A Mobile Crowdsensing System Enhanced by Cloud-based Social Networking Services

Xiping Hu[1], Qiang Liu[2], Chunsheng Zhu[1], Victor C.M. Leung[1], Terry H.S. Chu[3], Henry C.B. Chan[3]

[1] Dept. of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada

[2] School of Computer, National University of Defense Technology, Changsha, China

[3] Dept. of Computing, The Hong Kong Polytechnic University, Hongkong

{xipingh, qliu, cszhu, vleung}@ece.ubc.ca, {cshschu, cshchan}@comp.polyu.edu.hk

ABSTRACT

This paper presents TripleS, a novel mobile crowdsensing system enhanced by social networking services, which enables mobile users to participate and perform mobile crowdsensing tasks in an efficient manner. TripleS provides a flexible and universal architecture across mobile devices and cloud computing platforms by integrating the service-oriented architecture with multi-agent frameworks for mobile crowdsensing, with extensive supports to application developers and end users. The customized platform of TripleS enables dynamic deployments and collaborations of services and tasks during run-time of mobile devices. Our practical experiments show that TripleS performs its tasks with a considerable computation efficiency, and low computation and communication overhead on mobile devices. Also, the mobile crowdsensing application developed on TripleS demonstrates the functionalities and practical usage of TripleS.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: COMPUTER COMMUNICATION NETWORKS - Distributed Systems

General Terms

Design, Human Factors, Economics.

Keywords

Mobile crowdsensing, cloud, social networks, system architecture

1. INTRODUCTION

In our daily lives, the ubiquitous and popular usage of mobile devices such as smart phones and tablets with integrated sensors such as cameras, Global Positioning System (GPS) receivers and accelerometers results in many available sources of sensing data. The potential use of such sensing data to enable sophisticated context-aware mobile applications and services has already motived a variety of innovative research.

Mobile crowdsensing refers to a broad range of community-based sensing paradigms employing mobile devices and wireless networks [1]. Different from conventional sensing solutions using specialized networks of sensors, mobile crowdsensing leverages human intelligence to collect, process, and aggregate sensing data using individuals' mobile devices (e.g., using a camera to capture a specific target), so as to realize a higher quality and more efficient sensing solution. In general, crowdsourcing [2] aims to provide an efficient and effective mechanism to involve participants in the general public to solve specific problems in collaborations. Mobile crowdsensing is a special form of crowdsourcing that solves specific problems or answer specific questions by utilizing sensing data from participants' mobile devices.

A remarkable trend in mobile services is the increasing use of mobile devices to access social networking services. The wide availability of sensing modules in mobile devices enables social networking services to be extended to incorporate location based services, media tag services, etc. Therefore there is growing interest in fusing social networking services with real-world sensing, such as crowdsensing [3]. Social networking can provide an ideal platform to encourage mobile users to participate in crowdsensing. For instance, people recruited in crowdsensing could in turn recruit their friends through the popular social networking services to contribute to a more pervasive crowdsensing solution by greatly increasing the number of participants.

Different from conventional wireless sensor networking applications, crowdsensing applications need to meet the diverse and ubiquitous service requirement of different participants contributing sensing data using many different devices [1]. Currently, most of the existing mobile crowdsensing solutions are application-specified [3-5], and they usually follow different service interaction standards. Consequently, two different crowdsensing applications may not share the same type of sensing data and related services directly. Furthermore, use of complex purpose-built sensing applications may seriously constrain the number of participants in a mobile crowdsensing exercise. The work in [6] that utilizes Twitter to construct a crowdsensing system capable of supporting multiple crowdsensing applications is a step in the right directly. However, this system lacks the flexibility to meet the diverse service requirements of different participants engaged in multiple crowdsensing tasks.

