Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 4287834, 8 pages
http://dx.doi.org/10.1155/2016/4287834

Research Article

A Stream Processing System for Multisource Heterogeneous Sensor Data

Liang Hu, Rui Sun, Feng Wang, Xiuhong Fei, and Kuo Zhao

College of Computer Science and Technology, Jilin University, Changchun 130012, China

Correspondence should be addressed to Kuo Zhao; zhaokuo@jlu.edu.cn

Received 21 December 2015; Revised 17 February 2016; Accepted 22 February 2016

Academic Editor: Liling Fu

With the rapid development of the Internet of Things (IoT), a variety of sensor data are generated around everyone's life. New research perspective regarding the streaming sensor data processing of the IoT has been raised as a hot research topic that is precisely the theme of this paper. Our study serves to provide guidance regarding the practical aspects of the IoT. Such guidance is rarely mentioned in the current research in which the focus has been more on theory and less on issues describing how to set up a practical system. In our study, we employ numerous open source projects to establish a distributed real time system to process streaming data of the IoT. Two urgent issues have been solved in our study that are (1) multisource heterogeneous sensor data integration and (2) processing streaming sensor data in real time manner with low latency. Furthermore, we set up a real time system to process streaming heterogeneous sensor data from multiple sources with low latency. Our tests are performed using field test data derived from environmental monitoring sensor data collected from indoor environment for system validation. The results show that our proposed system is valid and efficient for multisource heterogeneous sensor data integration and streaming data processing in real time manner.

1. Introduction

With the highly rapid development of networks, IoT technologies, and web related technologies in recent years, massive data service applications are developed. And these systems live in independent, distributed, and heterogeneous environment, respectively, on the web or offline. In the traditional situations, different departments of an enterprise or government have independent information systems, which usually store information about personnel, products, or other things to ease operation of the department. However, due to lack of comprehensive design and inability to foresee future development, it is often hard to share data among systems belonging to different departments. It is also a common problem in scientific research, especially when interdisciplinary study is inevitable where interoperability among independent scientific data management systems is the use case.

Compared to these conventional database systems, a mass of IoT applications arise to collect, unify, share, and publish sensory data. Mobile Crowd Sensing (MCS) [1] is the typical situation in recent hot research fields, in which domain it is often required to acquire, integrate, and further process data from wireless sensor networks, traditional databases, configuration files, online data services, and other potential heterogeneous sources.

The integration of heterogeneous data is not a new problem; despite the semantic web approach dedicated to data representation perspective, people pay enough attention to methodologies to query heterogeneous data sources [2]. A normal methodology is Wrapper-Mediator [3], where mediator is responsible for handling user's request and providing a unified data accessing view, and each wrapper is associated with each data source, mediator transforms user's query into wrappers' query, respectively, and wrappers interact with concrete data sources to mask heterogeneity. In our study, Global Sensor Network (GSN) is just the IoT middleware realizing such methodology. GSN abstracts the mediator concept as virtual sensor, which serves as an integrated data view that users can query with unified SQL query sentences. The abstraction wrapper inside GSN handles the actual complexity and heterogeneity of data sources.

These years have witnessed the stunningly growing number of sensors for people's everyday life like smart phones, wearable devices, wireless sensor networks, and social networks (they are kind of virtual sensors). These sensors are around us and sense the environment we live in and in order to provide more contextual and more accurate services they produce data all the time. In the meantime, the data overload problem has emerged in insular system and leads to inability to effectively utilize real time information, however.

In our research, we aim to provide IoT application architecture paradigm that comprises procedures spanning data acquisition, integration, and utilization. As mentioned before, in modern IoT data process applications, there are two common and fundamental challenges, integration of heterogeneous data and manipulating the data in real time manner. For realistic IoT applications usually collecting data from multiple heterogeneous sources, we use GSN to handle this problem, and, after the acquisition and integration procedures, we program to populate data into Apache Storm system. Apache Storm is a general purpose real time distributed data processing system in contrast to Hadoop, which features its distributed batch processing ability.

The motivation of this paper is to provide guidance with respect to practical engineering aspects of the IoT. There is a gap in current literature, where researchers concentrate more on solving theoretical questions and less on describing how to establish a practical system. The rest of the paper is organized as follows. In Section 2, the development of IoT as well as middleware atop IoT is reviewed. Section 3 depicts the architecture of the system. Section 4 details the concrete implementation of the system. In Section 5, we conclude our current work and furthermore point out our future avenues of research.

