Utilizing Crowdsourcing to give QoS to Mobile Cloud Computing

SAYYAD SAFA SAMRIN PASHA[1], GOVARDHAN REDDY KAMATHAM[2]

[1]PG Scholar, Dept. Of CSE, *G. Pulla Reddy Engineering College Kurnool, Andhra Pradesh, India.*

[2]Associate Professor, Dept. Of CSE, *G. Pulla Reddy Engineering College Kurnool, Andhra Pradesh, India.*

Abstract:

Quality of Service (QoS) is one of the vital variables for the achievement of cloud suppliers in mobile cloud computing. Context-awareness is mostly known strategy for self-activating awareness with the portable condition and picking the most right cloud supplier. Absence of context data may hurt the clients' confidence in the application supplying it pointless. Thus, cell phones should be constantly aware of the environment and to test the execution of each cloud supplier, which is inefficient and squanders energy. Crowdsourcing is a significant innovation to find and select cloud services so as to give intelligent, efficient, and stable finding of services for portable clients as a result of gathering decision.

This article presents a Crowdsourcing-based QoS strengthen Mobile cloud service structure that fulfills portable clients' fulfillment by recognizing their context data and giving suitable services to each of the clients. As a result of client's activity context, social context, service context, and device context, our structure progressively adjusts cloud service for the requests in various types of situations. The context awareness based management approach efficiency accomplishes a reliable cloud service supported stage to supply the Quality of Service on cell phone.

Key Terms--- Quality of Service, Context-awareness, Crowdsourcing

1. Introduction:

Next generation computing infrastructure, Cloud computing, has represented various difficulties to mobile client user. One of the main worry is the Quality of Service (QoS), to a great extent because of the diversity of types of services and the complexity of the mobile environment. As the clients' portability, they frequently do not have the abilities or information of service providers and network environments in the heterogeneous places. They don't know how to pick the appropriate cloud service individually. A few strategies are implemented for single mobile client to aware local network environments. However, this local context-aware method can only aware constrained environment knowledge and the device need to consistently aware of the environments when they move to another place. This causes battery issues and the reasonable service with suitable QoS may not be found because of constrained context knowledge.To make mobile devices more robust by using distributed online computing resources, mobile cloud computing is expected. They are the primary interest for the business applications of cloud computing using the Platform as a Service (PaaS), Infrastructure as a service (IaaS), and Software as a service (SaaS) conveyance models. In those platforms,

programming and assets are facilitated on the cloud rather than with the customer, who pays for the required assets according to their resource usage. In order to increase the efficiency and solidness of the cloud service for mobile clients, the web service arrangement is presented. Web service composition gives an approach to consolidate basic web services (potentially offered by various suppliers) and esteem included services to meet the needs of users. For a group of candidate services with the same useful capacities, QoS plays an important role in service selection and service composition. While, QoS can help clients to stay away from asset wastage and higher monitory cost, when the service asked for mobile application may surpass the ability of the device in its present context environment. If the cloud services have solid QoS, users only trust mobile cloud computing.

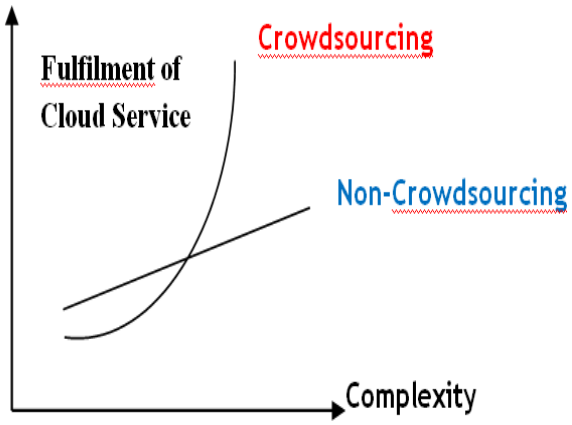

Fig: the relationship between client's fulfillment, crowdsourcing of decider, and awareness complexity

