SenseWeb: An Infrastructure for Shared Sensing

Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao
Microsoft Research

Peer-produced systems can achieve what might be infeasible for stand-alone systems developed by a single entity. Take Wikipedia, for example. Each contributor to the site provides a small piece of information. Coordinated effectively, the sum of the compositions yields something of far greater value than all the compositions individually.

This kind of approach is at work in systems produced by labs and businesses such as SETI@home, YouTube, and Linux. In the area of sensor networks, SenseWeb (http://research.microsoft.com/nec/senseweb/) enables peer production of sensing applications, producing new kinds of media and applications over existing data networks. Contributors deploy their own sensors or sensors network. This might be designed for their own dedicated application, as in a security surveillance camera network, parking occupancy counter, weather monitoring home system, or heart rate monitor for runners. However, if these sensors are placed into and shared in a single development system, many more applications can be built.

When people share weather sensors from their homes on the Weather Underground forecasting Web site (http://www.wunderground.com/), for instance, it results in large-scale weather forecasts (for more information about this site, see also the *IEEE Computer Graphics and Applications* article at http://doi.ieeecomputersociety.org/10.1109/MCG.2007.124). Information from hikers' GPS-enabled body-worn sensors shared on MotionBased (http://www.motionbased.com/) and SlamXR (http://www.msslam.com/slamxr/slamxr.htm) helps others choose appropriate travel routes for their fitness and endurance levels. Essentially, in these system, the system sensor owners upload their data which is then made available through an application-specific GUI. While these examples reveal some of the many benefits that shared sensing offers, that's just the beginning.

Imagine, however, the potential if you could build new applications that leverage data from sensors already available in the deployed systems. For instance, if a retailer with several outlets at shopping malls across a nation could access video streams from security cameras at those malls, the retailer could build custom business applications for understanding customer behavior (such as the video analysis business tools from IBM's Smart Surveillance System[1]).

Imagine, then, if you could configure data collection for the specific needs of a new sensing application. Say, for example, a rainstorm hits and a cab dispatch back-end automatically requests from commuters with cell-phone cameras images of flooded road segments. Geology research labs could deploy soil sensors near their local university, and other researchers could task those sensors to collect data for new experiments.

SenseWeb's goal is to enable these kinds of capabilities. Using SenseWeb, applications can initiate and access sensor data streams from shared sensors across the entire Internet. The SenseWeb infrastructure helps ensure optimal sensor selection for each application and efficient sharing of sensor streams among multiple applications.

Editor's Note

Web 2.0 is an emerging paradigm for applications and user interactions. In this article, Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao from Microsoft Research discuss the development of SenseWeb, a peer-produced sensor network environment, used for everyday life decisions.

—*William I. Grosky and Utz Westermann*

System architecture

We designed the SenseWeb architecture to let multiple concurrent applications share sensing resources contributed by several entities in a flexible but uniform manner (see Figure 1). The architecture's key components are the coordinator; sensors, sensor gateways, and mobile proxy; data transformers; and applications.

Coordinator

The coordinator is the central point of access into the system for all applications and sensor contributors. The functions of the coordinator are internally divided between two components: the tasking module and senseDB (streaming sensor database).

The tasking module accepts applications' sensing requirements and tries to satisfy these from available sensing resources. To do this, it takes into account the capabilities, sharing willingness, and other characteristics of the available sensors.

The senseDB leverages the overlap among multiple application needs. When multiple applications need data from overlapping space–time windows, senseDB attempts to minimize the load on the sensors or the respective sensor gateways by combining the requests for common data and using a cache for recently accessed data. For instance, if data collected in response to an application's query is cached, other queries with partially overlapping needs might be served partly from the cache, and the number of sensors accessed might be reduced. Data extracted from the cache and sensors is streamed to appropriate applications with requested aggregation. SenseDB is also responsible for indexing the sensor characteristics and other shared resources in the system to enable applications to discover what is available for their use.

Sensors, sensor gateways, and mobile proxy

Sensing resources form the foundation of the entire system. Sensors can measure various physical variables and might be static or mobile, and carried by humans, on vehicles, or in robots.

Because contributors may build sensors using many different platforms that vary in processing power, energy, and bandwidth capabilities, the sensors might have different interfaces to access them. Low-powered wireless sensor nodes can use ZigBee radios to communicate while higher power and higher bandwidth sensors might need FireWire or similar interfaces. The sensors might or might not be connected at all times.

