Spheres: A Web Services Framework for Smartphone Sensing as a Service

Mark Allen Gray
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County (UMBC)
mgray2@umbc.edu

***Abstract* — Smartphones are mobile phones that execute application programs. Smartphones are practically everywhere, in every country and culture. Smartphones can tell where they are by GPS, a spatial sense, and what time it is, a temporal sense, relative to their GPS location. Groups of smartphone users form communities, creating a social sense. Smartphones often contain built-in sensors including temperature, proximity, accelerometer, gyroscope, magnetometer, barometric pressure, ambient light, camera, microphone, etc., as well as the ability to connect to external sensors over their wireless interfaces forming wireless sensor networks. Smartphones are connected to the Internet through their cellular data connections providing a globally established deployment of "things" on the Internet of Things (IoT). People carrying their smartphones represent a global source of continuous mobile Internet-connected sensor data. The resulting human, social, and environmental data represents an untapped global information resource of epic proportion. This paper explores the current research related to smartphone sensing as a service (S^{2}aaS) and proposes an architecture for a web services framework for S^{2}aaS called Spheres. Spheres will integrate wireless sensor network, mobile Internet, and cloud technologies into an open-source service-oriented architecture (SOA) providing crowdsourced smartphone sensor data aggregation and sharing in secure cloud deployments for private and public consumption. Spheres collected data can be overlaid visually on maps or incorporated into applications that require the data from sensors embedded in, or connected to, smartphones over specific regions and times of interest. In addition to specifying an architecture, this paper will identify the challenges expected to implement and demonstrate a prototype of Spheres on at least one smartphone platform with the cloud platform and web services deployed on the UMBC BlueWave high performance computing system.**

Keywords* — Smartphone Sensing as a Service; S^{2}aaS; Sensing as a Service; Mobile Sensing; Web Services; Service Oriented Architecture; SOA; Internet of Things; IoT.

I. INTRODUCTION

Just as the Big Bang evolved into the physical world in which we live and breathe, the Internet has evolved into the information world that connects our digital lives and the things that we create and use. We call this new information world of connected things the Internet of Things (IoT) and like our early physical universe, it is expanding and changing rapidly. Firmly embedded within this new world of IoT there are smartphones, and smartphones are practically everywhere, in every country and culture with an estimated 1.75 billion users, 24.4% of the world population, in 2014. Growth is expected to rise to 2.5 billion users, 33.8% of the world population, by 2017 [1].

What is a smartphone? A smartphone is a mobile phone that provides mobile phone call functions as well as other services through application software that executes on the phone device. Smartphones can tell where they are by GPS, a spatial sense, and what time it is, a temporal sense, relative to their GPS location. Groups of smartphone users form communities, creating a social sense. Smartphones often contain built-in sensors including temperature, proximity, accelerometer, gyroscope, magnetometer, barometric pressure, ambient light, camera, microphone, etc., as well as the ability to connect to external sensors over their wireless interfaces forming wireless sensor networks. Smartphones are connected to the Internet through their cellular data connections. People carrying their smartphones represent several billion global continuous mobile Internet-connected sensor data sources. The resulting human, social, and environmental data represents an untapped global information resource of epic proportion.

This paper explores the current research related to smartphone sensing as a service (S^{2}aaS) and proposes an architecture for a web services framework for S^{2}aaS called Spheres. Spheres will integrate wireless sensor network, mobile Internet, and cloud technologies into an open-source service-oriented architecture (SOA) providing crowdsourced smartphone sensor data aggregation and sharing in secure cloud deployments. Spheres collected data can be overlaid visually on maps or incorporated into applications that require the data from sensors embedded in, or connected to, smartphones over specific regions and times of interest. In addition to specifying an architecture, this paper will identify the challenges expected to implement and demonstrate a prototype of Spheres on at least one smartphone platform with the cloud platform and web services deployed on the UMBC BlueWave high performance computing system [2].

Spheres will consist of: (1) a cloud platform with a cloud management dashboard for deploying autoscalable virtual machine images comprising the web services software stack, (2) a preconfigured virtual machine image containing the Spheres web portal for admin and user account management and a Spheres web services application programming interface (API) for the Spheres mobile apps and application developers to access the core web services, (3) a set of mobile apps that use the API for supervisors to create and manage projects and participants to join projects and share their data, and (4) interfaces to extend smartphone APIs with services to support sensors external to smartphones. The core web services will comprise a set of project management services for project creation and participation and a set of sensor data services for data streaming, storage, retrieval, mapping, and visualization. Application developers will have access to real-time and recorded smartphone sensor data streams through the web services API.

My expectation is to demonstrate a prototype of Spheres on at least one smartphone platform with the cloud platform and web services deployed on the UMBC BlueWave high performance computing system. My research so far has identified some challenges that will be addressed as research and development continue. These challenges include: (1) the diversity of smartphone platforms, both operating systems and smartphone products that use them, (2) the diversity of smartphone sensors, both internal and external, and access to the sensor data from public APIs, (3) sensor data security and privacy, (4) smartphone sensing participation, and (5) system complexity and integration. Through the use of the BlueWave system at UMBC, the various cloud and web service components will be incrementally integrated and demonstrated. Results from these incremental efforts will be documented and shared through peer reviewed published research papers and technical conference presentations.

In this paper, I first introduce the concept of S^2aaS and describe the primary components of a web services framework for S^2aaS called Spheres. Section 2 describes my previous related work on a web services framework for wireless sensor networks and Section 3 describes the Spheres ecosystem in terms of the types of users and their roles and the devices and applications that they will use. Section 4 points out the market opportunity for Spheres based on current technology trends and proposes a business case for monetizing the services. In section 5, current work on S^2aaS is explored from several different use case perspectives and different smartphone sensing paradigms and incentive mechanisms for participation are identified. Section 6 describes the Spheres architecture as a set of software layers each comprising a system software stack identifying available open source software components as well as the open source components to be developed and integrated. Section 7 describes the challenges expected to develop and demonstrate a Spheres prototype. Section 8 provides a conclusion and section 9 provides acknowledgment to individuals that supported this research. The final section 10 provides a list of the references indexed throughout.