There is a need for a mobile system architecture for mobile crowdsensing with standard and universal service interactions, so as to support the efficient development of different customized mobile crowdsensing applications, and to effectively manage and coordinate the diverse sensing data and tasks of mobile crowdsensing participants during run-time. This architecture should include services that assist users in performing multiple crowdsensing tasks simultaneously on their mobile devices, as users could easily miss some important sensing data due to inability to stay focused on specific sensing tasks. Furthermore, there is a need to integrate such an architecture with social networking services to help expand the scope of crowdsensing participation, ease the dissemination of crowdsensing results, and facilitate user interactions through

MCS'13, December 9- 13, 2013, Beijing, China.

interfaces people are familiar with. To the best of our knowledge, such an architecture does not currently exist. The main objective of this work is to develop and validate an open mobile architecture for ubiquitous crowdsensing that is enhanced by social networking services.

To address the needs identified above that are not efficiently met by existing systems, in this paper, we propose a novel mobile distributed system – social networking services enhanced crowdsensing, TripleS, for mobile crowdsensing. Leveraging the popular social networking services by integrating our previously proposed Aframe [7] and MS2A [8], TripleS adopts a series of open source techniques to provide a systematic approach for mobile crowdsensing in multiple and diverse mobile scenarios. Our experiments show that TripleS has affordable computation and networking overheads, and is efficient and effective for practical applications. Our major contributions are as follows:

- We propose TripleS, a mobile system architecture enhanced by social networking services with software agent support, for mobile crowdsensing across mobile devices and cloud computing platforms, and present its design and implementation. TripleS not only provides an augmented service with regard to speed and quality, but also a comprehensive, flexible, and universal solution that supports both application developers and end users.
- We deploy and evaluate TripleS through a set of real-world scenarios, which not only verify the feasibility of it, but also provide practical experience that inspire future research and development of mobile crowdsensing applications.

The rest of the paper is organized as follows. Section 2 gives some background on mobile crowdsensing applications, and discusses the requirements of mobile crowdsensing systems. Section 3 presents the overall architecture and key components of TripleS, and discusses how it meets the requirements for mobile crowdsensing systems. Section 4 presents strategies for the implementation of TripleS. Section 5 shows practical experiments to evaluate TripleS. Section 6 reviews other systems developed for crowdsensing and compares them with TripleS. Section 7 concludes this paper.

2. BACKGROUND

There are many examples of crowdsensing applications. In this section we first briefly present the existing and potential mobile crowdsensing applications, and then based on these, we discuss various requirements of mobile crowdsensing systems, and propose the techniques we used to develop TripleS to address these requirements.

Vehicular social networking: A large number of people in urban areas spend hours on their daily commute to and from work, traveling along the same routes at about the same time. Their travel patterns are highly predictable and regular. Consequently, there is an opportunity to form recurring virtual mobile communication networks and communities between these travelers or their vehicles, i.e., vehicular social networks (VSNs). Through mobile crowdsensing, a VSN could aggregate travel information to measure the potential traffic congestion [9], find the most appropriate (e.g., lowest fuel consumption, shortest time) route in real-time [10], or provide recommendations of parking spaces available nearby [11].

Disease report and crisis management: As many mobile devices are equipped with an array of sensors, diverse sensing data and citizen reports from mobile devices can be triaged and acted on in real-time by individuals/communities, which will facilitate disease report and crisis management. For instance, based on a crowdsourced interactive mapping application, the Ministry of Health in Cambodia uses GeoChat [12] for disease reporting and to send staff alerts and rapidly escalate responses to potential outbreaks. Also, Ushahidi [13] has been used to crowdsource and map crisis information from multiple sensing data streams in real-time through mobile devices, so as to coordinate field teams' activities and provide remote support from outside the earthquake zone.

The above examples show that mobile crowdsensing is of much practical use in daily lives. However, if we target to develop a mobile crowdsensing system to support these examples, as discussed in Section 1, there are several key technical requirements need to be addressed. In order to address these requirements, in developing the overall architecture of TripleS, we adopt the design principles of REpresentational State Transfer (REST)-ful Web Services [14], which is already widely used for mobile applications and provides an ideal service-oriented architecture (SOA) to support the standard service interactions both for the development stage and at run-time. In addition, we adopt the agent techniques developed in our former work [7] to enable the system to adapt itself to dynamic sensing tasks.

