International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Review on Various Architectural Models in Mobile Crowdsensing

S. Gunasekaran[1], J. Rathnamala [2]

[1, 2] Coimbatore Institute of Engineering and Technology, Narasipuram Road, Narasipuram, Tamilnadu, India

Abstract: ***Sensors are the basic components of various wireless sensor networks. They are used in many devices like smart phones, music players and in vehicle sensing devices as application. This may lead to the evolution of Internet of Things. In this survey we examine the mobile crowd sensing where a group of mobile crowd sensing application users may sense and collectively share information. Further we present the features and characteristics of the MCS and their application. Also, we compare the existing mobile crowd sensing applications such as Vita and Medusa a programming framework for MCS application.***

Keywords: sensors, crowdsensing, crowdsourcing, environmental, social, infrastructural

1. Introduction

The evolutions of sensors that are used in recent days are embedded with daily computing devices that may result in the change of using embedded internet or the internet of things. Furthest evolution of mobile computing devices such as iphone and google nexus, music players such as ipods, sensor gaming devices such as Wii, Xbox kinect and vehicular sensing networks such as GPS,OBD-II.

The advancement of above devices has become more popular nowadays that provides various sensing facility and wireless capability when connected through internet. Sensing can be determined by two aspects individual and by community. Individual happening where related to sensing a particular device where it can take movement sample such as human activities like running, walking , climbing, and transportation activities such as biking driving, taking a bus riding the subway, ATM transactions, getting provision from a store, cooking and listening music where as community happening such as pollution detection, monitoring rainfall at a particular place, pertaining traffic congestion pot holes in a road can be collected to a common goal.

The community sensing are used in both existing methods of sensing known as participatory sensing or opportunistic sensing. Participatory sensing[1] where an individual can actively participate to report a sensing data such as potholes in the road whereas opportunistic sensing is more individual and user part is less. The remarkable trend in community sensing that may range a wide bandwidth of using participatory and opportunistic sensing[2] where mobile crowd sensing that collect, aggregate and analyse data through mobile devices also it can enable social networking services to incorporate location based services, media tag services and so on. Therefore, combining social networking services with real world sensing methods such as crowd sensing[3].

Social networking not only can provide an ideal platform to encourage mobile users to participate in crowd sensing but also help to improve the context awareness of mobile applications and better assist users in mobile crowd sensing by analysing and utilizing their social contexts. Context awareness[4] and mobile crowd sensing together maximizes the data quality for the requestors of crowd sensing by delivering the personalized services relevant to their application context. Also, it helps to delegate sensing tasks to coordinate the appropriate participants to complete the crowd sensing tasks efficiently.

2. Features of Crowdsensing and Challenges

Crowd-sensing have many advantages over traditional sensors, mobile crowd sensing [5] has many characteristic which is very unique in nature where it creates many opportunities as well as difficulties. The mobile devices which are used nowadays have more storage resources than traditional sensors computing and communication are equipped with multi possibility sensing capabilities. Secondly, there are millions of mobile users who carry the equipment wherever they go and whatever they do and by influencing the application in large scale and with low cost. Next, the data reuse of different applications is different from traditional sensor networks. The data quality that depends upon the accuracy, latency and confidence may change concurrently according to the displacement of mobile devices. In MCS same sensor can be use for different application where in traditional sensor it is intended to a single application.

The main challenge of crowd sensing is the resource utilization which includes energy, bandwidth and computation. The data that are collected by the sensors which is highly dynamic and therefore figuring the energy and bandwidth is difficult, also the prediction becomes more complex than the traditional sensors. Next comes the concurrent access of different application becomes delegate due to the resource allocation (e.g. in Car Tel app the sampling rate is reduced according to the priority of application). The interdependencies between different sensing capabilities are performed due to multi-possibility sensing. Resource consumptions become a trade-off where different data is used for same task.

MCS applications that collects sensor data and some of its data should kept private. There are many solutions provided in the privacy of data in existing applications. For example the anonymous data is deleted before they are sent to the third person, in order to protect the privacy of an individual. Also, by adding some noise to the data before it is sent to a community.

