An Infrastructure Supporting Considerate Sensor Service Provisioning

Jia Zhang[1], Nimish Radia[2], Zhipeng Li[1], Norman Xin[1], Yuan Ren[1], Prateek Sachdeva[1], Puja Subramanyam[1], Sky Hu[1], Song Luan[1], Lydian Lee[1], Bo Xing[2], Du Li[2], Jordan Cao[3], Ted Selker[1], Bob Iannucci[1], Martin Griss[1], Anthony Rowe[4]
[1]Carnegie Mellon University – Silicon Valley, USA
[2]Ericsson, USA
[3]SAP, USA
[4]Carnegie Mellon University, USA
jia.zhang@sv.cmu.edu, nimish.radia@ericsson.com, du.li@ericsson,com, jordan.cao@sap.com, ted.selker@sv.cmu.edu, bob@sv.cmu.edu, martin.griss@sv.cmu.edu, anthony.rowe@cmu.edu

***Abstract*—**The Internet of Things (IoT) aims to integrate the digital world of the Internet with our encompassing physical world. However, existing IoT systems do not provide considerate services, meaning that sensors dynamically "collaborate" to provide context-aware federated sensor data, tuned to human needs. This paper reports our on-going work developing a sensor service federation and provisioning infrastructure. A novel approach is presented to build social sensor networks to record and study historical interaction patterns among sensors. Workflow provenance is carried by a dynamic virtual device concept that we have introduced. A case study describes how to leverage an in-memory database to monitor and manage real-time sensor service provisioning.

I. Introduction

Sensor-equipped connected devices (Internet of Things or IoT) have been helping people sense the physical world [1] and yielding unprecedented services [2]. However, the *considerate* attribute of sensor service provisioning has not been significantly studied. In the human world, we think of people as considerate when for example they introduce themselves and their intentions without interrupting an on-going conversation [3]. The term *considerate* applied to a software system addresses attempts to reduce user disruptions during communication [4]. To facilitate human interactions, the system might coordinate, prioritize or time shift communications. It might also add other annotative statements as well to introduce and orient the information receiver. By considerate sensor service, we mean that sensors dynamically "collaborate" to provide considerate context-aware federated sensor data.

This paper reports our on-going efforts to develop a service oriented computing-empowered technology layer to provide "considerate" services over the Internet of Things. We have three primary technical contributions: (1) We have developed a novel approach to build social sensor networks to record and study historical interaction patterns among sensors. (2) We have designed and developed a sensor application server, a service-oriented middleware that supports considerate sensor service discovery and provisioning. (3) We have developed a concept of dynamic virtual device to embody workflow provenance management and analysis.

The remainder of the paper is organized as follows. In Section II, we describe a motivating example that will be used to discuss our work throughout the paper. In Sections III, IV, V, VI, VII, we successively present our sensor service modeling, discovery, application server, infrastructure, and scalability study. In Section VIII, we discuss related work and inSection IX, we draw conclusions.

II. Motivating Example

Our motivating example is a physical therapy scenario illustrated in Fig. 1. A patient John visits a physical therapist Bill at his office. Bill prescribes a therapy plan (a multi-step procedure called a *workflow*), which advises John to move his legs up to the horizontal level, for 20 times within 8 minutes. John is advised to repeat same exercise twice a day for 3 weeks prior to a follow-up visit. Due to John's medical condition, every time when he does the exercise, the process has to monitor his vital signs (i.e., heart rate, blood pressure, and body temperature), say every 2 minutes.

In this scenario, sensor data is used to measure John's medical condition during the exercise. John may attach a mobile phone to his leg as a motion sensor. When he moves his leg, the motion sensor will sense the activity and provide readings that can be used to study the frequency and quality of the movement. Based on John's medical profile, after 20 leg movements in the 8-minute time period, his body temperature and heart rate, and perhaps his blood pressure should move up a certain amount. At the same time, the room temperature and humidity in which John does the exercise are also factors that may have an impact on the measurement results.

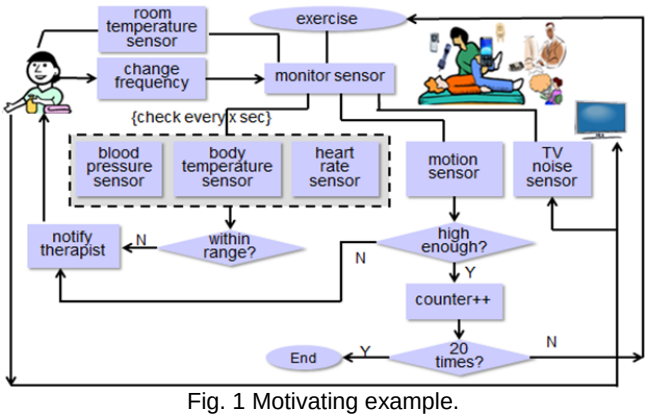


Fig. 1 Motivating example.

