MECA: Mobile Edge Capture and Analysis Middleware for Social Sensing Applications

Fan Ye, Raghu Ganti, Raheleh Dimaghani, Keith Grueneberg, Seraphin Calo
IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
{fanye, rganti, rbdilmag, kgruen, scalo}@us.ibm.com

ABSTRACT

In this paper, we propose and develop MECA, a common middleware infrastructure for data collection from mobile devices in an efficient, flexible, and scalable manner. It provides a high level abstraction of *phenomenon* such that applications can express diverse data needs in a declarative fashion. MECA coordinates the data collection and primitive processing activities, so that data can be shared among applications. It addresses the inefficiency issues in the current *vertical integration* approach. We showcase the benefits of MECA by means of a disaster management application.

1. INTRODUCTION

The past decade has seen the rise of social networks such as Facebook, Twitter, FourSquare, and Google+ and the emergence of smartphones equipped with sensors and Internet connectivity capabilities. The marriage of these technologies, smartphones and social networks, will result in novel applications that leverage the data collection capabilities of large numbers of smartphones by crowdsourcing. For example, real-time traffic monitoring for Google maps is enabled through individuals sharing their location and speed information from their smartphones. We believe that this integration will be a significant driver for social networking applications for disaster management. For example, the recent BP oil spill can be monitored by individuals by sharing pictures of the spill through Twitter or Facebook. A chemical spill or the air quality (using an air quality sensor) can be monitored in a similar fashion [7, 2]. Such information can be aggregated, processed, and then consumed by individuals (e.g., Do not go through Main Street as it is flooded) or by decision makers and/or public agencies (e.g., Main Street is flooded and ten people are trapped, send disaster response teams promptly to Main street).

Existing approaches to such applications have largely been based on *vertical integration.* Each application needs its own software agent running on the devices collecting data specific to the application's purpose, and a backend module for aggregating and processing the collected data to generate the desired results. Such a vertical approach aims to optimize the performance of a single application. However, with the proliferation of such applications, great inefficiency and even conflicts may arise for both the software agents and the backend modules. Many times, these applications need the same type of raw data, and access the same physical sensor. The software agents compete for access, and collect the same data again and again. Such uncoordinated collection activities should be avoided. For applications in the same domain (e.g., traffic and transportation related), they also repeat common primitive processing of the raw data to extract information of higher semantic content. For example, the raw time series of acceleration can be processed to detect the existence of potholes [6].

In this paper, we propose MECA (Mobile Edge Capture and Analytics), a common middleware for data collection from mobile devices that addresses such issues. MECA is intended to support a diverse variety of data and information needs from many different applications. It provides a high level abstraction of *phenomenon*, such that applications can easily express their data and information needs in a declarative fashion. MECA is able to identify common data and information needs across different applications. It ensures that the same kind of primitive processing of raw data is done only once, and the results are shared among these applications. Thus it avoids the redundancy and conflicts in the vertical approach.

MECA uses a configurable framework to select and configure devices based on the requirements from many applications. A common software agent capable of collecting different types of data runs on the devices. It receives instructions from MECA and sends back the desired data. MECA conducts optional primitive processing on the raw data to extract higher level information, and passes back the "half-cooked" data to applications. We have developed the MECA prototype, and we will showcase its benefits and capabilities by means of a disaster management application that provides services for both individuals and authorities in a chemical spill scenario.

We will present the high level abstractions and architecture of MECA in detail in Section 2. Then we describe the application for disaster response in Section 3. We discuss related work in Section 4 and state our conclusions in Section 5.

