Sensor Cloud: A Cloud of Virtual Sensors

Sanjay Madria, Vimal Kumar, and Rashmi Dalvi
Missouri University of Science & Technology

// The Missouri S&T (science and technology) sensor cloud enables different networks, spread in a huge geographical area, to connect together and be employed simultaneously by multiple users on demand. //

THE INDUSTRIAL WIRELESS sensor network market is expected to grow at a yearly rate of 43.1 percent and reach US$3.795 billion by the year 2017.1 Although the usage of wireless sensors continues to grow (see the sidebar for further reading), their full potential is bounded by the model of computation used to handle them. In traditional wireless sensor network (WSN) applications, a user needs to own a WSN, program the wireless sensors, deploy them and spend time and resources to maintain the network. The user is also restricted to one application per sensor network.

In this article, we describe a new paradigm of computation for wireless sensor networks, the sensor cloud, which decouples the network owner and the user and allows multiple WSNs to interoperate at the same time for a single or multiple applications that are transparent to users. We define a sensor cloud as a heterogeneous computing environment spread in a wide geographical area that brings together multiple WSNs consisting of different sensors. Each WSN can have a different owner. The sensor cloud then virtualizes the wireless sensors and provides sensing as a service to users. Because users buy sensing services on demand from the sensor cloud, use of large-scale sensor networks becomes affordable with ease of use.

A sensor cloud is composed of virtual sensors built on top of physical wireless sensors, which users automatically and dynamically can provision or deprovision on the basis of applications’ demands. This approach has a number of advantages. First, it enables better sensor management capability. The users can use and control their view of WSNs with standard functions for a variety of parameters such as region of interest, sampling frequency, latency, and security. Second, data captured by WSNs can be shared among multiple users, which reduces the overall cost of data collection for both the system and user. Because data reusability in WSNs is transparent to the sensor cloud users, redundant data capture is reduced, thus increasing efficiency. Third, the system is transparent regarding the types of sensors used. The user doesn’t need to worry about low-level details such as which types of motes and sensors are used and how to configure them; the sensor cloud automatically handles these details.

As a running example for the rest of this article, we’ll consider the following scenario. Traffic flow sensors are widely deployed in large numbers in places, including Washington, D.C., and Ohio. These sensors are mounted on traffic lights and provide real-time traffic flow data. Drivers can use this data to better plan their trips. In addition, if the traffic flow sensors are augmented with low-cost humidity and temperature sensors, they can provide a customized and local view of temperature and heat index data on demand. The National Weather Service, on the other hand, uses a single weather station to collect environmental data for a large area, which might not accurately represent an entire region.
virtual sensors to the user for providing sensing as a service. Gerd Kortuem and his colleagues discussed various components that constitute a sensor cloud system, its management, and the control flow of various components. Nayot Poolsappasit and his colleagues provided a layered architecture of a sensor cloud and outlined the security challenges of such a system. Sanem Kabadayi and her colleagues first described the abstraction of virtual sensors. Navdeep Kaur Kapoor and her colleagues designed the allocation algorithm to reduce the energy consumption, and scheduling algorithms are designed to reduce the response time of sensors in multiplexed scenarios. Global sensor networks (GSN) is a middleware designed to rapidly deploy heterogeneous wireless sensors. GSN also relies on virtual sensors; however, unlike approaches where virtual sensors are defined using classes, the virtual sensors in GSN are defined using XML. GSN, the closest approach to our sensor cloud, offers a ready-to-use system, which can integrate large numbers of wireless sensors networks. However, it can’t be classified as a sensor cloud because GSN’s purpose is to provide efficient and flexible deployment of multiple stand-alone WSNs.

We must note that our definition of a sensor cloud is different from others, and is an extension of the concept of the Internet of Things. The Internet of Things integrates the services provided by sensing devices with cloud computing over the Internet. Our sensor cloud, on the other hand, is a cloud of virtual sensors built on top of physical sensors, and it provides virtual sensors to the user for providing sensing as a service. The virtual sensors contain metadata about the physical sensors and the user currently holding that virtual sensor. Additionally the virtual sensor can have a data processing code, which can be used to process data in response to complex queries from the user. We have implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations.