Based on observed sensor data, Bill may dynamically change his query to sensor data. For example, if John's heart rate goes up too fast, Bill may decide to conduct a closer monitoring, e.g., every 15 seconds instead of every 2 minutes. Such a change of request may lead to different sensor data gathering and analysis activities.

As shown in Fig. 1, a TV is present in the environment. Assume that John watches TV while he does the exercise. If John does not raise his leg high enough in the exercise, Bill will be notified and he will guide John. If the TV is too loud, its volume will be automatically turned down in a considerate manner to facilitate effective communication between Bill and John.

This example illustrates that, to provide a considerate service, multiple sensors may need to be identified in real time, and their data need to be dynamically federated to provide data analytics. Now that sensors can communicate with each other in sensor networks, a key motivation of our project is to determine how to leverage the latest technology advancement and establish an infrastructure over the sensor networks to enable and facilitate considerate service provisioning.

III. Sensor Service Modeling

Our study is fundamentally based on our innovative mapping between sensors and services. Each sensor measures its surrounding environment and provides to the outside world its sensor readings, which can be considered as a service. Hence, sensors can be viewed as service providers. Thus we are exploring how to apply services computing techniques to sensor services and their relationships.

In our previous work, we have introduced the concepts of virtual sensor and virtual device [5] to model sensor data federation. In this work, we extend our concepts to model workflow-oriented sensor services. A *virtual sensor* thus becomes a user-defined sensor service that reveals some information of a predefined environment. For example, a virtual sensor may be defined as a service to expose the average temperature of the room where John does exercises to help monitor his body condition. A virtual sensor may involve one or more physical sensors. A *virtual device* refers to a user-defined abstract container that comprises one or more (virtual) sensors. It defines a user's interest areas and it may represent a dynamic concept. For example, John may carry both a mobile phone and a medical watch, each containing a collection of sensors. In this sense, John can be viewed as a virtual device, hosting multiple sensors.

In the service level of our study, we only care about virtual sensors that possess business logic (established by the context). Each virtual sensor that provides a (business) sensor service to the outer world will thus become an atomic unit of our study. A typical multi-step procedure, called a *workflow*, may invoke multiple virtual sensor services. Fig. 1 is an example of such a workflow involving a collection of sensor services.

As shown in the motivating example, every time when John conducts the therapy exercise (a *workflow run*), the surrounding sensors will monitor and detect his body condition. Such data will help therapist Bill better understand John's reaction to the therapy and better personalize the treatment for John. Therefore, we associate each workflow with a virtual device in order to record workflow provenance (history) for later data analysis. The dependency between a workflow and sensor services is defined as follows.

Definition 1. A workflow $\mathcal{W}$ comprises a global schema $\mathcal{G}$ and a virtual device with a finite set of actions for each sensor service s_i of $\mathcal{W}$. An action of service s_i is an expression *Update :- Condition* where *Condition* is a query over local schema $\mathcal{L}$, and Update is a non-empty sequence of positive and negative relational literals over $\mathcal{L}$ such that each variable occurring in a negative literal also occurs in Condition.

Based on the association between a workflow and a virtual sensor, we model the relationship between a workflow and related sensor services. Leveraging social network theory and techniques, we will analyze usage history and behaviors of sensor services in a workflow context.

We model all IoT sensors as social entities. The rationale is that sensors may be intelligently used together by people to better sense the world. We start by modeling the "use" relationship between a workflow and a sensor service. The workflow in the motivating example Fig. 1 monitors a set of sensors: a heart rate sensor, a blood pressure sensor, a body temperature sensor, and two room temperature sensors. We establish a social tie between them. A workflow-sensor network is thus established based on their inclusive relationships. Fig. 2(a) illustrates a segment of the network: an edge exists between a workflow and a sensor if the sensor's readings are used in the workflow.

Such a network relationship can be formalized into a matrix Q that describes the involvement relationships between workflows and sensors:

$Q = [q_{ij}], 0 \le i \le m, 0 \le j \le n$, where:

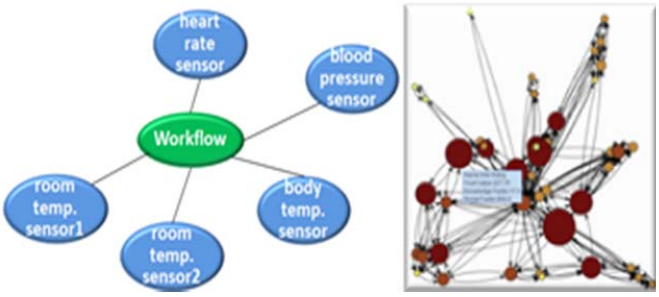
$q_{ij} = 1$ if workflow i retrieves sensor j; m workflows and n sensors are included.

As discussed earlier, all workflow runs are maintained in provenance. Therefore, such provenance data can be analyzed to extract the above relationships. Fig. 2(b) illustrates a part of a workflow-sensor network.

Based on the above relation Q, relation S can be retrieved:

$S = Q^T \bullet Q = [s_{ij}], 0 \le i, j \le n$, where:

s_{ij} = number of workflows where both sensors i and j are retrieved; s_{ii} = number of workflows where only sensor i is retrieved.



