.

Coming back to the original frames of reference the approaches put forth by Social Clouds are equally applicable to a software crowdsourcing environment. The implicit and explicit use of social networks provide models to: (1) establish trust between individuals and projects; (2) provide the ability to infer competence based on previous interactions and endorsements; and (3) utilize inherent social incentives and disincentives associated with participation and delivery.

5 Related Work

Until now there has been little research into the implementation of software crowdsourcing applications. Most examples such as TopCoder, uTest and user-contributed application stores are commercially developed and focus on specific aspects of the software development lifecycle. As yet, these approaches offer only limited infrastructure capabilities for their projects, for example TopCoder offers cloud resources on specific projects.

There is however, much literature relating to the exchange or sharing of resources using social fabrics. For example, Intel's "progress thru processors"[3] Facebook application enables contribute of excess compute power to individually selected scientific projects. Users are not rewarded for their contribution as such, however they can view and publish statistics of their contributions. Upon joining the application users may post information to their news feed, or inform friends of the application. The progress thru processors application relies on a generic resource layer constructed by deploying a BOINC [2] application on the users machine.

McMahon and Milenkovic [34] proposed Social Volunteer Computing, an extension of traditional Volunteer Computing, where consumers of resources have underlying social relationships with providers. This approach is similar to the nature of a Social Cloud, but it does not consider the actual sharing of resources, as there is no notion of bilateral exchange.

Pezzi [38] proposes a Social Cloud as a means of cultivating collective intelligence and facilitating the development of self-organizing, resilient communities. In this vision the social network and its services are provided by network nodes owned by members of the network rather than by centralized servers owned by the social network. Pezzi's work is in its infancy and has no architectural details or implementation.

[3]http://www.facebook.com/progressthruprocessors.

Ali et al. [1] present the application of our Social Cloud model to enable users in developing countries to share access to virtual machines through platforms like Amazon EC2. In effect they subdivide existing allocations to amortize instance cost over a wider group of users. Using a cloud bartering model (similar to our previous virtual credit model), the system enables resource sharing using social networks without the exchange of money and relying on a notion of trust to avoid free riding. Like our approach, they use a virtual container (LXC) to provide virtualization within the existing virtual machine instance.

Mohaisen et al. [35] present an extension to our definition of a Social Cloud. The authors investigate how a Social Compute Cloud could be designed, and propose extensions to several well known scheduling mechanisms for task assignments. Their approach considers resource endowment and physical network structure as core factors in the allocation problem.

Gracia-Tinedo et al. [18–20] propose a Friend-to-Friend Cloud storage solution, i.e. dropbox via a social network: F2Box. They analyze and discuss how to retain a reliable service whilst using the best effort provisioning of storage resources from friends. They identify that a pure friend-to-friend system cannot compare in terms of quality of service with traditional storage services. Therefore, they propose a hybrid approach where reliability and availability can be improved using services like Amazon's S3. This approach provides a valuable consideration in the realization of a Social Cloud, but is not necessarily transferable to our setting.

Wu et al. [45, 46] describe a lightweight framework to enable developers to create domain specific collaborative science gateways. They use social network APIs (OpenID, OAuth and OpenSocial) and virtualized cloud resources to facilitate collaboration as well as fine grained and dynamic user controlled resource sharing using social network-based group authorization. These same authorization models are then used to facilitate execution of computational bioinformatics jobs on customized virtual machines shared between users.

Kuada and Olesen [31] propose opportunistic cloud computing services (OCCS): a social network approach for the provisioning and management of enterprise cloud resources. Their idea is to provide a governing platform for enterprise level social networking platforms consisting of interoperable Cloud management tools for the platform's resources, which are provided by the enterprises themselves. The authors, present the challenges and opportunities of an OCCS platform, but there is no indication that they have yet built an OCCS. Similarly, Diaspora,[4] and My3 [36] apply similar concepts to host online social networks on resource provided by their users.

There have also been several publications on economic models for a Social Cloud. Zhang et al. [47] and we [21] discuss different types of incentives users face during their participation in a Social Cloud, and describe the challenges of providing the right incentives to motivate participation. While in another study [22], we investigated how the infrastructure of a Social Cloud can be co-operatively provided by the participating members, and present an economic model that takes individual incentives and resource availability into account.

[4]https://joindiaspora.com/.

6 Conclusion

As software crowdsourcing becomes an increasingly viable alternative to dedicated software development teams the infrastructure required to support such dynamic collaborations will continue to increase. Already, software crowdsourcing projects are turning to cloud resources as a model for providing resources on which tasks can be completed. However, we argue that these approaches will not scale and may become costly as projects become larger and more common. In this chapter we have described an alternative approach in which the very same crowdsourcing principles are applied to acquire infrastructure resources. Through the use of an infrastructure crowdsourcing model users can assemble a virtual infrastructure on which software development processes can be performed.