Mobile computing environments are intricate and uncertain. The lack of information of understanding environments and inefficient finding services hurts the clients' Quality of experience (QoE) in the service subsequently rendering it waste. QoE is utilized to measure client's experiences on the whole cloud service experience. Context-awareness has been created and generally utilized in supporting Quality of Service, to optimize QoE. In view of Context information, the Service adaptor (SA) can comprehend mobile environments and intelligently tak decisions to pick right cloud service without interfering the client. To make SA more keen and effective, we need all clients involved to complete this complex issue. Every client will update his or her usage history and context information to a third party platform. At that point, we will utilize aggregate insight to accomplish the smart cloud service picked issue. Crowdsosurcing framework has been built to finish complex information collecting tasks.

Fig. demonstrates the relationship between client's fulfillment, crowdsourcing of decider, and awareness complexity. As shown in Fig. crowdsourcing improves a client's satisfaction of using cloud service. When we have greater sureness of context information, the service adaptor (SA) will be more comprehensible and efficient. Utilizing crowdsourcing technology, we can learn in client habits from history. At that point, we can supply greater quality cloud services for a client. Crowdsourcing can help SA to see more mobile environments and make the correct choice. While there is indirect relationship between service quality and context information, we have to fabricate several suitable mechanisms to accomplish

qualified services for mobile clients.

In this paper, we bring a crowdsourcing based Quality of Service system for mobile cloud service: crowdsourcing based QoS Adaptor (CQA) for heterogeneous types of mobile applications. CQA is intermediate layer that empowers dynamic adaptation of cloud service, and protections crisis service request, efficient asset usage, and savings in monetary provision costs. By observing quality of resource and quality of device, CQA will react to cloud service request for following QoS need level. The most of the activities are controlled by context reasoning part in CQA.

The important contribution of this work are outlined as follows:

- ✓ To supply QoS for mobile cloud service, design a crowdsourcing platform.
- ✓ For picking the best cloud service, propose a crowdsourcing based server discovery schemes.
- ✓ Implement a prototype platform and evaluate the efficiency of crowdsourcing model with traditional schemes.

Remaining part of this article is structured as follows. We briefly present Mobile Cloud Computing and QoS needs. At that point, we display the architecture of the framework CQA and QoS modeling methods. In the following segment, the QoS control algorithms utilized as a part of context reasoning component are depicted in detail. At last, conclusions and upcoming studies are explained.

2. Related works

Quality of Service in Mobile Cloud Computing environment describe the treatment of data as it is exchanged between mobile client and cloud service. The Quality of Service (QoS) in mobile cloud environment measures service in accessibility, need, cost, reaction time, and all through. For the distinctive kinds of cloud service, an operator is needed to be developed to achieve diverse treatment inside the environment for them to function properly.

2.1Mobile cloud Service

Cloud computing is a huge scale distributed network framework in view of various servers in data centers. The models of cloud services can mainly be categorized in based on a layer idea. In the upper layers of this worldview, Infra Structure as service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are stacked. Because of the limitations of cell phone, the typical Mobile Cloud Computing services can be usefulness gathered into two categories:

2.1.1 Storage service:

This service purpose at solving the problem of storage limitations on cell phones. The applications require extensive information transmission between mobile client and server. Net- work accessibly, react time, and all through are the fundamental concerns of this service. Mobile commerce, mobile healthcare, mobile learning, and mobile multimedia are representative applications having a place with this type of service. Mobile commerce for the most part requires low network cost and all through, however high accessibility and reaction time. Some different services may require high QoS for high all through.

2.1.2Computing services:

The computing services move the heavy computing task from cell phone to cloud and achieve the outcomes. The applications unload the task and information to cloud, which is an appropriate solution to deal the problems of computational power and battery lifetime. The common applications which require extensive preparing assets are mobile multimedia, mobile designing and mobile internet gaming. Likewise few of them require high QoS for short reaction time and high throughout.

In addition, some categories of services are don't have guarantee reaction time and priority. These are called best achieved services, for example, email, file reinforcement, and status update.

