Cluster Computing (2019) 22:455–470
https://doi.org/10.1007/s10586-018-2843-2

Crowdcloud: a crowdsourced system for cloud infrastructure

Mahmood Hosseini[1] · Constantinos Marios Angelopoulos[1] · Wei Koong Chai[1] · Stephane Kundig[2]

Received: 15 November 2017 / Revised: 15 February 2018 / Accepted: 20 August 2018 / Published online: 30 August 2018

Abstract
The widespread adoption of truly portable, smart devices and Do-It-Yourself computing platforms by the general public has enabled the rise of new network and system paradigms. This abundance of well-connected, well-equipped, affordable devices, when combined with crowdsourcing methods, enables the development of systems with the aid of the crowd. In this work, we introduce the paradigm of *Crowdsourced Systems*, systems whose constituent infrastructure, or a significant part of it, is pooled from the general public by following crowdsourcing methodologies. We discuss the particular distinctive characteristics they carry and also provide their "canonical" architecture. We exemplify the paradigm by also introducing *Crowdcloud*, a crowdsourced cloud infrastructure where crowd members can act both as cloud service providers and cloud service clients. We discuss its characteristic properties and also provide its functional architecture. The concepts introduced in this work underpin recent advances in the areas of mobile edge/fog computing and co-designed/co-created systems.

Keywords Crowdsourcing · Crowdsourced systems · Cloud services · Crowdcloud · Mobile edge computing

1 Introduction

Cloud computing is a method of providing computing resources as a service rather than a product. It is extensively used by both for-profit organisations such as Google App Engine [13, 72] and non-profit organisations such as Science Cloud [43] to provide services in three different ways: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The many powerful characteristics of cloud computing, such as cost reduction, device and location independence, easier maintenance, higher performance capabilities, more reliability, and higher scalability have helped to expand the notion of cloud computing and broaden its applications and usage [24].

These "as-a-service" architectures have leveraged new models of resource orchestration and task execution, and can be combined with crowdsourcing for novel, unprecedented applications. *Crowdsourcing* is a method of outsourcing tasks to a typically large, undefined group of people via an open call [38]. Crowdsourcing provides an opportunity for crowdsourcers to increase the efficiency of executing the crowdsourced task both in terms of the incurred costs and the time required. The reduction of both money and time needed in obtaining possible solutions, plus opening the in-house innovation and problem-solving processes to the large diverse crowd can also lead to attracting more creativity and wisdom that might otherwise not be found inside organisations.

The widespread adoption of truly portable, hand-held smart devices (such as smart phones and smart wearables) by the general public, as well as the rise of Do-It-Yourself computing platforms (such as the Arduino or the Raspberry Pi) have formed a new reality where devices with significant computational and communication capabilities are abundant. This ubiquitous presence of smart devices provided by the general public offers an unprecedented ability of augmenting traditional computer networks and systems

✉ Mahmood Hosseini
mhosseini@bournemouth.ac.uk

Constantinos Marios Angelopoulos
mangelopoulos@bournemouth.ac.uk

Wei Koong Chai
wchai@bournemouth.ac.uk

Stephane Kundig
stephane.kundig@unige.ch

[1] Bournemouth University, Poole, UK

[2] University of Geneva, Geneva, Switzerland

with *crowdsourced resources*. This qualitatively extends the notion of crowdsourcing from that of pooling human resources (either in the form of manpower or crowd wisdom) to that of pooling the ICT infrastructure needed for the execution of a task, which we refer to as crowdsourced systems.

Crowdsourced systems are, roughly speaking, systems whose constituent infrastructure (or at least a significant part of it) is pooled from the general public following crowdsourcing methodologies. A well-established example of such systems are the *Mobile Crowdsensing Systems* (MCSs) [22, 45], a special case of crowdsourced systems focusing on collecting sensory data from the general public with the use of smart phones. In this article, we introduce another example of a crowdsourced system, namely the *crowdcloud*.

Our contribution we formally introduce the notion of *Crowdsourced Systems*. First, we provide the background that motivates the introduction of this new paradigm in Sect. 2. Then we provide the corresponding definition and also identify the main components of such systems and characterise their interactions by providing the core architecture of textitcrowdsourced systems in Sect. 3. In Sect. 4, we provide a high-level comparison between crowdsourced systems and similar paradigms such as Internet of Things (IoTs) as a service.

Following, we introduce *crowdcloud* in Sect. 5, a crowdsourced system paradigm that refers to the availability of cloud infrastructure, cloud platform, and cloud software services to the crowd by the crowd with or without a legally binding contract. We outline the scope of crowdcloud, present its architecture and explain its relationship to crowdsourced systems in general. In Sect. 6, we illustrate our ongoing work on the pathway to the implementation of a framework for crowdsourced cluster computing. We conclude the paper and provide the future work in Sect. 7.

2 Related work

2.1 Background on crowdsourcing

Crowdsourcing is usually defined as the practice of collecting and aggregating needed services, information, or other kind of resources provided by the general public [32]. Crowdsourcing as a practice has been utilised in numerous domains of study, such as business, computer science, and medicine. Crowdsourcing has also been utilised in several commercial and non-commercial platforms, such as Amazon Mechanical Turk [40] and Threadless [8], and has been structured in several forms, such as micro tasking [44] and crowdfunding [50].

Furthermore, when applied in complex tasks, crowdsourcing can leverage the so called *wisdom of the crowd* [62], e.g., crowdsourcing for the purpose of applying the wisdom of the crowd within enterprises [30]. The notion in this particular application of crowdsourcing is that apart from being an efficient and cost effective method of pooling resources, crowdsourcing can also provide qualitative benefits in task execution, i.e., by employing the collective intelligence of the crowd, ordinary people can often outperform individual experts.

In spite of the common understanding, crowdsourcing has been employed as a method for many years in various forms. One example is the provision of juries in several judicial systems. Another example that is widely used in the literature comes from 1906, when statistician Francis Galton observed that the aggregated result from the estimations of the weight of an ox from 800 people was accurate within the 1% error margin [21]. The latter example nicely demonstrates the mechanism that underpins crowdsourcing; individual contributions can be seen as sampling points of a probability distribution with the true answer as its mean value. This interpretation characterises crowdsourcing methodologies as relying on statistical sampling and therefore provides hints towards a wise crowd (e.g., homogeneous coverage of the sampling space, stochastic independence of trials, etc.).

In his book [62], Surowiecki has identified the four characteristics a crowd needs to have in order to be "wise". These characteristics are:

- *Diversity*, i.e., each individual of the crowd should carry its own contribution/information, even if this is an eccentric one.
- *Independence*, i.e., each individual's contribution should not be determined or largely influenced by other individuals.
- *Decentralisation*, i.e., each individual's contribution should be formed based on locally available information and knowledge.
- *Aggregation*, i.e., a correct mechanism should exist for aggregating individual contributions into a collective outcome.

Later, we will refer to these characteristics when characterising the system requirements for Crowdsourced Systems.

2.2 Background on mobile crowdsensing systems

During the past few years, smart phones and other truly portable devices (such as tablets, smart watches and smart glasses) have evolved into sophisticated multi-sensory computing platforms, thus fuelling the rise of MCSs. In

[22], an overview is provided of the current state of applications that are based on MCS. The main challenges recognised refer to resource limitations, such as available energy, bandwidth and computational power, privacy issues that may arise due to the correlation of sensor data with individuals, and the lack of a unifying architecture that would optimise the cross-application usage of sensors on a particular device or even on a set of correlated devices (e.g., if they are located in the same geographical area).

In [55], authors use the notion of Participatory Sensing (PS) to describe such systems. They consider the problem of efficient data acquisition methods for multiple PS applications while taking into consideration issues such as resource constraints, user privacy, data reliability, and uncontrolled mobility. They evaluate heuristic algorithms that seek to maximise the total social welfare via simulations that are based on mobility datasets consisting of both real-life and artificial data traces.

In a previous work, we identified the basic design issues of MCS and investigated some characteristic challenges [5]. In particular, the core elements of an MCS were defined—the *task*, the *server*, and the textitcrowd—along with the functions governing their interactions. For a given type of *task*, and a finite *budget*, the *server* makes offers to the agents of the *crowd* based on some *incentive policy*. Then, each individual of the *crowd* makes a decision on whether to contribute to the execution of the *task* based on its own *utility function*. From this formulation, interesting results are extracted on the heuristics the *server* can follow in order to increase the efficiency of the system subject to the available *budget*.

