A Web Services Environment for Internet-Scale Sensor Computing

Rui Peng Kien A. Hua Georgiana L. Hamza-Lup
School of Computer Science, University of Central Florida
Orlando, FL 32816, U.S.A
{rpeng, kienhua, ghamza}@cs.ucf.edu

Abstract

In this paper, we present a Web-Services framework for publishing, browsing, and analyzing real-time sensor data. Existing work, based on database technologies, assumes a known set of sensors, deployed for a specific application, intended for a group of users who are aware of the application. In contrast, our work provides a new environment for publishing real-time sensor data, and sharing sensor-based applications with unsolicited users over the Internet. Providers register their sensors with a UDDI registry where any user can discover them over the Internet. Once the desired sensors are located, the user can employ two new communication protocols, namely SSCP and SSTP, to control the incoming sensor streams. As the data arrive at the user browser software, a corresponding plug-in such as a data animation module is activated to allow the user to “see” the data stream presented in an intelligible way.

1. Introduction

Nowadays pervasive sensing systems ranging from short-range wireless ad hoc sensor networks to more powerful capture devices such as video cameras are becoming a reality. However, data captured from these systems are normally buried in the Internet, partly due to the dynamic nature of the data and the lack of a common framework for sharing such information in the public. As more and more sensor-based services become part of our daily life, they call for new technologies that enable publishing, searching, browsing, and integrating sensor data on the Internet. Consider the following scenario. A stream monitoring sensor network is deployed in a brook by a group of geographers. The sensors measure water temperatures, turbidity, and precipitation, and periodically send back the data to an Internet-connected computer. These real-time data arrive as a stream and are published on the Internet by the geographers. Independently, a biologist, studying the life zone in the same brook, searches for the information on the Internet. Within a few seconds, she learns about the data published by the geographers, and starts to browse and analyze the data stream on her own computer. As the biologist keeps discovering more sensors deployed by other researchers in other parts of the brook, she begins trying to keep track of the very part of the brook having the highest temperature. As another example, a government might want to stimulate economy by publishing real-time traffic data, collected through sensing devices along expressways. Availability of such information would encourage development of new commercial products such as a more intelligent GPS system that can reroute a trip when recognizing slow traffic ahead. To enable such services, we have developed a WISE (Web-based Intelligent Sensor Explorer) framework to provide the following services: (1) a publishing facility to allow sharing of real-time sensor data on the Web; (2) a search mechanism to enable unsolicited users to discover relevant sensors; and (3) an intelligent browser to provide the user interface to view, analyze, and query the sensor data streams in an intelligible ways. We envision an all-new Internet where innumerable sensing systems, deployed by numerous publishers, expose different data which can be shared freely with a wide range of unsolicited users, much like the way web pages are publicly shared today.

Although different Internet architectures can be considered for the WISE implementation, we opt for the Web Services approach because this technology has become mature with well-defined specifications of itself [18] and its closely related components (e.g. [15][16]. Furthermore, the following characteristics of Web Services [5] make it particularly suited for the WISE environment:

• Interoperability: A Web Service provides hardware and software platform independence. Any client who uses the standard Web Services technology can easily access a Web Service.
• **Encapsulation**: A Web Service can easily be consolidated in any application, regardless of the internal programming details of the component.

• **Availability**: A Web Service developer is able to publish enough information for any other developer to use and create a Web Service client.

• **Modularity**: Web Services provide modular advantages such as reusability and extensibility.

• **Self-description**: Web Services have the ability to describe themselves in a way that can easily be recognized.

• **Public accessibility**: Web Service descriptions are provided through a public repository, and can be found and used by any user.

In this study, we are able to leverage the above features in developing the WISE framework. Our main objective is to build an Internet-scale environment that enables sensor owners to conveniently publish information about their sensors, and assists unsolicited users to locate relevant sensors and intelligently browse the data streams in a wide variety of ways.

The remainder of this paper is organized as follows. We survey related work and identify the challenges in Section 2. Our proposed solution is introduced in Section 3. In Section 4, we describe the prototype, and discuss the experimental study. Finally, we offer our concluding remarks in Section 5.

2. Related Work and Challenges

In recent years, there has been a growing interest in sensor data management, with the main focus on building sensor databases atop wireless ad hoc sensor networks. Madden et al proposed a technique, called Fjords [10], for query processing on continuous, never-ending sensor data streams. In this work, the sensor network is modeled as a streaming data source by providing a sensor proxy as the sensor’s interface into the query processor. Yang et al modeled sensor streams as virtual relations [19]. In [11], Madden et al developed the TAG service that distributes declarative queries into the sensor network and coordinates sensors on in-network aggregation. This scheme pushes query operators into the network and aggregates partial results at intermediate nodes, greatly improving the query efficiency while cutting back power usage. These studies provide effective techniques for sensor databases and query systems, but their main limitation is the problem scale, being within a single sensor network. In this paper, we aim at a higher level, namely Internet-scale sensor-based services and applications, while utilizing these existing techniques to hide the lower-level sensor management details.