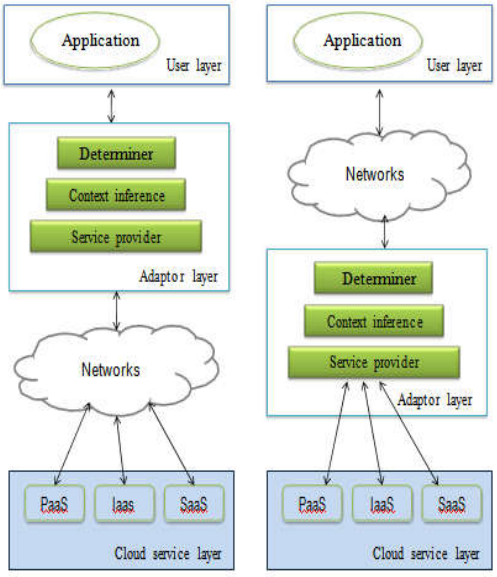

Fig :Two configurations of Context-aware mobile cloud services

As presented above, different types of cloud services may need diverse QoS requirement. To understand completely service request require and present mobile environment. We use Context-awareness method to reach this goal.

2.2Context-aware Mobile Cloud service:

Context-awareness is a fantastic solution to understand mobile conditions and wisely choose the best cloud service. Summarizing, the two fundamental Context-aware mobile cloud services designs, as shown in Fig. Service Adaptor SA running on mobile side (Fig.- a), and SA running on cloud (Fig.-b). At the point when SA is running on a cell phone, the service providers are located in various places, which we called distributed design. At the point when SA is settled on the cloud side, each of the services are gathered together for a cell phone. We named this condition Centralized design.

Context-aware mobile cloud service design contains three layers, which are user layer, adaptor layer and cloud service layer. Adaptor layer is answerable for cloud service matching process. In Contrast cloud services register their service on service provider element in Adaptor layer. Context inference senses client's condition and provider context information for determiner. Determiner chooses the most appropriate cloud service from service provider pool based on current context data.

2.2.1Distributed design:

For distributed design, the service providers required to register in the context service adaptor on a cell phone. The service adaptor will choose to adopt the relevent cloud service based on the context environment to meet the QoS request. As the network environment is always showing signs of change, different providers will perform distinctive QoS on a cell phone. It is more strong for service adaptor to have various options than the

centralized design. In any case, it will spend more on finding and keeping up those services.

2.2.2 Centralized design:

The adaptor layer is situated on the cloud side and running as cloud service. It will assemble the context data from a cell phone and find server candidate services meeting functionalities required by the customer. At that point, it decides the most relevant form of adaptor. Finally, the adapted service is conjured and returns the outcomes to the service customer. As the adaptor running on a remote server, the cell phone may lose the connection to the adaptor.

2.3 Mobile Crowdsourcing:

Crowdsourcing has been well applied in commercial applications: Mechanical Turk, iStockPhoto, and Innocentive. There are likewise a several reaches on mobile crowdsourcing. Create a mobile crowdsourcing framework, that contracts clients with little benefit to finishing simple tasks, for example, translation, transcription and filling surveys. Design incentive mechnisms for mobile user sensing. Considered location context as one parameter for distributing tasks for specialists.

3. Quality of service management:

In this area, an service quality model for mobile cloud computing environment is presented. The cloud service providers and network service carriers are the fundamental elements influencing the Quality of cloud services for mobile clients.

3.1 Service Quality model:

As illustrated in Fig. , the formation of mobile cloud service can be classified into three parts: mobile users, network carriers and cloud service providers. Both the cloud service providers and network service carriers can affect the QoS on mobile devise. To use different cloud services, the mobile users can freely pick various network. 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 service property provided by a cloud service provider and the quality of service at service side are indicated by F and QoS respectively.