II. Previous Work

This research builds on my previous work on a web services framework for wireless sensor networks [3]. The focus on that work was on fixed-location wireless sensor node deployments where a group of wireless sensor nodes would connect to a gateway device running the framework's sensor server where the sensor nodes and the gateway server formed a wireless sensor network (WSN). Users wanting to contribute their WSN data to a cloud deployment for storage, sharing, and computation on cloud resources would create an account through the service's web portal and install and configure the framework's sensor server software on their WSN's gateway device. The core web services in the cloud deployment are accessed through a REpresentational State Transfer (REST) [4] API. The system is architected entirely of open source software. An organization wishing to deploy the system for private or public use would configure the various service components on a private or public cloud. The work resulted in a preliminary specification for the API. We successfully demonstrated an architecture and initial implementation. A test case WSN using temperature, barometric pressure, and humidity sensor data was simulated by pulling actual real-time weather data from a published weather reporting API and feeding the data into the framework. Each framework component was individually constructed, tested, and demonstrated. The cloud storage and compute resources were provisioned from the UMBC BlueGrit high performance computing system [5].

With Spheres, instead of supporting users of fixed-location wireless sensor networks as in my previous research, the focus is users of mobile smartphones. In fact a smartphone with externally attached sensors comprises a mobile wireless sensor network. Architectural features and elements that are carried over into Spheres include: (1) open source system, (2) REST API of web services on a cloud deployed server, (3) sensor data aggregation on cloud storage, and (4) a web portal user interface to add users and admin the system. New architectural features and elements include: (1) mobile apps for project creation, participation, sensor data collection, and data streaming to the cloud services, (2) integration of the OpenStack [6] cloud computing infrastructure, (3) integration of Google Maps [7] for mobile sensor data mapping and visualization, and (4) integration of the Hierarchical Data Format (HDF) [8] for sensor data storage and streaming.

III. Spheres Ecosystem

Figure 1 is a high-level illustration of the overall Spheres ecosystem. User communities with differing interests and organizational structures may create a deployment of the web services on a private cloud within their organization or on a public facing cloud for public engagement of users outside of the organization. The cloud deployment provides all of the advantages of cloud computing including virtualization of resources and autoscaling of those resources based on demand. Within the cloud deployment are the web services where the sensor data is aggregated and shared with the user community.

Within a user community there are four user roles: admin, supervisor, participant, and developer. An admin is a member of the organization that deployed Spheres. An admin has access permissions to the entire Spheres software deployment, maintains the software, and approves the registration of all users according to the policies of the organization. A supervisor creates projects, adds participants to projects, schedules projects, analyzes results, and directs participants. A supervisor must register for an account which is approved by an admin. A participant participates in projects created by and managed by a supervisor. The participant's sensor data is made available to the project for analysis. Each participant has access only to their sensor data. Participants use a Spheres mobile app on their smartphones. A participant must register for an account which is approved by an admin. A developer uses an API to the web services to build applications that make use of Sphere's data. Each developer is assigned a unique API key that they use to access the web services from their programs. A developer must register for an account which is approved by an admin.

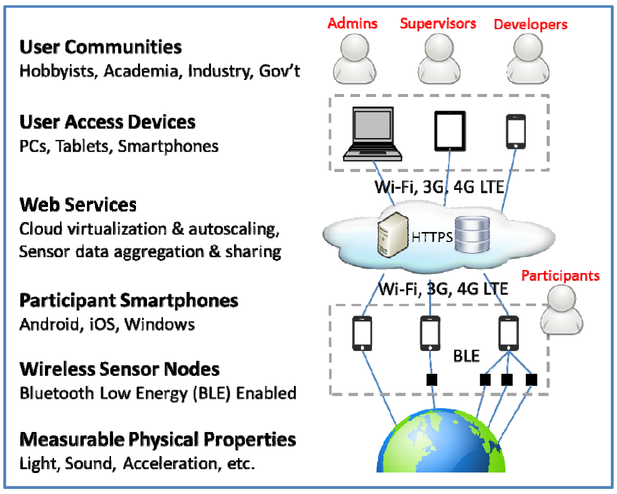
Admins
Supervisors
Developers
User Communities
Hobbyists, Academia, Industry, Gov't
User Access Devices
PCs, Tablets, Smartphones
Wi-Fi, 3G, 4G LTE
Web Services
Cloud virtualization & autoscaling,
Sensor data aggregation & sharing
HTTPS
Participants
Wi-Fi, 3G, 4G LTE
Participant Smartphones
Android, iOS, Windows
BLE
Wireless Sensor Nodes
Bluetooth Low Energy (BLE) Enabled
Measurable Physical Properties
Light, Sound, Acceleration, etc.

Figure 1. Spheres Ecosystem

User access to the cloud hosted web services is through any Internet connected device. For mobile devices, Wi-Fi and 4G LTE, as well as 3G and other international cellular data connections can provide the required Internet connectivity. Spheres will operate on, not within, an Internet Protocol (IP) network using Hypertext Transfer Protocol Secure (HTTPS). For participants, a smartphone running a Sphere's participant mobile app is required. Although the design goal of Spheres is to support Google Android, Apple iOS, and Windows Phone smartphones, the research prototype will focus on Android which currently holds the largest share of the smartphone market [9]. Android is also open source, supporting the open source spirit of Spheres.

Participants will use a mobile app to participate in sensing projects defined by supervisors where sensing criteria are defined based on regions of interest and times of interest. Sensing is either opportunistic where sensing and streaming to the web service occur automatically when the criteria are met, or participatory where sensing and streaming are manually performed by the participant when the criteria are met. Participants will always have full control over the data they elect to sense and contribute to a project, including data from the sensors internal to their smartphones as well as wireless sensor nodes external to their smartphones that they may connect over a wireless interface such as Bluetooth Low Energy (BLE) which is becoming a popular interface for connecting sensor nodes to smartphones. Regardless of whether the sensor is internal to the smartphone or external, the sensed data must be available through a public API within the smartphone's software development environment in order to use that data in sensing projects.