2. MECA ARCHITECTURE

2.1 High Level Abstractions

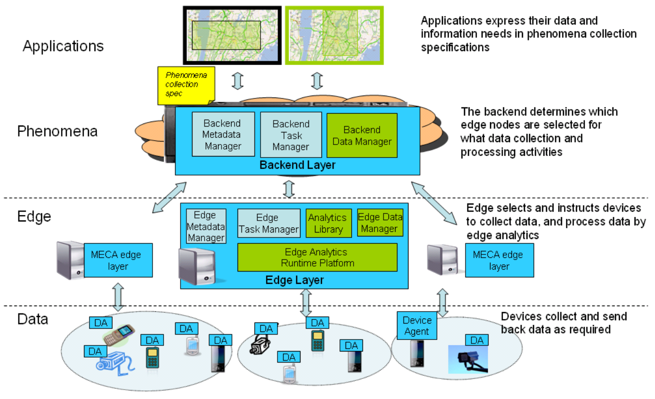

Applications
Applications express their data and information needs in phenomena collection specifications
Phenomena collection spec
Phenomena
Backend Metadata Manager
Backend Task Manager
Backend Data Manager
Backend Layer
The backend determines which edge nodes are selected for what data collection and processing activities
Edge
Edge Metadata Manager
Edge Task Manager
Analytics Library
Edge Data Manager
Edge Analytics Runtime Platform
Edge Layer
MECA edge layer
MECA edge layer
Edge selects and instructs devices to collect data, and process data by edge analytics
Data
DA
Device Agent
Devices collect and send back data as required

Figure 1: The three-layer architecture and components in MECA.

MECA provides a common infrastructure to collect real time data for different kinds of applications simultaneously. The middleware exposes a high level abstraction to applications, such that they can express their data needs in a declarative fashion using *phenomenon collection specifications*. A *phenomenon* is essentially the occurrence of a certain kind of event at a particular location and time. For example, the detection of a pothole on a road at a certain location and time, is a phenomenon. It is intended to provide information at semantic levels higher than the *raw data* as captured by device sensors directly.

Such a high level abstraction is motivated by two observations. First, the raw data generated by physical sensors on devices usually are high volume and not directly consumable by applications. For example, a public facility maintenance application may need to detect potholes on a road so that proper repairs can be done on time. We know that potholes can be detected from the 3-axis acceleration data from smartphones [7] carried by drivers. However, certain processing on the raw time series data has to be performed to identify the location of potential potholes. It is shown that such processing can be done on devices to improve the semantic level of information, and greatly reduce the volume of data. Second, different applications, especially those in the same domain, may have common information needs. For example, the raw GPS samples from passengers and drivers can be aggregated to identify their commuting trajectories, which are useful for both real time traffic alerts, and long term urban road network planning. To avoid duplicate efforts in these applications, it is more efficient to share the data collection and primitive processing inside MECA and have the two applications tapping the same stream of trajectory data.

A *phenomenon collection specification* consists of three parts: the type of the phenomenon needed, the geographic scope, and the time window in which data should be collected. Each type of phenomenon has a clear definition of the data structure and semantics. For each type, there is at least one *edge analytic* that can transform certain kinds of raw data into the phenomenon. These analytics are dynamically invoked by MECA when needed. The collection of available analytics, thus phenomenon types, are extensible. Once a new edge analytic is added to the analytics library, the phenomenon type it produces will be made available to applications.

2.2 Architecture Components

The MECA architecture (shown in Figure 1) consists of three different logical layers: phenomena, edge and data. The phenomena layer usually resides at the backend (e.g., a data center). It is responsible for receiving phenomenon collection specifications from applications, for coordinating the overall data collection according to the stated policies, and for sending back the phenomenon data to applications. The edge layer resides on the network edge (e.g., base stations in cellular networks). Its main function is to receive collection requirements from the phenomena layer, manage the data collection among a subset of local devices, and run edge analytics for primitive data processing. The data layer is on devices. It is a software agent running on devices, receiving data collecting instructions from the edge node, producing data and sending them back to the edge.