3. Mobile Crowdsensing Applications

Mobile crowd-sensing has various applications that are exist for various research challenges and they are broadly classified into three categories

3.1 Environmental

The Environmental MCS applications are based on the natural environmental aspects like measuring the pollution level and water levels in creeks and also wildlife habitats. For example, prototype Common Sense[6] used to measure various air quality (e.g. CO2 and NOx) and communicate with mobile phones to report. Creek watch was developed by IBM that determines the quality of water. By using these types of technologies we can be able to alert the people with unsafe conditions in a timely manner they can also promote social relationship between them and information is shared in an efficient manner.

3.2 Infrastructure

In Infrastructure application used to measure the public infrastructure example traffic congestion, road conditions etc. To maintain traffic congestion application named CarTel[7] which was developed by MIT and Neircell[8] that was developed by Microsoft Research. Neircell is an application where it not only focuses on traffic congestion but also on the honking level and determines average speed of the vehicles. In Vehicular social networking, there is a type of application which helps to find route (i.e., many people used to travel in the same route at same time everyday which is periodic). By mobile crowd sensing VSN collects and aggregates the traffic congestion potentially and even fuel consumption and shortest route.

3.3 Social

The Social application is entirely based on the individual data of an user example the daily exercise routine(e.g. BikeNet[9]) and Diet Sense used to compare various food habits of an individual by taking pictures and sharing it periodically. Also disease report and crisis management can be included in social application. GeoChat[10] a crowd sourced application used in Ministry Of Health in Cambodia for disease report and responses to outbreaks. The above mentioned examples are MCS applications which are related to our daily lives. But when we examine the cyber physical system and their technical requirements are more. For example in environmental monitoring the sensing applications are heterogeneous and also need some standard architecture to communicate with multiple crowd sensing applications. In highly dynamic networks such as VSN we have to make certain stability and the correctness.

4. Various Mobile Crowdsensing Architecture and its Comparison

The current existing state of MCS application and figure out their drawbacks. At present the MCS applications has only two components. They are (i) one on the device and other (ii) on the backend. At device end the sensors are used to collect the data and propagate it and on the backend to analyse and aggregate data. Figure 1 describes application silos makes difficult to develop and deploy various mobile crowd sensing application.

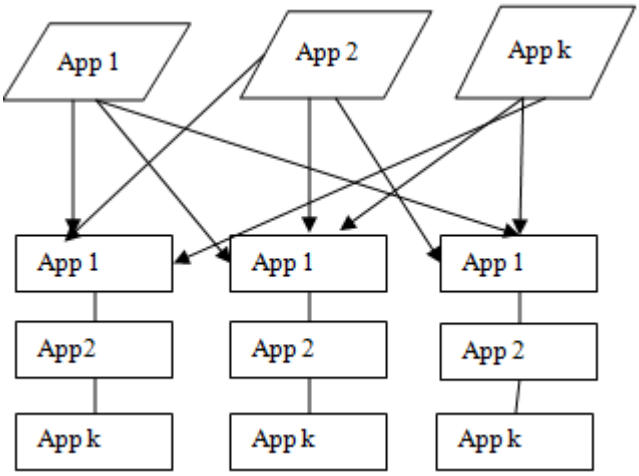

Figure 1: Application Silos[5] where each application is built ground up and independent from each other hence they face many challenges.

The main hindrance in the application silos was to develop program for an application. Because by developing the program we should face various challenges such as energy, privacy and data quality. Also there in need of deploying the various local analytics if it should run on many different types of devices and various types of Operating systems. Secondly, the inefficiency in sensing and processing different applications independently without the knowledge of future consequences, that may result in low efficiency on already resource constraint platform. Lastly, the scalability of the application is less because it cannot accommodate many applications concurrently and to keep track of large number of applications.

5. Medusa

Medusa[11] is a different programming framework which is designed specially to run mobile crowd sensing applications. They provide high-level user abstractions to complete a crowd sensing task. Medusa can be used in participatory sensing system[12] as well as crowd sourcing framework[13]. Figure 2 represents the Medusa system architecture that illustrates crowd sensing application. Medusa is a high-level language for crowd-sensing. In Medusa, programmers specify crowd-sensing tasks as a sequence of stages video documentation task using a description that looks approximately like this:

Recruit -> TakeVideo -> ExtractSummary-> UploadSummary -> Curate -> UploadVideo

This describes the sequence of steps in the task. The Medusa runtime executes these tasks. The design of the Medusa runtime is guided by three architectural decisions that simplify overall system design. Principle #1: Partitioned Services. Medusa should be implemented as a partitioned system that uses a collection of services both on the cloud and on worker smart phones. This constraint follows from the following observation. Some of the requirements are more easily and robustly accomplished on an always Internet-connected cloud server or cluster: task initiation, volunteer recruitment, result storage, and monetary transactions. Others, such as sensing and in network processing, are better suited for execution on the smartphone.