IV. SPHERES OPPORTUNITY

In a 2013 report [10] on disruptive technologies published by The McKinsey Global Institute, an economic research group that studies global economic trends for their big business clients, the report identified IoT as one of the top 12 disruptive technologies that will have the greatest global economic impact by 2025. The report describes IoT as "… embedding sensors and actuators in machines and other physical objects to bring them into the connected world—is spreading rapidly". The McKinsey Global Institute report identified two other disruptive technologies in their list of the top 12 that are relevant to this research: mobile Internet and cloud technology. As depicted in Figure 2, the intersection of the Internet of Things, mobile Internet, and cloud technology represents a unique opportunity which is the focus of Spheres.

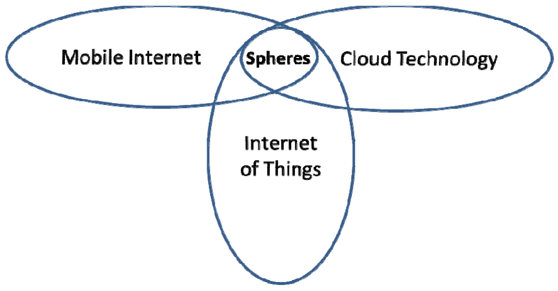

Figure 2. Spheres Opportunity

Figure 3 proposes a business case for a Spheres deployment. In this case, the Spheres service provider acts as a broker for the Spheres service consumers which comprise the participants that are sellers of their sensor data and the developers which are the buyers of the sensor data for use in their application programs.

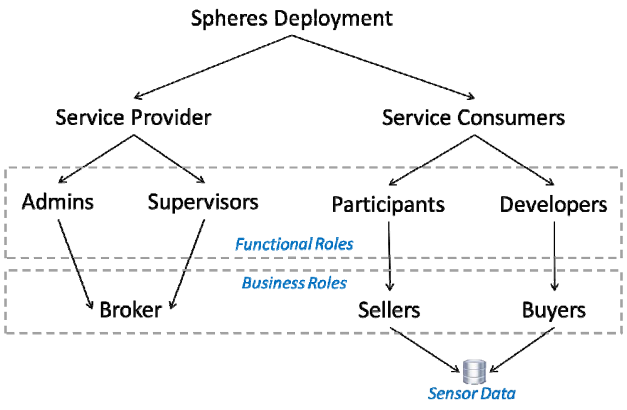

Figure 3. Spheres Business Proposition

V. SMARTPHONE SENSING AS A SERVICE

This section examines current research in sensing as a service from different perspectives including government, academic, and commercial perspectives. In addition, the different smartphone platforms and their characteristics are explored as well as the sensors commonly found within or attached to smartphones. Concepts on smartphone sensing paradigms are described and participation incentive mechanisms are explored.

A. *Sensing as a Service*

With the advent of cloud computing and the plethora of anything "as a service" services that build on clouds, we are seeing the emergence of "sensing as a service". In this section I explore this new idea from several perspectives including: a standards perspective, an academic research perspective, a business development perspective, a chip design company perspective, and a Department of Defense (DoD) agency perspective.

1) A Standards Perspective

The United States Department of Commerce National Institute of Standards and Technology (NIST) published *The NIST Definition of Cloud Computing* in 2011 [11] which defines these five essential characteristics of cloud computing: On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, and Measured service. It then defines these three cloud service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Sensing as a Service, and more specifically Smartphone Sensing as a Service (S^2aaS), is a cloud service model that is receiving recognition in research circles. It provides the five essential characteristics of cloud computing for the sensors embedded in, and attached to, smartphones.

2) An Academic Research Perspective

The first published reference that I found on Sensing as a Service was a 2012 IEEE published research paper by Sheng, Xiao, Tang, and Xue [12] titled "Sensing as a Service: A Cloud Computing System for Mobile Phone Sensing". They introduce the concept of providing sensing services using smartphones via a cloud computing system and identify the following functional requirements: a web interface, generating sensing tasks, scheduling sensing activities, recruiting mobile (smartphone) users, managing sensors, and storing sensor data. My research is in agreement with these fundamental functional requirements for a S^2aaS system; however, there are others to consider including security and privacy, cloud virtualization and scalability, and sensor data mapping and visualization.

3) A Business Development Perspective

In an April 2014 presentation titled "Sensing-as-a-Service: The New Internet of Things (IoT) Business Model" by Dr. Mazlan Abbas from MIMOS Berhad, Malaysia's international center for research and development, Dr. Abbas presents a strong business case for leveraging smartphones as sensing devices for "mobile crowdsensing" within the context of a larger "IoT Eco-System" [13]. In this new "Sensing-as-a-Service" business model that Dr. Abbas presents, there are four primary functional blocks: Sensor block comprising sensor and sensor owners, Communications block comprising sensor gateways for Internet connectivity, Enterprise block comprising cloud-based web services, and Customers comprising all of the users of the sensor data across a wide range of industries. The Spheres ecosystem embodies a similar functional partitioning but with a singular focus on smarthphones with well-defined user roles and cloud-based web services to support them.

4) A Chip Design Company Perspective

In a January 2014 white paper titled "Sensors as a Service on the Internet of Things", by Willard Tu at ARM Limited, Tu builds a strong technical case for the convergence of embedded sensor devices with IoT and the power of this convergence to drive future innovation and applications [14]. Tu refers to the integration of numerous MEMS sensors in smartphones and the continuing trend to pack more into less space including multiple sensor types into a single package. This integration of multiple sensor types localized with powerful mobile processors allows smartphones to have a sense of context awareness through software that fuses various sensor data sources to create knowledge and awareness from the disparate data. Most sensor vendors that are members of the MEMS Industry Group provide libraries that translate readings from multiple sensors into context. For example, STMicrolectronics offers the iNEMO Engine, a sensor fusion library implementing Kalman filtering for adaptive prediction.