Another related work is the IrisNet [4]. In this study, Deshpande et al present techniques for building and querying wide area sensor databases. This work deals with widely deployed resource-abundant, powered sensors like webcams and organizes them into a distributed database. The entire database is logically treated as a single XML document which is partitioned among multiple sites. The techniques proposed are effective for fragmenting and partitioning a database, routing and processing queries, and caching remote data locally. This work, however, does not consider streaming data and only works with a predefined database, i.e. all the sensors are assumed to be owned by the same organization.

Unlike existing work, our proposed WISE environment is not a sensor database management system. Rather, it is an extension to the Web to enable exchange of sensor data and sharing of sensor applications. We summarize some of the differences between the current database approach and the proposed environment in Table 1.

![Table 1. Differences between Sensor DB Management Systems and WISE](image)

Due to differences in requirements between the sensor database management approach and the proposed browser-based sensor computing environment, WISE needs to address the following challenges:

• **Publish sensor data.** Sensors are generally limited in power and memory [2], and must deliver data continuously at well-defined intervals in the form of streams [10]. Since publishing is the first step
for sharing data with unsolicited users, our WISE framework must provide a mechanism to make streaming sensor data sources available online. While current web servers show good efficiency when publishing static objects such as HTML web pages, they are not capable of publishing dynamic sensor streams. It is not feasible in terms of timeliness by first saving the streaming data into a file, and then sharing it as a static object.

- **Effectively manage sensor meta-data.** Meta-data are structured descriptions of the actual data collected by sensors or sensor networks. Meta-data provide the semantics of the real data, and play an important role in helping integrate sensor data sources. On one hand, sensor owners need to use meta-data to advertise their sensors; on the other, data users rely on them to identify and access the appropriate data. A major challenge for effective meta-data management is the heterogeneity in sensor meta-data. There is currently neither a standard for sensor meta-data creation, nor a shared framework to manage them. Different research groups gather data with different goals in mind, and most probably employ a meta-data schema best for the data collection, not for sharing. An effective meta-data management framework should allow sensor registrations, build indices and facilitate sensor discovery (e.g., keyword search).

- **Efficiently deliver real-time sensor data.** Current real-time communication protocols focus on multimedia delivery and are not designed specifically for streaming sensor data. They incur unnecessary overhead for sensor streams due to the complexity of encoding/decoding, synchronization, etc. RTSP [13] and RTP [14] are two standardized communication protocols for controlling real-time transmission of streaming media. RTSP is an application-level protocol that provides an extensible framework to enable controlled, on-demand delivery of real-time data, typically audio and video. RTP is a transport protocol that provides end-to-end network transport functions suitable for applications transmitting real-time data. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery. While RTSP, RTP, and some other popular protocols are mainly designed to address various issues concerning streaming multimedia transmission, they cause significant unnecessary overhead for sensor data delivery that has a more relaxed real-time requirement. New light-weight application layer protocols are needed to more efficiently handle sensor streams.

- **Provide effective browsing methods for sensor data.** While HTML web pages contain all the information for a standard web browser to display, streaming sensor data do not. If original sensor data are shown on the screen without interpretation, the user will only see a screen of meaningless numbers. To provide more insight and intuitive browsing experience, various approaches can be used to access streaming sensor data under different settings, including visualization (e.g. temperature monitoring), analysis (e.g. security anomaly detection), and query processing. Ideally, the framework needs to be extensible to allow seamless integration of various sensor applications.

To address these key challenges, we have designed our WISE framework based on the Web Services architecture using the techniques discussed in the following section.

### 3. Proposed Framework

To make the paper self-contained, we first briefly describe the Web Services architecture before discussing the proposed WISE framework in detail. A **Web Service** is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifact; and supports direct interactions with other software applications using XML based messages via Internet-based protocols [17]. Some of the key technologies are as follows:

- **SOAP (Simple Object Access Protocol)** [16] is a W3C standardized protocol intended for exchange of structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.

- **WSDL (Web Service Description Language)** [18] is also a W3C standard that provides a model and an XML format for describing Web services. WSDL describes a Web service in terms of the messages it sends and receives, typically using an XML Schema.

