MECA: Mobile Edge Capture and Analysis Middleware for Real Time Decision Making

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

Abstract—In real time decision making scenarios (e.g., a coalition environment), relevant and accurate information must be made available to decision makers in a timely manner. However, decision makers face a dual problem of both information overloading and starvation: the overwhelming majority of information from various sources may not be relevant to the investigation goal, while a significant gap exists in useful and pertinent information. To address such problems, we propose and develop MECA, a common middleware infrastructure for data collection and analysis from various data sources, especially modern mobile devices. MECA provides a high level abstraction of a phenomenon specification such that applications can accurately express their information needs in a declarative fashion. Thus only data relevant to the investigation are gathered. MECA supports primitive processing within the infrastructure to transform raw data to information at semantic levels much closer to decision making. It coordinates the data collection and transformation activities, such that data and information can be shared across applications without duplicate efforts. We showcase the capabilities of MECA through disaster management and military applications. Finally we discuss how to expand it to further automate the decision making process.

keywords: data collection, decision making, real time

I. INTRODUCTION

Decision makers are often under stringent time limits to make the most appropriate decision based on all information that is available. Many times they face a dual problem of both information overloading and starvation. On one hand, they may have at their disposal large amounts of data and information from numerous sources, possibly coming back in real time. On the other hand, they may not possess enough relevant information that is valuable to their current investigation goal. There may exist a significant gap between what is needed for the specific investigation and what is available: the majority or most information available at hand might be irrelevant.

In this paper, we propose MECA (Mobile Edge Capture and Analytics), a common middleware for data and information collection to aid decision makers in acquiring information relevant to their investigation goals. MECA can collect information from various sources, especially modern mobile devices (e.g., smartphones) that are equipped with various physical sensors and are capable of producing different kinds of data and information. This is important because such devices have emerged in the past several years and resulted in novel applications that leverage the data collection capabilities of large numbers of devices by crowdsourcing. For example, real-time traffic monitoring for Google maps is enabled partly through individuals sharing their location and speed information from their smartphones. A chemical spill or the air quality (using an air quality sensor) can be monitored in a similar fashion [1], [2].

We believe that this integration will be a significant driver for real time decision making applications. Data and information from large numbers of mobile devices can be aggregated, processed, and then consumed by decision makers. In a disaster response scenario such as earthquakes or hurricanes, the planners need to identify the locations and numbers of stranded people, and determine the best routes to send emergency supplies via roads that are still passable. In an urban scouting scenario, the military commanders need to gather information about the surrounding environment, vehicles and personnel. The mobile devices carried by coalition soldiers or friendly civilians can help gather such information for purposes such as identifying potential threats.

There has been quite some academic research work in building applications leveraging data and information from such mobile devices. However, 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 needs, and a backend module for aggregating and processing the collected data to generate desired results. Such a vertical approach aims to optimize the performance of a single application.

In contrast, MECA is intended to support a diverse variety of data and information needs from many different and possibly concurrent applications. With the proliferation of such applications, great inefficiency and even conflicts may arise for both the software agents and the backend modules in the vertical approach. 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 [3].

To address such issues, MECA provides a high level abstraction of a 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 a MECA prototype, and we will showcase its benefits and capabilities by means of applications in disaster management and military surveillance scenarios, via services provided for both individuals and authorities.

We will present the high level abstractions, architecture and implementation of MECA in detail in Section II. Then we describe the applications for disaster response and military surveillance in Section III. We discuss how we may further automate the data acquisition process to alleviate the burden for decision makers in Section IV. We compare MECA with related work in Section V and state our conclusions in Section VI.

II. MECA ARCHITECTURE

A. High Level Abstractions

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 possibly 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 [1] 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. The geographic scope can take different forms, such as a polygon shape with the coordinates of its vertices, or a postal address which is more human friendly. The time window can be a period until some future time point, possibly with recurring frequencies.

MECA supports phenomenon types at different semantic levels: complex, simple and raw. For each complex type, there is at least one edge analytic that can transform certain kinds of data into the phenomenon. These analytics are dynamically invoked at edge nodes when needed. They represent information that requires processing at depth or breadth not possible on individual devices. For example, the average or distribution of temperature readings in an area cannot be provided by any single device; while the extraction of vehicle plate numbers from pictures may require computing resources beyond what is available on individual devices. Simple types are those that can be produced within the processing capability of individual devices, such as spikes in acceleration time series beyond certain threshold value. Finally, raw data are those captured by device sensors as-is. A geo-coded image can be an event about the observation of certain objects at particular time and location. Applications can require raw data being passed back directly if needed.