2. Related Work

The IoT would become a basic infrastructure in society just like highway, water supply, and network. Beyond the current network, the IoT is more than a revolution of the Internet as it extends human beings' sense and feeling of the physical world; it is a dynamic autonomous network based on its internal communication protocols. In such network, all the physical and virtual items are identified by global unique identification and communicate sensory data with each other to achieve information sharing through intelligent interfaces [4, 5]. These intelligent interfaces connect and communicate with users, society, and environment context on the basis of the agreed protocols. IoT is an extension and expansion of the network based on the Internet to achieve intelligent identifying, locating, tracking, monitoring, and managing.

As IoT draws extensive attention, a great number of researches are focused on the study of IoT enabling technologies, comprising identification, sensing, communication, and middleware [4, 6]. And to ease the process of IoT application development, middleware for IoT is highly emphasized and is becoming a widely employed approach [7–12]. Middleware of IoT bridges the wide gap between physical things and high-level applications and furthermore provides some dedicatedly designed services for one or more clients to employ locally or remotely through network connection. The implementation technologies of middleware comprise agent-based methods [13], context awareness methods [12], service oriented architecture (SOA) based [14] methods, and other development paradigms.

Several studies have focused more on wireless sensor networks. In [10], the authors described the outlook of middleware for wireless sensor networks. Authors in [15] designed and implemented a lightweight middleware platform for distributed computation on wireless sensor networks. As wireless sensor network is becoming a new paradigm [16] in the realm of Internet of Everything, some studies have been made on facilities virtualization. In [16], authors bring an overview on the research of WSN virtualization by middleware and discuss the design goals, system architecture, service qualities, and challenges of WSN virtualization.

Data processing is always a critical issue in the domain of IoT. Scientific data processing is more specific but somehow distinguished because large-scale scientific data are typically written into parallel file systems for fast writing speed, which, in turn, leads to poor I/O performance while reading [17]. Furthermore, the generic operation of collecting data from heterogeneous data sources as well as joining the data also makes the reading and analysis process time-consuming [18]. And, to solve the problems, a variety of applications [17–20] are developed in recent years.

However, beyond what people have achieved, we are also confronted with quite a few issues in the frontier of IoT, which concerns standardization, network addressing, security, privacy, scalability, mobility, heterogeneity abstraction, and some other problems [4, 6, 21].

Recently some new research hotspots arise. As cloud computing is highlighted in both academics and industry, cloud-centric IoT finds its way into mainstream. Reference [22] discusses the realization and challenges in cloud-centric IoT. Authors in [23] make an investigation on data mining for IoT. Context aware computing also enters into the realm of IoT [24]. Semantic or ontology related approach is utilized in IoT as well [25, 26].

Despite the challenges, IoT based applications have emerged in many domains over the years, such as healthcare [27], agriculture management [26, 28, 29], social relations study [30], context aware computing [24, 31], smart city [32, 33], and smart house [34].

3. System Architecture

Our proposed system architecture consists of three layers as Figure 1. At the bottom are physical sensors. In our case, we build a wireless sensor network using the two different types of wireless sensor nodes and a programing board directly connected to a PC through USB cable. Then, the serial port data is listened to by GSN (Global Sensor Network), which is going to receive the sensor data with further aggregation and filtration. GSN can act as a general purpose sensor middleware capable of accepting heterogeneous data from a variety of sensors, particularly including virtual sensors

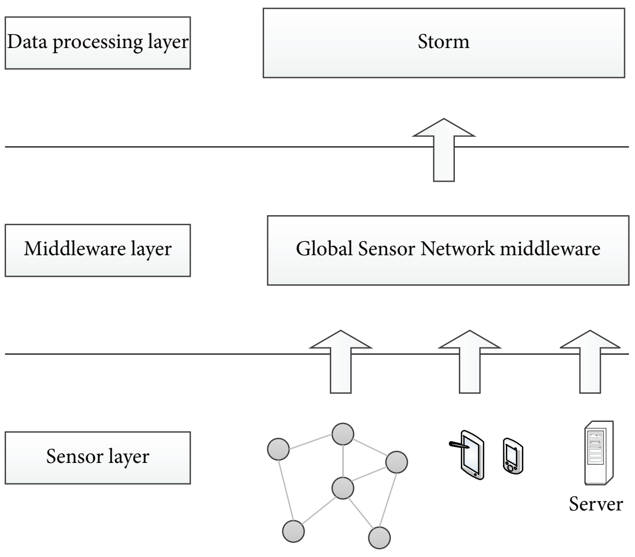


Figure 1: System architecture.

(simulated). It can intrinsically aggregate and filter data through SQL-format queries. GSN as a sensor data server also provides several ways of publishing data. Through the APIs of GSN, spouts (one type of component of Storm) can absorb data into the distributed computing system.

