Using Crowdsourcing to Provide QoS for Mobile Cloud Computing

Dezhong Yao, *Student Member, IEEE*, Chen Yu, *Member, IEEE*, Laurence T. Yang, *Senior Member, IEEE*, and Hai Jin, *Senior Member, IEEE*

Abstract—Quality of cloud service (QoS) is one of the crucial factors for the success of cloud providers in mobile cloud computing. Context-awareness is a popular method for automatic awareness of the mobile environment and choosing the most suitable cloud provider. Lack of context information may harm the users' confidence in the application rendering it useless. Thus, mobile devices need to be constantly aware of the environment and to test the performance of each cloud provider, which is inefficient and wastes energy. Crowdsourcing is a considerable technology to discover and select cloud services in order to provide intelligent, efficient, and stable discovering of services for mobile users based on group choice. This article introduces a crowdsourcing-based QoS supported mobile cloud service framework that fulfills mobile users' satisfaction by sensing their context information and providing appropriate services to each of the users. Based on user's activity context, social context, service context, and device context, our framework dynamically adapts cloud service for the requests in different kinds of scenarios. The context-awareness based management approach efficiency achieves a reliable cloud service supported platform to supply the Quality of Service on mobile device.

Index Terms—Mobile cloud computing, crowdsourcing, context-awareness, quality of service, service discovery

1 Introduction

Next generation computing infrastructure, cloud computing, has posed a number of challenges to mobile client users. One of the main concerns is the Quality of Service (QoS), largely due to the diversity of kinds of services and the complexity of the mobile environment [1]. As the users' mobility, they often lack the capabilities or knowledge of service providers and network environments in the different places. They do not know how to choose the suitable cloud service on their own. Some methods [2], [3], [4], [5] are implemented for single mobile user to aware local network environments. However, this local context-aware method can only aware limited environment knowledge and the device need to continually aware of the environments when they move to a new place [6]. This causes battery issues and the suitable service with appropriate QoS may not be discovered due to limited context knowledge.

To overcome these shortcomings, users are hired to share the usage experience of choosing a suitable mobile cloud service in different context environments. The globe environment awareness task is achieved by crowd user. Globe context-aware method has more capability and gathers more knowledge of service providers and network environments than the local context-aware. Single user's awareness result can be gathered to solve a complex problem efficiently. The participant's result can be used for new arrivers, so the users do not need to continue being aware of the environment and will quickly discover the suitable services when they arrive in a new place. The framework introduced in this paper is based on a attractive type of outsourcing called crowdsourcing [7]. The term crowdsourcing describes a new distributed business model that resolves the complex problems. Recent studies [8], [9], [10] demonstrate the effective adoption of crowdsourcing techniques for collecting multiple and reliable decision tasks.

Mobile cloud computing is expected to make mobile devices more powerful by using distributed online computing resources. They are the main interest in the commercial applications of cloud computing using the Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS) delivery models [2]. In those platforms, software and resources are hosted on the cloud instead of with the client, who pays for the required resources according to their resource usage. In order to increase the efficiency and stability of the cloud service for mobile users, the web service composition is introduced [11], [12]. Web service composition provides a way to combine basic web services (possibly offered by different providers) and value-added services to meet the needs of users. For a group of candidate services with the same functional capabilities, QoS plays a key role in service selection and service composition. On the other hand, QoS can help users to avoid resource wastage

- *D. Yao, C. Yu, and H. Jin are with the Services Computing Technology and System Laboratory and the Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: {dyao, yuchen, hjin}@hust.edu.cn.*
- *L.T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada. E-mail: ltyang@gmail.com.*

Manuscript received 20 May 2015; revised 19 Oct. 2015; accepted 15 Dec. 2015. Date of publication 30 Dec. 2015; date of current version 5 June 2019. Recommended for acceptance by C.H. Foh, S.N. Srirama, E. Benkhelifa, B. Kantarci, P. Chatzimisios, and J. Wu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2015.2513390

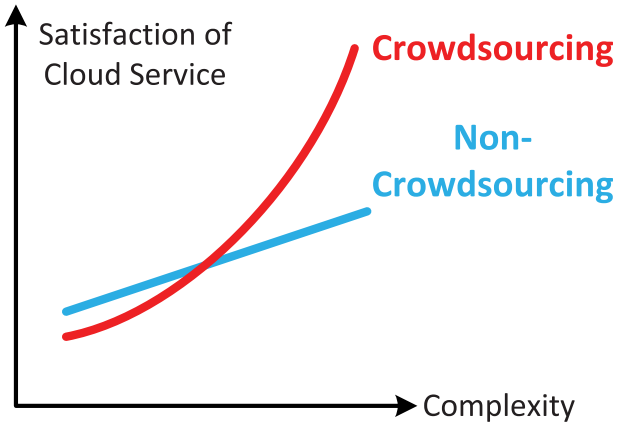

Fig. 1. Crowdsourcing will improve user satisfaction when a correct service is chosen to meet the user's need, but it will harm satisfaction when it is lack of knowledge that uncertain what action to take with the current environment to supply QoS.

and higher monetary cost, when the service requested by mobile application may exceed the capability of the device in its current context environment. Users only trust mobile cloud computing when the cloud services have solid QoS.

Mobile computing environments are complex and unpredictable. The presence of lack of knowledge of understanding environments and inefficient discovering services harms the users' quality of experience (QoE) in the service thus rendering it useless [13]. QoE is used to measure user's experiences on the entire cloud service experience [14]. Context-awareness has been invented and roughly exploited in supporting Quality of Service [5], [15] to optimize QoE. Based on context information, the service adaptor (SA) can understand mobile environments and intelligently make decisions to choose suitable cloud service without interrupting the user. To make SA more intelligent and efficient, we want all users involved to solve this complex problem. Each user will update his or her usage history and context information to a third party platform. Then, we will use collective wisdom to achieve the intelligent cloud service chosen problem. Crowdsourcing system has been built to complete complex data collecting tasks [9].

Fig. 1 shows the relationship between user's satisfaction, crowdsourcing of decider, and complexity of awareness. The relationship is summarized from [13], [16], [17], [18]. As shown in Fig. 1, crowdsourcing improves a user's satisfaction of using cloud service. When we have more certainty of context information, the service adaptor will be more intelligible and efficient. Using crowdsourcing technology, we can learn user habits from history. Then, we can supply more quality cloud services for a user. Crowdsourcing can help SA to understand more mobile environments and make the right decision. While there is no direct relationship between service quality and context information, we need to build several reasoning mechanisms to achieve qualified services for mobile users.

In this paper, we propose a crowdsourcing based quality of service framework for mobile cloud service: crowdsourcing based QoS adaptor (CQA) for different kinds of mobile applications. CQA is a middleware approach that enables dynamic adaptation of cloud service, and safeguards emergency service request, efficient resource utilization, and savings in monetary provision costs. By monitoring quality of resource and quality of device, CQA will respond to cloud service request following QoS priority level. All of the actions are under control of context-reasoning component in CQA.

The key contributions of this work are summarized as follows:

- Design a crowdsourcing platform to supply QoS for mobile cloud service.
- Propose a crowdsourcing-based server discovery schemes for choosing the optimal cloud service.
- Implement a prototype platform and evaluate the efficiency of crowdsourcing model with traditional schemes.

The rest of this article is structured as follows. We briefly introduce mobile cloud computing and QoS needs. Then, we present the architecture of the framework CQA and QoS modeling method. In the next section, the QoS control algorithms used in context-reasoning component are described in detail. Finally, conclusions are drawn and future studies are proposed.

2 Related Works

Quality of Service in mobile cloud computing environment describes the treatment of information as it is exchanged between mobile client and cloud service. The Quality of Service in mobile cloud environment measures service in availability, priority, cost, response time, and throughout. For the different types of cloud service, an operator is needed to be developed to achieve different treatment within the environment for them to function properly.