Principle #2: Dumb Smartphones. Medusa should minimize the amount of task execution state that is maintained on smartphones. This design principle precludes, for example, large segments of a task from being completely executed on the smartphone without the cloud being aware of execution progress. We impose this constraint to enable Principle #3: Opt-in Data Transfers. Medusa should automatically require a user's permission before transferring any data from a smartphone to the cloud. Data privacy is a significant concern in crowd-sensing applications. Our principle ensures that, at the very least, users have the option to opt-out of data contributions. Before workers opt-in, they may view a requestor's privacy policy. Discussion: Other designs of the runtime are possible. For example, it might be possible to design a purely peer-to-peer crowd sensing system, and it might also be possible to empower smart phones to exclusively execute crowd-sensing tasks. In our judgment, our architectural principles enable us to achieve all of the requirements outlined in the previous section without significantly complicating system design. Furthermore, these principles do not precisely determine exactly what functionality to a place on the cloud or on the phone: our design is one instantiation that adheres to these principles.

5.1 Evaluation of Medusa

5.1.1 Concurrent Task Instance

Task execution in medusa is quite complex where the computational and sensing stages can be interleaved with worker mediation. Thus dynamics of execution are based on the human actions. To illustrate such a property of medusa, they experimented one task in ten prototype. Also, allowed four volunteers to sign in Amazon Mechanical Turk worker account. The task completion took 30 minutes by four volunteer worker. Execution dynamics are transparent to task requestor in Medusa.

5.1.2 Scalability and Overhead

In Medusa the scalability and overhead of the prototype was measured in individual steps by task execution instance and execution is done on both phone runtime as well as cloud runtime. Initially overhead on server to initiate the task tracker was calculated by taking completing task such as requesting the AMT and receiving SMS/MMS to the phone comparatively receiving the SMS/MMS to the phone was more by taking at least 20 seconds. Indeed the second delay was human to sign up for the task.

Latency time between the worker manager and task tracker was eventually more because of network delay. Next, we concentrate on runtime overhead on phone. The main delay of component was based on the time to instantiate the stage implementation binary. The binary alignment methods need to optimize the stage loading.