The modules inside each layer are shown in Figure 1. The phenomena layer has three components: the Collection Task Manager (CTM), the Backend Metadata Manager (BMM), and the Backend Data Manager (BDM). The CTM exposes an interface to applications to receive their phenomenon collection specifications. Upon receiving a specification, it queries the BMM, which maintains metadata about edge nodes, including which phenomenon types are available, and the respective geographic scope. It then selects appropriate edge nodes, sends the specification to them so that they can start data collection from devices. The CTM creates and maintains the state information for each collection task, such as which edge nodes are involved. When a task finishes either due to the end of the time window or termination by the application, the states are cleared. The BDM is responsible for receiving and aggregating data from edge nodes, such that data intended for one collection task can flow continuously to the application.

The edge layer is responsible for the data collection from a set of local devices, and for running edge analytics for primitive processing required by the specified phenomena. It consists of several components: the Edge Task Manager (ETM), which maintains the state information about collection tasks at the network edge (e.g., which sensing activities on which devices are involved for which task), and coordinates the device activities and edge processing; the Edge Metadata Manager (EMM), which maintains registration and status information about devices, such as their locations and energy levels; the Edge Analytics Library, which maintains a collection of edge analytics for the ETM to invoke; the Edge Analytics Runtime Platform, which is a container in which to deploy edge analytics; and, the Edge Data Manager (EDM) is in charge of aggregating data from different devices intended for the same collection task, and send the data to the Backend Data Manager for further aggregation.

Upon receiving a specification from the phenomena layer, the ETM first queries the EMM to identify which edge analytics can produce the required phenomena, and which devices can produce the raw data needed. Then based on the locations, energy levels, and the cost of data collection and processing, a subset of devices are chosen, and data collection instructions are sent to them. If an edge analytic is required for primitive processing, the ETM invokes the analytic from the library, and runs it on the platform.

The data layer is the software agent running on devices. Its main purposes are two-fold: receive configuration and collection instructions from the edge, and send back data generated by physical sensors. Each device needs to register itself with a certain edge node (e.g., physically closest) to make itself available for data collection. It reports the types of raw data it is capable of producing, and periodically updates the edge node about its location and energy level such that the edge node can make a proper selection and configuration determination when a collection task arrives.

The raw data from devices will be sent to the EDM for aggregation. If an edge analytic was invoked, it will take the aggregated data and transform them into the desired phenomenon. Such phenomena are passed to the BDM at the backend, and eventually sent back to applications.

MECA facilitates sharing of both the raw and phenomenon data across applications. When the same kind of phenomenon data is requested multiple times at one edge node, the existing collection and processing activities will be reused as much as possible. Raw data from a device, or phenomenon data from an edge analytic, will be sent to the EDM and shared by multiple applications.

MECA seeks to strike a balance between efficiency, flexibility and complexity. The phenomenon abstraction is intended to provide *primitive* processing that improves the efficiency, but not to substitute for the more complex processing unique to each application. Depending on its goal, each application may have distinct algorithms and procedures to further process the data to obtain the final results. Such processing should be carried out by applications, not MECA. To support a wide range of applications, MECA does not preclude the collection of raw data. The primitive processing of raw data depends on the availability of edge analytics that transform raw data into "half-cooked" data. If no edge analytics are available to perform certain primitive processing required by an application, it can still request raw data from MECA and conduct the processing itself at the application level.