The Web Service is the fundamental service gave by various cloud service providers. The fundamental services offer virtual machine service, virtual work desktop service, storage service and so on. Four QoS properties are utilized to depict the abilities of various supplier's servicess, which incorporates computing capacity (CPU), storage (STO), reliability (RL) and price(PR).

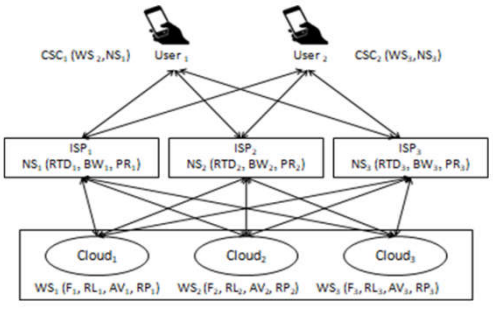

Fig: Mobile cloud service composition model: An example.

Web Service (WS) represents to the number of computing and storage request for the cloud service provider can process every second. The reliability of a cloud service is indicated by RL, which can be assessed by the recurrence of

use. PR is the charge that a client pays for a single service request.

Definition 2 (network service, ns). Network Service is a non-practical portrayal: NS (RTD, BW), which describes the quality of network environment.

RTD is the round-trip postpone time between cell phone and cloud service. BW is the network bandwidth observed from end user for distinct service. A similar cloud service may have distinctive bandwidth through various network transporters. We expect that the mobile client can freely utilize distinctive kinds of wireless network.

The Web Services can't be called by mobile clients straightforwardly. The mobile client needs to utilize the cloud service through various network. The network bandwidth is considered in NS. That is on the grounds that a poor network environment will limit the network bandwidth for mobile clients even through the cloud service provider supports high network bandwidth.

Definition 3 (cloud service context, csc). Cloud Service Context portrays 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 comprises of set of service properties, including web service properties and network service properties as shown in Fig. Those service properties can be assessed at end client's cell phone straightforwardly. As various cloud service provider and network carrier have different service qualities, we have to assess the overall performance and pick the appropriate service providers. In this paper, we assume that the end client can pick any cloud service and any network service whenever.

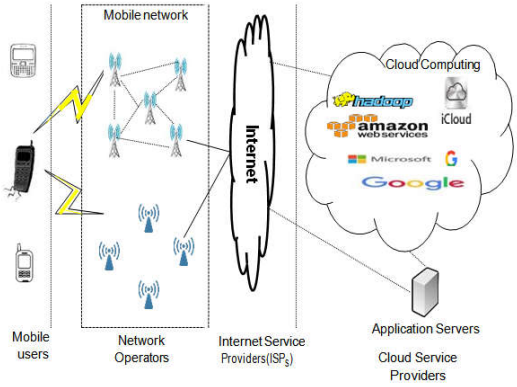

Fig: Mobile Cloud Computing architecture

Diverse kinds of mobile applications have distinctive quality of service requirements. For instance: Streaming applications for the most part worry about network bandwidth, Real time applications focus on network reaction time and Data synchronization applications are sensitive to cost and energy utilization. A defined quality of service is desired for specific kind of application.

Definition4 (Quality of Mobile Service, QMS). Quality of Mobile Service quantitatively measures the quality of each cloud service observe at mobile client To quantitatively measures quality of various services, we use the context information in CSC as the input aspects. Given N cloud service providers and M network service carriers, every mobile client can pick N × M kinds of services as appeared in Fig. Each

cloud service can be associated through various network. From enduser side, the available service dataset is a group of cloud services and network services. Every cloud service and network service information are depicted as a row vector in Eq. (1) and Eq. (2).

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

$$WS=\begin{pmatrix} NS_1 \\ NS_2 \\ \cdots \\ NS_N \end{pmatrix}\begin{pmatrix} RTD_1 & BW_1 \\ RTD_2 & BW_2 \\ \cdots & \cdots \\ RTD_N & BW_N \end{pmatrix} \quad (2)$$

We use a combination matrix S=WS⊗NS to describe all the available services in fig.