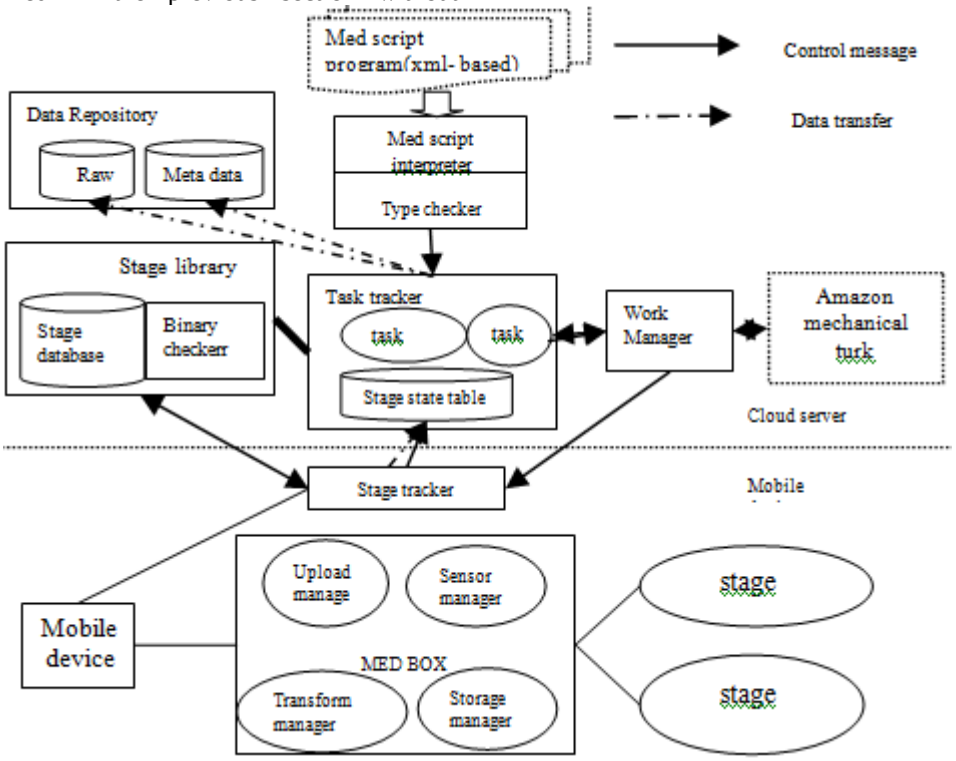

Figure 2: Architecture and working of medusa[11]

5.1.3 Robustness

Medusa robustness was calculated by failure recovery, static analysis on stage binary limiting resource usage. Failure recovery of Medusa was done by switching off the phone during scanning stage where it was scanning a wifi therefore medusa maintains some internal time out recovery mechanism that may caused by either message failure or battery exhaustion or by any user action. It is because of state which is maintained on cloud.

5.1.4 Static analysis on stage binary

They implemented some methods to evaluate the efficiency of the Medusa by producing some malicious code to an application. The attacks was done on the nine modification of code such as browsing(e.g. opening the HTTP connection),starting the sleep mode in the application, accessing the SD card in the mobile. Therefore, the static analyser caught seven attacks and not possible to find attacks like infinite recursion and exhaustion of heap because they require dynamic tracking.

5.1.5 Limiting resource usage

To enforce the limits of usage Medusa created single long running stage which can run to instances at a time. There is a watchdog times in Medusa where it checks the stage resource every three seconds. They mentioned this type of timer to protect CPU from resource exhaustion attacks. Also, excessive network usage can be limited in Medusa by per task basis.

5.2 Advantages Over Other Application

Applications such as participatory sensing system PEIR[15] and sound sense[16] does not involve in human mediation and it will not support in programmable data collection. Privacy aware systems are explored complex data processing and profile based compile time partitioning. Medusa involves both incentive as well as reverse incentive human mediation and support for curation into its framework.

6. Vita

In recent years cyber physical system became more popular where mobile cyber phsical system is a sub category and it is been widely used in smartphones which provides more advanced facility and used in mobile sensing application that creates a bridge between humans and physical world. Vita[17] is a mobile cyber physical system that performs mobile crowdsensing tasks very efficiently. They have created a MCS application called a smartcity to demonstrate funtionalities. At present there are many mobile CPS for MCS application meant to be application specific and /or run on different software and hardware in different capacity. For example, consider two application that was designed for transportation one to verify traffic congestion and other for finiding shortest route. Hence, this is no chance to share the same data. Therefore, it may reduce the interaction between digital and physical world.

Also, there may arise some complicated constraint where it wont work properly at low capacities[18]. The work that was presented in paper[19] is capable to support such application but they may lack in flexibility to meet ubiquitous service requirements. In extent, other than conventional mobile embedded system, which is focused on system computations,MCPS or human centric and insists to human to participate and operate some specific task

Currently the research work on mobile CPS application mostly focus on optimization of computing task[3](e.g. server over provisionig to mobile devices). But, they still lack in producing in efficiently and effectively. Hence by developing MCPS toward crowd sensing that may be very customized with standard and universal standard interaction. Also, the system should support computation which is diverse in optimized manner and to run human based MCPS tasks simultaneously. Additon to it due to the crash of mobile operating system, battery exhaustion, and network disconnection may lead to service failure. There are many solution available already that may not support MCS, Vita proposes mobile distributed system that leverages social networking service by integrating MS2A and also orchestrate a service of open source techniques to develope CPS for multiple and differentmobile crowd sensing application systamatically.

6.1 Architecture of Vita

Architecture of vita provides two platforms: mobile platform and cloud platform. Mobile platform is the initial environment and ubiquitous service to enable users to participate in and operate crowd sensing task on their mobile devices. The cloud platform is the central platform which co ordinates to store and integrate the different data and to develope crowd sensing application.