(a) (b)

Fig. 2 A "social" relation between Internet of Things.

Relation *S* represents a "social network" among sensors. Such a network of sensors forms a Social Internet of Things (SIoT). The semantic meanings of its comprising connections are: if two sensors are retrieved by the same workflow, there is a social tie between them. Note that rich context information may be carried by the edges in the network as labels. For example, an edge between two sensors may be labeled with the corresponding workflow run that retrieves sensor readings among them. By analyzing the details of a specific workflow run, we can understand when and under which circumstances the two sensors' readings were federated. The collaboration relationship maps to the association rules in social network analysis.

In addition, the social ties can be further differentiated into *tight* and *loose* collaboration relationships. Recall that our concept of *virtual sensor* implies that some formula is specified to federate readings from multiple homogeneous sensors, e.g., calculating an average room temperature from two sensors in a room. We call this relationship a tight collaboration tie, in the sense that their sensor readings are federated, and the original sensor readings are not shown in the final values.

In contrast, the relationships among multiple sensors in a *virtual device* are called loose collaboration relationship. Sensors in a virtual device may not be the same type of sensors. Even if they belong to the same type of sensor (e.g., the body temperature sensor and room temperature sensors in the motivating example), their readings are simply aggregated to show that there may be some correlation between them. For example, room temperature may have an impact on body temperature. Such loose collaboration relationship may be useful for users to conduct further data analysis.

After establishing the workflow-sensor networks, we calculate various metrics over them to comprehend the interaction patterns between sensors and workflows, as well as patterns among sensor usages. We adopt the tradition of calculation of *centrality and prestige* in social network analysis, including degree centrality (popularity), betweenness centrality, and clique. Through *degree centrality* analysis, highly used sensors can be identified based on the popularity of corresponding nodes. Through b*etweenness centrality* analysis, hinge sensors can be identified. Through *clique*, we can identify collaborative sensors, i.e., association rules among services.

If a sensor is used multiple times in multiple workflows, popularity can be considered as an annotation to be used for future purpose. For example, if a sensor is used heavily by many queries, then its readings may need to be pre-fetched, pre-processed, or redirected, and so on. Using the *k-path* betweenness algorithm, we can identify the key sensors, in the sense that they collaborate with other sensors in user queries (i.e., workflow runs).

Through these analyses, we can answer user queries regarding sensor query (usage) behaviors. For example, "how are different sensors used together in workflows?" And "in what types of workflows is a sensor usually used?" As another example on sensor-sensor relationship, we can ask "are there many sensors that collaborate with each other in workflows, and how?" And "what are the key sensors in these collaborations?" Such information will help our Sensor Service Discovery (SSD) service to help select proper sensors to provide considerate service provisioning. This will be discussed in the next section.

IV. Sensor Service Discovery

Since we are studying considerate service provisioning, one core requirement is to understand personalization. For a specific service oriented to a specific user, a considerate service implies that we need to discover related sensors that can contribute to understand the context of the user, as well as the specific service query.

Based on the workflow instance assigned by therapist Bill to patient John, we need to dynamically construct its associated virtual device to find physical sensors whose features satisfy the considerate requirements. In the motivating workflow, a motion sensor, a blood pressure sensor, abody temperature sensor, a heart rate sensor, and room temperature sensors at the user's location should be discovered. Along with this, users can also have various preferences, e.g., nearby and popular sensors will have higher priority.

Synchronization among sensors may also need to be considered. In the motivating example the temperature sensor in the room in which John does his exercise may be triggered to wake up to send readings if the motion sensor in the mobile phone on his leg has sensed that John has started an exercise. After the exercise is done, the temperature sensor does not have to send in readings. The temperature sensor does not have to send readings unless necessary – saving energy as another considerate service provisioning.

Dynamic sensor service discovery is an integral and key part of our system. We have designed a service discovery engine that takes into account user preferences before identifying the sensors to be involved in a specific workflow run. Each sensor has a sensor type and carries values for a set of QoS attribute dimensions. Users may configure their own preferences by selecting a subset of these QoS dimensions. Without loss of generality, we consider the following five QoS dimensions: location, availability/reliability, popu-

larity, accuracy, and usage.

Location: We care more about relative location than absolute GPS location. For example, if John does exercise in Building 23, room 120 - the sensors whose locations relative to the room within a threshold of distance are identified.

Availability/Reliability: Historical sensor data readings are examined to evaluate the availability and reliability of a sensor. Example measurements are: mean time between failures of the sensor, whether the sensor bandwidth exhibits some problem; or whether there is a need for real-time data or achieved data is sufficient.

Popularity: This dimension refers to the number of users who have used this sensor in the past, in other words, their workflows (virtual devices) invoke the sensor service.

Accuracy: The value of this dimension can be measured based on: whether the sensor readings are semantically reasonable; or whether the sensor readings are consistent to those from sensors close by.