2.1 Mobile Cloud Service

Cloud computing is a large-scale distributed network system based on a number of servers in data centers. The models of cloud services can mainly be categorized based on a layer concept. In the upper layers of this paradigm, Infrastructure as a Service, Platform as a Service, and Software as a Service are stacked. Due to the limitation of mobile device, the typical mobile cloud computing services can be functionality grouped into two categories:

2.1.1 *Storage Service*

This type of service aims to solve the problem of storage limitations on mobile devices. The applications need large data transmission between mobile client and server. Network availably, respond time, and throughout are the main concerns of this type of service. Mobile commerce, mobile healthcare [16], mobile learning, and mobile multimedia are typical applications belonging to this kind of service. Mobile commerce usually requires low network cost and throughout, but high availability and response time. Some other services may need high QoS for high throughout.

2.1.2 *Computing Service*

The computing services transfer the heavy computing task from mobile device to cloud and achieve the results. The applications offload the task and data to cloud [19], which is a suitable solution to address the issues of computational power and battery lifetime. Mobile designing, mobile online

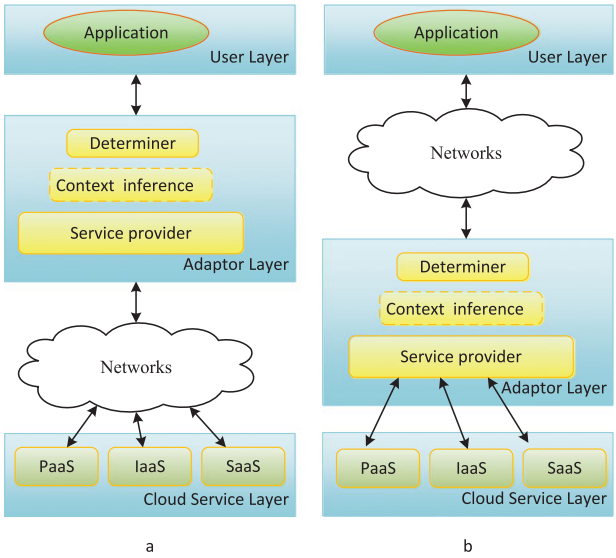

Fig. 2. Two configurations of context-aware mobile cloud services.

gaming, and mobile multimedia are common applications which require large processing resources. Some of them also need high QoS for short response time and high throughout.

In addition, there are a class of services that do not guarantee response time and priority. These are called best effort services, such as email, file backup, and status updates.

As described above, different types of cloud services may need different QoS requirement. To understand fully service request need and current mobile environment, we use context-awareness method to achieve this goal.

2.2 Context-Aware Mobile Cloud Service

Context-awareness is an excellent solution to sense mobile environments and intelligently choose the best cloud service [2], [3], [4], [20]. Summarizing [1], [5], [17], [21], [22], [23], there are two main context-aware mobile cloud services architectures, as shown in Fig. 2. One is service adaptor running on mobile side (Fig. 2a), and the other is SA running on cloud (Fig. 2b). When SA is running on a mobile device, the service providers are located in different places, which we called distributed architecture. When SA is settled on the cloud side, all the services are grouped together for a mobile device. We named this environment centralized architecture.

There are three layers in context-aware mobile cloud service architecture: user layer, adaptor layer and cloud service layer. Adaptor layer is in charge of cloud service matching process. Difference cloud services register their service on service provider component in adaptor layer. Context inference senses user's environment and provider context information for determiner. Determiner selects the most suitable cloud service from service provider pool based on current context information.

2.2.1 Distributed Architecture

For distributed architecture, the service providers need to register in the context service adaptor on a mobile device. The service adaptor will decide to choose the most suitable cloud service based on the context environment to meet the QoS request. As the network environment is constantly changing, different providers will perform different QoS on a mobile device. It is more robust for a service adaptor to have multiple choices than the centralized architecture. However, it will spend more on discovering and maintaining those services.

2.2.2 Centralized Architecture

The adaptor layer is located on the cloud side and running as cloud service. It will gather the context information from a mobile device and discover server candidate services meeting functionalities required by the consumer. Then, it determines the most appropriate form of adaptor. Finally, the adapted service is invoked and returns the results to the service consumer. Due to the adaptor running on a remote server, the mobile device may lose the connection to the adaptor.

2.3 Mobile Crowdsourcing

Crowdsourcing has been successfully applied in commercial applications: Mechanical Turk, iStockPhoto, and Innocentive. There are also several reaches on mobile crowdsourcing. Eagle et al. [24] developed a mobile crowdsourcing system, txteagle, that hires users with little benefit to completing simple tasks such as translation, transcription, and filling surveys. Yang et al. [25] designed incentive mechanisms for mobile user sensing. Alt et al. [26] considered location context as one parameter for distributing tasks for workers. For the participatory sensing system, Kumrai et al. [27] introduced an incentive mechanism to satisfy both the users and the service providers. However, none of them can be directly applied to QoS control platform.

2.4 Motivation for Crowdsourcing-Based QoS Platform

The mobile users may have some issues such as congestion due to the limitation of wireless bandwidths, network disconnection, and the signal attenuation caused by mobile users' mobility. To continue using cloud services, we need to reconfigure the system settings by hand for different mobile environments. Furthermore, lack of provider's information is also a shortcoming to choose suitable cloud service. Context-awareness is an excellent solution to sense mobile environments and intelligently choose the best cloud service. In addition to crowdsourcing technology, we can achieve the goal to choose intelligently the best cloud service to provide QoS for mobile device. There is some background we need to understand first.

2.4.1 Macro Perspective: Quality of Provider

The service quality of a main provider refers to the capability of a provider. The capability of a provider includes the computing power and storage capacity. The higher the capability offered by a provider, the better the cloud service will be. Different providers will have different service quality for mobile users. A mobile user does not know each provider's capability, so we need a central registration center to collect that information. Moreover, the mobile users may face complex network situations in a mobile cloud environment. They cause delays when users want to communicate with the cloud, so QoS is reduced significantly.

2.4.2 Micro Perspective: Quality of Environment

The same cloud service provider may perform different Quality of Service for a mobile user in different network

TABLE 1
Measured Internet 2 Round-Trip Times Latency Between Representative Sites

	Min	Mean	Max	Lower bound
Berkeley-Canberra	174.0	174.7	176.0	79.9
Berkeley-New York	85.0	85.0	85.0	27.4
Berkeley-Trondheim	197.0	197.0	197.0	55.6
Pittsburgh-Ottawa	44.0	44.1	62.0	4.3
Pittsburgh-HongKong	217.0	223.1	393.0	85.9
Pittsburgh-Dublin	115.0	115.7	116.0	42.0
Pittsburgh-Seattle	83.0	83.9	84.0	22.9

environments. The network bandwidths are different and network latencies are also not the same. Table 1 shows a measurement result [28] of network latency between different cloud services. In the same place, e.g. Pittsburgh, the internet 2 round-trip times is totally different from different cloud servers. From Table 1, we can see that the quality of a cloud service will be significantly different due to the different network environments. Therefore, the service quality of a provider mainly depends on network bandwidth, network latency [28]. The network bandwidth is influenced by what kind of wireless connection is used, how many people use it, and user's mobility. So the service quality is related to multiple context information. The network latency reduces interactive performance even with good bandwidth [28]. Therefore, the quality of a cloud service can not just be observed from network environment, the network and cloud service providers need to be measured both.

3 Quality of Service Management

In this section, a service quality model for mobile cloud computing environment is introduced. The cloud service providers and network service carriers are the main factors affecting the quality of cloud services for mobile users.