One-to-Many Configurations
In this configuration, one physical sensor corresponds to many virtual sensors. Although individual users own the virtual image, the underlying physical sensor is shared among all the virtual sensors accessing it. The middleware computes the physical sensor’s sampling duration and frequency by taking into account all the users; it reevaluates the duration and frequency when new users join or existing users leave the system. Hence, this system is dynamic. Figure 1a shows this configuration.

Many-to-One Configurations
In this configuration, the geographical area is divided into regions and each region can have one or more physical sensors and sensor networks. When a user requires aggregated data of specific phenomena from a region, all underlying WSNs switch on with the respective

Virtual Sensors
A virtual sensor is an emulation of a physical sensor that obtains its data from underlying physical sensors. Virtual sensors provide a customized view to users using distribution and location transparency. In wireless sensors, the hardware is barely able to run multiple tasks at a time and the question of being able to run multiple VMs, such as in traditional cloud computing, is out of the question. To overcome this problem, in our sensor cloud we implement virtual sensors as an image in the software of the corresponding physical sensors. The virtual sensors contain metadata about the physical sensors and the user currently holding that virtual sensor. Additionally the virtual sensor can have a data processing code, which can be used to process data in response to complex queries from the user. We have implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations.

One-to-Many Configurations
In this configuration, one physical sensor corresponds to many virtual sensors. Although individual users own the virtual image, the underlying physical sensor is shared among all the virtual sensors accessing it. The middleware computes the physical sensor’s sampling duration and frequency by taking into account all the users; it reevaluates the duration and frequency when new users join or existing users leave the system. Hence, this system is dynamic. Figure 1a shows this configuration.

Many-to-One Configurations
In this configuration, the geographical area is divided into regions and each region can have one or more physical sensors and sensor networks. When a user requires aggregated data of specific phenomena from a region, all underlying WSNs switch on with the respective

Virtual Sensors
A virtual sensor is an emulation of a physical sensor that obtains its data from underlying physical sensors. Virtual sensors provide a customized view to users using distribution and location transparency. In wireless sensors, the hardware is barely able to run multiple tasks at a time and the question of being able to run multiple VMs, such as in traditional cloud computing, is out of the question. To overcome this problem, in our sensor cloud we implement virtual sensors as an image in the software of the corresponding physical sensors. The virtual sensors contain metadata about the physical sensors and the user currently holding that virtual sensor. Additionally the virtual sensor can have a data processing code, which can be used to process data in response to complex queries from the user. We have implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations.

One-to-Many Configurations
In this configuration, one physical sensor corresponds to many virtual sensors. Although individual users own the virtual image, the underlying physical sensor is shared among all the virtual sensors accessing it. The middleware computes the physical sensor’s sampling duration and frequency by taking into account all the users; it reevaluates the duration and frequency when new users join or existing users leave the system. Hence, this system is dynamic. Figure 1a shows this configuration.

Many-to-One Configurations
In this configuration, the geographical area is divided into regions and each region can have one or more physical sensors and sensor networks. When a user requires aggregated data of specific phenomena from a region, all underlying WSNs switch on with the respective

Virtual Sensors
A virtual sensor is an emulation of a physical sensor that obtains its data from underlying physical sensors. Virtual sensors provide a customized view to users using distribution and location transparency. In wireless sensors, the hardware is barely able to run multiple tasks at a time and the question of being able to run multiple VMs, such as in traditional cloud computing, is out of the question. To overcome this problem, in our sensor cloud we implement virtual sensors as an image in the software of the corresponding physical sensors. The virtual sensors contain metadata about the physical sensors and the user currently holding that virtual sensor. Additionally the virtual sensor can have a data processing code, which can be used to process data in response to complex queries from the user. We have implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations.

One-to-Many Configurations
In this configuration, one physical sensor corresponds to many virtual sensors. Although individual users own the virtual image, the underlying physical sensor is shared among all the virtual sensors accessing it. The middleware computes the physical sensor’s sampling duration and frequency by taking into account all the users; it reevaluates the duration and frequency when new users join or existing users leave the system. Hence, this system is dynamic. Figure 1a shows this configuration.