Social Clouds are a well studied approach for facilitating the exchange of infrastructure resources using various economic and non-economic protocols. They provide a model for exchanging heterogeneous resources between individuals connected via a social graph. Where the social graph provides the ability to derive relationships and therefore infer pre-existent trust between users, which in turn can be used to increase quality and trustworthiness with respect to shared infrastructure. Social Clouds provide the necessary mechanisms to ensure resources are used securely and to manage allocation across a pool of participants. In previous work, we have developed three Social Clouds, focused on storage, content delivery, and compute. We have also explored the use of different allocation protocols based on credit, social network, and preference models. These Social Clouds and their associated allocation models provide a generic basis on which other services can be developed.

The use of a Social Cloud model as a basis for, or a companion to, a software crowdsourcing system will enable individual projects to leverage not only the skills contributed by participants but also their infrastructure resources. This integration will allow the deployment and operation of important software development services hosted collectively by the project's members. The use of a infrastructure crowdsourcing approach is perhaps the most appropriate model for provisioning infrastructure given that the philosophies behind crowdsourcing software and infrastructure are the same. Finally, the same social network analysis algorithms used in a Social Cloud to infer trust and competency may also provide value in a software crowdsourcing model.

References

1. Ali, Z., Rasool, R.U., Bloodsworth, P.: Social networking for sharing cloud resources. In: 2012 Second International Conference on Cloud and Green Computing (CGC), pp. 160–166 (2012)
2. Anderson, D.P.: Boinc: a system for public-resource computing and storage. In: 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004)
3. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (WS-agreement). In: Open Grid Forum, vol. 128 (2007)

4. Cappos, J., Beschastnikh, I., Krishnamurthy, A., Anderson T.: Seattle: a platform for educational cloud computing. In: The 40th Technical Symposium of the ACM Special Interest Group for Computer Science Education (SIGCSE'09), Chattanooga, TN USA (2009)
5. Caton, S., Haas, C., Chard, K., Bubendorfer, K., Rana, O.: A social compute cloud: allocating and sharing infrastructure resources via social networks (2014)
6. Caton, S., Dukat, C., Grenz, T., Haas, C., Pfadenhauer, M., Weinhardt, C.: Foundations of trust: contextualising trust in social clouds. In: 2012 Second International Conference on Cloud and Green Computing (CGC), pp. 424–429. IEEE (2012)
7. Caton, S., Rana, O.: Towards autonomic management for cloud services based upon volunteered resources. Concurr. Comput.: Pract. Exp. **23** (2011). Special Issue on Autonomic Cloud Computing: Technologies, Services, and Applications
8. Chard, K., Bubendorfer, K.: Using secure auctions to build a distributed meta-scheduler for the grid. In: Buyya, R., Bubendorfer, K. (eds.) Market Oriented Grid and Utility Computing. Wiley Series on Parallel and Distributed Computing, pp. 569–588. Wiley, New York (2009)
9. Chard, K., Bubendorfer, K., Caton, S., Rana, O.: Social cloud computing: a vision for socially motivated resource sharing. IEEE Trans. Serv. Comput. **99**(PrePrints), 1 (2012)
10. Chard, K., Caton, S., Rana, O., Bubendorfer, K.: Social cloud: cloud computing in social networks. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 99–106 (2010)
11. Chard, K., Caton, S., Rana, O., Katz, D.S.: A social content delivery network for scientific cooperation: vision, design, and architecture. In: The Third International Workshop on Data Intensive Computing in the Clouds (DataCloud 2012) (2012)
12. Czajkowski, K., Ferguson, D.F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W.: The WS-resource framework. Technical report, Globus. http://www.globus.org/wsrf/specs/ws-wsrf.pdf (2004). Accessed Dec 2010
13. Dukat, C., Caton, S.: Towards the competence of crowdsourcees: literature-based considerations on the problem of assessing crowdsourcees' qualities. In: International Workshop on Crowdwork and Human Computation at the IEEE Third International Conference on Cloud and Green Computing (CGC), pp. 536–540. IEEE (2013)
14. Foster, I.: Globus online: accelerating and democratizing science through cloud-based services. IEEE Internet Comput. **15**(3), 70–73 (2011)
15. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. **15**, 200–222 (2001)
16. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. **69**, 9–15 (1962)
17. Geiger, D., Seedorf, S., Schulze, T., Nickerson, R.C., Schader, M.: Managing the crowd: towards a taxonomy of crowdsourcing processes. In: AMCIS (2011)
18. Gracia-Tinedo, R., Sanchez-Artigas, M., Garcia-Lopez, P.: Analysis of data availability in F2F storage systems: when correlations matter. In: 2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P), pp. 225–236. IEEE (2012)
19. Gracia-Tinedo, R., Sánchez-Artigas, M., Garcia-Lopez, P.: F2box: cloudifying F2F storage systems with high availability correlation. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 123–130. IEEE (2012)
20. Gracia-Tinedo, R., Sánchez-Artigas, M., Moreno-Martinez, A., Garcia-Lopez, P.: Friendbox: a hybrid F2F personal storage application. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 131–138. IEEE (2012)
21. Haas, C., Caton, S., Weinhardt, C.: Engineering incentives in social clouds. In: Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011), pp. 572–575 (2011)
22. Haas, C., Caton, S., Chard, K., Weinhardt, C.: Co-operative infrastructures: an economic model for providing infrastructures for social cloud computing. In: Proceedings of the Forty-Sixth Annual Hawaii International Conference on System Sciences (HICSS), Grand Wailea, Maui, USA (2013)