2.3 Background on Internet of Things

In a more applied work presented in [6], an IoT testbed architecture for *smart buildings* was presented that enables the seamless and scalable integration of crowdsourced resources, such as smart phones and tablets. The purpose of this integration is two-fold; first, the embedded sensory capabilities of the resources provided by the crowd are combined with the sensing capabilities of the building for efficient smart actuations. Second, the system is able to interact with its users in a direct, personal way both for incentivising them to provide sensory data from their devices and for receiving feedback on their preferences and experienced comfort. This work is among the very first demonstrating the use of crowdsourcing in order to opportunistically augment the infrastructure of an ICT system. It also highlights the dual nature of crowdsourcing where the crowd not only contributes to the system but also ameliorates it, thus receiving services of higher quality. The same principles are also followed in [19], although in a different context. Here, the focus is on employing crowdsourcing as a powerful tool not only for conducting research, but also for driving research via the co-design of experiments for problems proposed by the crowd.

This brings us to the focal point of this study, i.e., the notion of crowdsourced systems and the application of crowdsourcing in the domain of cloud computing. The stimulating advantages of crowdsourcing and the extensive capabilities of cloud computing, plus the existence of similar characteristics (e.g., reducing costs and increasing diversity), facilitate a solid ground for the unification of the two practices. Such a unification has already been noticed and utilised in some cloud projects such as SETI@home [4] and BOINC [2]. However, these cloud projects belong to corporations and organisations, i.e., the crowd resources have been utilised not by other crowd members but by organisations. For example, SETI@home belongs to Berkeley and Microsoft Azure belongs to Microsoft. As a result, we believe that there is still a lack of a comprehensive cloud infrastructure that can actually be stemmed from the crowd, be organised by the crowd, and be utilised for the crowd. While Torrent clients and similar peer-to-peer platforms do exist, they are mainly used for file sharing and not for sharing other cloud resources such as computing power, cloud storage, and software on-demand services.

2.4 Background on cloud market models

The cloud marketplace, as the online storefront for providing cloud services by cloud service providers, is not a recently devised concept [7]. However, the cloud market is still mainly dominated by a handful of mostly private (and sometimes public) cloud providers, which are both willing and capable of investing in their cloud service provision and the required infrastructure [7]. Examples of well-known cloud marketplaces include the Amazon AWS cloud marketplace, the Oracle cloud marketplace, the Microsoft Azure cloud marketplace, and the Salesforce AppExchange cloud marketplace [47].

With the move from Cloud 1.0 to Cloud 2.0, which adds the new Web 2.0 social networking functionalities to the cloud marketplace [52], more emphasis has been put on the role of new players in this marketplace. For example, the Open Cloud Exchange (OCX) has been proposed as a public cloud marketplace where several stakeholders participate in implementing and operating the cloud, as opposed to only one cloud provider [7]. Intercloud is another attempt to support and utilise the scaling of applications across multiple vendor clouds in the cloud marketplace [9], which we will discuss later as well. Other similar cloud marketplaces and their characteristics have been extensively researched and proposed, e.g., in [14, 48].

Most of these newer paradigms of cloud computing still have two very distinct players: cloud service providers and cloud service clients. In other words, cloud service clients usually cannot be cloud service providers at the same time. On the other hand, more recent attempts to unify cloud service providers and clients, such as Social Cloud [11], downplay the role of traditional cloud service providers and instead rely heavily on social networking components and services, but limit cloud service provision to those clients with whom a social networking link has already been established in the process. Such a limitation prevents Social Cloud from being a truly open marketplace to everyone, regardless of their social networking status. Consequently, a cloud paradigm that could benefit from the principles and fundamentals of crowdsourcing could be an advance in the cloud marketplace.

3 The paradigm of crowdsourced systems

The notion of crowdsourced systems is a relatively new one, which is different from the notion of crowdsourcing platforms. In this section, we will present the paradigm of crowdsourced systems by elaborating on the characteristics and high-level architecture for crowdsourced systems.

3.1 The rise of crowdsourced systems

The high adoption rates of truly portable smart devices (e.g., smart phones and smart watches) by the general public, as well as the emergence of Do-It-Yourself computing platforms [49] (e.g., Raspberry Pis and Arduinos) have paved the way for new paradigms of computing and networking with strong distributed and ad-hoc characteristics. Both of these classes of devices are highly affordable and are supported by appropriate development tools that enable the public to use them in developing applications and systems with relative ease. When such applications and systems are designed to be *open*, then crowdsourcing methods can be employed for them to grow and scale.

One real-life example of such a project is Safecast [57], which was developed after the devastating earthquake and tsunami which struck eastern Japan in 2011, and the subsequent meltdown of the Fukushima Daiichi Nuclear Power Plant. Safecast enabled citizens to build their own DIY radiation monitoring sensor kits by employing open source and open data methodologies. Essentially, the crowd was enabled to monitor, collect, and openly share information on environmental radiation and other pollutants that was then aggregated and visualised in radiation maps.

The same rationale spawned FLOAT [20], in which citizens of Beijing were able to generate their own data on air quality in the city using sensor kits attached on kites. In this case, the design of the kits was also crowdsourced, thus carrying strong elements of co-creation as well.

Apart from such ad-hoc examples, that emerged somehow spontaneously, *Mobile Crowdsensing Systems* (MCSs) [22] have been thoroughly studied in the research community. The main technological enablers for MCS are smart phones; truly portable and personal devices equipped with a variety of sensors which are able to support several communication interfaces. MCS seek to exploit these characteristics by orchestrating the collaborative operation of multiple smart phones towards performing a task. Tasks in the context of MCS mainly focus on collecting sensory data and therefore *crowdsensing systems* can be regarded as distributed sensing infrastructures whose sensing points are crowdsourced.

By extending this line of thought, one could identify systems that also employ other types of devices (e.g., DIY computing platforms) and that focus on other application areas than collecting sensory data (e.g., sharing computing power or storage space). This is the core idea that underpins the definition of a new system paradigm; namely that of *crowdsourced systems*.

3.2 Characteristics and high level architecture

Crowdsourced systems are systems whose constituent infrastructure is pooled or augmented via crowdsourcing methods. A clear distinction needs to be made between crowdsourced systems and crowdsourcing systems, also known as crowdsourcing platforms. Crowdsourcing systems or platforms act as tools enabling or facilitating the execution of a task via crowdsourcing. For instance, they may act as the gateway to the crowd (e.g., a web service individuals use) or provide supporting mechanisms such as a directory of active contributors or aggregation mechanisms. On the other hand, crowdsourced systems are themselves created or heavily rely on infrastructure that is crowdsourced. For instance, in [6] a smart building equipped with ambient luminance sensors is able to opportunistically augment its sensing infrastructure (and therefore improve the quality of service to the end users) by employing the embedded sensory capabilities of the smart phones of the end users. This also demonstrates the advantages of crowdsourcing in developing scalable systems in a cost effective way.

While a crowdsourced system may be developed with the aim of performing a particular task (such as in the case of Safecast and FLOAT), in the general case a crowdsourced system should be application agnostic. In order to characterise the "canonical" crowdsourced system—and therefore define the corresponding system paradigm—we

revisit the prerequisites of the "wise" crowd as those are specified by Surowiecki (see Sect. 2.1).

- *Diversity* a canonical (i.e., application agnostic) crowdsourced system needs to rely on diverse infrastructure and therefore to be able to crowdsource heterogeneous devices. It should not rely on specialised hardware that is application specific.
- *Independence* each crowdsourced device should be autonomous and non-reliable on other system components for its operation. The crowdsourced infrastructure should not include central nodes or dependencies among the devices (e.g., gateways). However, ephemeral connections can be provisioned.
- *Decentralisation* no central mechanism should exist that dictates the operation of the individual crowdsourced devices. However, orchestrating mechanisms that supervise the task execution and manage the crowd as a whole (e.g., micro-payments or other incentive mechanisms) may exist.
- *Aggregation* an aggregation mechanism should be defined that consolidates the individual contributions towards executing the task. Such a mechanism may be centrally operated by the task issuer or may be intrinsic to the crowdsourced system (e.g., in the form of a distributed protocol).