As GSN itself has already provided convenient APIs for data access, applications can retrieve sensor data directly from GSN and apply some procession according to its application logic. But we add a data procession layer above the middleware layer. One of the most important reasons is that sometimes complex computation may be applied on the aggregated sensor data. As can be seen, sensors in GSN are virtual sensors in GSN's abstractions; they can be related to real physical sensors or something unreal indicating a virtual sensor, such as an email notifier reporting email content whenever a new email is received or even just a generator generating random integers periodically. A sensor featured its ability of producing data; in this sense, anything particularly software artifacts producing data like stream APIs of Twitter can be virtual sensors. And data format could be diverse, such as binary, double, strings, and big images.

Therefore, data is multisource and heterogeneous.

Computation applied on this multisource heterogeneous data could be quite complex, requiring powerful computing capability. Moreover, in modern IoT applications, the data are usually real time data stream and computation employed on the stream is also required to be real time to lower processing latency and provide better user experience. These factors make a real time distributed data processing layer a necessity.

4. Implementation of the System

4.1. Brief Introduction to GSN. In the past, people may build their applications directly on top of sensor networks, meaning the embedded operating systems. However, with the emergence of numerous heterogeneous sensor networks and other sensor forms, it became a tedious task to interact with the sensor systems.

Another challenge arises as the IoT industry grows rapidly around the globe that the sensor infrastructures belonging to different organizations cannot be adequately employed in the process of data integration of a wide area or even throughout the globe [35].

And here the Global Sensor Networks (GSNs) middleware and other similar middleware emerge in response to that hard condition in proper time, which lead to the new programing paradigm as the middleware interact with the sensor networks and are responsible for gathering data, with which the applications interact.

GSN is one of those middleware. Virtual sensor is a key abstraction in GSN [35]. GSN can be plugged in with arbitrary sensors as you will. The sensor presentation in GSN is XML file abstracted as virtual sensor, which defines data streams as sources and queries on the streams. Another closely related abstraction is wrapper. A wrapper is a Java class, which is truly interacting with data source, like physical sensor networks, sockets, web services, and so on. Data streams defined in virtual sensor must specify data sources and wrappers.

As you can see, one virtual sensor can contain multiple data streams and SQL-format queries, meaning that it is convenient to implement data integration among various data sources.

Writing wrappers by yourself could still be a tedious task; thereby, GSN has shipped with massive common wrappers. Most of the time, you do not have to write one. There are varieties of wrappers, such as TinyOS [36] wrapper and JDBC wrapper. A specific wrapper is remote wrapper, and it is able to retrieve data from a remote virtual sensor in another GSN server on the Internet. In this way, GSNs around the globe can be interconnected, and the sensors data gathered by them can be taken full advantage from.

As IoT middleware, GSN designed easy-access APIs for feeding data into applications. There are four types of APIs [37]: connection oriented data distributor, connectionless data distributor, web service, and restful APIs. All are easy to retrieve real time data or historical data in a time window in compliance with GSN's time model.

4.2. Brief Introduction of Apache Storm. Apache Storm is a free and open source distributed streaming data processing system. It is intended for real time data processing in contrast to batch data processing as Hadoop. Originated in 2011, now Storm has been used in wide domains. Yahoo, Twitter, Spotify, Flipboard, Alibaba, Baidu, and many other enterprises have been taking advantage of it.

Storm defines how we should write distributed programs since it provides a simple and efficient programing paradigm. It defines basic programing components, "spout" and "bolt." "Spout" acts as water source; it generates data in some way and pushes the data out as tuples periodically. "Bolt" is component that takes in tuples from "spouts" or other "bolts" and employ some computation and then might produce new tuples and push them out. Programers need to implement

Figure 2: Wireless sensor nodes.

Figure 3: A wireless sensor node.

their own "spouts" and "bolts" and specify how these components are connected, indicating how data flows in the system. Such a data flow graph consisting of "spouts" and "bolts" is called "Topology," another key abstraction in Storm.

In fact, "Topology" is a job to process on the Storm cluster. In order to exploit the potential of distributed computing, a group of computers that installed Apache Storm needs to be connected through local network, forming a Storm cluster. The distributed program in compliance with Storm's programing paradigm is usually packaged into a runnable jar file, which will be submitted to Storm cluster afterwards. This submitted program running on Storm cluster is a Storm job or "Topology" in Storm's terminology. Different from batch jobs, jobs running on Storm cluster will usually not stop, because they are meant to execute real time tasks. Several jobs can run on Storm cluster simultaneously. Storm programs are robust, for its feature of parallelization, partition, and retrying on failure when necessary.

The basic components spout and bolt are inherently paralleled; you can specify parallel number in Topology settings. Even after the Topology has been submitted and running, you can still adjust parallelism of every component according to hardware conditions. Playing a role of "Hadoop of real time," Storm greatly eases the progress of implementing parallel real time computation.