3. SYSTEM DESIGN

As shown in Figure 1, the overall architecture of the TripleS system consists mainly of two parts: the mobile platform and cloud platform. The mobile platform provides the initial environment and ubiquitous services to enable users to

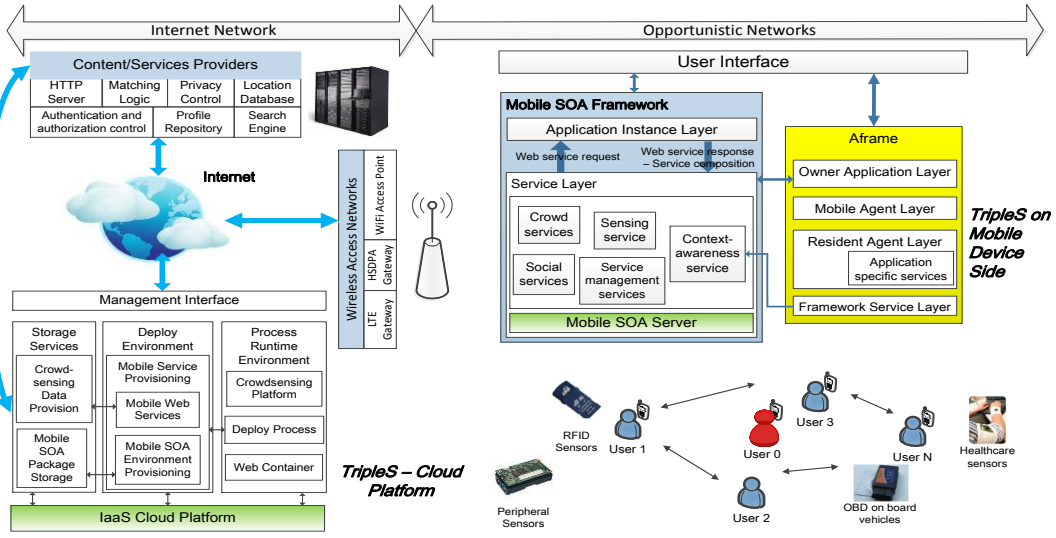


Figure 1. Overall architecture of the TripleS system

participate in and operate crowdsensing tasks through their mobile devices. The cloud platform provides a central coordinating platform to store and integrate the diverse data from mobile crowdsensing and social networking service providers, as well as the development environment to support developing mobile crowdsensing applications. In this section, we present the key components of TripleS, and discuss how they meet the requirements of mobile crowdsensing as stated above.

A. TripleS – cloud platform

For the design of the cloud platform of TripleS, we also adopt RESTful Web Service based architecture design methodologies and specifications, so as to provide a universal and seamless architecture across the mobile and cloud platform of TripleS. The cloud platform of TripleS mainly consists of four components: management interface, storage service, deployment environment, and process runtime environment.

Management interface: It provides the development environment and application programming interfaces (APIs) to support application developers and enable third party service providers to participate in the development of different applications, services, and application-specific platforms for mobile crowdsensing. Also, it makes use of open APIs provided by commercial social network websites like Facebook to access social networking services and disseminate some crowd based social information to the popular social networks.

Storage service: It supports automatic backup of TripleS system data, such as data related to software services, installation files of TripleS in the mobile devices, task lists and results of mobile crowdsensing, and sensing data uploaded by the mobile devices through TripleS.

Deployment environment: It enables dynamic deployments of the mobile platform and various web services of TripleS to mobile devices, so as to support their users' participation in different mobile crowdsensing applications. Also, it could automatically deploy new applications and services uploaded by developers to the process runtime environment for executing diverse mobile crowdsensing applications.

Process runtime environment: It is based on the open source techniques (more details will be provided in Section 4), and provides the crowdsensing platform, which could coordinate, process, and combine multiple crowdsensing results from different mobile devices in real-time.

B. Mobile SOA framework

The mobile SOA framework is an extensible and configurable framework that is based on the specifications and methodologies of RESTful Web Services. It integrates social networking services and adopts an SOA to support the development of multiple mobile crowdsensing applications and services in an efficient and flexible way, with standard service interaction specifications that enable dynamic sensing service collaboration during mobile devices' run-time. As shown in Fig. 1, this framework mainly consists of two layers: the service layer and the application instance layer, both of which can run on the same mobile device simultaneously.