MECA seeks to strike a balance between efficiency and flexibility. The phenomenon abstraction is intended to provide common and most likely primitive processing that improves efficiency, but not to substitute the more complex, possibly application-specific processing. 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.

On the other hand, 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 some unique processing required by an application, it can still request raw data from MECA and conduct the processing itself to accomplish its goal.

The collection of available analytics, thus phenomenon types, are extensible. MECA maintains metadata about the types of phenomenon available to applications. Once a new edge analytic, or new type of device sensor (thus raw/simple phenomenon types), is added into the system, the new phenomenon type will be made available to applications.
**B. 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 detailed control and data flows of how different components interact with each other across the three layer are shown in Figure 2 and Figure 3. 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 be aggregated and sent back to applications.

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.

**C. Implementation**

We have implemented MECA middleware in Java. The Cloud Task Manager provides an RMI interface to applications for them to send phenomenon collection specifications. The application receives a handle to the collection task by which it can manipulate the task, such as invoking termination. To receive data without periodic polling, the application also passes a callback handle to the CTM. The handle is invoked by the Cloud Data Manager when data have been received for the application.

The API that CTM provides to applications is as follows:

```java
public interface CTMIfc extends Remote {
    public EventCollectionTaskIfc receiveEventCollectionSpec(EventCollectionSpec spec, ApplicationIfc app) throws RemoteException;
}
```

where "ApplicationIfc" is a handle to the application itself,
Fig. 2. The control plane of the MECA architecture transforms the application’s phenomenon collection specifications down to instructions in appropriate devices.

and "EventCollectionSpec" is the phenomena collection specification.

To facilitate sharing of phenomenon data across applications, existing collection and processing activities will be reused whenever possible. To this end, the Edge Task Manager maintains two sets of state information, one for active edge analytics, the other for the set of active data collection instructions for each subordinate device.

When a new phenomenon collection specification is received, the ETM finds out what kinds of edge analytics are needed, and which devices should run what data collection instructions. It then examines the two sets to find out whether any existing edge analytics and data collection instructions can be reused. If so, their corresponding reference counters are increased and the data is shared, without invoking new analytic or instruction instances. When a collection task is to be terminated, the reference counters of involved edge analytics and data collection instructions are decreased. They are terminated only when the reference counters reach zero – meaning that they are no longer needed by any collection task.

We have implemented a MECA agent on Android. The agent first registers itself with an edge node upon startup. One issue is how to notify devices efficiently when changes are made to their data collection instructions. We have the device agent periodically poll the associated edge node. The edge node maintains a monotonically increasing version number for the set of instructions for each device. Whenever a change is made to the set of instructions, the version number is increased by one.

The agent first sends its current version number to the edge node, which compares that with its current version. If the current version is newer, the edge node sends all the instructions to the agent, which then compares them to its current ones to find out which new ones to launch or old ones to terminate. Otherwise the agent does not need to change anything. This avoids frequent sending of instructions, which is unnecessary since most data collection activities run for relatively long periods and changes are not as frequent.

We plan to further optimize the process by using push notification instead of periodic polling. The edge node can send a notification to the device agent whenever changes are needed. Thus the agent does not need to periodically contact the edge node.

III. APPLICATION

In this section, we demonstrate the capabilities of our proposed middleware compared to existing vertical approaches with two applications where the applications consume data and information collected by MECA prototype. Some characteristics of that support provided by MECA to these applications...
High level abstraction of phenomenon 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 phenomena 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.

The first application presents a disaster management scenario where data collected from mobile devices is used for both individuals and authorities in a chemical spill scenario. This can be applied to both natural disaster scenarios and terrorist attacks.

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 moves 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 isolate the high-risk area, also known as a “hot zone” from the 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 4, 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 a 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.

In Figure 4, 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 a 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 about the potential danger from 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 to detect non-movement or slow motion.

The context of the application determines the types of phenomenon that are needed from the MECA middleware. The two phenomena of interest in this scenario are processed from different data types: 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.