Many-to-One Configurations
In this configuration, the geographical area is divided into regions and each region can have one or more physical sensors and sensor networks. When a user requires aggregated data of specific phenomena from a region, all underlying WSNs switch on with the respective

Virtual Sensors
A virtual sensor is an emulation of a physical sensor that obtains its data from underlying physical sensors. Virtual sensors provide a customized view to users using distribution and location transparency. In wireless sensors, the hardware is barely able to run multiple tasks at a time and the question of being able to run multiple VMs, such as in traditional cloud computing, is out of the question. To overcome this problem, in our sensor cloud we implement virtual sensors as an image in the software of the corresponding physical sensors. The virtual sensors contain metadata about the physical sensors and the user currently holding that virtual sensor. Additionally the virtual sensor can have a data processing code, which can be used to process data in response to complex queries from the user. We have implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations.

One-to-Many Configurations
In this configuration, one physical sensor corresponds to many virtual sensors. Although individual users own the virtual image, the underlying physical sensor is shared among all the virtual sensors accessing it. The middleware computes the physical sensor’s sampling duration and frequency by taking into account all the users; it reevaluates the duration and frequency when new users join or existing users leave the system. Hence, this system is dynamic. Figure 1a shows this configuration.

Many-to-One Configurations
In this configuration, the geographical area is divided into regions and each region can have one or more physical sensors and sensor networks. When a user requires aggregated data of specific phenomena from a region, all underlying WSNs switch on with the respective

Virtual Sensors
A virtual sensor is an emulation of a physical sensor that obtains its data from underlying physical sensors. Virtual sensors provide a customized view to users using distribution and location transparency. In wireless sensors, the hardware is barely able to run multiple tasks at a time and the question of being able to run multiple VMs, such as in traditional cloud computing, is out of the question. To overcome this problem, in our sensor cloud we implement virtual sensors as an image in the software of the corresponding physical sensors. The virtual sensors contain metadata about the physical sensors and the user currently holding that virtual sensor. Additionally the virtual sensor can have a data processing code, which can be used to process data in response to complex queries from the user. We have implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations.

One-to-Many Configurations
In this configuration, one physical sensor corresponds to many virtual sensors. Although individual users own the virtual image, the underlying physical sensor is shared among all the virtual sensors accessing it. The middleware computes the physical sensor’s sampling duration and frequency by taking into account all the users; it reevaluates the duration and frequency when new users join or existing users leave the system. Hence, this system is dynamic. Figure 1a shows this configuration.

Many-to-One Configurations
In this configuration, the geographical area is divided into regions and each region can have one or more physical sensors and sensor networks. When a user requires aggregated data of specific phenomena from a region, all underlying WSNs switch on with the respective
phenomena enabled, and the user has access to the aggregated data from these WSNs. The sampling time interval at which all underlying sensors sense is equal to the sampling time interval requested by the user. This configuration can be used to provide fault tolerance if the underlying physical sensors fail. A virtual sensor communicates with a number of underlying physical sensors and it shows the aggregate view of the data to the user. When physical sensors fail, the WSN-facing layer of the sensor cloud captures the failure and the virtual sensor communicates it. A working sensor, which provides data within the quality-of-service (QoS) limits, can gather the required data. Thus, the virtual sensor adapts to a change in topology and the WSN-facing layer is transparent to the user.

**Many-to-Many Configurations**

This configuration is a combination of the one-to-many and many-to-one configurations. A physical sensor can correspond to many virtual sensors, and it can also be a part of a network that provides aggregate data for a single virtual sensor (see Figure 1a).

**Derived**

A derived configuration refers to a versatile configuration of virtual sensors derived from a combination of multiple physical sensors. This configuration can be seen as a generalization of the other three configurations, though, the difference lies in the types of physical sensors with which a virtual sensor communicates. While in the derived configuration, the virtual sensor communicates with multiple sensor types; in the other three configurations, the virtual sensor communicates with the same type of physical sensors.

Derived sensors can be used in two ways: first, to virtually sense complex phenomenon and second, to substitute for sensors that aren’t physically deployed. Two examples can help us to understand this better.

**Example 1.** Many different kinds of physical sensors can help us answer complex queries such as, “Are the overall environmental conditions safe in a wildlife habitat?” The virtual sensor can use readings of a number of environmental conditions from the physical sensors to compute a safety level value and answer the query.

**Example 2.** If we want to have a proximity sensor in a certain area where we don’t have one mounted on a physical wireless node, the virtual sensor could use data from light sensors and interpolate the readings and the variance in the light intensity to use as a proximity sensor. Figure 1b shows examples of derived sensors.