Service layer of mobile SOA framework: This layer is integrated with the mobile SOA server to address the needs of application developers. Using the service-oriented programming model, developers can easily implement a variety of web services based mobile applications and services for crowdsensing. The web services in this layer are enabled by the mobile SOA server. Application developers can take full advantage of SOA to efficiently and flexibly extend the web services, or integrate the exist web services at this layer to generate different mobile crowdsensing applications. Moreover, developers can also design mechanisms or algorithms for service compositions at this layer, so that multiple applications developed on TripleS can automatically and intelligently access, use or collaborate with the running web services shared among the mobile devices at run-time. In the current version, there are mainly five types of web services in this layer.

Sensing services: are used to aggregate and process the sensing data in the mobile devices, such as location information service, metadata of pictures/photos service, and service to obtain a vehicle's status through the onboard diagnostic module. Also, it contains the web service that could receive and process the sensing data collected by mobile agents (through Aframe) from local opportunistic networks.

Social services: make use of the open APIs provide by the popular social networking services like Facebook and Twitter, and encapsulate them to web services. They enable the crowdsensing requests to be posted through popular social networks, which could help to spread the requests quickly and encourage more people to participate in crowdsensing. They can also provide basic social information that could be integrated with other modules and services of TripleS; e.g., these services could enrich the sensing information when integrated with sensing services.

Crowd services: support the posting of crowdsensing tasks, fetching the crowdsensing task list from the cloud platform of TripleS, processing the results of mobile crowdsensing in the local mobile devices, and submitting the crowdsensing results through the Internet to the cloud platform of TripleS for further analysis in combination with the crowdsensing results from other mobile users. In addition, as these services utilize the communication service between the mobile devices and cloud platform of TripleS through the Internet, they could also support the balancing of computation tasks between the mobile devices and the cloud platform of TripleS, so as to improve the task efficiency of crowdsensing and save energy consumption in mobile devices.

Context-awareness service: is based on our former work [9]. It could dynamically and automatically match the sensing data to the appropriate crowdsensing applications during mobile devices' run-time.

Service management services: are used to manage all the available web services in TripleS, such as services for registering new services, monitoring currently available services, and scheduling the services for collaborations. These services also allow application developers to design new mechanisms or extensions that can be incorporated into new applications.

Application instance layer of mobile SOA framework: This layer is oriented towards the realization of concrete applications, and every application instance in this layer can work independently. Application developers only need to develop the user interface with the related application instances in this layer, through which the end users of TripleS can directly access the corresponding application.

C. Aframe

Aframe [7] is an agent-based application programming framework for mobile wireless ad-hoc networks, which we have designed and implemented previously. As shown in Fig. 1, it consists of four layers: framework service layer, resident agent layer, mobile agent layer, and owner application layer. The framework service layer provides the generic services, which support collaborations among multiple agents working in multiple devices simultaneously, such as supporting multiple mobile agents to dynamically and self-adaptively execute different applications around opportunistic networks. The other

three layers are open to application developers. Based on the programming model of Aframe, developers could easily and efficiently develop agent based applications and services, such as developing different mobile agents to automatically and/or opportunistically collect and process the sensing data locally in a mobile device, and developing owner applications to define the strategies about the processing of sensing data that are collected by the mobile agents. In the TripleS system, we mainly adopt the Aframe to automatically and dynamically aggregate and process the sensing data with the services in the mobile SOA framework locally in mobile devices and across opportunistic local networks for mobile crowdsensing.

4. IMPLEMENTATION STRATEGIES

In this section, we discuss how the two main parts of the TripleS system can be implemented in practice.

A. Implementation of cloud platform of TripleS

In the current version of TripleS, we adopt the Amazon Web Service (AWS) infrastructure services (i.e., EC2 and S3) and a series of open source techniques, such as Apache ODE, Apache Tomcat, Business Process Execution Language (BPEL) and its extension BPEL4People [15] for the implementation of TripleS's cloud platform. Other infrastructure-as-a-service (IaaS) cloud computing platforms that are not AWS based could also be used to implement the TripleS cloud platform. It consists of four parts: management interface, process runtime environment, storage service, and deployment environment.