4.3. Detailed Implementation of the System. In order to establish the environmental monitor system, in sensor layer, we deployed IRIS sensor nodes in our laboratory in a number of different positions. The nodes are displayed in Figures 2 and 3. All these sensor nodes comprise voltage sensor, humid sensor, heat sensor, press sensor, optical sensor, and accelerator sensor of three axes. Even though the hardware

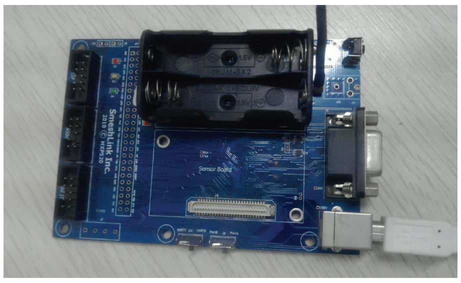


Figure 4: Programing board.

of the nodes is the same, they are divided into two groups according to two different embedded applications installed on them. One is sht11, and the other is mxp430. The symbols sht11 and mxp430 are two types of sensors chips, respectively, and in this article we use them to denote the corresponding sensor nodes and the embedded applications running on them. The mxp430 nodes collect all types of sensor data mentioned above, while the sht11 nodes only sense light, temperature, and voltage of the battery equipped with them. Both of the two types of nodes also provide information of the network Topology comprising sensor node id, parent node id, and group id as well as metadata of the message as timestamp and message sequence id. These sensor nodes form a wireless sensor network, whose sink node is a sensor node plugged on a programming board (Figure 4) connected to a Linux server using USB adapter cable. These sensor nodes use TinyOS [36] as their embedded operating system and run TinyOS application for collecting environmental data.

The Linux server is installed with Ubuntu 14.04.1 LTS, along with Global Sensor Network middleware. The raw data collected from wireless sensor network will be interpreted, stored, and integrated for further usage.

Wrapper inside GSN is actually where data is acquired from an outside source. Wrappers are like borders, through which data enter into the realm of GSN. Therefore, obtaining a proper wrapper is put in the first place. Usually, GSN has already shipped with a great amount of wrappers, but in our situation the raw data format is customized by the hardware product provider; therefore we write our own wrappers. The raw sensor data is firstly collected and interpreted by a sensor network server and then gets pushed out in a well-defined XML style through sockets. So the wrappers we customized are actually for TCP connection and XML parsing. In the initialization phase, the wrappers retrieve the parameters they need, such as host address and port, and then decide the output structure to persist, which is corresponding to the table structure to be stored in the database. Then, in the work thread of the wrappers, firstly they connect to the sensor network server and, secondly, in a loop they read XML-formatted data and parse it and store parsed data into a database. Both of the wrappers of sht11 and mxp430 sensor data work in this manner.

After the wrappers have been prepared, the XML files of virtual sensors are required. Box 1 shows the virtual sensor for sht11 message.

```
<virtual-sensor name="smesh sht11" priority="10">
  <processing-class>
    <class-name>gsn.vsensor.BridgeVirtualSensor</class-name>
    <init-params />
    <output-structure>
      <field name="nodeid" type="double"/>
      <field name="parent" type="double"/>
      <field name="seqid" type="double"/>
      <field name="groupid" type="double"/>
      <field name="board_id" type="double"/>
      <field name="light" type="double"/>
      <field name="temp" type="double"/>
      <field name="voltage" type="double"/>
    </output-structure>
  </processing-class>
  <description>This sensor gets xml data from smesh server
  </description>
  <life-cycle pool-size="10" />
  <addressing>
    <predicate key="geographical">Sensor 114 @ EPFL</predicate>
    <predicate key="LATITUDE">46.520000</predicate>
    <predicate key="LONGITUDE">6.565000</predicate>
  </addressing>
  <storage history-size="5m" />
  <streams>
    <stream name="input1">
      <source alias="source1" sampling-rate="1" storage-size="1">
        <address wrapper="sht11">
          <predicate key="rate">500</predicate>
          <predicate key="host">192.168.1.135</predicate>
          <predicate key="port">9005</predicate>
        </address>
        <query>SELECT nodeid,parent,seqid,groupid,board_id,light,temp,voltage
            FROM wrapper</query>
      </source>
      <query>SELECT nodeid,parent,seqid,groupid,board_id,light,temp,voltage
           FROM source1</query>
    </stream>
  </streams>
</virtual-sensor>
```

Box 1: XML file representing a virtual sensor.

Up to now, data is under the management of GSN. Next step is to connect GSN to Storm. As GSN provides http stream data API, we implement the Storm spout in the way the wrapper captures data.