The second application illustrates a military scenario where coalition partners share and access data from a variety of ISR (Intelligence, Surveillance and Reconnaissance) assets. Access to the data from these assets is controlled by policy rules defined by each coalition. In this scenario, ISR assets include...
acoustic sensors, video cameras, and sensor platforms that are used to identify and track enemy activity. Human intelligence is gathered about a situation in which an enemy vehicle has been seen transitioning through a civilian area. Command and Control (C2) assets of coalition partner A would like to collect more information from sensors deployed in the geographical area of the enemy sighting. It is determined that the sensor feeds controlled by coalition partner A are not sufficient for detecting the vehicle in the area of interest. C2 needs to discover sensors available from other coalition partners that can provide data feeds of the appropriate type in the specified area. A policy specification will be sent to the data collection service that indicates the attributes of the data that needs to be collected and the conditions that need to be satisfied. For example, a coalition partner may only want video imagery of a certain format from cameras in the specified area. Also, they may only want data from high-quality sources, so the policy may include a Quality of Information (QoI) threshold. On the other hand, coalition partner B may have policies that restrict the data that can be sent to coalition partner A. The data collection service will provide the available data feeds based on the specification and the evaluation of the policy rules controlling access to the sensors.

MECA processes the specifications, and identifies edge nodes that are capable of producing relevant phenomenon types with operational collection areas that intersect 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 identify that there exist edge analytics that can produce the movement mode and approaching zone phenomena, 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. In any given application, data is processed and the phenomena are sent to the application, which generates proper notification to be sent to decision makers to alert them about a danger zone, movement mode warnings for individuals and authorities, or suspicious activities under surveillance.

MECA utilizes the high-level abstractions to identify the phenomena of interest and manages the collection of related data for several applications. Different applications can utilize the middleware to collect the same raw data only once but use it for different purposes by transforming them into different phenomena types with respective edge analytics. This avoids undesired duplication in data collection and hence improves bandwidth utilization and energy efficiency. For example acceleration data can be utilized to extract the movement mode for both applications presented in this section.

IV. DISCUSSION ON AUTOMATED ACQUISITION

We envision that MECA can automate the data acquisition process for decision makers in the future. In many scenarios, decision makers do not have all the information needed to make the proper decisions when starting an investigation. For example, one may need to determine the threat level of some vehicles near the location of a group of soldiers. However, initially the decision makers only have very limited knowledge about the subject, e.g., only the detected location of the vehicles, and the time. They do not possess other knowledge, such as the vehicle make/model, or the number, age, and appearances of the passengers inside.

It may take multiple collection specifications to gather all such information, likely over an extended period of time. During this period, constant changes have to be made to the set of active collection specifications, such as adding new specifications to acquire certain missing information, adjusting the parameters of specifications to better acquire some needed information, or removing them upon successful acquisition of such information. It is quite labor-intensive for human decision makers to constantly watch what has been obtained, what is still missing and make constant updates to the set of active collection specifications.

We believe decision makers should be relieved from such burdens. The system should automatically acquire relevant information, and humans need to be consulted only once in a while to guide the system. To this end, we plan to investigate a semantic driven approach to automate the data acquisition process. We break a complex and potentially lengthy investigation process into a series of interactive question and answer sessions. Each question is focused on one aspect of the subject. To answer the question, certain knowledge, much of which might be missing, is required. After the decision maker asks the question, the system constructs appropriate phenomenon collection specifications and adjusts them over time to automatically acquire relevant knowledge. Based on the obtained information, the decision maker poses the next question. This is repeated until eventually the decision maker has enough information to make a proper decision.

To drive such an automated approach, we plan to use an ontology to describe the knowledge relevant to each type of question. For example, to answer how likely a reported vehicle sighting matches the profile of a fleeing suspect, the decision maker needs to know the make/model/color/plate of the vehicle, and the height/clothes/hair-eye colors of the driver. An ontology snippet can describe these conceptual entities and their relationship. For each type of question, a corresponding ontology snippet can guide the system about what are the pieces of required information.

To facilitate the extraction of information from raw data, the ontology is augmented with semantic descriptions of analytics. For each entity in the ontology, multiple analytics that are capable of producing information on that entity, and their input data types, can be described as well. For example, to acquire the color of a vehicle, a color extraction analytic can produce it from an image, or a keyword detection analytic can extract it from a text message.