3.1 Service Quality Model

As shown in Fig. 3, the structure of mobile cloud service can be divided into three parts: mobile users, network carriers and cloud service providers. The mobile users can freely choose different networks to use different cloud services. The quality of service on mobile device is affected by both the cloud service providers and network service carriers. We first give the definition for the two types of services in mobile cloud computing environment:

Definition 1 (Web Service, WS). *Web Service is a tuple: $WS < F, QoS >$, where F denotes the service property supplied by a cloud service provider and QoS represents the quality of service at service side.*

The Web Service is the basic service provided by different cloud service providers. The basic services offer virtual machine service, virtual desktop service, storage service and so on. Four QoS properties [1] are used to describe the capabilities of different provider's services, which includes computing capacity(**CPU**), storage(**STO**), reliability(**RL**) and price(**PR**). Web Service (**WS**) represents the number of computing and storage requests the cloud service provider is able to process per second. The reliability of a cloud service is denoted by **RL**, which can be measured by the frequency of usage. **PR** is the fee that a user pays for a single service request.

Definition 2 (Network Service, NS). *Network Service is a non-functional description: NS(RTD, BW), which describes the quality of network environment.*

RTD is the round-trip delay time between mobile device and cloud service. **BW** is the network bandwidth observed from end user for different service. The same cloud service may have different bandwidth through different network carriers. We assume that the mobile user can freely use different types of wireless network.

The Web Services can not be called by mobile users directly [29]. The mobile user needs to use the cloud service through different network. The network bandwidth is considered in NS. That is because a poor network environment will limit the network bandwidth for mobile users even through the cloud service provider supports high network bandwidth.

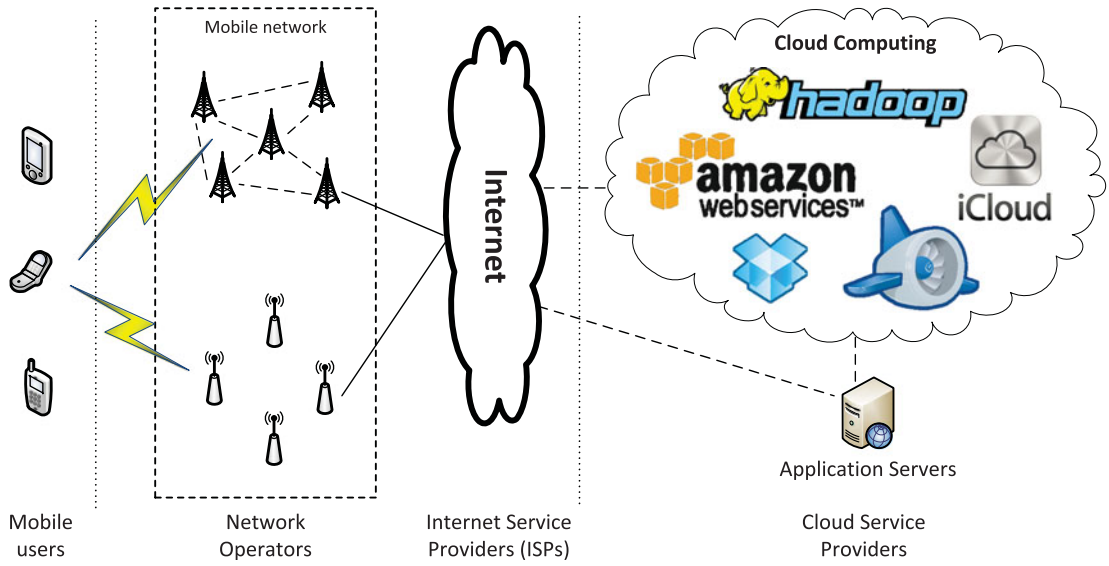

Fig. 3. Mobile cloud computing architecture.

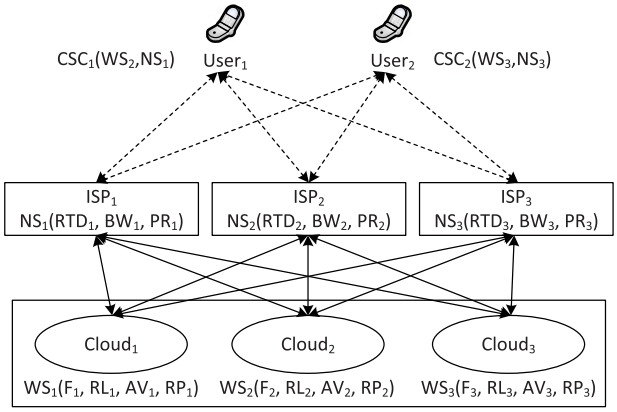

Fig. 4. Mobile cloud service composition model: An example.

Definition 3 (Cloud Service Context, CSC). *Cloud Service Context describes the context information of remote cloud service and current network environment. A CSC is formally defined as (F, CPU, STO, RL, PR, RTD, BW).*

The Cloud Service Context consists of a set of service properties, including web service properties and network service properties as shown in Fig. 4. Those service properties can be evaluated at end user's mobile device directly. As different cloud service provider and network carrier have different service qualities, we need to evaluate the overall performance and choose the suitable service providers. In this paper, we assume that the end user can choose any cloud service and any network service at any time.

Different types of mobile applications have different quality of service requirements. For example: Streaming applications mainly concern about network bandwidth [30], Realtime applications focus on network response time [31], [32] and Data synchronization applications are sensitive to price and energy consumption [33], [34]. A defined quality of service is desired for certain types of application.

Definition 4 (Quality of Mobile Service). *Quality of Mobile Service quantitatively measures the quality of each cloud service observed at mobile client.*

To quantitatively measure quality of different services, we use the context information in CSC as the input aspects. Given N cloud service providers and M network service carriers, each mobile user can choose $N \times M$ types of services as illustrated in Fig. 4. Each cloud service can be connected through different network. Observed from end-user side, the available service dataset is a combination of cloud services and network services. Each cloud service and network service information are described as a row vector in Eqs. (1) and (2).

$$WS = \begin{pmatrix} WS_1 \\ WS_2 \\ \dots \\ WS_N \end{pmatrix} = \begin{pmatrix} CPU_1 & STO_1 & RL_1 & PR_1 \\ CPU_2 & STO_2 & RL_2 & PR_2 \\ \dots & \dots & \dots & \dots \\ CPU_N & STO_N & RL_N & PR_N \end{pmatrix} \quad (1)$$

$$NS = \begin{pmatrix} NS_1 \\ NS_2 \\ \dots \\ NS_M \end{pmatrix} = \begin{pmatrix} RTD_1 & BW_1 \\ RTD_2 & BW_2 \\ \dots & \dots \\ RTD_M & BW_M \end{pmatrix} \quad (2)$$

We use a combination matrix $S = WS \otimes NS$ to describe all the available services in Fig. 4.

$$S = \begin{pmatrix} S_{1,1} \\ S_{1,2} \\ \dots \\ S_{i,j} \\ \dots \\ S_{N,M} \end{pmatrix} \quad i \in \{1, \dots, N\}, j \in \{1, \dots, M\} \quad (3)$$

$$\begin{aligned} S_{i,j} &= (WS_i, NS_j) \\ &= (CPU_i, STO_i, RL_i, PR_i, RTD_j, BW_j) \end{aligned}. \quad (4)$$

The goal of service discovery mechanism is to find a suitable $S_{i,j}$ in matrix S for end-user in different context environment. All the context properties in matrix S can be quantitatively measured using a real-valued [1]. For different application requirement, we only need to rank the performance of the same context property in one column in S to find the most suitable services. After we use a quantitative description of each context properties in CSC, we normalize all the properties in matrix S to make all properties with the same weight.

$$S_{norm} = norm \begin{pmatrix} S_{1,1} \\ S_{1,2} \\ \dots \\ S_{i,j} \\ \dots \\ S_{N,M} \end{pmatrix}. \quad (5)$$

Without loss of generality, a weight vector $W = [p_{\text{CPU}}, p_{\text{STO}}, p_{\text{RL}}, p_{\text{PR}}, p_{\text{RTD}}, p_{\text{BW}}]^\top$ is used to describe the importance of each property. So the Quality of Mobile Service can be computed as:

$$\begin{aligned} &QMS = S_{norm} W, \\ &\text{s.t. } p_{\text{CPU}} + p_{\text{STO}} + p_{\text{RL}} + p_{\text{PR}} + p_{\text{RTD}} + p_{\text{BW}} = 1 \end{aligned} \quad (6)$$

3.2 Context-Awareness Based Service Discovery Model

The quality of cloud service at end user is affected by WS and NS as shown in Fig. 4. The context-awareness method is to find a suitable service combination for mobile user based on current network environment. As the quality of each combination service can be calculated by QMS, to find a suitable service combination is to find the maximum quality of service in QMS:

$$< i, j > = \arg \max_{<i,j>} (QMS). \quad (7)$$

By given a weight vector W and services' context information matrix S_{norm} aware from environment, the quality of each combination service is computed by Eq. (6). The weight vector W can be calculated using least squares method based history data or directly defined for different applications. The context information matrix S_{norm} is collected from context-awareness platform.