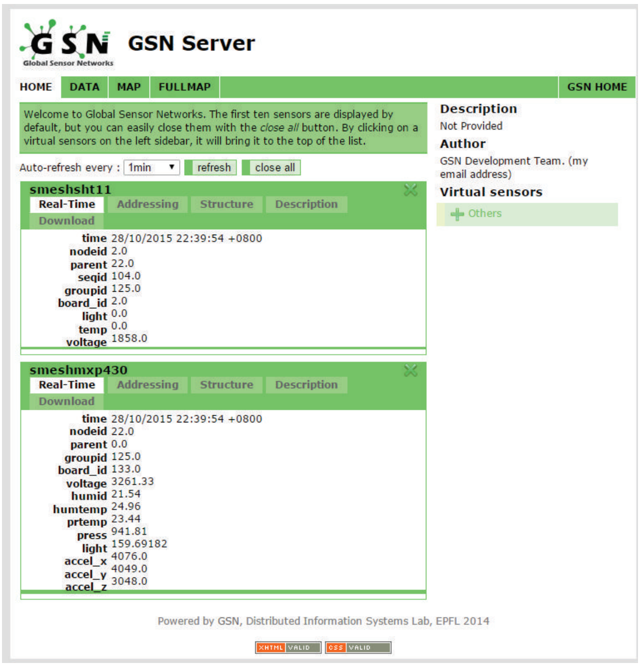
4.4. Evaluation. Deployed as mentioned above, the system can successfully capture data from IRIS sensor nodes. Table 1 shows the original sensory data, coming from mxp430 sensor nodes, and original sensory data of sht11 sensor nodes is similar. Figure 5 exhibits the data acquired by GSN server, which is accomplished by customized wrappers and well-defined virtual sensors. Now the sensor data flows into middleware layer. Finally, through the data access component "spout" in Storm Topology, data is retrieved from GSN into Storm cluster. The spout can successfully parse received data as shown as in Figure 6. As testified by practice, our proposed system architecture is feasible for IoT data acquisition, integration, and real time distributed procession.

5. Conclusion and Future Work

In this paper, we propose and discuss system architecture for IoT stream data processing. We basically divide the system architecture into three different layers, sensor layer, middleware layer, and data processing layer from bottom to top. And we implement the system and conduct the experiment using some popular open source projects. In the sensor layer, TinyOS is the embedded operating system

Table 1: Captured sensor data of mxp430 nodes.

nodeid	Parent	Group	board_id	Voltage	Humid	humtemp	prtemp	Press	Light	accel_x	accel_y	accel_z
22	0	125	133	3261.33	22.38	24.88	23.39	941.91	159.69182	3840	4063	3060
21	**3**	**125**	**133**	**2509.72**	**−4.65**	**−39.6**	**23.34**	**933.34**	**141.45**	**48**	**4032**	**3077**
22	0	125	133	3261.33	22.59	24.74	23.25	941.93	166.463	8	4072	3061
21	3	125	133	2509.72	−4.65	−39.6	23.35	933.36	141.45	25	4026	3325
22	0	125	133	3261.33	22.67	24.7	23.24	941.91	166.463	247	4066	3043
21	3	125	133	2509.72	−4.65	−39.6	23.31	933.38	141.45	35	4026	3310
22	0	125	133	3261.33	22.67	24.75	23.31	941.94	159.69182	13	4072	2816
21	3	125	133	2504.7	−4.65	−39.6	23.26	935.93	141.45	33	4037	3070
22	0	125	133	3261.33	22.67	24.75	23.32	941.86	159.69182	26	4055	3054
21	3	125	133	2509.72	−4.65	−39.6	23.29	936.64	141.45	33	4026	3054
22	0	125	133	3261.33	22.67	24.78	23.34	941.91	166.463	1	4073	3056
21	3	125	133	2509.72	−4.65	−39.6	23.25	933.17	141.45	32	4037	3064
22	0	125	133	3261.33	22.63	24.74	23.32	941.86	159.69182	12	4056	3047
21	3	125	133	2509.72	−4.65	−39.6	23.2	933.35	141.45	34	4025	3046
22	0	125	133	3261.33	22.6	24.81	23.35	941.84	159.69182	3843	4065	3056
21	3	125	133	2509.72	−4.65	−39.6	23.27	935.13	141.45	39	4030	3057
22	0	125	133	3261.33	22.53	24.82	23.38	941.88	159.69182	3848	4054	3061
21	3	125	133	2504.7	−4.65	−39.6	23.29	933.26	141.45	33	4027	3065
22	0	125	133	3261.33	22.53	24.86	23.41	941.84	159.69182	4095	4063	3050



GSN Server

Welcome to Global Sensor Networks. The first ten sensors are displayed by default, but you can easily close them with the *close all* button. By clicking on a virtual sensors on the left sidebar, it will bring it to the top of the list.