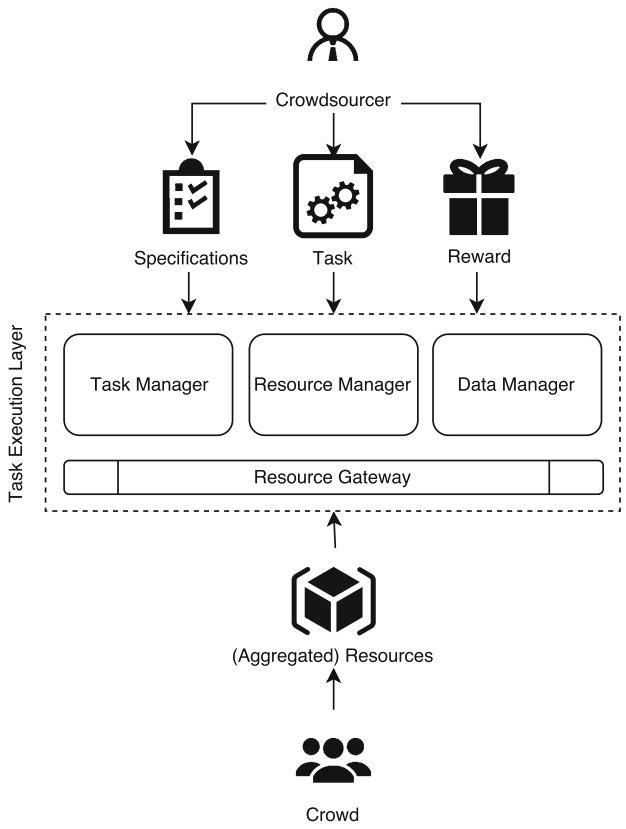

Fig. 1 High-level architecture of a crowdsourced system

In light of the above discussion, Fig. 1 depicts the high-level architecture of a canonical crowdsourced system.

In general, a crowdsourced system has the purpose of performing a task and its architecture consists of three layers; the *Crowd layer*, the *Crowdsourcer layer* and the *Task Execution layer*. The base of a crowdsourced system is the crowd that contributes to the system by providing the infrastructure that is necessary for performing the task; these are referred to as crowdsourced resources. The crowdsourced resources are then employed by the crowdsourced system in order to provide data and/or services. On the top of the architecture lies the crowdsourcer, who is the key beneficiary of the operation of the crowdsourced system. The crowdsourcer issues the task to be executed and may also provide additional specifications related to the task, such as specifications for the incentive mechanisms to be employed or the available budget. Note that the crowdsourcer is identified as the issuer of the task; a role that may or may not be assumed by an individual of the crowd, depending on the context of operation of the system.

The Task Execution layer lies in between the Crowd layer and the Crowdsourcer layer, providing the necessary corresponding abstraction mechanisms. The Resources Gateway module provides connectivity between each individual resource and the Task Execution layer in such a way that the heterogeneity of the crowd is hidden from the upper layers. The Task Manager module provides the interface between the Task Execution layer (and subsequently the Crowd layer) and the Crowdsourcer layer. It enables the crowdsourcer to issue the task to the crowd while remaining agnostic of the complexity and the potential diversity of the underlying mechanisms. Towards facilitating the task execution, this layer also provides core management mechanisms. The Resource Manager module supports the curation of the crowdsourced resources and therefore of the crowd. The mechanisms embedded in this module may include (but not be limited to) maintaining a resource directory, implementing the incentive mechanisms, maintaining the corresponding ledgers for managing the available budget for incentives, and so on. Similarly, the Data Manager module supports the curation of the collected data by providing services related to normalising the received raw data according to a data model, mining the data and semantically annotating them, data aggregation services, etc.

In this work, we only provide a high level presentation of the architecture for a canonical crowdsourced system. There exist several other aspects that need to be addressed, such as trust and privacy issues, preserving anonymity for the crowd, the efficient use of incentive mechanisms for efficient task execution, the use of open interfaces and open data in crowdsourced systems, and so on. These will be

elaborated in our future work. In the following section, we will compare the notion of crowdsourced systems to other already-existing paradigms.

4 Comparison to other paradigms

4.1 Comparison to cloud-based paradigms

As already mentioned, the paradigm of crowdsourced systems is underpinned by the high acceptance rates of DIY computer platforms [49], i.e., affordable hardware platforms that can be used by amateurs and the general public to develop small ICT projects. Certain DIY computer platforms, such as some flavours of the Raspberry Pi, while being affordable are also characterised by significant computational resources and communication capabilities. As depicted in Fig. 1, the paradigm of crowdsourced systems provisions the federation of several such devices towards a distributed system that provides services and infrastructure to a set of users.

While this may seem similar to or a special case of cloud-based paradigms, such as the Intercloud [26], crowdsourced systems carry unique characteristics and pose distinct challenges that justify their identification as a new paradigm. In particular, crowdsourced systems differ in at least the following ways.

Infrastructure and services provision and management in cloud-based systems the cloud provider provisions and provides access to the ICT infrastructure and the services that run on top of it. This means that cloud systems are centralised systems in the sense that they are centrally managed. This, of course, does not preclude the adoption of distributed architectures in the way the hardware and the services are deployed and managed, but the management and the provision of those is carried out centrally by the cloud provider (e.g., consider the Amazon Web Services). Cloud paradigms like Intercloud [26] consider the federation of multiple individual clouds in the context of distributed architectures. Such federations allow individual clouds to share and exchange data, and to distribute loads among them, overall providing improved services to their end users in terms of availability, scalability and elasticity. Crowdsourced systems radically differ as their infrastructure is not provided by a centralised provider but is pooled from the crowd consisting of independent participants. This means that the infrastructure of a crowdsourced system is not centrally managed and therefore it is characterised by heterogeneity and uncertainty regarding its availability. Furthermore, a crowdsourced system is not a federation of individual, stand-alone clouds but a system whose constituent infrastructure is crowdsourced by contributing individuals. One way to regard it is that such systems follow similar organisation principles to cooperative initiatives where the provider is also a consumer, i.e., a "prosumer" [56].

Functional and technical requirements crowdsourced systems have radically different functional and technical requirements to other paradigms that are cloud-based. Clouds are typically designed to accommodate the needs of large numbers of end users or applications processing large volumes of data. In this respect, cloud-based systems are specifically engineered with high-end technical requirements such as those for High Performance Computing or Big Data. On the other hand, crowdsourced systems are not specifically engineered systems since their infrastructure is crowdsourced. This also implies that this infrastructure is typically characterised by constraints in terms of computational capabilities, resources (e.g., memory), and availability. The latter is due to the fact that the ownership (and therefore the control) of the crowdsourced components belongs and remains to the contributing individual(s). Finally, contrary to the cloud, crowdsourced systems lie on the edge of the network, close to the end user.

The human factor as mentioned above, crowdsourced systems pool their infrastructure via means of crowdsourcing from the general public. One direct implication is finding the answer to why someone should contribute to the crowdsourced system. Therefore, a crowdsourced system should incorporate an incentive mechanism, either implicitly (e.g., the contributors benefit from the operation of the system itself) or explicitly (e.g., monetary incentives via micro-payments). Another important aspect is the fact that DIY platforms [49] (e.g., the Raspberry Pi)—that are key enablers of crowdsourced systems—are commonly used in small scale projects by amateurs and are deployed in sensitive premises such as homes and work environments. This means that aspects such as anonymity, privacy, and trust lie in the core of crowdsourced systems. Overall, due to their nature, the human factor and the challenges it poses are more profound in crowdsourced systems than in cloud-based systems.

4.2 Comparison to crowdsourcing systems

Crowdsourcing systems and crowdsourced systems both rely on contributions from the general public (i.e., the crowd). However, in spite of the similarity of the terms, crowdsourced systems clearly differentiate from crowdsourcing systems. Crowdsourcing systems typically refer to digital platforms that collect and consolidate input from the crowd. In this case the input may come in several forms; a recommendation or a review, a vote, a video, sharing of location, or sensory data in the context of a crowdsensing application. A crowdsourcing platform may or may not be centralised or distributed in terms of system architecture.

Most often, however, the platform is deployed and centrally managed. On the other hand, in crowdsourced systems the general public contributes to the system itself. Individuals do not only provide input to the system but also provide the means for collecting, processing, and curating data and information.

4.3 Comparison to IoT as a Service

As already mentioned, one of the key enabling technologies for the paradigm of crowdsourced systems is the DIY computer platforms such as Raspberry Pi and Arduino. Such platforms are also regarded as key enablers for the IoTs, mainly due to their small size and their use in small automation projects. IoT as a Service is a system paradigm in which smart devices are interconnected with a remote cloud infrastructure via which their functionalities (e.g., sensor measurements) are made available [10]. Here, although the IoT devices may be deployed in several different areas (thus allowing such systems to be characterised as distributed), the core of the system remains centralised—at least from a management perspective—as it is based on a cloud provider. This is a fundamentally different approach to crowdsourced systems, where the components of the system itself are crowdsourced. Also, the scope of a crowdsourced system is much broader as it is not restricted in IoT applications.

5 Crowdcloud: an instance of crowdsourced systems

In this section, we will introduce and present crowdcloud as an instance of crowdsourced systems. In crowdcloud, the crowd follow the principles of the free market and supply their services on the cloud while also demanding for other crowd members' cloud services.