- **UDDI (Universal Description, Discovery, and Integration)** [15] is a registry protocol that defines a set of services supporting the description and discovery of (1) businesses, organizations, and other Web services providers, (2) the Web services they make available, and (3) the technical interfaces which may be used to access those services. Based on a common set of industry standards (including HTTP, XML, XML Schema, and SOAP), UDDI
provides an interoperable and foundational infrastructure for a Web Services environment for both publicly available services and services only exposed internally within an organization.

We recall that the proposed WISE environment has the following functionality:

- Providing a publishing mechanism for sensor owners to publish their data sources.
- Managing sensor meta-data and helping users search the Web to quickly discover desired sensors satisfying certain criteria.
- Delivering real-time sensor data to users in a streaming fashion.
- Providing a user interface to interact with users and assist them in understanding (e.g., querying, visualizing, and analyzing) sensor data.

This environment is illustrated in Figure 1. We revisit the example of the geographers to explain how this framework facilitates this application. To publish the data, the geographers provide relevant meta-data for their sensors; and a service description as well as a WSDL file are automatically created on a Sensor Data Server (SDS) to hold the meta-data. The service description is then uploaded to a UDDI registry. Sometime later, a biologist looking for related sensors enters search criteria into a local Sensor Data Browser (SDB). This software connects to the UDDI registry and looks for relevant service descriptions. As the browser displays a list of relevant service descriptions on the screen, the biologist identifies the desired sensor and clicks on it. In response, the browser interacts with the SDS using the operations defined in a Sensor Stream Control Protocol (SSCP). As the browser gets the real-time stream using a Sensor Stream Transport Protocol (SSTP), it feeds the data to a Visualizer for display. The various components are described in the following subsections.

3.1. Service Description

To publish a sensor service, appropriate meta-data must be uploaded to a UDDI registry. The current UDDI specification (Version 3) only allows a fixed set of fields, such as Name, Description, and URI. It is not possible to add sensor meta-data as new fields. In WISE, we overcome this limitation by entering meta-data in the Description field as a structured XML document. Normally, this field contains unstructured text describing the published business or organization.

![Figure 1. High-level environment](image1)

Each service description advertises a sensor service consisting of a collection of related sensors. For example, we can publish traffic monitoring sensors deployed at busy locations throughout the city to support various navigation applications. The service description is generated according to the XML schema illustrated in Figure 2. A sample service description is given in Figure 3. As shown in this example, a sensor service has a ServiceName, and a ServiceDescription, providing a general description about the service. Furthermore, it includes information for each sensor employed in this service. This sensor information has the following standard data fields:

![Figure 2. Description schema](image2)
Figure 3. Sample description

- **SensorName**, **SensorDescription**, and **SensorLocation**: These data fields provide the general description of each sensor.
- **Protocol**: By default, we set Protocol to be SSCP.
- **DataField**: This user-defined field allows the sensor provider to customize the format of the data stream. For instance, the sample description in Figure 3 indicates four distinct data fields in the sensor data: Temperature, Turbidity, Precipitation, and Time. We define Data frame as the unit for sensor data transmission. Thus, a data stream is a sequence of data frames, each consisting of a datum for each of the data fields.

3.2. Sensor Data Server

The Sensor Data Server is responsible for accepting incoming data from sensors and delivering the data to clients. As shown in Figure 4, it has three major components, namely the Publish Manager, the Stream Manager, and the Session Manager.

The Publish Manager allows a user to provide sensor meta-data, from which a service description is generated together with a WSDL file. The publish manager then automatically connects to a UDDI registry and registers the sensor data source.

![Figure 4. Sensor Data Server architecture](image)

The Stream Manager accepts incoming sensor data and directs them to either local applications (e.g. local file system, database), or remote clients through the Server Session Manager.
manager spawns a new SSTP session; whereas each subsequent request for the same stream joins the existing session. Essentially, a client uses two separate channels to interact with the server, one for control messages, and one for data delivery. The control channel is bi-directional, but the data channel only goes in one direction.

### 3.3. Sensor Data Browser

The current SDB consists of two components, the *Client Session Manager* and the *Plug-in Manager*. We discuss them as follows.

The *Client Session Manager* has three major functionalities. First, it is responsible for accepting user search criteria and discovering relevant sensors on the Internet. Second, it connects to SDS’s and establishes a client session for each incoming stream. A client session has two channels, control channel and data channel. Third, it directs data to the *Plug-in Manager* for presentation to the user. Optionally, it can also dump the data into a local file or database.