5) A DoD Agency Perspective

In a May 2014 report titled "Sensing as a Service, An ISR Horizons Future Vision" published by the United States Air Force, the report highlights the need for a paradigm shift to "transform the focus from sensors to sensing, from data to decisions, from stove-pipe to network" [15]. Sensors in this context are not smartphone sensors or wireless sensor nodes; they are large complex systems, for example, radar sensor systems. However, the idea of sensing as a service and the paradigm shift is the same for smartphone sensing. Their vision is to increase the integration of sensors with warfighters and decision systems across all agency operations. Likewise, the vision of Spheres is to increase the integration of sensors with mobile participants and project supervisors across a sensing community of users supporting their operational needs.

B. *Smartphone Platforms*

A smartphone platform is distinguished by the following primary characteristics: operating system, product vendor, and product model.

1) Operating System

As identified previously, a smartphone is a mobile phone that executes application programs. A smartphone contains a processor for executing these programs as well as memory, file storage, user interface devices, sensors, batteries, and external communication interfaces, all of which must be managed by an operating system. A smartphone platform comprises all of these things, with the most defining characteristic between smartphones being the operating system (OS). According to International Data Corporation (IDC) market research [9], as of the third calendar quarter of 2014, the most popular smartphone OS by a wide margin is Google's Android at 84.4% followed by Apple's iOS at 11.7%, with Windows Phone coming in third at 2.9%. The Blackberry OS market share has fallen from 9.6% in 2011 to only 0.5% and all other smartphone OSs comprise only 0.6% of the market share.

2) Product Vendor

The second distinguishing characteristic of a smartphone platform is the vendor that manufactured the smartphone. According to IDC market research [16], as of the third calendar quarter of 2014, the most popular smartphone vendor is Samsung at 23.4% followed by Apple at 11.7%. Xiaomi, Lenovo, and LG are all practically tied at 5.1% ±0.1%. Almost one half, 49.3%, of the market share is spread across more than 175 other smartphone vendors.

3) Product Model

The third distinguishing characteristic of a smartphone platform is the model. Newer model smartphones contain increasingly more sensor integration than older model smartphones. Also, there is a strong correlation between product model numbers and the smartphone OS versions that support them; it is important to fully understand these relationships in order to maximize smartphone and sensor support within the Spheres framework.

C. Smartphone Sensors

A review of current smartphone platforms through Internet searches of the smartphone vendor products shows that many of the new smartphones contain some combination of the sensors described in Table 1.

Table 1: Internal Smartphone Sensors

Sensor	**Data**	**Use at the phone**
Proximity	Distance up to 5cm or simply "near" and "far"	Automatically turn off the touchscreen when the phone is held to ear to prevent unintentional touch input.
Accelerometer	Acceleration along three axes	Automatically change the screen between portrait and landscape as the phone is moved. Shake to undo. Activity tracking. Gaming.
Gyroscope	Rate of rotation around three axes	Image stabilization. Gaming.
Light	Ambient light intensity	Automatically adjust the brightness of the screen to conserve power.
Water	Visual indicator	To determine if the phone may have been damaged due to exposure to water
Thermometer	Ambient temperature	Internal temperature monitoring for over heating; some research has been done to attempt to measure external temperature
Barometer	Ambient atmospheric pressure	GPS location calculation, Indoor elevation localization
Hygrometer	Humidity	Some recent phones contain this sensor; need more research
Magnetometer	Magnetic field	Compass app. Metal detector apps.
Microphone	Sound	Voice calls, recordings, commands. Language translation apps.
Back camera	Images	Pictures, video recording, heart rate apps
Front camera	Images	Pictures, video recording, video conferencing

An in-depth analysis is required to determine which platforms, defined by the smartphone OS, product vendor, and product models, are required to fully understand the specific sensor data types, sizes, formats, units, and data rates that Spheres will be required to operate with. This information will be important for estimating transfer bandwidth and data storage requirements. Some of the questions to be answered with further research include:

1. Which sensors are supported by which smartphone operating systems?
2. Which versions of the smartphone OSs support them?
3. Does the OS software development kit (SDK) provide a public API?
4. For the public API for each sensor, what are the programming considerations, for example: sample rate and how that may affect power consumption? Security and privacy considerations? Other considerations?
5. What are the specific data types, sizes, formats, units, and data rates?

In addition to the sensors built into smartphones, many sensor vendors are building mobile wireless sensor nodes that can be carried on the body and connect to the smartphone either directly through a wired connection from the node to the phone's audio jack or USB port or wirelessly through Bluetooth Low Energy or Wi-Fi Direct. These vendors often provide mobile apps that communicate with the smartphone and present the data to the user through the app and often they provide APIs for developers to build their own apps which is a requirement for Spheres to be able to access the data. Research on these external sensors is required to answer some of the questions posed above with respect to the internal sensors as well as these additional questions: What vendors make sensor products that connect to smartphones? How do they connect? Audio port, USB port, BLE, Wi-Fi Direct, other? Do they have an API?

D. Smartphone Sensing Paradigms

In the work performed by Sheng, et. al [12] and by Abbas [13], they identify two smartphone sensing paradigms: opportunistic sensing and participatory sensing. Opportunistic sensing occurs automatically without the participant's intervention by software running on the smartphone (a mobile app using the sensor APIs) when the sensing criteria are met for a user participating in a sensing project. Participant agreement to opportunistic sensing is the first criteria to be met. For example if the participant that has agreed to opportunistic sensing enters a region of interest (spatial criteria) during a time of interest (temporal criteria) then sensing is enabled and proceeds. Sensing disables when the sensing criteria are no longer met, for example the participant no longer agrees or leaves the region or time of interest. Participatory sensing requires manual interaction of the participant when they enter the region and time of interest through alerts by the software or unscheduled due to unscheduled events, for example providing data in the event of an accident or natural disaster. Agreement by the user is inherent by their action to actively participate. Participatory sensing occurs by the active manual engagement of the participant in the sensing activity. For example, a participant takes pictures at the scene of a traffic accident and shares them through the service.