3. A DISASTER MANAGEMENT APPLICATION

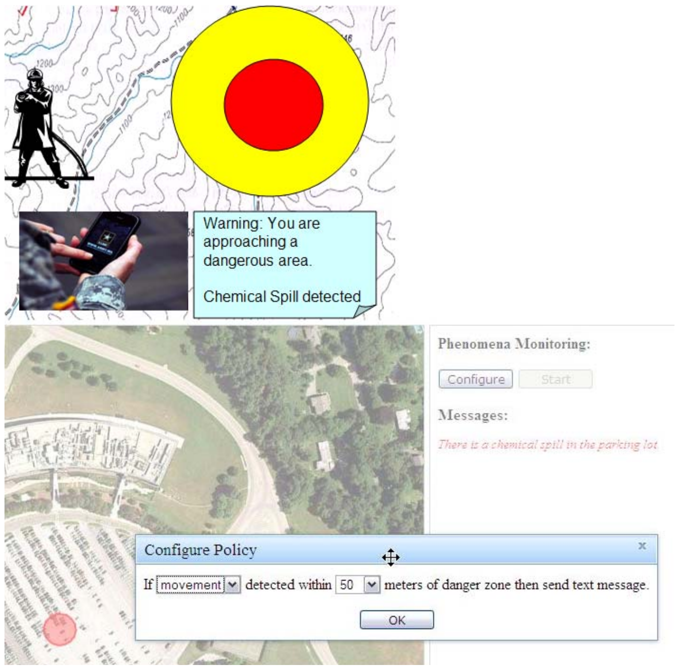
In this section, we will describe a disaster management application that provides services for both individuals and authorities in a chemical spill scenario. The application uses the data and information collected by the MECA prototype. It demonstrates the benefits of MECA compared to existing vertical approaches.

- *High level abstraction of phenomena collection specification.* The application does not need to interact directly with any of the mobile devices. Unlike the vertical approach, it does not need to be concerned about the dynamic changes such as device mobility and resource variations. MECA handles all those dynamics and makes them transparent to the application. The application only needs to send phenomenon collection specifications about the phenomenon types, geographical scopes and time durations to MECA.

- *Concurrent collection of phenomenon and data of different types.* The application provides an alert service to individuals when they get too close to a danger zone, and warning services for authorities such as the fire department to track the movements of fire fighters. These services require phenomenon and raw data of different types, which are collected by MECA simultaneously.

- *Intelligent and efficient processing at the network edge.* MECA has edge analytics that conduct primitive processing on the raw data collected. The resulting phenomenon data has much less volume, and carries higher level semantics that are more easily consumable by the application.

- *Metadata and policy driven device selection and configuration.* MECA maintains the metadata of devices such as their locations, and data collection capabilities. There are also policies regarding how devices should be chosen and configured based on their resource levels. MECA selects and configures a subset of devices based on the metadata and policies.


Warning: You are approaching a dangerous area.

Chemical Spill detected

Phenomena Monitoring:

Messages:

There is a chemical spill in the parking lot

Configure Policy

If movement detected within 50 meters of danger zone then send text message.

Figure 2: Sending Alert Messages with MECA

We have developed a disaster management application using data collected from mobile devices. It illustrates the benefits of MECA for social networks in both natural disaster and terrorist attack scenarios. The application sends alert messages to individuals who have subscribed to the system to receive notifications of nearby dangers. Examples include approaching a downed power line, a chemical spill, or radioactive contamination. It also enables the authorities to track the status of emergency response personnel: if a person does not move for a long period of time, or move very slowly, it may indicate an injury or difficulty that needs attention.

In a chemical spill scenario, an individual notices hazardous material and calls to report the incident. A dispatcher receives the call and collects information such as the type of emergency, location and time when first noticed and informs first responders. Police, HAZMAT, firefighters, and a medical team are among the responders sent to the area. Police or firefighters are usually first to arrive and the first thing they do is to isolated the high-risk area, also known as a "hot zone" from so-called "cold-zone" (e.g., the safe area). A small area separates these two zones which is referred to as a "warm zone."

In Figure 2, we present the hot zone and warm zone as concentric circles; the former as the inner circle and the latter as the outer circle. Everyone who enters the hot zone is contaminated and has to go through decontamination process to enter the cold zone. Other responders including the medical team cannot enter the hot zone until it is declared safe by special forces. The fire captain and other team leaders need to keep their team members safe and away from the hot zone. If a person enters the warm zone, an alert message is sent to warn that person from potential danger of chemical exposure.

After special forces including HAZMAT clean the area and safely separate contaminated people, they declare the area safe. Hence other first responders including the medical team and firefighters can enter the area. From this point forward, it is critical for team leaders to monitor the movement mode of their team members in the response if non-movement or slow motion are detected.