To hide much of this complexity, sensors are connected to a sensor gateway that provides a uniform interface to all components above it. The gateway implements sensor-specific methods to communicate with the sensor. However, other components of SenseWeb access the gateway to obtain sensor data streams, submit data collection demands, or access sensor characteristics through a standardized Web service API.

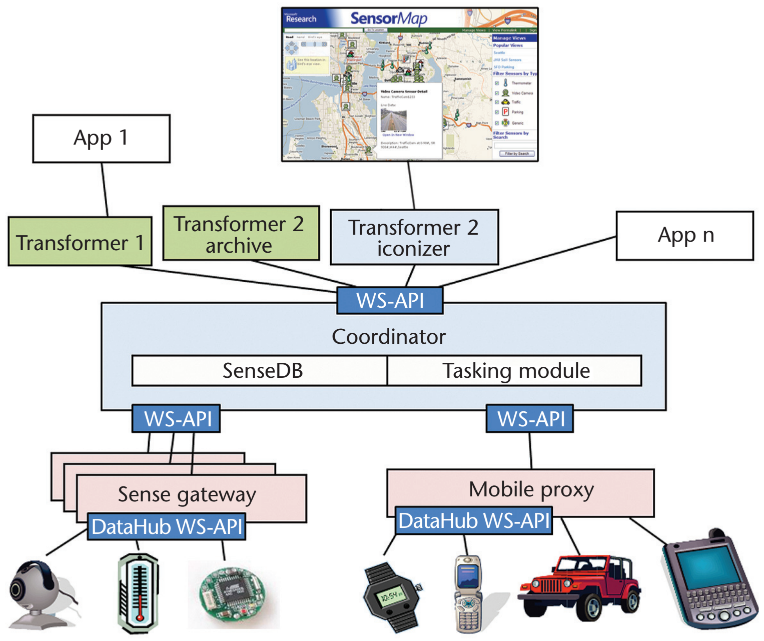

Figure 1. SenseWeb's open system architecture for flexible sharing. WS = Web service.

Each sensor contributor might maintain his or her own gateway. The gateway might also implement sharing policies defined by the contributor. For instance, the gateway might maintain all raw data in its local database—possibly for local applications the sensor owner runs — but only make certain nonprivacy-sensitive parts of the data or data at lower sampling rates available to the rest of SenseWeb.

We implemented a shared gateway in the SenseWeb prototype that could be used by sensor contributors who don't want to maintain their own gateway. This shared gateway, called DataHub, supports communications with sensors through a Web service API; drivers for common types of sensors, including wireless motes and network cameras, are provided to sensor contributors.

A mobile proxy is a special gateway built for mobile sensors. The mobile sensors form a highly volatile swarm of sensing devices that can potentially provide coverage where no static devices are available. However, given an application's region and time window of sensing interest, it's difficult for applications to track which mobile devices were or will be present in that required space–time window and which, if any, will be able or willing to provide samples.

The mobile proxy makes the mobility of sensing devices transparent to the applications. It provides location-based access to sensor readings. Applications simply express their sensing needs and the mobile proxy returns data from any devices that can satisfy those needs. Kansal and

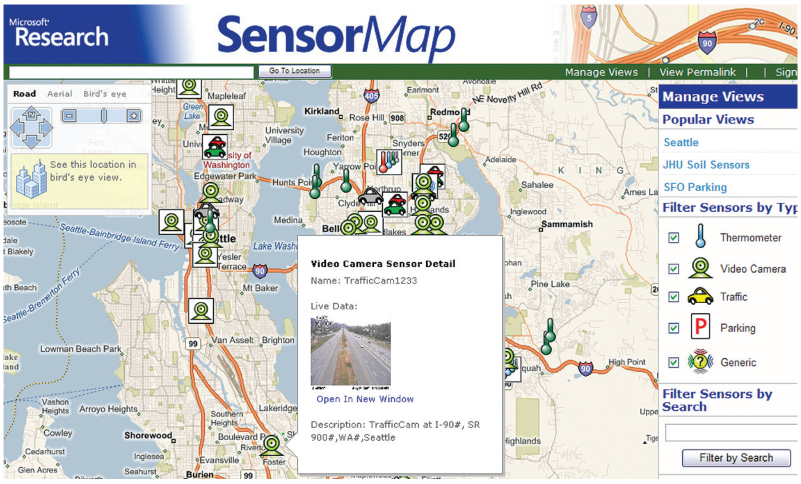

Figure 2. The SensorMap application user interface.

