TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/09 pp491-504
Volume 16, Number 5, October 2011

LiveWeb: A Sensorweb Portal for Sensing the World in Real-Time

Xiaogang Yang, Wenzhan Song**, Debraj De

Sensorweb Research Laboratory, Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA

Abstract: A state-of-the-art *sensorweb* is a global observation system for varied sensory phenomena from the physical world and the cyber world. This paper presents the architecture, design, and application of a *sensorweb* service portal called the *LiveWeb*. This portal has been published on the Internet and is used by researchers, students, and also other communities. This system has been used to represent and monitor real-time physical sensor data and cyber activities from ubiquitous sources. *LiveWeb* meets its goal of providing an efficient and robust sensor information oriented web service, enabled with real-time data representation and notification. Living in the current world with the immense magnitude of information, it is important to keep different communities updated and informed with their context specific data. There are search engines available in the Internet to find relatively static items, but not to observe the events in real-time. In addition, mostly the sensed data and events have meaning only when accompanied with corresponding geographic information. *LiveWeb* is an effective and efficient sensorweb service, that tries to fulfill the aforementioned requirements. The *LiveWeb* system consists of three main components: (a) special features of a PHP web application, (b) a Java background processing program, and (c) a database. It is a robust system and is currently running efficiently under the environment of Ubuntu 6.10, Apache 2.0, PHP 5.0, JAVA 1.60, and MySQL 1.6.

Key words: sensorweb; geographic information; sensor network; real-time

Introduction

The Internet is a web connecting computers all over the world. It has changed people's lives by providing efficient information sharing and enhanced user experiences. On the other hand, there are huge numbers of sources of dynamic information from physical sensors and information providers. As a type of Geographic Information System (GIS), sensorweb is the corresponding counterpart that can efficiently connect those ubiquitous sources of dynamic sensory information and then integrate them as a part of the greater network. Sensorweb is becoming more and more necessary and also popular in all sectors of life. It provides a powerful tool that gives accurate and up-to-date information from the physical world to the cyber world. People in daily life get plenty of conveniences from these kinds of systems and services. As a type of GIS, sensorweb service is being implemented by many industries (structural, medical, and surveillance, to name a few), organizations (e.g., law enforcement agencies, federal and state governments, etc.). Public and private sectors are utilizing their necessary dynamic data, supporting existing and new demands of numerous applications. Sensorweb is gaining huge importance and is adopted by almost every industry due to the rapid growth and development in this field, and the application needs. The Internet has World Wide Web (WWW) standards for accessing globally shared information conveniently. However, there is no

Received: 2011-06-21; revised: 2011-08-29
** To whom correspondence should be addressed.
E-mail: wsong@gsu.edu

standard framework for accessing *sensorweb* information. In building such a system, there are several challenges such as variation in format of data source, data management, etc. These all have motivated us to build an efficient interface for this purpose. The scenario and the effectiveness of *LiveWeb* is shown in Fig. 1

We have observed and attempted to solve the following potential challenges in building a state-of-the-art sensorweb interface.

(1) Nature of data The distinction between sensorweb data and the normal data is the features like spatial and temporal contexts. Spatial information is usually used for applications like guiding direction, physiognomy observation, etc. Temporal information can give the impression of some live phenomenon and trend prediction with history of data. Any high level information in sensorweb usually does not have meaning without these features.

(2) Data format Sensorweb generally contains heterogeneous information sources having different data formats. This increases the difficulty in standardizing the data format for retrieval of data.

(3) Data representation With huge varieties of information, it is critical to design a good data representation format. There can be continuous streams of incoming information, as well as discrete information. Also some information only has meaning when related to specified events. These all bring challenges to represent them meaningfully to the users.

(4) Data management There is a tremendous amount of incoming data to be stored and managed in a sensorweb system. Previously the data in a sensorweb was distinguished by source IDs or source categories, but these cannot work well for large scale heterogeneous data.

All these challenges are with respect to the system. But there are also a number of user specific requirements for such desired sensorweb portal systems.

(1) Access from anywhere Users want to access these useful services from anywhere. They may not have the convenience of separately installing an interface for such sensorweb services. In such a scenario the Web approach is the most popular method.

(2) Access at anytime With more mobility in people's lives and greater use of portable devices, it's becoming necessary that all the data is available anytime for user access.

(3) User preference User preference based services are becoming an important part of modern applications. Users intend to get only the data that is useful to them. For example, people may want to know just the temperature distribution close to their location.

Motivated by all these, we have proposed a *sensorweb* service portal called *LiveWeb*. This system has been used to represent and monitor real-time data from physical sensors and also from other information providers. The *LiveWeb* adopts content-based searches for sensorweb data and supports real-time monitoring, subscribe/alert, and publishing services. *LiveWeb* gives facilities to users to publish their data efficiently by simply starting their own serial forwarder and submitting the configuration files. Then *LiveWeb* can automatically import these data and provide content-based services based on these data.

The outline of the paper is as follows. In Section 1 we discuss the related work. In Section 2 we give an overview of the *LiveWeb* system architecture. Then we describe a universal data collection and storage method in Section 3. In Section 4, we explain the information search algorithm. Then we describe the

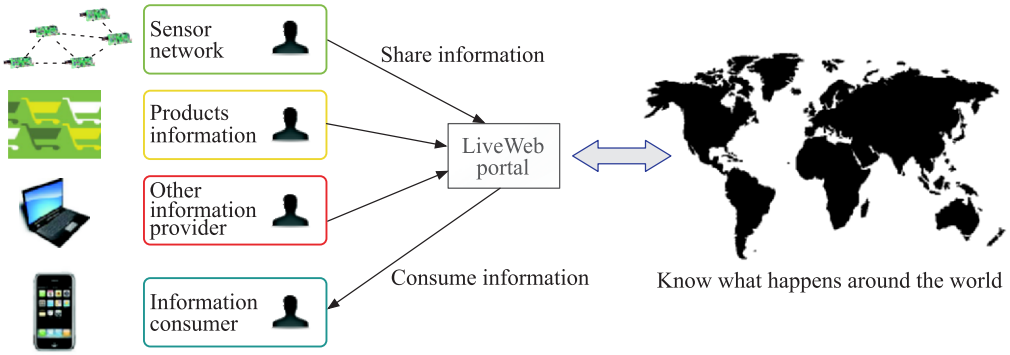


Fig. 1 ***LiveWeb*** **scenario**

subscribe/alert mechanism in Section 5. In Section 6 we demonstrate some case studies of *LiveWeb* usage. Finally we conclude the paper and discuss future work on *LiveWeb* in Section 7.

1 Related Work

In this section we discuss related work done in the area of real-time sensor network data management and representation.