3.2.1 Service Discovery Algorithm

Confronted with a variety of network environments, our purpose is to select an optimal combination service based

on application's requirement. The context-awareness based service discovery method can be viewed as two steps: context aware and QoS ranking. Given the information of cloud service providers and network providers (ISPs), Context Aware is collecting the QoS performance of each service combination. When the context collection is finished, QoS ranking searches for the optimal combination service by calculating QoS score QMS. The process is shown in Algorithm 1.

Algorithm 1. Context-Awareness Based Service Discovery

Input: W, $ISP = \{ISP_1, ISP_2, \ldots ISP_M\}$,
$CLOUD = \{CLOUD_1, CLOUD_2, \ldots CLOUD_N\}$
Output: $< i, j >$
1: Initialization;
2: {**Step 1**} Context aware:
3: **for** ISP_j in ISP **do**
4: **for** $CLOUD_i$ in $CLOUD$ **do**
5: Test performance: (WS_i, NS_j)
6: $S_{i,j} \leftarrow (WS_i, NS_j)$
7: **end for**
8: **end for**
9: {**Step 2**} QoS ranking:
10: $S_{norm} \leftarrow Normalize(S)$
11: $QMS \leftarrow S_{norm}W$
12: $< i, j > \leftarrow \arg\max_{<i,j>}(QMS)$
13: **return** $< i, j >$

3.2.2 Quality Constraints

Due to the diversity of application requirements, quality of service can be measured in several aspects. Each application has its certain constraint on quality of service. Based on previous studies [1], [6], [32], [33], five main QoS constraints are considered for different quality requirements: bandwidth, response time, price, energy and security.

- Bandwidth: Given an service combination, the bandwidth describes the data transmission capacity of network. The weight vector can be set as $W_b = [0, 0, 0, 0, 0, 1]^\top$.
- Response time: Given an service combination, the response time measures the expected delay in seconds between the mobile device sends the request and receives the result. The weight vector can be set as $W_r = [0, 0, 0, 0, 1, 0]^\top$.
- Price: The price constraint is the fee that a user needs to pay for a single service. The mobile user wants to complete a task with a lowest cost. The weight vector can be set as $W_p = [0, 0, 0, 1, 0, 0]^\top$.
- Energy: The energy constraint is the power consumption which is spent on a complete service process. The power consumption is affect by the data transfer time. For a certain amount of data, the network service which has higher bandwidth and reliable service should be selected. The weight vector can be set as $W_e = [0, 0, 0.5, 0, 0.5, 0]^\top$.
- Security: The security constraint focus on the cloud service's reliability. The cloud service is rated by its uptime and reputation. The weight vector can be set as $W_s = [0, 0, 1, 0, 0, 0]^\top$.

The mobile application can also change the properties of the weights in the vector W for different purposes. Our model is suitable for a general QoS constraint.

3.2.3 Analysis of Discovery Time

The general context-awareness based service discovery method evaluates the performance of N cloud service providers and M network service carriers. The mobile user tests all the service combination (WS_i, NS_j) before QoS ranking. The time complexity to finish all the evaluation is O($N * M$).

4 CQA Design

The CQA is a third-party platform which continues sensing environment, monitoring resource, and making decisions based on different service requests. The platform has running an agent on the client side to collect each user's context information. In this section, we will introduce each component in the CQA platform.

4.1 CQA Overview

Fig. 5 presents a high-level view of the CQA functional components and their interdependencies. As shown in this figure, the CQA framework acts as an intermediate layer between mobile applications and cloud services. The adaptor layer is the middleware that receives service request and returns the suitable service to the requestor. A brief explanation of the most important components is provided below.

Context inference—It senses the environment information: user context, device context, and resource context. Based on the context information, we can have more certainty of current environment. User context contains information about activity, location, routine pattern, and social relationship. Device context information contains network bandwidth, network load, wireless network model, signal strength, and battery life. QoS status explains the availability of service, service cost, and response time for each cloud service. Context inference collects the information through the sensors on a mobile device. It provides the necessary algorithms in order to estimate the current unavailable context based on information available from a parent community; for example, converting location coordinates to user friendly tags (e.g., home, office, campus, mall); and estimation of user's physical activity (e.g., using inertial sensors to determine whether the user is standing, sitting, walking , running, etc.).

Context collector—The context information gathered from users is stored in two databases: Context DB and Provider DB. The Context DB stores user context information and device context information. Together with user context, device context describes an environment context. We assign a unique identity to index for different context environments in Context DB. For each context environment, we record the QoS performance of the used cloud service in this environment and store in Provider DB.

Determiner—This module is the core of the CQA. It plays three main roles in the CQA: service request scheduling, environment matching, and provider selection. Service request scheduling chooses the highest priority request to run. As the service request has different QoS needs, we give the priority for each request to meet the demand. Environment matching is, at runtime, triggered by a message from

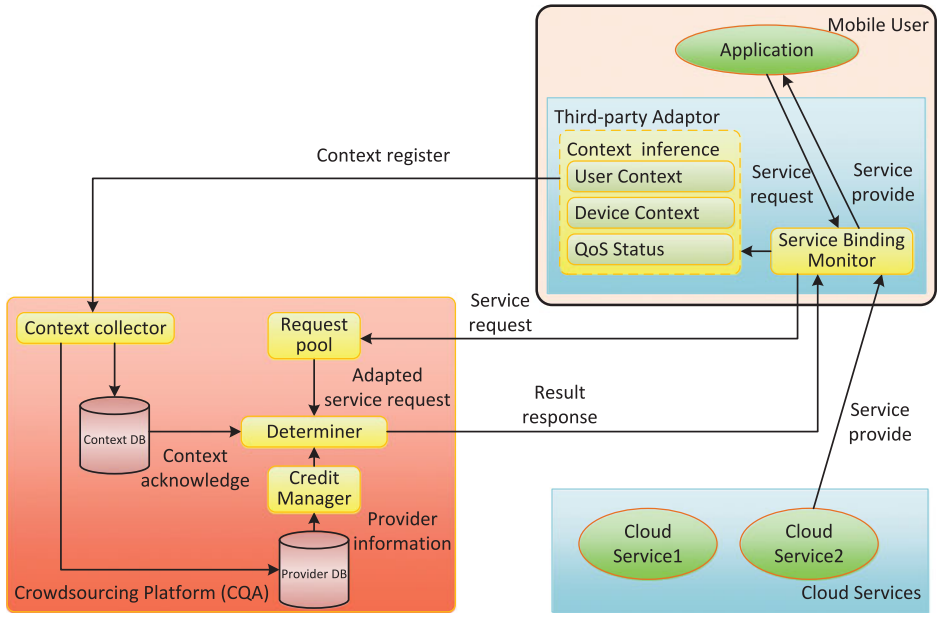

Fig. 5. The architecture of crowdsourcing based QoS supported mobile cloud service. CQA platform is running on a third-party server. The adaptor is implemented in a mobile user's local phone. Cloud services are maintained by different business companies.

the request pool to find the records of some of the best match context environment descriptions. Based on the context environment described in the service request, the determiner will query the most similar context environments stored in context DB and generate the identity list. Provider selection selects the most suitable cloud service providers in Provider DB to meet current service request. The providers are ranked by QoS performance in the request context environment. The ranking strategy is made by decision tree, which is built by initial rules and history service usage record. Based on the context environments, we select the suitable providers which were previously used in this environment. After the providers are selected, the determiner will notice the service binding monitor on the user side.