6.1.1 Mobile SOA Framework

It is configurable which is based on methods of RESTFUL webservices[20] (e.g Facebook, Google+)are some of socialnetwork it intergrates. They are used to support development of multiple mobile web services in an efective and flexible way. SOA acts as a bridge between mobile platform and cloud platform

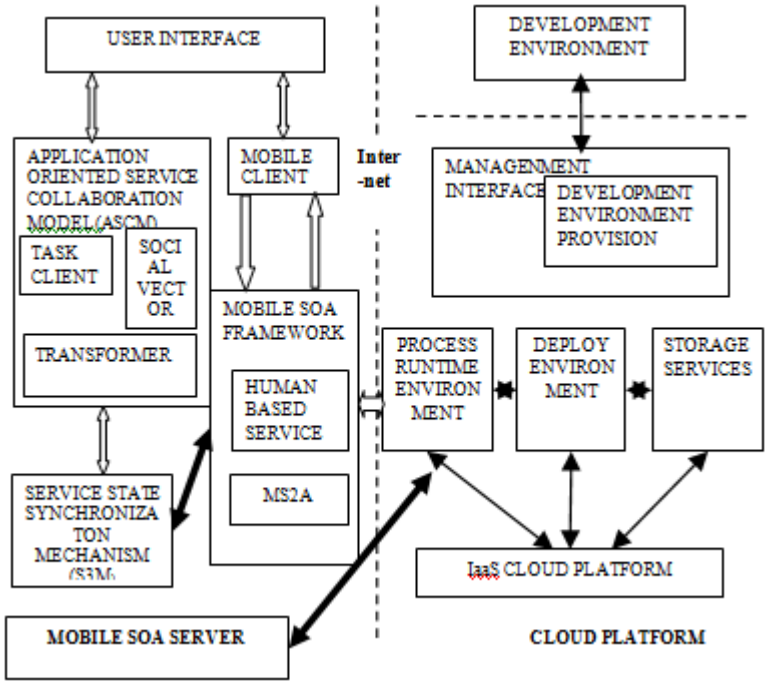

Figure 2: Architecture and working of Vita

6.1.2 Vita cloud platform

The vita proposed cloud platform for it, to adapt RESTful Webserive-based architecture design methodologies and specification, hence to offer open as well as extensible and seamless architecture accross the mobile and cloud platform. They contain four parts: Management Interface, Storage Service, Process Runtime Service, Deployment Environment.

(a) Management Interface:

It is used to provide development and application programming interfaces(API's) to support application developers and also to enable third party service provides to participate in the development of devices in different applications related to mobile crowdsensing. Also, provides open API for social networking(e.g. Facebook) and distribute crowd based social information.

(b) Storage service:

They are useful for storing automatic backup for vita such as software services, installation files of vita in the mobile devices, task lists and resukts of mobile crowd sensing and sensing data uploaded by mobile devices through some popular social networks.

(c) Deployment environment:

As in need of supporting their users participation in different mobile crowdsensing applications, it enables dynamic deployment of mobile platform and various web services of vita to different mobile device according to their capacities and practical application requirements.

(d) Process runtime environment:

It is based on open source techniques, and provides the crowd sensing platform, which could co-ordinate, process and combine multiple crowd-sensing results from different mobile devices in real-time. Thus, compared to traditional SOA-based solutions for developing and deploying mobile applications, one advantage of this architecture design is that the application developers do not need to be concerned with mapping relations about specific service requests to corresponding service resources, but can focus on the development of the application itself. For example, the work flow of application development in Vita consists of five steps,

(i) Once the Deployment Environment Provision model on the cloud platform receives the development environment request with configuration, it can automatically deploy all of the components of the software needed to construct the development environment according to the assessment of the developer's software and hardware configurations, so as to set up the development environment for the application developer who intends to develop new services based on the Vita system.

(ii) Based on the service-oriented programming model of the mobile SOA framework, the developers can easily and efficiently implement diverse mobile crowd sensing applications. After a developer has finished the development, s/he can submit the deployment request with the related software package and configuration file. Further, the Deployment Environment analyzes whether the software package can be deployed to the cloud platform and/or to a mobile device.