1.1 Open Geospatial Consortium standard (OGC)

One of the most popular standards in this area is the Open Geospatial Consortium Standard, known as OGC[1]. OGC has developed the Sensor Web Enablement (SWE) which specifies interfaces and meta-data encodings that enable real time integration of heterogeneous sensorweb data into the information infrastructure. Then developers can create their applications, platforms, and products based on the standards. In SWE, XML standards of geospatial content and services, data processing, and exchange are described by the following components: (a) Observation & Measurement (O&M), (b) Sensor Model Language (SensorML), Transducer Model Language (TransducerML or TML), Sensor Observation Service (SOS), Sensor Planning Service (SPS), Sensor Alert Service (SAS), and Web Notification Service (WNS). These components together build up the architecture of the sensorweb service and the application space of the sensorweb.

1.2 Microsoft SensorMap

Microsoft has also proposed a system to monitor and present physical sensors in the real world. It is called the SensorMap[2]. SensorMap allows owners of sensor networks to register their physical sensors and publish their data on SensorMap. They use GeoDB to store the sensor network information, DataHub to retrieve the new sensor data to enable real time services, and the aggregator to summarize sensor data in a specific area to clients.

1.3 Google PowerMeter

Google PowerMeter[3] is a service that focuses on the real usage of powermeter sensor networks. If users have specific devices belonging to specific utility companies, they can monitor the power consumption and represent the summary of the usage on the Internet. Then people can track distribution of power consumption and can make better decisions for using power to save energy.

1.4 Device description language

Device Description Language (DDL)[4] uses XML encodings to describe a device. DDL summarizes devices to properties and operations; each operation includes input, output, and processing. This way, outputs of the devices are unified, helping the data representation. DDL gives a perspective that is different from SensorML. DDL is based on a device, while SensorML is based on a process. DDL provides higher granularity services for sensor register and discovery.

1.5 Sensorweb applications

Currently there are some sensorweb applications for sensor networks for specific purposes. MQTT-S[5] proposed service integrating SNs and a publish/subscribe mechanism, but is based on applications running on the mote and is not content-based. This leads to a limited scope of applications. The work in Ref. [6] introduced a practical medical response system to serve the patient's daily life needs and the diagnosis by doctors. MediaAlert[7] is a system for automatic monitoring and timely dissemination of multimedia information to a range of mobile information appliances based on interest profile of users. This system is not for a sensor network, but the content retrieved by this system can be treated as a sensor because of its content variation and its real-time property.

As a result of the involvement of many communities of researchers and engineers, the OGC SWE[1] standard is relatively complete and popular. However, most of such existing systems and standards don't mention data sharing between sensor networks. There are hundreds and thousands of sensor networks in research labs, houses, facilities, and public communities. If all of these sensor networks cannot share their data with each other, the scope of sensor network will be very limited. SensorMap emphasizes the publication of a sensor network and its data, but its complex interface and construction sometimes leads to difficulty for non-technical people. Google PowerMeter actually is a specific application that helps people monitor energy

usage of a home or bigger communities. However, Google PowerMeter only provides power consumption monitoring, and serves only limited users. These are of limited scope with respect to the bigger goal of sharing sensor data globally. *LiveWeb* aims at providing a platform that can unify different sensor networks, represent their data in a reasonable way, and promote sensor networks in variety of applications for improving life in all the communities.

2 LiveWeb Architecture

In this section, we first analyze the requirements which guided us to design and implement our *LiveWeb* architecture. Then we describe the components of *LiveWeb*.

Here we analyze the requirements for state-of-the-art performance of *LiveWeb*:

(1) Universal data aggregation *LiveWeb* is constructed to support heterogeneous information sources to improve ubiquitous applicability. It is required to keep the complex underlying sensor data transparent to the users. In this way, the users don't need separate applications to handle and share their data.

(2) Real-time data representation Currently there is no specific standard algorithm to predict and analyze the real-time data. In this aspect, real-time representation is unreplaceable to utilize the temporal and spatial properties of sensorweb data. Since the user community of sensorweb has grown from researchers to non-technical people, reasonable representation of data is necessary.

(3) Content-based information search There are many different ways to find sensorweb information such as ID-based, topic-based, and content-based[8]. The ID-based data approach is widely used in sensor network research because it is straight forward, easy to use, and reliable for a single network without ID conflicts. The topic-based data approach is used mostly in current sensorweb applications because of limited categories of sensors. With topics specified for sensors, users can observe the data such as temperature, humidity, etc. The third one is a content-based method. This method is used commonly in web searches. But in sensorweb, the content-based approach is not used widely because of its complex database requirement. However, the topic-based method has been inefficient due to the explosion of new categories and functions of sensors. People are willing to use their own criteria for searching for what they need. So the content-based method is the best choice and is promising.

(4) Offline service Even users can access useful information when they are online, but they cannot be notified of some important events when they are offline. This may lead to loss of precious information or opportunity to make right decisions. Thus, enabling offline service is essential.

(5) Quality of service The system should meet the various QoS of the different services it provides. For instance, a search should respond with the right results in short time and an alert system should be reliable and efficient. Otherwise, all the features in *LiveWeb* will be in vain.

(6) Nice interface A good Interface can guide users to use the system easily without much instruction. This is also one of the primary goals of *LiveWeb*.

To meet these requirements, we have comprehensively designed the *LiveWeb* system architecture. *LiveWeb* is composed of several components. These components have their unique separate roles, but also cooperate with each other for providing the overall system. Mainly there are three components: (a) the sensor data collector, (b) the data storage center, and (c) the web server. These three components have important sub-components to make *LiveWeb* functional. Figure 2 illustrates the data flow and the architecture of *LiveWeb*. In the lower layer, first ubiquitous information is imported from the aggregator. Then the data is stored in a data center in a special structure. Finally, the data can serve the data user in real-time.

We now explain the components of *LiveWeb*.

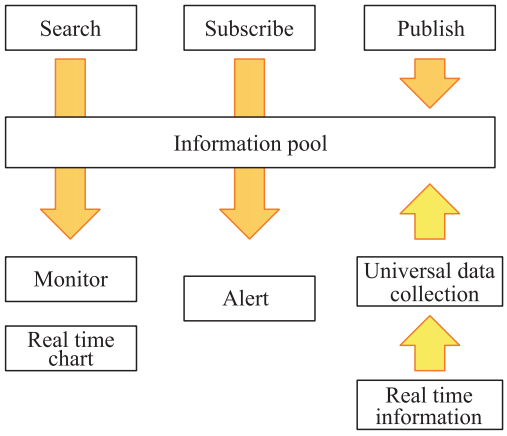


Fig. 2 *LiveWeb* architecture and data flow

(1) Sensor data collector As *LiveWeb* is designed to be a geo-information portal, collection of data from various data sources is the first step to realize *LiveWeb*. This component is different from traditional data collectors. We use a standard parser to collect the large range of different kinds of data, instead of developing individual collecting interfaces. This sensor data collector can parse any data into a standard format. This component, written in JAVA, enables the raw data to be usable for the following components.