Management interface: The Management interface is implemented by integrating the Apache ODE management interface and the development environment provision interface. For instance, as shown in Figure 2, implementation of the development environment provision enables the ability to download sources and/or compile releases of the software packages that are required for setting up the development environment, and related documentation and examples.

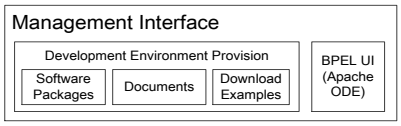


Figure 2. Components of the Management interface

Process run-time environment: To implement the process run-time environment, we adopt the apache tomcat server for the setting up of the BPEL running environment - Apache ODE. Based on this open source business process run-time environment, BPEL processes and their extension BPEL4People processes can be deployed, and the related crowdsensing platform can be set up on top of it. Moreover, as in the mobile platform of TripleS, its services are REST-based, while BPEL only supports Simple Object Access Protocol (SOAP) based web services. Thus, the BPEL4People processes are not supported by Apache ODE. To address this issue, we use a REST-SOAP Adapter. This adapter can receive the SOAP service invocation requests, and transform them into the REST service invocation requests.

Storage service: Based on the AWS S3, the storage service wraps the APIs for all of the data storage requirements from other modules: the sensing data both from crowdsensing participants and/or service providers of social networks through opportunistic networks and Internet, the related software packages, examples of documents for the development provisioning, and the mobile SOA environment provisioning.

Deployment environment: The deployment environment is composed of three modules. We integrate the Apache ODE deployment environment to form a base for the Management Interface to support BPEL and the BPEL4People development environment. Based on the storage service module, we implemented the mobile SOA environment provision module.

B. Implementation of the mobile platform of TripleS

In the current version of TripleS, the mobile platform of it is implemented based on the Android operating system. More technical details about the Aframe and mobile SOA framework, as well as a prototype and source codes can be found in the website of our project [16].

Aframe in Android: In our previous work, based on AmbientTalk [17], we have implemented a version of Aframe, which owner applications (developed using standard Java) could initiate mobile agents from a PC to travel around mobile devices and process tasks automatically and self-adaptively on them over a wireless ad-hoc network. Considering that in mobile crowdsensing application scenarios, mobile agents should normally be released by mobile devices, therefore we have developed a new version of Aframe based on the Android system, which works independently in mobile devices but could also take full advantage of AmbientTalk.

This new version of Aframe is realized by adding a new layer - mobile Aframe AmbientTalk library on top of the AmbientTalk mobile libraries and class libraries of Android. In this layer, we encapsulate the AmbientTalk virtual machine, and related basic networking APIs provided by it as a new library of Android. On top of this layer, we develop the Aframe Java class library, and design a mechanism which can invoke the method evalAndPrint(String script, PrintStream output) in the original class library of AmbientTalk to exactly map the class attributes between these two layers. Thus the agent-based applications of Aframe developed by Java on Android could automatically invoke the APIs in the mobile Aframe AmbientTalk library layer.

Based on the APIs provided by the mobile Aframe AmbientaTalk library that we implemented, we then developed the framework services on the Android operating system. Mobile agents can invoke these services through resident agents to get real-time network information when they are dynamically traveling around the local opportunistic network and collecting the sensing data. After the mobile agents have successfully collected the sensing data and returned to the original mobile devices, they will submit the data to the owner applications, which then automatically transfer the data to the context-awareness service in the mobile SOA framework, and/or end users through the user interface directly; the related strategies could be pre-defined by the application developers or end users.