Zhao discussed the specialized methods to help design the mobile proxy.[2]

Data transformers

A transformer converts data semantics through processing. For example, a transformer might extract the people count from a video stream. Data transformers can also convert units, fuse data, and provide data visualization services. Moreover, application developers can extend SenseWeb's processing functionality by writing new transformers on top of the coordinator's primitive access methods. Domain experts might use various transformers for different sensor data using suitable domain-specific algorithms.

Transformers are indexed at the coordinator and applications might discover and use them as needed. Transformers provide an interface similar to that of a sensor gateway, but serve processed data. One example of a transformer is the iconizer implemented in our prototype: It converts raw sensor readings into an icon that represents sensor type in its shape and sensor value in its color. Graphical applications might use output of this transformer instead of raw sensor values.

Applications

Applications are all users of sensor data. These might be interactive applications where human users specify their data needs manually (such as user queries for average hiker heart rate over the last season on a particular trail). Or they might be automated applications in back-end enterprise systems that access sensor streams for business processing—such as an inventory management application that accesses shopper volume from parking counters or customer behaviors from video streams—and correlates them with sales records.

SensorMap is an example of an interactive application (see Figure 2) that demonstrates access to several sensor data types through a geocentric interface deployed as part of our prototype (http://atom.research.microsoft.com/sensormap/). SensorMap is a mash-up application that combines sensor streams obtained using SenseWeb with maps from Virtual Earth (http://msdn2.microsoft.com/en-us/virtualearth/default.aspx). SensorMap lets sensor contributors add their sensors to it through a standardized Web service interface. This lets application developers use SensorMap in visualizing their own sensor deployment.

As another example, in a prototype deployment, we installed sensors at a small number of homes near our lab. A graphical application queried these sensors for their temperature readings and produced the temperature contour map shown in Figure 3.

Other SenseWeb applications

Several other applications are being prototyped using the SenseWeb infrastructure. Researchers at Vanderbilt University are building a fine-grained pollution monitoring application using car-mounted devices that will enable visualizing real-time and historic pollution distribution within a city (http://www.isis.vanderbilt.edu/maqumon).

A team at The Ohio State University is developing applications to provide human location, network health, and home plant soil moisture data through SenseWeb (http://69.211.157.230/kansei/).

A collaborative effort at National Tsing Hua University and Feng Chia University is using SenseWeb to deploy a debris flow monitoring and warning application (http://www.ccrc.nthu.edu.tw/projects/debrisflow/debrisflow.html).

A national scale weather study system deployed by a team at the Nanyang Technological University is using SenseWeb to manage its sensor streams and visualizations at multiple user sites (http://nwsp.ntu.edu.sg/sensormap/).

Researchers at the University of Virginia are prototyping a system that would use SenseWeb to share data from urban shopping areas and assisted living facilities. SenseWeb has also been used to

- provide RFID-tag-based data for inferring indoor events at the University of Washington,
- study the Great Barrier Reef at the University of Melbourne,

- develop city-scale sensing infrastructures at Harvard, and
- collect biological and physiological data at the University of Illinois.

SenseWeb architecture enables not only shared heterogeneous sensors, but provides for the reuse of several resources required for sensor network deployment. Developers can reuse deployed sensors, spatiotemporal indexing and caching, data collection methods, and relevant transformers already present in the system so that they can focus their efforts on their specific applications. They might extend the system to meet new needs and in turn share results from their development efforts as additional transformers.

SenseWeb helps coordinate the multiple entities involved and optimizes the operation through effective reuse of data collection and processing tasks as well as the selection of appropriate sensors for various tasks. It simplifies the collection of complex data from heterogeneous sensors, provides abstractions that hide unreliable device connectivity and mobility of sensors from applications, and standardizes access to heterogeneous platforms.

Challenges

The peer production paradigm has several advantages. It enables large spatiotemporal coverage—not easy to achieve with a dedicated system because of cost or jurisdiction issues. Resources can be shared opportunistically or on demand; hence the costs can be amortized over a large number of applications. Further, a shared infrastructure enables a community effect, which makes the development of new sensing applications feasible. For instance, an individual with a mobile phone camera can take pictures of damaged sidewalks that might not yield significant data over an interesting space and time window. Now, if multiple mobile phone owners share their data, the aggregation might become significant enough to plan running routes and to repair facilities. However, when the resources are shared across multiple concurrent applications, several new challenges arise.

Heterogeneity

Unlike the sensors comprising an application-specific system, components of a shared sensor network might be highly heterogeneous along several dimensions: how data are produced, consumed, and understood. The production dimension has several types of heterogeneities, including types of sensors (such as wireless embedded devices, mobile phones, network cameras, pollution sensors, and even RSS feeds); their connectivity to the Internet (constant, intermittent, or affected by a firewall); their sharing willingness or privacy sensitivity; and the resource capabilities for processing data locally. Further, some sensors might need human assistance in collecting samples, such as a phone camera, and that adds a new set of issues to the design.