(2) Data storage *LiveWeb* uses this component to store data. The data is composed of two parts. One part is assisting data, the other is history data. The assisting data provides the sensor or data source information for future services. The history data provides the history of specific sensor data to enable various data representations. Storing these data in a database cannot be done with a simplistic approach. The mechanism to store them has a direct effect on the efficiency of *LiveWeb* services.

(3) Web server *LiveWeb* aims to be an effective sensorweb portal. We have developed web GUI with PHP, which is the most popular and efficient dynamic language for developing web applications. *LiveWeb* provides all services including search, subscribe, and publish through a web browser. Users can access the data and use them anywhere anytime without limitation.

(4) Alert system The alert system, developed in JAVA, is running all the time the web server is running. This alert system is in charge of the *LiveWeb* notification service. To realize real-time service, off-line notification is critical. The alert system is based on an efficient subscription/alert mechanism and various communication methods. *LiveWeb* treats the new item included in a users' criteria, as an event. The alert system sends the event notification to the users automatically.

With these modules, *LiveWeb* provides diversified services based on rich real-time data from a sensorweb.

3 Universal Data Collection

The first feature needed in *LiveWeb* is importing enough information to provide users with good information services. As is known, geo-information is always designed for specific purposes. Because of the huge size and complexity of GIS, specific data format means easy evaluation for future processing. For example, a home scale sensor network can collect real time information in some home environments. Its data may have just reading value fields. But, for some sensor network monitoring activities in a volcano, the data may include more complex fields like GPS data, seismicdata, lightning data, etc. This variety of data formats are effective for the corresponding researchers but not easy for data collection from a high level point of view. We have summarized below the difficulties of collecting varied geo-information.

- **Various format** As mentioned before, the information format is a key obstacle in integrating ubiquitous geo-information around the world. There is no "official language" or standard in this field. Also we cannot require the data providers to use the standard format to satisfy our system purpose. If we have no such standard language or protocol, we need to have a good translator which can translate different formats into the one suitable for *LiveWeb*.
- **Unpredictable data flow** If two programs on the Internet want to communicate with each other, they need to build up a connection between them as do the data sources and *LiveWeb*. However, no one can predict when the data will come in and when they will start again, because some information is transmitted periodically while some is transmitted just when necessary. We cannot schedule connections ahead of time. For different data sources, we cannot build separate collectors for them because that will require significant resources for the system.
- **Authentication** *LiveWeb* aims to integrate data as much as possible, but it doesn't mean it will accept any kind of data blindly. *LiveWeb* just maintains useful and qualified data. Uncontrolled data collection will lead to DNS attacks automatically. To avoid such situations, data authentication is necessary.

3.1 Standard data translation

We have designed data gathering middleware which can translate data from any source with the format definition provided. For example, most systems of sensor networks are developed based essentially on the TinyOS[9] operating system. If someone wants to recognize data from such a system, they have to use MIG[9] or develop their own tool. This middleware can help users avoid this situation. With the mechanism

shown in Fig. 3, any type of message could be decoded based only on the XML file provided by users without doing the code-level changes. With this generic XML based decoding mechanism, the recognized data can be used to reorganize it to the required JSON format. The philosophy of this middleware is to help data users develop their applications based on sensor networks transparently, because the data formats in sensor networks are still different from each other. This kind of middleware can increase the development efficiency.

The XML file used in the middleware has three parts: Message Hierarchy, Message Format, and Data Format. Message Hierarchy is related to the hierarchical structure of the received message. The middleware needs first to identify the received message based on the structure description. Message Format includes message field, size, and offset information, which are used to help it decide where to get correct information. After identification and localization of the incoming messages, it decodes the data payload according to the Data Format.

3.2 Storage indexing

Sensor data management is the foundation of *LiveWeb*. There are two issues involved to guarantee the performance of the system.

(1) Sensors are continuously sensing the environment and sending back data periodically. Let one sensor's sampling rate be 50 Hz and sample data size about 20 Bytes including the timestamp, sensing value, and ID. There will be $50\times20\times3600=3600$ KB data for one sensor in just one hour. If added with other sensor information, the data size could be even greater.

(2) There are many types of sensors. They are used to measure and detect a huge variety of conditions including: temperature, pressure, level, humidity, speed, motion, distance, light, and many other conditions. There are many versions of each type which may use a different sensing principle. For each sensor network, there are lots of sensors deployed. Sensor network data amounts differ greatly for various uses.

The first issue does not influence the efficiency very much because Sensor ID and update time can help the system retrieve the data easily, except the data will occupy more space in a database after a long time of data aggregation. However, the second issue is different. Content-based observation cannot depend on a sensor's ID. A brute force search in sensors which satisfy the specified content must lead to huge inefficiency because the scale is unpredictable. Learned from search strategies from the existing literature in Refs. [10-12], indexing the sensor data in a proper way can improve the performance significantly.

The mechanism of indexing sensors is described as follows. Indexing is based on sensor ID (SENID) and sensor name (SN). Then two tables will be established, one is "keywords", the other is "keywords list". "keywords" contains the keyword ID (KEYID) and the keyword. If one sensor contains keyword A, it will be appended on the keywords list led by keyword A. But just naively indexing the sensors is not a final solution, because a sensor name is the combination of hundreds