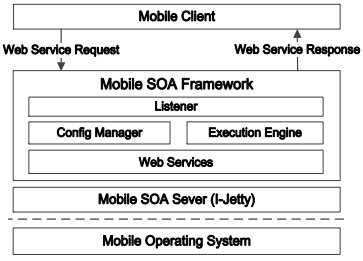


Figure 3. Architecture of mobile SOA framework

Mobile SOA framework: Similar to our work on MS2A [8], we adopt the open source web container I-Jetty [18] to implement the mobile SOA server, and based on the Android system and the specifications of RESTful Web Services, we develop the mobile SOA framework that integrates with the mobile SOA server, as shown in Fig. 6. But different from MS2A that targets disaster rescue and works independently over wireless ad-hoc networks, in TripleS, the mobile SOA framework functions as a bridge between the mobile device and

the crowdsensing platform in the cloud over the Internet. The communications between the cloud platform of TripleS and its mobile platform employ the standard web service format based on the HTTP protocol and XML data format. In addition, although BPEL interactions on the TripleS cloud side are SOAP based, and its services in the mobile platform are RESTful Web Services based, the SOAP-REST transformation can be achieved using additional adapters in between, similar to the method described above in the cloud side of TripleS.

5. EXPERIMENT

In this section, we evaluate the performances of TripleS in two aspects: performance of mobile agents executing tasks over opportunistic networks, and the system performance of TripleS's crowdsensing platform.

A. Performance of TripleS over Opportunistic Networks

The experiment consists of five persons each carrying an Android device (three Android phones and two Android tablet, all of which running Android version 4.0 or above) equipped with 802.11n WiFi module and running TripleS. We use one of the Android devices to act as a WiFi hotspot and let the others connect to it through WiFi. As shown in Figure 4, each person carries a mobile device denoted as node Mx and moves in an area of about 150×150 m2. There are nine predefined positions, and each person may initiate mobile agents to collect the sensing data from others when he/she moves from one position to another at a normal walking speed. There are two parts of this experiment as following.

Figure 4. The map area of the experiment

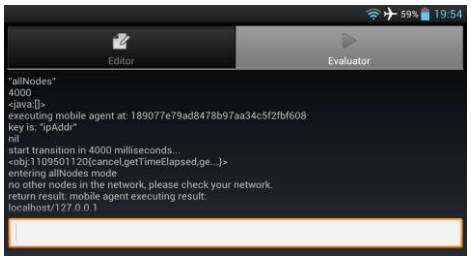


Figure 5. Status monitoring of mobile agent

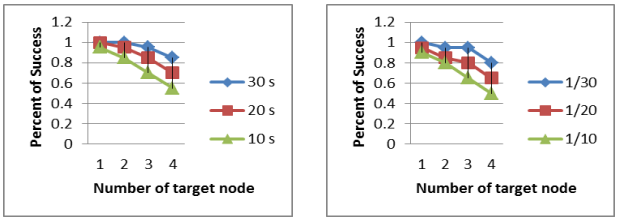
1) Tasks execution success rate of mobile agent

We evaluate the tasks execution success rate of mobile agents under simulated conditions of mobile nodes being disconnected in constant time intervals (T) of 30s, 20s, and 10s, or in exponentially distributed time with rate parameter λ= 1/30, 1/20, and 1/10, and each person initial a mobile agent to collect the sensing data four times. The average results are calculated. Also, as shown in Figure 5, we can monitor the status of mobile agents through the user interface of Aframe in Android. The results in Figure 6 show that the mobile agent of TripleS could automatically finish most of the sensing tasks and simplify the operation of mobile crowdsensing, except under conditions of T=10s orλ= 1/10, which are atypical in real environments.

2) Time efficiency of mobile agent finishing sensing tasks

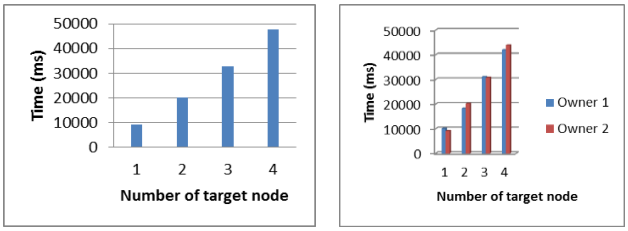
Similar to part 1, in this part, each person initiate one or two mobile agents to collect sensing data. The average time consumed under each condition is calculated. The mobile nodes do not go to offline mode in this set of experiments. The results in Figure 7(a) were obtained when only one person initiated a mobile agent every time, and while Figure 7(b) shows the results when two persons initiated two mobile agents simultaneously every time. Comparing the results of these two cases, we find that less time is consumed when two mobile nodes initiate mobile agents to execute the same sensing tasks, because the earlier agent may finish the sensing task and share the results with mobile agents arriving later, which saves the time to duplicate the sensing tasks.