Usage: This dimension considers power consumption. For example, a patient asked to do the exercise 20 times may use the sensor data more often than a patient asked to do the exercise 10 times. This implies that a motion sensor used in the former case would consume more power than that used in the latter case. Therefore, if a sensor is required for a longer time, we will recommend a low power consuming sensor.

Here we model a user preference space P for user u_i as $p_{u_i} = \{p_1, p_2, \ldots, p_k\}$, where $p_i \in Q$ and p_i denotes a dimen-dimension affecting u_i's decision. $P \subseteq$ Q.

Definition 2. Sensor Dominance: A sensor service s_i is said to dominate another sensor service s_j on Q iff $\forall q_i \in Q$, $s_i, q_i \geq s_j, q_i$ and $\exists q_t \in Q$, $s_i, q_t \geq s_j, q_t$.

Definition 3. Sensor Skyline: A sensor service s_i is a sky-skyline sensor iff there does not exist a sensor service $s_j \neq s_i$ dominating s_j.

Definition 4. User Preference-Aware Sensor Dominance (UPD): Given a user preference space P, a sensor service s_i is said to dominate another sensor service s_j on P iff $\forall p_j \in P$, $s_i, p_j \geq s_j, p_i$ and $\forall p_t \in P$, $s_i, p_t > s_j, q_t$.

Definition 5. User Preference-aware sensor Skyline (UPS): A sensor service s_i is a skyline sensor on P iff there does not exist a sensor service $s_j \neq s_i$ dominating s_i on P.

We thus decompose the process of sensor service discovery into the following two phases:

Local Optimization: Given a workflow $\mathbb{W}$, a user pref-preference space P, and a candidate sensor service class S_i for a task $t_i \in \mathbb{W}$, compute the UPD(S_i,P)|t_i.

Global Optimization: Given UPD(S_i,P) for each task $t_i \in \mathbb{W}$, compute the top-k Virtual Devices for workflow $\mathbb{W}$.

As shown in the example below, the local optimization phase aims to select the top-k sensor services based on their QoS attributes, e.g., availability, accuracy, and usage. Each QoS dimension is associated with an index. We apply a multi-index based algorithm to compute the UPD of indi-individual QoS dimensions. For a workflow w_i, the UPD will be calculated to partition sensors based on user preferences, such that sensors are sorted in a descending order of maximum value in that dimension.

Availability	Accuracy	Usage
S3(**0.90**,0.45,0.55)	S4(0.44,0.**98**,0.54)	S1(0.34,0.54,0.**88**)
S2(**0.83**,0.55,0.67)	S5(0.67,0.**76**,0.65)	S6(0.67,0.59,0.**78**)

The sensors in the UPD are then ranked based on the scores and the top-k sensors are identified. Thus we get the top-k available sensors for the workflow locally. The score of a sensor is calculated as:

$$Score(s_i) = \sum_{j=1}^{a} \frac{q_{j,max} - q_j}{q_{j,max} - q_{j,min}}$$

where $q_{j,max}$ and $q_{j,min}$ are the maximum and minimum values of the candidate sensor class on the dimension j and q_j is the value of the sensor i for dimension j.

For global optimization, we leverage "social" relationships among sensors (as described in Section III above) to select composite services. For each service cluster in the virtual device, we obtain top-k candidate services from the local optimization procedure. We then get all combinations of composite services. A social tie of each combination is calculated. The top one will be recommended to the user. The pseudo code is summarized as below.

Alg. 1: Global Optimization Algorithm (top-k):

```
1. Virtual_Device ← ∅;
2. Heap = Root Node comprising services with lowest score;
3. initialize(ParentTable);
4. while Heap ≠ ∅ do
5.   x = top(Heap);
6.   if ( ∄y ∈ Virtual_Device, dominated(x,y)) then
7.     insert(x, Virtual_Device)
8.   end if
9.   Child_Nodes = Expand (x);
10.  for ∀n ∈ Child_Nodes do
11.     Number_of_Parents (n)--;
12.     if Number_of_Parents (n) == 0 then
13.        insert(n,Heap);
14.     end if
15.  end for
16. end while
```

As illustrated, top-k services are sorted based on their scores. All combinations of composite services are enumerated using a lattice, based on a min-Heap strategy that extracts the composite services with minimum scores and compares the social coalition against other virtual devices. A composite service is added to the final list of virtual devices only if it is not dominated by any other composite services in the list.

The performance of the Global Optimization Algorithm is decided by two major factors:

1. Heap operations: It is affected by the heap size, which grows exponentially with the number of sensors. This behavior is observed because the size of the heap is bounded by the number of nodes on the middle level of the lattice. After every extraction the heap needs to be reorganized. With increasing number of members in the heap this operation becomes more costly.

2. Social tie comparisons: It grows exponentially with the number of sensors involved. The high computational complexity of Global Optimization Algorithm makes it impractical to compute the virtual devices with a large number of sensors. A bottom-up strategy may be adopted, using the same algorithm but iteratively. Instead of creating combinations of all sensor types at one time, this algorithm iteratively combines these services thus providing a faster and more scalable solution.