5.1 Foundation of crowdcloud

Crowdcloud [28] refers to the provision of computing services at different levels of IaaS, PaaS, and SaaS by the crowd and for the crowd. The crowd, in this definition, can include both individuals and organisations. Crowdcloud acts like an online free market where every individual and every organisation can supply their resources or demand for other crowd members' resources, following the regulations of the free market. The idea of crowdcloud applies several features of crowdsourcing such as largeness, diversity, and incentives provision [29] in the cloud. Crowdcloud also builds upon the notion of crowdsourced systems in terms of architecture and application. Crowdcloud lets the crowd provide their idle resources to other individuals or organisations in the crowd, and also request their required resources from other crowd members. This can happen at the infrastructure level (i.e., IaaS), e.g., by providing or asking for CPU power and storage space, at the platform level (i.e., PaaS), e.g., by providing or asking for runtime libraries and web servers, or at the software level (i.e., SaaS), e.g., by providing or asking for email applications and on-demand software systems.

Ordinary cloud services, such as Amazon EC2 or Google Drive, are cloud services which are provided by organisations for other organisations or people. These cloud services come with a legally binding contract between the cloud service provider and the cloud service client and are mostly costly for other organisations, but they are usually free or inexpensive for individuals to use. In some cases where provided cloud services are free of charge, organisations usually compensate for the costs by introducing advertisements along with their free cloud services. Furthermore, these services are generally managed in a centralised way, and this has instigated issues related to data control and privacy [74] as well as legal issues [17]. Crowdcloud, on the other hand, is fully decentralised, provides cloud services from the crowd to the crowd, and can be contract-free.

Crowdcloud bears some similarities with a few concepts in the literature, such as *cloudsourcing*, *volunteer cloud computing*, and *social cloud*. However, the differences between crowdcloud and these concepts render crowdcloud as a novel idea and make crowdcloud stand out as an entirely free market model for cloud service provision. These differences will be discussed in detail in the following paragraphs.

Cloudsourcing refers to outsourcing various elements of a business or organisation IT infrastructure to other companies or organisations which provide such services in the cloud [23, 41]. Therefore, the first difference between cloudsourcing and crowdcloud lies both in the cloud service providers and cloud service clients. In cloudsourcing, organisations provide some cloud services to other organisations. In crowdcloud, however, the crowd provide some cloud services to the crowd. The second difference between the two is that cloudsourcing is centralised while crowdcloud is not. The last difference between cloudsourcing and crowdcloud is that cloudsourcing is always based on a contract between two organisations, while crowdcloud may or may not be contract-based.

Volunteer cloud computing, also known as peer-to-peer computing or global computing, refers to the use of computers volunteered by the general public for distributed scientific computations [3, 15]. SETI@home and BOINC are examples of volunteer cloud computing. In this case, and apart from its aforementioned purpose, two main differences exist between volunteer cloud computing and

crowdcloud. The first difference is that in volunteer cloud computing, the crowd provides a service, such as CPU power or storage, solely for organisations (and normally for research purposes) and not for other people. The second difference is that volunteer cloud computing, unlike ordinary cloud services, is not based on a contract and people have no obligations whatsoever to provide cloud services or keep providing cloud services to their beneficiaries. Crowdcloud, however, can be both contract-free and contract-based.

Social cloud is probably the closest in meaning and application to crowdcloud. Social cloud refers to a framework for sharing resources and services based on relationships amongst the members of a social network [12]. The notion of social cloud implies three ideas that form the differences between social cloud and crowdcloud. The first difference is that social cloud depends on a social network and relationships amongst the members of that social network. This limits the cloud service provision to socially connected members within the social network. Crowdcloud, on the other hand, is not necessarily a social network, and can exist independently as an online free market for cloud services. This provides a wider range of services to acquaintances and non-acquaintances alike. The second difference is that social cloud works solely on social contracts, while crowdcloud can work on legally binding contracts or be contract-free. Finally, social cloud explicitly limits the use of each individual's resources to other individuals. Crowdcloud, on the other hand, is open to both individuals and organisations, for-profit or non-profit, for the use of resources.

The differences between crowdcloud and other similar cloud services are shown in Table 1. These differences include who the service providers and service clients are, how these platforms are managed, and whether a contract is needed between cloud service providers and cloud service clients for the use of cloud services.

5.2 Crowdcloud architecture

The proposed architecture for crowdcloud is presented in this section along with a short description of its constituents. It is depicted in Fig. 2.

As Fig. 2 illustrates, crowdcloud has the following three constituents: the cloud, the crowd, and the crowdcloud platform. The crowdcloud platform utilises three distinct modules which interact with each other. These modules are explained below:

- *Crowd management module* this module is responsible for managing the crowd and their interactions. In particular, this module manages crowd members' registration, records their service agreements, handles availability of service contracts (if any), and facilitates interactions amongst cloud service providers and cloud service clients in the crowd. The crowd management module in crowdcloud can be mapped to the task manager module and data manager module in crowdsourced systems architecture.
- *Cloud management module* this module is responsible for managing the resources provided by the crowd in the cloud. In particular, this module records the list of all available cloud resources, documents their providers and clients, keeps track of cloud resources availability status, and manages cloud resource allocation. The cloud management module in crowdcloud can be mapped to the resource manager module and resource gateway module in crowdsourced systems architecture.
- *Platform management module* this module is responsible for managing the crowdcloud platform. In particular, it serves as the interaction gateway between the other two modules and coordinates their functionalities. The platform management module in crowdcloud can be mapped to the overall architecture of the crowdsourced systems.

5.3 Advantages and challenges of crowdcloud

In this section, some of the characteristics and advantages of crowdcloud are presented. In the same fashion, some

Table 1 Differences between cloud paradigms

Cloud service	Service providers	Service clients	Management	Contract
Ordinary cloud services	Organisations	People or organisations	Centralised	Yes
Cloudsourcing	Organisations	Organisations	Centralised	Yes
Volunteer cloud computing	People	Organisations	Centralised	No
Social cloud	People	People	Decentralised	No
Crowdcloud	People or organisations	People or organisations	Decentralised	Yes/no

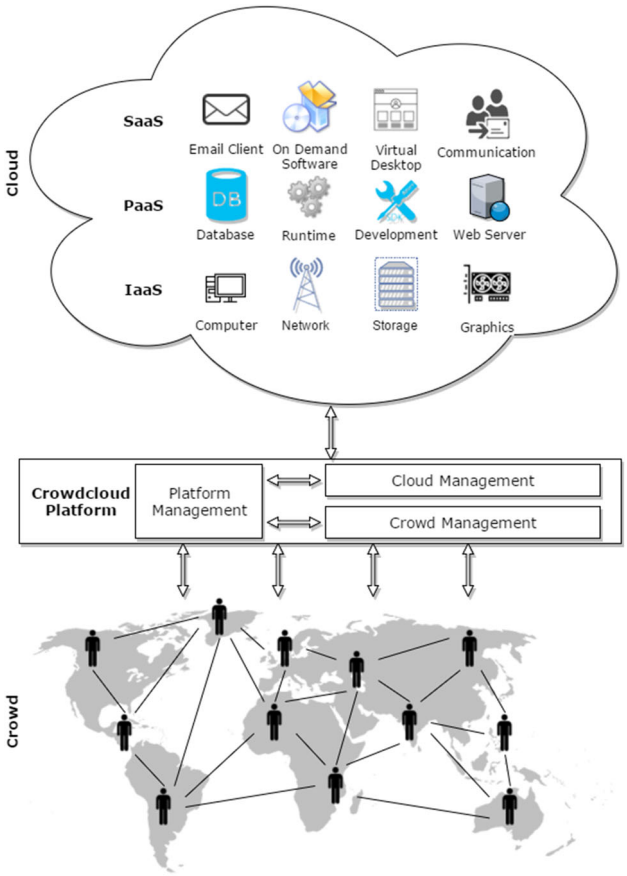

Fig. 2 The proposed architecture for crowdcloud; Notice that each crowd member will have their own local version of "Crowdcloud Platform", which means that "Crowdcloud Platform" is fully decentralised

challenges that the implementation of crowdcloud can introduce are discussed and possible solutions to avoid or mitigate them are presented. It should be noted that crowdcloud, as a novel concept, will need more theoretical research to be conducted before any implementation attempts are made in order to guarantee a quality service which addresses all benefits and possible challenges accordingly.

5.3.1 Crowdcloud characteristics and benefits

Crowdcloud brings about a set of features and advantages that can be exploited to the benefit of the crowd, as well as organisations, and which can differentiate crowdcloud from other cloud services already in existence and give it a competitive advantage. These features include, but are not limited to, the following features.

Decentralised resource management resource management, as a crucial factor in cloud computing [66], is usually performed by organisations in ordinary cloud services and in cloudsourcing, and is therefore conducted in a centralised way, i.e., they are centrally managed by one organisation. In volunteer cloud computing the resources are distributed inherently, but their management is still usually centralised by the service clients, i.e., the service client decides when and how to use the resources.