E. Participation Incentive Mechanisms

One of the challenges for S^2aaS is the acquisition of users to participate in sensing projects. It is a crowdsourcing problem that is better stated as "crowdsensing". By definition of the problem, the human carrying their smartphone as they go about living their lives is required to collect the data of interest which is data that has human social and human environment centric value to the projects created by an organization that has deployed Spheres. Since S^2aaS is a relatively new research area, there is not a lot of research out there on mechanisms to incentivize people to participate in smartphone sensing projects.

Receiving payment for participation is an obvious incentive which is an approach taken by Amazon with their Amazon Mechanical Turk [17]. In the Amazon Mechanical Turk approach, there are two primary roles: requesters that request work and set the price and workers that perform the work and get paid. In the Spheres ecosystem, the requesters are the supervisors that create and manage the sensing projects and the workers are the participants that join projects and collect and contribute their sensor data. By monetizing the participation in a Spheres deployment, the organization comprising the supervisors will pay the participants for the sensor data that the participants contribute.

There are, of course other incentives for participants such as voluntary work out of a sense of social responsibility to contribute to a project that they have a personal interest in. Also, if the participants are employees of the organization, they may be required under employment policy to participate, where payment is realized through their paycheck, which is different from the Amazon Mechanical Turk approach. Table 2 summarizes these different participation incentives.

Table 2: Participation Incentive Mechanisms

Incentive Mechanism	**Description**
Turk Participation	A Spheres supervisor defines the request through the sensing projects that they create and manage and set pricing. A Spheres participant chooses to participate in projects of their choice and are paid for the sensor data that they collect for the projects in which they participate.
Voluntary Participation	Depending on the Spheres organization, some or all of the sensing projects may be on a voluntary unpaid basis. Participants choose to participate in projects based on their personal desire to contribute to projects that they have personal interest in out of a sense of social or environmental awareness and wanting to volunteer their time and battery power.
Mandatory Participation	Participants are employees of the Spheres organization and are required by employment policy to participate in sensing projects. Participants receive payment for their work through their paychecks

In the research performed by Sheng et al. [12], they explored three studies that ultimately failed to deliver successful incentive mechanisms. In a fourth study by Yang et al. [18] they proposed two incentive mechanism models: platform-centric model and user-centric model. Both models are based on the turk participation mechanism where participants chose to participate or not and are paid for the sensor data that they contribute to a project, however they differ by control of the price setting. In the platform-centric model, the platform provider sets the price that it will pay for sensing services. In the user-centric model, the users (the participants) ask for a price for their service by setting a reserve price. The research concludes that the design of incentive mechanism models is an important area for future research.

VI. Spheres Software Architecture

This section describes the Spheres architecture as a set of software layers each comprising a software system stack identifying available open source software components as well as the open source components to be developed and integrated. At the center of this architecture is the Spheres object abstraction and encapsulation in Hierarchical Data Format (HDF). Each sphere in a Spheres system maps to one participant smartphone and one HDF file in cloud storage. The software layers comprising a Spheres system include an operational layer for admins and supervisors, a server layer for cloud resource management and web services, a participant layer for executing projects and collecting data, and a sensor layer that integrates sensors external to a participant's smartphone over a wireless interface. This partitioning of the software into well-defined layers, stacks, components, and interfaces will allow for incremental builds and the evaluation and demonstration of each build.

A. Spheres Object Abstraction

A sphere encapsulates all data associated with one unique smartphone including:

- Physical properties of each sensor (unit of measure, min/max values, current value)
- Spatial sense (GPS location and altitude)
- Temporal sense (UTC time)
- Social sense (proximity to other spheres)
- Provenance (identification, ownership, history)
- Security controls (privacy settings, security groups, encryption keys)

A sphere is the fundamental unit of object abstraction within the Spheres framework. Think of a sphere as a 3-dimensional (3-D) spherical object sensing a region of interest specified by its current location, time, and sensing range capability, the size of which is a function of the current value of a "focus" sensor within its min/max range. Visualizing a sphere as a spherical object in 3-D space and time with a size that is a function the current value of a focus sensor can be applied to the visualization of Spheres data on mapping images for human consumption of the data.

The Hierarchical Data Format (HDF) version 5 [8] is a good match for implementing a sphere. HDF version 5 is called "HDF5". It has these features:

- Designed to store and organize large amounts of numerical data
- Datasets are multidimensional arrays of a homogeneous type
- Groups are container structures which can hold datasets and other groups
- Resources are accessed using the POSIX-like syntax /path/to/resource
- Metadata is stored in user-defined, named attributes attached to groups and datasets
- HDF5 works well for time series data that can be accessed much more quickly than the rows of a structured query language (SQL) database

The Spheres server stack will manage a database that will be used to manage Spheres users and sensing projects. Projects in the database will be defined as collections of spheres, where:

- Each sphere will map to a participant's unique smartphone with one HDF5 file per sphere,
- Each sensor project participation session will map to one HDF5 file group within the HDF file,
- Each smartphone sensor will map to one HDF5 file dataset within a group,
- HDF5 metadata, attached to groups and datasets, will track the sphere-related metadata.

B. *Spheres Software Layers*

The Spheres ecosystem described previously will be partitioned into four layers of software with the interfaces shown in Figure 4.

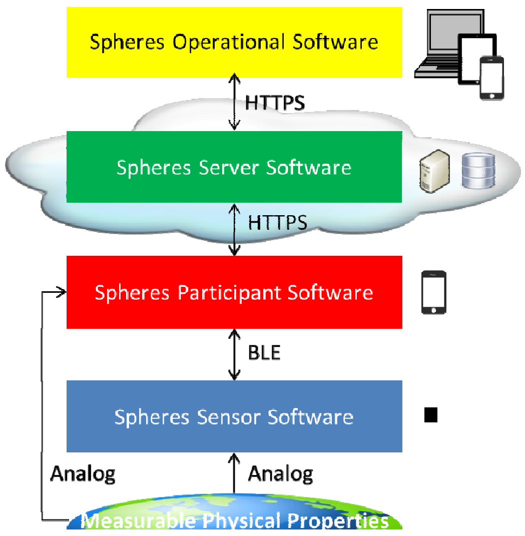
Spheres Operational Software
HTTPS
Spheres Server Software
HTTPS
Spheres Participant Software
BLE
Spheres Sensor Software
Analog
Analog
Measurable Physical Properties

Figure 4. Spheres Software Layers

The operational software layer comprises the software stacks running on the user access devices providing the user interfaces for cloud resource management, user account management, supervisor project management, project data results visualization, and the application programming interface libraries for developers of applications. The software stacks, components, and interfaces for this layer are described in detail in Section VI-C below.