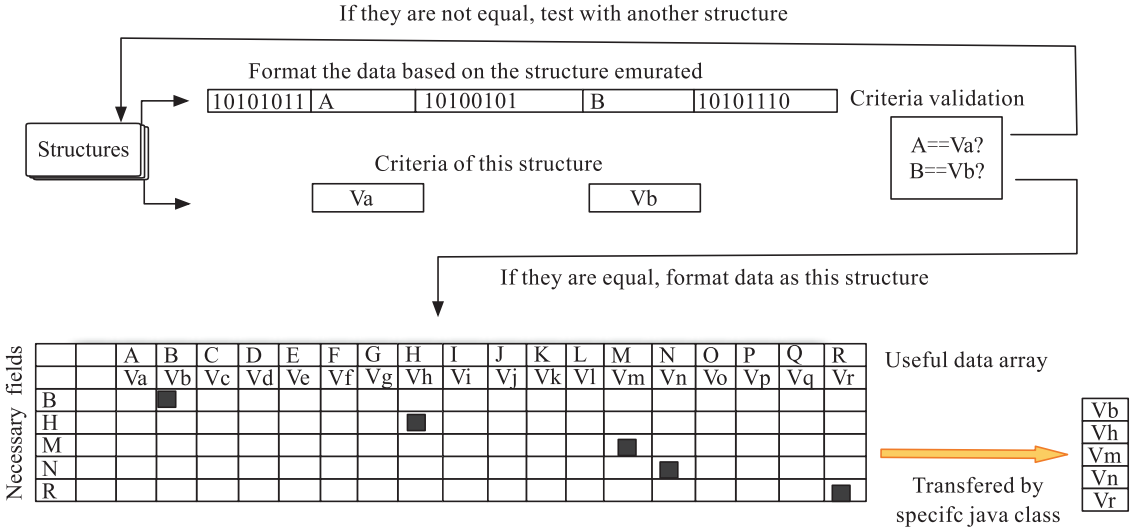


Fig. 3 XML based data decoding

and thousands of symbol and word derivatives. If each word or symbol leads a list, there will be too much redundancy and too many useless entries. This may not yield service performance because of the benefit of multi-indexing. But this will bring in some new problems. One is the database complexity problem. Too many trivial entries won't link many sensors. For example, if a sensor's name contains a sequence of non-letter and non-digit symbols, there is no need to build a list for it specially, because maybe it is the only one containing these symbols. The key point is that these symbol sequences have no meaning in spoken languages. The other problem is the access limitation problem. "network" and "networks" are actually two words with the same meaning. The sensors containing either of them should be put into a keyword list. A fuzzy search is based on this feature. Considering these two problems, the sensor name will be parsed into a unique word array before being appended. The data indexing process used in *LiveWeb* is as follows.

(1) Remove symbols except letters and digits.

(2) Split the name into a words array.

(3) For each word, use the stemmer algorithm[13] to replace it with its root.

(4) Remove the duplicate word from the array.

After this process, the sensor will be appended to the lists headed by these words. Then the indices of sensors are built. In the development process, we found that users may not know the sources in our system and sometimes cannot recall the right name of sensors. The try and see approach can exhaust the users in this case. In the work in Ref. [14], the authors have developed a fuzzy word search to suggest underlying data to users instead of a simple complete word search. *LiveWeb* proposes a solution for this situation, called word connection. Figure 4 illustrates the mechanism of sensorweb data storage.

3.3 Publish mechanism

As a sensorweb portal, *LiveWeb* provides a standard and easy interface for users uploading their data. We have described how *LiveWeb* imports data based on an XML file instead of modifying code-level applications. Based on universal data collection, *LiveWeb* can automatically import data from any user after they have submitted a qualified XML file. These users need the following to publish their data. They need to have their own serial forwarder running on their own server. This serial forwarder can export data continuously. The data can be sent to a *LiveWeb* server through the Internet. The *LiveWeb* server will import these data after checking their frequency and authentication information. As shown in Fig. 5, what users need to do is to submit their data format XML file and some accessory information such as item name, serial forwarder IP address, and authentication information. Users also can submit their own data processing class file. This is because even raw data can be interpreted, some of these raw data cannot be used directly and need a specific process for transformation before they can be used for display or alert. Users can submit their own processing class file based on the interface defined by *LiveWeb*.

4 Real-Time Search Engine

In this section we have explained the real-time search mechanism in *LiveWeb*.

4.1 Search

The search service bridges users and information systems. *LiveWeb* can return results of the search after

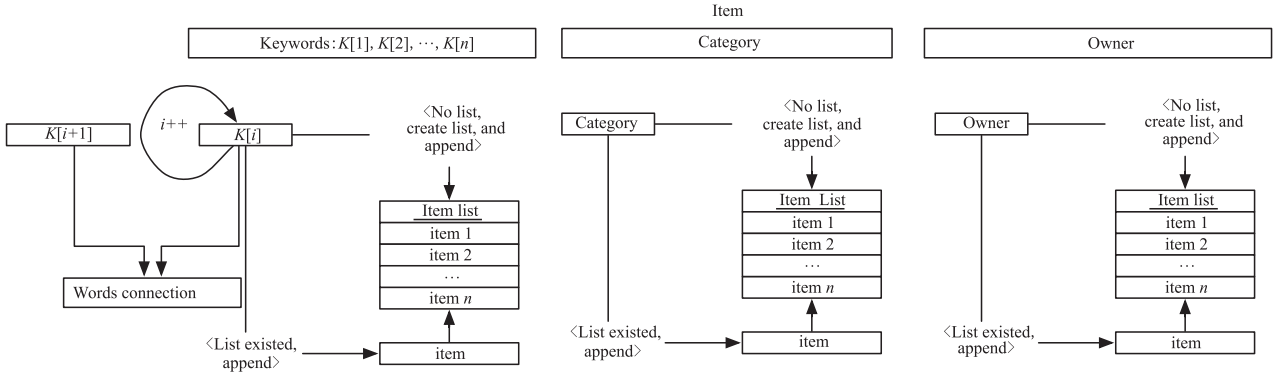


Fig. 4 Sensorweb data storage

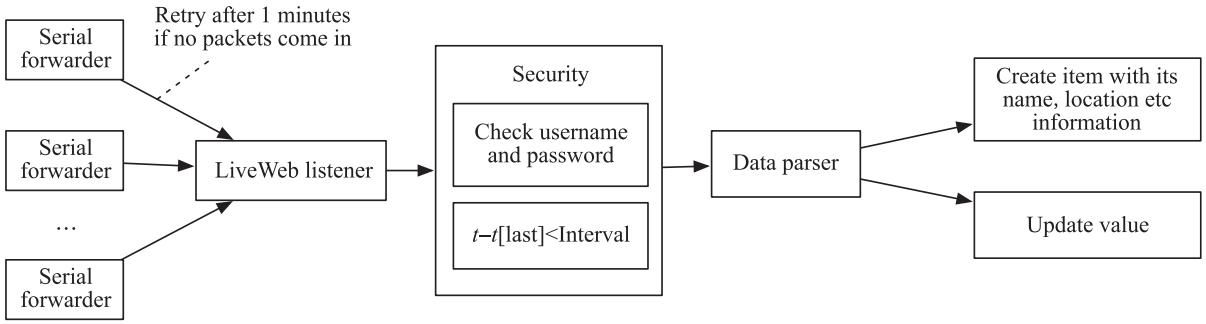


Fig. 5 Publish mechanism

searching in the system based on the query provided by the users. In Section 3, we have described the indexing mechanism to store data in *LiveWeb*. With the help of indexing, the searching can be much faster and more efficient. In general, search procedure includes four steps: query, parse, search, and return results.

LiveWeb uses this semantic analyzer to understand the meaning of a users' criteria. *LiveWeb* parses a query as shown in Table 1.

Searching is a reverse procedure of indexing. Its target is to get the result set satisfying the criteria of the query. The method of searching can be transferred into a Boolean formula. First get the set of the right category and the lists lead by "Any of the words". Then get the intersection of these two sets. Then use the lists of "All of the words" and "None of the words" to cut the set. Then use "exact phrase" to examine the set and to delete unqualified elements. Finally, use value range to delete those exceeding the range. The last set is the search result satisfying the users' query. Its work flow is illustrated in Fig. 6.