Request pool—This is a single queue whose requests basically follow first-in-first-out rules. We have a priority function to evaluate the importance of the new service request and add it to the right position of the chain. Some services will be held for a minute until the previous service releases the resource.

Service Binding Monitor—This module takes charge of adapting a service request to the Broker and monitoring the service in use. When an appropriate service is available, the Service Binding Monitor forwards the service request and responds to the requestor. It also gathers the resource usage by each service and reports it to the Determiner. It guarantees QoS for each service.

Service provider—It merges different types of cloud service for a mobile device and supplies the profile information of each cloud service. After introducing the components in crowdsourcing platform architecture, we will explain the workflow of our CQA platform. There are three main steps to finish the whole system task as shown in Fig. 6: 1) Context gathering, 2) Crowdsourcing computing, 3) QoS ranking. After gathering enough context information from each user, we adopt crowdsourcing method to model the context information into the knowledge database. Then, we can respond to the user's service request by selecting the more suitable provider from the service pool under knowledge database's guidance.

4.2 Context Gathering

Individually, users will update their cloud service usage report to CQA platform. The context environment, the type of cloud service, and performance result will be sent to CQA center. As shown in Fig. 6, crowd user update their context information to the crowdsourcing platform CQA. The context information includes user's environment, network provider and cloud service provider. We set context update interval to be 12 hours for each user. If one user dose not find any context information for current location, the user will start a new aware task and update the result to CQA.

All the context information is anonymous in order to protect user privacy. The new cloud service will also register on CQA platform. The more information they gather, the more suitable the cloud service they will select. The mobile side context can be described as:

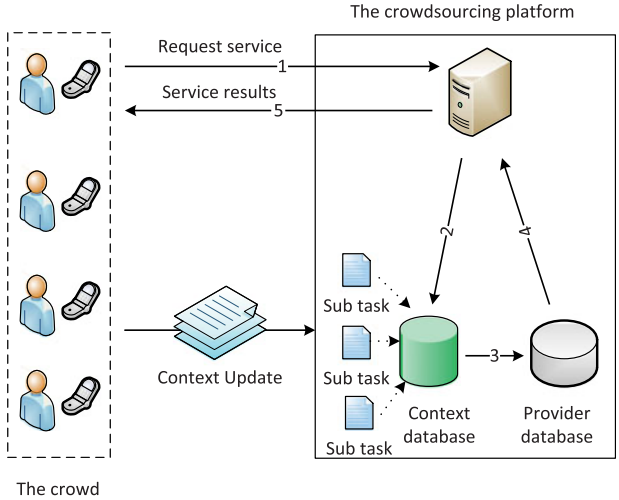

Fig. 6. The workflow of the crowdsourcing based service discovery process.

$$C = (User, Provider, Performance) = ([CUM, CUL, CNT], [CPN, CPD], [S_{i,j}]). \quad (8)$$

The explanation of each symbol is as follows:

- *CUM*: context of user mobility;
- *CUL*: context of user location;
- *CNT*: context of network type;
- *CPN*: context of provider name;
- *CPD*: context of provider description;
- $S_{i,j}$: performance of cloud service i through network provider j.

We use [CUM, CUL, CNT] to describe the mobile client environment and use [CPN, CPD] for cloud service provider's information according to the summary in [1]. The performance of certain providers is evaluated by $S_{i,j} = (WS_i, NS_j)$.

4.3 Service Discovery

The service discovery process can be described as a query for the most suitable result from gathering data. The workflow of crowdsourcing based service discovery is given in Fig. 6. Each mobile user's service usage update will be considered as a sub task to meet service discovery process. The update data will be stored in a context data base, which describe the relationship between context environment and cloud service providers. After collecting enough context information, we can build a determiner, which can rank the performance of each cloud service in a certain kind of context environment. The context database stores the relationship between context environment and available cloud providers. The QoS performance is also attached to this relationship.

The service request contains two main parts: requestor's information and service constraint. The description of the request can be denoted as:

$$Q = (User, Provider, Constraint) = ([CUM, CUL, CNT], [CPN, CPD], W). \quad (9)$$

The requestor's information is the user context which is a condition for the service query. After the alternative services retrieved from the crowdsourcing database, the quality of services is calculated under the constraint W. The whole crowdsourcing based service discovery process is summarized in Algorithm 2.

Algorithm 2. Crowdsourcing Based Service Discovery

Input: $Q = ([CUM, CUL, CNT], [CPN, CPD], W)$
Output: $< i, j >$

```
Initialization;
if ContextDB is empty then
    run Algorithm 1
    add C=(User, Provider, Performance) to ContextDB
    return ∅
else
    select S_{i,j} from ContextDB where (C.User=Q.User and C.Provider=Q.Provider)
    S_norm ← Normalize(S)
    QMS ← S_norm W
    < i, j > ← arg max_{<i,j>} (QMS)
    return < i, j >
end if
```

4.4 Service Selection Engine

When the user requests a cloud service, the two parameters [*User, Performance*] need to be provided to query the available results. Our ranking model calculates the QoS performance after the available cloud providers are found. Then, we select the top-rank provider as the result and send back to the requestor. The more data we collect, the more accurate the service we can choose. Those steps are shown in the numbered line in Fig. 6.

The service selection engine uses the similarity-based method to choose the most suitable providers. The similarity-based decision algorithm intends to determine the available mobile cloud providers in a given context environment by using the similarity distance. By evaluating the similarity distance between request context vector and history context vector in the database, we can query for a list of providers in the database. These providers can run in the requestor's context environment as the two parameters [*User, Performance*] are given. Then, we rank the QoS performance using the history record and select the best one as the result.

The service selection engine (as Determiner in Fig. 5) is running on a third-party server in the crowdsourcing platform (CQA). The Determiner in CQA selects the appropriate cloud provider for mobile user. When a mobile user needs to deploy an application to the cloud, it will send a request to CQA platform. The CQA will use a user-centric view of the expected quality to select the most appropriate provider. At last, the result will send back to mobile user. The mobile user will evaluate the performance of different cloud service. The CQA will gather those performance reports and selects the appropriate cloud provider for a new requestor.

4.5 Credit Manager

We design a credit manager component which is in charge of evaluating the reliability of each service provider. The credit is the statistical results of the successful service times. After the context information C is updated from users, the crowdsourcing platform will record the available service provider in provider database as shown in Fig. 5. Base on the provider's uptime and usage frequency, the credit manager calculates the reliability score **RL** for each service **WS**.

$$RL = \frac{\text{frequency of usages}}{\text{frequency of attempts to connect to the service}}. \quad (10)$$

4.6 Theoretical Analysis

From Algorithm 2, we can see that the service discovery process is a data matching process. The service performance records are selected from database by requestor's constraint. The service discovery query time is constant time, so the time complexity of this process is O(1). Compare with the time complexity O($N * M$) of Algorithm 1, the crowdsourcing based service discovery method is significantly reduced the service discovery time.

5 Prototype Implementation

The proposed platform is implemented using network simulator NS-3 [35]. The crowdsourcing server, mobile devices, and cloud services are all considered as node class in NS-3.