The server software layer comprises the software stacks running on a cloud computing platform providing a dashboard and services for cloud resource management, a web portal and services for user account management, and a machine-to-machine REST web services API for sensor data storage and sharing services. The software stacks, components, and interfaces for this layer are described in detail in Section VI-D below.

The participant software layer comprises the software stack running on a smartphone owned by a participant participating in a sensing project. A user interface allows a participant to join a project and configure their settings. Sensor data matching the project criteria are transmitted to the server software for storage and sharing. The software stacks, components, and interfaces for this layer are described in detail in Section VI-E below.

The sensor software layer comprises the software stack running on a wireless sensor node that connects to a smartphone over BLE and transmits its sensor data to the smartphone for processing by the participant software layer. The software stacks, components, and interfaces for this layer are described in detail in Section VI-F below.

C. *Spheres Operational Software*

As illustrated in Figure 5, the operational software layer is composed of a set of software stacks and each stack contains a set of software components. Some components will be developed as part of this research and some are open source off-the-shelf components that will be integrated. It also illustrates that developers can extend the operational software layer by building their own application programs. This layer of the Spheres system software is primarily composed of the system's user interfaces and the stacks they operate on for client devices that use the server software layer. The interface from the operational software layer to the server software layer is over HTTPS, either through the client-side web browser to the server-side web portals or through an API library on Google Android that encapsulates the required REST request and response messaging interface to the server-side web services.

In this operational software layer, admins will use the Spheres OpenStack Dashboard to manage cloud resources and the Spheres Web Portal's admin pages to manage user accounts and portal content. All users will use the Spheres Web Portal's user pages to manage their accounts including their passwords and API keys for accessing the web services through the mobile apps or directly from programs they may create if they are application developers. The dashboard and the web portal are platform independent and will operate on popular web browsers including Chrome, Firefox, Internet Explorer, and Safari.

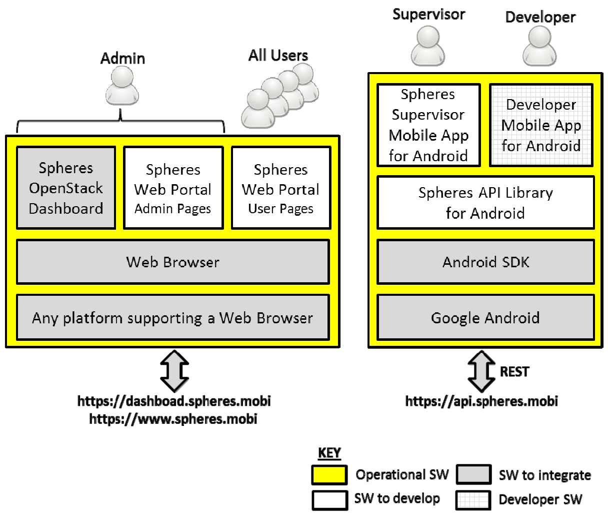
Admin
All Users
Supervisor
Developer
Spheres OpenStack Dashboard
Spheres Web Portal Admin Pages
Spheres Web Portal User Pages
Web Browser
Any platform supporting a Web Browser
Spheres Supervisor Mobile App for Android
Developer Mobile App for Android
Spheres API Library for Android
Android SDK
Google Android
REST
https://dashboad.spheres.mobi
https://www.spheres.mobi
https://api.spheres.mobi
KEY
Operational SW
SW to integrate
SW to develop
Developer SW

Figure 5. Spheres Operational Software

Users with the supervisor role will download the Spheres Supervisor Mobile App for Android from Google Play (as established previously, the Spheres prototype will be developed first for Android smartphones). This app will allow supervisors to create projects, define participation criteria, add participants to projects, schedule projects, direct participants, and view the results on time plots and maps. The app will use the Spheres API Library for

Android. This API to the Spheres Web Services within the server software layer is a REST API that uses HTTPS. All HTTPS request/response transactions are encapsulated within this library for ease of use from Android applications. The API library will be integrated with the Android SDK. Likewise, application developers subscribing to Spheres projects may use the Spheres API Library for Android to develop mobile apps that use that project's sensor data. Application developers may also use the REST API directly to create applications from any platform supporting HTTPS.

D. Spheres Server Software

As illustrated in Figure 6, the server software layer is composed of a set of software stacks and each stack contains a set of software components. Some components will be developed as part of this research and some are open source off-the-shelf components that will be integrated. This layer of the Spheres system software is primarily composed of the system's web services and the stacks and cloud instances that they operate on. The interface from the operational software layer to this server software layer is over HTTPS, either through client-side web browsers to the web portals in this layer or through a REST request and response messaging interface over HTTPS to the Spheres Web Services. The server stacks in the diagram are virtual machine (VM) images running on OpenStack on the UMBC BlueWave platform. There is a single OpenStack Admin Instance for managing the OpenStack deployment and for managing the Spheres Server Instances which can be autoscaled and load balanced. The Spheres Server Instance may use other public web services, in this case, the free tier usage level of the Google Maps Web Services will be used for the prototype demonstration of sensor data mapping.