(iii) If the software package needs to be deployed on the cloud platform, the Deployment Environment will deploy it to the Process Runtime Environment according to the specification of BPEL and the service description of the Web Service Definition Language (WSDL). After the Process-Runtime Environment receives and finishes the related deployment request, it will return the deployment result to the Deployment Environment.

(iv) If the software package needs to be deployed on mobile devices later, the Deployment Environment will deploy the

related software components and data by submitting the request message consisting of (package, data) to the Storage Service module. Then the Storage Service module will store them in the cloud infrastructure and return the related deployment result.

(v)Finally, the Deployment Environment will return the deployment result to Development Environment, and present the results to the developer.

6.1.3 Application-Oriented Service Collaboration Model

Application-oriented service collaboration model is a novel resource optimization mechanism used to allocate human based tasks among individuals and computing tasks between mobile devices and cloud computing platform in an efficient and effective manner. It consists of three components namely:

(a) Task client

It acts as a bridge between the ASCM and the mobile SOA framework. They receive the request from the client and invoke the existing services in SOA framework to finish the task. Also, the task client can invoke the social networking services in the mobile SOA framework to social information.

(b) **Transformer**

It is used to transfer the diverse application requests and tasks users to a standard format as a web service.

(c) Social vector

The social vector obtains the tasks information from the Transformer and task client helps to receive the service and social information from mobile SOA framework. To facilitate the deployment of computing tasks and human based tasks among different devices and mobile crowd sensing application the social vector quantifies the distance and relationship between two physical elements and virtual elements such as software services.

6.1.4 Evaluations

(a) Development support

The standalone lines of code (LOC) for the implementation of the applications are 275 and 326. Aside from the codes to implement the web services, other codes are mainly XML-based (for real-time service composition in mobile devices). Even though the LOC looks longer than that of the web services taken together, this is not indicative of the development effort, which is much smaller than the latter. Also, because the applications are developed in Vita, which is based on the standard specifications of RESTful web service, the services can readily be reused and composited. Thus, the total LOCs needed to implement these two applications together are less than the sum of LOCs to implement them individually.

6.1.5 Comparison over Medusa

(i) Vita is flexible distributed system that co-ordinate between mobile and cloud platform. It dynamically balances the allocation of computation concurrently between mobile and cloud platform to specific application requirements and scenario also efficient allocation of human based tasks.
(ii) Amazon AMT adopted by Medusa in the execution of crowd sensing tasks can only be used inside the USA.AMT commercial use and its not open source.
(iii) It does not support the customized crowd sensing mechanism and platform by third parties, while vita adopts open source techniques and industrial standards such as BPEL4 people, and leverages the advantages of social networking services to set up the cloud platform, which is therefore be more universally used and supports flexible extension and customized redevelopment.

The vita also has some disadvantages over medusa, There is not enough of incentive mechanism where people could not effectively participate in crowdsensing. Secondly, the security mechanism is another major criterion of Vita where it does not include any security in it. But Vita has open architecture that is easily adopted by developer to design various customized mechanism to address these types and issues.

Table 1: Evaluation of Vita and Medusa

PARA-METER	*DATASET OF VITA AT VANCOUVER*		*DATASET OF VITA AT HONG KONG*		*MEDUSA*
Time delay	Common Avg: 10577 Min: 2399 Max: 22636	With S3M Avg: 12804 Min: 10388 Max: 21319	Common Avg: 10577 Min: 2399 Max: 22636	With S3M Avg: 12804 Min: 10388 Max: 21319	Delay breakdown on server (msec): Avg: 63988.86 Min: 40617.16 Max: 138758.3
Battery Consumption	80 mAh / 45mins	140mAh / 45mins	80 mAh / 45mins	140mAh / 45mins	N / A
Network overhead	0.85 MB / 220 requests	1.27 MB / 216 requests	0.83B/ 216 Requests	1.23 MB / 223 requests	Delay break down on the phone (msec): Avg: 459.8 Min: 598 Max: 409

7. Conclusion

Mobile crowd sensing has created many application oriented platform that gives flexible environment for smartphone users. Medusa and Vita are example of MCS application where they are evaluated with various parameters. Medusa is a programming frame work that uses Medscript programming language that gives high level of abstraction for crowd sensing tasks. The task description are concise and the run time system has low overhead. Vita is a novel approach which creates many advantages of social computing, service computing and many open source techniques to provide a systematic approach to both users and developers.