**Application**

In our running example, the virtual sensor can fall under any of the four configurations. A user might interact with one particular traffic flow sensor to assess traffic condition. Multiple users might also use the same sensor. A user might configure a virtual sensor to provide the average temperature of a region, which may involve multiple sensors. A user might also configure derived virtual sensors to calculate a heat index from temperature and humidity data.

**Missouri S&T Sensor Cloud**

The sensor cloud infrastructure at the Missouri University of Science and Technology (S&T) campus is divided into three prominent layers: client-centric, middleware, and...
sensor-centric (see Figure 2). The client-centric layer connects the users to the sensor cloud, whereas the middleware layer performs service negotiation, provisioning and maintenance of virtual sensors, and communication of data from the sensor-centric layer to the client-centric layer. The sensor-centric layer deals with the physical wireless sensors and their maintenance as well as routing of data and commands. From an implementation point of view, we can condense the layered architecture of Figure 2 to the block diagram of Figure 3. Each block in Figure 3 comprises one or more related functionalities in the layered architecture. The block diagram representation in Figure 3 is a more implementation-friendly illustration of these functionalities.

The Client-centric Layer
The client-centric layer acts as the gateway between the sensor cloud and the user. It’s a collection of components that facilitate and manage the interactions between the user and the core of the sensor cloud—that is, the virtual sensors. The client-centric layer comprises the user interface, session management, membership management, and the user repository components.

The Missouri S&T sensor cloud user interface lets users specify parameters such as regions of interest, sensing phenomena, sampling frequency, sensing duration, and mode (secure or not secure). The Web application parses the request and the parameters and communicates them to the back-end application server.

The session management component handles the secure creation, management, and termination of sessions between the middleware and the user. The membership management component of the client-centric layer takes care of the authorization of users and provision of access to the services for which they are authorized. The user repository component stores detailed user information in the system such as account credentials, payment history, billing information, and so on. It also keeps track of data sent by WSNs and accessed by the end users.

In our running example, the client-centric layer will expose a GUI showing the locations of the available sensors to the user. The user can create virtual sensors based on regions. The information about the selected region, sampling frequency duration, and QoS agreement would be sent to the middleware.

The Middleware Layer
The middleware layer acts as the

---

**FIGURE 2.** A layered sensor cloud architecture from the Missouri University of Science & Technology sensor cloud. It’s divided into three prominent layers: client-centric, middleware, and sensor-centric.
intermediary between the client-centric and sensor-centric layers and connects the client requests with the data collected from the sensors. The middleware performs a number of functions such as provision management, image life-cycle management, and billing management. Provision management facilitates the service negotiation between the user and the sensor cloud and provisions virtual sensors for each incoming request. This component of the middleware resides in the Web application server block of Figure 3. It receives requests from UIs of multiple users with their parameters and modes. If a request can be fulfilled, control is passed onto the virtual sensor server block which triggers the WSNs associated with the selected region and specified parameters via the back-end application server. Once an agreement between the user’s requirement and the sensor cloud’s capabilities has been reached, the virtual sensors are created.

The image life-cycle management component is implemented in the virtual sensor server block of Figure 3. The virtual sensor server receives requests for instances from the provision management component and takes care of creating instances for the virtual sensors provisioned for the users. After creating the instances, it communicates with the back-end application server to request a corresponding service instance. Each service instance of the back-end application is registered in the remote method invocation (RMI) registry before any request for that region can be satisfied. As shown in the block diagram in Figure 3, with respect to the user’s selection of region, the associated service instance of the back-end applications and their registered names are identified by querying the database. Later, these RMI-registered names are looked up from the registry to connect a particular user’s virtual sensors to the respective service instance. The virtual sensor server mediates the communication between the Web application and the back-end application and ultimately displays a different set of sensor data to each end user connected to the sensor cloud. The billing management component of the sensor cloud is responsible for keeping track of each user session with respect to the types of sensors used, number of sensor used, and mode of sensor usage to generate the invoice based on previous service agreements.
In our example, the middleware layer receives information about the user session from the user-centric layer and creates the virtual sensor. The virtual sensor configuration is decided on the basis of the type of data the user wants, the user’s region of interest, and the agreed upon QoS. If multiple users request information from the same sensors (for example, traffic information from the same location), the middleware consolidates the requests.