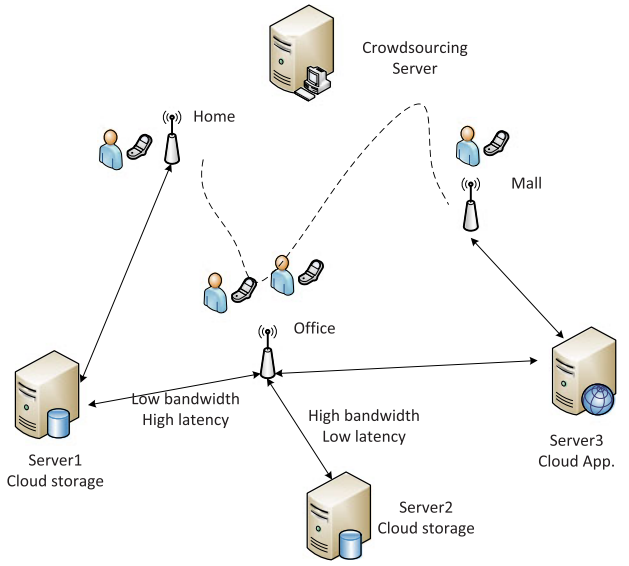

Fig. 7. A simple crowdsourcing working scenario. The CQA platform provides optimal service information at different locations.

A simple simulation scenario is illustrated in Fig. 7. The crowdsourcing server node will gather the context information from different users. When a mobile device requests a cloud server, it will send its context information to crowdsourcing server node. Then, the server will tell the requestor which cloud service provider is the most suitable based on the requestor's context using ranking scheme. Finally, the request node will communicate with the cloud server directly. The detailed working steps of the crowdsourcing platform are as follows:

a Initial adaptor and generate context: location, network type, network service carrier,
b The application requests cloud service from the adaptor,
c The adaptor sends a request and device context to crowdsourcing platform,
d Ranking the similar records order by cloud QoS performance,
e If there are records in the knowledge database, go to step g,
f Ask the devices in the location to test the performance of all the cloud service providers in the database, send the benchmark to platform, and go to step d,
g Send back the result to the requestor,
h The adaptor chooses the cloud service provider for the application.

5.1 CQA Adaptor

The context inference and service binding monitor at mobile user side run in the background when CQA adaptor is turned on. As a third-party adaptor, CQA provides a simple interface to receive service requests and return the alternative services list order by QoS score. An application sends the service queries to the CQA interface and waits for the result. After received the result, the requestor will connect to the remote cloud server directly.

The CQA adaptor will monitor the provider's status and timely update the service's performance to CQA platform. When the user arrives at a new location, the CQA adaptor will query for all available services at current location and test performance.

5.2 CQA Server Platform

The crowdsourcing platform maintains the context database and responses the service queries. As a service provider, the CQA platform can also be considered as a cloud service which provider service information. It supplies the quality evaluation results of each cloud service.

The CQA platform is a user-centric crowdsourcing model [25] that the mobile device has the right to select which task to run. This mechanism is scalable because its running time is linear in the number of users. The user-centric model is computationally efficient and truthful than other models [25].

5.3 Applications

We simulated two applications on top of CQA: an online media application focusing on downloading and a photo backup application focusing on uplink evaluation.

5.3.1 Online Media

To simulate online media application, multiple video files are downloaded from cloud servers. It requires high bandwidth and stable network environment to protect the quality of online video service. Those video files are stored in different cloud servers and mobile users randomly select which file to download. The mobile user is also randomly moving around in different places. After one video file is downloaded, the mobile user will attempt to download another video file. The download time and download speed are recorded for evaluation during the simulation process.

5.3.2 Photo Backup

The photo backup is a typical cloud storage application, which is usually impact by the network availability. With multiple sensors on mobile device, *e.g.* camera, GPS and temperature, more and more files are generated, and uploaded them to flickr, facebook, *etc.* Those log files are is small but many, the background backup application need to continue upload files to server. Since photo backup is delay tolerant, the quality constraint is not as strong as online media service. However multiple files needs to be uploaded to server continually, the number of successful uploads is used to evaluate the reliability of network provider and cloud service.

6 Simulation Setup and Performance Evaluation

To observe and analyze how crowdsourcing can supply the efficient and stable cloud service with QoS, we built a simulation environment to evaluate the platform performance using NS-3 [35].

6.1 Case Study

The experiment scenario is shown in Fig. 8. Our experiment focuses on comparing the efficiency and stability of crowdsourcing platform and general context-awareness QoS scheme. A mobile device will automatically choose a suitable

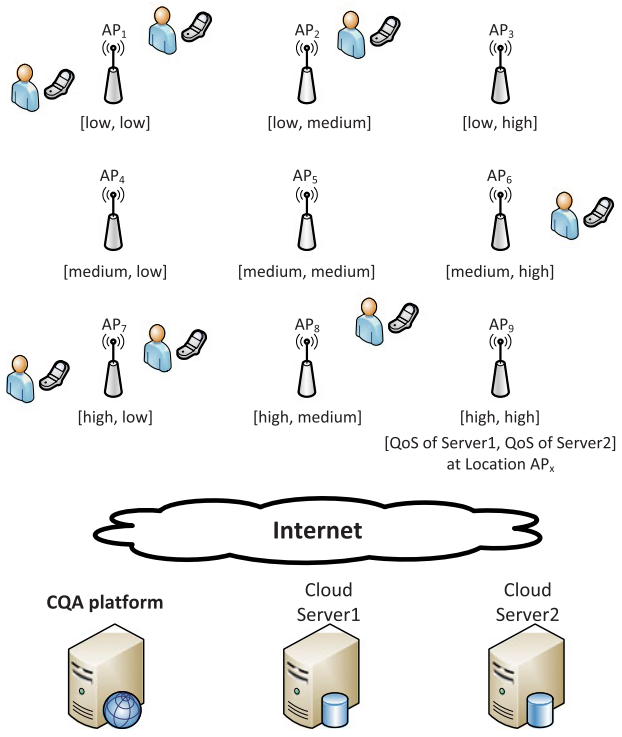

Fig. 8. Simulation scenario. [low, medium] means quality of cloud service1 at this location is low level and quality of cloud service2 is medium level. The test case for nine locations is illustrated with different quality of service performance.

cloud service at different locations. As the user will choose different types of network service carrier and service distance problem, the cloud service will result in different Quality of Service for the user. We will fully consider all these situations in our case study.

Currently, there are three types of cloud services: IaaS (Amazon, VMWare), PaaS (Google App Engine), SaaS (iCloud, Hotmail). Therefore, we consider three cloud services in our experiment environment. Each service has three levels (low, medium, high) of quality service at different locations. The location is an area with an Access Point (AP), where a mobile device can connect to networks. The AP connects to cloud services via a wired connection. Considering all the possible service qualities at each location, there are nine different location places where each cloud service can only provide one kind of quality service. For example, at location AP_1, the quality of each cloud is [low, low], and at location AP_2, the quality of each cloud is [low, medium]. So the nine different locations will have nine different combinations of service quality.

The scenario area is 500 $\times$ 500 m. The nine APs are randomly located at different places. The signal range is 10 m. There are 50 mobile devices randomly moving in this area. 25 mobile devices use crowdsourcing platform and others use context-awareness mechanism to supply QoS of cloud service. Each device will connect to the network when it is in the signal range of an AP. The device will randomly stay 0$\sim$3h within this location. When a device is covered by two location ranges, it will choose the best single one.

The weight vector W is initialled by mobile client. The user can set each quality constraints manually for different applications or learn from previous setting history. In our experiment, we give pre-configured settings for each test applications. For steaming application, we let $W = W_b$. For gaming applications, we choose W_r. For sync application, we use W_p.

TABLE 2
Experiment Parameters

Parameter	Value
Area	500$\times$500 m
Last time	24 h
No. of AP	9
Signal range	10 m
Network bandwidth	1, 10, 100 mbps
No. of mobile user	50
Mobility	0,0.5,1,2,3 m/s
Standstill period	0$\sim$3 h
No. of service request	0.1 per min
No. of cloud service provider	2
No. of Crowdsourcing platform	1

We used NS-3 to simulate the whole environment. All the parameters we used in our experiment are listed in Table 2. The two applications: online media and photo backup are installed in each mobile node. The two cloud services supply CBR streaming data transmission on two node servers. The platform runs in another node to collect context information and responds to service query.

The algorithm running on crowdsourcing platform is described in Section 3.1. We use the following two cases to evaluate the efficiency of our platform.