Consumers of the data are heterogeneous in terms of their data needs: some might need only a current snapshot while others might need streams extending over the past archived data and even future data. Their needs in terms of data quality, spatial resolution, and sampling rates vary. Further, different applications might need disparate processing or filtering of data. The understanding of the data depends on a wide variety of semantic information about it.

From an infrastructure perspective, the type of data might range among scalar, vector, image, video, or other complex structures that have varying storage, latency, and bandwidth needs. How the data might be indexed also depends on the type of data. For instance, while scalar data might be indexed by value, multimedia data needs more sophisticated indexing methods. From an application's perspective, the interpretation of data in terms of what it measures becomes crucial since even data with a common unit—say, temperature—sensors might not be measuring the same metric (for example, one sensor might be measuring the ambient temper-

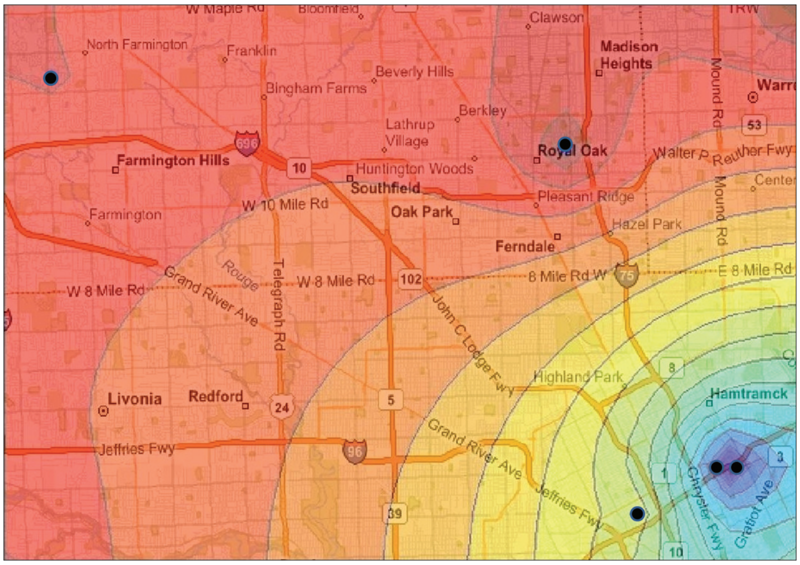

Figure 3. Temperature data retrieved by an application using sensors contributed on SenseWeb. The black dots show the sensor locations.

ature and another might be measuring point temperature).

SenseWeb tries to address problems because of heterogeneity in various ways. The differences in network connectivity are hidden from applications using standardized Web service interfaces for sensor gateways. Efficient communication techniques suited to the power and bandwidth constraints of the sensor devices can be implemented at the gateways. SenseWeb addresses the heterogeneity in sensor quality and willingness to share by providing additional metadata in sensor descriptions and learning sensor characteristics at runtime. Contributors can specify sensor precision, the maximum sampling rate they are willing to contribute, and the specific applications or groups of users with whom they are willing to share resources.

Applications can also specify their quality requirements and corresponding tolerances. The coordinator learns sensor characteristics such as connectivity disruptions and network bandwidth at runtime. It then uses both specified characteristics in combination with the learned properties to optimize the selection of sensors serving different applications.

Scalability

A shared system becomes more useful as the participant number grows, creating the community effect. This introduces significant challenges for scalability. As the number of applications grows, the demand for resources increases. Requiring all the contributing sensors to share their raw data streams is not scalable in a large system. In stand-alone systems, developers might improve scalability by reducing the amount of raw data transmitted from sensors through application-specific event detection or compression at the sensors. Doing the same in a shared system with multiple applications is challenging. Also, more than one application might use data from the same sensors in a shared system. To keep the resource usage scalable and to avoid unnecessarily denying access to more applications, it's essential for the system to reuse common data and sensing resources among overlapping application needs. The scalability challenges are further compounded by the need for maintaining extensibility to unknown applications, new types of sensors, and domain-specific data processing, aggregation, and visualization.

SenseWeb addresses scalability in two ways. First, data collection from sensors is minimized whenever possible by reusing the common data accessed by multiple applications and by computing approximate answers on a carefully chosen subset of sensors, instead of on all the sensors, within the query region.