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

$$S_{i,j} = (WSi,NSj)$$
$$=(CPUi,STOi,RLi,PRi,RTDj,BWj) \quad (4)$$

The aim of service discovery mechanism is to find an appropriate $S_{i,j}$ in matrix S for end usert in various context environment. All the context properties in matrix S can be quantitatively measured using a real valued. For various application requirements, we just need to rank the performance of a similar context property in one column in S to find the most reasonable services. After we use a quantitative depiction of every context properties in CSC, we normalize every one of the properties in matrix S to make all properties with a similar weight.

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

Without loss of generality, a weight vector W = [PCPU, PSTO, PRL, PPR, PRTD, PSW]^T is used to depict the significance of every property. So the Quality of Mobile Service (QMS) can be processed as:

$$QMS = S_{norm}W$$
$$\text{s.t.Pcpu+PSTO+PRL+PPR +PRTD +PSW = 1} \quad (6)$$

3.2 Context awareness based Service Discovery Model

The quality of cloud service at end client is affected by WS and NS as illustrated in Fig. The context-awareness method is to find an appropriate service combination for mobile client in based on present network environment. As the every combination service quality can be calculated by QMS, to find a appropriate service combination is to find the highest 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 Snorm aware from environment, the quality of every combination service is evaluated by Eq. (6). The weight vector W can be calculated using least squares method based history data or

straightforwardly defined for various applications. The context data matrix Snorm is gathered from context awareness platform.

3.2.1 Service discovery Algorithm:

Confronted with a different network environments, our motivation is to choose an optimal combination service based on application's requirements. The context awareness based service disclosure method can be seen as two stages: context aware and QoS ranking. Given the data of cloud service providers and network provides (ISPs), Context Aware is gathering the QoS performance of each service combination. When the context collection is finished, QoS ranking looks for the optimal combination service by calculating QoS score QMS. The procedure is appeared in **Algorithm 1.**

Algorithm 1: Context-awareness based Service Discovery

Input: W, ISP = {ISP1, ISP2, ...ISPM},

CLOUD = {CLOUD1, CLOUD2,...CLOUDN}

Output: < i , j >

1: Initialization;

2: {Step 1} **Context aware:**
3: for ISPj in ISP do
4: for CLOUDi in CLOUD do
5: Test performance: (WSi , NSj)
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 application diversity requirements, quality of service can be measured in several aspects. Each application has its certain limitations on quality of service. Based on previous studies five main QoS limitations are considered for different quality requirements: bandwidth, reaction time, price, energy and security.

Bandwidth:

Given an service combination, the bandwidth portrays the information transmission ability of network. The weight vector can be set as W_b = [0,0,0,0,0,1]T.

Response time:

Given an service combination, the response time measures the expected delay in seconds between them mobile device sends the request and receives the outcome. The weight vector can be set as Wr = [0,0,0,0,1,0]T.

Price:

The price limit is the fee that a user needs to pay for a solitary service. The mobile client needs to finish a task with a minimum cost. The weight vector can be set as Wp = [0,0,0,1,0,0]T.

Energy:

The energy limit is the power utilization which is spent on an entire service process. The power utilization is influence by the information exchange time. For a specific quantity of information, the network service which has higher band width and reliable service to be chosen. The weight vector can be set as We = [0,0,0.5,0,0.5,0]T.

Security:

The security limit focus on the cloud service's reliability. The cloud service is

evaluated by its uptime and status. The weight vector can be set as Ws = [0,0,1,0,0,0]T.

The mobile application can likewise change the properties of the weights in the vector W for various purposes. Our model is appropriate for a general QoS constraint.

3.2.3. Analysis of discovery time:

Based on general context awareness service finding strategy evaluate the performance of N cloud service providers and M network service carriers. The mobile client tests all the service combination (WSi,NSj) before QoS ranking. The time multifaceted nature to finish all the evaluation is O(N*M).

4. CQA design:

CQA is a platform which continues understanding environment, take service requester based decisions and monitoring resource. It is working as an agent at client side to gather context information of requester. In this part, we will insert each component in the CQA platform.