6.1.1 Discovery Time Comparison

In this case, we focus on analyzing the response interval between the cloud service sent and the final cloud service is established. At the startup, there is no context information on the crowdsourcing platform and the mobile node is randomly located at different places. The node using context-awareness method will test the quality of cloud service one by one. The node will sense the network environment and the quality of cloud provider at current location. After one node's awareness task complete, the adaptor will send the results and context information to the platform. The other node will speed up the new awareness tasks. The node using crowdsourcing platform will save more awareness time when there are more cloud service providers.

We will use the package record to analyze the performance of the platform. From the log file we can observe how the response times change with time. The comparison between traditional context-awareness method and crowdsourcing method on service discovery query time is shown in Fig. 9. Our method is more efficient than traditional context-awareness method.

The service discovery time and accuracy depend on the context information the CQA platform gathered. The more users update the data, the more complete the contextDB will be. The impact of number of user on service discovery time is shown in Fig. 10. The crowdsourcing platform provider more efficient services after more users join the platform. As the traditional context-awareness method is running individually, the performance is not impact by the number of users.

From Fig. 11, we can see that our method is more stable than the traditional context-awareness method. The service

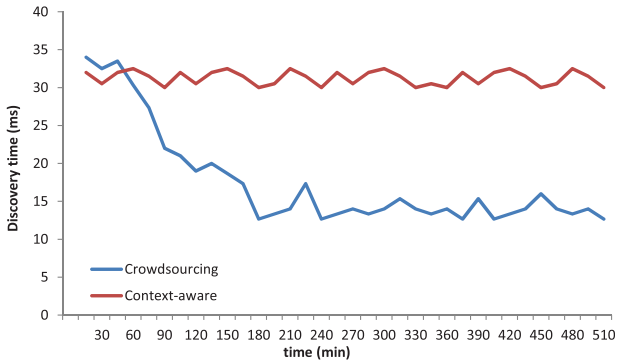

Fig. 9. Service discovery time comparison between two schemes. Num. of User = 50, Mobility = 1 m/s.

discovery time is not influenced by the user's mobility. As the user is moving fast, the cloud service's quality may out of date after the traditional context-awareness sensing complete. The CQA platform can discovery an optimal result in a shorter time.

6.1.2 Stability Evaluation

As the information can not share with each other in the traditional context-awareness sensing system, it wastes lots of computational and network resources. Each user needs to aware the environment to discovery a suitable service, which will cause large amount of traffic in network. The crowdsourcing platform CQA can avoid this. The mobile users can use other's awareness results to discovery a suitable service. This can save a lot of network resources. The Fig. 12 illustrates the network overhead for the two schemes when the user changes. The service discovery message grows quickly when the number of user increases.

6.2 Observation and Analysis

In order to reduce the experimental results of causal factors, each case was tested 10 times and the average result calculated. Fig. 9 shows the response time, when an application queries for service until the application receives the cloud service's response. As we can see in Fig. 9, there is no difference at the beginning of the simulation. However, when the crowdsourcing platform gathers enough context data for a location, the service discovering time is reduced rapidly. After all the locations' data were collected, the response time reached a stable level. The nodes using context-awareness method still need to test the quality performance of each cloud service. Thus, the response time could not be reduced.

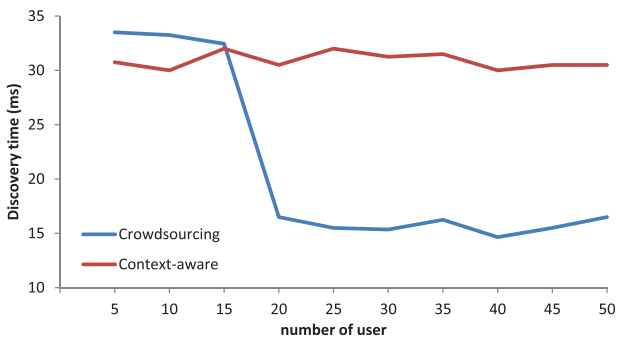

Fig. 10. The impact of number of user on service discovery time. Mobility = 1 m/s.

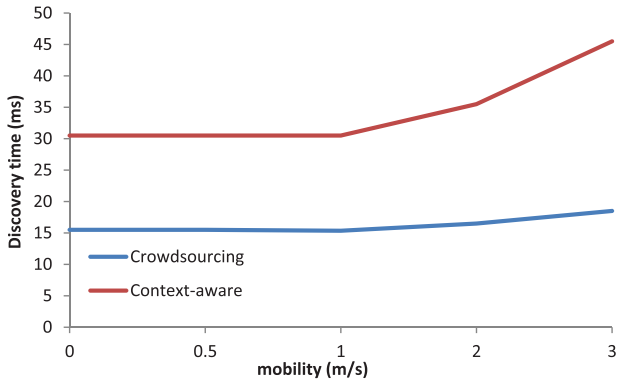

Fig. 11. The impact of mobility on service discovery time. Num. of User = 50.

6.3 Discussion

Fig. 9 shows that the capability of crowd users is more powerful than single user. Crowdsourcing platform can solve the complex cloud services discovering and evaluation problem nearly half times faster than individual. It builds the knowledge database to speed up the QoS evaluation works. The user does not need to run the complex evaluation tasks locally.

Although we use a third-party platform to gather users' context information, we do not record users' identity information. The proposed system just sense the environment situation and cloud providers' performance at different location. Mobile users do not need to have privacy concern about our system.

7 Conclusions

In this article, we briefly introduced crowdsourcing based QoS adaptor (CQA), and its key components and QoS control structures. It can be applied to mobile cloud computing environments in order to provide QoS management for cloud service. We presented the system design, together with its implementation. The context parameters associated with the concept are also discussed. We explained how CQA intelligently provides QoS control using context-awareness method's results. The simulation results show that the crowdsourcing based awareness method can reduce the cloud service discovery time than the traditional local context awareness method, especially for frequently moving user. Current work also denotes that the crowdsourcing model is an efficient way to solve massive parallel task.

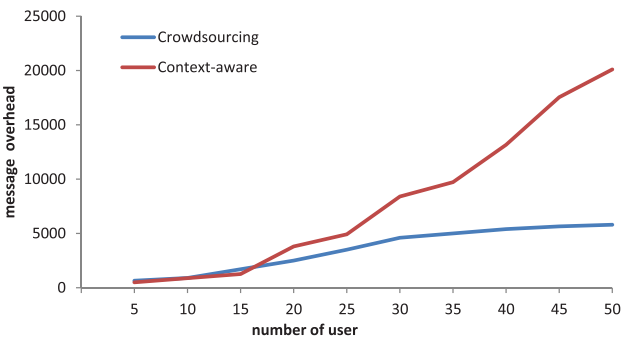

Fig. 12. Network overhead comparison between two schemes. Mobility = 1 m/s.

ACKNOWLEDGMENTS

The work is partly supported by NSFC (No.61472149) and Technology Innovation Fund of Huazhong University of Sci. and Tec. (No.CXY13Q018).

REFERENCES

[1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, 2011, pp. 1607–1623.

[2] P. Papakos, L. Capra, and D. S. Rosenblum, "Volare: Context-aware adaptive cloud service discovery for mobile systems," in *Proc. 9th Int. Workshop Adaptive Reflective Middleware*, Nov. 30, 2010, pp. 32–38.

[3] I. Roussaki, N. Kalatzis, N. Liampotis, P. Kosmides, M. Anagnostou, K. Doolin, E. Jennings, Y. Bouloudis, and S. Xynogalas, "Context-awareness in wireless and mobile computing revisited to embrace social networking," *IEEE Commun. Mag.*, vol. 50, no. 6, pp. 74–81, 2012.

[4] D. Yao, C. Yu, A. K. Dey, C. Koehler, G. Min, L. T. Yang, and H. Jin, "Energy efficient indoor tracking on smartphones," *Future Gener. Comp. Syst.*, vol. 39, pp. 44–54, 2014.