The Sensor-centric Layer
The sensor-centric layer directly communicates with the physical sensors using the WSN registration, WSN maintenance, and data collection components. When network owners want to provide service through the sensor cloud interface, they need to register their WSNs. The sensor clouds verify the physical sensors and their capabilities, which should also provide location information. At this point in time, we aren’t considering mobile sensors, therefore predeployed location information in the form of longitude, latitude, region IDs, and cluster IDs should suffice. The information collected in the WSN registration phase is used in cataloging the information about physical and later virtual sensors.

Once a WSN is registered, the network owner is in charge of keeping the physical sensors in good health. The registration binds the WSN owner and the sensor cloud in a trust relationship where the WSN owner is expected to provide accurate, untampered sensor readings, and the sensor cloud is expected to correctly provide compensation for the received sensor readings. The trust between the two parties can be enhanced by using secure and trusted data collection and aggregation techniques on the WSN side and by performing data anonymization and doing computations on encrypted data on the cloud side. (A more detailed discussion of the security, privacy, and trust issues in sensor cloud can be found elsewhere.3)

The WSN maintenance component provides interoperability of the heterogeneous mote platforms, periodically checks the health of each mote in the sensor cloud, provides synchronization between sensors, and collects metadata information about the motes and the networks. To handle noninteroperability issues, we installed a GumStix computer on module (www.gumstix.com) at the junction of two or more incompatible WSNs. The GumStix is hooked up with a number of different motes at different ports. The GumStix collects data from one port and transmits it to other ports as needed.

To check the health of the motes and collect metadata information, the WSN maintenance component periodically pings each network. The motes in the network reply with information such as their battery level, the mote to which they were last connected, their location, their region, and so on. Although we haven’t yet tackled the issue of synchronization between the networks and sensors, in the future, global time information can be piggybacked on the ping packets. This would provide time synchronization between the networks and the sensors.

To provide a finer-grained synchronization, we can also use the powerful GumStix nodes. The data collection component of the sensor-centric layer connects the system directly to the wireless sensor networks. Each WSN is connected to the data collection component through a base station. The data collection component, which resides in the back-end application server, runs one service instance for each base station. The service instance opens two dedicated ports, one to communicate with the base station and one to communicate with the associated virtual sensor on the virtual sensor server.

The sensor-centric layer in our example receives a query packet from the middleware and replies with the requested data. This layer handles the routing protocol, fault tolerance, and other network-related issues.

Software Design
The sensor cloud is a multitiered client-server software architecture with each layer logically separated from the other. The sensor-centric layer is the data tier. The layer consists of physical wireless sensors that generate real-time data. The middleware is the application or the logic tier, which controls the data collection. The client-centric layer represents the presentation tier.

From a software developer’s point of view, there are two different facets to developing a sensor cloud: the system side and the sensor side. The system side consists of the client-centric and middleware layer (the presentation and application tiers) and is basically used to manage physical resources. The end user’s view will vary depending on the application. The heart of any sensor cloud application, though, is going to be the middleware layer. The middleware layer is expected to be flexible enough to handle issues when physical and virtual sensors are scaled up and down. It’s also expected to aggregate the user requirements and redirect data according to these requirements from the sensors to the users, in addition to handling its regular tasks.
The second facet of a sensor cloud is the physical sensors side, which consists of the sensor-centric layer. WSNs are distributed networks, where a number of physically separate entities work together toward a goal (in this case, generating data according to the user’s requirements). WSNs face the same general issues as distributed systems such as synchronization, fault tolerance, and security. While developing for WSNs, developers need to keep these issues in mind, in addition to the wireless sensor-specific constraints such as low bandwidth, low processing power, and finite energy sources.

**QoS in Sensor Cloud**

A sensor cloud manages QoS at two levels: at the sensor-centric layer and at the virtual sensors. The sensor-centric layer handles network-related issues such as responding to node failures, network partitioning, and packet losses. The virtual sensor layer then works on top of the sensor-centric layer’s services to manage QoS parameters such as reliability, data accuracy, and coverage on top of the network layer. As we explained earlier, the many-to-one virtual sensor configuration can be used to provide data reliability via aggregation. The accuracy of data in such cases depends on the spatial and temporal correlation between the nearby sensors. The virtual sensor layer makes use of the correlations to retrieve data within the QoS limits from nearby sensors. The virtual sensor layer can also switch between configurations, for example, from one-to-many to many-to-many to provide data to multiple users within the specified QoS limits.