Second, all sensors are not required to share all their available data. Data is collected in response to application needs only. Sensors are dynamically tasked to carry out sensing or event detection tasks according to the query workload. This not only avoids the need for all sensors to stream their raw data at the highest sampling rates, but also lets actuated sensors, such as pan-tilt-zoom cameras, sense in their most relevant pose or configuration. Another advantage of this approach is that in cases when only a limited amount of data can be obtained from a sensor—for example, because of bandwidth, battery, or incentive budget—the most relevant data can be obtained, rather than an arbitrary set of samples.

Incentives and cost sharing

Sensing resources' contributors must usually realize some benefit for the shared system to function. The benefit, however, need not be monetary. In many cases, there is an inherent incentive for the user to take samples and share them within a larger user group, such as in the earlier examples of a runner mapping broken sidewalks for her health enthusiasts' e-group, for car drivers in vulnerable river valleys to report flooded roads, and for homeowners to map noise levels in their community.

In other instances, the application users who need the sensor data can provide incentives. For example, a surfer who lives some distance from the beach might place a request for current images showing wave conditions at the beach in a region of his interest (by using a map-based interface, for example) and offer a small monetary compensation in return. News networks might use similar techniques to get instant news coverage even before their correspondent reaches the scene. In other instances, the contributor might advertise usage rates much like rental rates for other equipment usage. For example, a shopping mall might charge a retailers' headquarters if the retailer accesses mall-surveillance video data for their back-end enterprise applications on a pay-per-use or monthly basis.

Security, privacy, and trust

Other practical considerations in enabling widespread adoption of a shared system arise in ensuring security of shared resources against misuse, protecting the privacy of users who are being sensed or sharing parts of their data, and providing estimates of reliability or verifiability of sensed data against malicious intervention or inadvertent errors.

Many applications are based on sensor data that users might share without significant privacy concerns. However, where privacy or data ownership is a concern, the data might be shared within restricted groups only; while the trusted coordination system might archive and index all data, applications might only be able to access restricted portions of it. Such an approach has been used in existing image-sharing systems that archive all images but let users access only data to which they are authorized.

For highly sensitive data, the contributors could locally compute summaries and share only that data. For instance, for image or video, scale-invariant key features that do not reveal the human-understandable visual content might be shared. The coordinator could use the features to build multimedia data indices. When an application requests the actual data, the data owners serve the data, according to their desired control and access policies. Sensor contributors can use sophisticated data-hiding methods to reveal aggregate results without revealing individual sensor values. Further, they can share data to varying spatial resolution and sampling rates among different application groups.

Trustworthiness of data is always a concern in shared systems. Systems where independent users publish information have their share of incorrect or offensive content. To some extent such problems are solved through community feedback, such as ratings associated with different contributors assigned by their users. Additional methods can be used to verify sensor data when correlation models in sensor values corresponding to a common phenomenon are known or can be learned.

Building a sensing infrastructure out of shared resources deployed and controlled by a large number of independent contributors rather than by a single governmental or commercial infrastructure agency has security and trust challenges. But the former is more immune to the visions of a "Big Brother" world, and that encourages us to overcome the challenges in the shared system rather than falling back on a centralized one.

Participate!

The increasing availability of digitized data enabled by information technology advances and the Internet has shown tremendous benefits for personal lives and efficiency of business processes in recent decades. We believe that easy access to measurements of physical variables that characterize various aspects of our social and economic life will help realize the full potential of this trend. While many types of sensors are now available and even deployed, applications still do not have seamless on-demand access to the relevant sensor streams. SenseWeb can provide the tools and infrastructure to enable such access by leveraging peer-produced sensing resources for data collection and sharing.

Several open sensor sharing interfaces have already been deployed in our research prototype and we encourage sharing sensor data, be it from embedded wireless sensors, network cameras, or cell phones. **MM**

References

1. L. Brown et al., "IBM Smart Surveillance System (S3): An Open and Extensible Architecture for Smart Video Surveillance," white paper, IBM, Aug. 2005; http://research.microsoft.com/iccv2005/demo/IBM_S3/IBMS3_ICCV05Demo.PDF.
2. A. Kansal and F. Zhao, "Location and Mobility in a Sensor Network of Mobile Phones," *Proc. ACM SIGMM 17th Int'l Workshop on Network and Operating Systems Support for Digital Audio & Video* (NOSSDAV), ACM Press, 2007; http://www.nossdav.org/2007/files/file-31-session6-paper1-kansal.pdf.

Readers may contact the authors at kansal@microsoft.com.

Contact Media Impact editor William I. Grosky at wgrosky@umich.edu and editor Utz Westermann at westermann@gmx.tm.