Finally, for the system to keep track of what knowledge has been acquired and what is still missing, a status is associated with each entity in the ontology (see "color" as an example in Figure 5) . The status describes whether information on that entity has been obtained, and potentially how reliable the information is. Based on such a status, the system determines
what phenomenon collection specifications to construct for acquiring missing information, or how to adjust the parameters of existing ones to improve the reliability of existing information.

The benefits of such an approach are two-fold: First, the decision makers do not need to keep track of the details in maintaining phenomenon collection specifications. They just need to pose questions and wait for the system to acquire needed information. Second, new types of questions can be supported easily by defining the corresponding ontology snippets. Once defined, a snippet can be reused to answer future questions of the same type. This extensibility is critical for the system to support a wide variety of application domains, rather than it being tied to some pre-determined ones.

We also plan to design algorithms to optimize the latency of the data acquisition process. There are two aspects: the same knowledge can be acquired via different kinds of data, most likely with different costs and latencies; and, the same raw data, after going through different analytics, can result in different pieces of knowledge. The optimization algorithm needs to determine, based on the current status of available knowledge, which kinds of raw data and what analytics to use that would minimize the time in acquiring the knowledge.

V. RELATED WORK

Applications that rely on sensor data collection and sharing from mobile devices within a large community are termed as participatory/opportunistic sensing [4] applications. These applications and the corresponding data collection mechanisms are the most relevant to our work, as they address the situation where data needs to be collected from mobile devices in a continuous manner.

Early deployments of participatory sensing measured traffic congestion levels in cities, examples of which include MIT’s CarTel [3] and Microsoft Research’s Nericell [1]. CarTel utilizes specialized devices installed in cars to measure the location and speed of cars and transmit the measured values using public WiFi hotspots to a central server. This central server can then be queried to provide information such as least delay routes or traffic hotspots. On the other hand, Nericell utilizes individuals’ mobile phones to not only determine average speed or traffic delays, but also to detect honking levels (especially in countries like India where honking is common) and potholes on roads. Another example is ParkNet [5], an application that detects available parking spots in cities using ultrasonic sensing devices installed on cars combined with smart phones.

Another class of example applications is in the area of environmental monitoring, where individual mobile nodes collect sensory measurements from various environmental phenomena. An example prototype deployment for pollution monitoring is Common Sense [2]. Common Sense uses specialized air quality sensing devices that communicate with mobile phones (using Bluetooth) to measure various air pollutants (e.g., CO$_2$, NO$_x$). These devices when deployed across a large population, collectively measure the air quality of a community or a large area. Similarly, one can utilize microphones on mobile phones to monitor noise levels in communities. These measurements can be applicable to military scenarios, where a group of soldiers can measure various phenomena (e.g., crew levels, noise) collectively.

Yet another class of applications are in the social networking domain, where individuals share sensed information amongst themselves (through a social network). As an example, individuals can share their exercise data (e.g., how much time one exercises in a day) and compare their exercise levels with the rest of the community. They can use this comparison to help improve their daily exercise routines. Example deployments include BikeNet [6] and DietSense [7]. In BikeNet, individuals measure location and bike route quality (e.g., CO$_2$ content on route, bumpiness of ride) and aggregate the data to obtain “most” bikeable routes. In DietSense, individuals take pictures of what they eat and share it within a community to compare their eating habits. A typical use case for this is for a community of diabetics to watch what other diabetics eat and control their diet or provide suggestions to others.

We observe that the above 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 [8]. We propose MECA as an infrastructure that can address these inefficiencies and lead to the evolution of the co-existence of multiple such applications that will drive applications requiring data from a large number of mobile devices (while monitoring...
a phenomenon of interest).

There has been quite some research work about the optimal matching of information requests to sensors [9], [10], [11], possibly using more expressive ontology descriptions [12], [13]. MECA currently uses very simple phenomena type matching to identify suitable devices to particular phenomena collection requests. It is possible to leverage such previous work for more flexible and adaptive device matching. A few proposals [14], [15] have studied context predication and impact of human when integrating sensor networks with mobile devices. MECA can be extended to manage data from sensor networks as well, provided that a gateway exists to bridge edge nodes and a sensor network.

VI. CONCLUSIONS

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 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 capabilities 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. We also describe a military application, where coalition partners can help each other collect information that is not directly observable to the requesting party.

ACKNOWLEDGMENTS

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

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES