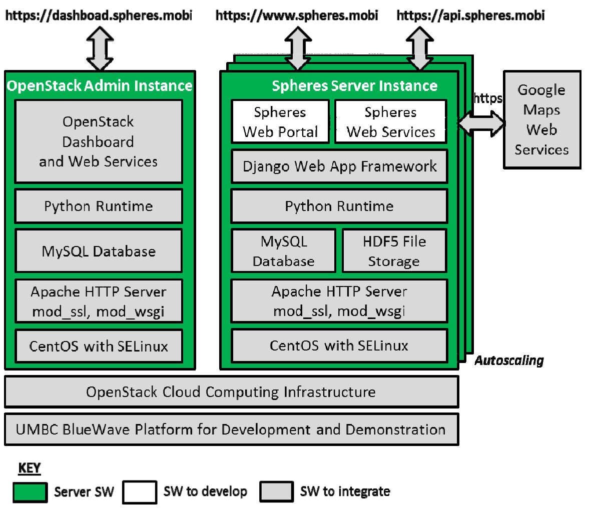

Figure 6. Spheres Server Software

The two virtual machine instance stacks in the server software layer utilize similar components and programming language to increase the productivity of Spheres software development, system software maintenance, and system administration. The software will be deployed on an OpenStack installation on the UMBC BlueWave computing platform. OpenStack is a free open-source cloud computing infrastructure-as-a-service solution written in Python for building public and private clouds. OpenStack is a modular component-based architecture with virtualization components for compute, storage, networking, identity, image, and other services including autoscaling and load balancing, all features that will be leveraged by Spheres. The OpenStack admin instance provides a dashboard graphical user interface for managing OpenStack resources. It is used to create, deploy, and scale the Spheres server instances.

The Spheres server instance will host the Django web application framework which comprises a set of Python web applications. Django implements a model view controller (MVC) design pattern. The Django open-source developer community provides and maintains apps for user and content management (as well as a large library of other apps) that will be used to construct the Spheres Web Portal. Additional apps will be constructed and integrated that will implement the Spheres Web Services. Django contains a URL dispatcher for directing service requests to the appropriate app service handlers that all share a common MySQL database. HDF file storage will be integrated separately on this instance. The projects and users maintained within the MySQL database will refer to the HDF files within the HDF file storage each file implementing a sphere, one per participant smartphone.

Both virtual machine instances will be constructed on the open source CentOS Linux distribution. The Security Enhanced Linux (SELinux) Linux kernel security module will be integrated in the spirit of enhanced security for those Spheres deployments that require this level of security. In the event of integration problems in the prototype deployment, the Ubuntu Linux distribution with AppArmor, a Linux kernel security module for Ubuntu, will be evaluated as a replacement for the security requirement. CentOS and Ubuntu are currently the two most popular Linux distributions used on OpenStack deployments, with Ubuntu currently being the most popular.

E. Spheres Participant Software

As illustrated in Figure 7, the participant software layer is composed of a software stack containing a set of software components. Some components will be developed as part of this research and some are open-source off-the-shelf components that will be integrated. This layer of the Spheres system software is a single mobile app that provides the user interface and services for sensing project participants. The interface from this layer to the server software layer is over HTTPS using the Spheres API Library for Android that encapsulates the required REST request and response messaging interface to the server-side web services. The interface to sensors internal to the smartphone is through the Android SDK's Sensors API. The interface to external sensors in wireless sensor nodes is over BLE through the Android SDK's BLE API using the BLE Channel Description and BLE Channel Manager specifications shared between the Spheres Participant Mobile App and the Spheres software running on wireless sensor nodes.

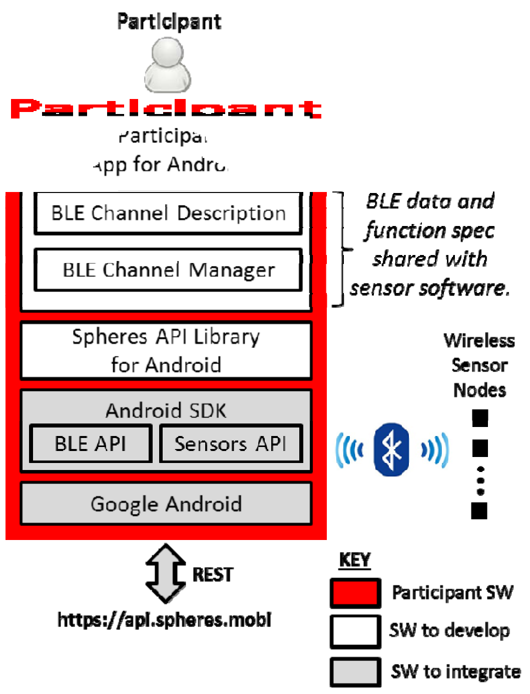

Figure 7. Spheres Participant Software

Users with the participant role will download the Spheres Participant Mobile App for Android from Google Play (as established previously, the Spheres prototype will be developed first for Android smartphones). This app will allow participants to participate in projects, configure privacy settings, configure the sensor data they wish to contribute, and view their contributions on time plots and maps. The Android SDK and associated libraries will be used to build the Participant Mobile App for smartphones running Android.

F. Spheres Sensor Software

As illustrated in Figure 8, the sensor software layer is composed of a software stack containing a set of software components that execute on a microcontroller unit (MCU) module within a wireless sensor node. The wireless interface is Bluetooth Low Energy although other interfaces could be supported; however BLE is currently the most popular. The level of integration and capability of a wireless sensor node depends on the vendor providing the node. In some cases, vendors provide an API library that you use on another device within your software that will connect to their node (in our case, this would be the Spheres Participant Mobile App). In other cases they may provide only the MCU runtime and a set of drivers for initializing node devices (devices within the MCU module, the sensor module, and the BLE module). In most cases, either the vendor of the wireless sensor node or the sensor module will provide sample code that will convert the analog-to-digital converted sensor readings into measurement values with units. Figure 9 shows the case where there is no API to the node and the software must be developed which will consist of an MCU Sensor App and the BLE Channel Description and BLE Channel Manager specifications shared with the Spheres Participant Mobile App.