The context of this application determines the types of phenomenon that are needed from the MECA middleware. The two phenomena of interest in this scenario and processed from different data types are: entering a zone of interest which utilizes location data from GPS, and a movement mode phenomenon utilizing acceleration data. Decision makers or individuals each specify the phenomenon of interest at a given location and within a time window. They input the phenomenon collection specifications through a user-friendly template by identifying the phenomenon of interest in a rectangular geographical area and a certain time duration.

MECA processes the specifications, and identifies edge nodes that are capable of producing these two phenomenon types and whose responsible collection areas overlap with the specified geographical scopes. A subset of suitable edge nodes are selected and the phenomena layer forwards the collection specifications to them. These edge nodes in turn identifies that there exist edge analytics that can produce the movement mode and approaching zone phenomenon, which further require the acceleration and GPS location data from devices as raw data input. Thus the edge nodes instruct these devices for relevant data collection. Finally the data are processed and the phenomena sent to the application, which generates danger zone alerts and movement mode warnings for individuals and authorities.

4. RELATED WORK

Applications that rely on sensor data collection and sharing from mobile devices within a large community are termed as participatory/opportunistic sensing [1] applications. Examples of these applications are CommonSense [2], CarTel [6], Nericell [7], BikeNet [3], and GreenGPS [4]. CommonSense uses specialized air quality sensing hardware integrated with a smartphone and enables pollution monitoring at scale by processing data collected and shared from individuals. CarTel and Nericell collect GPS, acceleration, and microphone sensor data to provide services for traffic monitoring applications. GreenGPS collects fuel consumption data from vehicles using smartphones to compute fuel-optimal routes. BikeNet provides a biking quality map through shared data collected by individuals. These applications are largely based on the vertical integration model where application specific device agents and backend software are developed. When there is a multitude of such applications, their co-existence on the same set of underlying devices is problematic. Inefficiencies and conflicts are inevitable. An in-depth analysis and summary of the drawbacks of such vertical approaches can be found in [5].

5. CONCLUSION

We have presented MECA, a common infrastructure for the simultaneous collection of diverse data and information from mobile devices for different applications. MECA provides a high level abstraction to applications such that they can express their data and information needs in a declarative way using phenomenon collection specifications. MECA translates the specifications into the selection and configuration of edge nodes and devices for proper raw data collection and edge analytics processing. Thus applications do not need to interact directly with devices and can be built more easily by focusing on application specific logic.

MECA enables the sharing of data and phenomena commonly required by different applications. It improves the bandwidth utilization and energy efficiency by collecting the same raw data and conducting the same primitive processing only once. MECA avoids duplicate data collection and redundant primitive processing by identifying such commonalities and reusing the data across applications.

We illustrate the benefits of MECA in a disaster management application where the location and acceleration data collected from mobile devices are used to help individuals avoid danger zones, and authorities track the status of emergency response personnel.

6. ACKNOWLEDGMENTS

We thank Doug Freimuth for the valuable discussions and suggestions; we are also grateful for the insightful feedbacks and guidance from Dinesh Verma which are indispensable for this project.

7. REFERENCES

[1] J. Burke et al. Participatory sensing. Workshop on World-Sensor-Web, co-located with ACM SenSys, 2006.

[2] P. Dutta et al. Demo abstract: Common sense: Participatory urban sensing using a network of handheld air quality monitors. In *Proc. of ACM SenSys*, pages 349–350, 2009.

[3] S. B. Eisenman et al. The bikenet mobile sensing system for cyclist experience mapping. In *Proc. of SenSys*, November 2007.

[4] R. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. Abdelzaher. Greengps: A participatory sensing fuel-efficient maps application. In *Proc. of MobiSys*, pages 151–164, 2010.

[5] R. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[6] B. Hull et al. Cartel: a distributed mobile sensor computing system. In *Proc. of SenSys*, pages 125–138, 2006.

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