4.1CQA overview:

Fig. shows a high level view of the CQA functional components and their interdependencies. The CQA system acts as an agent between mobile applications and cloud services. The Adaptor layer is the middleware that gets service request and returns the appropriate service to the requestor. The most important components are given explained below.

Context inference –

The environmental information: user, device and resource contexts are sensed by this component. We can have more clarity on present environment based on the information. User context data comprises of exercises such as activity, location, routine pattern, and social relationship. Device context data comprises of exercises such as network bandwidth, network load, wireless network model, signal strength, and battery life. QoS status elucidate the availability of service, service cost, and reaction time for each cloud service. Context inference gathers the data through the sensors on a mobile device. It gives the essential Algorithms in order to assess the present deficient context based on data obtain from a parent community; for instance, transforming location interconnect to user friendly tags (e.g. home, office, campus, shopping center); and estimation of client's physical activity(e.g. using inertial sensors to establish whether the client is standing, sitting, walking, running, etc.).

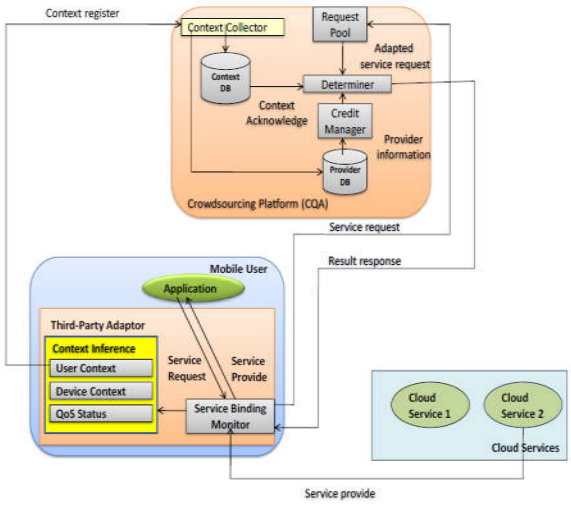

Fig: The architecture of crowdsourcing based QoS supported mobile cloud service

Context collector –

The context data assembled from clients is kept in two databases: Context DB

and Provider DB. The Context DB keeps client context data and device context data. Together with client context, device context outlines an environment context. We assign a unique identity to list for different context environments in Context DB. For every 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 key of the CQA. It plays three mojaor jobs in the CQA: service request planning, environment matching, and provider selection. Service request planning chooses the highly prefered request to run. As the service request has different QoS needs, we give the priority for each request to fulfill the demand. Environment matching is, at runtime, activated by a message from the request pool to find the records of a some of the best match context environment descriptions. Based on the context environment depicted in the service request, the determiner will query the most homogeneous context environments stored in context in DB and generate the identity list. Provider selection chooses the most appropriate cloud service providers in Provider DB to meet present service request. The suppliers are positioned by QoS performance in the request context environment. The decision tree will make the ranking strategy, based on initial rules and history service usage record. We select the suitable providers based on the context environments, which were previously used in this environment. The determiner after selecting the providers will notice the service binding monitor on client side.

Request pool –

This is a single queue whose requests initially act according to first-in-first-out rules. We have a need capacity to assess the significance of the new administration demand and add it to the correct position of the chain. To evaluate the importance of the new service request we have a priority function and add it to the correct position of the chain. Until the previous service releases the resource, certain services will be wait for a minute.

Service binding monitor –

This module is responsible for adjusting a service request to the Broker and monitoring the service in use. The service request is forwarded by service binding monitor and responds to the requestor, when a suitable service is accessible. From each service it collects the usage of resource and reports it to the Determiner. It guarantees QoS for each service.

Service provider –

It combine different cloud services for a mobile device and provide the outline information of each cloud service. We will explain the workflow of our CQA platform, After establishing the components in crowdsourcing platform architecture. There are three important steps to complete the system task as shown in Fig they are 1) Context gathering, 2) Crowdsourcing computing, 3) QoS ranking. We adopt crowdsourcing method to model the context data into the knowledge

database, after collecting enough context data from each client. At that point, we can react to the client's service request by choosing the more relevant provider from the service pool under knowledge database's guidance.