23. Haas, C., Kimbrough, S., Caton, S., Weinhardt, C.: Preference-based resource allocation: using heuristics to solve two-sided matching problems with indifferences. In: 10th International Conference on Economics of Grids, Clouds, Systems, and Services (Under Review) (2013)
24. Halldórsson, M.M., Iwama, K., Miyazaki, S., Yanagisawa, H.: Improved approximation results for the stable marriage problem. ACM Trans. Algorithms (TALG) **3**(3), 30 (2007)
25. Ipeirotis, P.G., Provost, F., Wang, J.: Quality management on Amazon Mechanical Turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, pp. 64–67. ACM (2010)
26. Irving, R.W., Leather, P., Gusfield, D.: An efficient algorithm for the optimal stable marriage. J. ACM **34**(3), 532–543 (1987)
27. John, K., Bubendorfer, K., Chard, K.: A social cloud for public eResearch. In: Proceedings of the 7th IEEE International Conference on eScience. Stockholm, Sweden (2011)
28. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
29. Kern, R., Zirpins, C., Agarwal, S.: Managing quality of human-based eservices. Service-Oriented Computing-ICSOC 2008 Workshops, pp. 304–309. Springer, New York (2009)
30. Kittur, A., Nickerson, J.V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., Horton, J.: The future of crowd work. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, pp. 1301–1318. ACM (2013)
31. Kuada, E., Olesen, H.: A social network approach to provisioning and management of cloud computing services for enterprises. In: The Second International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2011, pp. 98–104 (2011)
32. Kugler, K., Chard, K., Caton, S., Rana, O., Katz, D.S.: Constructing a social content delivery network for escience. In: 2013 IEEE 9th International Conference on eScience (eScience), pp. 350–356 (2013)
33. Lonstein, E., Lakhani, K., Garvin, D.: Topcoder (a): developing software through crowdsourcing. Technical report, Harvard Business School General Management Unit Case (2010)
34. McMahon, A., Milenkovic, V.: Social volunteer computing. J. Syst. Cybern. Inf. (JSCI) **9**(4), 34–38 (2011)
35. Mohaisen, A., Tran, H., Chandra, A., Kim, Y.: Socialcloud: using social networks for building distributed computing services. arXiv:1112.2254 (2011)
36. Narendula, R., Papaioannou, T.G., Aberer, K.: My3: a highly-available P2P-based online social network. In: 2011 IEEE International Conference on Peer-to-Peer Computing (P2P), pp. 166–167. IEEE (2011)
37. Oleson, D., Sorokin, A., Laughlin, G.P., Hester, V., Le, J., Biewald, L.: Programmatic gold: targeted and scalable quality assurance in crowdsourcing. Hum. Comput. **11**, 11 (2011)
38. Pezzi, R.: Information technology tools for a transition economy, September 2009
39. Pfadenhauer, M.: Competence-more than just a buzzword and a provocative term? Modeling and Measuring Competencies in Higher Education, pp. 81–90. Springer, New York (2013)
40. Pisano, G.P., Verganti, R.: Which kind of collaboration is right for you. Harv. Bus. Rev. **86**(12), 78–86 (2008)
41. Thal, R.: Representing agreements in social clouds. Master's Thesis, Karlsruhe Institute of Technology (2013)
42. Thaufeeg, A.M., Bubendorfer, K., Chard, K.: Collaborative eResearch in a social cloud. In: 2011 IEEE 7th International Conference on E-Science (e-Science), pp. 224–231 (2011)
43. Tokarchuk, O., Cuel, R., Zamarian, M.: Analyzing crowd labor and designing incentives for humans in the loop. IEEE Internet Comput. **16**(5), 45–51 (2012)
44. Wu, W., Tsai, W.-T., Li, W.: An evaluation framework for software crowdsourcing. Front. Comput. Sci. **7**(5), 694–709 (2013)
45. Wu, W., Zhang, H., Li, Z.: Open social based collaborative science gateways. In: 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp, 554–559. IEEE (2011)
46. Wu, W., Zhang, H., Li, Z., Mao, Y.: Creating a cloud-based life science gateway. In: 2011 IEEE 7th International Conference on E-Science (e-Science), pp. 55–61. IEEE (2011)
47. Zhang, Y., van der Schaar, M.: Incentive provision and job allocation in social cloud systems. To appear in IEEE J. Sel. Areas Commun. (2013)