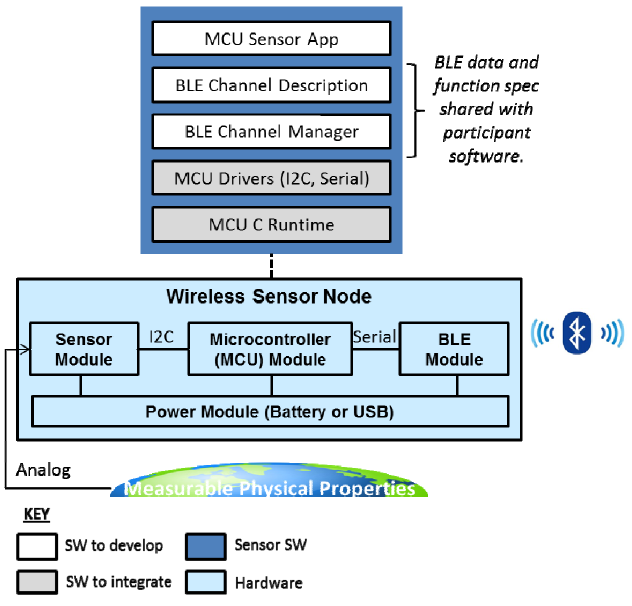

Figure 8. Spheres Sensor Software

The MCU Sensor App services requests from the Spheres Participant Mobile App over the BLE Channel. Services include: connect, disconnect, read, and reset. The BLE Channel Description defines the sensor data channels (channel name and data type) for the read request. The BLE Channel Manager performs the requested channel operation. The MCU Drivers use the MCU Runtime for sensor channel configuration & reading. The MCU Runtime is typically a C runtime environment supporting both interrupt and polling methods for sensor data reading at the request of the BLE Channel Manager.

VII. Challenges

There will be several challenges to develop a Spheres prototype. These challenges and risk mitigation plans include: Challenge #1 - the diversity of smartphone platforms, both operating systems and smartphone products that use them; to mitigate risk, one platform will be chosen; Challenge #2 - the diversity of smartphone sensors, both internal and external, and access to the sensor data from public APIs; to mitigate risk, a subset of internal sensors and one well-supported external sensor with public APIs will be demonstrated; Challenge #3 - sensor data security and privacy; security and privacy issues will be addressed at each incremental build phase of the prototype; Challenge #4 - smartphone sensing participation; for the prototype, volunteers will be solicited in compliance with all UMBC regulations regarding research participation; Challenge #5 - system complexity and integration; the risk is mitigated by incremental builds and testing of the software layers, stacks, components, and interfaces which will be performed on the UMBC BlueWave high performance computing system.

VIII. Conclusion

In this paper, I compiled the results of my research on sensing as a service using smartphones and to defined a web services framework called Spheres that implements smartphone sensing as a service using mobile, cloud, and wireless sensor network technologies and open source software components. Several related research studies were presented and related to Spheres. The architecture of Spheres was presented as a set of functionally cohesive software layers with loosely coupled standards-based interfaces. Each software layer was presented as a set of software stacks each comprising a set of software components. The software components comprising the system are either available as open source or will be developed as open source and shared publically. Several challenges to develop and demonstrate a prototype were identified.

IX. Acknowledgment

I would like to acknowledge the support of my Ph.D. research advisor, Milton Halem, Ph.D., Research Professor, Department of Computer Science and Electrical Engineering, UMBC. Dr. Halem provided significant guidance and steering of the direction of my research. I would also like to acknowledge the support of a professional colleague, Lorin Hochstein, Ph.D., Lead Software Engineer at SendGrid and active developer on OpenStack. Dr. Hochstein provided helpful feedback on the architecture of Spheres.

X. References

[1] "Smartphone Users Worldwide Will Total 1.75 Billion in 2014" [Online]. Available: http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536. Retrieved October 7, 2014.

[2] BlueWave [Online]. Available: http://chmpr.umbc.edu/home.html. Retrieved January 15, 2015.

[3] Mark Allen Gray, Philip Newsam Scherer ,"Web Services Framework for Wireless Sensor Networks", SERVICE COMPUTATION 2014 : The Sixth International Conferences on Advanced Service Computing, IARIA, May 2014, pp. 15-23.

[4] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Dissertation, University of California, Irvine, 2000.

[5] BlueGrit [Online]. Available: http://bluegrit.cs.umbc.edu/userdocs.php. Retrieved January 15, 2015.

[6] OpenStack [Online]. Avaiable: http://www.openstack.org/. Retrieved December 17, 2014.

[7] Google Maps Web Services [Online]. Available: https://developers.google.com/maps/documentation/webservices/. Retrieved December 17, 2014.

[8] HDF5, [Online].Avaialable: http://www.hdfgroup.org/HDF5/. Retrieved December 11, 2014.

[9] International Data Corporation, "Smartphone OS Market Share, Q3 2014" [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Retrieved December 16, 2014.

[10] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs, "Disruptive technologies: Advances that will transform life, business, and the global economy", McKinsey Global Institute, May 2013.

[11] Peter Mell and Timothy Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, United States Department of Commerce, September 2011.

[12] Xiang Sheng, Xuejie Xiao, Jian Tang, and Guoliang Xuey, "Sensing as a Service: A Cloud Computing System for Mobile Phone Sensing", Sensors, 2012 IEEE.

[13] Mazlan Abbas, "Sensing-as-a-Service: The New Internet of Things (IOT) Business Model", MIMOS Berhad, April 2014.

[14] Willard Tu, "Sensors as a Service on the Internet of Things", White Paper by ARM Limited, 2014, pp 1-8.

[15] Dr. Steven Rogers, Lt. Col. David Ryer, and Mr. Jeffrey Eggers, "Sensing as a Service", Deputy Chief of Staff for Intelligence, Surveillance, and Reconnaissance, ISR Horizons Future Vision, United States Air Force, May 2014, pp.1-6.

[16] International Data Corporation, "Smartphone Vendor Market Share, Q3 2014" [Online]. Available: http://www.idc.com/prodserv/smartphone-market-share.jsp. Retrieved December 16, 2014.

[17] Amazon Mechanical Turk [Online]. Available: https://www.mturk.com. Retrieved December 17, 2014.

[18] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing", MobiCom'2012.