4.2 Real-time search

The search service enables *LiveWeb* to provide an easy approach to access and observe sensors and their data. *LiveWeb* also enables real-time searches. A real-time search is one of the most important topics in field searching. In this regard the work in Ref. [15] proposes a real-time mechanism for real-time operation and cooperation on a web page. Work in Ref. [16] proposes a real-time search of web pages. This new technology matches properties of real-time data very well. Real-time data is different from other kinds of data. Its effectiveness and its representation influence the value of data directly. As known, waveform, raw data, and images[17] are common formats for representing data. *LiveWeb* can provide different representation formats of sensor data. One format is a

Table 1 Query semantic rules

Symbol	Meaning	Label
ANYW	Any of the words	Independent words
ALLW	All of the words	Start with " "
NONEW	None of the words	Start with " - "
EP	Exact phrase	Phrases in quotes
CAT	Category	The value of "category" option, in the format of [ParameterName] = [Value]
MIN, MAX	Value range	The value of "min" and "max", in the format of [ParameterName] = [Value]

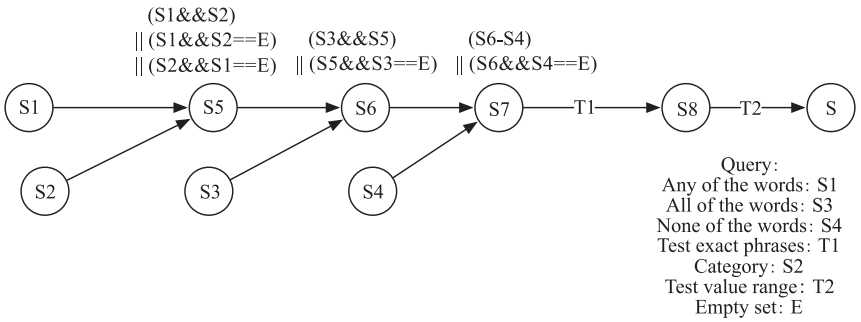


Fig. 6 Search mechanism

real-time display of the current value of sensor readings, we call "monitor". The other is a real-time waveform of the data, we call "real-time chart". In both of these ways, we adopt AJAX technology to increase the efficiency instead of refreshing the page automatically. AJAX's mechanism is illustrated in Fig. 7. The browser can send a request to the server through the javascript interface. After the server processes the request, it responds with certain strings. When the browser receives the response, it updates a specific area on the webpage.

4.2.1 Monitor

A monitor enables users to watch the real-time value (instead of the static value) of items they have searched for. As discussed in the previous section, searching can efficiently return results users want. Based on the result of searching, what *LiveWeb* does is to update the value part of each item periodically. First, users need to search their items of interest. After the items have been listed on the page, users add them to the monitor. The browser records the IDs of these items as $(ID_1, ID_2, \cdots, ID_n)$. This ID list is then sent to the server. Having known the ID list, the server generates a newest value set from the database and sends this set to the browser. The Javascript function catches this response and decodes them by the ID. Also, based on the ID, javascript can find the HTML elements and fill them with these values.

4.2.2 Real-time chart

A real-time chart gives another option to monitor real-time data in a more straight-forward way. With this function, users can see the waveform of any item or multiple items from anywhere. Waveform is one of the most popular ways to represent the data. With some analysis on the waveform, users can summarize the variation of history and can predict its trend. Users

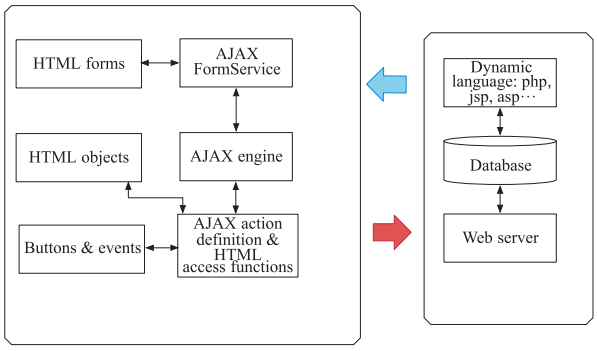


Fig. 7 AJAX technology

can also control the waveform parameters through a webpage. *LiveWeb* realizes this function with the support of AJAX and the flash technique. After users have chosen the items they want to see in the waveform, the ID list is sent to the server. This server replies with the newest value set to the browser as well as for the monitor function. However, with the updated values, javascript does not use them directly to fill up the HTML element. It sends them to the drawing component separately. Then the drawing component redraws the waveform with new data to enable dynamic data representation.

In *LiveWeb* real-time charts, the x-axis indicates the time, and the y-axis indicates the value. By default, the chart displays last the 15 minutes of data. When multiple sensors are selected, the period will be inversely proportional to the number of sensors. In this way, the web page loading speed can be guaranteed to be fast. Besides this, *LiveWeb* enables zoom functions. If users zoom in, the period will be extended. If users zoom out, the period will be shortened. With longer time intervals, users can view the general sensor data trend. With shorter time intervals, users can see the details of the sensor data pattern.

5 Subscribe/Alert Mechanism

Users can access useful information when they are online. They cannot be notified of some important events when they are offline. This may lead to loss of precious information or opportunities to make right decisions. The real-time search engine allows users to search for already existing items in the sensorweb, that is retrospective in nature. In contrast, a prospective search should allow users to submit a query that will then be evaluated by the search engine against items encountered in the future. A naïve implementation of a prospective search engine might simply execute all the subscription queries periodically against any newly arrived items. However, if the number of subscriptions is very large, this would result in a significant delay in identifying new matches if we only execute the queries very rarely, or in a significant query processing load for the engine. In *LiveWeb*, we reverse the items and these subscription roles. We index the subscriptions. For each newly arriving item, we issue a small set of

subscriptions for testing.

5.1 Subscribe

The subscription procedure is the same as the search at the beginning. The users can upload the query through the interface. However, the difference is that subscription can store the queries for future use. We assume there are n queries: $q_1, q_2, q_3, \cdots, q_n$. For each query, q includes keywords, category, and value range. The keywords contain different types of parameters: any of the words, all of the words, none of the words, and exact phrases.

- Any of the words: $A = \{a_1, a_2, a_3, ...a_{n_a}\}$, n_a is the number of terms.
- All of the words: $B = \{b_1, b_2, b_3, ...b_{n_b}\}$, n_b is the number of terms.
- None of the words: $C = \{c_1, c_2, c_3, ...c_{n_c}\}$, n_c is the number of terms.
- Exact phrases: $D = \{d_1, d_2, d_3, ...d_{n_d}\}$, n_d is the number of terms.