Auto-refresh every : 1min

smeshsht11

```
    time 28/10/2015 22:39:54 +0800
  nodeid 2.0
  parent 22.0
   seqid 104.0
 groupid 125.0
board_id 2.0
   light 0.0
    temp 0.0
 voltage 1858.0
```

smeshmxp430

```
    time 28/10/2015 22:39:54 +0800
  nodeid 22.0
  parent 0.0
 groupid 125.0
board_id 133.0
 voltage 3261.33
   humid 21.54
 humtemp 24.96
  prtemp 23.44
   press 941.81
   light 159.69182
 accel_x 4076.0
 accel_y 4049.0
 accel_z 3048.0
```

Description
Not Provided
Author
GSN Development Team. (my email address)
Virtual sensors

Powered by GSN, Distributed Information Systems Lab, EPFL 2014

Figure 5: Web page exhibiting captured sensor data in GSN.

on the wireless sensor node; in the middleware layer, the Global Sensor Network functions as sensor data server; in the data processing layer, the Apache Storm is responsible for distributed real time sensor data processing. In the construction of the system, data input and output of GSN are key procedures, and the wrappers in GSN and spout in Apache Storm need to be developed manually. Both the wrappers and the spout are responsible for retrieving data out of another system by socket connection.

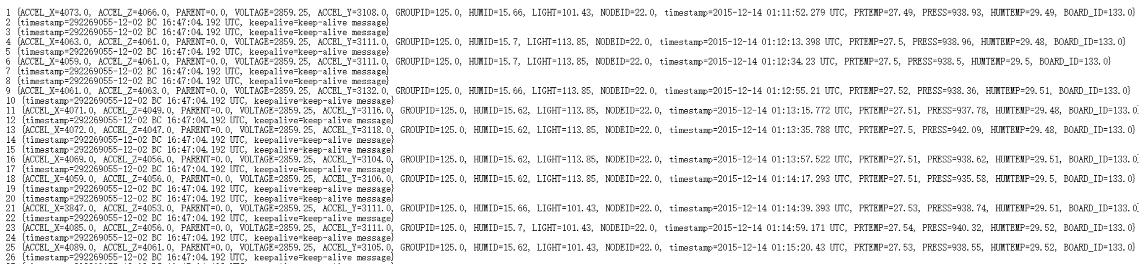
The benefit of the architecture is apparent. It is capable of retrieving and integrating multisource heterogeneous data and finally processing it in a distributed and real time manner. Our future work aims at constructing a context aware recommender system. As can be seen from this paper, we can capture information from wireless sensor networks, mobile phones, and other equipment and then combine physical data like weather and position with social networks which might indicate an individual's activity and mood and other things. Through the individual oriented information, we are focusing on establishing a real time context aware recommender system.

Competing Interests

The authors declare that there are no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is funded by European Framework Program (FP7) under Grant no. FP7-PEOPLE-2011-IRSES, National Natural Science Foundation of China under Grant nos. 61073009 and 61103197, National High Tech R&D Program 863 of China



```
{ACCEL_X=4073.0, ACCEL_Z=4066.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3108.0, GROUPID=125.0, HUMID=15.66, LIGHT=101.43, NODEID=22.0, timestamp=2015-12-14 01:11:52.279 UTC, PRTEMP=27.49, PRESS=938.93, HUMTEMP=29.49, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4063.0, ACCEL_Z=4061.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3111.0, GROUPID=125.0, HUMID=15.7, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:12:13.398 UTC, PRTEMP=27.5, PRESS=938.96, HUMTEMP=29.48, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4059.0, ACCEL_Z=4061.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3111.0, GROUPID=125.0, HUMID=15.7, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:12:34.23 UTC, PRTEMP=27.5, PRESS=938.5, HUMTEMP=29.5, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4061.0, ACCEL_Z=4063.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3132.0, GROUPID=125.0, HUMID=15.66, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:12:55.21 UTC, PRTEMP=27.52, PRESS=938.36, HUMTEMP=29.51, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4071.0, ACCEL_Z=4049.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3116.0, GROUPID=125.0, HUMID=15.62, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:13:15.772 UTC, PRTEMP=27.51, PRESS=937.78, HUMTEMP=29.48, BOARD_ID=133.0
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4072.0, ACCEL_Z=4047.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3118.0, GROUPID=125.0, HUMID=15.62, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:13:35.788 UTC, PRTEMP=27.5, PRESS=942.09, HUMTEMP=29.48, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4069.0, ACCEL_Z=4056.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3104.0, GROUPID=125.0, HUMID=15.62, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:13:57.522 UTC, PRTEMP=27.51, PRESS=938.62, HUMTEMP=29.51, BOARD_ID=133.0
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4059.0, ACCEL_Z=4056.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3106.0, GROUPID=125.0, HUMID=15.62, LIGHT=113.85, NODEID=22.0, timestamp=2015-12-14 01:14:17.293 UTC, PRTEMP=27.51, PRESS=935.58, HUMTEMP=29.5, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=3847.0, ACCEL_Z=4053.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3111.0, GROUPID=125.0, HUMID=15.66, LIGHT=101.43, NODEID=22.0, timestamp=2015-12-14 01:14:39.393 UTC, PRTEMP=27.53, PRESS=938.74, HUMTEMP=29.51, BOARD_ID=133.0
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4085.0, ACCEL_Z=4056.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3111.0, GROUPID=125.0, HUMID=15.7, LIGHT=101.43, NODEID=22.0, timestamp=2015-12-14 01:14:59.171 UTC, PRTEMP=27.54, PRESS=940.32, HUMTEMP=29.52, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
{ACCEL_X=4089.0, ACCEL_Z=4061.0, PARENT=0.0, VOLTAGE=2859.25, ACCEL_Y=3105.0, GROUPID=125.0, HUMID=15.62, LIGHT=101.43, NODEID=22.0, timestamp=2015-12-14 01:15:20.43 UTC, PRTEMP=27.53, PRESS=938.55, HUMTEMP=29.52, BOARD_ID=133.0}
{timestamp=292269055-12-02 BC 16:47:04.192 UTC, keepalive=keep-alive message}
```