V. Sensor Application Server

Based on our sensor service modeling and discovery techniques, we have developed a sensor-oriented infrastructure. We name it Sensor Application Server, analogous to the software-oriented application servers such as WebLogic and WebSphere. This supports our two main goals: service collaboration provenance management, and sensor service discovery and composition.

As explained in Section III, we have introduced a virtual device concept to carry contextual information (sensor data provenance) of a specific workflow execution (workflow run). Following this idea, we differentiate three concepts: workflow template, workflow instance, and workflow run. A workflow template represents an abstract definition of stepwise activities; a workflow instance represents a personalized workflow template for a particular user; a workflow run represents an actual execution of a workflow instance.

As shown in Fig. 3, a workflow template is retrieved from a medical workflow repository representing a type of physical therapy procedure. Then Bill checks John's medical record, in order to configure the workflow specifically for John. Such a workflow instance is stored back in the database, as a customized workflow for John for a specific time period. Each time when John does the exercise, the corresponding workflow instance is triggered for execution as a workflow run. The provenance of each workflow run is also stored into the database for Bill to analyze in later diagnosis.

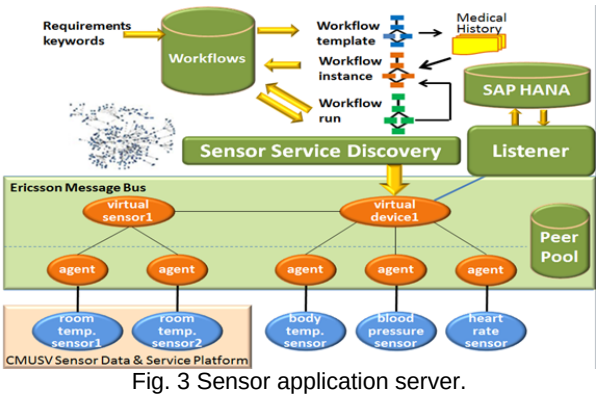


Fig. 3 Sensor application server.

As shown in Fig. 3, each workflow run will query the sensor service discovery (SSD) service to dynamically form a virtual device to support the execution of the workflow. Using the motivating example shown in Fig. 1, SSD locates two room temperatures in the room where John does his exercise. Therefore, Fig. 3 illustrates a formed virtual device comprising a virtual sensor representing the average room temperature based on two temperature sensors, a body temperature sensor, a heart rate sensor, and a blood pressure sensor. As shown in Fig. 3, a workflow run dynamically formed a virtual device comprising multiple sensors. In other words, these sensors collaborate with each other in the context of the workflow.

A. Message Bus

At Ericsson lab, we have developed a prototype of a Message Bus. In contrast to generic message bus [6], to enable sensor-oriented information changing through common message-based communication. As shown in Fig. 4, sensors are not isolated. Instead, they can "talk" to each other via message passing with the support of a centralized Messaging system. Each message notifies an event happening at the corresponding sensor. As the following example shows, a sensor may declare that a corner red light is turned on by publishing the following message:

```
publish("turn on light", null, "color=red&location=corner");
```

A publish/subscribe pattern is adopted in the Message Bus. Sensors can publish messages to the Message Hub; and sensors can subscribe to interested messages. The following example shows that a sensor is interested in receiving notifications when some corner red light is turned on.

```
subscribe("http://eus2.fuatara.com/callback",
  "turn on light",
  ["color=red", "location=corner", "eco-friendly=true"]);
```

As shown in Fig. 4, sensor peers thus form a loose coupling relationship. The Messaging Catalog acts as a filter, so that only allowed topics and metadata can be published and subscribed to devices. In addition, four forms of messaging are supported: unicast, multicast, broadcast, and publish-subscribe.

B. Message Bus-based Service Collaboration

As shown in Fig. 3, all physical sensors are mapped into an agent in the Message Bus as its representative. It is analogous to an Entity Bean in a (software) application server to its persistent object. It is an agent's responsibility to monitor the health of the corresponding physical sensor. In other words, sensors interact with each other through their agents in the Message Bus.

Inspired by the EJB concepts that differentiate between

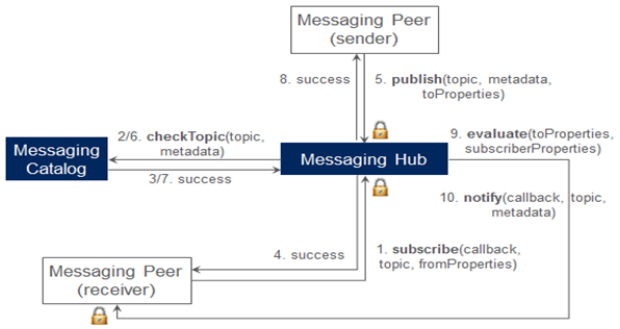


Fig. 4 Message Bus.

entity beans and session beans, we differentiate between sensor-oriented agents and virtual entities, i.e., virtual sensors and virtual devices. As shown in Fig. 3, in the Message Bus, virtual entities are modeled as agents as well. But in addition, they carry business logic.