Currently, the only available plug-in is the Visualizer, which is responsible for displaying the incoming data stream on the screen. For a data stream with potentially unlimited information arriving at a very high rate, summarization is a good way for the user to “see” the stream. Different types of summarization methods can be used under different circumstances, including visualization, analysis, and querying, where visualization is probably the most intuitive way for most situations. The current Visualizer implements a simple Presentation Frame concept analogous to a frame in a video stream. A Presentation Frame is the minimum unit of data for display. That is, a presentation frame is a collection of data frames that are visualized together on the screen. For example, the Visualizer may display the presentation frame by drawing a time series chart where every point corresponds to a data frame. We can view a continuous stream arriving at the Visualizer as a sliding window over the data stream. By visualizing these consecutive frames at a high display rate, we create the experience of motion similar to video. In other words, the standard WISE browser summarizes and presents sensor data to the end user in the form of data animation.

### 3.4. Communication Protocols

The *Sensor Stream Control Protocol (SSCP)* is an application-layer protocol built atop SOAP. It defines the operations how an SDB can interact with an SDS. As a simple analogy, the SSCP acts as the network remote control for sensor stream playback by providing a set of useful methods. SSCP does not deliver the stream itself, but rather uses SSTP for real-time data transmission. The semantics of the control methods are listed below with an illustration presented in Figure 6.

- **PREPARE.** Request the SDS to allocate resources for a sensor stream and prepare for the data delivery. If this is the first request for a particular
stream, the SDS creates a new server session and adds the requesting SDB to the session’s SDB list. Subsequent requests can share this stream by simply adding their SDB to this session.

- **START.** Request the SDS to start the data transmission through the data channel. If this is the first request for a stream, the SDS starts an SSTP transmission on the SSTP port to the SDB. For subsequent requests for this stream, the SDS lets their SDB join the existing SSTP transmission. This operation can also be used to resume data transmission after a pause operation.

- **PAUSE.** Temporarily stop the stream transmission without releasing server resources. Any data transmission during the pause period are lost. If the PAUSE request is from the last active SDB on this stream, the SDS stops the data transmission.

- **STOP.** Notify the SDS to stop the data transmission, free server resources, and close the corresponding server session if this is the last active SDB receiving this stream; otherwise, the SDS continues the data transmission for other SDBs while purging the resources for the stopping SDB.

The Sensor Stream Transport Protocol (SSTP) is another application-layer protocol for transmitting streaming data. It is based on UDP to leverage its broadcast/multicast capability so that the system scales to a large number of clients. The rationale for using SSTP rather than SOAP in sensor data transmission is to reduce overhead. Though SOAP is designed to be a light-weight protocol, it is not necessary to wrap every data packet with a SOAP envelop. In contrast, the current SSTP only adds a sequence number to a data packet. The *sequence number* increments by one for each SSTP packet and is used by the SDB to detect packet loss and ensure in-order delivery.

4. Prototype and Experimental Study

As a proof of concept, we have developed a prototype using the techniques discussed in Section 3. It has a simple user interface with a basic set of functionalities. A screen shot of the Sensor Data Browser (SDB) is shown in Figure 7. The browser is both a registry browser for UDDI registry entries and a data presenter. To search for a sensor data service, the user enters a set of keywords as the search criteria. To make the search more specific, the user can also enter additional keywords for the three provided metadata fields. The keywords entered for *Datatables* are compared with the names of the sensor data fields declared by the sensor providers.

After the “Search” button is clicked, the browser connects to the UDDI registry server and retrieves matching entries from the server. The sensor meta-data are shown on the screen as in Figure 8. Our sensor data browser displays only the standard descriptive information such as the name of the service, its description, and the locations of the sensors to help the user identify the desired sensor service. The user can now select a service by clicking on its name. This will activate the Visualizer, which prompts the user to choose from a number of different visualization methods. Figure 9 shows a screen shot when the user views the variation of the water temperature on a time series chart.

The current prototype is intended as a proof of concept. Future research will provide a more complete browser with functionalities such as a set of useful plug-ins for data processing.

![Figure 7. Browser interface – search](image)

![Figure 8. Browser interface – search results](image)
5. Concluding Remarks

The major contribution of this paper is the development of WISE, the first Internet-scale framework for publishing, searching, and “browsing” sensor data. We envision a future Internet where innumerable sensing systems are deployed by numerous publishers exposing different real-time data over the Internet to share them freely with a wide variety of unsolicited users. We identified the key challenges in supporting such an environment, and proposed solutions that are effective and flexible. The significance of this framework lies in its potential to facilitate the development and deployment of sensor-based applications on the Internet. For example, the powerful scientific visualization and analysis tools are to be developed as plug-ins to the WISE framework. We have built a prototype as a test bed to evaluate these techniques. Future research can focus on developing more sophisticated general-purpose plug-ins for browsing, querying, and analyzing sensor data. It is also worthwhile to leverage the proposed framework to develop and deploy new interesting sensor-based applications. Extending the current framework with the firewall-like capability to allow enterprises to develop sensor applications for internal use only is also highly desirable.

6. References