Figure 6: Data parsed by parser of spout.

under Grant no. 2011AA010101, National Sci-Tech Support Plan of China under Grant no. 2014BAH02F03, National Sci-Tech Major Projects of China under Grant nos. SinoProbe-09-01-03 and 2012ZX01039-004-04-3, Key Sci-Tech Program of Jilin Province of China under Grant nos. 2011ZDGG007 and 20150204035GX, and Fundamental Research Funds for Central Universities of China under Grant nos. JCKY-QKJC46 and 2412015KJ005.

References

[1] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, no. 1, article 7, 2015.

[2] X. Fengguang, H. Xie, and K. Liqun, "Research and implementation of heterogeneous data integration based on XML," in *Proceedings of the 9th International Conference on Electronic Measurement and Instruments (ICEMI '09)*, pp. 4711–4715, IEEE, Beijing, China, August 2009.

[3] G. Wiederhold, "Mediators in the architecture of future information systems," *Computer*, vol. 25, no. 3, pp. 38–49, 1992.

[4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] L. D. Xu, W. He, and S. Li, "Internet of things in industries: a survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[6] D. Bandyopadhyay and J. Sen, "Internet of things: applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.

[7] F. Wang, L. Hu, J. Zhou, Y. Wu, J. Hu, and K. Zhao, "Software toolkits: practical aspects of the internet of things—a survey," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 534378, 9 pages, 2015.

[8] G. Fersi, "Middleware for internet of things: a study," in *Proceedings of the 11th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '15)*, pp. 230–235, Fortaleza, Brazil, June 2015.

[9] S. Hadim and N. Mohamed, "Middleware: middleware challenges and approaches for wireless sensor networks," *IEEE Distributed Systems Online*, vol. 7, no. 3, pp. 1–23, 2006.

[10] L. Mottola and G. P. Picco, "Middleware for wireless sensor networks: an outlook," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 31–39, 2012.

[11] M. M. Wang, J. N. Cao, J. Li, and S. K. Dasi, "Middleware for wireless sensor networks: a survey," *Journal of Computer Science and Technology*, vol. 23, no. 3, pp. 305–326, 2008.

[12] G. Q. Liang and J. N. Cao, "Social context-aware middleware: a survey," *Pervasive and Mobile Computing*, vol. 17, pp. 207–219, 2015.

[13] M. Haghighi and D. Cliff, "Sensomax: an agent-based middleware for decentralized dynamic data-gathering in wireless sensor networks," in *Proceedings of the International Conference on Collaboration Technologies and Systems (CTS '13)*, pp. 107–114, San Diego, Calif, USA, May 2013.

[14] F. Wang, L. Hu, J. Zhou, and K. Zhao, "A data processing middleware based on SOA for the internet of things," *Journal of Sensors*, vol. 2015, Article ID 827045, 8 pages, 2015.

[15] S. Gaglio, G. L. Re, G. Martorella, and D. Peri, "A lightweight middleware platform for distributed computing on wireless sensor networks," *Procedia Computer Science*, vol. 32, pp. 908–913, 2014.

[16] Z. Khalid, N. Fisal, and M. Rozaini, "A survey of middleware for sensor and network virtualization," *Sensors*, vol. 14, no. 12, pp. 24046–24097, 2014.