[5] J. Wu, I. Bisio, C. Gniady, E. Hossain, M. Valla, and H. Li, "Context-aware networking and communications: Part 1," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 14–15, 2014.

[6] S. Nath, "ACE: Exploiting correlation for energy-efficient and continuous context sensing," *IEEE Trans. Mobile Comput.*, vol. 12, no. 8, pp. 1472–1486, Aug. 2013.

[7] H. Simula, "The rise and fall of crowdsourcing?" in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 7-10, 2013, pp. 2783–2791.

[8] N. Madnani, J. Tetreault, M. Chodorow, and A. Rozovskaya, "They can help: Using crowdsourcing to improve the evaluation of grammatical error detection systems," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics: Human Lang. Technol.*, Jun. 19-24, 2011, pp. 508–513.

[9] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *Proc. 18th ACM Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 22-26, 2012, pp. 269–280.

[10] Y. Zhao and Q. Zhu, "Evaluation on crowdsourcing research: Current status and future direction," *Inform. Syst. Frontiers*, vol. 16, no. 3, pp. 417–434, 2014.

[11] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service composition for mobile environments," *Mobile Netw. Appl.*, vol. 10, no. 4, pp. 435–451, Aug. 2005.

[12] Y. Liu, J. Wu, Z. Zhang, and K. Xu, "Research achievements on the new generation internet architecture and protocols," *Sci. China Inform. Sci.*, vol. 56, no. 11, pp. 1–25, 2013.

[13] P. Damián-Reyes, J. Favela, and J. Contreras-Castillo, "Uncertainty management in context-aware applications: Increasing usability and user trust," *Wireless Personal Commun.*, vol. 56, no. 1, pp. 37–53, 2011.

[14] M. Dong, T. Kimata, K. Sugiura, and K. Zettsu, "Quality-of-experience (qoe) in emerging mobile social networks," *IEICE Trans. Inform. Syst.*, vol. 97-D, no. 10, pp. 2606–2612, 2014.

[15] D. Chalmers and M. Sloman, "QOS and context awareness for mobile computing," in *Handheld and Ubiquitous Computing*, H.-W. Gellersen, Ed. New York, NY, US: Springer, 1999, vol. 1707, pp. 380–382.

[16] K. Wac, A. Van Halteren, R. Bults, and T. Broens, "Context-aware QOS provisioning in an M-health service platform," *Int. J. Internet Protocol Technol.*, vol. 2, no. 2, pp. 102–108, 2007.

[17] P. Makris, D. Skoutas, and C. Skianis, "A survey on context-aware mobile and wireless networking: On networking and computing environments' integration," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 362–386, 2013.

[18] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Gener. Comp. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[19] D. Yao, C. Yu, H. Jin, and J. Zhou, "Energy efficient task scheduling in mobile cloud computing," in *Proc. 10th IFIP Int. Conf. Netw. Parallel Comput.*, Sep. 19–21, 2013, pp. 344–355.

[20] A. M. Otebolaku and M. T. Andrade, "Supporting context-aware cloud-based media recommendations for smartphones," in *Proc. 2nd IEEE Int. Conf. Mobile Cloud Comput., Serv., Eng.*, Apr. 8–11, 2014, pp. 109–116.

[21] H. J. La and S. D. Kim, "A conceptual framework for provisioning Context-aware mobile cloud services," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 5–10, 2010, pp. 466–473.

[22] A. Vallejo, A. Zaballos, J. Selga, and J. Dalmau, "Next-generation QOS control architectures for distribution smart grid communication networks," *IEEE Commun. Mag.*, vol. 50, no. 5, pp. 128–134, 2012.

[23] J. Wu, I. Bisio, C. Gniady, E. Hossain, M. Valla, and H. Li, "Context-aware networking and communications: Part 2," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 64–65, 2014.

[24] N. Eagle, "Txteagle: Mobile crowdsourcing," in *Proc. 3rd Int. Conf. Internationalization, Des. Global Develop.*, Jul. 19–24, 2009, pp. 447–456.

[25] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Proc. 18th ACM Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 22–26, 2012, pp. 173–184.

[26] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based crowdsourcing: Extending crowdsourcing to the real world," in *Proc. 6th Nordic Conf. Human-Comput. Interaction*, Oct. 16–20, 2010, pp. 13–22.

[27] T. Kumrai, K. Ota, M. Dong, and P. Champrasert, "An incentive-based evolutionary algorithm for participatory sensing," in *Proc. IEEE Global Commun. Conf.*, Dec. 8–12, 2014, pp. 5021–5025.

[28] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[29] H. Bao and W. Dou, "A QOS-aware service selection method for cloud service composition," in *Proc. 26th IEEE Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, May 21–25, 2012, pp. 2254–2261.

[30] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 20–27, 2012.

[31] P. Angin and B. Bhargava, "Real-time mobile-cloud computing for context-aware blind navigation," *Int. J. Next-Gener. Comput.*, vol. 2, no. 2, pp. 89–101, Jul. 2011.

[32] M. Souil and A. Bouabdallah, "On QOS provisioning in context-aware wireless sensor networks for healthcare," in *Proc. 20th Int. Conf. Comput. Commun. Netw.*, Jul. 31–Aug. 4, 2011, pp. 1–6.

[33] Z. Ye, A. Bouguettaya, and X. Zhou, "QOS-aware cloud service composition based on economic models," in *Proc. 10th Int. Conf. Service-Oriented Comput.*, Nov. 12–15, 2012, pp. 111–126.

[34] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 870–883, Jun. 2013.

[35] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in Modeling and Tools for Network Simulation, 2010, pp. 15–34. [Online]. Available: http://www.nsnam.org/

Dezhong Yao (S'15) received the BS degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006. He is currently working toward the PhD degree in the Service Computing Technology and System Laboratory and the Cluster and Grid Computing Laboratory, HUST. From 2010 to 2012, he was a visiting scholar at the Human-Computer Interaction Institute (HCII), Carnegie Mellon University, Pittsburgh, PA, under the guidance of Professor Anind K. Dey. His research interests include ubiquitous computing, context-aware computing, wireless networks, and data mining. He is a student member of the ACM and the IEEE.

Chen Yu (M'09) received the BS degree in mathematics and the MS degree in computer science from the Wuhan University, Wuhan, China, in 1998 and 2002, respectively, and the PhD degree in information science from the Tohoku University, Sendai, Japan, in 2005. From 2005 to 2006, he was a Japan Science and Technology Agency postdoctoral researcher with the Japan Advanced Institute of Science and Technology. In 2006, he was a Japan Society for the Promotion of Science Postdoctoral Fellow with the Japan Advanced Institute of Science and Technology. Since 2008, he has been with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, where he is currently an associate professor and a special research fellow, working in the areas of wireless sensor networks, ubiquitous computing, and green communications. He received the Best Paper Award, in the 2005, IEEE International Conference on Communication, and the nominated Best Paper Award in the Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications in 2007. He is a member of the IEEE.

Laurence T. Yang (SM'97) received the BE degree in computer science and technology from the Tsinghua University, China, and the PhD degree in computer science from the University of Victoria, Canada. He is a professor in School of Computer Science and Technology, Huazhong University of Science and Technology and the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, big data. He has published more than 200 papers in peer-reviewed international journals (around one-third on *IEEE/ACM Transactions and Journals*, others mostly on *Elsevier*, *Springer* and *Wiley* Journals). His research has been supported by the National Sciences and Engineering Research Council, and the Canada Foundation for Innovation. He is a senior member of the IEEE.

Hai Jin (SM'06) received the PhD degree in computer engineering from the HUST, in 1994. He is a Cheung Kung Scholars Chair Professor of computer science and engineering at the Huazhong University of Science and Technology (HUST), in China. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He was at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China, in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientists of National 973 Basic Research Program Project of Virtualization Technology of Computing System, and Cloud Security. He has co-authored 22 books and published over 700 research papers. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security. He is a senior member of the IEEE and a member of the ACM.