All these terms can be arranged in a Boolean formula. This Boolean formula can help *LiveWeb* store them and test easily. For any of the words, the Boolean formula will be $\bigcup_{i=0}^{n_a} a_i$. For all of the words, it will be $\bigcap_{i=0}^{n_b} b_i$. For none of the words, it will be $\bigcap_{i=0}^{n_c} c_i$. For exact phrases, it will be $\bigcap_{i=0}^{n_d} d_i$. In summary the final formula is

$$\bigcup_{i=0}^{n_a} a_i \bigcap \bigcap_{i=0}^{n_b} b_i \bigcap \bigcap_{i=0}^{n_d} d_i - \bigcup_{i=0}^{n_c} c_i .$$

Then we use indexing to these subscriptions based on the keywords. Like import indexing, each keyword, k_i, leads a list, list_k. Assuming there is n_k keywords, the subscription is appended to these lists: $\text{list}_1, \text{list}_2, ..., \text{list}_{n_k}$. Besides this, we have the category of subscription: cat, so this subscription will be appended to list_{cat}, which is led by the category. Query always includes the value range, min max. In *LiveWeb*, we build a *valuetree* to hold these subscriptions. Because there exists an uncountable value range, while there is limited queries, we need to store the ranges that users have uploaded. However, storing them naively cannot help them search quickly, so we store these value ranges in a tree: $\text{value}_{\text{tree}}$. There are two kinds of trees, min-tree and max-tree. Each value range can be separated into two parts: min and max. These two parts will be inserted into two different trees separately, as shown in Fig. 9. Figure 8 illustrates how a subscription is indexed.

5.2 Alert

Alert is the successor of a subscription. First, issue a list of subscription-based indices. Then use the user-preferred communication mode to alert them immediately. Section 5.1 discusses how to index subscriptions. In this section we will discuss how we distribute the alert information. Everytime new items come in, the items are labeled as $t_1, t_2, t_3, \cdots, t_n$. Each item has its keywords $k_1, k_2, k_3, \cdots, k_n$, To make an alert component work efficiently, we have indexed subscriptions, also called queries. With these keywords, we can extract the subscriptions led by these keywords: $\text{list}_{k_1}, \text{list}_{k_2}, \text{list}_{k_3}, \cdots, \text{list}_{k_n}$. Since each incoming item has its category, we can get another list: list_{cat}. The value, v, of the item, is used to generate another two lists. The subscriptions' value range should cover this value, which means min should be smaller than v

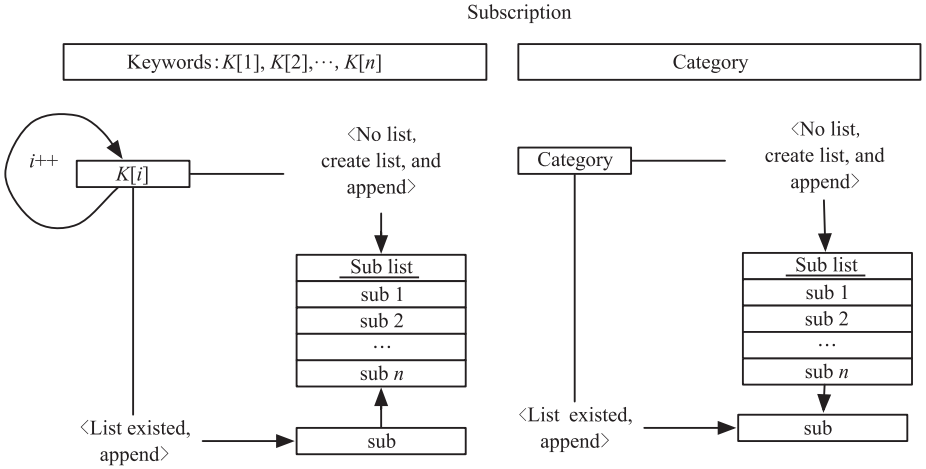


Fig. 8 Subscribe mechanism

and max should be larger than v. Therefore, in the value tree, we need first to find the branch with the root smaller than v in the min-tree and the branch with the root larger than v in the max-tree. This is shown in Fig. 9. Then get list_v. The subscriptions should be: $\bigcup_{i=0}^{n}\text{list}_{k_i} \bigcap \text{list}_{\text{cat}} \bigcap \text{list}_v$.

However, this is not enough. The subscriptions chosen are not all qualified because these subscriptions have other criteria which are not reflected in the indices. These much smaller subscriptions are tested one by one with the new incoming item. Then the final subscriptions set S are obtained. Figure 10 shows the work flow of the alert component. With subscriptions set S, we can deliver the new information to the users through alert interface.

The alert interface is designed for sending the alert message through the user preferred mode. Various communication modes have enabled people to be reached easily, such as Email, SMS, telephone call, wireless internet client, etc[18]. After it gets the subscriptions set, S, it can retrieve the user information based on these subscriptions. Most users have their contact information including cellphone or email address. *LiveWeb* reorganizes the subscriptions belonging to one user, then sends these alert messages.

6 Case Studies

LiveWeb is a new sensorweb portal to bridge varied sources of sensor data and different public communities. Besides mechanisms behind the portal, *LiveWeb* is also designed to be easy to use. The following describes some case studies and some useful tips on *LiveWeb*.

6.1 Motion detector

To evaluate *LiveWeb*'s performance, a network of TelosW[19] sensor motes is used to set up the testbed. We used 8 TelosW motes with motion sensors to test activity in a $15\,\text{m}\times 8\,\text{m}$ class room and a $6\,\text{m}\times 8\,\text{m}$