Crowdcloud, on the other hand, is completely decentralised in its resource management, meaning that each cloud service provider (i.e., an individual or an organisation) in the crowdcloud environment is responsible for managing the cloud resources they have provided, and there is no central authority to manage all the provided resources on a crowdcloud platform. The cloud management module on the crowdcloud platform is not a central entity either, but a distributed one where every crowd member will manage their own cloud resources through their own local copy, while interacting with other crowdcloud platforms over the network for coordination and interaction purposes.

Bidirectional service exchange in ordinary cloud service providers, cloudsourcing and volunteer cloud computing, organisations either provide services (e.g., Google Drive provides storage space for its clients) or they request services (e.g., SETI@home requests CPU power to analyse radio telescope data). This means that service provision in these cloud environments in unidirectional.

Crowdcloud, similar to social cloud, provides the opportunity for the ordinary crowd to both provide services and request them, thus providing a bidirectional service exchange. For example, a crowd member may provide a storage space for another crowd member while requesting a specific software program from the same or different member. The key difference, however, between social cloud and crowdcloud is that crowdcloud is not restricted to individuals, and organisations can also play the role of single entities in providing and requesting cloud services.

Democratised service provision when cloud service providers are corporations and organisations (for-profit or non-profit), they are fundamentally the ones who set the trends, determine the prices, obligate terms of services, etc. This means that the crowd will have no say and no control over these domains and have to abide by the rules set for them.

On crowdcloud, however, and similar to social cloud, everything is determined by the people. It is a free market where it is people who decide, possibly in a democratic way, which services to provide and how these services should be priced, how quality of service should be ensured, etc. Furthermore, it is possible for the crowd members to bargain over prices, terms and conditions, etc. This means that, for example, you may even find free virtual desktops or free on-demand software services for your needs. Here, the difference between crowdcloud and social cloud is that social cloud is based on a social network and relations

amongst people, which limits the provision of services to a pre-defined list of friends, or friends of friends, etc. In crowdcloud, however, resource requests can originate from anybody in the crowd towards anybody in the crowd, whether such requests are accepted or not.

Pricing using several existing cloud services usually incurs a price on the clients, especially when the client is an organisation. Free cloud services do exist, but they also usually come with a set of limitations, such as bandwidth or data volume limitations, or with unwanted, usually obtrusive advertisements. However, paying for cloud services is justified because of the costs of running, maintenance, upgrade, personnel, etc.

Crowdcloud offers the possibility of getting cloud services either for free or for a nominal cost. In social cloud, this possibility also exists, but only for a certain group of people in an individual's social network, e.g., one's friends or friends of friends. In crowdcloud, on the other hand, people may have different motivations to provide free or inexpensive cloud services, friendship only being one of them. Other reasons might include a mutual agreement to use each other's resources, getting social incentives such as a better visibility in an online forum, or simply as an act of altruism and helping towards a noble cause, such as providing one's resources for global awareness about a certain topic. While it is acknowledged that this may not be a noticeable difference between crowdcloud and other cloud services, the possibility of tapping into an overwhelmingly large and possibly free pool of cloud services is still an advantage of crowdcloud.

Free market model crowdcloud follows a free market model where the crowd provide and request cloud services. In this free market model, the crowd can determine the revenues and the costs, they can bid for services, and they can exchange one service for another. The crowd can cooperate on service provision or compete to receive them, and in the long run, the crowd will learn from past experiences and adapt to new emerging situations. Last but not least, the crowd will self-organise their interactions, service provisions and requests, and their forthcoming challenges. This idea of free market is probably the most prominent feature of crowdcloud.

Flexibility one challenge to traditional clouds is the need for rapid provision of resources with low latency to support new and emerging resource-hungry (real-time) applications. The volatile workload requiring constant auto-scaling and load balancing in the cloud is posing a real engineering challenge to current providers [69]. Various effort in decentralising computing resources have been proposed in the literature (e.g., [58, 63, 70, 73]). Crowdcloud presents itself as a potential solution to this as resources can be sourced from nearby incentivised participants and/or organisations rather than relying on resources statically provisioned at some central data centres which may incur long response time. By sourcing the resources nearby, crowdcloud reduces the latency as well as allowing the flexible adaptation of resources based on workload. As such, we see crowdcloud as an enabler to such edge/fog cloud technologies in providing seamless services under uncertainties. Note that this is different from forming a federated cloud or interlinking different clouds with multiple cloud providers enter into agreements [with or without contractual enforcement for example via Service Level Agreements (SLAs)] as we are focusing on sourcing sufficient local resources on demand. We refer readers to [25] for different forms of such cloud federation.

5.3.2 Challenges in crowdcloud

While the idea of crowdcloud can potentially offer several advantages to the clients of cloud services, it amplifies several already existing challenges in cloud service provisioning that should be addressed and introduces new challenges to this paradigm as well. Without appropriate consideration of these challenges, a successful, useful implementation of crowdcloud cannot be guaranteed. Below, these major challenges are discussed and possible solutions are proposed.

Availability one issue that should be addressed in crowdcloud is availability. In ordinary cloud services and cloudsourcing, a certain percentage of availability is guaranteed for cloud services clients [1]. Although sometimes organisations fail to hold onto their promises about the level of availability [64], clients can generally rely on these promises, and they usually get compensated in situations where availability is compromised.

Such availability promises are often absent and cannot be guaranteed in a crowdcloud environment. When service providers in crowdcloud provide their services in an ad-hoc manner with no contractual bindings, they may withdraw from service provision without proper notice at any time. But even with contracts made between cloud service providers and clients in crowdcloud, the nature of crowdcloud still overshadows its availability. While this may not be a big issue in some instances such as CPU power, i.e., the client may connect to another client and use their CPU power, this can cause a huge problem in some other instances such as cloud storage, i.e., if a client becomes unavailable while providing cloud storage, all files on their storage device will be unavailable to their client during that period of time.

To overcome availability issues, a number of solutions can be adopted. One possible solution, when applicable, can be redundancy, i.e., a client may use several similar cloud services in crowdcloud to guarantee their desired level of availability for that certain service, e.g., a client

may copy their files on several storage services just to make sure they will never lose the availability of their data. Another possible solution is for service providers and service clients to sign a binding contract, negotiating and detailing the level of service availability. Another possible partial solution is the trust-based solution, i.e., people will less likely stop providing their cloud service to their friends without at least notifying them about it in a reasonable time to let them find alternative cloud solutions. Reputation systems can also be put in practice to make clients aware of those cloud service providers who have gained more reputation based on their service provision availability.

Security another prominent issue that cloud clients may encounter in crowdcloud is the security. Ordinary cloud services have their own security issues [42], but it can get much aggravated in crowdcloud. When well-established, reputable organisations provide cloud services, their clients generally trust them as they request resources from them. If any security breaches occur, clients are usually assured that certain measures will be taken to both reduce the adverse effects and to ensure that the possibilities of such breaches in the future are minimised.

On the other hand, this is not the case on a crowdcloud platform. Given that resources are provided by ordinary people, clients cannot be guaranteed to benefit from any security measures if and when security breaches occur. Furthermore, trusting cloud service providers or clients is also an issue when they are individuals rather than organisations, especially in the absence of a legally binding contract. For example, providing CPU power as a resource to another individual could raise the possibility of one's system being hacked, inducing harm to the service provider. Similarly, receiving storage space as a resource from another individual could also raise the possibility of ones' personal information being misused, inducing harm to the service client.

It should be noted that even big organisations sometimes fail to take good care of their clients' security, leading to many research into security issues in the cloud [61]. Furthermore, in a crowdcloud platform, a trust-based system can be formulated between every service provider and service client as a possible solution to security issues. For example, clients should be able to share certain resources only with certain people, e.g., only with their friends or with their friends and their friends of friends, as in the case of social cloud. Furthermore, leaving a certain degree of responsibility of ensuring security on its stakeholders is not a new idea and many cloud service providers are already practising this. For example, it is up to the Google Drive client to decide which files to share and with whom. While the consequences of sharing infrastructure resources with the wrong people is probably more dire, it is still observed as a good practice to leave it for the people to decide on it and enforce it when and if necessary.

Privacy given the nature of crowdcloud, which is providing cloud services by the ordinary crowd to the ordinary crowd, privacy issues constitute an instant threat. In ordinary cloud computing, cloudsourcing and volunteer cloud computing, organisations have a data security policy that ensures data privacy and enforces clients' data protection. Even with big organisations and cloud service providers, privacy always remains an issue in the cloud [39] and it makes big news in the media every now and again [51], but it can get even more exacerbated in the case of crowdcloud.

Cloud service providers in crowdcloud usually comprise of ordinary individuals, whose locations might sometimes be unknown, and whose local privacy regulations may differ significantly from privacy regulations in the countries where cloud service clients reside. Then it is also the issue of malicious service providers, which arises in many contexts where the crowd is given authority or responsibility [46, 67]. Combined together, these issues can form significant threats to clients' privacy.