Both set of experiments described in part 1 and part 2 above last about 30 mins. We recorded the battery consumption of the Android devices, and found that the battery consumption due to running TripleS is relatively low, e.g., only about 2% in a Google Nexus 10.



(a) (b)

Figure 6. Tasks execution success rate



(a) (b)

Figure 7. Time efficiency of finishing sensing task

B. System performance of TripleS's crowdsensing platform

We evaluate the system performances of TripleS's crowdsensing platform in terms of three parameters: time efficiency, energy consumption, and networking overhead on mobile devices, as these parameters are of particular concern to mobile users when they are participating in mobile crowdsensing. The communications between the cloud side of TripleS and its mobile platform used the standard web service format based on the HTTP protocol and XML data format, and the experimental environment is: Hardware: Amazon EC2 M1 Medium Instance; 3.75 GiB memory; 2 EC2 Compute Unit (1 virtual core with 2 EC2 Compute Unit); 410 GB instance storage; 32-bit or 64-bit platform; I/O Performance: Moderate; EBS-Optimized Available: No. Software: operating system: Ubuntu 12.04.1; Servers: ApacheTomcat 7.0.33; BPEL engine BPEL4People environment: ODE1.3.5.

Three Google Nexus 10 (Android 4.2.1 version, battery capacity 9000mAh) were used as the mobile devices in these experiments, from which we obtained three sets of data simultaneously. A total of 10 tests were run over 3 days, and the average results were calculated. Each experiment lasted 30 minutes. Each Nexus 10 sent crowdsensing requests to the servers on the cloud side of TripleS according to a Poisson distribution with an arrival rate of E=5/min, and the screen was shut off during the runtime. The time delay refers to the delay from the time that the Nexus 10 initiates a crowdsensing (with no data process) request to the cloud side and the time that it

receives the responses from the servers on the cloud side of TripleS.

The experimental results are presented in Table 1. We find that the results of the three sets of data are very similar, with all averaging about 12s. Moreover, we make a simple comparison with the related work Medusa [19], where the similar time delay is about 64s, although their runtime environments are different. The main reasons for this are that Medusa adopts the commercial Amazon AMT as the crowdsensing platform and Short Message Service (SMS) to deliver the message, which incur delays of about 31s and 27s, respectively, while TripleS uses open source to develop the customized crowdsensing platform and sends the service request through a standard web service message.

Table 1. System performance of TripleS

Parameters	Data set 1	Data set 2	Data set 3
Time delay (msec)	Max.: 21319, Min.: 10388	Max.: 20326, Min.: 10336	Max.: 22636, Min.: 2399
	Ave.: 12629	Ave.: 12803	Ave.: 11354
Battery consum.	53mAh/30mins	53mAh/30mins	54mAh/30mins
Network overhead	0.53MB/135 requests	0.51MB/132 requests	0.70MB/155 requests

6. RELATED WORK

There have been several research works about mobile crowdsensing. The different works are differentiated by: (i) Targeting some specific application scenarios [3-5]; (ii) Using specific techniques to construct standalone crowdsensing system to support multiple mobile crowdsensing applications, e.g., using Twitter [6], and using an in-node hardware abstraction layer and overlay management protocol to allow multiple applications sharing sensing data across different mobile nodes [20]; and (iii) Supporting efficient development of different mobile crowdsensing applications; e.g., [21] aims to enable developers to write server-side programs in lieu of distributed programs, so as to ease the development of crowdsensing applications on smartphones.

Partly inspired by but different from these works, TripleS aims to provide a general approach at the system and architecture design level by leveraging the advantages of a number of techniques, which salient parts are orchestrated into a flexible, efficient and economic platform across Internet and opportunistic networks for mobile crowdsensing applications. It not only supports developing multiple customized mobile crowdsensing applications with standard service interaction, but also provides a mobile distributed system with augmented cloud computing platform, that enabling users to easily and efficiently participate in and perform multiple and diverse crowdsensing tasks on their mobile devices.