References

[1] J. Burke et al., "Participatory sensing ," Workshop on World-Sensor-Web, co-located with ACM SenSys, 2006. [Online].Available:http://www.sensorplanet.org/wsw2006/
[2] N. Lane et al., "A survey of mobile phone sensing," IEEECommunications Magazine, vol. 48, no. 9, pp. 140–150, 2010.
[3] G. Cardone et al., "Fostering Participaction in SmartCities: A Geo-Social Crowdsensing Platform," IEEE Commun.Mag., vol. 51, no. 6, June 2013, pp. 112–19.
[4] M. Raento et al., "ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications," IEEE.Pervasive Computing, vol. 4, no.2, 2005, pp. 51–59.
[5] R. K. Ganti, F. Ye, and H. Lei, "Mobile Crowdsensing:Current State and Future Challenges," IEEE Commun.
[6] P. Dutta et al., "Demo abstract: Common sense: Participatory urban sensing using a network of handheld air qualitymonitors," in Proc. of ACM SenSys, 2009, pp. 349–350.
[7] B. Hull et al., "Cartel: a distributed mobile sensor computing system," in Proc. of SenSys, 2006, pp. 125–138.
[8] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in Proc. of ACM SenSys, 2008, pp. 323–336.
[9] S. B. Eisenman et al., "The bikenet mobile sensing system for cyclist experience mapping," in Proc. of SenSys, November 2007.
[10] InSTEDD. (2006). GeoChat, Sunnyvale, CA, USA [Online]. Available:http://instedd.org/technologies/geochat/
[11] M. Ra, B. Liu, T. Porta, and R. Govindan, ``Medusa: A programming framework for crowd-sensing applications," in Proc. 10th Int. Conf.Mobile Syst., Appl. Services (MobiSys), 2012.
[12] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin,M. Hansen, E. Howard, R. West, and P. Boda. "Peir, thepersonal environmental impact report, as a platform for participatory sensing systems research". In Proc. ACMMOBISYS, 2009.
[13] Amazon mechanical turk, https://www.mturk.com/.
[14] C. Dwork. Differential privacy. In ICALP, pages 1–12.Springer, 2006.
[15] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin,M. Hansen, E. Howard, R. West, and P. Boda. "Peir, the personal environmental impact report, as a platform for participatory sensing systems research". In Proc. ACM MOBISYS,2009
[16] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In Proc. ACM MOBISYS,2009.
[17] X. Hu et al., "Vita: A Crowdsensing-Oriented Mobile Cyber Physical System," IEEE Trans. Emerging Topics in Computing, vol. 1, no. 1, June 2013, pp. 148–65.
[18] J. White, S. Clarke, B. Dougherty, C. Thompson, and D. C. Schmidt, ``R&D challenges and solu-tions for mobile cyber_physical applications and supporting internet services," J. Internet Services Appl., vol. 1, no. 1,pp. 45_56, May 2010.
[19] M. Demirbas, M. Bayir, C. Akcora, and Y. Yilmaz, ``Crowd-sourced sensing and collaboration using twitter," in Proc. IEEE Int. Symp. WoWMoM,Jun. 2010, pp. 1_9.
[20] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful Web Services vs. 'Big' Web Services: Making the Right Architectural Decision," Proc. WWW, 2008, pp. 805–14.

Author Profile

S.Gunasekaran received the B.E. and M.E. degrees in Computer Science and Engineering from Kumaraguru College of Technology in 1997 and 2003 , respectively. During 2007-2010, he started Phd research in Computer Science and Engineering at Anna University, Coimbatore, India. His area of interest are Cloud computing, MANET, Data Mining and Data Warehousing. His professional Experience is about 10 years. Presently, working as Head of the Department at Coimbatore Institute of Engineering and Technology TamilNadu,India.

J. Rathnamala received the B.E graduation in Computer Science and Engineering from Ranganathan Engineering College in 2009-2012 respectively and currently pursuing Master of Engineering in Computer Science and Engineering at Coimbatore Institute of Engineering and Technology, TamilNadu, India.