There are a number of measures to take in order to minimise clients' privacy breaches in crowdcloud. Reputation systems help the clients understand which cloud service providers are well-famed or ill-famed, and request for their services accordingly. Trust-based solutions also help the clients and providers in determining where their data can be stored, who can use the CPU power, etc. User-driven privacy enforcement [27] is another solution that can help increase privacy in crowdcloud.

Legal issues providing cloud services at the infrastructure level or platform level may pose legal threats, but providing SaaS probably introduces the majority of legal issues in crowdcloud. As for organisations, they usually provide SaaS when they own the rights to the software or to the provision of the SaaS. In this case, the organisation will have all the rights to determine how to distribute the software and to determine pricing mechanisms. Apart from this, organisations usually provide their clients with SLAs which usually clearly define each party's rights and duties, which can later be referred to when disagreements arise between service providers and service clients [53].

The ordinary crowd, on the other hand, may own the software they have on their systems through the purchase of that software, but they usually do not have the rights to redistribute the software or provide it as a service to other people. Furthermore, there are usually no contracts or SLAs between service providers and service clients in crowdcloud, making it difficult to settle disagreements and legal disputes if and when legal issues arise.

In order to resolve general legal issues in crowdcloud, it seems necessary that service providers and service clients should be required by the crowdcloud platform to agree

with certain rights and responsibilities before committing to any service provision or initiating any service request. However, providing SaaS will be impossible for several software systems under copyright laws with current legislation. Providing other forms of SaaS, such as freeware software, open source software, and software under copyleft laws, may not pose legal threats.

Transparency transparency is an emerging requirement of the people in the age of information technology [36]. With so many Web-enabled devices and people's personal information being constantly transmitted over the Web, transparency is becoming increasingly important. It should be noted that people generally understand and accept the fact that their personal information needs to be accessed by Web entities, such as Web merchants and business Websites, in order for them to receive tailored services and products. What these Web entities fail to provide to the people, however, is how (and sometimes why) this information is being handled. For example, a lot of people would like to know why a certain product appears in the advertising panel of their e-mail provider or their search engine when they just recently searched for the same or similar product in a totally e-commerce platform.

In crowdcloud, transparency requirements become more desirable as cloud service providers and clients can be ordinary people about whom little or no information might be available. Furthermore, transparency about the details of cloud service provision or request can play an important role in crowd's engagement with crowdcloud. For example, someone requesting CPU power may state *why* they need to use somebody else's CPU and *how* they intend to use it, or someone offering cloud storage may explain *how* they ensure the privacy and security of the stored information on their storage device. Engineering approaches towards transparency [31, 34, 35] can significantly help in formalising and formulating such requirements, and the power of the crowd engaged in crowdcloud can be harnessed to meet such transparency requirements [33, 37].

Retention a significant issue in crowd-based systems is that, in the absence of an appropriate type of motivation, crowd members may lose their interest in actively participating in crowdsourced activities. Crowdcloud is no exception in this case and will require an incentive model for the engagement and retention of the crowd in providing cloud services.

The problem of crowd engagement and retention in crowdcloud can be circumvented in several ways. First, crowd members' active participation can be encouraged through the free market model, where crowd members can fulfil their cloud requirements through a supply and demand model. That is to say, crowd members will need to stay engaged and active on the crowdcloud if they want to maintain their access to cloud services they require.

Second, and similar to other crowd-based systems and platforms [16, 59], gamification can be a plausible method of motivating and retaining crowd engagement through points, badges, and leaderboards [68]. For instance, crowd members can be incentivised for each cloud service provision through points, and they can appear on leaderboards based on the most positive feedback they get from their clients. When carefully engineered and implemented, such immersion in gamification can cause crowd members to fully engage in crowdcloud, while it also makes them commit to higher quality cloud service provision for more points, more badges, and higher ranks in leaderboards.

6 Ongoing and future work: a crowdcloud implementation for cluster computing

In this section, we present an ongoing implementation of a framework for crowdsourced cluster computing, which follows the architecture and design guidelines of the crowdcloud. The proposed implementation delivers a crowdsourced application execution environment, which is able to leverage computing resources even from resource-constrained devices in an incentivised manner which rewards participation.

We use Apache Spark [60] as the core software enabler for this implementation; Spark is a general-purpose cluster computing system which supports the MapReduce programming model and achieves great speed and scalability thanks to in-memory data abstractions [71]. It provides high-level APIs in some of the most widely used programming languages, namely Java, Scala, Python, and R, as well as a rich set of built-in tools including Spark SQL for structured data processing, MLlib for machine learning, and Spark Streaming for handling live data streams. Figure 3 depicts the functional architecture of a Spark cluster; Spark applications run as independent sets of processes, coordinated by the SparkContext object on the driver program in a master/slave configuration. SparkContext can connect to several types of cluster managers which allocate resources across applications. Once connected, the cluster

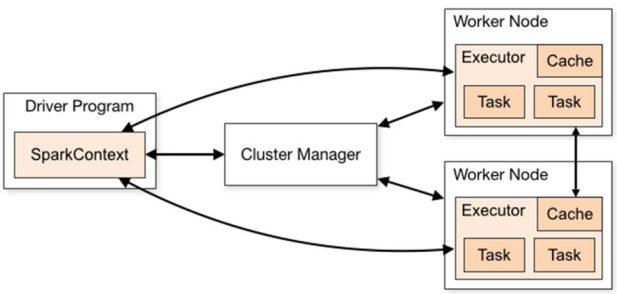

Fig. 3 Spark cluster overview

manager acquires executors on nodes in the cluster, which are processes that run computations and store data for applications.

Following the crowdcloud architecture described in Sect. 5.2, the cloud management module is represented by the native Spark standalone cluster manager which we use in this prototype. Other open source resource managers such as Mesos or YARN could be used equivalently for this crowdsourcing implementation. We choose to use the standalone mode as it is the simplest one to set up when not using an existing Hadoop or Mesos cluster. To expose their resources on the cluster, users have simply to mount the spark-worker image on their device and register in the crowdsourcing platform. Once this is done, the cloud manager can retrieve part of the device's computing cores according to the established plan and policy.

Metadata regarding the crowd participants, such as the number of cores, available memory, etc., will be communicated from the cluster manager to the crowd management module, which is where all the marketplace and incentivisation functions will be encapsulated. Implementation of the crowd management module is still part of our future work and so far different open source tools, such as HIVE [65] and PYBOSSA [54], are being evaluated for its deployment. The cloud and crowd management modules will act as two discrete entities orchestrated by a higher-layer of management which will compose the platform management module, where the various service layer functions will take place.

Diving deeper into the crowd layer requirements for this use-case, a critical factor regarding the viability of the proposed framework is the capability of the cluster manager to leverage computing resources in small chunks from many peers; crowd participants usually do not possess the powerful machines which are traditionally used in cluster computing and even in that case the total deprivation of their computing resources is not desired. Therefore, efficient ways to adjust existing cluster computing methodologies, which are based on resource-rich fully available machines, or develop novel ones, will be an imposed research challenge.

In order to test and validate the proposed framework with regard to the above requirement, we deployed a local Spark cluster of Raspberry Pi devices (Fig. 4), in the context of Syndesi Testbed [18] at the University of Geneva. Syndesi is an IoT framework and testbed which includes a wireless sensor network and mobile crowdsensing modules interconnected via a gateway server. For this cluster we chose the Raspberry Pi 3 Model B version which possesses a quad-core processor at 1.2 GHz and 1 GB of RAM, specifications which fall into the range of devices crowd participants may own. This prototype aims to serve as a proof of concept for the computing challenges

Fig. 4 Raspberry Pi Spark cluster at University of Geneva. Master node (bottom right) and worker nodes (three stacks of four) connected via an ethernet switch in a local network configuration

imposed in a crowdsourced cluster computing framework and networking constraints are not taken into consideration in this ethernet-based configuration. That being said, with the recent technological advances in mobile networks, such as the emergence of 5G generation networks which will provide a throughput of order of Gbit/s, we believe that related issues will be easily tackled.

Providing the reader with a disclaimer of this being ongoing work, initial testing of the Spark computing framework on the Raspberry Pi cluster indicates that efficient crowdsourced computing with resource-constrained devices is possible. Nevertheless, an initial stage of calibration and configuration of the system parameters is undoubtedly required to obtain the desired behaviour.

7 Conclusion

In this work we introduced the new paradigm of *Crowdsourced Systems*; systems whose constituent infrastructure is pooled from the general public. While heavily relying on crowdsourcing, crowdsourced systems are not to be confused with crowdsourcing platforms. While the latter act as tools or mediators in order to access the crowd and consolidate its input, crowdsourced systems rely on crowd contributions to pool and augment their infrastructure. We also introduced *Crowdcloud* as an instance of the crowdsourced systems paradigm. Essentially, crowdcloud is a crowdsourced cloud infrastructure. We also discussed and compared the particular characteristics of these new concepts and how they are different to already existing ones, such as crowdsourcing platforms and the cloud. Finally, we gave a brief overview of how crowdcloud, as an example of a crowdsourced system, can be implemented by presenting some of our ongoing work on crowdsourced cluster computing with Raspberry Pis.

The concept of enabling a crowd not only to provide a service or perform a task, but to co-create and pool the technical means to do so is a disruptive one. It directly correlates to and firmly underpins recent advances in fog/edge computing. Consider a neighbourhood being able to set up its own local data centre based on a community provided network, totally independent of any centralised control or dominating vendors. In our future work, we plan to demonstrate the use and the efficient performance of such crowdsourced systems by fully implementing a crowdsourced cluster computing framework.

Acknowledgements This work was partially funded by the Engineering and Physical Sciences Research Council (EPSRC) under the CHIST-ERA CONCERT (A Context-Adaptive Content Ecosystem Under Uncertainty), Project Number I1402.

References

1. Addis, B., Ardagna, D., Panicucci, B., Zhang, L.: Autonomic management of cloud service centers with availability guarantees. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 220–227. IEEE (2010)
2. Anderson, D.P.: BOINC: a system for public-resource computing and storage. In: Fifth IEEE/ACM International Workshop on Grid Computing, 2004. Proceedings, pp. 4–10. IEEE (2004)
3. Anderson, D., Fedak, G.: The computational and storage potential of volunteer computing. In: Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006. CCGRID 06, vol. 1, pp. 73–80 (2006). https://doi.org/10.1109/CCGRID.2006.101
4. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@ home: an experiment in public-resource computing. Commun. ACM **45**(11), 56–61 (2002)
5. Angelopoulos, C.M., Nikoletseas, S.E., Raptis, T.P., Rolim, J.D.P.: Characteristic utilities, join policies and efficient incentives in mobile crowdsensing systems. In: 2014 IFIP Wireless Days, WD 2014, Rio de Janeiro, Brazil, 12–14 November 2014, pp. 1–6 (2014)
6. Angelopoulos, C.M., Evangelatos, O., Nikoletseas, S.E., Raptis, T.P., Rolim, J.D.P., Veroutis, K.: A user-enabled testbed architecture with mobile crowdsensing support for smart, green buildings. In: 2015 IEEE International Conference on Communications, ICC 2015, London, UK, 8-12 June 2015, pp. 573–578 (2015)
7. Bestavros, A., Krieger, O.: Toward an open cloud marketplace: vision and first steps. IEEE Internet Comput. **18**(1), 72–77 (2014)
8. Brabham, D.C.: Moving the crowd at threadless: motivations for participation in a crowdsourcing application. Inf. Commun. Soc. **13**(8), 1122–1145 (2010)
9. Buyya, R., Ranjan, R., Calheiros, R.N.: Intercloud: utility-oriented federation of cloud computing environments for scaling of application services. In: International Conference on Algorithms and Architectures for Parallel Processing, pp. 13–31. Springer (2010)
10. Celesti, A., Fazio, M., Giacobbe, M., Puliafito, A., Villari, M.: Characterizing cloud federation in IoT. In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 93–98. IEEE (2016)
11. 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. IEEE (2010)
12. Chard, K., Bubendorfer, K., Caton, S., Rana, O.: Social cloud computing: a vision for socially motivated resource sharing. IEEE Trans. Serv. Comput. **5**(4), 551–563 (2012). https://doi.org/10.1109/TSC.2011.39
13. Ciurana, E.: Developing with Google App Engine. Apress (2009). https://doi.org/10.1007/978-1-4302-1832-6
14. Comi, A., Fotia, L., Messina, F., Rosaci, D., Sarné, G.M.: A partnership-based approach to improve QoS on federated computing infrastructures. Inf. Sci. **367**, 246–258 (2016)
15. Costa, F., Silva, L., Dahlin, M.: Volunteer cloud computing: MapReduce over the internet. In: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), pp. 1855–1862. IEEE (2011)
16. Dalpiaz, F., Snijders, R., Brinkkemper, S., Hosseini, M., Shahri, A., Ali, R.: Engaging the crowd of stakeholders in requirements engineering via gamification. In: Gamification, pp. 123–135. Springer, Cham (2017)
17. De Filippi, P., McCarthy, S.: Cloud computing: legal issues in centralized architectures. In: VII International Conference on Internet, Law and Politics (2011)
18. Evangelatos, O., Samarasinghe, K., Rolim, J.: Syndesi: a framework for creating personalized smart environments using wireless sensor networks. In: 2013 IEEE International Conference on Distributed Computing in Sensor Systems, pp. 325–330 (2013). https://doi.org/10.1109/DCOSS.2013.35
19. Fernandes, J., Nati, M., Loumis, N., Nikoletseas, S., Raptis, T.P., Krco, S., Rankov, A., Jokic, S., Angelopoulos, C.M., Ziegler, S.: Iot lab: towards co-design and IoT solution testing using the crowd. In: 2015 International Conference on Recent Advances in Internet of Things (RIoT), pp. 1–6. IEEE (2015)
20. FLOAT: http://civicus.org/thedatashift/wp-content/uploads/2015/07/Float-Beijing-case-study.pdf. Accessed 14 Nov 2017
21. Galton, F.: Vox populi (the wisdom of crowds). Nature **75**(7), 450–451 (1907)
22. Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Commun. Mag. **49**(11), 32–39 (2011)
23. Géczy, P., Izumi, N., Hasida, K.: Cloudsourcing: managing cloud adoption. Glob. J. Bus. Res. **6**, 57–70 (2011)
24. Gong, C., Liu, J., Zhang, Q., Chen, H., Gong, Z.: The characteristics of cloud computing. In: 2010 39th International Conference on Parallel Processing Workshops (ICPPW), pp. 275–279. IEEE (2010)
25. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. Softw. Pract. Exp. **44**, 369–390 (2012). https://doi.org/10.1002/spe.2168
26. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. Softw. Pract. Exp. **44**(3), 369–390 (2014)
27. Henze, M., Hermerschmidt, L., Kerpen, D., Häußling, R., Rumpe, B., Wehrle, K.: User-driven privacy enforcement for cloud-based services in the Internet of Things. In: 2014 International Conference on Future Internet of Things and Cloud (FiCloud), pp. 191–196. IEEE (2014)

28. Hosseini, M.: Crowdcloud: cloud of the crowd. In: The IEEE 5th International Conference on Future Internet of Things and Cloud. IEEE Computer Society (2017)
29. Hosseini, M., Phalp, K., Taylor, J., Ali, R.: The four pillars of crowdsourcing: a reference model. In: 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), pp. 1–12. IEEE (2014)
30. Hosseini, M., Moore, J., Almaliki, M., Shahri, A., Phalp, K., Ali, R.: Wisdom of the crowd within enterprises: practices and challenges. Comput. Netw. **90**, 121–132 (2015)
31. Hosseini, M., Shahri, A., Phalp, K., Ali, R.: Towards engineering transparency as a requirement in socio-technical systems. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp. 268–273. IEEE (2015)
32. Hosseini, M., Shahri, A., Phalp, K., Taylor, J., Ali, R.: Crowdsourcing: a taxonomy and systematic mapping study. Comput. Sci. Rev. **17**, 43–69 (2015)
33. Hosseini, M., Shahri, A., Phalp, K., Ali, R.: Crowdsourcing transparency requirements through structured feedback and social adaptation. In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), pp. 1–6. IEEE (2016)
34. Hosseini, M., Shahri, A., Phalp, K., Ali, R.: Foundations for transparency requirements engineering. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 225–231. Springer (2016)
35. Hosseini, M., Shahri, A., Phalp, K., Ali, R.: A modelling language for transparency requirements in business information systems. In: International Conference on Advanced Information Systems Engineering, pp. 239–254. Springer (2016)
36. Hosseini, M., Shahri, A., Phalp, K., Ali, R.: Four reference models for transparency requirements in information systems. Requir. Eng. **23**, 1–25 (2017)
37. Hosseini Moghaddam, S.M.: Engineering of transparency requirements in business information systems. PhD Thesis, Bournemouth University (2016)
38. Howe, J.: The rise of crowdsourcing. Wired Mag. **14**(6), 1–4 (2006)
39. Ion, I., Sachdeva, N., Kumaraguru, P., Čapkun, S.: Home is safer than the cloud!: privacy concerns for consumer cloud storage. In: Proceedings of the Seventh Symposium on Usable Privacy and Security, SOUPS '11, pp. 13:1–13:20. ACM, New York (2011). https://doi.org/10.1145/2078827.2078845
40. Ipeirotis, P.G.: Analyzing the Amazon Mechanical Turk marketplace. XRDS Crossroads ACM Mag. Stud. **17**(2), 16–21 (2010)
41. Joint, A., Baker, E., Eccles, E.: Hey, you, get off of that cloud? Comput. Law Secur. Rev. **25**(3), 270–274 (2009). http://www.sciencedirect.com/science/article/pii/S0267364909000570
42. Kandukuri, B.R., Rakshit, A., et al.: Cloud security issues. In: IEEE International Conference on Services Computing, 2009. SCC'09, pp. 517–520. IEEE (2009)
43. Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., Tsugawa, M.: Science clouds: early experiences in cloud computing for scientific applications. Cloud Comput. Appl. **2008**, 825–830 (2008)
44. Kittur, A., Chi, E., Suh, B.: Crowdsourcing for usability: using micro-task markets for rapid, remote, and low-cost user measurements. In: Proceedings of CHI 2008 (2008)
45. Krause, A., Horvitz, E., Kansal, A., Zhao, F.: Toward community sensing. In: Proceedings of the 7th International Conference on Information Processing in Sensor Networks, IPSN 2008, St. Louis, Missouri, USA, 22–24 April 2008, pp. 481–492 (2008)
46. Lasecki, W.S., Teevan, J., Kamar, E.: Information extraction and manipulation threats in crowd-powered systems. In: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW '14, pp. 248–256. ACM, New York (2014). https://doi.org/10.1145/2531602.2531733
47. Menychtas, A., Vogel, J., Giessmann, A., Gatzioura, A., Gomez, S.G., Moulos, V., Junker, F., Müller, M., Kyriazis, D., Stanoevska-Slabeva, K.: 4CaaSt marketplace: an advanced business environment for trading cloud services. Future Gener. Comput. Syst. **41**, 104–120 (2014)
48. Messina, F., Pappalardo, G., Rosaci, D., Santoro, C., Sarné, G.M.: A trust-aware, self-organizing system for large-scale federations of utility computing infrastructures. Future Gener. Comput. Syst. **56**, 77–94 (2016)
49. Mohomed, I., Dutta, P.: The age of DiY and dawn of the maker movement. GetMobile Mob. Comput. Commun. **18**(4), 41–43 (2015)
50. Mollick, E.: The dynamics of crowdfunding: an exploratory study. J. Bus. Ventur. **29**(1), 1–16 (2014)
51. O'Connor, L.: Celebrity nude photo leak: Just one more reminder that privacy does not exist online and legally, there's not much we can do about it. GGU Law Review Blog Paper 30 (2014)
52. Pallis, G.: Cloud computing: the new frontier of internet computing. IEEE Internet Comput. **14**(5), 70–73 (2010)
53. Patel, P., Ranabahu, A.H., Sheth, A.P.: Service Level Agreement in Cloud Computing. The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) Publications (2009)
54. PYBOSSA S: Scifabric pybossa. https://pybossa.com/ (2013)
55. Riahi, M., Papaioannou, T.G., Trummer, I., Aberer, K.: Utility-driven data acquisition in participatory sensing. In: Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, 18–22 March 2013, pp. 251–262 (2013)
56. Ritzer, G., Jurgenson, N.: Production, consumption, prosumption: the nature of capitalism in the age of the digital prosumer. J. Consum. Cult. **10**(1), 13–36 (2010)
57. Safecast: https://blog.safecast.org/. Accessed 14 Nov 2017
58. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2018)
59. Shahri, A., Hosseini, M., Ali, R., Dalpiaz, F.: Gamification for volunteer cloud computing. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), pp. 616–617. IEEE (2014)
60. Spark A: Apache Spark: lightning-fast cluster computing. http://spark.apache.org (2016)
61. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011). http://www.sciencedirect.com/science/article/pii/S1084804510001281
62. Surowiecki, J., Silverman, M.P.: The wisdom of crowds. Am. J. Phys. **75**(2), 190–192 (2007)
63. Taleb, T., Ksentini, A.: Follow me cloud: interworking federated clouds and distributed mobile networks. IEEE Netw. **27**(5), 12–19 (2013)
64. The AWS Team: Summary of the Amazon EC2 and Amazon RDS service disruption in the US east region (2011). http://aws.amazon.com/message/65648/
65. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a Map-Reduce framework. Proc. VLDB Endow. **2**(2), 1626–1629 (2009). https://doi.org/10.14778/1687553.1687609
66. Ullrich, M., Lssig, J., Gaedke, M.: Towards efficient resource management in cloud computing: a survey. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 170–177 (2016). https://doi.org/10.1109/FiCloud.2016.32
67. Wang, G., Wang, T., Zheng, H., Zhao, B.Y.: Man vs. machine: practical adversarial detection of malicious crowdsourcing

workers. In: 23rd USENIX Security Symposium. USENIX Association, Berkeley (2014)
68. Werbach, K., Hunter, D.: For the Win: How Game Thinking Can Revolutionize Your Business. Wharton Digital Press, Philadelphia (2012)
69. Yang, B., Chai, W.K., Pavlou, G., Katsaros, K.: Seamless support of low latency mobile applications with NFV-enabled mobile edge-cloud. In: 5th IEEE International Conference on Cloud Networking (CloudNet), Pisa, Italy
70. Yang, B., Chai, W.K., Xu, Z., Katsaros, K.V., Pavlou, G.: Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications. IEEE Trans. Netw. Serv. Manag. (2018). https://doi.org/10.1109/TNSM.2018.2790081
71. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache Spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016). https://doi.org/10.1145/2934664
72. Zahariev, A.: Google App Engine. In: Helsinki University of Technology Seminar on Internetworking (2009)
73. Zeng, D., Gu, L., Guo, S., Cheng, Z., Yu, S.: Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans. Comput. **65**(12), 3702–3712 (2016)
74. Zhuang, H., Rahman, R., Aberer, K.: Decentralizing the cloud: how can small data centers cooperate? In: 14th IEEE International Conference on Peer-to-Peer Computing (P2P), pp. 1–10 (2014). https://doi.org/10.1109/P2P.2014.6934303

Mahmood Hosseini is a Lecturer in Business Computing at Bournemouth University. His research interests include crowdsourcing in requirements engineering and the analysis of transparency requirements. Dr. Mahmood Hosseini received a PhD in Engineering Social Informatics from Bournemouth University.

Constantinos Marios Angelopoulos is Lecturer in Computing at Bournemouth University (UK) since 2016, specializing in future and emerging paradigms of computer networks and distributed systems. Previously, he spent 3.5 years as a Postdoctoral Researcher at University of Geneva (CH) under the prestigious Swiss Government Excellence Scholarship for Foreign Researchers. Marios is the Founding Program Leader of the three IoT Master courses offered in BU; namely, MSc in Internet of Things, MSc in IoT and Cyber Securtiy, and MSc in IoT and Data Analytics. He is also Founder and Head of the Future and Complex Networks (FlexNet) Research Group. Marios is the Lead Editor of the ITU-T Recommendation (Standard) on "Requirements and Functional Architecture of IoT-related Crowdsourced Systems", Co-editor of another 2 recommendations on IoT, Artificial Intelligence and Smart Cities, and the Liaison Co-Rapporteur to the SCV for Study Group 20.

Wei Koong Chai is a Senior Lecturer in Bournemouth University (BU), UK. He heads the Future and Complex Networks Research Group (FlexNet) in the Department of Computing and Informatics in BU. He is also currently a Visiting Academic in the Department of Electronic and Electrical Engineering, University College London (UCL), UK. Previously, he was a Senior Research Associate in UCL, where he led the research activities of several research projects, conduct collaborative research and contribute to the teaching and supervision of students at both undergraduate and postgraduate levels. He has successfully raised research funding from both EU and UK funding bodies. He has published papers in fully refereed international conferences and journals and contributed chapters to books. His research work has been constantly receiving citations, recording over 1000 citations to date (*source* Google Scholar). He also serves on the Technical Program Committee of several IEEE/ACM international conferences and workshops. Dr. Chai received the BEng (Hons) Degree in Electrical Engineering from the Universiti Teknologi Malaysia, Johor Bahru, Malaysia, in 2000, and the MSc (Distinction) and PhD Degrees from the University of Surrey, Surrey, UK, in 2002 and 2008, respectively.

Stephane Kundig is a research assistant and PhD candidate at the TCS-Sensor Lab of the University of Geneva. Previously he obtained his Diploma Ing. Degree in Electrical and Computer Engineering from the National Technical University of Athens (NTUA). His research interests lie in the field of modern networks, wireless sensor networks, collaborative computing and cognitive IoT.