Message Bus requires that the agents be deployed ahead of time. Therefore, for scalability, we have preconfigured and deployed a pool of anonymous peers (agents) when the Message Bus is deployed onto the Web server. At run time, peers will be extracted from the pool and dynamically assigned to act as various roles for workflow runs. Upon completion, the peers will be released back to the pool to be used by other workflow runs.

As described earlier, sensors communicate with each other via message passing. As shown in Fig. 3, our strategy is to establish a listener over the Message Bus. All sensor messages will be scanned and stored. As shown in Fig. 3, the Message Bus is treated as a standalone generic service. After a processing a message, the Messaging Hub forwards the message to our message listener as a JSON object. We have built a message bus listener as a separate service. A logging mechanism is available in the Message Hub. Every event in Message Hub will trigger a log message to be sent to the listener server.

By building a listener for message bus, the sensor application server knows what exactly happens in the system at runtime. In this way, we can use this information to determine which service/sensor needs to be used in real-time conditions, and how to properly compose services to make the system considerate based on real-time needs. By representing sensors as agents with methods, we can model a certain service workflow by using methods as primitives. Sensors (via their corresponding sensor agents) publish their readings onto the Message Bus. Workflow runs subscribe to specific sensor readings (through service discovery) and will receive the sensor readings. At the same time, workflow runs leverage analytical methods to monitoring the incoming streaming data at real time (will be discussed in the next section). If there is a change of plan (e.g., to change the frequency of heart rate monitoring), the workflow run will publish a request to the Message Bus.

We have designed a two-phase publish/subscribe pattern. When a workflow instance is established for a workflow run, it will construct a representative agent in the Message Bus. A two-way relationship is built between the workflow agent and the corresponding sensor agents. On one hand, the workflow run will subscribe to all identified sensor agents for their readings. On the other hand, the sensor agents will subscribe to the workflow agent to receive particular requests.

VI. Infrastructure Supporting Considerate Services

We have designed a service-oriented infrastructure supporting considerate service provisioning. As shown in Fig. 5, our infrastructure builds a layer on our Sensor Data & Service Platform (SDSP) and Message Bus. It comprises four horizontal layers and four vertical layers. For all sensors registered in our SDSP, the service modeling and publishing layer analyzes their interactions and establishes a sensor social network. The service composition and collaboration layer monitors the sensor interactions through a listener service, and handles dynamic service composition. The service visualization and analytics layer helps to visualize and analyze sensor relationships. The service provisioning layer interfaces with external sensors and deliver considerate services.

Vertically, the service discovery layer is in charge of dynamic sensor service discovery and integration. The considerate layer controls the quality of service provisioning focusing on the considerate property. The data architecture layer decides the data model and structure of service management. The governance layer monitors the health of the overall system.

One specific feature that we consider is the scalability. Our previous work adopted the traditional client-server model [5], where SDSP receives all sensor readings and forwards them to persistent storage (i.e., Amazon DynamoDB). Although straightforward, the SDSP server becomes the bottleneck. To support streaming sensor data and real-time data analytics, we decided to adopt the Message Bus in our infrastructure. As shown in Fig. 5, all registered sensors directly send notifications to the message bus.

To make the sensor network considerate, it is critical to seamlessly integrate human interaction. Without losing gen-

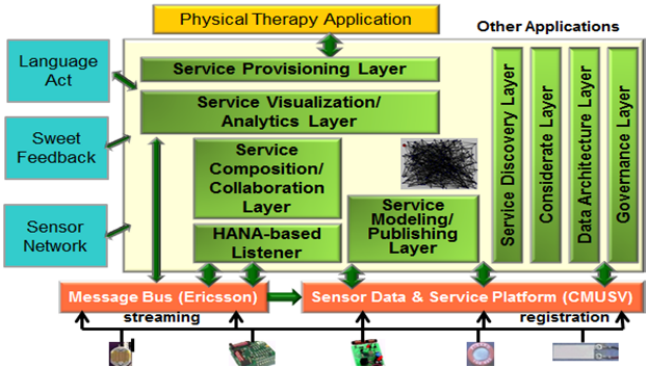


Fig. 5 Supporting considerate service provisioning.

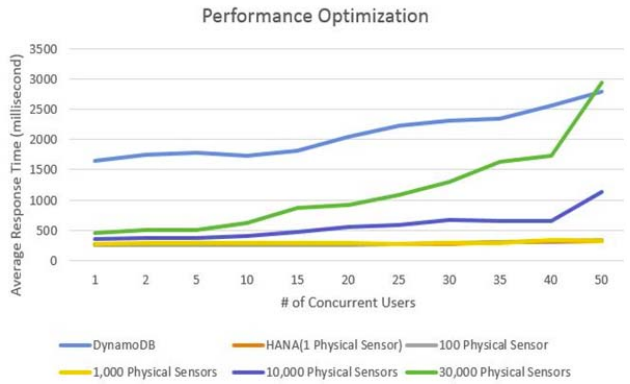


Fig. 6 Scalability study.

erality, we have incorporated language processing and language feedback to our social sensor network. John (or other treated subject) interacts with the social sensor networks through voice, and the social sensor networks provide voice feedback according to the subject's words, the prescription for this subject, and the data collected from this subject. Such a feature becomes extremely useful when the patient feels uncomfortable during a physical treatment, which indicates the necessity of the modification of the prescription.