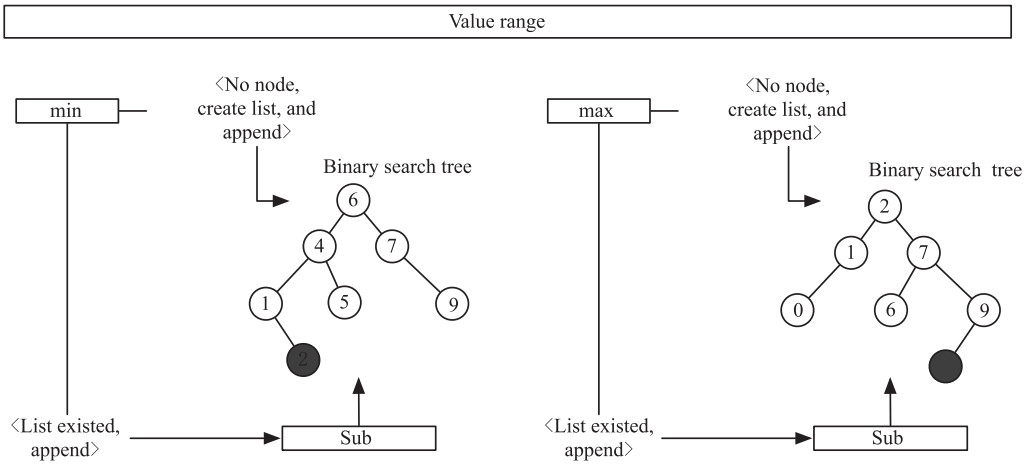


Fig. 9 Value range indexing

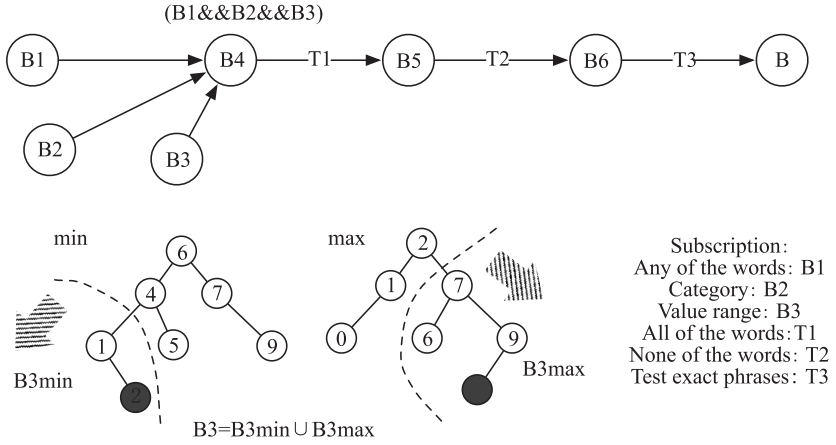


Fig. 10 Alert mechanism

lab room. Figure 11a illustrates the deployments of these motes. In the application, only if there is an activity, the sensor mote can be awakened to send its message. *LiveWeb* lists the eight sensors in one page and monitors their activities. When one person goes through them along the route labeled as in Fig. 11b, the waveform of these eight sensors can show up in the corresponding motion information in *LiveWeb*. The following real-time chart in Fig. 12 indicates when *motion sensor* 97 detected activities. Users can monitor the activity in this room remotely.

As discussed in Section 5, *LiveWeb* provides alert service to users. To monitor the activities of these two rooms offline, what users need to do is to search them first and add them into the monitor list. *Motion sensor* 97 is added into the monitor list, and the value range is set as 1. When this sensor detects motion, users can be alerted. This is an example of how *LiveWeb* bridges a sensor network and people's daily lives.

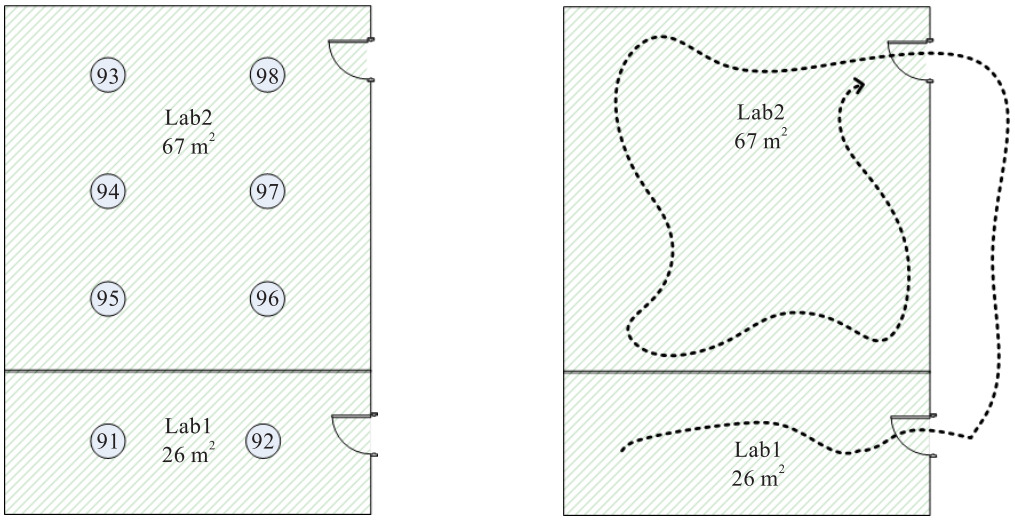
6.2 Sensing market

Geographic information technology has matured. However, as it has developed, the concept of the sensor has changed. The definition of a sensor is an open issue. The essential part of a sensor is detecting events in unattended environments. If there is something that can detect an event, that is a sensor. Briefly, anything which detects anything changing can be treated as a sensor. In this way, what *LiveWeb* can do is serve not only as a portal for geographic information, but also as a portal for a changing world. News, supermarkets, and job occupations, can be imported to *LiveWeb* to make people's lives easier. Currently, *LiveWeb* imported exchange information from some famous supermarkets and exchange websites. If new products come in, *LiveWeb* will alert users whose criteria the products satisfy. With a *LiveWeb* real-time chart, users can see the products' history of prices and predict their trends. Figure 13 illustrates this scenario.

The housing information in Vancouver, WA is a "sensor". Looking for an apartment before the semester is a headache for most students in college. Rent information available from the Internet helps them a lot. However, that is still not enough. Busy students are distracted from work to check the information every day, or even every hour. With *LiveWeb*, users can rely upon it for performing this job. Here we are looking for an apartment with one bedroom, whose price is lower than \$600, in Vancouver, WA. First we query "house one bedroom Washington Vancouver max=600". Then we just need to add it to the monitor list of our account and choose a preferred alert method. The second day, we receive the email telling us some one has published an apartment with one bedroom close to I5 for \$545 a month. In this way, *LiveWeb* relieves people from exhausted searching and manual monitoring, thus making their life easy.

7 Conclusion and Future Work

In this papar, we have presented a sensorweb portal with real-time search, monitor, and notification



(a) (b)

Fig. 11 The layout of motion sensors and the path of movement

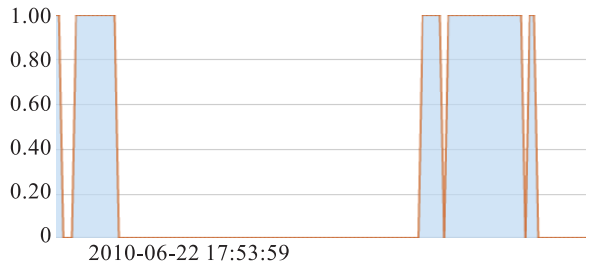


Fig. 12 Sensor 97 detects motion.

functions. We call it *LiveWeb*. Cloud computing is replacing traditional computing. More and more applications are moving into the clouds partly or completely. Google Doc is already a classical web application. People don't need to install those big size applications on each computer. Also they don't need to use removable devices to transfer files here and there. Integrating sensor networks with cloud computing technology is very promising[20]. *LiveWeb* steps into the trend of cloud computing. In conclusion, *LiveWeb* has the following beneficial properties: (1) The system enables sensorweb service accessible from anywhere. (2) The system makes sensor network data readable by anyone. (3) Sensor network sharing data becomes much easier because of *LiveWeb*. (4) The system make sensor data format transparent to data users. (5) Real-time data display suits sensor data properties. (6) An offline alert system strengthens real-time features. A possible extension in *LiveWeb* is the research about security to protect sensor publisher's privacy and priorities.