7. CONCLUSION

In this paper, we have presented TripleS, a novel mobile system for crowdsensing, which leverages the advantages of numbers of open source techniques across mobile devices and cloud platforms, to provide a systematic approach that supports both application developers and users for mobile crowdsensing. Our practical experiments have demonstrated that TripleS performs its tasks with a considerable time efficiency, low battery consumption and low communication overhead on mobile devices. To the best of our knowledge, TripleS is the first mobile system architecture that supports efficiently developing, deploying and managing multiple mobile crowdsensing applications/tasks in a comprehensive, flexible and open platform. Also, TripleS is the first mobile distributed system that supports agent based sensing data fusion, real-time service collaborations between sensing services and social networking services across opportunistic networks and Internet, which enables mobile users to participate in and process crowdsensing tasks in an efficient and ubiquitous manner.

8. REFERENCES

[1] X. Hu, T.H.S. Chu, H.C.B. Chan, and V.C.M. Leung. Vita: A Crowdsensing-Oriented Mobile Cyber-Physical System. *IEEE Trans. Emerging Topics in Computing, 1(1)*, 2013.

[2] J. Howe, "The Rise of Crowdsourcing," Wired, Jun. 2006.

[3] H. Lu et al. Campbell. SoundSense: scalable sound sensing for people-centric applications on mobile phones. *In Proc. MobiSys*, pp. 165-178, 2009.

[4] R. K. Balan, K. X. Nguyen, and L. Jiang. Real-time trip information service for a large taxi fleet. *In Proc. MobiSys*, pp. 99-112, 2011.

[5] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: predicting bus arrival time with mobile phone based participatory sensing," *In Proc. MobiSys*, pp. 379-392, 2012.

[6] M. Demirbas, M. Bayir, C. Akcora, and Y. Yilmaz. Crowd-sourced sensing and collaboration using twitter. *In Proc. WoWMoM*, pp. 1-9, 2010.

[7] X. Hu, W. Du, and B. Spencer. A Multi-Agent Framework for Ambient Systems Development. *Procedia Computer Science*, vol. 5, pp. 82-89, 2011.

[8] X. Hu, V. C. M. Leung, W. Du, B. C. Seet, and P. Nasiopoulos. A Service-oriented Mobile Social Networking Platform for Disaster Situations. *In Proc. HICSS*, 2013.

[9] X. Hu, V.C.M. Leung and W. Wang. VSSA: A Service-oriented Vehicular Social-Networking Platform for Transportation Efficiency. *In Proc. ACM DIVANet*, 2012.

[10] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. *In Proc. SenSys*, pp. 323-336, 2008.

[11] S. Mathur et al. Parknet: Drive-by sensing of road-side parking statistics. *In Proc. MobiSys*, pp. 123-136, 2010.

[12] GeoChat by InSTEDD. http://instedd.org/technologies/geochat/.

[13] The Ushahidi Platform. http://ushahidi.com/products/ushahidi-platform.

[14] C. Pautasso, O. Zimmermann, and F. Leymann. RESTful web services vs. "big" web services: making the right architectural decision. *In Proc. WWW*, pp. 805-814, 2008.

[15] OASIS, Web Services Business Process Execution Language Version 2.0, April 2007. http://docs.oasis-open.org/wsbpel/ 2.0/wsbpel-v2.0.pdf

[16] Mobile SOA and Aframe, 2013. http://mobilesoa.appspot.com/

[17] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt and W. De Meuter, "Ambient-oriented Programming in AmbientTalk," *In Proc. ECOOP*, pp. 230-254, 2006.

[18] I-Jetty, 2012. http://code.google.com/p/i-jetty/

[19] M. Ra, B. Liu, T. La Porta, and R. Govindan. Medusa: A Programming Framework for Crowd-Sensing Applications. *In Proc. MobiSys*, pp. 337-350, 2012.

[20] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft. SenShare: Transforming Sensor Networks into Multi-application Sensing Infrastructures. *In Proc. EWSN*, 2012.

[21] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden. Code in the air: simplifying sensing on smartphones. *In Proc. HotMobile*, 2012.