When the exercise starts, the therapist may start a query, to ask to send to him the patient's average heart rate every 2 minutes. The service layer will then generate a workflow, caching the heart rate sensor's streaming data and send back average values every 2 minutes.

During the exercise (workflow run), the patient may feel uncomfortable. Bill may then change his query as to send the patient's heart rate every 15 seconds. Then the service will not cache heart rate sensor readings for later analysis. Instead, all readings will be sent to the therapist directly for him to better monitor the patient's body situation.

VII. Scalability Study

A. Sensor Service Analytics

The scalability and performance are the major concerns of our infrastructure. For analyzing sensor behaviors, all messages are stored. We have decided to adopt in-memory database to address the performance concern. Our preliminary study focuses on the feasibility of our infrastructure to support message-based considerate service discovery. Although our current environment is built on an SAP HANA database instance configured at Amazon Web Service (AWS), the techniques developed are generic enough to be applied to other environments.

We use SAP HANA as a scalable solution to the volumes of data. HANAis an in-memory data platform that can be deployed in the cloud. Unlike other database engines, SAP HANA provides column-based storage which allows us to easily monitor data. This helps provide real-time analytics. As explained in Section V, sensor communications are represented by topic-based messages. Each topic is represented by a <key, value> pair. The sensor messages can be stored in SAP HANA and can be used for real-time analytics. Since we have to store messages passed around, the data can be stored easily. We store raw sensor readings using the schema of <time stamp, device id, sensor type, sensor reading>, and also aggregated the readings in a 1-minute time window.

The platform is used for providing real-time analytics. We run the K-means algorithm (directly provided by HANA's Predictive Analysis Library) to monitor sensor health. It detects any anomaly in a sensor reading's interval, and flags outliers as sensor readings having an abnormal interval from the previous readings.

B. Experimental Study

We have conducted simulations to evaluate the scalability of our infrastructure. We simulated different scales of a sensor network, which comprising from 1 to 30,000 physical sensors. On each of the simulated sensor network, we simulated different numbers of concurrent users: from 1 to 50 users. As shown in Fig. 6, when the scale of a sensor network is moderate (comprising less than 1,000 physical sensors), our system scales well even when the number of concurrent users go up to 50. To better illustrate the experimental results, we keep only significant data in Fig. 6.

Note that the aforementioned experiments were conducted based on the SAP HANA database. In our earlier implementation, all sensor readings were stored in NoSQL DynamoDB database. We repeated the same set of experiments. As shown in Fig. 6, the scalability of the system relying on HANA significantly surpasses that on DynamoDB. This symptom is resulted from the specific pattern of frequent query over certain types of sensors, e.g., a user is working on a virtual sensor involving multiple temperature sensors. Thus, the specific column-based query used by HANA yields better results.

VIII. Related Work

Mohamed and Al-Jaroodi [7] surveyed the requirements of service-oriented middleware for wireless sensor network. Internet-scale Resource-Intensive Sensor Network Services (IrisNet) [8] offers a set of generic functionalities and APIs to allow user to query and process distributed collections of high-bit-rate sensors. On top of IrisNet, Chen et al. [9] proposed *X-Tree Programming*, a database-centric approach to programming over a large-scale of Internet-connected sensing devices. Microsoft SenseWeb [10] provides a Web 2.0 platform for users to upload and access sensor data streams from shared sensors on the Internet. SensorBase [11] built a centralized data storage and management platform that allows users to publish and share ("slog") sensor network data using a blog-like approach. Global Sensor Networks (GSN) [12] adopts a scalable P2P model in favor of integrating heterogeneous sensor network technologies. In this work, we explore how SOA can be leveraged to provide considerate sensor services.

Some researchers focus on building tools to support sen-

sor data manipulation. For example, the Desthino (Distributed Embedded Things Online) project identifies a practical set of software tools to help users collect and store sensor data from heterogeneous distributed sensors [13].

Sensor Observation Service (SOS) is a standard Web service specification, aiming to standardize the way of requesting, filtering, and retrieving sensors and sensor data to enhance sensor interoperability [14]. Some researchers explore how the Semantic Web can be integrated with a Sensor Web, such as SemSOS [15] and Semantic Sensor Web [16]. Some researchers, such as Liu et al. [17], study the scalability of sensor networks. A concept of virtual sensor is introduced in GSN [12] to abstract sensor data as temporal streams of relational data, and to represent derived views or a combination of sensor data from different sources. SenseBox [18] introduced an autonomous computing unit encapsulating environment and REST APIs. In our earlier work, we have developed an SOA-based Web 2.0 platform that allows users to federate heterogeneous sensor data sources [5]. In this project, we have extended our earlier work and build an infrastructure to support considerate sensor service discovery and composition.