References

[1] Botts M, Percivall G, Reed C, et al. OGC sensor web enablement: Overview and high level architecture. *GeoSensor Networks*, 2008, **4540**: 175-190.

[2] Nath S, Liu Jie, Zhao Feng. SensorMap for wide-area sensor webs. *Computer*, 2007, **40**(7): 90-93.

[3] Google powermeter. http://www.google.com/powermeter, 2009.

[4] Chen Chao, Helal A. Device integration in soda using the device description language. In: The 9th Annual International Symposium on Application and the Internet (SAINT 2009). Bellevue, WA, USA, 2009: 100-106.

[5] Hunkeler U, Truong H L, Stanford-Clark A. MQTT-S a publish/subscribe protocol forwireless sensor networks. In: 2nd Workshop on Information Assurance for Middleware Communications. Bangalore, India, 2008: 791-798.

[6] Hashmi N, Myung D, Gaynor M, et al. A sensor-based, web service-enabled, emergency medical response system. In: Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services (EESR'05). Seattle, WA, USA, 2005: 25-29.

[7] Wei Bin, Renger B, Chen Y F, et al. MediaAlert: A broadcast video monitoring and alerting system for mobile users.

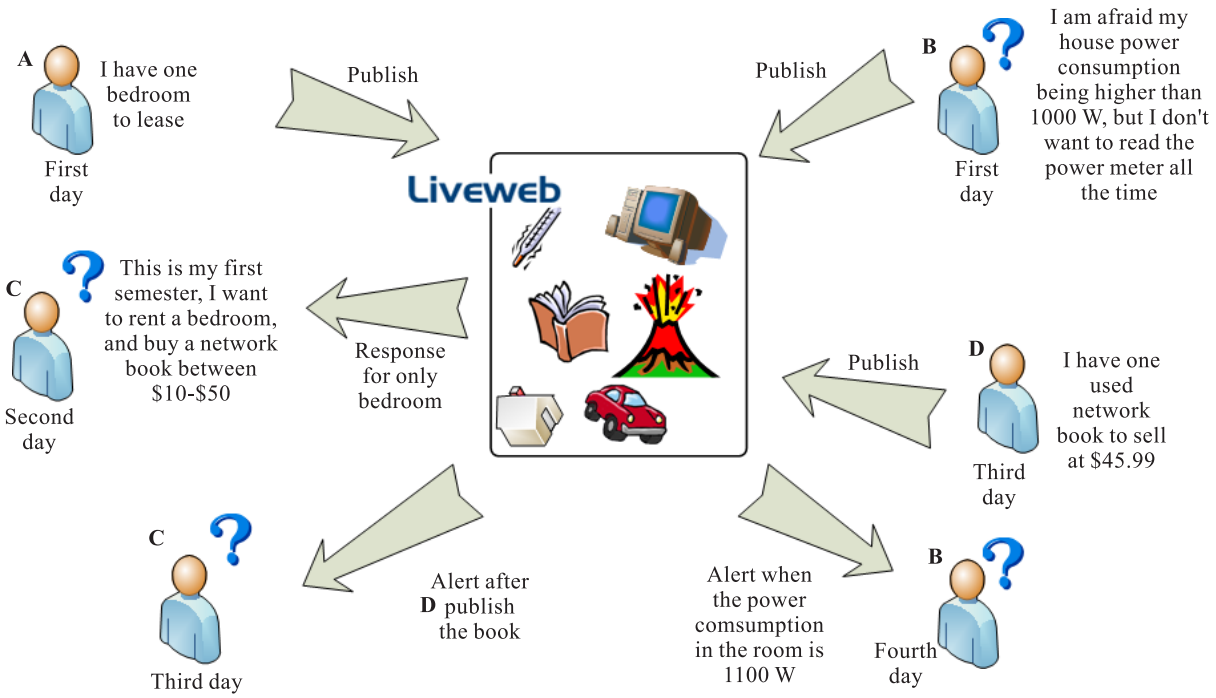


Fig. 13 *LiveWeb* helps exchange in real-time.

In: International Conference on Mobile Systems, Application, and Services. Seattle, USA, 2005: 25-38.

[8] Li Guoli, Jacobsen H A. Composite subscriptions in content-based publish/subscribe systems. In: Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware. New York, USA, 2005.

[9] Farshchi S, Nuyujukian P H, Pesterev A, et al. A tinyos-based wireless neural sensing, archiving, and hosting system. In: 2nd International IEEE EMBS Conference on Neural Engineering. Arlington, VA, USA, 2005: 671-674.

[10] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data. In: Operating Systems Design and Implementation. Seattle, WA, USA, 2006: 205-218.

[11] Irmak U, Mihaylov S, Suel T, et al. Efficient query subscription processing for prospective search engines. In: USENIX. Boston, MA, USA, 2006: 375-380.

[12] Zhu Yingwu, Shen Haiying. An efficient and scalable framework for content-based publish/subscribe systems. *Peer-to-Peer Networking and Applications*, 2008, **1**(1): 3-17.

[13] Porter M F. An Algorithm for Suffix Stripping. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1980: 130-137.

[14] Ji Shengyue, Li Guoliang, Li Chen, et al. Efficient interactive fuzzy keyword search. In: Proceedings of the 18th International Conference on WWW. Madrid, Spain, 2009.

[15] Yue Chuan, Chu Zi, Wang Haining. RCB: A simple and practical framework for real-time collaborative browsing. In: Proceedings of the 2009 Conference on USENIX Annual Technical Conference. San Diego, CA, USA, 2009.

[16] Hardtke D, Wertheim M, Cramer M. Demonstration of improved search result relevancy using realtime implicit relevance feedback. In: SIGIR. Boston, USA, 2009.

[17] Dork M, Williamson C, Carpendale S. Towards visual web search: Interactive query formulation and search result visualization. In: WSSP. Madrid, Spain, 2009.

[18] Cui Yanqing, Roto V. How people use the web on mobile devices. In: WWW. Beijing, China, 2008: 905-914.

[19] Lu Gang, De D, Xu Mingsen, et al. TelosW: Enabling ultra-low power wake-on sensor network. In: 7th International Conference on Networked Sensing Systems (INSS'10). Kassel, Germany, 2010.

[20] Lim H B. Sensor cloud: Towards sensor-enabled cloud services. http://www.ntu.edu.sg/intellisys, 2009.