**Implementation**

We used the Linux platform for multiple back-end servers, which cater to multiple regions. We programmed the Web application and back-end server application in Java. We used RMI along with socket communication for communication among the various back-end servers. We used TelosB motes as wireless sensors, each equipped with a humidity, temperature, light intensity, and infrared sensor. We programmed the sensors using TinyOS 2.2 (http://tinyos.stanford.edu/tinyos-wiki/index.php/Installing_TinyOS).

**Data Streaming for a Multiuser Environment**

The virtual sensor model can effectively support a multiuser environment. A single wireless sensor provides data for multiple users, where the users can request data at varying frequencies and of different phenomena. When the Web application server in Figure 3 receives user requests, they are transferred to the virtual sensor server. The virtual sensor server performs the mapping between the user’s virtual and physical sensors. If multiple virtual sensors correspond to one physical sensor, the virtual sensor server combines the request by combining the sampling duration, sampling frequency, and sensing phenomena. The combined request is then forwarded to the appropriate service instance of the back-end server. The service instance communicates with the WSN and collects the data. Data from each WSN is sampled at the minimum frequency of all requests. This data is time stamped with the

---

**FIGURE 4.** Virtual sensor examples: a hierarchy of a user’s region of interest and (b) hierarchy data object tables for virtual sensors.
local time and stored in the database and displayed to the user at the requested frequency. Users can also select data from multiple base stations at the same time, either by selecting locations that includes multiple WSNs or by selecting multiple locations. The data will then be aggregated periodically at the requested frequency on the basis of the selected locations’ hierarchy.

Virtual Sensor Implementation
In this section, we describe the usage of many-to-one virtual sensors; however, one-to-many, many-to-many, and derived sensors are also implemented in a similar fashion. The geographical area that the sensor cloud covers is divided into regions that are arranged hierarchically. Aggregated data from various networks in a region are based on the user’s selection of the region and the selected region’s hierarchy. The scheme can be understood well by visualizing a network of WSNs (see Figure 4a). The topology in Figure 4a shows the hierarchy of WSNs, where each intermediate node and the root node is a network in itself and can have any number of children. In this example, we assume that an end user needs data from region 1, which is a virtual node. Similarly, all intermediate nodes (2 and 3) and the root node in the hierarchy are virtual nodes. On the other hand, all leaf-level nodes (4, 5, 6, and 7) represent physical WSNs. Thus, we can call this topology a network of VSNs (virtual sensor networks) and WSNs. The hierarchy information stored in data table (see Figure 4b) is static and is required while aggregating the information at intermediate and root-level nodes. On the other hand, a hierarchy data object is created for each user request. Once a many-to-one mapping of the virtual sensor is obtained, the virtual sensor server sends a request to the back-end application server, which then forwards the request to the concerned WSNs. Once all WSNs providing data to selected VSNs in the hierarchy are switched on, the back-end application server starts relaying data, which is parsed according to the VSN hierarchy.

Sensor clouds aim to take the burden of deploying and managing the network away from the user by acting as a mediator between the user and the sensor networks and providing sensing as a service. In addition to the working model we have presented in this article, we plan our future work to include working toward a complete plug-and-play model of application deployment and energy efficient scheduling over multiple applications over wireless sensors.

Reference

SANJAY MADRIA is a full professor in the department of computer science at the Missouri University of Science and Technology. His research interests include mobile data management, cyber security, and trust management in sensor networks and cloud computing. Madria received a PhD in computer science from the Indian Institute of Technology, Delhi. He’s a senior member of IEEE. Contact him at madrias@mst.edu.

VIMAL KUMAR received a PhD in computer science from the Missouri University of Science and Technology. His research interests include the broad areas of sensor clouds, wireless sensor networks, and cloud computing, particularly in security. Kumar received a BS in computer science and engineering from Indraprastha University. He’s a student member of IEEE. Contact him at vkq3f@mail.mst.edu.

RASHMI DALVI is a master’s student at the Missouri University of Science and Technology. Her research interests include sensor clouds and scheduling and allocation of sensors in sensor clouds. Dalvi received a post-graduate diploma in advanced computing from Yashwantrao Chavan Pratishthan. Contact her at rgd8t6@mst.edu.

ABOUT THE AUTHORS

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.