Our previous work also leverages social network techniques to facilitate workflow reuse, by analyzing usage history and behaviors of software components [19]. In this project, we have extended our techniques to facilitate sensor federation.

IX. Conclusions

This paper has reported our efforts of exploring how to develop a service oriented computing-empowered infrastructure to provide “considerate” services. By mapping sensors to services and modeling sensor federation as social networks, we leverage past sensor usage history to provide more considerate services.

In our future work, we plan to refine our prototyping system and provide our middleware for the community to use. We also plan to conduct a performance study between using various databases as backend, including an in-memory database, a NoSQL database, and a relational database.

X. Acknowledgement

This project is partially sponsored by research gifts provided by Ericsson and SAP to Carnegie Mellon University.

VIII. References

[1]. R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a Global Coordinate System from Local Information on an ad hoc Sensor Network", in Proceedings of *2nd International Conference on Information Processing in Sensor Networks (IPSN)*, 2003, Palo Alto, CA, USA, pp. 333-348.

[2]. J. Yick, B. Mukherjee, and D. Ghosal, "Wireless Sensor Network Survey", *Computer Networks*, Aug. 22, 2008, 52(12): pp. 2292-2330.

[3]. R. Rajan, J. Hsiao, and T. Selker, ""Roger that!" - The Value of Adding Social Feedback in Audio-mediated Communications", in Proceedings of *Interact*, Sep., 2013, Cape Town, South Africa, pp.

[4]. T. Selker, "Understanding Considerate Systems - UCS (pronounced: You See Us)", in Proceedings of *The 2010 International Symposium on Collaborative Technologies and Systems*, May 17-21, 2010, Chicago, IL, pp. 1-12.

[5]. J. Zhang, B. Iannucci, M. Hennessy, K. Gopal, S. Xiao, S. Kumar, D. Pfeffer, B. Aljedia, Y. Ren, M. Griss, S. Rosenberg, and A. Rowe, "Sensor Data as a Service - A Federated Platform for Mobile Data-Centric Service Development and Sharing", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Jun. 26-Jul. 2, 2013, Santa Clara, CA, USA, pp. 446-453.

[6]. TIBCO, "TIBCO Rendezvous", accessed on: Oct. 27, 2013, Available from: http://www.tibco.com/multimedia/ds-rendezvous_tcm8-826.pdf.

[7]. N. Mohamed and J. Al-Jarood, "A Survey on Service-Oriented Middleware for Wireless Sensor Networks", *Service Oriented Computing and Applications*, 2011, 5(2): pp. 71-85.

[8]. P.B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "IrisNet: An Architecture for a World-Wide Sensor Web", *IEEE Pervasive Computing*, Oct.-Dec., 2003, 2(4): pp. 22-33.

[9]. S. Chen, P.B. Gibbons, and S. Nath, "Database-Centric Programming for Wide-area Sensor Systems", in Proceedings of *1st International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Jun. 30-Jul. 1, 2005, Marina del Rey, CA, USA, pp. 89-108.

[10]. W.I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao, "SenseWeb: An Infrastructure for Shared Sensing", *IEEE MultiMedia*, Oct.-Dec., 2007, 14(4): pp. 8-13.

[11]. K. Chang, N. Yau, M. Hansen, and D. Estrin, "SensorBase.org-A Centralized Repository to Slog Sensor Network Data", in Proceedings of *International Conference on Distributed Computing in Sensor Network (DCOSS)/Euro-American Workshop on Middleware for Sensor Networks (EAWMS)*, 2006, San Francisco, CA, USA, pp.

[12]. K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks", in Proceedings of *International Conference on Mobile Data Management*, May 7-11, 2007, Mannheim, Germany, pp. 198-205.

[13]. S. Santini and D. Rauch, "Minos: A Generic Tool for Sensor Data Acquisition and Storage", in Proceedings of *19th IEEE International Conference on Scientific and Statistical Database Management*, 2008, pp.

[14]. OGC, "Sensor Observation Service (SOS)", Open Geospatial Consortium, accessed on: 12/30/2012, Available from: http://www.opengeospatial.org/standards/sos.

[15]. C.A. Henson, J.K. Pschorr, A.P. Sheth, and K. Thirunarayan, "SemSOS: Semantic Sensor Observation Service", in Proceedings of *2009 International Symposium on Collaborative Technologies and Systems (CTS)*, May 18-22, 2009, Baltimore, MD, USA, pp. 44-53.

[16]. A. Sheth, C. Henson, and S. Sahoo, "Semantic Sensor Web", *IEEE Internet Computing*, Jul./Aug., 2008: pp. 78-83.

[17]. Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao, and X.-Y. Li, "Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs", in Proceedings of *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 10-15, 2011, pp. 873-881.

[18]. A. Bröring, A. Remke, and D. Lasnia, "SenseBox-A Generic Sensor Platform for the Web of Things", *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 2012, 104: pp. 186-196.

[19]. J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Jul. 4-9, 2011, Washington DC, USA, pp. 48-55.