4.2 Context gathering:

Personally, the usage of cloud service report will be updated by users to CQA platform. The cloud service type, performance result and the context environment will be sent to CQA center. As illustrated in Fig, crowd client update their data to the crowdsourcing platform CQA. User's environment, network provider and cloud service provider comes under context data. We set context update 12 hours interval for every client. The user will start a new aware task and update the result to CQA, if one user dose not find any context information for present location. All the context data is unknown in order to protect user privacy. The new cloud service will also register on CQA platform. The more data they collect, the more appropriate the cloud service they will choose. The mobile side context can be depicted as:

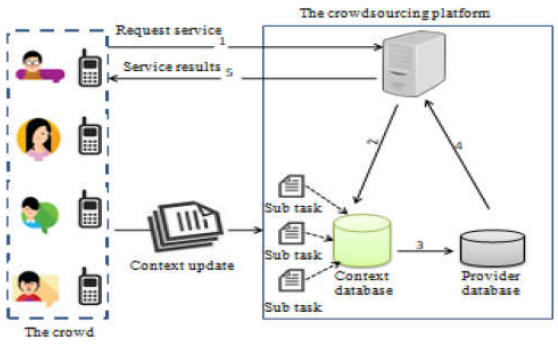

Fig : The workflow of the crowdsourcing based service discovery process.

C = (User, Provider, Performance)
= ([CUM, CUL, CNT], [CPN, CPD], [Si,j]) (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;
- Si,j: performance of cloud service i through network provider j.

We use [CUM, CUL, CNT] to depict the mobile client environment and use [CPN, CPD] for cloud service providers data. The performance of certain providers is assessed by

$S_{i,j} = (WS_i , NS_j)$.

4.3 Service discovery:

The service discovery process can be depicted as an inquiry for the most appropriate result from collecting data. The workflow of crowdsourcing based service discovery is given in Fig. Every mobile client's service usage update will be regarded as a sub task to meet service discovery process. The update information will be stored in a context data base, which depict the connection between context environment and cloud service providers. We can construct a determiner, which can rank the performance of each cloud service in a certain kind of context environment, after gathering enough context data. The relationship between context environment and accessible cloud provider are stored in context database. The QoS performance is additionally joined to this relationship.

The service request consist of two important parts: requestor's data and service constraint.The description of the request can be denoted as:

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

The requestor's data is the client context which is a condition for the service query. After the alternative services retrieved from the crowdsourcing database, the quality of service is calculated under the limitation W. The entire crowdsourcing based service discovery process is compressed in Algorithm 2.

4.4 Service Selection Engine:

The two parameters [User, Performance] need to be provided to query the accessible outcomes, when the user requests a cloud service. After the accessible cloud providers are found, our ranking model calculates the QoS performance. At that point, we select the best rank provider as the outcome and send back to the requestor. The more perfect service we can pick, when the more information we gather. Those steps are appeared in the in Fig.

To choose the most appropriate providers, service selection engine uses the similarity based method. By using the similarity distance, We propose the similarity based decision algorithm to determine the accessible mobile cloud providers in a given context environment. 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 keep running in the requestor's context environment as the two parameters [User, Performance] are given. At that point, we rank the QoS performance using the history record and select the best one as the outcome.