[17] B. Dong, S. Byna, and K. Wu, "SDS: a framework for scientific data services," in *PDSW '13 Proceedings of the 8th Parallel Data Storage Workshop*, D. Hildebrand and K. Schwan, Eds., pp. 27–32, ACM, New York, NY, USA, 2013.

[18] D. Bin, S. Byna, and W. Kesheng, "Spatially clustered join on heterogeneous scientific data sets," in *Proceedings of the IEEE International Conference on Big Data (Big Data '15)*, pp. 371–380, Santa Clara, Calif, USA, 2015.

[19] P. G. Brown, "Overview of sciDB: large scale array storage, processing and analysis," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 963–968, ACM, 2010.

[20] R. J. Green, J. Davies, M. C. Davies, C. J. Roberts, and S. J. B. Tendler, "Surface plasmon resonance for real time in situ analysis of protein adsorption to polymer surfaces," *Biomaterials*, vol. 18, no. 5, pp. 405–413, 1997.

[21] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: the internet of things architecture, possible applications and key challenges," in *Proceedings of the 10th International Conference on Frontiers of Information Technology (Fit '12)*, pp. 257–260, IEEE, Islamabad, Pakistan, December 2012.

[22] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[23] C. Tsai, C. Lai, M. Chiang, and L. T. Yang, "Data mining for internet of things: a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 77–97, 2014.

[24] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: a survey,"

IEEE Communications Surveys and Tutorials, vol. 16, no. 1, pp. 414–454, 2014.

[25] Z. Ming, F. Hong, and M. Yan, "Semantic annotation method of IOT middleware," in *Proceedings of the 4th International Conference on Intelligent Control and Information Processing (ICICIP '13)*, pp. 495–498, IEEE, Beijing, China, June 2013.

[26] S. Q. Hu, H. O. Wang, C. D. She, and J. F. Wang, "A semantic middleware of grain storage internet," in *Computer and Computing Technologies in Agriculture IV, Part 4*, D. L. Li, Y. Liu and, and Y. Y. Chen, Eds., IFIP Advances in Information and Communication Technology, pp. 71–77, Springer, Berlin, Germany, 2011.

[27] J. Granados, A.-M. Rahmani, P. Nikander, P. Liljeberg, and H. Tenhunen, "Towards energy-efficient HealthCare: an Internet-of-Things architecture using intelligent gateways," in *Proc. 4th International Conference on Wireless Mobile Communication and Healthcare, MOBIHEALTH 2014, November 3, 2014–November 5, 2014*, pp. 279–282, Institute of Electrical and Electronics Engineers, 2015.

[28] H. Z. Wang, G. W. Lin, J. Q. Wang, W. L. Gao, Y. F. Chen, and Q. L. Duan, "Management of big data in the internet of things in agriculture based on cloud computing," in *Proc. 3rd International Conference on Manufacturing Engineering and Process, ICMEP 2014, April 10, 2014-April 11, 2014*, pp. 1438–1444, Trans Tech Publications Ltd, 2014.

[29] J. Hwang and H. Yoe, "Study on the context-aware middleware for ubiquitous greenhouses using wireless sensor networks," *Sensors*, vol. 11, no. 5, pp. 4539–4561, 2011.

[30] J. An, X. L. Gui, W. D. Zhang, J. H. Jiang, and J. W. Yang, "Research on social relations cognitive model of mobile nodes in Internet of Things," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 799–810, 2013.

[31] D. Q. Zhang, H. Y. Huang, C. F. Lai, X. D. Liang, Q. Zou, and M. Y. Guo, "Survey on context-awareness in ubiquitous media," *Multimedia Tools and Applications*, vol. 67, no. 1, pp. 179–211, 2013.

[32] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, "Developing an IoT Smart City framework," in *Proceedings of the 4th International Conference on Information, Intelligence, Systems and Applications (IISA '13)*, pp. 1–6, IEEE, Piraeus, Greece, July 2013.

[33] M. Nakamura and L. D. Bousquet, "Constructing execution and life-cycle models for smart city services with self-aware IoT," in *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC '15)*, pp. 289–294, Grenoble, France, July 2015.

[34] S.-P. Tseng, B.-R. Li, J.-L. Pan, and C.-J. Lin, "An application of Internet of things with motion sensing on smart house," in *Proceedings of the IEEE International Conference on Orange Technologies (ICOT '14)*, pp. 65–68, Xian, China, September 2014.

[35] K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for data processing in large-scale interconnected sensor networks," in *Proceedings of the 8th International Conference on Mobile Data Management (MDM '07)*, pp. 198–205, IEEE, Mannheim, Germany, May 2007.

[36] Tinyos, "TinyOS Alliance," July 2010, http://www.tinyos.net/.

[37] GSN Documentation, https://github.com/LSIR/gsn/wiki/Documentation.