Algorithm 2: Crowdsourcing Based Service Discovery

Input: Q=([CUM, CUL, CNT], [CPN, CPD], W)

Output: < i,j >

1: Initialization;
2: if ContextDB is empty then
3: run Algorithm 1
4: add C=(User, Provider, Performance) to ContextDB
5: return ∅
6: else
7: select $S_{i,j}$ from ContextDB where (C.User = Q.User and C.Provider=Q. Provider)
8: $S_{norm} \leftarrow$ Normalize(S)
9: QMS$\leftarrow$ snormW
10: $< i,j > \leftarrow \arg\max_{<i,j>}(QMS)$
11: return < i,j >
12: end if

The service selection engine (as Determiner in Fig) is running on a third party server in the crowdsourcing platform (CQA).The Determiner in CQA chooses the proper cloud provider for mobile client. When a mobile client needs to convey an application to the cloud, it will send a request to CQA platform. The CQA will use a client centric view of the expected quality to choose the most suitable provider. At last, the outcome will send back to mobile client. The mobile client will evaluate the performance of various cloud service. The CQA will collect those performance reports and chooses the suitable cloud provider for a new requestor.

4.5 Credit manager:

We outline a credit manager component which is responsible for assessing the reliability of each service providers. The credit is the statistical outcome of the successful service times. The crowdsourcing platform will record the accessible service provider in database as shown in Fig. , after the context data C is updated from clients. Base on the provider's uptime and usage frequency, the credit manager calculates the reliability score RL for each service WS.

$$RL = \frac{Frequency\ of\ usage}{Frequency\ of\ attempts\ to\ connect\ to\ the\ service} \quad (10)$$

4.6 Theoretical analysis:

From Algorithm 2, we can see that the service discovery process is an information matching procedure. The requestor's constraints made selection of service performance records from database. 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. Conclusion:

QoS control structures, crowdsourcing based QoS adaptor (CQA), and its key components are shortly presented in this article. In order to provide QoS management for cloud service, Crowdsourcing can be applied to mobile cloud computing environments. System design with its implementation and context parameters associated with the concept are discussed. By using context awareness method's results we explain how CQA wisely provides QoS control. It is concluded that especially for frequently moving user, the Crowdsourcing based awareness method can reduce the cloud service discovery time than the traditional local user. To solve massive parallel task, Crowdsourcing model is an efficient way.

6. References

1. H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, 2011.

2. H. Simula, "The rise and fall of crowdsourcing?" in *Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS)*, Wailea, HI, USA, January 7-10 2013, pp. 2783–2791.

3. N. Madnani, J. Tetreault, M. Chodorow, and A. Rozovskaya, "They can help: Using crowdsourcing to improve the evaluation of grammatical error detection systems," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, June 19-24 2011, pp. 508–513.

4. Z. Yang, C.Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in Proceedings of the 18th ACM Annual International Conference on Mobile Computing and Networking (Mobicom), Istanbul, Turkey, August 22-26 2012, pp. 269–280.

5. Y. Zhao and Q. Zhu, "Evaluation on crowdsourcing research: Current status and

future direction," Information Systems Frontiers, vol. 16, no. 3, pp. 417–434, 2014.

6. Y. Liu, J. Wu, Z. Zhang, and K. Xu, "Research achievements on the new generation internet architecture and protocols," SCIENCE CHINA Information Sciences, vol. 56, no. 11, pp. 1–25, 2013.

7. M. Dong, T. Kimata, K. Sugiura, and K. Zettsu, "Quality-ofexperience (qoe) in emerging mobile social networks," IEICE Transactions on Information and Systems, vol. 97-D, no. 10, pp. 2606– 2612, 2014.

8. *D. Chalmers and M. Sloman, "Qos and context awareness for mobile computing," in Handheld and Ubiquitous Computing, H.-W. Gellersen, Ed. Springer, 1999, vol. 1707, pp. 380–382.*

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

10. N. Eagle, "txteagle: Mobile crowdsourcing," in Proceedings of the 3rd International Conference on Internationalization, Design and Global Development (IDGD), San Diego, CA, USA, July 19 24 2009, pp. 447–456.

11. D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in Proceedings of the 18th ACM Annual International Conference on Mobile Computing and Networking (Mobicom), Istanbul, Turkey, August 22-26 2012, pp. 173–184.

12. F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based crowdsourcing: extending crowdsourcing to the real

13. World," in Proceedings of the 6th Nordic Conference on Human-Computer Iteraction, Reykjavik, Iceland, October 16-20 2010, pp. 13–22.